PROTÓTIPO DE SISTEMA DE VIGILÂNCIA E CONTROLE DE ACESSO DE VEÍCULOS UTILIZANDO VISÃO COMPUTACIONAL

Thiago O. Barbosa[1], Antônio Carlos Fava de Barros[1]

[1]Ciência da Computação -- Universidade Federal de Viçosa Campus Florestal (UFV-CEDAF)
Rodovia LMG 818, km 06 – 35690-000 – Florestal – MG – Brasil

{thiago.o.barbosa,antonio.fava}@ufv.br

Resumo. *Até algumas décadas atrás, os veículos automotores não eram acessíveis a maioria da população brasileira, porém, nos dias atuais são uma realidade na casa de muitas pessoas. E são produtos visados tanto para roubo, como também são utilizados para praticar os mesmos. Com este aumento crescente dos veículos em circulação, e a sua utilização para furtos surgiu também o interesse em monitorar estes veículos, seja para controle de tráfego, gestão de veículos em estacionamentos, captura de infratores e outras atividades. Pensando nisso, este trabalho propõe o desenvolvimento de um algoritmo que seja capaz de reconhecer as placas de trânsito de veículos em imagens (fotos) e vídeos, e extrair seus caracteres, permitindo uma análise sobre os dados. O objetivo é que a partir dos resultados deste trabalho, seja possível desenvolver uma aplicação real capaz de realizar o controle de fluxo de veículos, como uma garagem, estacionamento ou sistema de vigilância. Para isto foi utilizado o modelo YOLO para a rede neural convolucional, treinada com dataset e aplicada a vídeos feitos no trânsito real. Os resultados apontam um modelo capaz de localizar placas com 90% de precisão e uma extração de caracteres da placa com 50% de sucesso.*

1. Introdução

Segundo o IBGE [Ministério da infraestrutura 2020], a frota brasileira de veículos subiu de 45 milhões de veículos em 2006 para 107 milhões em 2020. Estes veículos são utilizados para lazer, transporte de pessoas e cargas, mas também podem ser utilizados para realização de furtos e transporte de mercadorias ilícitas e outras atividades ilegais. Também são objetos de furtos, que podem ocorrer durante o trânsito ou quando estão estacionados, incentivando assim pesquisas na área de vigilância e controle de propriedade.

A computação é uma área que oferece uma infinidade de possibilidades, e está em constante avanço, criando soluções para melhorar a vida e bem-estar das pessoas e empresas. Dentre as áreas que estão em desenvolvimento, temos a área de Processamento Digital de Imagens, que associado ao campo da Inteligência Artificial, permite a identificação de objetos, rastreamento e outras possibilidades. Através do uso de uma *Deep Neural Network* (DNN ou rede neural profunda), foram desenvolvidos uma série de trabalhos onde as máquinas foram capazes de reconhecer objetos em imagens ou até

mesmo gerar obras de arte, utilizando-se do recurso de *artistic style transfer* [Leon A. Gatys 2021]. Este trabalho apresenta um dos recursos da utilização das DNN, com um modelo para a identificação de placas de veículos e posterior extração dos caracteres da mesma através de um OCR para fins diversos (consulta, armazenamento, controle de tráfego e etc.).

2. Referencial Teórico

O reconhecimento de objetos em imagens é um tema ainda em desenvolvimento. Uma série de algoritmos já foram desenvolvidos, dentre eles as redes neurais artificiais (RNA). Segundo [Haykin et al. 2009], as RNA são formadas por conjunto de neurônios conectados, interligados em camadas, por onde flui a informação de entrada até uma camada de saída. Nestas camadas, temos os pesos, que são parâmetros a serem ajustados de tal forma que a informação aplicada na entrada da rede resulte uma saída desejada. Neste trabalho, será utilizada uma variação das RNA, denominadas redes neurais convolucionais (CNN), explanada por [Chollet, 2017]. O autor apresenta as CNN contendo quatro principais camadas, sendo que destacaremos apenas as camadas convolucionais e de *Max-pooling*.

Segundo o autor, a camada convolucional tem a função de extrair as características da imagem de entrada, através da operação de convolução de uma máscara com a imagem original, semelhante ao processo de filtros digitais, citados por [Gonzalez, Woods E Eddins, 2010]. Este processo de convolução está apresentado em [Silva, 2018]. Já a camada de *pooling* tem a função de reduzir a dimensionalidade dos mapas de características resultantes das camadas anteriores, diminuindo assim o número de parâmetros para a próxima camada, e, consequentemente, o custo computacional para o treinamento e uso da rede. A figura 1 apresenta um esquema simplificado de uma CNN, apresentando as camadas de convolução.

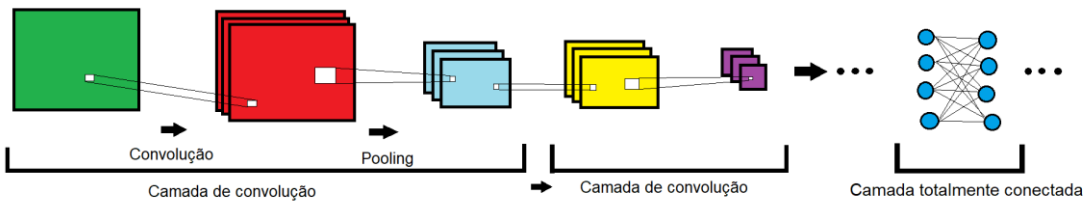

Figura 1. Camadas de uma rede neural convolucional [Saad Albawi 2017].

Como as camadas convolucionais são utilizadas para extrair características das imagens, e a camada *fully-connected* é responsável pela classificação dos objetos, as CNN podem aproveitar os pesos obtidos durante a etapa de treinamento com os dados da ImageNet, disponibilizados pelos autores da YOLO, e o pesquisador pode apenas treinar a camada de reconhecimento com seu próprio banco de imagens, através da técnica conhecida como *transfer learning*. A ideia central do *transfer learning* é aproveitar o aprendizado de uma rede já treinada, para a partir desta retreinar somente algumas camadas para dados novos, a fim de adquirir um conhecimento novo com menos esforço, conforme citado por [Karl Weiss, 2016] e [Daniel De Los Reyes, 2019].

A utilização das CNN incentivou a criação de competições de performance, como a *ImageNEt Large Scale Visual Recognition Challenge* (ILSVRC), e diversas arquiteturas de redes têm sido testadas desde então, como a VGG, VGG-19, ResNet, a Inceptron (da Google), e o YOLO. Um comparativo entre alguns destes modelos, abordando métricas de comparação e o funcionamento básico de cada um foi apresentado por SILVA [2018]. Dentre as redes estudadas, a que apresentou melhor acurácia foi a ResNet50, com 93.4%.

O método YOLO foi apresentado por [Redmon et al. 2016], e segundo seu autor, a rede é rápida o bastante para identificar objetos em imagens em aplicações de tempo real, com uma média de precisão considerada alta quando comparada com outras arquiteturas, atingindo a performance de 88% no ILSVRC de 2012, o que incentivou o seu uso neste trabalho, onde foi utilizado o modelo YOLOv4 para o reconhecimento e extração de placas de veículo, e o software Tesseract [*Google and Community,* 2022] para extração dos caracteres das placas reconhecidas pela rede. Para isto o trabalho foi divido entre as seguintes etapas: aquisição do *dataset*; preparação do *dataset*; configuração da rede para o YOLOv4; treinamento e teste. Por fim, a placa de veículo extraída pela rede foi pré-processada, com aplicação de filtros digitais e operações morfológicas e repassada para o Tesseract para a extração dos caracteres.

As seções deste trabalho podem ser divididas em: 1. Introdução; 2. Referencial Teórico; 3. Trabalhos Relacionados; 4. Metodologia; 5. Resultados; 6. Conclusão; 7. Trabalhos Futuros.

3. Trabalhos Relacionados

O trabalho desenvolvido por YONTEN JAMTSHO [2021] propõe um modelo para reconhecimento de placas de motocicletas, com o objetivo de capturar aqueles cujos condutores e/ou passageiros não se encontram em regularidade com relação ao uso de capacetes. O objetivo é utilizar uma única rede neural capaz de reconhecer tanto o capacete quanto a placa. Caso seja detectado o não uso do capacete, é extraída a placa e enviada a uma rede neural para extração dos caracteres. Neste trabalho foi utilizado a versão do YOLOv2 para a rede neural, o Opencv para automação do algoritmo e a métrica do F1-*score* para mensurar a precisão do modelo. Ao final o melhor modelo foi treinado por 7.000 *epochs* e obteve uma precisão de 74.72%.

Já IMAMURA ET. AL. [2021] desenvolveu uma aplicação para detecção e extração de caracteres de placas de licenciamento veicular. Para isto utilizou o YOLO, juntamento do framework Darknet para detecção das placas e reconhecimento dos caracteres. As imagens foram pré-processadas, com etapas de equalização e binarização, para melhor identificar os caracteres. Neste artigo o autor explica também os casos onde não foram possíveis a extração dos caracteres devido às condições geradas em função da detecção em tempo real, visto que várias imagens foram tomadas em ambientes não controlados, o que dificulta a identificação dos caracteres.

Outro trabalho para identificação de placas de veículos em tempo real foi proposto por LEITE [2017], com o objetivo de ser colocado na cancela de entrada do Instituto Federal Catarinense. Para a identificação das placas foi utilizado o método conhecido como *Haar Cascade*, e para a identificação dos caracteres foi utilizado a biblioteca Pytesseract. Se tratando de uma aplicação de tempo real, o autor conseguiu apresentar os

resultados pela própria webcam, identificando na imagem as placas movidas com a mão em frente a câmera e extraindo os caracteres, obtendo 93,54%. O alto valor obtido pode ser explicado pelo ambiente controlado em que as imagens foram obtidas das placas.

4. Metodologia

O objetivo é desenvolver um sistema capaz de executar de maneira rápida e eficiente a identificação e extração de placas de carros em veículos e aplicar estas placas a um software de reconhecimento de caracteres, o Tesseract. Para realizar esta tarefa foi necessário a execução de algumas etapas que serão melhor descritas nesta sessão. A metodologia aqui utilizada consiste em: aquisição do *dataset* de treinamento, que já é fornecido com seus rótulos, a conversão da notação destes rótulos para o modelo YOLO, preparação do *dataset*, e treinamento/validação da rede para o YOLOv4. Após este processo, o modelo será utilizado para detectar e fornecer as coordenadas para a extração da imagem da placa na imagem original. Como as imagens utilizadas foram obtidas e ambientes reais, apresentam diversos ruídos na aquisição, como placas manchadas, escuras e outras. Para minimizar estes problemas foram utilizadas 3 camadas de pré-processamento das imagens, com aplicação de filtros, segmentação e operações morfológicas. Após esta etapa, a imagem foi aplicada ao aplicativo Tesserat, para reconhecimento e extração dos caracteres.

Foram utilizados os seguintes recursos para estas operações: *dataset* obtidos do Kaggle e da plataforma OIDv4 ToolKit, python, Opencv, *framework* Darknet, o software Tesseract 5.0.0 para OCR e o ambiente Google Colab para desenvolvimento.

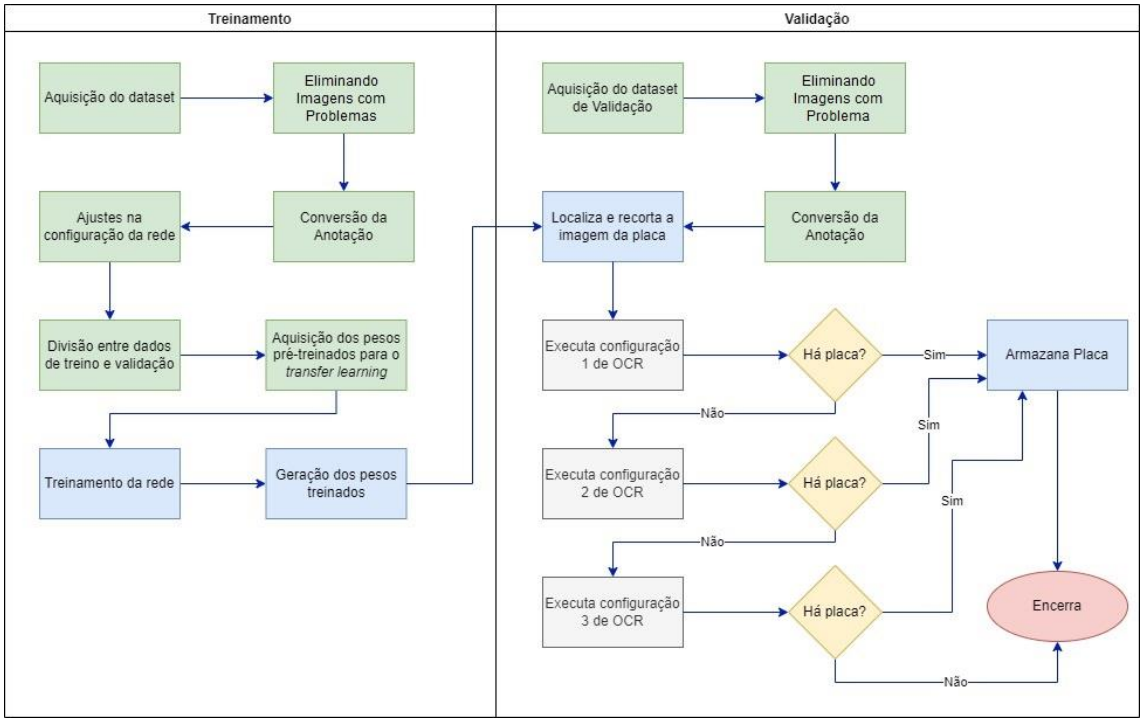

Figura 2. Diagrama de fluxo que representa a metodologia.

4.1. Preparação do banco de imagens

Para a rede neural foi utilizada a arquitetura do YOLOv4, com um banco de imagens formado pela plataforma Kaggle [Bjarki, 2022] e também imagens fornecidas pela plataforma "OIDv4 ToolKit" [Vittorio, 2018]. Foram selecionadas 500 imagens do Kaggle e 432 imagens do OIDv4, totalizando 932 imagens. Porém, realizando uma inspeção manual das imagens, foram excluídas imagens repetidas ou aquelas que não apresentavam corretamente as placas de veículos ou até imagens que não possuíam placas de veículos e sim adesivos em veículos, reduzindo para um total de 671 imagens.

As imagens utilizadas possuem arquivos de notação informando a área na imagem onde se encontra o objeto da busca, porém o YOLO possui uma notação própria que é armazenada em arquivos de extensão ".txt" no formato: "number_of_class x_center y_center width height". Contudo os arquivos do Kaggle são fornecidos com uma notação diferente, em formato xml, e os arquivos do segundo dataset são fornecidos com uma notação seguindo o formato: "name_of_the_class x_left y_top x_right y_bottom", foi necessária a conversão das mesmas para o padrão YOLO.

4.2. Treinamento da rede para o YOLOv4

Neste trabalho foi realizado o treinamento com a arquitetura do YOLOv4, utilizando a ferramenta Google Colab e o *framework* Darknet. Para realizar este treinamento, o arquivo de configuração da rede foi ajustado conforme a tabela 1, onde foram editados os seguintes parâmetros do arquivo de configuração: *subdivisions*, *max_batches*, *steps*, *classes* e *filters*. Os valores assumidos para cada um deles podem ser visualizados na tabela 1.

Subdivisions	Valor
Match-batches	64
Steps	2000
Filters	1800,2200
Classes	18
Subdivisions	1

Tabela 1. Parâmetros ajustados para configuração da rede YOLO.

Após esta configuração, o próximo passo foi a importação do arquivo de pesos de um modelo já treinado. Para isto foi utilizado o arquivo disponível encontrado no próprio site do Darknet [Redmon 2016], treinado a partir do *dataset* ImageNet. O último passo é treinamento da rede neural para o banco de imagens formado pelas imagens deste trabalho, relatados na seção 4.1. Para realizar esta tarefa foi utilizada o *framework* Darknet.

4.3. **Pré-processamento das imagens e extração dos caracteres**

Com a rede treinada, a próxima etapa foi a utilização da mesma para a identificação e extração das placas dos veículos. Apresentando uma nova imagem à rede, a saída desta fornece um arquivo contendo os rótulos, que são coordenadas do objeto reconhecido na imagem. Estas coordenadas foram utilizadas para extrair as placas das imagens, para posterior reconhecimento dos caracteres pelo Tesseract.

Como as placas utilizadas neste trabalho não foram obtidas em um ambiente controlado, são necessários alguns pré-processamentos na imagem, como filtro de média, binarização, dilatação e reconhecimento de borda. Estas operações permitiram eliminar ruídos na imagem, bem como destacar o objeto de interesse (os caracteres das placas).

Estes 4 processamentos apresentaram bons resultados visuais no melhoramento da imagem. Contudo isso varia do estado em que se encontra a imagem. Placas muito limpas e placas muito manchadas ou enferrujadas requerem tratamentos diferentes. Por isso foram realizados os seguintes ajustes: limiar da segmentação e área da dilatação dos tons de branco, conforme a tabela 2.

Limiar (0 a 255)	Dilatação (nº de pixels)	Qualidade da Imagem
100	1	Boa
70	5	Com ruído
70	10	Com muito ruído

Tabela 2. Parâmetros de configuração do OCR.

É possível notar que em ambos os casos os caracteres foram devidamente reconhecidos com a configuração 3, contudo muitas vezes a configuração 3, na tentativa de eliminar ruídos, acaba eliminando características importantes dos caracteres, dificultando a identificação pelo Tesseract.

Para melhorar a qualidade e desempenho do algoritmo, em cada placa encontrada, as configurações são aplicadas em ordem crescente e o Tesseract é executado em cada uma delas. Quando o número de caracteres encontrados está no do padrão das placas de licenciamento do país, o algoritmo cancela a execução das outras configurações e vai para a próxima placa. Em cada configuração, o Tesseract é executado 4 vezes, sendo uma vez após cada processamento realizado na imagem, ou seja, após o limiar, o filtro de média, a dilatação e o reconhecimento de borda. Esta medida melhora a qualidade, pois há alguns casos onde o excesso de processamento dificulta o reconhecimento dos caracteres. Também melhora o desempenho, pois se a placa for reconhecida nos primeiros processamentos, ela não executa para os próximos, economizando tempo e custo de processamento.

4.4. **Execução em vídeos e validação do modelo**

O objetivo final é a execução deste algoritmo em vídeos, de forma que possa ser executado em câmeras de segurança. O algoritmo desenvolvido lê os *frames* do vídeo, que são aplicados à rede já treinada. Após identificada a placa, o algoritmo só extrai os caracteres caso ele encontre algum objeto na imagem. Além disso, quando o algoritmo realiza a extração dos caracteres, para cada uma das configurações implementadas, ele as executa em ordem crescente até encontrar uma sequência de caracteres que satisfaça a quantidade de caracteres das placas, sejam o modelo antigo ou o modelo novo, adotado no Mercosul. Caso não seja satisfeito, ele não executa as outras configurações de pré-processamento. Para agilizar este processamento, o algoritmo realiza um salto entre 5 *frames* consecutivos, para evitar que uma mesma imagem seja processada diversas vezes.

As imagens de vídeo foram obtidas utilizando um *smartphone* Samsung Galaxy, modelo M30, com imagens tomadas durante o trânsito na cidade de Juatuba, Minas Gerais, sendo imagens que apresentam trepidação, tomadas de placas em diferentes ângulos, reflexões de luz e outras situações em que são gerados ruídos na imagem. Porém, o projeto prevê a adoção em câmeras de segurança, posicionadas em cancelas de estacionamento ou portarias, em que o ângulo e distância sejam determinados para melhor aquisição, e com automóvel parado ou em velocidade reduzida.

Estas configurações de pré-processamento podem ter alto custo computacional, ainda mais que ao final de cada uma delas (e também durante, já que são executados após cada processamento) é executado o processo de reconhecimento de caracteres com o Tesseract.

O último passo foi a validação da qualidade do modelo. Foi utilizada a métrica mAP (*mean Average Precision*), que realiza o cálculo da média de todas as precisões do modelo (*recall* e *precision*) para encontrar o objeto na imagem, ou seja, as precisões com que foram detectadas as placas de carro.

A partir destes valores, que levam em consideração os verdadeiros positivos (TP), falso positivos (FP) e falso negativos (FN), que o AP é calculado. Para cada limiar, o algoritmo multiplica a diferença entre os recalls (entre o recall para um limiar e outro) pela precisão [Etten 2019]. A fórmula que representa a equação do AP é:

$$AP = \sum_{k=0}^{k=n-1} \big(recall(k) - recall(k-1)\big) * precision(k)$$

Sendo *AP* o valor da métrica, *n* o número de valores limiar utilizados no cálculo e *recall(k)* e *precision(k)* os respectivos valores de *recall* e *precision* pra cada valor de limiar. Este cálculo é realizado para cada *label* treinado na rede. A média dos APs de cada rótulo é conhecido como mAP.

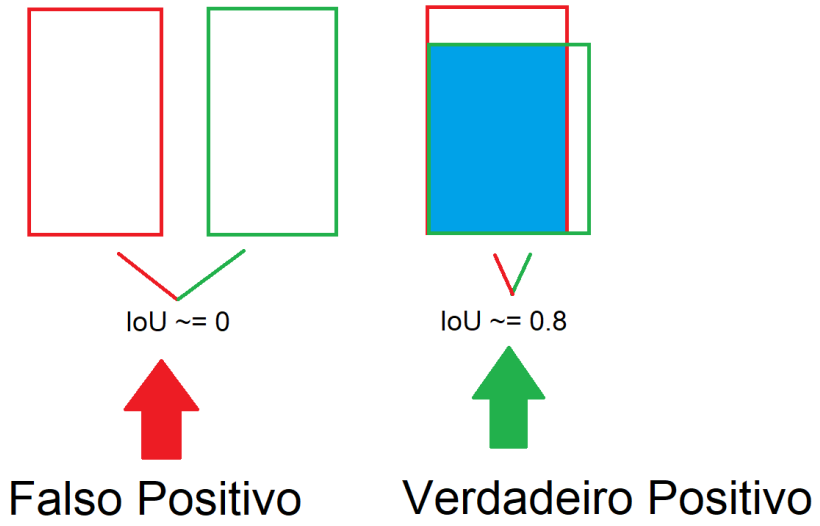

Figura 3. Representação do IoU para um threshold igual a 0.5 [Yohanandan 2020].

5. Resultados

Os resultados de validação do algoritmo demonstram uma qualidade satisfatória, sendo capaz de reconhecer placas de trânsito em ambientes não controlados e sobre condições até mesmo adversas com 100% de precisão e com *recall* de 93% (ambos com valores diferentes de limiar*)*. Para validar a qualidade do modelo foi utilizado o valor da métrica mAP. Ao todo foram separadas 64 imagens, que foram submetidas ao *IoU* com limiar de 0.5 para determinar se a caixa delimitadora foi capaz de prever corretamente a área da placa. Para encontrar os melhores valores de limiar (dessa vez o limiar do rótulo da rede neural) para comparação, o algoritmo calculou o *recall*, *precision* e *f1 score* para 13 diferentes valores de limiar estando eles entre [2.0, 8.0] com 0.05 de intervalo entre os valores. Os gráficos nas figuras 4, 5 e 6 mostram os resultados desses cálculos.

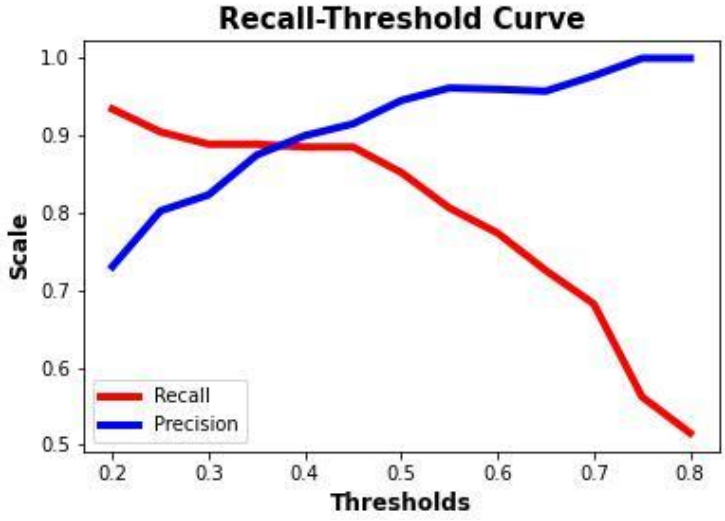

Figura 4. Variações dos valores de *Recall* e *Precision* pelo limiar

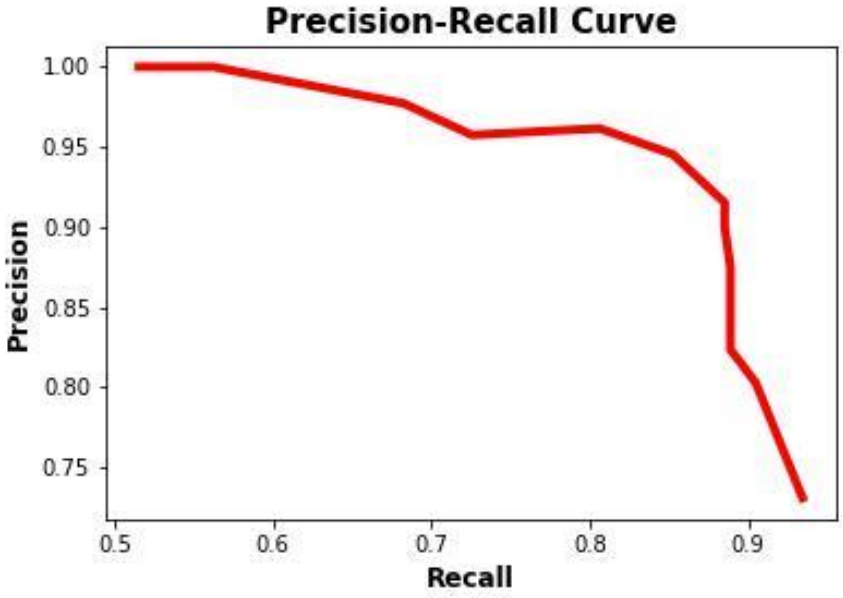

Figura 5. Variação do F1 Score em função do limiar.

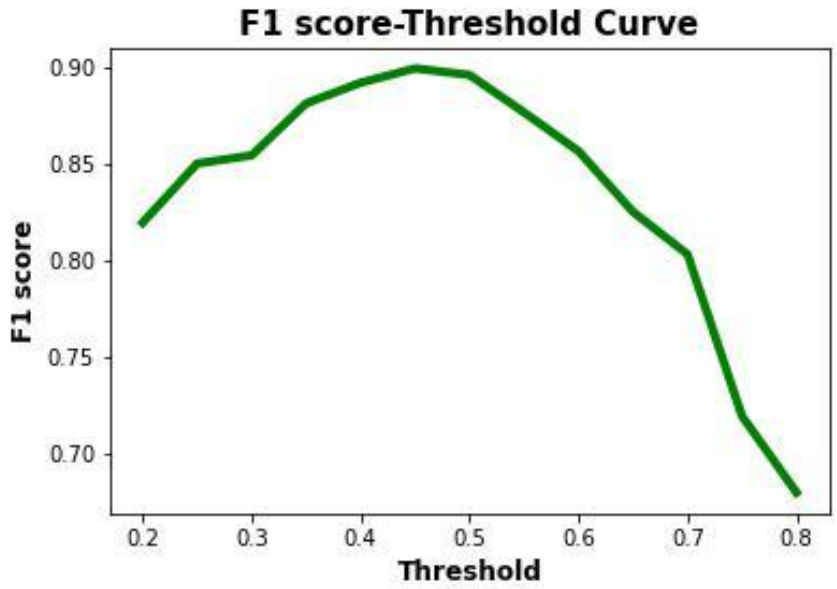

Figura 6. Valores Precisão por *recall*.

A figura 4 mostra as variações do recall e da precisão em função dos valores de limiar nos testes de validação. É possível notar que à medida que se aumenta o limiar, aumenta-se também a precisão, mas o recall cai. Já a figura 5 ela mostra a curva da precisão pelo recall, cuja área abaixo dela representa o mAP. Quanto maior este valor, melhor é a qualidade do modelo, já que ele balanceia bem os valores de recall e precisão. Para o cálculo do mAP, a própria Darknet realiza este cálculo durante o treinamento, alcançado o valor de 76,74% para as 170 imagens de validação (os 25% separados antes

de realizar o treinamento). Na figura 6 é possível notar os valores de f1 *score*, que demonstrou ser muito bom, tendo o seu maior valor igual a 0,9. Este número representa o melhor equilíbrio entre o *recall* e a *precision*, informando também o melhor valor de limiar para se trabalhar, sendo este igual a 0,45. Estes valores serão melhor apresentados na tabela 3.

Figura 7. Exemplo de um verdadeiro positivo

Figura 8. Exemplo de um falso positivo.

As figuras 7 e 8 mostram exemplos de verdadeiro positivo e falso positivo. Os retângulos na imagem representam a caixa delimitadora real (em azul) e a caixa delimitadora predito pelo algoritmo (em vermelho). É possível notar pela figura 7 que a

caixa delimitadora da predição é relativamente semelhante a caixa delimitadora real, contudo ele pode confundir determinados elementos com placa conforme mostrado na figura 8.

A partir da variação do limiar foi possível coletar alguns valores sobre a qualidade do modelo gerado. Abaixo é apresentada uma tabela que mostra os valores de precisão, *recall*, *f1 score*, TP, FP e FN para cada valor de limiar.

Limiar	Verdadeiro Positivo	Falso Positivo	Falso Negativo	Precisão	*Recall*	*F1 score*
0,2	57	21	4	0,73	0,96	0,82
0,25	57	14	6	0,80	0,90	0,85
0,3	56	12	7	0,82	0,89	0,85
0,35	56	8	7	0,88	0,89	0,88
0,4	54	6	7	0,9	0,89	0,89
0,45	54	5	7	0,92	0,89	0,90
0,5	52	3	9	0,95	0,85	0,90
0,55	50	2	12	0,96	0,81	0,88
0,6	48	2	14	0,96	0,77	0,86
0,65	45	2	17	0,96	0,73	0,83
0,7	43	1	20	0,98	0,68	0,80
0,75	36	0	28	1	0,56	0,72
0,8	33	0	31	1	0,52	0,68

Tabela 3. Variação nos valores da matriz de confusão, f1 score, recall e precisão, com cada valor de limiar.

Para cada um destes valores de limiar, a rede neural realizou a validação com as 64 imagens escolhidas manualmente. É possível notar que à medida que aumenta o Limiar, diminui o número de falsos positivos, explicando assim o aumento na precisão, contudo também aumenta o número de falsos negativos, diminuindo assim o *recall*. O ideal é balancear bem os dois valores, conforme mostram os valores do *f1 score*, que

apresenta o limiar de 0,45 como o melhor.

Para executar estes algoritmos foi utilizada uma máquina com as seguintes configurações: processador ryzen 7 modelo 4800H, 16 GB de RAM e uma placa de vídeo NVIDIA GeForce GTX 1650 Ti, com um tempo médio para extração das imagens foi de 0,2 segundos de extração. Já a medição dos tempos de realização do reconhecimento dos caracteres das placas resultou um valor médio de 0.4 segundos. Além disso, com o excesso de pré-processamento, as características da placa podem ser alteradas, gerando resultados semelhantes, contudo diferentes, conforme mostra figura 9.

Figura 9. Dois carros diferentes tiveram duas numerações "diferentes" extraídas para cada um.

Para validar a qualidade do mecanismo de OCR sugerido, foram utilizadas um total de 12 imagens, retiradas dos vídeos mencionados na sessão 4.4, utilizando uma câmera de celular. As imagens apresentam uma baixa qualidade e algumas não estão em um ângulo favorável para o reconhecimento (de frente para a placa). Contudo é possível reconhecer visualmente os caracteres que estão presentes nas placas.

A realização dessa validação consiste em basicamente em aplicar o mecanismo de OCR proposto para as imagens e coletar os resultados delas, depois comparar com os valores reais presentes nas placas. Para melhor exemplificar foi montada uma tabela contendo os valores reais das placas utilizadas nos testes e os valores que o mecanismo de reconhecimento de caracteres encontrou.

A tabela abaixo apresenta os valores reais das placas, os valores encontrados pelo mecanismo de OCR e a configuração que foi responsável por encontrar estes valores.

PLACA REAL	OCR	CONFIGURAÇÃO
LTH7H09	‘’	3
GXG-4766	GYG-4766	1
QXR7A28	QXR7A28	1
DUI-3563	DUI3563	1
HEX-7587	‘’	3
GXK-5045	GXK-5045	2
HEW-1234	‘’	3
HCJ-5050	‘’	3
HBQ-3549	HBQ-3549	1
ENA5G99	‘’	3
HNZ0J55	‘’	3
QXR7A29	OXR7028	2
OQA-4173	‘’	3

Tabela 4. Comparação entre os valores reais de uma placa e o resultado da extração.

O mecanismo conforme já explicado, executa as configurações em ordem crescente enquanto não encontrar uma placa. Observando a tabela é possível notar que muitas das placas utilizadas no teste, os seus caracteres não foram devidamente identificados. E em algumas dessas imagens isto ocorre por causa das condições das imagens. Imagens que se encontram inclinadas, o Tesseract possui dificuldades de encontrar os caracteres dela. Além disso, algumas das imagens estavam com resolução muito ruim, e por causa disso, mesmo realizando os pré-processamentos, não foi possível extrair os caracteres da imagem. Em algumas outras placas é possível notar que o valor encontrado é diferente do valor real. Por causa do excesso de pré-processamento, isso pode gerar perda das características da imagem, levando a um reconhecimento errôneo.

Para melhor entender estes resultados, a figura 10 mostra cada uma das 12 imagens utilizadas nos testes e os caracteres que foram extraídos pelo mecanismo de OCR proposto.

Figura 10. Imagens utilizadas na validação e resultados do OCR.

Ao final, houve 50% de sucesso na extração de caracteres das placas utilizadas nos testes e destas placas 83% delas foram reconhecidas corretamente.

6. Conclusão

Ao longo deste trabalho foi possível desenvolver um modelo de aprendizagem capaz de realizar identificação de placas de licenciamento de veículos em ambientes não controlados e em tempo real. Ele apresenta resultados satisfatórios na identificação, contudo não apresentou a mesma satisfação na extração dos caracteres, tendo sucesso na

extração de apenas 50% das placas e dessas 83% estavam corretas.

De acordo com o f1 *score*, utilizando o valor de 0,45 para o limiar, são obtidos os melhores valores entre precisão e recall, que são respectivamente 0,91 e 0,88. Além disso, das 64 imagens utilizadas nos testes para este limiar, 54 foram reconhecidas como verdadeiros positivos, 5 como falsos positivos e 7 como falsos negativos, com resultado de 84,75%.

Os resultados obtidos incentivam maiores pesquisas na área e segurança e controle de acesso de veículos baseados na identificação das placas utilizando CNN, com a YOLO apresentando bons valores na identificação e na velocidade de reconhecimento. Porém, o OCR utilizado não apresentou bons resultados.

7. Trabalhos futuros

Há uma série de etapas que podem ser implementadas em trabalhos futuros, sendo que uma delas é a utilização de bibliotecaa como a easyOCR, ou mesmo a utilização de uma rede CNN para realziar esta tarefa.

Os resultados obtidos incentivam a realizar testes em câmeras de vigilância posicionadas fixamente em cancelas ou portarias de universidades e outros espaços que tenham acesso de veículos.

Referências

Bjarki, G. Car license plates dataset. https://www.kaggle.com/andrewmvd/car-plate-detection. Acessado em 02/02/2022.

Chollet, Francois. Deep learning with Python. Simon and Schuster, 2021.

Daniel De Los Reyes, Everton Thomas, L. L. d. R.-W. G. N. (2019). Predição de sucesso acadêmico de estudantes: uma análise sobre a demanda por uma abordagem baseada em transfer learning. Revista Brasileira de Informática na Educação, 27(01):01.

Etten, A. V. (2019). Satellite imagery multiscale rapid detection with windowed networks. 2019 IEEE Winter Conference on Applications of Computer Vision (WACV).

Gonzalez, Rafael C.; Woods, Richard E.; Eddins, Steven L. Digital image processing using MATLAB. Pearson Prentice Hall, 2010.

Google and Community (2022). Tesseract-ocr. https://github.com/tesseract-ocr/tessdoc.

Jamtsho,Yonten; Riyamongkol ,Panomkhawn R. W. (2021). Real-time license plate detection for non-helmeted motorcyclist using yolo. ICT Express, 7(1):104–109.

Karl Weiss, Taghi M. Khoshgoftaar, W. D. (2016). A survey of transfer learning. Journal of Big Data, 3(1):9.

Leon A. Gatys, Ecker Alexander S., M. B. (2021). A neural algorithm of artistic style.

Toward Data Science.

Leite, L.; Antonello, Ricardo. Identificação Automática De Placa De Veículos Através De Processamento De Imagem E Visão Computacional. Semana da Ciência e Tecnologia. SECITEC, vol.2, no.1, Luzerna set/2017, 2017.

Imamura, M. E., Silva, F. A. da, Almeida, L. L. de, Pereira, D. R., Artero, A. O., & Piteri, M. A. (2021). Detecção e reconhecimento de placas de licenciamento veicular em tempo real usando CNN. Colloquium Exactarum. ISSN: 2178-8332, 13(1), 89–99. Recuperado de https://revistas.unoeste.br/index.php/ce/article/view/4143.

Ministério Da Infraestrutura, D. N. d. T. (2020). Frota de veículos. https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?ano=2020.

Redmon, J. (2013–2016). Darknet: Open source neural networks in c. http://pjreddie.com/darknet/.

Saad, Albawi; Tareq, Abed Mohammed, S. A.-Z. (2017). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET), pages 1–6.

Silva, Rodrigo Emerson Valentim da. Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens. Trabalho de Conclusão de Curso. Universidade Federal do Ceará. 2018

Vittorio, A. (2018). Toolkit to download and visualize single or multiple classes from the huge open images v4 dataset. https://github.com/EscVM/OIDv4_ToolKit.

Yohanandan, S. (2020). map (mean average precision) might confuse you! https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2.