apter]quadroloqQuadro

UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

DIEGO BRUNING

APRENDIZADO POR TRANSFERÊNCIA APLICADO NA DETECÇÃO DE PLACAS DE LICENCIAMENTO DE VEÍCULOS

CAXIAS DO SUL

2024

DIEGO BRUNING

APRENDIZADO POR TRANSFERÊNCIA APLICADO NA DETECÇÃO DE PLACAS DE LICENCIAMENTO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Carine Geltrudes Webber

CAXIAS DO SUL

2024

DIEGO BRUNING

APRENDIZADO POR TRANSFERÊNCIA APLICADO NA DETECÇÃO DE PLACAS DE LICENCIAMENTO DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 02/07/2024

BANCA EXAMINADORA

Profa. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

Prof. Dr. Marcos Eduardo Casa
Universidade de Caxias do Sul - UCS

Prof. Dra. Marilda Machado Spindola
Universidade de Caxias do Sul - UCS

RESUMO

A identificação dos veículos tem importância por vários aspectos, como a identificação de posse de propriedade, coibição de roubo, e até mesmo o controle de fluxo. Diversos métodos são empregados para a identificação de veículos, como números gravados em suas estruturas ou peças. Entre esses meios a placa de licenciamento veicular se destaca pela praticidade em diversas funcionalidades, uma vez que costuma estar localizada em pontos facilmente visíveis do veículo, porém, o reconhecimento automático de placas de veículos continua sendo um desafio significativo. Isso pode ser atestado quando se observa a diversidade nas condições de captura de imagem, incluindo variações na iluminação, ângulo de captura e layouts de placas, os quais podem variar entre diferentes regiões e países. Muitos algoritmos foram desenvolvidos para realizar a detecção e reconhecimento de placas de veículos, todos possuem vantagens e falhas em diferentes situações. Nos últimos anos a comunidade científica fez avanços em metodologia e desempenho, estudos recentes tem demonstrado resultados interessantes no reconhecimento e segmentação de placas de licenciamento de veículos, utilizando Redes Neurais Convolucionais (RNCs) em conjunto com técnicas de aprendizado por transferência. Este trabalho aborda a aplicação do aprendizado por transferência para a detecção e segmentação de placas de veículos utilizando redes neurais convolucionais. Foi realizada uma revisão sistemática para compreender o estado da arte na área. Em seguida, foi adaptado e treinado uma arquitetura MobileNet pré-treinada com o conjunto de dados ImageNet. O treinamento e validação foi feito utilizado imagens do dataset UFPR-ALPR, que contém 4500 imagens de veículos nacionais. O modelo final foi avaliado usando diversas métricas de desempenho. Os resultados indicaram que a combinação de técnicas de aprendizado por transferência e ajuste fino melhora significativamente a precisão na detecção de placas. Este estudo contribui para o avanço no reconhecimento automático de placas, destacando a importância do pré-processamento das imagens e do uso de funções de perda especializadas para a otimização do modelo.

Palavras-chave: Aprendizado de Máquina. Aprendizado por Transferência. Placa de Licenciamento. Inteligência Artificial. Redes Neurais.

ABSTRACT

Vehicle identification is important for various reasons, such as property ownership identification, theft prevention, and even traffic flow control. Various methods are used for vehicle identification, such as numbers engraved on their structures or parts. Among these, the vehicle license plate stands out for its practicality in several functionalities, as it is usually located in easily visible points of the vehicle. However, automatic license plate recognition remains a significant challenge. This is evident when considering the diversity in image capture conditions, including variations in lighting, capture angle, and plate layouts, which can vary between different regions and countries. Many algorithms have been developed to detect and recognize vehicle plates, each with advantages and shortcomings in different situations. In recent years, the scientific community has made advances in methodology and performance, with recent studies showing interesting results in the recognition and segmentation of vehicle license plates using Convolutional Neural Networks (CNNs) in combination with transfer learning techniques. This work addresses the application of transfer learning for the detection and segmentation of vehicle plates using convolutional neural networks. A systematic review was conducted to understand the state of the art in the field. Subsequently, a pre-trained MobileNet architecture with the ImageNet dataset was adapted and trained. Training and validation were performed using images from the UFPR-ALPR dataset, which contains 4500 images of national vehicles. The final model was evaluated using various performance metrics. The results indicated that the combination of transfer learning techniques and fine-tuning significantly improves plate detection accuracy. This study contributes to the advancement of automatic license plate recognition, highlighting the importance of image preprocessing and the use of specialized loss functions for model optimization.

Keywords: Machine learning. Transfer learning. Licensing plate. Artificial intelligence. Neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OCR Optical Character Recognition

IA Inteligência Artificial

RNC Rede Neural Convolucional

SIFT Scale-Invariant Feature Transform

ReLU Rectified Linear Unit

PNG Portable Network Graphics

CONTRAN Conselho Nacional de Transitos

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento automático de placas de veículos é um processo que visa a extração de informações de placas a partir de imagens capturadas por câmeras eletrônicas ((DU *et al.*, 2013). Esta tarefa é de grande importância no âmbito do transporte inteligente e da vigilância, podendo ser útil em várias situações, tais como: para a aplicação automática das leis de trânsito, detecção de veículos roubados, identificação de violações de pedágio e controle eficiente do fluxo de tráfego (SILVA; JUNG, 2020).

Nesse sentido a identificação veicular por meio de números de licença tornou-se uma prática global (GOU *et al.*, 2016). No entanto, o reconhecimento automático de placas de veículos continua sendo um desafio significativo, isso pode ser atestado quando se observa a diversidade nas condições de captura de imagem, incluindo variações na iluminação, ângulo de captura e layouts de placas, os quais podem variar entre diferentes regiões e países (SILVA; JUNG, 2020).

Dessa forma áreas como o processamento de imagens e a Inteligência Artificial (IA) têm sido exploradas para buscar soluções neste contexto. Utiliza-se assim, da combinação de várias técnicas, tais como: a detecção de objetos, o processamento de imagens e o reconhecimento de padrões (DU *et al.*, 2013).

O reconhecimento de padrões é a área de estudo científico que tem como escopo a classificação de objetos em um número de categorias ou classes (THEODORIDIS; KOUTROUMBAS, 2006). Esta área de estudo encontra-se no centro de uma série de áreas de aplicação, incluindo a análise de imagens. O processamento digital de imagens é o estudo de teorias e técnicas para manipulação de imagens por computador (BEZDEK *et al.*, 2005). Por outro lado, a detecção de objetos envolve a identificação de instâncias de uma ou várias classes em uma imagem.

A IA tem contribuído na busca de soluções em todas as sub-tarefas associadas ao reconhecimento de placas: na detecção, segmentação e no reconhecimento de caracteres. Em especial, os avanços na área de aprendizado profundo possibilitam que estudos e produtos sejam desenvolvidos. Nesse contexto uma abordagem que tem ganhado destaque é a combinação do poder das arquiteturas de aprendizado profundo com a versatilidade trazida pelos métodos de aprendizado por transferência Yosinski *et al.* (2014)

O aprendizado profundo (do inglês Deep Learning), também chamado de redes neurais profundas é um subconjunto do aprendizado de máquina que utiliza as redes com um grande número de camadas e parâmetros. As camadas das redes neurais são organizadas de uma forma onde as superiores aprendam dados mais complexos com base no aprendizado das inferiores (SHINDE; SHAH, 2018), possibilitando capturar representações complexas e hierárquicas de dados (GOODFELLOW; BENGIO; COURVILLE, 2016). Por isso, o aprendizado profundo é adequado para analisar e extrair conhecimento útil de grandes quantidades de dados, além de informações

coletadas de diferentes fontes (ZHANG; WANG; LIU, 2018).

Aprendizado por transferência é uma estratégia fundamental na instrução de máquinas que buscam aproveitar o conhecimento adquirido em uma tarefa de aprendizado para melhorar o desempenho em uma tarefa relacionada (WEISS; KHOSHGOFTAAR; WANG, 2016). Em vez de começar o processo de aprendizado do zero, a ação envolve a transferência de representações aprendidas geralmente em forma de pesos e arquiteturas de rede neural, de uma tarefa-fonte para uma tarefa-alvo (OLIVAS *et al.*, 2009). Isso é especialmente benéfico quando os dados na tarefa-alvo são limitados, pois a informação compartilhada da tarefa-fonte pode fornecer uma base sólida para o aprendizado da tarefa-alvo. O processo de aprendizado por transferência não apenas acelera o método de treinamento, mas também pode melhorar significativamente a generalização e o desempenho do modelo, tornando-o uma ferramenta essencial para enfrentar desafios complexos em diversas áreas da inteligência artificial e aprendizado automático (OLIVAS *et al.*, 2009).

O treinamento de modelos para reconhecimento de objetos em imagens demanda uma quantidade considerável de dados para treinagem e muitas vezes pode ser difícil reunir um conjunto de dados grande o suficiente para esta tarefa específica (GOODFELLOW; BENGIO; COURVILLE, 2016). Ao aplicar este aprendizado espera-se reutilizar modelos pré-treinados em tarefas relacionadas, como a detecção de objetos ou reconhecimento de caracteres, e adaptá-los para a tarefa de reconhecimento de placas. Busca avaliar se de fato, esse processo também será bem sucedido para acelerar o treinamento e melhorar a capacidade de generalização, considerando que os dados específicos para placas são limitados.

A utilização de aprendizado por transferência oferece diversas vantagens substanciais para o problema de segmentação e reconhecimento de objetos em imagens (OPBROEK *et al.*, 2015). Ao observar trabalhos que envolvem aquisição e processamento de imagens, os métodos de aprendizado por transferência tem revelado resultados promissores. Para estudo de caso e foco deste trabalho que trata do reconhecimento de placas de veículos, têm-se como objetivo avaliar se tais métodos podem ser aplicados e se oferecem as mesmas vantagens substanciais encontradas em outros domínios.

Em suma, a aplicação do aprendizado por transferência ao reconhecimento de placas é uma estratégia que pode se revelar eficaz para superar a ausência de dados, melhorar a precisão e a robustez dos sistemas. Ao aproveitar o conhecimento prévio de tarefas relacionadas, os modelos podem realizar detecção e reconhecimento de placas de maneira mais eficiente e confiável, demonstrando assim a importância da combinação de abordagens para enfrentar os desafios inerentes a essa tarefa específica.

1.1 OBJETIVOS

O objetivo geral do trabalho é avaliar uma implementação baseada em um modelo obtido por intermédio do aprendizado por transferência, para a tarefa de segmentação de placas de veículos em imagens.

Como objetivos específicos têm-se os seguintes:

1. Identificar modelos pré-treinados, aptos a tarefas de aprendizado por transferência para reconhecimento de imagens.
2. Implementar um protótipo de modelo de aprendizado para a tarefa segmentação de placas de veículos em imagens.
3. Realizar testes, comparar e examinar os resultados obtidos.
4. Situar os resultados obtidos frente outros trabalhos prévios na área.

1.2 ESTRUTURA DO TRABALHO

Este documento de conclusão de curso está organizado em 5 capítulos. O Capítulo 2 aborda a revisão da literatura e conceitos importantes para o trabalho. O Capítulo 3 descreve os passos realizados para o desenvolvimento do modelo de segmentação de placas de carros. No Capítulo 4 é demonstrado os resultados encontrados. Por fim, as considerações finais, descritas no Capítulo 5, apresentam uma síntese do desenvolvimento, resultados e dificuldades encontrados e os próximos passos para o desenvolvimento de trabalhos futuros.

2 APRENDIZADO POR TRANSFERÊNCIA PARA O RECONHECIMENTO VEICULAR

A identificação dos veículos tem importância por vários aspectos, como identificação de posse de propriedade, coibição de roubo e até mesmo controle de fluxo (JAWALE *et al.*, 2023; SILVA; JUNG, 2020). Diversos métodos são empregados para a identificação de veículos, como números gravados em suas estruturas ou peças. Entre esses meios a placa de licenciamento veicular se destaca pela praticidade em diversas funcionalidades, uma vez que costuma estar localizada em pontos facilmente visíveis do veículo.

Muitos trabalhos foram desenvolvidos para o reconhecimento de placas de licenciamento, utilizando as técnicas de aprendizado de máquina, aprendizado profundo e mais recentemente, aprendizado por transferência (JAWALE *et al.*, 2023). O aprendizado de máquina é uma área de estudos na Ciência da Computação, que busca o desenvolvimento de algoritmos e modelos que aprendam a partir de experiência (MITCHELL, 1997). A aprendizagem profunda é uma sub área de aprendizado de máquina que utiliza redes neurais de muitas camadas. Já o aprendizado por transferência estuda técnicas dentro do aprendizado de máquina para aproveitar conhecimento de modelos pré-treinados em uma tarefa para uso em outras atribuições (OLIVAS *et al.*, 2009).

Estudos recentes tem demonstrado o sucesso na aplicação do aprendizado por transferência para reconhecimento de placas veiculares em imagens e vídeos. Neste contexto, o trabalho estudou e se aprofundou na etapa de detecção e segmentação de placas de licenciamento veicular, utilizando os métodos de aprendizado por transferência que serão descritos adiante.

2.1 RECONHECIMENTO DE PLACAS DE LICENCIAMENTO VEICULAR

O reconhecimento automático de placas de licenciamento veicular desempenha um papel fundamental em diversos setores, sendo importante para a identificação e monitoramento eficientes (UPADHYAY *et al.*, 2019). Dessa forma, sistemas de diversos segmentos se baseiam na identificação por placas de licenciamento, esses sistemas dependem de ser alimentados com informações corretas das placas (PHAN; HA; DO, 2022).

A diversidade nas condições de captura de imagem acrescentam desafios na tarefa de reconhecimento de placas. As variações na iluminação, ângulo de captura e layouts de placas estão entre as principais causas de dificuldade no momento de realização da tarefa, podendo variar entre diferentes regiões e países (SILVA; JUNG, 2020).

A Figura 1 ilustra algumas situações de captura com variações de luminosidade e cor.

Já a Figura 2, ilustra a variação dos modelos de layout de placas atualmente vigentes no Brasil.

Figura 1 – Versatilidade na cor, tamanho e resolução das imagens de placas

Fonte: Adaptado de Shafi *et al.* (2022)

Figura 2 – Diferentes layouts de placas vigentes no Brasil

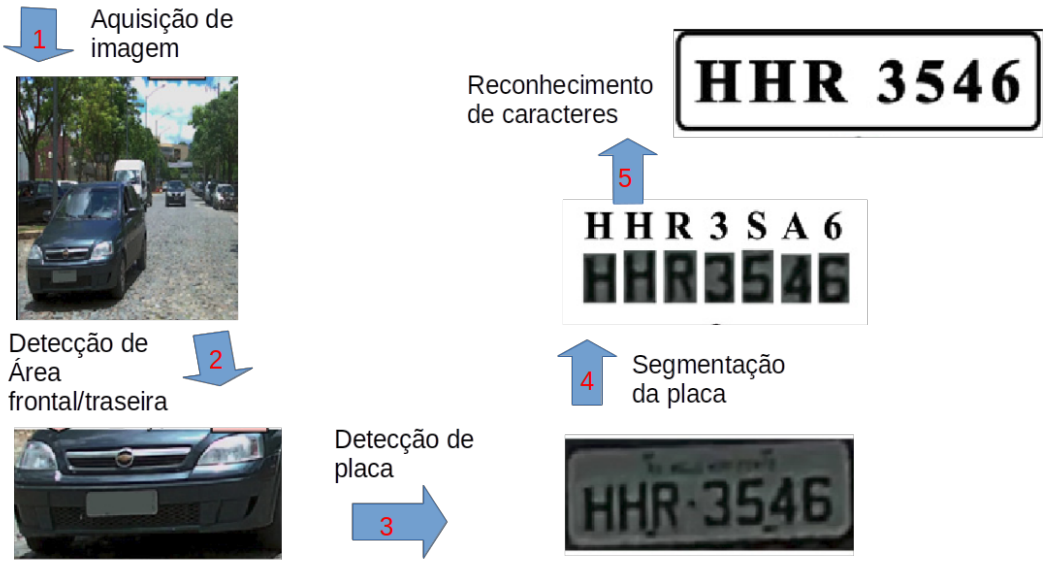
Fonte: Adaptado de Grupo Otimize (2022)

O reconhecimento automático de placas veiculares pode ser dividida em três sub-tarefas principais: a detecção, a segmentação e o reconhecimento de caracteres. Essas etapas constituem o padrão convencional dos sistemas de reconhecimento automático de placas de veículos, onde diversos estudos concentram-se em uma ou duas dessas etapas (SILVA; JUNG, 2020), po-

rém, alguns trabalhos na área podem incluir outros estágios, como a detecção do veículo (em especial a parte frontal e traseira).

A Figura 3 ilustra um possível fluxo de reconhecimento de placa levando em conta a detecção do veiculo. Na primeira etapa da ilustração é feito aquisição da imagem de uma base de dados ou de uma câmera de vídeo. Na segunda etapa é realizado a detecção da área frontal do veículo na imagem adquirida, em seguida a imagem que representa a área frontal do veículo é usada como entrada na terceira etapa, onde é feito a detecção da área da placa. Na quarta etapa os caracteres da placa são segmentados e por fim, é realizado na quinta etapa da ilustração o reconhecimento dos caracteres segmentados.

Figura 3 – Etapas do processo do reconhecimento automático de placas



Fonte: Adaptado de Silva e Jung (2020)

Antes de realizar qualquer uma das sub-tarefas que compõe reconhecimento automático de placas (detecção, segmentação e reconhecimento de caracteres), é necessário uma etapa de pré-processamento das imagens. Nessa etapa geralmente a imagem é processada para um espaço de cores em tons de cinza, também é feito o redimensionamento e a remoção de ruídos (JAWALE *et al.*, 2023).

A tarefa de detecção geralmente procede segmentação de placa de veículo, mas algumas vezes ambas as etapas são realizadas em conjunto Silva e Jung (2020). Na ação de detecção da placa de carro identifica-se se há ou não a presença de uma placa de licenciamento veicular na imagem. Já na tarefa de segmentação, busca-se identificar e isolar a área da imagem que contém a placa do veículo. O isolamento da área da placa permite que as etapas subsequentes do sistema se concentrem exclusivamente na região de interesse.

O layout das placas variam de acordo com a regulamentação de cada região, mas geralmente possuem algumas propriedades comuns: as placas de licenciamento são retangulares e

com largura maior que a altura, exceto para as motos, é formada por letras e números distribuídos horizontalmente (SILVA; JUNG, 2020). Sendo assim, se na etapa de segmentação a ação for realizada corretamente, poderá ser usado um algoritmo Optical Character Recognition (OCR), genérico para reconhecimento dos caracteres Silva e Jung (2020).

2.1.1 REVISÃO SISTEMÁTICA DA LITERATURA

Para identificar os principais trabalhos relacionados ao reconhecimento de placas por meio do aprendizado por transferência, foi adotado o método de revisão sistemática da literatura, uma abordagem que utiliza a literatura existente sobre um tema específico (SAMPAIO; MANCINI, 2007).

O processo seguiu cinco etapas conforme proposto por Sampaio e Mancini (2007), inicialmente foi formulado uma pergunta científica clara, delineando objetivos e restrições. Em seguida, definiu-se as palavras-chave e termos relevantes para a busca de evidências, estabelecendo estratégias e filtros para garantir a inclusão de todos os artigos relevantes.

Ao revisar os resumos dos artigos obtidos na busca, foram selecionados os trabalhos mais pertinentes com base na pergunta e nos critérios estabelecidos. Na etapa seguinte, analisou-se a qualidade metodológica desses estudos, avaliando a correta aplicação das metodologias e identificando possíveis erros nos resultados.

Para realizar essa revisão sistemática foram utilizadas as bases de dados *ScienceDirect* e *IEEE Xplore*, nesse sentido foram aplicados filtros para restringir a busca a artigos nas áreas de Ciência da Computação e Engenharia, publicados entre 2020 e 2023. Os termos-chave empregados foram: *transfer learning*, *license plate*, *recognition* e *detection*.

A primeira busca resultou em cerca de 58 artigos, mas observou-se que a maioria não tratava especificamente do reconhecimento de placas de veículos.

Dessa forma, os resultados foram refinados por meio de um filtro adicional para listar apenas artigos que continham a palavra *plate* no título. Assim, obtiveram-se 4 artigos na *ScienceDirect* e 10 na *IEEE Xplore*. Após uma revisão detalhada desses 14 artigos, 9 pesquisas foram selecionadas, descartando os demais por não abordarem adequadamente o reconhecimento de placas veiculares.

A Tabela 1 resume os artigos analisados listando os *datases* e modelos utilizados por cada autor, bem como os resultados apresentados, assim, verifica-se:

O primeiro estudo analisado apresentado por Onim *et al.* (2022), introduz um sistema inovador de reconhecimento de placas de veículos, utilizando o BLPnet, uma nova arquitetura de rede neural profunda. Este modelo é dividido em três fases distintas, na primeira emprega o NASNet-Mobile (ZOPH *et al.*, 2018), uma rede neural convolucional treinada em mais de um milhão de imagens do banco de dados ImageNet como base para identificar veículos, incorpo-

Tabela 1 – Artigos analisados na revisão sistemática.

Artigo	Dataset	Melhor modelo	Métrica	Resultados
Onim *et al.* (2022)	Stanford AI Lab	NASNet-Mobile + Inception-v3	precisão	95
Silva e Jung (2020)	AOLP, SSIG brasileiro, Stanford Cars Dataset, UFPR-ALPR, OpenALPR	YOLOv2	precisão	SSI: 92.41; UFALPR: 85.42
Phan, Ha e Do (2022)	GreenParking	YoloV4		
Cai, Zhang e Huang (2020)	–	MobileNet	precisão	96,57
Upadhyay *et al.* (2019)	próprio	YOLOv3 + SIFT	precisão	93,9
Elnashar, Hemayed e Fayek (2020)	–	YOLOv3	precisão	96,8
Amaanullah *et al.* (2022)	Kaggle dataset characters + conjunto popro de veículos indonésios	DenseNet121	precisão	96,42
Jain *et al.* (2021)	–	YOLOv4 + EasyOCR	–	–
Maduri *et al.* (2021)	conjuto próprio de carros de bangladeshi	YOLOv5+ EasyOCR	prcisão	95

rando camadas ocultas para aprimorar a detecção da caixa delimitadora do veículo. A segunda fase estende o modelo Inception-v3 (SZEGEDY *et al.*, 2015b) para identificar a região da placa, já na terceira fase, utiliza-se um novo mecanismo OCR chamado Bengali-OCR para o reconhecimento dos caracteres bengalis. O treinamento do modelo envolveu a aplicação de transformações aleatórias em um conjunto de dados de carros do *Stanford AI Lab*, o modelo proposto apresentou desempenho sólido e baixo custo computacional.

No estudo conduzido por Silva e Jung (2020), a abordagem proposta concentra-se na detecção e reconhecimento de placas de veículos em tempo real. Usando uma rede FAST-YOLO v2 (REDMON; FARHADI, 2017) em cascata, os autores segmentam as imagens em classes de vista frontal/traseira do carro e placa de licenciamento. Posteriormente, empregam uma rede YOLO (REDMON *et al.*, 2016) adaptada para reconhecer 35 classes para a identificação dos caracteres nas placas. Avaliaram a solução em diversos conjuntos de dados, incluindo AOLP, SSIG brasileiro, Stanford Cars Dataset, UFPR-ALPR (Laroca et al., 2018) e OpenALPR, comparando com trabalhos anteriores e aplicações comerciais. A solução obteve ótimos resultados, especialmente em placas brasileiras superando outras abordagens e competindo favoravelmente em placas europeias com a aplicação comercial.

No que se refere ao problema de reconhecimento de placas em imagens mal capturadas para sistemas de controle de veículos, Phan, Ha e Do (2022) propõem o uso de Rede Neural Convolucional (RNC) com o uso do YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020), redes Adversárias Gerativas com o algoritmo Pix2Pix (ISOLA *et al.*, 2018) para detecção e reconstrução

da placa. A RNC foi treinada com 1200 épocas e o Pix2Pix (ISOLA *et al.*, 2018) com 250, foram usadas duas métricas principais: o cálculo da perda e a distância Manhattan entre a imagem real e a imagem gerada. O conjunto de dados usado para treinamento e teste foi o GreenParking (THAIPHUNG, 2023), consistindo em 800 imagens que foram rotuladas e divididas em uma proporção de 9 para 1. No entanto, os resultados obtidos foram limitados e os autores indicam que seria necessário mais dados para melhorar o desempenho do modelo.

O trabalho de Cai, Zhang e Huang (2020) aborda o problema de reconhecimento do número e do estado nas placas veiculares dos 50 estados americanos com suas variações. A solução proposta envolveu a integração das redes neurais YOLOv3, Convolutional Recurrent Neural Network (CRNN) [1] e ResNet, com o aprendizado por referência. Para a detecção, foi utilizado o YOLOv3 customizado com MobileNet (HOWARD *et al.*, 2017) como módulo de extração de recursos integrando-se ao módulo de atenção de bloco convolucional (CBAM). O CRNN foi usado como modelo de reconhecimento de caracteres de número de placa, e o ResNet para classificação do estado. Foram usadas métricas como a interseção entre a caixa delimitadora gerada, a rotulada e a probabilidade de classificação para avaliar o desempenho do modelo. O conjunto de dados usado para treinamento e teste foi dividido em 4530 para o treinamento, 300 para a verificação e 300 para os testes, totalizando 5130 imagens. Os resultados obtidos foram bastante satisfatórios, com uma taxa de acerto superior a 96,57% na detecção das placas e 91,41% na precisão do reconhecimento dos números das placas.

No trabalho de Upadhyay *et al.* (2019), a solução proposta envolveu a detecção do veículo com YOLOv3, a detecção da placa usando a técnica Scale-Invariant Feature Transform (SIFT) (LOWE, 2004) em conjunto com uma CNN e a identificação dos caracteres com OCR. Foram utilizados os algoritmos *YOLOv3*, *K-means* e SIFT, as métricas usadas para avaliação foram precisão, *recall* e FPS. Dois conjuntos de dados foram utilizados, um próprio e outro não especificado, o resultado obtido foi uma precisão de 0,939, *recall* de 0,913 e FPS de 100.

Elnashar, Hemayed e Fayek (2020) fazem a detecção de placas de carros egípcios e a classificação em padrão e não padrão com uma CNN YOLOv3, o reconhecimento dos caracteres das placas não foram realizados. O resultado obtido foi uma precisão de detecção de 99,2% para placas de estilo padrão e 96,8% para placas não padronizadas.

Em outra análise, Amaanullah *et al.* (2022) comparou os modelos DenseNet121, MobileNetV2 e NASNetMobile, que foram treinados com conjuntos de caracteres para o reconhecimento de placas de veículos. A métrica utilizada para avaliação foi a precisão e o conjunto de dados, consistiu-se no Kaggle dataset characters e um conjunto próprio de placas de veículos indonésios. O resultado mostrou que o modelo DenseNet121 produziu a melhor precisão média de reconhecimento (96,42%), em comparação com os modelos MobileNetV2 e NASNetMobile,

[1] Um CRNN é um tipo de rede neural que combina camadas CNN e RNN. As camadas CNN são usadas para extrair uma sequência de vetores de recursos dos dados de entrada, e as camadas RNN são usadas para processar essas sequências em ordem

que alcançaram precisões médias de 92,97% e 95,47%, respectivamente.

Em outro trabalho relevante, Jain *et al.* (2021) propõem um sistema de gerenciamento de tráfego que abrange o reconhecimento de placas. Para isso, utilizam a arquitetura YOLOv4 e Easy-OCR para detecção e reconhecimento de placas de veículos, mas por se tratar de um trabalho em andamento, não deixam claros os resultados obtidos.

Muitos sistemas se baseiam em linguagens que utilizam o alfabeto romano e acabam tendo um desempenho menor no reconhecimento de placas que utilizam outros alfabetos, por isso Maduri *et al.* (2021) propôs uma solução para reconhecimento de placas que utilizam o alfabeto bengali. Essa solução consistiu no uso do algoritmo YOLOv5 para detecção de placas e do EasyOCR (que utiliza os algoritmos CRAFT, ResNet, LSTM e CTC) para reconhecimento dos caracteres. Os autores utilizam as métricas de precisão e *recall* para avaliação do modelo, o conjunto de dados utilizado foi composto por 345 imagens de carros Bangladeshi com matrículas visíveis, além de um conjunto de imagens próprias que foram aumentadas aplicando técnicas de transformação. O resultado obtido foi de 98% de precisão e 95% de recall no reconhecimento de placas.

2.2 APRENDIZADO DE MÁQUINA

O aprendizado de máquina é um campo da Inteligência Artificial, que preocupa-se com a questão de como construir programas de computador que melhorem automaticamente com a experiência (dados ou reforço), sem serem explicitamente programados (DEY, 2016). Dessa forma, foram desenvolvidas muitas aplicações de aprendizagem automática bem-sucedidas, essas aplicações tem utilidade desde a detecção de fraudes em cartões, até veículos autônomos que aprendem a conduzir com pouca interferência humana (MITCHELL, 1997).

As aplicações de aprendizado de máquina estão presentes em diversas áreas, como visão computacional, processamento de linguagem natural, reconhecimento de padrões, previsões/análises financeiras e diagnóstico médico (MITCHELL, 1997).

A Tabela 2 apresenta alguns exemplos de aplicações de aprendizado de máquina bem sucedidas, segundo Mitchell (1997).

2.2.1 O que é aprendizado?

É importante para o entendimento de aprendizado de máquina se definir com clareza o que é aprendizado. Para os propósitos deste trabalho, será utilizada a definição de aprendizado dada por Mitchell (1997), no qual afirma que um programa de computador aprende com a experiência E no que diz respeito a alguma classe de tarefas T e medida de desempenho P, se o seu desempenho nas tarefas em T, conforme medido por P, melhora com a experiência E.

Nesse sentido, para estabelecer um problema de aprendizagem bem definido devemos

Tabela 2 – Algumas aplicações bem sucedidas de aprendizagem de máquina.

Aplicação	Exemplo
Aprendendo a reconhecer palavras faladas	O sistema SPHINX (LEE, 1989) aprende estratégias específicas do falante para reconhecer os sons primitivos (fonemas) e palavras a partir do sinal de fala observado. Métodos de aprendizagem de redes neurais (WAIBEL *et al.*, 1989) e métodos para aprender modelos ocultos de Markov (LEE, 1989) são eficazes para personalizar automaticamente falantes individuais, vocabulários, características de microfone, ruído de fundo, etc. Técnicas semelhantes têm aplicações potenciais em muitos problemas de interpretação de sinais.
Aprendendo a dirigir um veículo autônomo	O sistema ALVINN (POMERLEAU, 1989) utilizou as suas estratégias aprendidas para conduzir sem assistência a 70 milhas por hora durante 90 milhas em estradas públicas, entre outros carros. Técnicas semelhantes têm aplicações possíveis em muitos problemas de controle baseados em sensores
Aprender a classificar novas estruturas astronômicas	Algoritmos de aprendizagem de árvore de decisão foram usados pela *NASA* para aprender como classificar objetos celestes do segundo *Palomar Observatory Sky Survey* (FAYYAD *et al.*, 1995). Este sistema agora é usado para classificar automaticamente todos os objetos no *Sky Survey*, que consiste em três terabytes de dados de imagem.
Aprendendo a jogar *world-class backgammon*	O melhor programa de computador do mundo para backgammon, TD-GAMMON (TESAURO, 1992), aprendeu sua estratégia jogando mais de um milhão de jogos de treino contra si mesmo. Agora joga em um nível competitivo com o campeão mundial humano. Técnicas semelhantes têm aplicações em muitos problemas práticos onde espaços de busca muito grandes devem ser examinados de forma eficiente.

Fonte: Mitchell (1997)

identificar três características essenciais: a classe de tarefas, a medida de desempenho a ser aprimorada e a fonte da experiência (MITCHELL, 1997).

Aplicando isso ao problema específico de detectar placas de carro em uma imagem, a um problema com a tarefa de detectar a placa do carro, a medida de desempenho sendo a porcentagem de falsos positivos e falsos negativos, e a experiência de treinamento consistindo na análise de mais imagens (MITCHELL, 1997).

2.2.2 Técnicas de aprendizado de máquina

O aprendizado de máquina geralmente é dividido em três categorias principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço (FACELI *et al.*, 2011). No aprendizado supervisionado, os algoritmos treinam os modelos em um conjunto de dados que inclui entradas e saídas rotuladas correspondentes, assim esses modelos podem prever saídas para novas entradas de dados (FACELI *et al.*, 2011). Algumas classes de problemas tipicamente resolvidas por meio da aprendizagem supervisionadas são: a classificação de dados e a regressão de dados (FACELI *et al.*, 2011).

Na classificação de dados o programa de computador é incumbido de determinar a categoria qual uma determinada entrada pertence. Algoritmos de aprendizado de máquina são comumente utilizados para criar funções que mapeiam conjuntos de valores para categorias no domínio de saída, pode se também empregar variações desse problema, como a geração de funções que produzem distribuições de probabilidade sobre as classes. (ESCOVEDO, 2020)

Por outro lado, na regressão de dados o programa é desafiado a prever um valor numérico com base em uma entrada específica, essa tarefa é semelhante à classificação, exceto pelo formato da saída, que neste caso é um valor numérico. (ESCOVEDO, 2020)

No aprendizado não supervisionado, os modelos buscam identificar padrões e estruturas nos dados sem a necessidade de saídas rotuladas. Uma classe de problemas resolvidas por meio da aprendizagem não supervisionada é o agrupamento de dados. Nesse caso, o problema consiste em dividir um conjunto de entradas em grupos agrupados por similaridade. Ao contrário da classificação os grupos não são conhecidos de antemão, tornando esta uma tarefa normalmente não supervisionada. (ESCOVEDO, 2020) Já no aprendizado por reforço, os modelos aprendem a selecionar ações em um ambiente para maximizar recompensas (FACELI *et al.*, 2011). Os modelos de aprendizado por reforço são utilizados quando não se dispõe de dados para a tarefa, eles aprendem por meio de respostas do ambiente e suas ações. As respostas podem ser de recompensas ou punições, reforçando ou enfraquecendo desta forma comportamentos.

2.2.3 Etapas de um processo de aprendizado de máquina

O processo de aprendizado de máquina envolve normalmente a escolha de um modelo, o treinamento com dados e por fim, a avaliação de seu desempenho usando conjuntos de teste. A escolha do modelo deve levar em conta os dados e tipo de tarefa. Durante o treinamento é realizado o ajuste fino, que é o ajuste dos parâmetros do modelo (FACELI *et al.*, 2011);

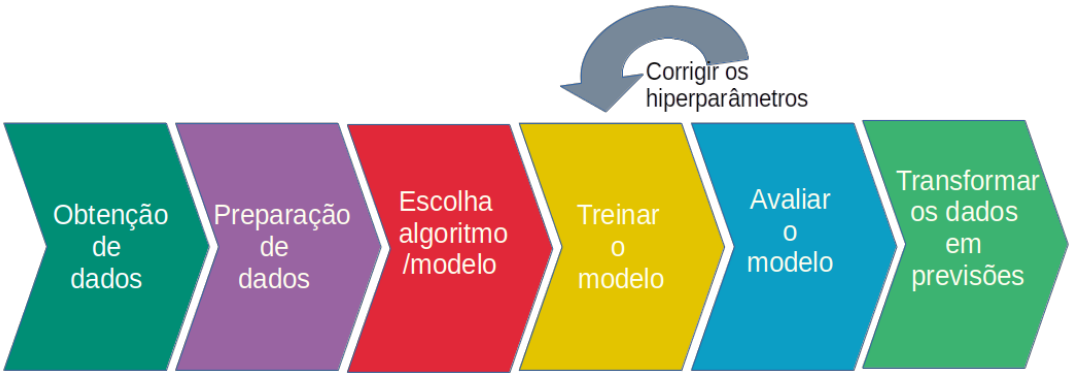
Existem algumas etapas básicas usadas para executar uma tarefa de aprendizado de máquina, conforme ilustrado na Figura 4. A primeira etapa trata da obtenção de dados, eles podem ser coletados de diversas fontes, como: arquivos, bancos de dados, etc. (HAPKE; NELSON, 2020).

A segunda etapa trata da preparação de dados, a ação é realizada para limpar os dados brutos e após serem coletados do mundo real, são transformados em um conjunto de dados limpos. Os dados brutos podem conter valores ausentes, valores inconsistentes, instâncias duplicadas, etc. Portanto, os dados brutos não podem ser usados diretamente para construir um modelo (GATEVIDYALAY, 2023).

Na terceira fase é feita a escolha do algoritmo ou modelo de aprendizagem, o algoritmo com melhor desempenho precisa ser pesquisado conforme o tipo de problema a ser resolvido e do tipo de dados disponíveis (FAYYAD; AL., 1996).

A quarta fase trata de treinar o modelo escolhido, os dados são divididos em conjuntos

Figura 4 – Etapas do processo de aprendizado de máquina



Fonte: Adaptado de (GATEVIDYALAY, 2023)

para treinamento e conjuntos para teste, geralmente em uma proporção 80/20 ou 70/30. É comum os dados de treinamentos serem subdivididos criando uma nova partição para validação (GATEVIDYALAY, 2023). Após a separação, o conjunto de dados de treinamento é usado pelo algoritmo na construção do modelo, durante a ação o modelo é validado sobre os dados de validação e os hiper-parâmetros são ajustados. O conjunto de dados de teste é reservado para testar o modelo (FAYYAD; AL., 1996).

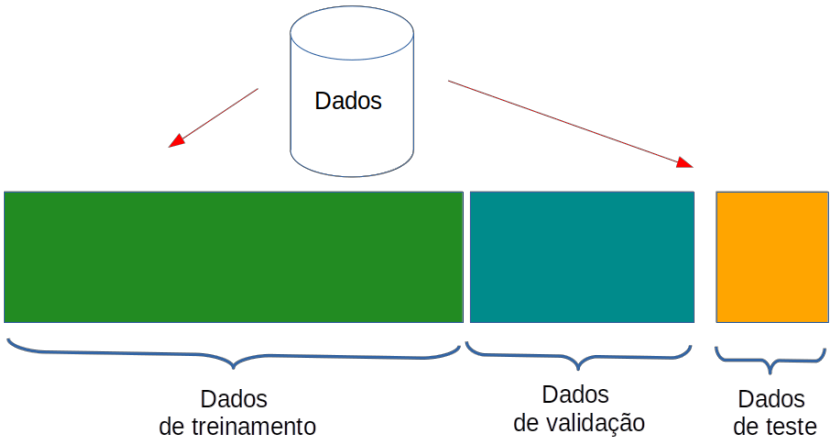
Sendo assim, a estratégia geralmente usada para garantir a generalização do modelo, ilustrada na Figura 5, consistem em dividir os dados em três grupos (AMIDI; AMIDI, 2023): dados do treinamento, dados de validação e dados do teste.

Dessa forma, conforme os autores, os dados de treinamento constituem o conjunto utilizado para o aprendizado do modelo de máquina, ou seja, para a adaptação dos parâmetros do modelo. Já os dados de validação são empregados para realizar uma avaliação imparcial de um modelo ajustado com base no conjunto de dados de treinamento, particularmente durante o ajuste dos hiper-parâmetros. Além disso, desempenham um papel relevante em outras etapas do preparo do modelo. Por fim, os dados de teste são utilizados para oferecer uma avaliação imparcial de um modelo final, que foi ajustado com base no conjunto de dados de treinamento.

Depois do treinamento, a quinta fase começa, na qual o modelo é validado. Para a avaliação do modelo é utilizado o conjunto de dados de teste. Esse conjunto foi separado anteriormente e não foi utilizado no treinamento ou para avaliação dos hiper-parâmetros. O uso de dados não utilizados no treinamento possibilita avaliar se o modelo não se adaptou demais aos dados de treinamento, perdendo a capacidade de generalização, isso é chamado de *overtraining* (CHOPRA, 2021).

Medições como acurácia, precisão e recall são aplicadas para avaliar o desempenho durante a resolução de problemas de aprendizado supervisionado. A precisão é calculada como a proporção de previsões corretas em relação ao total de casos examinados. É determinada pela divisão do número de previsões corretas (verdadeiros positivos + verdadeiros negativos) pelo

Figura 5 – Esquema de separação dos dados em conjuntos de treinamento, validação e teste



Fonte: Adaptado de Amidi e Amidi (2023)

número total de amostras. Já a precisão é a quantidade de verdadeiros positivos dividida pelo total de previsões positivas. Esta medida mostra a proporção de previsões positivas que são verdadeiramente positivas. Pelo contrário, recall é a razão entre os verdadeiros positivos e o total de instâncias positivas reais (verdadeiros positivos + falsos negativos). Essa medida avalia a habilidade do modelo em reconhecer todas as instâncias positivas de forma precisa.

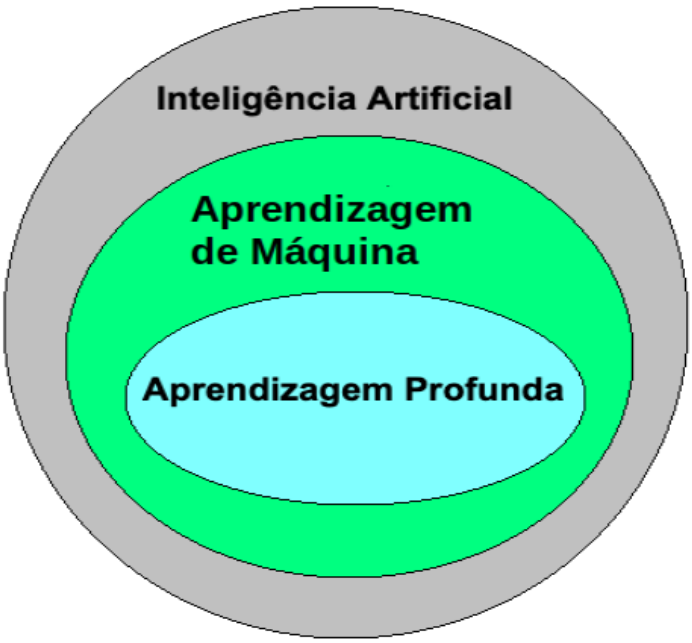
Se o modelo não funcionar bem, ele pode ser reconstruído usando diferentes hiperparâmetros (HAPKE; NELSON, 2020). Após a avaliação do modelo e constatação da sua adequação ao processo envolvido, ele irá integrar um sistema preditivo. Nesta última fase o sistema preditivo dispõe do modelo em situações reais de uso e novas avaliações por meio de monitoramento devem ser feitas. Todo modelo tem uma vida útil, devendo ser monitorado e acompanhado a fim de que se verifique em que momento ele pode estar se tornando defasado, necessitando de ajustes e treinamentos por meio de novos dados (GATEVIDYALAY, 2023).

2.3 APRENDIZAGEM PROFUNDA

Aprendizagem profunda é uma área de estudo dentro do aprendizado de máquina que utiliza redes neurais profundas (KEIJSERS, 2010). Tais redes são formadas por muitas camadas. A relação entre as áreas da aprendizagem profunda, da aprendizagem de máquina e da inteligência artificial é ilustrada na Figura 6. Observa-se na imagem um diagrama circular em que a área da aprendizagem profunda está contida na aprendizagem de máquina, que por sua vez, está contida na área de inteligência artificial.

Várias pesquisas foram realizadas para desenvolvimento de técnicas e modelos para simular o funcionamento do cérebro. As redes neurais artificiais são modelos matemáticos fruto dessas pesquisas. Elas utilizam algoritmos de aprendizado inspirados no cérebro para armazenar informações (DATA SCIENCE ACADEMY, 2023).

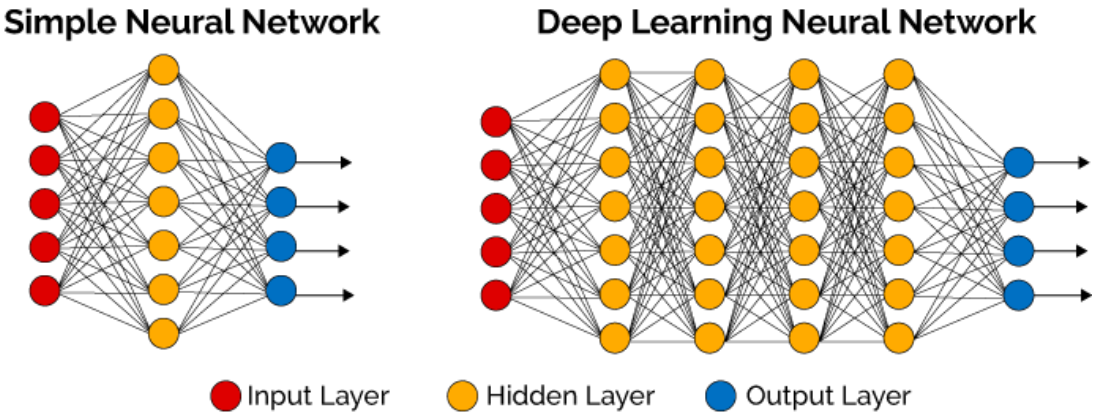
Figura 6 – Aprendizado profundo na inteligência artificial



Fonte: Adaptado de DICIOÁRIO TEC (2022)

As redes neurais são constituídas por muitos neurônios que possuem conexões entre eles formando, assim, camadas (DATA SCIENCE ACADEMY, 2023). As camadas são geralmente classificadas como: camada de entrada, a qual é a primeira camada em uma rede; camada de saída, a qual é a última camada da rede; camadas ocultas, que são todas as camadas entre a entrada e a saída. Cada camada pode ser considerada como um algoritmo simples e uniforme contendo um tipo de função de ativação (DATA SCIENCE ACADEMY, 2023). A Figura 7 apresenta as camadas de uma rede neural e permite um comparativo entre uma rede neural simples e uma rede neural profunda, no quesito número de camadas.

Figura 7 – Rede Neural Simples e Rede Neural Profunda



Fonte: DATA SCIENCE ACADEMY (2023)

É possível observar que a rede neural situada à esquerda é mais simples, sendo capaz de resolver apenas problemas simples. Já a rede da direita, por conter mais neurônios e uma

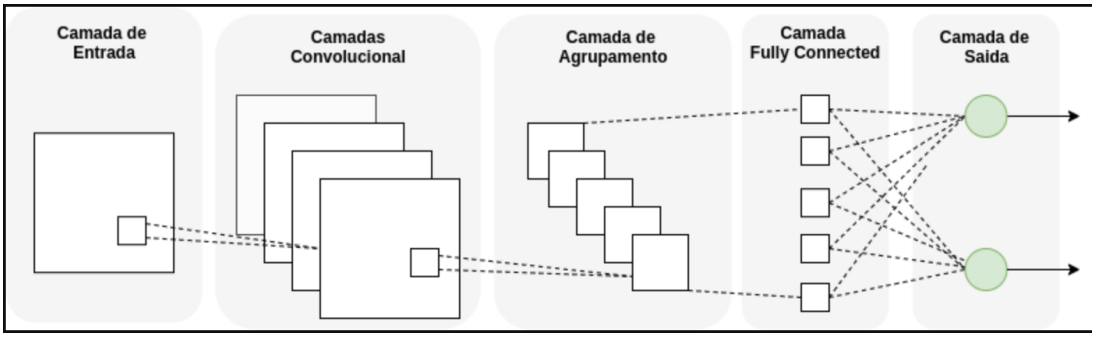
estrutura de conexões maior, está apta a resolver problemas com dados contendo associações mais complexas.

2.3.1 Rede Neural Convolucional

As redes neurais convolucionais são um tipo especial de redes artificiais com mais camadas que usa uma operação matemática chamada convolução em alguma de suas camadas (GOODFELLOW; BENGIO; COURVILLE, 2016). A convolução é um tipo especializado de operação linear (GOODFELLOW; BENGIO; COURVILLE, 2016). Nas redes convolucionais, cada camada é responsável por extrair determinadas informações dos dados de entrada.

A Figura 8 mostra de forma simplificada a estrutura de uma rede neural convolucional, no qual podemos ver a camada de entrada, a camada convolucional, a camada de agrupamento (*Poling*), a camada totalmente conectada (*Fully Conected*) ou camada densa e por fim a acamada de saída.

Figura 8 – Estrutura simplificada de uma rede neural convolucional



Fonte: Fonte Severiano *et al.* (2021)

Cada camada da rede convolucional é responsável por uma sub-tarefa específica. As principais camadas que compõem essas redes são: camada de convolução, camada de agrupamento (*pooling*) e camada densa ou totalmente conectada (f*fully connected*) (KARPATHY; JOHNSON; LI, 2023).

A camada de convolução utiliza pequenos blocos chamados filtros ou *kernels* para extrair e mapear o conteúdo da imagem, transformando em dados. A camada de agrupamento (*pooling*), por sua vez, recebe os blocos contendo as informações extraídas durante a convolução. Essa camada simplifica as informações, resumindo os dados do sub-bloco da imagem em um único valor e resulta na redução da dimensionalidade da imagem. Esta camada densa ou totalmente conectada (*fully connected*) inicia o processo de classificação das informações extraídas pelas camadas anteriores (KARPATHY; JOHNSON; LI, 2023).

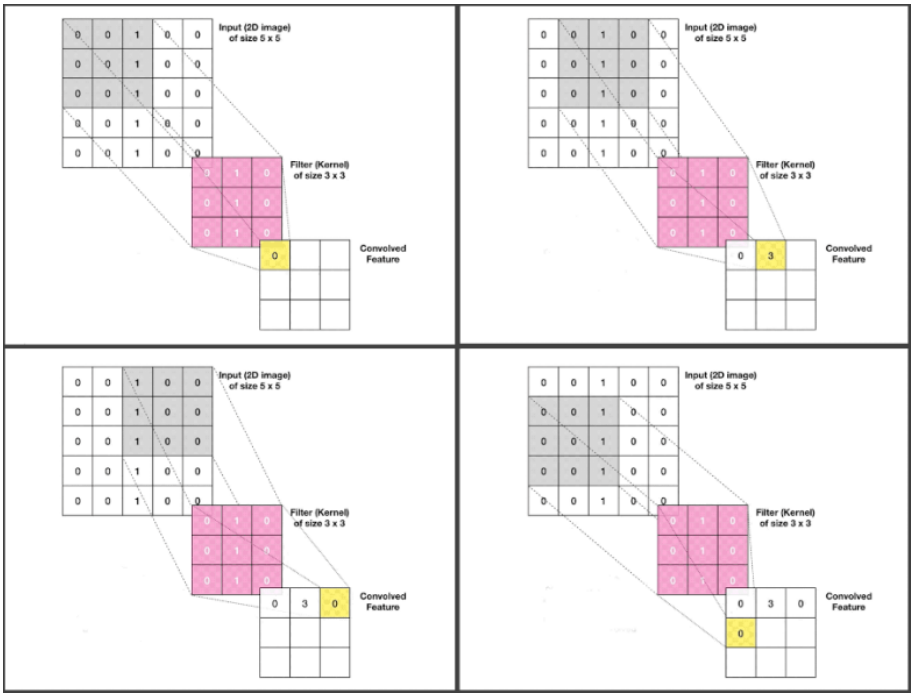
Aprendizagem profunda por meio de redes neurais convolucionais se dá pela repetição de camadas convolucional e de agrupamento.Cada sequência dessas camadas resulta em novas características que são extraídas da imagem. Por fim, a camada densa (*fully connected*) faz

a classificação utilizando as características extraídas, a classificação é atribuída à camada de saída.

Em uma operação convolucional existem duas entradas: a primeira é a imagem de entrada, cujas características são definidas por sua altura, largura e o número de camadas. Por exemplo, em uma representação de uma imagem RGB quadrada, a notação 10x10x3 indica a presença de três canais de cores. A segunda entrada consiste em um conjunto de filtros ou *kernels* denotado por K, esses filtros possuem dimensões n x m x C, onde C representa o número de canais de cores. Em contextos de imagens coloridas geralmente representadas por três camadas de cores, as dimensões n e m se referem ao tamanho do filtro (GOODFELLOW; BENGIO; COURVILLE, 2016).

A Figura 9 ilustra o processo de convolução, onde um filtro com tamanho 3x3 é aplicado sobre um volume de entrada 7x7x1. Nesse exemplo é utilizado uma passada de uma unidade. Por fim é gerado um volume de saída com tamanho 3x3x1.

Figura 9 – Operação de convolução



Fonte: Adaptado de Gandhi (2022)

Para uma imagem de três canais, três matrizes formam a entrada da operação de convolução com os filtros. Cada filtro desliza sobre as matrizes multiplicando a parte sobreposta da matriz de entrada. Após a realização do produto do filtro pela região sobreposta, é feito a média ponderada dos valores da matriz resultante, esse valor então compõe a matriz resultante da operação (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dessa forma, três hiper-parâmetros controlam o tamanho do volume de saída: a profundidade, que representa o número de filtros aplicados, cada um buscando extrair características

específicas da imagem e gerando uma nova matriz para cada canal. A passada, por sua vez, determina como os filtros se movem sobre a matriz, onde uma passada de um implica um deslocamento píxel a píxel, e uma passada de dois implica um deslocamento de dois píxeis por vez. Além disso, o preenchimento-zero é um hiper-parâmetro que envolve a conveniência de preencher o volume de entrada com zeros ao redor da borda, sendo o tamanho desse preenchimento determinado por tal parâmetro (KARPATHY; JOHNSON; LI, 2023).

A aplicação dos filtros no volume de dados podem ser feitas utilizando diferentes tipos de operações convolucionais, tais como: convolução discreta, convolução transposta, convolução dilatada e convoluções separáveis. A seguir será detalhado cada uma dessas operações.

Convolução Discreta: Na convolução discreta, a cada passo o produto do filtro e da área que ele esta sobrepondo é calculado para gerar um elemento da saída, mantendo a noção de ordenação e gerando uma diminuição de dimensionalidade (GANDHI, 2022).

A convolução transposta tem um objetivo contrário a convolução discreta, gerando uma saída com dimensionalidade maior que a dimensão de entrada (ROSEBROCK, 2018). Essa operação é utilizada para aumentar a resolução da imagem, para que isso seja possível e de forma interativa, cada elemento da entrada é multiplicado pelo filtro, o resultado é colocado em uma matriz intermediária com dimensionalidade maior aplicando-se o passo e assim, essas matrizes intermediárias são somadas gerando a saída (ROSEBROCK, 2018).

A operação de convolução transposta é demonstrada na Figura 10. Neste exemplo, um filtro de dimensão 2x2 é utilizado em uma entrada 2x2 com um passo de uma unidade, isso resulta em 4 sobreposições, gerando 4 matrizes de dimensão 3x3. Por fim, essas matrizes são somadas, resultando em uma saída de dimensão 3x3.

Na convolução dilatada o filtro é inflado e adicionado aos espaços entre os seus elementos, o campo receptivo das unidades de saída é então aumentado com pouco custo, sem aumentar o filtro. A taxa de dilatação é dada por um hiper-parâmetro adicional (VESTONI, 2023).

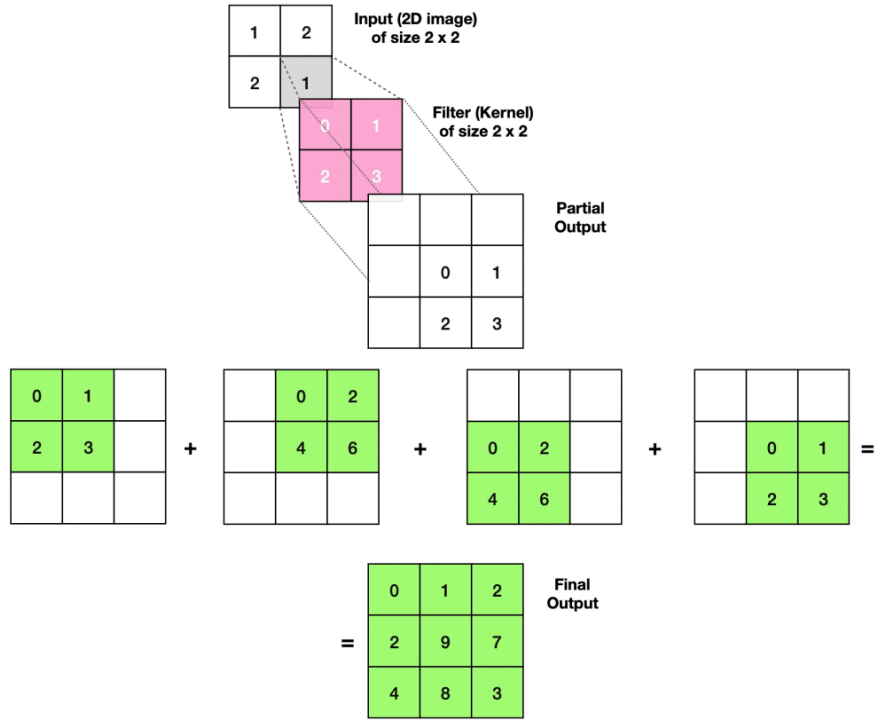
Convolução Separável, é uma técnica onde a convolução por um filtro é decomposta por duas convoluções com dimensionalidade menor, diminuindo o número de parâmetros (VESTONI, 2023). Por exemplo, uma convolução com filtro 3x3 poderia ser substituída por uma com um filtro 3x1, seguido de uma convolução por um filtro 1x3, assim o número de parâmetros diminui de 9 para 6.

2.3.2 Função de ativação

As saídas das operações de convolução precisam passar por uma função de ativação não linear. Algumas funções que já foram utilizadas são a sigmoide e tangente hiperbólica, mas atualmente a função mais comum é a Rectified Linear Unit (ReLU) (BEZERRA, 2016),

A função sigmoide é uma função crescente que assume valores entre 0 e 1 mantendo

Figura 10 – Convolução transposta



Fonte: Adaptado de (ROSEBROCK, 2018)

um balanço ente o comportamento linear e não linear (DATA SCIENCE ACADEMY, 2023). A sua equação segue o seguinte formato: $f(x) = \frac{1}{1+e^{-}x}$

Outra função, a tangente hiperbólica, assume valores positivos e negativos, mas preserva da forma de uma sigmoide. Por outro lado, a função ReLU converte todos os números negativos em zero e preserva os números positivos (DATA SCIENCE ACADEMY, 2023). Sua equação segue o formato: $f(x) = max(0, x)$

2.3.3 Função de perda

Uma função de perda é uma medida da precisão com que o modelo de aprendizado de máquina pode prever o resultado esperado (GEVORGYAN, 2022). Ao projetar uma rede neural completa, a escolha ou desenvolvimento de uma função de perda adequada é um desafio significativo. Em tarefas de aprendizagem profunda, a função de perda avalia geralmente a precisão, similaridade ou qualidade de ajuste entre os valores previstos e os valores reais (TIAN *et al.*, 2022).

A eficácia na detecção de objetos em visão computacional depende fortemente da definição da função de perda. As funções de perda convencionais para detecção de objetos geralmente se baseiam na agregação de métricas de regressão de caixas delimitadoras, como distância, área de sobreposição e proporção de aspecto entre as caixas previstas e as caixas de verdade (GEVORGYAN, 2022). Algumas funções de perda comuns utilizadas para esse propósito incluem:

- IoU: A função de perda *Intersection over Union (IoU)* (YU *et al.*, 2016) é amplamente utilizada para detecção de objetos. Essa função busca otimizar diretamente a pontuação IoU entre as caixas verdadeiras e as caixas previstas. Para calcular o IoU primeiro se obtém a área de intersecção entre a caixa prevista e a caixa verdadeira, em segundo se determina área da união das duas caixas e por fim a área da intersecção é dividido pela área da união.

- GIoU: A função de perda *Generalized IoU (GIoU)* (REZATOFIGHI *et al.*, 2019) também visa otimizar a pontuação IoU entre as caixas verdadeiras e as caixas previstas. A perda GIoU adiciona um termo de penalidade à perda IoU, que considera a área da menor caixa que abrange ambas as caixas utilizadas no cálculo do IoU.

- CIoU: A função de perda *Complete IoU (CIoU)* (ZHENG *et al.*, 2020) é uma extensão da perda GIoU, que incorpora a proporção e a distância central das caixas para penalizar a métrica.

2.4 APRENDIZADO POR TRANSFERÊNCIA

Aprendizado por transferência do inglês *Transfer Learning* é uma técnica de aprendizado de máquina onde o modelo treinado é reaproveitado em uma segunda tarefa relacionada (YOSINSKI *et al.*, 2014). O aprendizado por referência busca a melhoria em uma nova tarefa por meio da transferência de conhecimento, do que já foi aprendido em uma tarefa relacionada. Embora a maioria dos algoritmos de aprendizado de máquina sejam projetados para resolver tarefas únicas, o desenvolvimento de algoritmos que facilitam a aprendizagem por transferência é um tópico de interesse contínuo na comunidade de aprendizado de máquina Olivas *et al.* (2009)

A aprendizagem por transferência é popular na área de estudo para aprendizado profundo, apesar de não ser uma exclusividade. Alguns motivos dessa popularidade são os recursos necessários para treinar modelos de aprendizagem profunda, além dos grandes e desafiadores conjuntos de dados nos quais os modelos são treinados. Segundo Yosinski *et al.* (2014), a aprendizagem por transferência só funciona na profunda se os recursos do modelo aprendidos na primeira tarefa forem gerais, ou seja, adequados para tarefas base e tarefas alvo, em vez de específicos para a tarefa base.

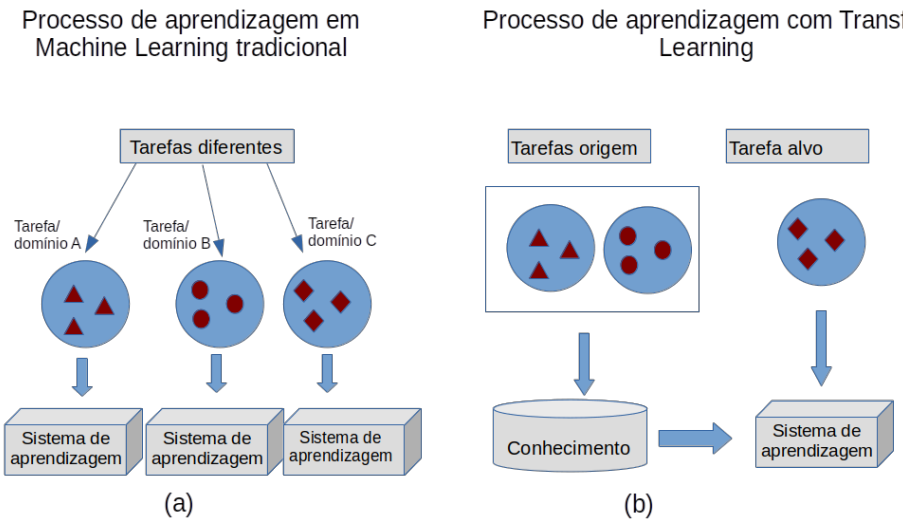
No contexto tradicional do aprendizado máquina, quando se deseja treinar um modelo para realizar uma determinada tarefa em domínio A, parte-se do pressuposto de que possui dados rotulados relacionados à mesma tarefa e ao mesmo domínio. Para treinar também um modelo para um outra tarefa de domínio B, exige-se dados rotulados no domínio B, essa situação é ilustrado na parte (a) da Figura 11.

Algoritmos comuns de aprendizado de máquina tratam tradicionalmente de tarefas isoladas. O aprendizado por transferência tenta mudar isso desenvolvendo métodos para transferir

o conhecimento aprendido em um ou mais tarefas de origem, afim de usá-lo para melhorar o aprendizado em uma área relacionada a tarefa alvo, conforme ilustrado na parte (b) da Figura 11.

A Figura 11 mostra a diferença entre os processos de aprendizagem das técnicas tradicionais e por transferência. Percebe-se na figura que as técnicas tradicionais de aprendizado de máquina tentam aprender cada tarefa do zero, enquanto as técnicas de aprendizagem por transferência tentam transmitir o conhecimento de algumas tarefas anteriores para um alvo quando se possui poucos dados de qualidade para o treinamento da tarefa alvo.

Figura 11 – Diferença do processo de aprendizagem entre (a) aprendizado de máquina tradicional e (b) aprendizado por transferência



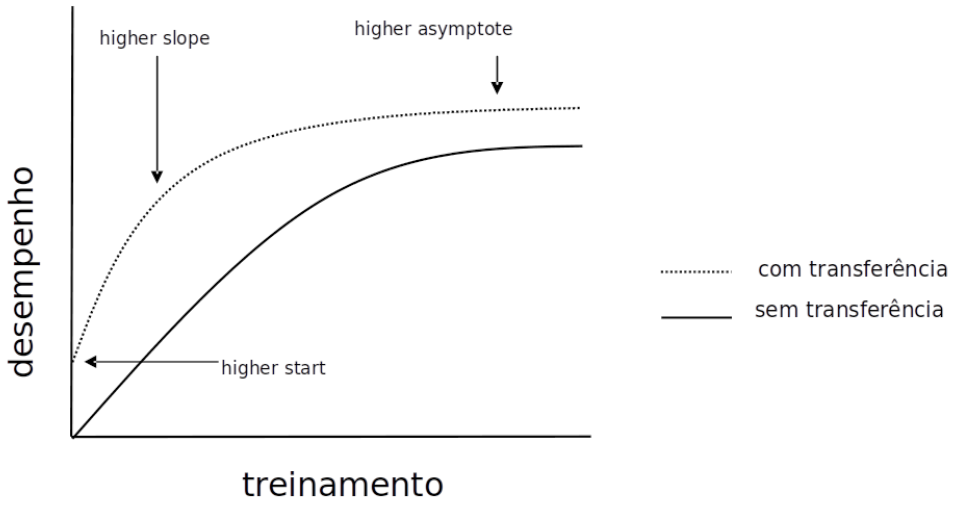
Fonte: Adaptado de Pan e Yang (2010)

Aprendizagem por transferência tem como objetivo aprimorar o processo de aprendizado na tarefa alvo, aproveitando o conhecimento adquirido na tarefa fonte. Segundo Olivas *et al.* (2009), existem três medidas comuns que demonstram como o aprendizado por transferência pode melhorar a aprendizagem. A Figura 12 ilustra de forma esquemática essas três medidas.

A primeira medida *higher start* na figura, refere-se ao desempenho inicial alcançado na tarefa alvo, que utilizou apenas o conhecimento transferido antes de algum aprendizado adicional em comparação ao desempenho inicial de um agente sem conhecimento prévio. A segunda medida *higher slope*, diz respeito ao tempo necessário para dominar completamente a tarefa alvo com base no conhecimento transferido, em comparação com o tempo necessário para aprender a tarefa do zero. Por fim, a terceira medida *higher asymptote*, avalia o nível de desempenho final atingido na tarefa alvo em comparação ao nível final que seria alcançado sem a transferência de conhecimento (OLIVAS *et al.*, 2009).

A transferência de aprendizado pode ser tanto positiva quanto negativa dependendo de seu impacto no desempenho da tarefa. Segundo Olivas *et al.* (2009), um dos principais desafios no desenvolvimento de métodos desse aprendizado é executar transferências positivas entre

Figura 12 – Medidas de desempenho: comparando aprendizado de máquina tradicional e aprendizado por transferência.



Fonte: Adaptado de Olivas *et al.* (2009)

tarefas adequadamente relacionadas e evitar ações negativas entre tarefas menos relacionadas.

2.4.1 Transferência na aprendizagem indutiva

A aprendizagem por transferência usada junto a profunda é chamada de transferência indutiva. É aqui que o escopo de modelos possíveis (viés do modelo) será reduzido de forma benéfica usando um ajuste de modelo em uma tarefa diferente, mas relacionada (OLIVAS *et al.*, 2009).

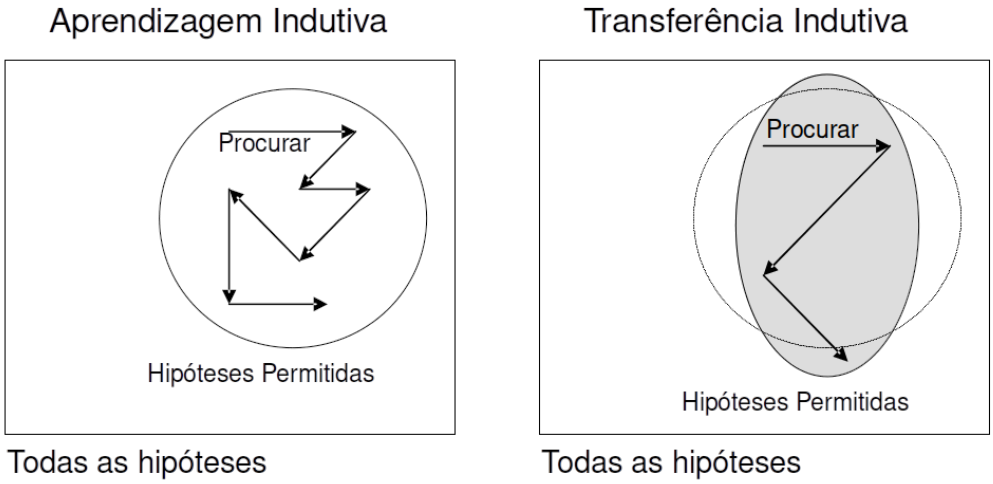
Aprendizagem indutiva é uma técnica automática para aquisição de conhecimento. Em uma tarefa de aprendizagem indutiva, o objetivo é induzir um modelo preditivo a partir de um conjunto de exemplos de treinamento (MITCHELL, 1997). Muitas vezes, o objetivo de um modelo de aprendizagem indutiva é a classificação, ou seja, atribuir rótulos de classe aos exemplos. Exemplos de sistemas de classificação são redes neurais artificiais e aprendizes de regras simbólicas (OLIVAS *et al.*, 2009).

Um algoritmo de aprendizagem indutiva deve gerar um modelo com uma capacidade alta de generalização que conduza previsões precisas tanto nos exemplos de treinamento, quanto em exemplos futuros ainda não conhecidos. Para produzir um modelo com esta capacidade de generalização, um algoritmo de aprendizagem deve ter um viés indutivo (OLIVAS *et al.*, 2009). O viés consiste em um conjunto de suposições sobre a verdadeira distribuição dos dados de treinamento, e é baseado no espaço de hipóteses que representa os possíveis modelos que o algoritmo considera durante o processo de aprendizado (OLIVAS *et al.*, 2009).

Nos métodos de transferência indutiva, o viés indutivo da tarefa-alvo é escolhido ou ajustado com base no conhecimento da tarefa-fonte. Segundo Mitchell (1997), a aprendizagem

indutiva pode ser vista como uma busca direcionada através de um espaço de hipóteses específicos. Como mostrado na Figura 13, a transferência indutiva utiliza o conhecimento da tarefa fonte para ajustar o viés indutivo, o que pode envolver a mudança do espaço de hipóteses ou das etapas de busca.

Figura 13 – Aprendizagem indutiva: Conjunto de hipóteses



Fonte: Adaptado de Olivas *et al.* (2009)

A transferência na aprendizagem indutiva tem geralmente o intuito de aumentar a velocidade que um modelo é aprendido ou melhorar a sua capacidade de generalização (OLIVAS *et al.*, 2009). A transferência funciona permitindo que o conhecimento da tarefa fonte afete o viés indutivo da tarefa alvo.

2.4.2 Notações e definições

Utilizaremos as definições e notações fornecidas por Pan e Yang (2010) para o conceito de aprendizado por transferência.

A aprendizagem por transferência envolve os conceitos de um domínio e uma tarefa. Um domínio $\mathcal{D}$ consiste em dois componentes: um espaço de recursos $\mathcal{X}$ e uma distribuição de probabilidade marginal $P(X)$, onde $X = x_1, \cdots, x_n \in \mathcal{X}$. Em geral, se dois domínios são diferentes, eles podem ter espaços de características diferentes ou distribuições de probabilidades marginais diferentes (PAN; YANG, 2010).

Dado um domínio $\mathcal{D} = \{\mathcal{X}, P(X)\}$, uma tarefa $\mathcal{T}$ consiste em dois componentes: um espaço de rótulos $\mathcal{Y}$ e uma distribuição de probabilidade condicional $P(Y|X)$, que normalmente é aprendido a partir dos dados de treinamento que consistem em pares $x_i \in X$.

Dado um domínio de origem $\mathcal{D}_S$ uma tarefa de origem correspondente $\mathcal{T}$, bem como um domínio de destino $\mathcal{D}_T$ e uma tarefa alvo $\mathcal{T}_T$, o objetivo da aprendizagem por transferência agora

é permiti aprender a distribuição de probabilidade condicional alvo $P(Y_T|X_T)$ em $\mathcal{D}_T$ com as informações obtidas $\mathcal{D}_S$ e $\mathcal{T}_T$ onde $\mathcal{D}_S \neq \mathcal{D}_T$. Na maioria dos casos presume-se que esteja disponível um número limitado de exemplos de destino rotulados, que é exponencialmente menor do que o número de exemplos de origem rotulados.

Dados os domínios de origem e destino $\mathcal{D}_S$ e $\mathcal{D}_T$ onde $\mathcal{D} = \{\mathcal{X}, P(X)\}$, tarefas de origem e destino$\mathcal{T}_S$ e $\mathcal{T}_T$ onde $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, as condições de origem e destino podem variar de quatro maneiras:

1. $\mathcal{X}_S \neq \mathcal{X}_T$. Os espaços de características do domínio de origem e de destino são diferentes.
2. $P(X_S) \neq P(X_T)$. As distribuições de probabilidade marginal do domínio fonte e alvo são diferentes.
3. $\mathcal{Y}_S \neq \mathcal{Y}_T$. Os espaços de rótulo entre as duas tarefas são diferentes. Geralmente este cenário ocorre com o cenário 4, pois é extremamente raro que duas tarefas diferentes tenham espaços de rótulos diferentes, mas as mesmas distribuições de probabilidade condicional.
4. $P(Y_S|X_S) \neq P(Y_T|X_T)$. As distribuições de probabilidade condicional das tarefas de origem e de destino são diferentes.

2.4.3 Ajuste fino

No processo mais comum de aprendizado por transferência são obtidas camadas de um modelo pré-treinado, as camadas são congeladas, novas camadas treináveis são adicionadas e treinadas com o novo conjunto de dados. Após esse treinamento, pode ser feito uma última etapa chamada de ajuste fino (KERAS, 2023b).

Na etapa de ajuste fino algumas ou todas as camadas do novo modelo obtido são descongeladas e é retreinado com uma taxa de aprendizagem muito baixa. Nessa etapa, os parâmetros herdados do modelo origem são ajustados para o conjunto de dados do modelo destino. Esse estágio pode melhorar a forma de aprendizado, porém pode levar a um sobre-ajuste (*overfitting*), por isso deve ser feito a uma taxa baixa de aprendizado (ZHANG *et al.*, 2023).

2.5 MODELOS PRÉ-TREINADOS PARA RECONHECIMENTO DE IMAGEM

As RNCs tem sido usadas para diversas aplicações, como reconhecimento de objetos, super-resolução de imagens, segmentação semântica, etc. Isso se deve ao seu robusto mecanismo de extração e aprendizagem de recursos. Várias arquiteturas RNC foram propostas para a tarefa de reconhecimento de objetos em imagens. A Tabela 3 mostra as arquiteturas de aprendizado profundo pré-treinadas na base de dados *ImagemNet* e disponíveis na biblioteca Keras.

A primeira coluna contem o nome da arquitetura. A secunda coluna contem a quantidade de parâmetros. Por fim, a terceira coluna, contem a profundidade topológica da rede.

Tabela 3 – Principais modelos pré-treinados no conjunto ImageNet.

Modelo	Parâmetros	Profundidade
Xception	22.9 milhões	81
VGG16	138.4 milhões	16
VGG19	143.7 milhões	19
ResNet50	25.6 milhões	107
ResNet50V2	25.6 milhões	103
ResNet101	44.7 milhões	209
ResNet101V2	44.7 milhões	205
ResNet152	60.4 milhões	311
ResNet152V2	60.4 milhões	307
InceptionV3	23.9 milhões	189
InceptionResNetV2	55.9 milhões	449
MobileNet	4.3 milhões	55
MobileNetV2	3.5 milhões	105
DenseNet121	8.1 milhões	242
DenseNet169	14.3 milhões	338
DenseNet201	20.2 milhões	402
NASNetMobile	5.3 milhões	389
NASNetLarge	88.9 milhões	533
EfficientNetB0	5.3 milhões	132
EfficientNetB1	7.9 milhões	186
EfficientNetB2	9.2 milhões	186
EfficientNetB3	12.3 milhões	210
EfficientNetB4	19.5 milhões	258
EfficientNetB5	30.6 milhões	312
EfficientNetB6	43.3 milhões	360
EfficientNetB7	66.7 milhões	438
EfficientNetV2B0	7.2 milhões	-
EfficientNetV2B1	8.2 milhões	-
EfficientNetV2B2	10.2 milhões	-
EfficientNetV2B3	14.5 milhões	-
EfficientNetV2S	21.6 milhões	-
EfficientNetV2 milhões	54.4 milhões	-
EfficientNetV2L	119.0 milhões	-
ConvNeXtTiny	28.6 milhões	-
ConvNeXtSmall	50.2 milhões	-
ConvNeXtBase	88.5M	-
ConvNeXtLarge	197.7M	-
ConvNeXtXLarge	350.1M	-

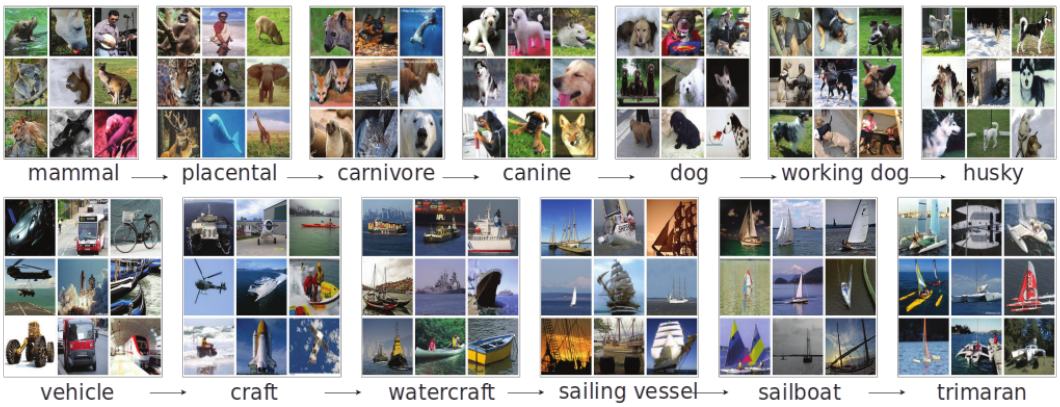
Fonte: Keras (2023a)

A base de dados ImagemNet introduzida por Deng *et al.* (2009), é uma base organizada

em hierarquia, onde cada nó da hierarquia é representado por centenas e milhares de imagens (ImagiNet, 2020). Os dados dessa base estão disponíveis gratuitamente aos pesquisadores para uso não comercial (ImagiNet, 2020).

A Figura 14 mostra uma representação da hierarquia do *ImageNet*: A primeira linha de imagens representa a subárvore de mamíferos; a segunda linha de imagens representa a subárvore de veículos.

Figura 14 – Hierarquia dos dados na base ImageNet



Fonte: Deng *et al.* (2009)

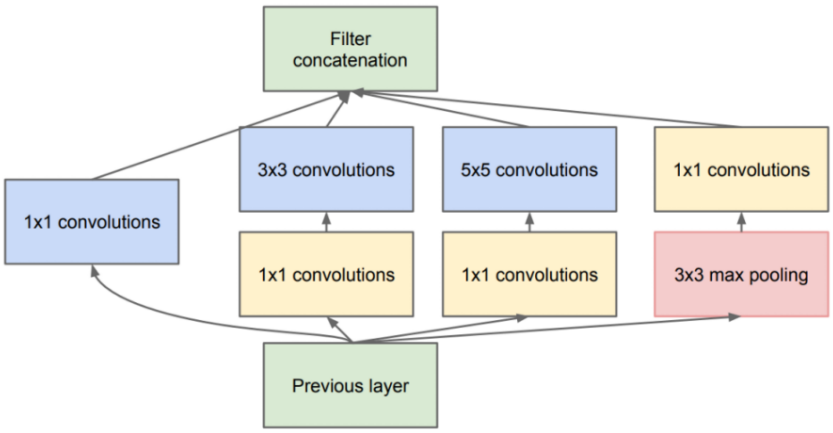
2.5.1 Arquitetura Inception

Arquitetura de RNC do Google conhecida como *Inception* (SZEGEDY *et al.*, 2015a) foi concebida em 2014. Antes da *Inception* as pesquisas que visavam aumentar a performance focavam apenas no aumento da profundidade das redes convolucionais. A arquitetura *Inception* representou um marco no desenvolvimento das RNCs ao empregar novas técnicas para aprimorar a velocidade e a acurácia da rede sem aumentar sua profundidade.

A arquitetura *Inception* é formada por módulos chamados módulos *Inception*, ilustrado na figura Figura 15. Na figura são aplicados filtros de tamanhos 1x1, 2x2 e 3x3 e a camada de agrupamento máximo 3x3 em paralelo. Nota-se na figura que antes dos filtros 3x3 e 5x5 é aplicado um filtro 1x1, percebe-se que após a camada de agrupamento é aplicado um filtro 1x1 para diminuir a dimensionalidade.

Diferente das arquiteturas que a antecederam, a *Inception* usa diferentes tamanhos de filtros (1x1, 3x3, 5x5) e uma camada de agrupamento máximo ao mesmo tempo, sobre a entrada (SZEGEDY *et al.*, 2015a). Com a aplicação de diferentes tamanhos de filtros busca-se capturar padrões em escalas diferentes (SZEGEDY *et al.*, 2015a), esses filtros são aplicados em paralelos e as suas saídas são concatenadas. Para diminuir a dimensionalidade do volume de dados e a carga computacional, arquitetura *Inception* também inclui blocos de convolução 1x1 antes das camadas 3x3 e 5x5 (SZEGEDY *et al.*, 2015a).

Figura 15 – Modulo *Inception*



Fonte: Szegedy *et al.* (2015a)

2.5.2 Arquitetura MobileNet

A *MobileNet* (HOWARD *et al.*, 2017) é uma arquitetura de redes neural convolucional de código aberto do Google, lançada em 2017. Nessa arquitetura são utilizadas convoluções separáveis em profundidade. A utilização de convoluções separáveis tem a intenção de diminuir o número de parâmetros.

A convolução separável em profundidade refere-se a aplicação de filtros em todas as dimensões do volume de dados. Na arquitetura MobileNet a convolução é dividida em duas etapas: uma convolução em profundidade e uma convolução pontual. Na convolução em profundidade um único filtro é aplicado a cada canal de entrada, em quanto na convolução pontual um filtro 1x1 é aplicado para combinar as saídas da convolução em profundidade (RODRíGUEZ, 2021).

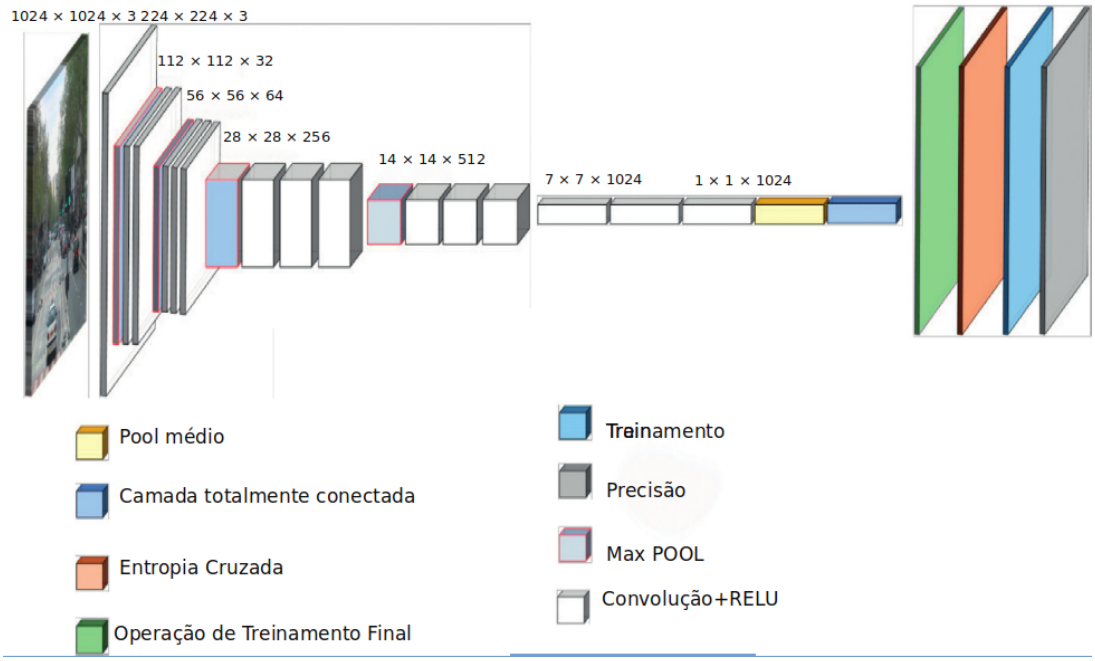
A Figura 16 ilustra a arquitetura do *MobileNet* na versão V1. A imagem de entrada com uma dimensão 224 x 224 e três canais é inserida na entrada. A primeira camada é a camada de convolução com valor de passada igual a dois. Em seguida, as camadas convolução profunda e pontual se revesam. A camada de convolução sempre são seguidas de uma função de ativação ReLU. A camada pontual sempre dobra o número de canais e diminui a dimensionalidade dos dados. Esse processo se repete até o volume de dados seja reduzido a uma dimensão 7x7 pixeis com 1024 canais. Por último uma operação de pooling médio é aplicada resultando em volume de 1x1x1024 (KADAM; AHIRRAO; KOTECHA, 2022).

No ano de 2018 a arquitetura *MobileNetv2* foi lançada, essa segunda versão do *MobileNet* introduz dois novos recursos na arquitetura: 1) gargalos lineares [2] entre as camadas e 2) conexões de atalho [3] entre os gargalos. As conexões de atalho são comumente usadas

[2] Gargalos lineares são camadas que realizam transformações lineares nos dados. Em CNNs essas camadas podem ser utilizados para reduzir a dimensionalidade dos dados antes de uma operação não linear. Mais detalhes em Sandler *et al.* (2019)

[3] Conexões de atalho pulam algumas das camadas da rede neural e alimenta a saída de uma camada como

Figura 16 – A arquitetura do MobileNet V1



Fonte: Adaptado de Kadam, Ahirrao e Kotecha (2022)

para conectar as camadas sem gargalo, mas no *MobileNetV2* os gargalos são conectados diretamente (SANDLER *et al.*, 2019). Testes sugerem que a versão V2 alcance precisão competitiva com significativamente menos parâmetros e menor complexidade computacional em comparação a primeira versão.

Em 2019 foi lançada a versão *MobileNetV3* (HOWARD *et al.*, 2019), entre outras melhorias a versão V3 acrescenta técnicas de aprendizado de máquina automatizada (AutoML) para escolha da melhor arquitetura de rede neural possível, para um determinado problema (RODRíGUEZ, 2021; HOWARD *et al.*, 2019).

entrada para as próximas camadas.

3 DESENVOLVIMENTO DE SISTEMA DE SEGMENTAÇÃO DE PLACAS DE VEÍCULOS

O presente capítulo se propõe a descrever os conceitos vistos e aplicados na implementação e treinamento de modelos de aprendizagem de máquina, baseados na arquitetura *MobileNetV2*. A partir do treinamento, o modelo deve ser capaz de reconhecer a região da imagem que contém a placa veicular, dentro de uma janela de deslocamento com um percentual satisfatório de acerto. Apresentam-se as etapas, ferramentas empregadas no desenvolvimento e o detalhamento dos testes e avaliações.

3.1 PERCURSO METODOLÓGICO

A proposta deste trabalho consistiu em uma pesquisa de natureza exploratória, a qual visou investigar, compreender e aplicar técnicas que utilizam aprendizado por transferência para a extração das regiões que contém placas veiculares.

Para isso, o trabalho foi desenvolvido em cinco etapas:

1. Etapa 1: Escolha de uma base de imagens de veículos com placas visíveis
2. Etapa 2: Processamento e preparação das imagens
3. Etapa 3: Ajuste da arquitetura original para treinamento e ajuste fino
4. Etapa 4: Execução dos algoritmos de aprendizado
5. Etapa 5: Avaliação dos resultados.

Essas etapas são descritas nas seções deste capítulo.

3.1.1 Ambientes de Programação Utilizados

Nesta seção apresentam-se as bibliotecas, ferramentas e plataformas que foram utilizadas nesse trabalho. Com a intenção de melhor administrar o tempo, foram utilizadas algumas bibliotecas de softwares já estabelecidas e testadas.

Para este trabalho foi utilizado o ecossistema *Python*, que contém várias ferramentas de aprendizado de máquina, visualização e processamento de dados. O desenvolvimento foi feito utilizando *notebooks* na plataforma *Google Colaboratory*. A manipulação das imagens foi realizada através do uso da biblioteca *OpenCv*. Para realizar o treinamento do modelo de segmentação das placas foram utilizadas as bibliotecas *TensorFlow* e *Keras*, que possuem a arquitetura MobileNetV2 pré-trinada no *dataset ImageNet*.

3.1.2 Etapa 1: Escolha de uma base de imagens de veículos com placas visíveis

Para treinar o modelo de aprendizado por transferência necessita-se de um conjunto de dados relevantes e confiável. É importante que os dados estejam limpos e sem ruídos, para não prejudicar a precisão dos resultados. Dentre os *dataset* encontradas na revisão sistemática, foi selecionado o *dataset UFPR-ALPR* (Laroca *et al.*, 2018) para o treinamento e validação do modelo de rede neural proposto para segmentação de imagens de placas de veículos. O *dataset* é composto por imagens de veículos com placas brasileiras, devidamente anotadas. Amostras das imagens estão ilustradas na Figura 17. Elas foram capturadas em ambientes urbanos e incluem vários tipos de veículos, tais como: viaturas, carros de passeio, motos, ônibus e caminhões.

Figura 17 – Amostras da base *UFPR-ALPR* com imagens de carros e placas

Fonte: (LAROCA *et al.*, 2018)

O *dataset UFPR-ALPR* totaliza 4.500 imagens de 150 veículos, totalmente anotadas. As imagens estão disponíveis no formato Portable Network Graphics (PNG) com tamanho de 1.920 × 1.080 pixeis (LAROCA *et al.*, 2018). Para coleta das imagens os autores utilizaram três câmeras diferentes, sendo elas: *GoPro Hero4 Silver*, *Huawei P9 Lite* e *iPhone 7 Plus* (LAROCA *et al.*, 2018). Cada uma das câmeras registrou 1500 imagens de placas. As imagens registradas estão divididas da seguinte forma: 900 carros com placas cinza, 300 carros com placa vermelha e 300 motos com placa cinza. A Figura 18 mostra exemplos das três placas contidas na base de dados, sendo respectivamente: placa cinza, placa vermelha e placa de moto.

Os autores do dataset *UFPR-ALPR* disponibilizam o conjunto de dados já divididos da seguinte forma: 40% para treinamento, 40% para teste e 20% para validação. Para cada imagem existe um arquivo de texto com as anotações. Conforme mostra a Figura 19, as anotações contidas no arquivo são: a câmera em que a imagem foi tirada, a posição do veículo e informações

Figura 18 – Exemplos dos três tipos de placas contidas na base UFPR-Alpr

(a) Placa cinza de carro (b) Placa vermelha de carro (c) Placa cinza de moto

Fonte: Adaptado de *UFPR-ALPR* (LAROCA *et al.*, 2018)

como tipo (carro ou moto), fabricante, modelo e ano, a identificação e posição da placa, bem como a posição dos caracteres da placa (LAROCA *et al.*, 2018).

Figura 19 – Arquivo de anotações da base *UFPR-ALPR*

```
camera: Huawei P9 Lite
position_vehicle: 373 153 721 719
        type: car
        make: Peugeot
        model: Boxer
        year: 2014
plate: AYE-3205
position_plate: 515 544 153 50
        char 1: 522 562 18 26
        char 2: 540 561 19 27
        char 3: 559 562 19 26
        char 4: 587 563 17 25
        char 5: 604 563 19 25
        char 6: 623 563 17 25
        char 7: 641 563 18 25
```

Fonte: O autor

O uso do dataset *UFPR-ALPR* tem como pontos positivos: estar adequado a realidade brasileira, tanto de veículos como de ambientes; e já ter outros artigos publicados que utilizaram essa base. Como pontos negativos está o sistema de placas contidos, que não engloba placas mais recentes. As placas registradas na base estão no modelo de placas de 1990 a 2018, definidas por uma resolução do Conselho Nacional de Transitos (CONTRAN) identificada como 231/2007 (CONTRAN, 2007). Esse modelo precedo o modelo de placas do Mercosul estabelecido pela resolução 729/2018 (CONTRAN, 2018). Porem o modelo de placas de 1990 a 2018 ainda está em validade no Brasil (CONTRAN, 2007).

3.1.3 Etapa 2: Processamento e preparação dos dados

Foi necessário preparar tanto as imagens quanto as anotações para se adequar ao modelo *MobileNetV2* e ao objetivo de segmentação das placas. Para isso, na etapa de pré-processamento foi realizado os seguintes principais processamentos: aumento do dataset, redimensionamento e preenchimento das imagens, extração de dados relevantes das anotações, conversão do formato das anotações, organização dos dados e a normalização dos dados.

O primeiro processamento realizado foi o aumento do dataset. Esse processamento foi realizado para ampliar a quantidade de imagens disponíveis para treinamento e aumentar a diversidade dos dados. Para realizar o aumento do dataset foi realizado uma transformação a cada imagem que consistem no enquadramento e recorte da região do veículo. Para recortar o veículo foi extraído a posição do veículo do arquivo de anotações. Após a imagem ser cortada, ela era redimensionada e tornada quadrada. A anotação da placa para a nova imagem foi calculada subtraindo a posição inicial do veículo da posição inicial da placa. Na Figura 20 temos exemplos das transformações aplicadas nas imagens. A imagem *a* é a original, a imagem *b* tem o recorte do veículo, por fim, em *c* temos a imagem redimensionada e preenchida para ficar com altura e largura de 224 pixeis.

Figura 20 – Exemplos das transformações aplicadas para aumento da base

(a) Imagem original (b) Recorte de veículo (c) Redimensionada e preenchida

Fonte: Adaptado de *UFPR-ALPR* (LAROCA *et al.*, 2018)

O segundo processamento realizado foi o redimensionamento das imagens originais. Esse processamento foi necessário, pois a rede *MobileNet* espera na camada de entrada dados com dimensões de 224x244x3 que não são compatíveis com as dimensões das imagens que tinham 1080 pixeis de altura e 1920 pixeis largura. Por isso as imagens foram redimensionadas e tiveram a largura preenchida para ficarem com dimensões de altura e largura igual a 224 pixeis. O resultado desse processamento é ilustrado na Figura 21 *b*, e a imagem original na Figura 21 *a*.

Além do processamento das imagens também foi necessário tratar as anotações. Foi realizando a extração das coordenadas das placas dos arquivos de anotações, isolando as infor-

Figura 21 – Exemplos do redimensionamento das imagens

(a) Imagem original (b) Imagem redimensionada e preenchida

Fonte: Adaptado de *UFPR-ALPR* (LAROCA *et al.*, 2018)

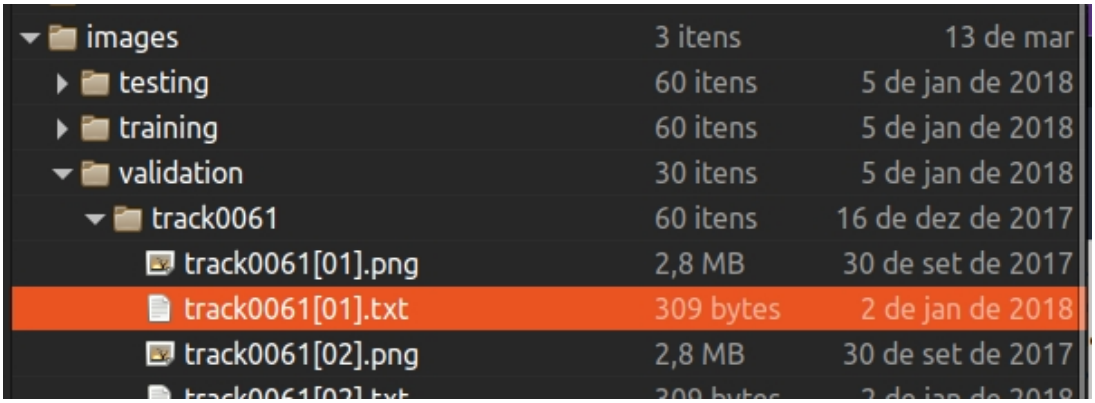
mações relevantes para a segmentação. As anotações do *dataset UFPR-ALPR* eram separadas em arquivos de texto, onde havia um arquivo para cada imagem do *dataset*. Nesse arquivo de texto encontrava-se na segunda linha a posição do veículo, necessário para o processamento de aumento da base, e na oitava linha a posição da placa que foi utilizado como dados de domínio do modelo de aprendizado.

Outro processamento realizado foi a conversão do formato das anotações. As anotações de posição eram compostas por quatro valores, sendo o primeiro e o segundo valores as coordenadas cartesianas do ponto que representa o canto superior esquerdo da caixa delimitadora. O terceiro e quarto valor eram, respectivamente, a largura e a altura da caixa delimitadora. Esse formato de coordenadas é nomeado de *XYWH* na documentação do *Keras*. Nesse processamento foi feito a conversão das anotações para um formato com dois pontos cartesianos, sendo eles o canto superior esquerdo e o cato inferior direito da caixa delimitadora. Esse formato é chamado de *XYXY* na documentação (Keras, 2023).

O quinto processamento realizado foi a organização dos dados. Os arquivos no *dataset* estavam separados em três níveis de sub-pastas. No primeiro nível há uma pasta para cada fase do aprendizado: treinamento, validação e teste. No segundo nível da estrutura há uma pasta para cada veículo. Por fim, no terceiro nível, encontram-se as imagens do veículo, juntamente com um arquivo de anotação para cada imagem. Essa estrutura é ilustrada na Figura 22. Nessa fase de processamento já tínhamos as anotações extraídas e as imagens processadas, então esses dados foram organizados em duas listas ordenadas: a primeira contendo os dados das imagens em memória, e a segunda contendo as anotações das placas nessas imagens.

No último processamento foi realizado a normalização dos dados, padronizando os valores de entrada para o modelo e facilitando o processo de treinamento. Nesse processamento

Figura 22 – Organização dos arquivos na base *UFPR-ALPR*



Fonte: O autor

foi convertido as imagens e anotações para o intervalo de zero a um utilizado pela rede *MobileNetV2*.

3.1.4 Etapa 3: Ajuste da arquitetura original

A arquitetura *MobileNetV2* precisou ser ajustada para o problema de segmentação das placas. Originalmente a arquitetura *MobileNetV2* é usada para um problema de classificação de objetos e tem na camada de saída mil duzentos e oitenta neurônios. Para ajustar a arquitetura foi adicionado cinco novas camadas. A primeira camada foi uma camada de achatamento, ou camada flatem, que torna a saída unidimensional. Em seguida, foi adicionado 4 novas camadas do tipo densas ao final da arquitetura. As camadas densas adicionadas continham respectivamente: 128, 64, 32 e 4 neurônios. O trecho de código do Algoritmo 1 demostra o ajuste feito na arquitetura.

Algoritmo 1 – Ajuste da arquitetura *MobileNet*

```
from tensorflow.keras.applications import MobileNetV2
#  Instanciar o MobileNetV2 com os pessos do imagenet
mobile_net = MobileNetV2(weights="imagenet",include_top=False,
                                input_tensor=Input(shape=(224,224,3)))
#  Congela todas as camadas da arquitetura original
mobile_net.trainable=False
for layer in mobile_net.layers:
    layer.trainable = False
headmodel = mobile_net.output
headmodel = Flatten()(headmodel)
#  Adiciona 4 camadas reduzindo a saida ate quatro neuronios
headmodel = Dense(128, activation="relu")(headmodel)
headmodel = Dense(64, activation="relu")(headmodel)
headmodel = Dense(32, activation="relu")(headmodel)
headmodel = Dense(4, activation="sigmoid")(headmodel)
# Monta o modelo
```

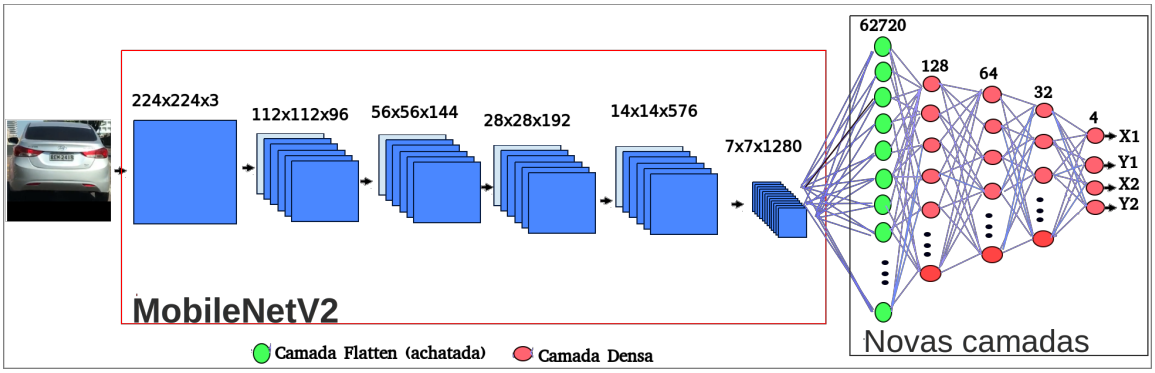
```
17 model = Model(inputs=mobile_net.input, outputs=headmodel)
```

Fonte: O Autor (2024)

A arquitetura *MobileNetV2* já foi treinada previamente com a base de dados *Imagenet*, os pesos desse treinamento foram usados para fazer a transferência de aprendizado para o problema de segmentação de placas. Por esse motivo a classe *MobileNetV2*, importada do Keras na linha 1, é instanciada com o parâmetro *weights* recebendo o valor *"imagenet"* na linha 3 do Algoritmo 1. Essa parametrização garante que os pesos sejam pré-carregados do *Imagenet*.

Após carregar a arquitetura com os pesos pré-carregados, na linha 8, as camadas da arquitetura são congeladas para que o treinamento não altere as camadas com pesos carregados. Na linha 10, é adicionado uma camada *Flatten* para converter a saída multi dimensional do modelo para uma única dimensão. Em seguida, nas linas de 12 a 15, são adicionadas quatro novas camadas densas ao modelo. A estrutura final da arquitetura proposta é ilustrado na Figura 23.

Figura 23 – Arquitetura proposta baseada em*MobileNetV2*



Fonte: O autor

Como mostrado na figura Figura 23, após a camada *Flatten* foi adicionado novas camadas densa. A primeira camada densa tem 128 neurônios e a cada nova camada o número de neurônios é diminuído pela metade, dessa forma a última camada fica com 4 neurônios que representam as coordenadas da localização estimada da placa. No primeiro momento do treinamento as camadas do *MobileNetV2* foram congeladas e apenas as novas camadas foram treinadas, foi posteriormente realizado o ajuste fino onde todas as camadas são descongeladas e a arquitetura é treinada com uma taxa de aprendizagem menor.

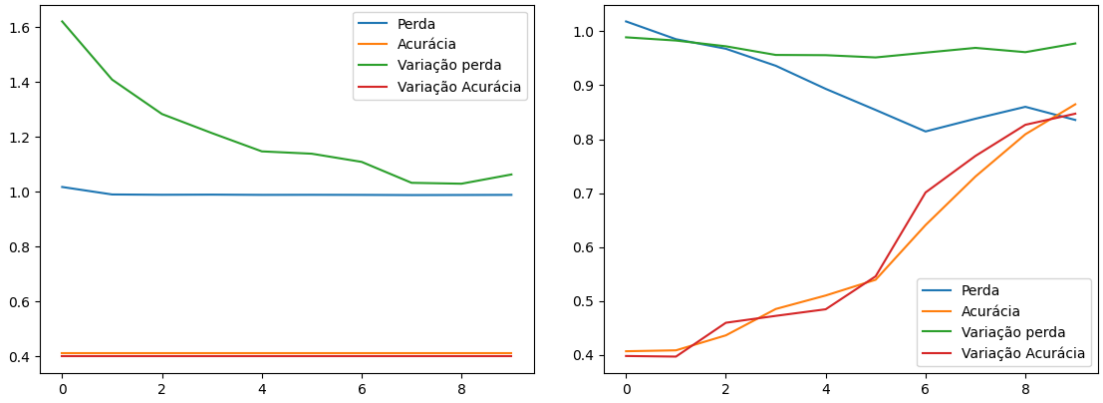
3.1.5 Etapa 4: Execução dos algoritmos de aprendizado

A arquitetura MobileNetV2 é uma rede complexa que possui 3,4 milhões de parâmetros. Para o treinamento eficaz de redes complexas como a MobileNetV2 é necessário grades recursos computacionais e uma quantidade substancial de dados. Como a base escolhida continha apenas 4500 imagens e não se tinha tempo nem recurso suficientes para treinar a rede do zero, optou-se pelo uso de técnicas de aprendizado por transferência. Utilizando essas técnicas

foi possível carregar a arquitetura proposta com pesos de treinamentos já efetuados na base ImageNet que possui mais de 1,2 milhões de imagens.

Para evidenciar os benefícios de utilizar um modelo pré-treinado, foi realizado um teste onde se treinou a rede proposta sem uma carga de pesos, e um segundo treino com o modelo pré-treinado. Esses treinos foram realizados por dez épocas cada, utilizando a função de otimização Adam e uma taxa de aprendizado de 6×10^{-5}. A primeira otimização que foi notada é o tempo gasto para rodar as dez épocas em uma máquina com 22,5GB de memória de GPU e 53GB de memória RAM, sendo 129 segundos para o treinamento do zero e 59 segundos no modelo pré-treinado. A Figura 24 ilustra os gráficos de variação das métricas de acurácia e perda parra esses treinamentos. No gráfico *a* é mostrado as métricas do treinamento a partir do zero, podemos notar que nas dez épocas a acurácia e a perda ficaram praticamente estagnadas, significando que não ouve nenhum aprendizado. Já no gráfico *b* podemos notar que houve uma melhora tanto da acurácia quanto da perda, significando que houve um aprendizado.

Figura 24 – Variação de métricas em treinamento do zero e com modelo pré-treinado



(a) Treinamento do zero

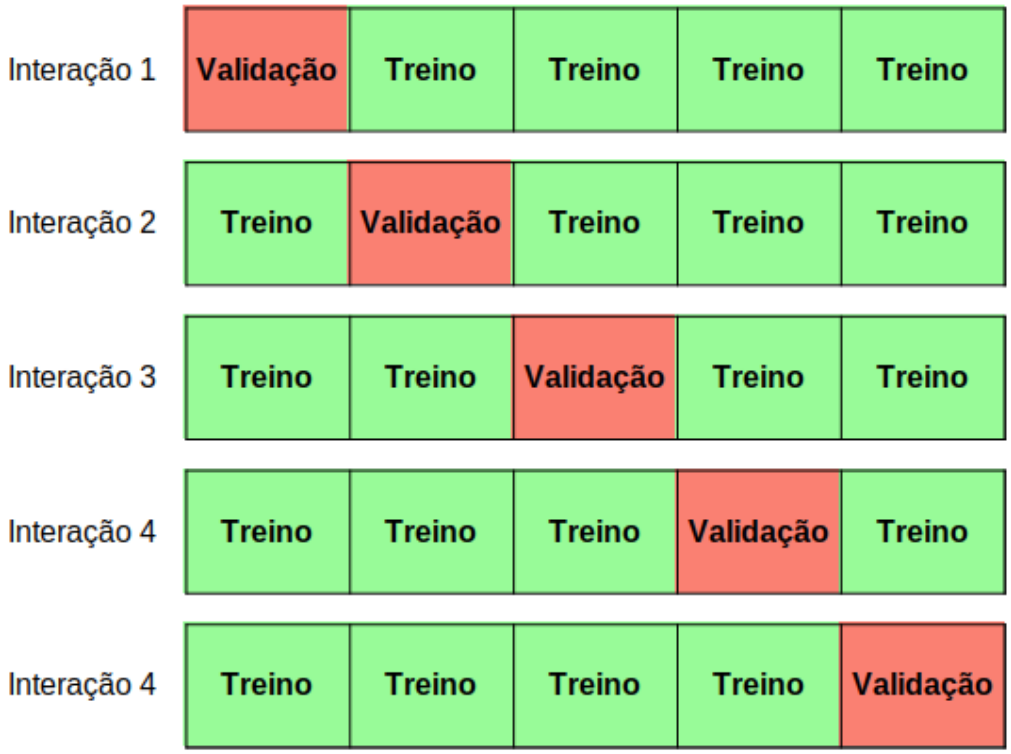
(b) Treinamento modelo pré-treinado

Fonte: O autor

Após estabelecer os benefícios do uso de modelos pré-treinados, o segundo passo nessa etapa foi a escolha dos hiper-parâmetros. Para isso foi realizar testes variados a fim de identificar os melhores hiper-parâmetros. Dentre eles, as escolhas dos valores para a função de perda, a função de otimização, a taxa de aprendizado, entre outros, foram determinantes para obter-se uma boa acurácia do modelo. Para analisar as possibilidades, foram realizados treinamentos prévios. Para isso, foram usados apenas dados de treinamento e validação. Esses dados foram unidos, randomizados e, por fim, divididos em cinco partições. Para cada conjunto de hiper-parâmetros foram feitas cinco execuções, retirando-se uma partição para avaliação do modelo, a Figura 25 ilustra o esquema de separação das partições e as interações realizadas.

Durante os treinamentos foram usadas para função de otimização as funções *Adam*, *AdamW*, *Nadam*, *Adamax* e *RMSprop*. As funções de otimização utilizam regras e algoritmos para ajuste dos pesos e parâmetros do modelo par minimizar a função de perda, tentando evitar

Figura 25 – Esquema de iterações com validação cruzada



Fonte: O autor

os mínimos locais e encontra o mínimo global. As funções de otimização recebem na instanciação como parâmetro a taxa de aprendizado inicial, cujo valor padrão é 1×10^{-3}. Encontra uma taxa de aprendizado adequada pode ser um desafio, pois se for usado uma taxa muito pequena o modelo pode demorar muito para aprender, já se for usado uma taxa muito grade o modelo pode convergir muito rápido para um mínimo local. Por isso os modelos foram treinados com três taxas de aprendizado: 1×10^{-3}, 1×10^{-4}, 1×10^{-5}.

Para executar os treinamentos foi preciso a alocação de recursos computacionais minimamente adequados. As bibliotecas utilizadas, como as bibliotecas de processamento de imagem e de estrutura e algoritmos de redes neurais, totalizavam um pouco mais de 600MB quando carregadas em memória. Além disso, o dataset ao ser carregado totalmente em memória utilizava cerca de 6.5GB de memória RAM. Porem ao executar o treinamento o uso de memória poderia ser multiplicado em várias vezes, podendo o uso ser menor ou maior dependendo da parametrização do tamanho do lote de processamento da arquitetura. Quanto menor o lote de processamento, maior o tempo de treinamento. Além da memória RAM, o treinamento também necessitava de uso intenso de processamento de GPU ou de CPU.

Inicialmente foi utilizado um ambiente de processamento local e um ambiente em nuvem gratuito disponibilizado pela Google na plataforma Colab. Os treinamentos eram divididos entre os dois ambientes, porem era necessários alguns dias para realizar todas as baterias de treinamentos devido às limitações de memória, processamento e de tempo de sessão no caso do

ambiente em nuvem. Com isso foi necessário contratar um plano pago na plataforma Google Colab. O plano pago Colab Pro dá acesso a máquinas com processamento em GPU poderosas, mais memória RAM e sem limite de tempo de sessão. A tabela Tabela 4 compara os ambientes Local, Google Colab e Google Colab Pro em termos de memória RAM, processador, modelo GPU, memória de GPU, tempo de sessão e custo. Como visto na última linha e ultima coluna ta tabela, o custo inicial do ambiente pago do Google Colab Pro era de R$58. Esse valor disponibilizava um número limitado de unidades de processamento, mas conforme necessário era possível adicionar mais unidades de processamento sobre demanda. No total foi gasto cerca de R$690 para fazer todos os treinamentos e validações.

Tabela 4 – Ambientes de execução.

	Local	**Google Colab**	**Google Colab Pro**
Memória RAM	16GB	13GB	80GB
Processador	Intel Core I5	Intel Xeon	Intel Xeon
Modelo GPU	Geforce 940mx	-	A100-SXM4
Memória de GPU	1GB	-	40GB
Tempo de sessão	-	4h	-
Custo	-	0	A partir de R$58

Fonte: O autor

A utilização de um ambiente de alto desempenho na plataforma Google Colab reduziu de forma expressiva o tempo de processamento para treinar e avaliar o modelo. Porem por se tratar de um ambiente em nuvem gerava um gargalo na carga dos dados para a memória, uma vez que era necessário carregar todas as imagens para o ambiente em nuvem utilizando a rede de internet. Esse processo demorava cerca de 40 muitos. Durante esse tempo erra necessário manter alocado um ambiente de alto desempenho mesmo não utilizando o seu poder de processamento, gerando um custo desnecessário. Para resolver esse gargalo, os dados foram carregados na máquina local, alocados em uma estrutura de dados do Python e então essa estrutura foi serializada em um único arquivo que foi hospedado em um ambiente remoto. A carga desse único arquivo e sua desserialização no ambiente em nuvem reduziu para cinco minutos a carga do dataset.

O método de EarlyStooping, da biblioteca Keras, foi empregado para garantir a execução dos treinamentos considerandos a perda e a fim de evitar a adequação do modelo aos dados, prejudicando a classificação de novas instâncias (*overfitting*). Esse método monitora uma métrica parametrizada e para o treinamento quando a métrica para de melhorar. Para a avaliação das parametrizações, o EarlyStooping foi configurado para parar somente quando houver três épocas sem melhoria.

Dado o volume de testes realizados, foram programadas rotinas específicas de controle das execuções e armazenamentos dos resultados. As rotinas foram executadas no ambiente colaborativo da Google Colab. A configuração de ambiente utilizada contava com uma GPU

A100 [1] com 40 gigabytes de memória. A maquina ainda contava com 80 gigabytes de memória RAM e armazenamento em SSD. O tempo médio de execução de cada treinamento foi de 29 segundos nessa configuração. Somando o tempo de carga do dataste no ambiente e preparação dos dados, um teste completo leva em média duas horas.

A tabela Tabela 5 apresenta uma lista descritiva dos principais hiper-parâmetros usados em cada treinamento e as médias das métricas de acurácia e perda considerando as cinco partições executadas. A lista está ordenada de forma decrescente para a métrica da acurácia média. Nas três primeiras colunas da tabela é apresentado os hiper-parâmetros, sendo, respectivamente, a funções de perda, a função de otimização e a taxa de aprendizado. Na quarta coluna é mostrado a média das cinco acurácias avaliadas para a parametrização indicada nas primeiras colunas. Por fim na quinta coluna é mostrado a média das perdas entre as cinco execuções, essa perda é calculada pela função de perda indicada na primeira coluna.

Observando os seis primeiros resultados da Tabela 5, podemos verificar que apenas esses ficaram com uma acurácia média de mais de 90%. Além disso, esses seis treinamentos tiveram uma diferença significativa de mais de cinco pontos percentuais para os demais treinamentos realizados. Levando em conta esses seis treinamentos podemos descartar a função de perda CIoU e a taxa de aprendizado de 1×10^{-3}. Pode-se observar que a diferença da acurácia media entre as cinco primeiras linhas da tabela é pouco significativa para afirmar a melhor parametrização. Porem a parametrização com a função de otimização Nadam e taxa de aprendizado de 1×10^{-4} se destaca por ter o menor valor de perda indicado na última coluna.

Como os testes demonstraram não haver diferença significativa entre as cinco parametrizações com maior acurácia, foi escolhido a parametrização com menor valor de perda para o treinamento e avaliação final. Assim, utilizou-se a função de perda GIoU com a parametrização padrão do Keras, a função de otimização Nadam, também com a parametrização padrão do Keras, e a taxa de aprendizado de 1×10^{-4}. Para realizar a técnica de ajuste fino após o treinamento inicial, a taxa de aprendizado foi alterada para 1×10^{-6} e o modelo foi treinado com essa nova taxa de aprendizado.

3.1.6 Etapa 5: Avaliação dos resultados

Após a avaliação dos diferentes hiper-parâmetros e realização do treinamento completo, o modelo gerado foi validado sobre os dados de testes e as métricas foram coletadas. Os resultados obtidos e as métricas coletadas são apresentadas no capítulo Capítulo 4.

[1] Informações disponíveis em <https://www.nvidia.com/en-us/data-center/a100/>

Tabela 5 – Resultado dos testes com hiper-parâmetros.

Função de perda	Função de otimização	Taxa de aprendizado	Acurácia média	Perda média
GIoULoss	Adamax	1×10^{-4}	0,947	0,6772
GIoULoss	Nadam	1×10^{-4}	0,944	0,6386
GIoULoss	Adam	1×10^{-5}	0,934	0,6953
GIoULoss	AdamW	1×10^{-5}	0,933	0,7055
GIoULoss	Nadam	1×10^{-5}	0,930	0,7046
GIoULoss	RMSprop	1×10^{-5}	0,913	0,7370
GIoULoss	Adamax	1×10^{-5}	0,864	0,8346
CIoULoss	Adamax	1×10^{-5}	0,829	1,0910
GIoULoss	RMSprop	1×10^{-4}	0,807	0,9405
GIoULoss	AdamW	1×10^{-4}	0,783	0,7923
CIoULoss	Adamax	1×10^{-3}	0,605	1,0198
GIoULoss	Nadam	1×10^{-3}	0,594	0,9902
GIoULoss	Adam	1×10^{-4}	0,579	0,9676
CIoULoss	Adamax	1×10^{-4}	0,513	1,2652
CIoULoss	RMSprop	1×10^{-3}	0,485	NaN
CIoULoss	Nadam	1×10^{-3}	0,482	NaN
GIoULoss	Adamax	1×10^{-3}	0,472	1,5604
CIoULoss	Adam	1×10^{-5}	0,414	1,1928
CIoULoss	AdamW	1×10^{-5}	0,399	1,2305
CIoULoss	Nadam	1×10^{-4}	0,391	1,2393
CIoULoss	Nadam	1×10^{-5}	0,363	1,1778
GIoULoss	AdamW	1×10^{-3}	0,359	1,3754
CIoULoss	AdamW	1×10^{-3}	0,350	NaN
CIoULoss	Adam	1×10^{-4}	0,336	1,2857
CIoULoss	AdamW	1×10^{-4}	0,266	1,2761
CIoULoss	RMSprop	1×10^{-4}	0,253	1,1805
GIoULoss	Adam	1×10^{-3}	0,244	1,5684
CIoULoss	RMSprop	1×10^{-5}	0,238	1,1538
GIoULoss	RMSprop	1×10^{-3}	0,237	1,7716
CIoULoss	Adam	1×10^{-3}	0,234	NaN

Fonte: O autor

4 RESULTADOS

Nesse capitulo será apresentado os resultados coletados na avaliação do modelo proposto. Para avaliação final do modelo, foi utilizado a separação de dados do dataset *UFPR-ALPR*. As imagens estão separadas em treinamento, validação e teste. O modelo foi treinado com os dados de treinamento e validação. Os dados de teste foram usados para testar após o treinamento e após o ajuste fino.

Alguns treinamentos foram feitos e o melhor resultado encontrado sem a aplicação do ajuste fino foi uma perda de 1,29 e uma acurácia de 85%. Após a aplicação do ajuste fino a acurácia subiu para 87,6% e a perda diminuiu para 0,959. A tabela Tabela 6 mostra essas informações.

Tabela 6 – Acurácia e Perda antes e depois do ajuste fino.

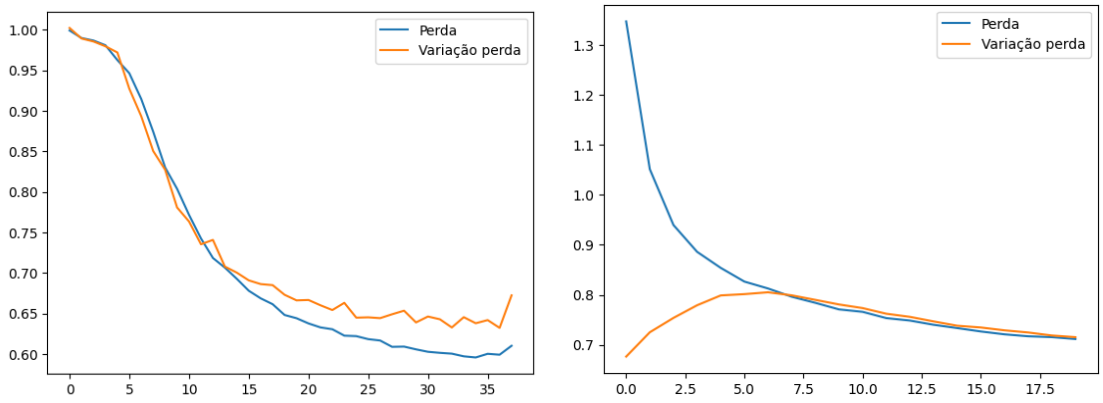
	Acurácia	**Perda**
Antes do ajuste fino	85%	1.29
Depois do ajuste fino	87.6%	0.96

Fonte: O autor

Os resultados encontrados demonstram que a aplicação do ajuste fino gerou uma melhoria significativa para métrica de perda. Para a aplicação do ajuste fino, primeiro os pesos foram pré-carregados de um aprendizado anterior sobre o dataset ImageNet, em seguida as camadas originais do modelo foram congeladas e somente as novas camadas foram treinadas. Por fim, após a primeira fase de treinamento todas as camadas foram descongeladas e treinadas com uma taxa de aprendizado de 1×10^{-6}. A Figura 26 ilustra os gráficos de evolução da métrica de perda durante o treinamento e durante o ajuste fino. O gráfico *(*a) demonstra a métrica de perda e a variação dessa métrica no decorrer das épocas do treinamento. Já o gráfico *b* mostra essas métricas durante o ajuste fino. Pode-se observar que, apesar de uma tendência de baixa da perda durante o treinamento, essa tendência não é constante de um uma época para outra, podendo aumentar ou diminuir, tendo um aumento nas últimas épocas. Já com o ajuste fino a tendência de baixa foi mais constante, assim foi possível corrigir os erros das últimas épocas de treinamento.

Para avaliar melhor o desempenho sobre cada tipo de imagem, as imagens de teste foram divididas por tipos. Primeiramente foram divididas entre imagens completas e imagens cortadas (onde o veículo foi enquadrado). Depois foram divididas entre imagens de motos e imagens de carros, gerando quatro grupo de imagens. O modelo foi avaliado sobre esses grupos para se verificar o desempenho em cada situação. A Figura 27 contem um infográfico que mostra as

Figura 26 – Evolução da métrica de perda durante o treinamento

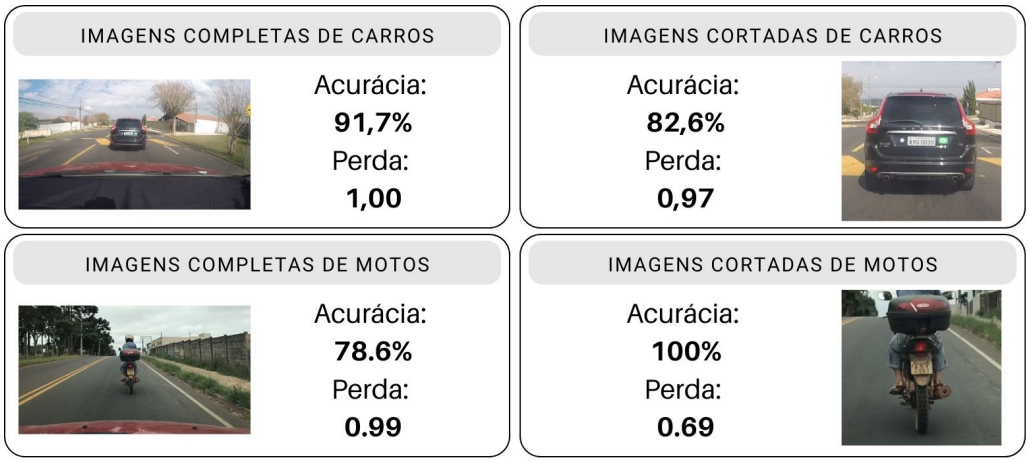


(a) Treinamento (b) Ajuste fino

Fonte: O Autor

métricas de acurácia e perda para cada um desses tipos de imagens.

Figura 27 – Infográfico de métricas por tipo de imagem

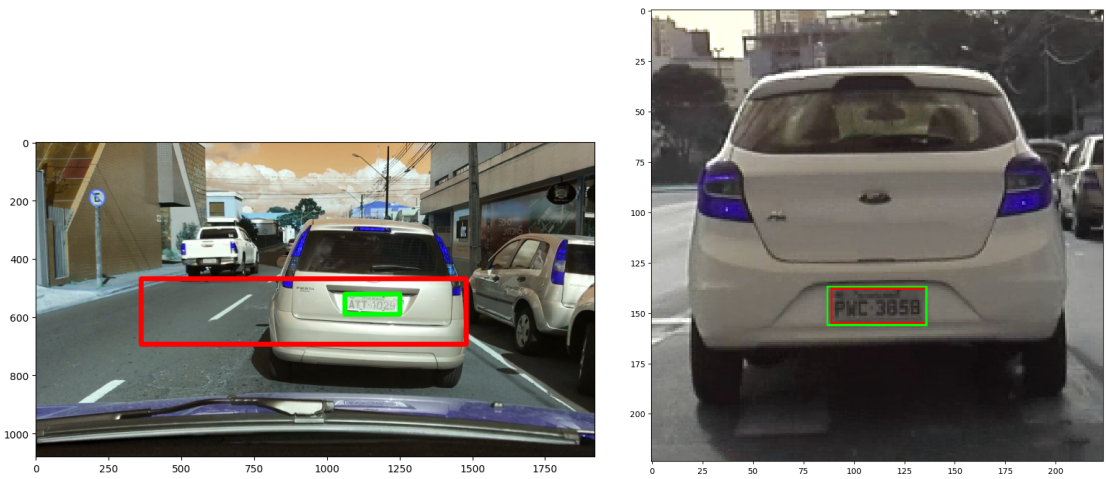


Fonte: O autor

Um ponto importante que os resultados demonstram, em particular a métrica de perda, é que o modelo tem dificuldade em segmentar a região da placa nas imagens completas, onde o veículo não foi enquadrado. A perda calculada para essas imagens é significativamente maior que para as imagens ajustadas. Nessas imagens a região da placa representa uma região muito pequena da imagem. Essas imagens ainda sofrem distorções ao serem redimensionadas para o tamanho de 224 pixeis de altura e largura esperado pelo modelo. Por outro lado, as imagens alteradas, em que a região do veículo foi cortada antes do redimensionamento, a região da placa na imagem representa uma porção maior. Além disso, essas imagens sofrem menos distorções ao serem redimensionadas, pois já tem um tamanho mais próximo ao esperado pelo modelo. A figura Figura 28 mostra exemplos dessas duas situações. As caixas em verde são a região anotada e as caixas em vermelho são as regiões previstas. Na imagem *a* a região da placa

prevista foi muito maior que região anotada, na imagem *b* as regiões anotadas e previstas não praticamente as mesmas.

Figura 28 – Caixas delimitadoras previstas pelo modelo



(a) Imagem original (b) Imagem com veículo enquadrado

Fonte: O autor

Para avaliar o modelo em situações não padronizadas, foi realizado a captura de novas imagens e submetido essas imagens a avaliação do modelo. As imagens foram capturadas com um celular modelo Galaxy S21 FE, utilizado uma resolução com 4000 pixeis de altura por 3000 pixeis de largura. Diferentemente das imagens do dataset essas imagens foram capturadas no modo retrato, ou seja, com altura maior que a largura. A Figura 29 mostra essas imagens já com a caixa delimitadora prevista pelo modelo. As imagens *a* e *b* são as imagens inteiras, já *c* e *d* são as imagens cordadas para enquadrar o veículo.

O primeiro veículo da Figura 29 possui a placa em um formato mais novo. O dataset utilizado para treinamento não continha placas no novo formato, mesmo assim o modelo conseguiu gerar uma caixa delimitador próxima da placa para a imagem completa. No caso do segundo veículo, a caixa delimitadora prevista para a imagem cortada foi mais próxima da placa do que para imagem completa. As caixas delimitadoras previstas pelo modelo nessas novas imagens, não conseguiram enquadrar corretamente a placa, cabe ainda uma avaliação dos motivos para isso e talvez um refinamento do modelo.

Figura 29 – Avaliação sobre imagens novas

Fonte: O autor

5 CONSIDERAÇÕES FINAIS

5.1 SÍNTESE DO TRABALHO

O objetivo deste trabalho constituiu em implementar e avaliar um modelo de aprendizado de máquina obtido por intermédio do aprendizado por transferência, para a tarefa de segmentação automática de placas de veículos em imagens. O modelo foi avaliado na aplicação prática em um contexto nacional utilizando o conjunto de dados *UFPR-ALPR*.

Para atingir esse objetivo, foram realizadas as etapas metodológicas descritas abaixo:

- **Revisão Bibliográfica:** Foi conduzido uma revisão sistemática da literatura para identificar o estado da arte em reconhecimento automático de placas veiculares e as principais técnicas de aprendizado por transferência.
- **Desenvolvimento do Modelo:** A arquitetura *MobileNetV2* foi adaptada para a tarefa de detecção e segmentação de placas, para isso novas camadas foram inseridas na arquitetura.
- **Pré-Processamento de Dados:** Foram desenvolvidos métodos para aumentar o conjunto de dados, extrair coordenadas das placas e converter anotações em um formato compatível com o modelo.
- **Treinamento e Validação:** O conjunto de dados *UFPR-ALPR* foi utilizado para treinar e validar o modelo, ajustando hiper-parâmetros e funções de perda para otimizar o desempenho. Foi avaliado a utilização de técnicas de aprendizado por transferência como o ajuste-fino.
- **Análise de Desempenho:** O modelo foi avaliado utilizando métricas de precisão, comparando diferentes funções de perda e técnicas de otimização.

Os resultados obtidos demonstram que o *MobileNetV2* pode ser adaptado para o problema de segmentação de placas, porém nem todas as técnicas de aprendizado por transferência trazem melhorias significativas para o problema. Destacam-se os seguintes achados:

- A abordagem proposta trouxe resultados práticos, porém cabe melhorias.
- A abordagem de ajuste-fino não trouxe melhorias significativas para o modelo de segmentação de placas.
- A função de perda *Generalized IoU (GIoU)* apresentou melhor desempenho na otimização da sobreposição entre caixas preditas e reais.

- As técnicas de pré-processamento, como redimensionamento e normalização de imagens, foram cruciais para a adequação dos dados à arquitetura do modelo.
- Esse tipo de problemas necessita de um volume grande de dados, o que resulta também na necessidade maior de recursos computacionais. Durante o desenvolvimento deste trabalho foi necessário fazer um upgrade para um ambiente pago do Google Colab para dispor de mais recursos de GPU e memória RAM.

5.2 CONTRIBUIÇÕES DO TRABALHO

Para atingir o objetivo deste trabalho foi apresentado um estudo sistemático da literatura. Neste estudo foram apresentados alguns trabalhos prévios com resultados satisfatórios na área de visão computacional centrados nas tarefas de detecção e segmentação de placas e veículos. Esses trabalhos também relacionaram uma ampla área de aplicações da tarefa de reconhecimento de placas, como a aplicação da lei de trânsito e o controle de fluxo e acesso em áreas particulares.

Muitas lições foram aprendidas no decorre do desenvolvimento desse trabalho, principalmente ao enfrentar os desafios encontrados. Um dos desafios superados foi a otimização do tempo de carga das imagens para o ambiente em nuvem, feito com o uso de técnicas de programação orientada a objetos, onde é feito a serialização de objetos em arquivo. Além dos desafios, os resultados encontrados, também trazem boas lições, mesmo não sendo tão bons quanto se esperava, eles apontam para uma nova solução, com uma faze preparatória para enquadramento do veículo, podendo ser com um novo modelo que extrai o veículo da imagem completa, ou até mesmo com pré-processamento das imagens.

Ao analisar o desenvolvimento desse trabalho, pode-se entender que o mesmo trouxe algumas contribuições relevantes para a aplicação de aprendizado por transferência aplicado na segmentação de placas de veículos. As principais contribuições são destacadas a seguir:

- **Desenvolvimento de uma arquitetura de rede neural convolucional especializada:** Foi proposto uma arquitetura de rede neural convolucional baseada na *MobileNetV2*, adaptada especificamente para a tarefa de reconhecimento de placas de veículos. A arquitetura proposta se especializa na segmentação da área da placa, diferente da arquitetura *MobileNetV2* original que classificava os objetos.
- **Implementação de técnicas avançadas de pré-processamento:** Foi introduzido métodos de pré-processamento de dados que incluem redimensionamento e normalização de imagens, além de estratégias de aumento de dados para lidar com a variabilidade nas imagens de placas de veículos.

- **Integração de funções de perda avançadas:** Foi avaliado duas funções de perda avançadas,*Generalized IoU (GIoU)* e *Complete IoU (CIoU)*, adaptadas para melhorar a precisão da detecção de objetos em termos de sobreposição de caixas delimitadoras.
- **Validação com conjunto de Dados Nacional:** foram realizados extensivos experimentos e validação utilizando o conjunto de dados *UFPR-ALPR*.

Por fim, com o desenvolvimento deste trabalho foi possível demonstrar que é viável a aplicação de técnicas de aprendizado por transferência para as tarefas de detecção e segmentação de placas de veículos em imagem.

5.3 TRABALHOS FUTUROS

Neste trabalho foi realizada uma pesquisa da literatura que mostrou que há várias abordagens que podem ser exploradas na tarefa de reconhecimento e segmentação de placas de veículos, como a utilização de outros modelos e até mesmo a aplicação de múltiplos modelos em cascata.

Como sugestão para trabalhos futuros, tem-se a utilização de outros modelos para detecção e segmentação de objetos em imagens. Um modelo que se destacou nas pesquisas realizadas é YOLO que atualmente se encontra na versão V8 e tem como diferencial trabalhar com a detecção em tempo real. Por fim, também fica como sugestão a utilização de outros *datasets* que tenham placas mais atuais e até mesmo com placas internacionais.

REFERÊNCIAS

AMAANULLAH, R. R. *et al.* Comparative transfer learning techniques for plate number recognition. In: **2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)**. [S.l.: s.n.], 2022. p. 112–116.

AMIDI, A.; AMIDI, S. **Dicas e Truques para Aprendizado de Máquina**. 2023. Acessado em: 18/09/2023. Disponível em: <https://stanford.edu/~shervine/l/pt/teaching/cs-229/dicas-truques-aprendizado-maquina#>.

BEZDEK, J. C. *et al.* **Fuzzy Models and Algorithms for Pattern Recognition and Image Processing**. [S.l.]: Springer, 2005. v. 4.

BEZERRA, E. Introdução à aprendizagem profunda. In: ____. [S.l.: s.n.], 2016. p. 57–86.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. 2020.

CAI, Y.; ZHANG, Y.; HUANG, J. American license plate recognition algorithm based on deep multi-model fusion. In: **2020 Chinese Automation Congress (CAC)**. [S.l.: s.n.], 2020. p. 6432–6437.

CHOPRA, R. **Machine Learning**. [S.l.]: KHANNA PUBLISHING HOUSE, 2021. 360 p. ISBN 9789386173423, 9386173425.

CONTRAN. **Resolução Nº 231, de 15 de Março de 2007**. 2007. Acesso em: 19 mai. 2024. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/conteudo-contran/resolucoes/resolucao_231.pdf>.

____. **RESOLUÇÃO Nº 748, DE 30 DE NOVEMBRO DE 2018**. 2018. Acesso em: 19 mai. 2024. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/conteudo-contran/resolucoes/resolucao7482018.pdf>.

DATA SCIENCE ACADEMY. **Deep Learning Book**. 2023. Disponível em: <https://www.deeplearningbook.com.br/>.

DENG, J. *et al.* Imagenet: A large-scale hierarchical image database. In: **2009 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2009. p. 248–255.

DEY, A. Machine learning algorithms : A review. In: . [s.n.], 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:40455026>.

DICIOÁRIO TEC. **Deep Learning**. 2022. Blog Post. Acesso em: 2023. Disponível em: <https://dicionariotec.com/posts/deep-learning>.

DU, S. *et al.* Automatic license plate recognition (alpr): A state-of-the-art review. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 23, n. 2, p. 311–325, 2013.

ELNASHAR, M.; HEMAYED, E. E.; FAYEK, M. B. Automatic multi-style egyptian license plate detection and classification using deep learning. In: **2020 16th International Computer Engineering Conference (ICENCO)**. [S.l.: s.n.], 2020. p. 1–6.

ESCOVEDO, T. Machine learning: Conceitos e modelos — parte i: Aprendizado supervisionado. **Medium**, Junho 2020. Disponível em: <https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-f0373bf4f445>.

FACELI, K. *et al.* **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FAYYAD, U. M.; AL. et. Knowledge discovery and data mining: Towards a unifying framework. In: **KDD**. [S.l.: s.n.], 1996. p. 82–88.

FAYYAD, U. M. *et al.* Automated analysis and exploration of image databases: Results, progress, and challenges. **Journal of Intelligent Information Systems**, v. 4, p. 1–19, 1995.

GANDHI, R. **Transposed Convolutional Neural Networks: How to Increase the Resolution of Your Image**. 2022. Disponível em: <https://towardsdatascience.com/transposed-convolutional-neural-networks-how-to-increase-the-resolution-of-your-image-d1ec27700c6a>

GATEVIDYALAY. **Machine Learning Workflow**. 2023. Disponível em: <https://www.gatevidyalay.com/machine-learning-workflow-process-steps/>.

GEVORGYAN, Z. SIoU loss: More powerful learning for bounding box regression. **arXiv preprint arXiv:2205.12740**, 2022.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOU, C. *et al.* Vehicle license plate recognition based on extremal regions and restricted boltzmann machines. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 4, p. 1096–1107, 2016.

Grupo Otimize. **Cores de placas de carros: saiba o que significam**. 2022. Disponível em: <https://grupootimiza.com.br/cores-de-placas-de-carros-saiba-o-que-significam>. Acesso em: [data de acesso].

HAPKE, H.; NELSON, C. **Building Machine Learning Pipelines**. O'Reilly Media, 2020. ISBN 9781492053149. Disponível em: <https://books.google.com.br/books?id=Ha_wDwAAQBAJ>.

HOWARD, A. *et al.* **Searching for MobileNetV3**. 2019.

HOWARD, A. G. *et al.* **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. 2017.

ImagiNet. **ImagiNet**. 2020. Disponível em: ://www.image-net.org. Acesso em: [data de acesso].

ISOLA, P. *et al.* **Image-to-Image Translation with Conditional Adversarial Networks**. 2018.

JAIN, S. *et al.* Machine learning-based real-time traffic control system. In: **2021 IEEE Mysore Sub Section International Conference (MysuruCon)**. [S.l.: s.n.], 2021. p. 92–97.

JAWALE, M. *et al.* Implementation of number plate detection system for vehicle registration using iot and recognition using cnn. **Measurement: Sensors**, v. 27, p. 100761, 2023. ISSN 2665-9174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2665917423000971>.

KADAM, K. D.; AHIRRAO, S.; KOTECHA, K. Efficient approach towards detection and identification of copy move and image splicing forgeries using mask r-cnn with mobilenet v1. **Computational Intelligence and Neuroscience**, Hindawi, v. 2022, p. 6845326, Jan 2022. ISSN 1687-5265. Disponível em: <https://doi.org/10.1155/2022/6845326>.

KARPATHY, A.; JOHNSON, J.; LI, F.-F. **Convolutional Neural Networks for Visual Recognition**. 2023. Disponível em: <https://cs231n.github.io/convolutional-networks/>.

KEIJSERS, N. Neural networks. In: KOMPOLITI, K.; METMAN, L. V. (Ed.). **Encyclopedia of Movement Disorders**. Oxford: Academic Press, 2010. p. 257–259. ISBN 978-0-12-374105-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780123741059004937>.

Keras. **Bounding Box API**. 2023. Acesso em: 19 maio 2024. Disponível em: <https://keras.io/api/keras_cv/bounding_box/>.

KERAS. **Keras Applications API Documentation**. 2023. Disponível em: <https://keras.io/api/applications/>.

____. **Keras Transfer Learning Guide**. 2023. Disponível em: <https://keras.io/guides/transfer_learning/>.

Laroca, R. *et al.* A robust real-time automatic license plate recognition based on the YOLO detector. In: **International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2018. p. 1–10. ISSN 2161-4407.

LAROCA, R. *et al.* A robust real-time automatic license plate recognition based on the yolo detector. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. IEEE, 2018. Disponível em: <http://dx.doi.org/10.1109/IJCNN.2018.8489629>.

LEE, K. **Automatic speech recognition: The development of the Sphinx system**. Boston: Kluwer Academic Publishers, 1989.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **Computer Science Department, University of British Columbia**, Vancouver, B.C., Canada, January 5 2004.

MADURI, P. K. *et al.* Smart eye of traffic management control system. In: **2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)**. [S.l.: s.n.], 2021. p. 794–798.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill, 1997.

OLIVAS, E. *et al.* **Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques**. Information Science Reference, 2009. ISBN 9781605667676. Disponível em: <https://books.google.com.br/books?id=gFpKXO8H_6YC>.

ONIM, M. S. H. *et al.* Blpnet: A new dnn model and bengali ocr engine for automatic licence plate recognition. **Array**, v. 15, p. 100244, 2022. ISSN 2590-0056. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2590005622000789>.

OPBROEK, A. van *et al.* Transfer learning improves supervised image segmentation across imaging protocols. **IEEE Transactions on Medical Imaging**, v. 34, n. 5, p. 1018–1030, 2015.

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010.

PHAN, V.-H.; HA, M.-Q.; DO, T.-H. A novel license plate image reconstruction system using generative adversarial network. In: **2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)**. [S.l.: s.n.], 2022. p. 222–228.

POMERLEAU, D. A. **ALVINN: An autonomous land vehicle in a neural network**. Pittsburgh, PA, 1989.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 6517–6525.

REZATOFIGHI, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. In: IEEE. **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.], 2019. p. 658–666.

RODRíGUEZ, J. **The Evolution of Google's MobileNet Architectures to Improve Computer Vision Models**. 2021. <https://medium.com/dataseries/the-evolution-of-googles-mobilenet-architectures-to-improve-computer-vision-models-ffb483ffcc0a>.

ROSEBROCK, A. **Transposed Convolutional Neural Networks: How to Increase the Resolution of Your Image**. 2018. <https://towardsdatascience.com/transposed-convolutional-neural-networks-how-to-increase-the-resolution-of-your-image-d1ec27700c6a>

SAMPAIO, R. F.; MANCINI, M. C. Estudos de revisão sistemática: um guia para síntese criteriosa da evidência científica. **Revista Brasileira de Fisioterapia**, São Carlos, v. 11, n. 1, p. 83–89, 2007.

SANDLER, M. *et al.* **MobileNetV2: Inverted Residuals and Linear Bottlenecks**. 2019.

SEVERIANO, M. *et al.* Segurança em redes 5g: Oportunidades e desafios em detecção de anomalias e predição de tráfego baseadas em aprendizado de máquina. In: **Minicursos do XXI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais**. [S.l.: s.n.], 2021. p. 145–189.

SHAFI, I. *et al.* License plate identification and recognition in a non-standard environment using neural pattern matching. **Complex Intelligent Systems**, v. 8, p. 3627–3639, October 2022.

SHINDE, P. P.; SHAH, S. A review of machine learning and deep learning applications. In: **2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)**. [S.l.: s.n.], 2018. p. 1–6.

SILVA, S. M.; JUNG, C. R. Real-time license plate detection and recognition using deep convolutional neural networks. **Journal of Visual Communication and Image Representation**, v. 71, p. 102773, 2020. ISSN 1047-3203. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1047320320300237>.

SZEGEDY, C. *et al.* Going deeper with convolutions. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2015. p. 1–9.

____. **Rethinking the Inception Architecture for Computer Vision**. 2015.

TESAURO, G. Practical issues in temporal difference learning. **Machine Learning**, v. 8, p. 257, 1992.

THAIPHUNG. Open Source Dataset, **greenParking Dataset**. Roboflow, 2023. <https://universe.roboflow.com/thaiphung/greenparking>. Visited on 2023-11-23. Disponível em: <https://universe.roboflow.com/thaiphung/greenparking>.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 3rd. ed. San Diego: Elsevier, 2006.

TIAN, Y. *et al.* Recent advances on loss functions in deep learning for computer vision. **Neurocomputing**, Elsevier, v. 470, p. 346–361, 2022.

UPADHYAY, U. *et al.* Analysis and architecture for the deployment of dynamic license plate recognition using yolo darknet. In: **2019 International Conference on Power Electronics, Control and Automation (ICPECA)**. [S.l.: s.n.], 2019. p. 1–6.

VESTONI, E. **Uma Introdução aos Diferentes Tipos de Convoluções**. Medium, 2023. Acessado em 20/11/2023. Disponível em: <https://estevestoni.medium.com/uma-introduÃğÃčo-aos-diferentes-tipos-de-convoluÃğÃμes-d3ce7cd81759>.

WAIBEL, A. *et al.* Phoneme recognition using time-delay neural networks. **IEEE Transactions on Acoustics, Speech and Signal Processing**, v. 37, n. 3, p. 328–339, 1989.

WEISS, K.; KHOSHGOFTAAR, T. M.; WANG, D. A survey of transfer learning. **Journal of Big Data**, Springer, v. 3, n. 1, p. 9, 2016.

YOSINSKI, J. *et al.* **How transferable are features in deep neural networks?** 2014.

YU, J. *et al.* Unitbox: An advanced object detection network. In: ACM. **Proceedings of the 24th ACM international conference on Multimedia**. New York, NY, USA, 2016. p. 516–520.

ZHANG, A. *et al.* **Dive into Deep Learning**. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>.

ZHANG, L.; WANG, S.; LIU, B. **Deep Learning for Sentiment Analysis : A Survey**. 2018.

ZHENG, Z. *et al.* Enhancing geometric factors in model learning and inference for object detection and instance segmentation. **arXiv preprint arXiv:2005.03143**, 2020.

ZOPH, B. *et al.* **Learning Transferable Architectures for Scalable Image Recognition**. 2018.

ANEXO A – CÓDIGOS PARA PŔE-PROCESSAMENTO DOS DADOS

Algoritmo 2 – Funções em Python para redimencionar as imagens

```
1  import numpy as np
2  import cv2 # OpenCv
3  import math
4
5  def fazer_quadrado(img):
6      """
7      Função para deixar uma imagem quadrada acresentando pixeis pretos
8      Parâmetros:
9      - img: numpy.ndarray[m, n, 3]
10     Retorna:
11     - numpy.ndarray com dimenção [x, x, 3] onde x in  [m,n]
12     """
13     # Obtendo o lado maior da imagem
14     s = max(img.shape[0:2])
15     # Criando um quadrado escuro com NUMPY
16     f = np.zeros((s,s,3),np.uint8)
17     # Copia a imagem para o quadrado
18     f[0:img.shape[0],0:+img.shape[1]] = img
19     return f
20
21 def pre_processar_imagem(img, dst_image_path, INPUT_WIDTH=224):
22     """
23     Função para prepara a imaem e gravar em um novo destino
24     Parâmetros:
25     - img: numpy.ndarray[m, n, 3]
26     - dst_image_path: str, cainho onde deve gravar a imagem
27     Retorna:
28     - a taxa/porcentagem de redimencionamento
29     """
30     hh, ww, dd =  img.shape
31     s = max(ww, hh)
32     # calcula a taxa de redimencionamento
33     tax = INPUT_WIDTH/s
34     # altera o tamanho da imagem
35     img = cv2.resize(img, (math.ceil(ww*tax), math.ceil(hh*tax)), cv2.
       INTER_AREA)
36     img = fazer_quadrado(img)
37     # grava no destio
38     cv2.imwrite(dst_image_path ,img)
39     return tax
```

Fonte: O Autor (2024)

Algoritmo 3 – Funçoes em Python para extração da anotação da placa

```python
import math

def get_anotacao_placa(anot_file, tax, linha_lp=7):
    """
    Função para para extrair a anotação da placa

    Parâmetros:
    - anot_file: _io.TextIOWrapper, Arquivo de anotaçoes
    - tax: a taxa/porcentagem de redimencionamento da imagem

    Retorna:
    - (x1, y1, x2, y2): cordenadas no formato XYXY
    """
    # pega todas as linhas
    Lines = anot_file.readlines()
    # obtem da linha 7 as coordenadas no formato xyWH
    x, y, w, h = Lines[linha_lp].strip().split(":")[1].strip().split(" ")
    #  converte as coordenadas para o novo tamanho da imagem
    nh, nw = math.ceil(int(h) * tax), math.ceil(int(w) * tax)
    nx, ny = math.ceil(int(x) * tax), math.ceil(int(y) * tax)

    # converte as anotaçoes para cordenadas para o formato XYXY
    x1, y1, x2, y2 = nx, nx + nw, ny, ny + nh
    return (x1, y1, x2, y2)
```

Fonte: O Autor (2024)

Algoritmo 4 – Função para aumentar o data-sete

```python
import math

def enquadrar_veiculo_nova_img(anot_file, img, dest_veicle_img,
    linha_veicle=1, linha_lp=7):
    """
    Cria auma nova imagem enquadrando o veiculo e retorna a notação da placa
      nessa imagem
    Parâmetros:
    - anot_file: _io.TextIOWrapper, Arquivo de anotaçoes
    - img: numpy.ndarray[m, n, 3], Imagem para recortar
    - dest_veicle_img: str, caminho para gravar a imagem
    Retorna:
    - (x1, y1, x2, y2): coordenadas da placa
    """
    # pega todas as linhas do arquivo de anotação
```

```
    Lines = anot_file.readlines()
    # seleciona a linha da placa e obtem as coordenadas no formato xyWH
    x, y, w, h = Lines[linha_lp].strip().split(":")[1].strip().split(" ")
    x, y, w, h = int(x), int(y), int(w), int(h)
    # seleciona a linha do veículo e  obtem as coordenadas no formato xyWH
    vx, vy, vw, vh = Lines[linha_veicle
                           ].strip().split(":")[1].strip().split(" ")
    vx, vy, vw, vh = int(vx), int(vy), int(vw), int(vh)
    #  ajusta as cordenadas da placa como referencia ao veículo
    x = x - vx
    y = y - vy
    # recorta o veículo dentro da imagem
    img = img[vy:vy+vh, vx:vx+vw, :]
    # redimencionar imagem para 224x224
    tax = pre_processar_imagem(img, dest_veicle_img)
    #  converte as coordenadas para o novo tamanho da imagem
    nh, nw = math.ceil(h * tax), math.ceil(w * tax)
    nx, ny = math.ceil(x * tax), math.ceil(y * tax)
    # converte as anotaçoes para cordenadas para o formato XYXY
    x1, y1, x2, y2 = nx, nx + nw, ny, ny + nh
    return (x1, y1, x2, y2)
```

Fonte: O Autor (2024)

Algoritmo 5 – Código em Python para pré-processar o data-sete

```
import numpy as np
import pandas as pd
import cv2 # OpenCv
import shutil
import math
import os

def processar(pathorigin, pathdest):
    # cria um data frame para guaradar as anotaçoes
    df = pd.DataFrame(columns=['filepath','xmin','xmax','ymin','ymax'] )
    nivel = 0
    for (dirpath, dirnames, filenames) in os.walk(pathorigin):
        #  descarta primeiro nível
        if nivel == 0:
            nivel +=1
            continue
        #  para cada arquivo
        for file in filenames:
            # segue se for uma imagem
            if not file.endswith(".png"):
                  continue
            # Monta o arquivo de destino
```

```
23            dst_image_path = os.path.join(pathdest,file)
24            # Monta o arquivo de origem
25            origin_image_path = os.path.join(dirpath, file)
26            # Arquivo de anotaçoes
27            anot_file_name = os.path.join(dirpath, file.split(".")[0]+".txt")
28            # se o arquivo de anotaçoes existe
29            if not os.path.exists(anot_file_name):
30                continue
31            # utiliza a função imread do pacote OpenCv
32            # para carregar a imagem em momória
33            img = cv2.imread(origin_image_path)
34            #  processa a imagem e retorna a taxa de redimencionamento
35            tax = pre_processar_imagem(img, dst_image_path)
36            # abre o arquivo de anotação
37            anot_file = open(anot_file_name, 'r')
38            #  obtem as anotaçoes da placa já redimencionadas
39            x1, y1, x2, y2 = get_anotacao_placa(anot_file, tax)
40            # gurada a anotação e o destino da imagem em um data-frame
41            df.loc[len(df)]=  ar = np.array([dst_image_path, x1, y1, x2, y2])
42
43            dest_veicle_img = dst_image_path[0:-4]+"_2"+dst_image_path[-4:]
44            #  aumenta a base criando uma nova imagem do enquadramento do veí
      culo
45            x1, y1, x2, y2 = enquadrar_veiculo_nova_img(anot_file, img,
46                                                        dest_veicle_img)
47            # gurda a anotação e o destino da imagem em um data-frame
48            df.loc[len(df)]=  ar = np.array([dest_veicle_img, x1, y1, x2, y2])
49
50      #  exporta o data-frame para um arquivo CSV
51      dest_csv_file = os.path.join(pathdest, "labels.csv")
52      df.to_csv(dest_csv_file, index=False )
53
54  # apaga a pasta de destino e todos seus os arquivos
55  if os.path.exists("./data_images"):
56       shutil.rmtree('./data_images/')
57  os.makedirs("./data_images")
58  os.makedirs("./data_images/test")
59  os.makedirs("./data_images/train")
60  os.makedirs("./data_images/valid")
61
62  processar("./images/validation/", './data_images/valid')
63  processar("./images/testing/"   , './data_images/test' )
64  processar("./images/training/"  , './data_images/train')
```

Fonte: O Autor (2024)

Algoritmo 6 – Normalizar os dados de imagens e coordenadas no intervalo de 0 a 1

```
1
2 from tensorflow.keras.preprocessing.image import load_img, img_to_array
3 import numpy as np
4 import pandas as pd
5
6 def carregar_ds(csv_path, TARGET_SIZE=(224,224) ):
7     '''
8     Carrega as imagem e anotaçoes e normaliza para ao intervalo [0,1].
9     Parâmetros:
10    - csv_path: caminho para o CSV de anotaçoes
11    Retorna:
12    - (X, Y): Imagem e Dominio, onde X são as imagens de carros
13        e Y são as anotaçoes
14    '''
15    #  carrega o CSV para um data-frame
16    df_train = pd.read_csv(csv_path)
17    #  separar as coordenadas
18    labels = df_train.iloc[:,1:].values
19    #  separar os caminhos das imagens
20    lst_images_path = list(df_train['filepath'])
21    X, Y = [], []
22    for ind in range(len(lst_images_path)):
23        image_path = lst_images_path[ind]
24        #  carrega a imagem com o OpenCv
25        img_arr = cv2.imread(image_path)
26        # Obtem as dimencoes da image,
27        h, w, _ = img_arr.shape
28        # Carrega a imagem com a funcao load_img do modulo tensorflow
29        load_image = load_img(image_path,target_size=TARGET_SIZE)
30        # Converter imagem para array com img_to_array do modulo tensorflow
31        load_image_arr = img_to_array(load_image)
32        # Normalizar para o intervalo [0,1]
33        norm_load_image_arr =  load_image_arr/255.0
34        # Normaliza as coordenadas para o intervalo [0,1]
35        xmin,xmax,ymin,ymax = labels[ind]
36        nxmin,nxmax = xmin/w,xmax/w
37        nymin,nymax = ymin/h,ymax/h
38
39        label_norm = (nxmin, nymin,nxmax,nymax)
40
41        X.append(norm_load_image_arr)
42        Y.append(label_norm)
43    return (X, Y)
```

Fonte: O Autor (2024)