Rede Neural Convolucional para Detecção de Pedestres Realizando Travessias de Risco

Milena L. dos Santos*, Claudio R. M. Mauricio*, Valéria N. dos Santos† and Fabiana F. F. Peres*
*Universidade Estadual do Oeste do Paraná
Centro de Engenharias e Ciências Exatas, Foz do Iguaçu, Paraná - Brasil
Email: mii.santos342@gmail.com
†Fundação Parque Tecnológico Itaipu, Foz do Iguaçu, Paraná - Brasil

***Resumo*—Indivíduos atuam como pedestres quando encontram-se andando ou correndo em uma via. As principais interações entre pedestres e veículos ocorrem nas travessias de pedestres. Essas interações expõem os pedestres ao risco de acidentes e atrasos nos deslocamentos. Os pedestres estão suscetíveis a ferimentos graves e lesões que levam à morte e incapacidade, alarmando a saúde pública e a segurança do tráfego a tomar providências para tornar os pedestres menos expostos às situações de riscos produzidas pelo trânsito. O objetivo deste trabalho é detectar pedestres realizando travessias de risco utilizando imagens de vídeos transmitidos em tempo real. Para isso, as tecnologias utilizadas foram CNN Yolov4-tiny para detecção e algoritmo SORT para rastreamento e contagem dos pedestres. O modelo final obteve uma precisão aproximada de 89%. Em média, a inferência da aplicação levou de 11 a 13 *frames* por segundo.**

***Abstract*—Individuals act as pedestrians when they are walking or running on a road. The main interactions between pedestrians and vehicles occur at pedestrian crossings. These interactions expose pedestrians to the risk of accidents and travel delays. Pedestrians are susceptible to serious injuries and injuries that lead to death and disability, alarming public health and traffic safety to take steps to make pedestrians less exposed to hazardous situations produced by traffic. The objective of this work is to detect pedestrians carrying out risky crossings using video images transmitted in real time. For this, the technologies used were CNN Yolov4-tiny for detection and SORT algorithm for tracking and counting pedestrians. The final model obtained an approximate accuracy of 89%. On average, application inference took 11 to 13 frames per second.**

I. Introdução

Os usuários mais vulneráveis no trânsito são pedestres, ciclistas e motociclistas, na qual correspondem cerca da metade do valor integral das mortes causadas pelo trânsito no mundo. O Brasil está em quinto lugar entre os países que ocorrem mais mortes no trânsito, somente atrás da Índia, China, Estados Unidos e Rússia [1].

Frente ao cenário atual de mortes por acidente de trânsito, diferentes pesquisas tem sido conduzidas para que a tecnologia atual possa auxiliar na prevenção e redução de acidentes de trânsito. Há muitas pesquisas recentes que buscam prever acidentes de trânsito ou acidentes fatais, utilizando modelos preditivos lineares [2]. Outras avaliam o comportamento dos pedestres antes de atravessar a rua, como também estimam se a ação do pedestre apresenta uma situação de risco [3]. Existem também pesquisas que utilizam *Deep Learning* para avaliar estradas seguras e suas condições [4]. Avanços do sistema de alerta de colisão em cruzamentos tem trazido melhores resultados que o sistema convencional [5].

Devido a realidade de acidentes de trânsito, criar soluções e meios que possam prevenir ou então reduzir os riscos de acidentes é essencial no contexto atual. Dessa forma, este trabalho objetiva detectar pedestres durante a travessia em uma via de trânsito, no momento onde há risco de sofrer um acidente. Assim, é possível contar o número de pedestres que fazem travessia perigosa em um determinado trecho de via de trânsito, gerando informações que possam auxiliar autoridades competentes na tomada de medidas de segurança.

II. Trabalhos Relacionados

A proposta de trabalho de [6] propõe um sistema de detecção de pedestres baseado em aprendizado profundo, adaptando uma rede convolucional de uso geral à tarefa em questão. O trabalho traz otimizações na maioria das etapas da detecção de pedestres, superando os métodos tradicionais baseados em atributos projetados a mão (*handcrafted*) e abordagens de aprendizado profundo. A abordagem foi validada implementando-a em um NVIDIA Jetson TK1.

O trabalho apresentado por [7] propõe um aplicativo para contagem de pessoas por meio da detecção e rastreamento, executando a fase de detecção a cada N quadros, possibilitando rastrear o objeto até alcançar o N-ésimo quadro. Para o desenvolvimento, os autores utilizaram o Yolov3 e o algoritmo de rastreamento por filtros de correlação. Os resultados foram considerados satisfatórios.

O trabalho de [8] tem o objetivo de comparar duas abordagens clássicas para detecção de pedestres relacionadas a Visão Computacional e Rede Neural Convolucional. As técnicas que foram comparadas são: *Features de Haar* com AdaBoost, HOG com SVM e a rede Alexnet. Nos testes realizados, a abordagem com a CNN obteve os melhores resultados na maioria dos casos, comprovando o fato de ser considerada a técnica em estado da arte.

O trabalho de [9] implementa um aplicativo de rastreamento de múltiplos pedestres usando a abordagem de rastreamento por detecção. Utiliza-se de um modelo pré-treinado para detecção de objetos e reidentificaçao de pedestres, predição linear de estado com filtragem recursiva, características geométricas e profundas como funções de dissimilaridade entre pedestres, e um algoritmo ótimo para o problema

de associação devido à presença de vários pedestres num único quadro de vídeo. O método alcança 79,5% na métrica MOTA.

Outro trabalho relacionado encontrado na literatura é o de [10] que propõe um sistema de detecção, rastreamento e reconhecimento de ação de pedestres/carros usando aprendizado profundo usando fluxos de vídeo. Os autores utilizaram a CNN SSD e *mobilenets* para um processo mais rápido e maior taxa de detecção.

III. Materiais e Métodos

A. Materiais

A CNN YOLOv4-tiny foi selecionada para o desenvolvimento do trabalho. A YOLO (You Only Look Once) se propõe a detectar objetos em tempo real e ao contrário de outras técnicas, este método observa a imagem apenas uma vez. A imagem é separada em várias sub-regiões para realizar a detecção. Numerosas células são estimadas para cada sub-região, juntamente com as chances de que cada uma delas contenha um objeto [11].

A probabilidade de uma célula conter um objeto é então usada para ordenar as células. Essa ordenação indica apenas se a célula provavelmente conterá um objeto. Somente as células com maior probabilidade de conter objetos são deixadas depois que as células abaixo de um limite são excluídas [11]. Resumidamente, Yolo é uma versão condensada de uma ampla gama de técnicas de visão computacional para detecção de objetos.

O algoritmo SORT funciona com o princípio de rastreamento por detecção. Utiliza um vetor de *bounding boxes* como entrada e consegue rastrear estes múltiplos objetos em um vídeo em tempo real, alcançando bons resultados com baixo processamento, devido a utilização de métodos simples, porém eficientes. O algoritmo é composto por quatro componentes principais: detecção, estimativa, associação e manutenção da vida útil de objetos rastreados [12].

Neste trabalho utilizou-se uma câmera pública disponibilizada no site da Catve [13]. Ela registra, em tempo real,imagens da Aduana da Ponte Internacional da Amizade. Foram gravados vídeos de 5 a 30 minutos durante 12 dias, entre as 8:00 horas da manhã e 16:00 horas da tarde, horário de maior fluxo de pedestres. Os vídeos foram gravados em diversas condições de iluminação e clima como tempo ensolarado, chuvoso, nublado ou parcialmente ensolarado. A Figura 1 ilustra a Aduana em diferentes tempos climáticos. Ao todo foram 21 GB de vídeos registrados para o *dataset*, da qual foram extraídas 1600 imagens por meio de um *script* em Python.

Como a câmera não tem uma qualidade ou resolução avançada, muitas imagens foram excluídas já que haviam distorções no formato dos pedestres atravessando a via. A Figura 2 apresenta uma imagem distorcida contendo um pedestre atravessando a via. As imagens que apresentaram essas distorções não foram inseridas no *dataset*, pois poderiam dificultar o aprendizado da CNN.

O software escolhido para a realização da tarefa de anotação de objetos em imagens foi o LabelImg [14], disponível sob licença gratuita em seu repositório web. Dentre as funcionalidades existentes neste software, ele permite a demarcação de *bounding boxes* nos objetos de interesse em cada imagem. É uma maneira simples e gratuita de anotar o conjunto de imagens.

Figure 1. Tempos climáticos diferentes registrados.

Figure 2. Imagem distorcida contendo pedestres atravessando a via.

Para aplicar técnicas de pré-processamento, foi utilizado o Roboflow [15]. Ele foi escolhido por ser uma plataforma para visão computacional que oferece métodos para coleta de dados, pré-processamento e treinamento de modelos, permitindo que os usuários criem modelos de visão computacional de forma precisa.

Darknet é uma rede neural de código aberto escrita em C e CUDA, que suporta computação de CPU e GPU. Darknet pode ser usado diretamente e não precisa de nenhuma outra biblioteca. Ela possui duas dependências opcionais, o OpenCV para uma variedade maior de tipos de imagem suportados e CUDA para computação GPU. O Darknet é famoso por sua arquitetura de detecção de objetos em tempo real: YOLO.

O Google Colaboratory também chamado de Colab, foi utilizado para realizar os treinamentos da rede neural. É um serviço baseado em nuvem e que não requer configuração. A GPU disponível na plataforma Colab é o modelo Tesla K80, P100, P4 e T4 com capacidade computacional de 7,5 GHz com 14 GB de RAM.

Para o processo de rastreamento e contagem de pedestres foi utilizado um notebook contendo uma GPU da Nvidia

GeForce GTX 1060 Mobile, bibliotecas como CUDA, cuDNN e OpenCV e a linguagem de programação Python.

O OpenCV é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto. Ele implementa um grande conjunto de algoritmos de visão computacional incluindo o módulo DNN. Este é o módulo do OpenCV que é responsável por todas as coisas relacionadas ao aprendizado profundo. Este módulo permite o uso de modelos pré-treinados para inferência de *frameworks* como Caffe, TensorFlow, Torch e Darknet. Isso significa que é possível treinar modelos usando o Darknet e fazer inferência com o OpenCV, tornando o código final compacto e simples.

B. Métodos

A plataforma Roboflow foi utilizada para redimensionamento das imagens de 1800 x 905 para 832 x 832 pixels. Para evitar erros com a orientação da imagem, foi aplicado a Auto Orientação e para aumentar o número de exemplos, aplicou-se o *Bounding box flip horizontal*, que inverte ou espelha somente o objeto de interesse. Dessa forma, forneceu exemplos de pedestres atravessando a via em direções diferentes.

Para o treinamento foi alterado alguns hiperparâmetros com objetivo de melhorar a detecção de objetos. Os hiperparâmetros ficam armazenados em um arquivo ".cfg". Existem diversos tipos de arquivos cfg para diferentes objetivos. Observou-se durante a fase de anotação das imagens, que havia *bounding boxes* grandes e pequenos, ou seja, dependendo da região da imagem, os pedestres podiam estar perto da câmera ou distantes, ocupando uma região pequena da imagem. Alexey [16], um dos criadores da YOLO e Darknet, disponibiliza um arquivo cfg, com hiperparâmetros específicos para yolov4 e yolov4-tiny que trata especificamente esse caso para melhorar a detecção de objetos. A Tabela I apresenta as diferenças entre os arquivos cfg yolov4-tiny padrão e yolov4-tiny específico. Vale ressaltar que alguns parâmetros são diferentes por consequência do aumento do número de camadas.

Table I
DIFERENÇAS ENTRE OS ARQUIVOS YOLOV4-TINY PADRÃO E ESPECÍFICO

Camada	YOLOv4-tiny padrão	YOLOv4-tiny específico
Convolucional	21	24
Yolo	2	3
Route	11	13
Upsample	1	2

Alguns hiperparâmetros foram alterados de acordo com a necessidade atual. Os hiperparâmetros alterados foram: *Batch*, *Subdivisions*, *width* e *height*, *Max_batches*, *steps* e *Random*. Os valores definidos para esses hiperparâmetros são recomendados pelo Alexey [16]. Na Tabela II são apresentados os valores utilizados nos treinamentos.

Table II
HIPERPARÂMETROS ALTERADOS NOS TREINAMENTOS

Hiperparâmetros	**Valores utilizados**
Batch	64
Subdivisions	16
width e *height*	416, 512, 608, 812
Max_batches	6000
steps	4800,5400
Random	1

IV. RESULTADOS

Inicialmente, realizou-se um treinamento utilizando 376 imagens para compreender as limitações da CNN. Observou-se que a CNN detectava e classificava ciclistas e motociclistas como pedestres. Com isso, foram adicionadas 396 exemplos negativos que continham principalmente motociclistas.

No *dataset* final obteve-se 1600 imagens, no qual foi dividido em 90% para treino e validação e 10% para teste. O *dataset* correspondente aos 90% foram divididos em 5 *folds*, no qual o conjunto de validação nunca foi o mesmo, sendo 1152 exemplos para treinamento, 288 para validação e 160 para teste, garantindo que cada pedestre no conjunto de dados estivesse em um conjunto de validação. Para cada *fold* foram aplicados *Bounding box flip horizontal* para aumentar o número de exemplos. Para o modelo final, o conjunto de treinamento e validação foram reunidos resultando em 1440 imagens para treino. A técnica de *transfer learning* foi utilizada em todos os treinamentos realizados.

Para o último treinamento, optou-se pelo valor 832 no hiperparâmetro *width* e *height*. A vantagem de ter um conjunto de teste que o modelo não tenha visto antes, durante as etapas de treinamento é que pode-se obter uma estimativa menos tendenciosa de sua capacidade de generalizar para novos dados. O conjunto de treinamento recebeu mais 76 exemplos utilizando a técnica para aumentar os dados, resultando em 1516 imagens.

Dos 303 pedestres identificados atravessando a rodovia no conjunto de teste, 274 foram corretamente classificados. O que não foi classificado corretamente: 29 não foram detectados pela rede (falsos negativos) e 138 foram identificados como pedestres atravessando a rodovia, mas não eram (falsos positivos). Os valores obtidos foram 90% de sensibilidade, 67% de precisão, 9.6% da taxa de erro da classe positiva e 38% da taxa de erro total.

O número alto de falsos positivos se deve ao fato de várias pessoas estarem no acostamento da via ou no meio dela, com o objetivo de entregar panfletos. Essa prática é muito comum na Aduana, pois muitas pessoas vão ao Paraguai para fazer compras.

A Figura 3, mostram várias pessoas no meio da rodovia. A CNN detecta 7 pessoas que estão no meio da rodovia, mas que não é considerado para este trabalho e realiza 2 detecções verdadeiros positivos.

A Tabela III apresenta os resultados das métricas IoU, AP, F1-Score. É possível compreender que a precisão resultou em 67% pelo número considerável de falsos positivos e consequentemente interferiu no resultado de F1-Score e AP.

Figure 3. Imagem do conjunto de teste ilustrando diversas pessoas na via sendo detectadas pela CNN.

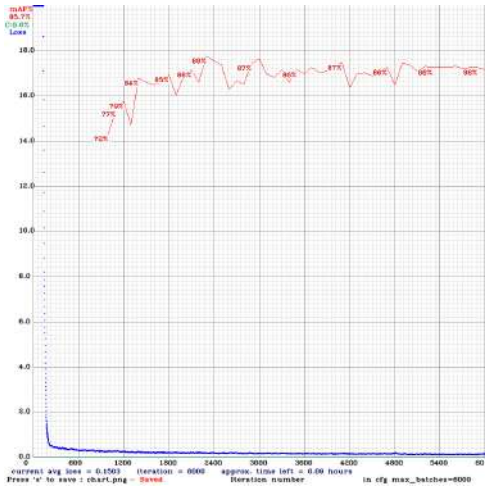

Figure 4. Gráfico que relaciona AP com a função de perda.

Ainda assim, obteve-se quase 89% de precisão média da CNN, considerado um resultado positivo. A Figura 4 apresenta o gráfico com o AP e a função de perda (*loss*) ao longo das 6000 iterações.

Table III
RESULTADOS DO MODELO FINAL.

Métricas	**Modelo final**
AP50	88.76%
IoU	50.86%
F1-Score	77%

O algoritmo Sort para rastreamento de objetos é utilizado para auxiliar na contagem de pedestres realizando uma travessia de risco. O parâmetro que define o número máximo de *frames* que mantém os identificadores de um objeto sem detecções associadas foi alterado para 50, visto a necessidade da aplicação. O valor mínimo do IoU definido é 0.5, somente

Figure 5. Linhas traçadas estrategicamente.

Figure 6. Contagem de pedestres realizando travessia de risco.

detecções com o IoU acima de 0.5 foram consideradas. Para realizar a contagem foram traçadas 3 linhas no meio da rodovia utilizando o OpenCV. A Figura 6 mostra as linhas traçadas para a contagem dos pedestres. Ao passar um pedestre por uma dessas linhas é realizado a contagem.

O ideal para a execução de uma aplicação em tempo real é de 30 a 60 FPS. No entanto, o tempo de execução da atual aplicação varia de 11 a 13 FPS. Uma maneira de solucionar esse problema seria executar a aplicação em uma GPU mais avançada. Contudo, essa possibilidade não foi possível.

Ao executar a aplicação foi possível observar que a CNN detecta motociclistas com pouca frequência. Quando uma contagem é feita, o *frame* correspondente é salvo como imagem e a contagem é registrada em um arquivo csv. A Figura 6 apresenta a contagem de pedestres atravessando a via de transporte.

V. CONCLUSÃO

Redes neurais convolucionais são algoritmos ideais para a tarefa de detecção de objetos, de acordo com os estudos e resultados obtidos. O treinamento do objeto de interesse foi simplificado através da implementação do modelo de rede Yolov4-tiny, utilizando a rede pré-treinada e usufruindo do conhecimento já aprendido pelo modelo.

Com os diversos treinamentos realizados, o modelo final demonstrou resultados satisfatórios, como média da precisão de 89% aproximadamente, média do IoU de 50% e média harmônica ponderada da precisão de 77%. Seguir as instruções dos autores da Yolo de como melhorar a precisão da detecção de objetos contribuiu para esse resultado final. Utilizar a rede da Yolov4-tiny com três camadas Yolo, alterar os parâmetros *random* e *width* e *height*, adicionar exemplos negativos influenciaram positivamente nos resultados.

Contudo, ainda é possível melhorar ainda mais esses resultados, adicionando mais exemplos negativos de ciclistas, motociclistas e pessoas que trabalham entregando panfletos na via, testando diferentes hiperparâmetros. Isso é crucial para o algoritmo de Sort que depende das detecções do modelo para rastrear os objetos.

AGRADECIMENTOS

Os autores gostariam de agradecer a Universidade Estadual do Oeste do Paraná e a Fundação Parque Técnológico Itaipu por concederem material para o desenvolvimento do projeto.

REFERENCES

[1] P. V. Margon and P. W. G. Taco, "Caracterização do comportamento de pedestres e motoristas durante a travessia de vias em faixa não semaforizada," in *Anais 18º PANAM-Congresso Panamericano de Engenharia de Trânsito, Transporte e Logística. Universidade de Cantabria–Santander, Espanha*, vol. 11, 2014.

[2] A. Chakraborty, D. Mukherjee, and S. Mitra, "Development of pedestrian crash prediction model for a developing country using artificial neural network," *International journal of injury control and safety promotion*, vol. 26, no. 3, pp. 283–293, 2019.

[3] S. Zhang, M. Abdel-Aty, J. Yuan, and P. Li, "Prediction of pedestrian crossing intentions at intersections based on long short-term memory recurrent neural network," *Transportation research record*, vol. 2674, no. 4, pp. 57–65, 2020.

[4] Z. Jan, B. Verma, J. Affum, S. Atabak, and L. Moir, "A convolutional neural network based deep learning technique for identifying road attributes," in *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 2018, pp. 1–6.

[5] D. Ka, D. Lee, S. Kim, and H. Yeo, "Study on the framework of intersection pedestrian collision warning system considering pedestrian characteristics," *Transportation research record*, vol. 2673, no. 5, pp. 747–758, 2019.

[6] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal processing: image communication*, vol. 47, pp. 482–489, 2016.

[7] A. F. Cordeiro, L. Pedro Filho, C. A. Ojeda, and G. R. Valiati, "Rastreamento e contagem de pedestre em tempo real por meio de imagens digitais," in *Anais do XVI Congresso Latino-Americano de Software Livre e Tecnologias Abertas*. SBC, 2019, pp. 146–149.

[8] A. C. G. Vargas, A. Paes, and C. N. Vasconcelos, "Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres," in *Proceedings of the xxix conference on graphics, patterns and images*, vol. 1, no. 4. sn, 2016.

[9] M. d. C. J. Ize, "Multiple pedestrian tracking using geometric and deep features," 2019.

[10] H. Song, I. K. Choi, M. S. Ko, J. Bae, S. Kwak, and J. Yoo, "Vulnerable pedestrian detection and tracking using deep learning," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2018, pp. 1–2.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[12] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[13] Catve, "Câmeras ao vivo," https://catve.com/aduana.

[14] Tzutalin, "Labelimg," https://github.com/tzutalin/labelImg.

[15] Roboflow, "Roboflow," https://roboflow.com/.

[16] A. Bochkovskiy, "How to improve object detection," https://github.com/AlexeyAB/darknethow-to-improve-object-detection.