O USO DE VISÃO COMPUTACIONAL PARA DETECÇÃO DE CONDUÇÃO SONOLENTA

Caio Farias Felipe dos Santos, Lucas Baggio Figueira

Faculdade de Tecnologia de FATEC Ribeirão Preto (FATEC)

Ribeirão Preto, SP - Brasil

caio.santos114@fatec.sp.gov.br,
lucas.figueira@fatec.sp.gov.br

Resumo: *Este artigo explora o uso de técnicas e ferramentas de visão computacional e aprendizado de máquina para detectar sonolência de motoristas e como elas podem ser aplicadas na prevenção de acidentes de trânsito envolvendo condução sonolenta.*

Abstract: *This article explores the use of computer vision and machine learning techniques and tools to detect drowsiness in drivers and how they can be applied to prevent traffic accidents involving drowsy driving.*

1. Introdução

A fadiga é definida como o estado de esgotamento, seguido a um período de esforço mental ou físico, caracterizado por uma queda na capacidade para trabalhar e reduzida eficiência para responder aos estímulos (BVS, 1999). Fatores como: privação de sono, esforço físico, estresse, doenças e determinadas medicações, podem levar a pessoa a um certo nível de exaustão corporal e/ou mental.

A manifestação da fadiga não somente afeta o indivíduo em suas atividades rotineiras, mas também é considerado um fator de risco no trânsito e pode se tornar a causa de acidentes. Na verdade, poucos são os que têm noção do quão perigoso é a prática do exercício de condução sob efeito de fadiga, que no seu extremo pode levar ao sono, o que, em ambiente rodoviário pode significar a morte do próprio e/ou de terceiros. (COSTA, 2014)

Segundo a Associação Brasileira de Medicina de Tráfego (ABRAMET, 2019), cerca de 20% dos acidentes de trânsito estão relacionados à sonolência. Essa é uma das principais causas de mortes nas rodovias. A mesma entidade também realizou uma pesquisa em 2017, em conjunto com a Academia Brasileira de Neurologia e o Conselho Regional de Medicina, entrevistando aproximadamente 500 motoristas e concluiu que mais de 20% costumam dirigir com sono. Quase 40% conhecem alguém que já se envolveu em acidente por esse motivo. (ABRAMET, 2019).

Levando em consideração as graves consequências associadas à condução enquanto sonolento, é de vital importância implementar medidas preventivas e criar ferramentas para combater o problema da sonolência ao volante. Segundo Wessel (2022), detectar condutores fatigados e alertá-los para cuidarem de sua segurança, fazendo uma pausa se estiverem sonolentos, é uma maneira de abordar esse problema. Com a capacidade dos computadores nos dias atuais e o surgimento da área da visão computacional, é possível, segundo Salles (2022), "[...] analisar, interpretar e extrair informações relevantes de imagens e/ou vídeos para que decisões possam ser tomadas,

ou para gerar dados relevantes para uma aplicação futura.”. Sob essa perspectiva, o objetivo deste trabalho é desenvolver uma aplicação de visão computacional capaz de detectar a sonolência do motorista com base nas suas expressões faciais.

2. Visão Computacional

Visão Computacional é um campo que faz uso da Inteligência Artificial e do Aprendizado de Máquina (*Machine Learning*) para buscar e extrair informações significativas da perspectiva visual, a saber, imagens e vídeos, a fim de tomar uma determinada ação ou identificar padrões. Para isso, duas metodologias são essenciais: *deep learning* e CNN (*convolutional neural network* ou rede neural convolucional).

2.1 *Deep Learning*

O *deep learning*, ou aprendizagem profunda em português, é um ramo do aprendizado de máquina em que redes neurais artificiais identificam padrões a partir de grandes volumes de dados.

> “Esses métodos de *deep learning*, como redes neurais artificiais profundas, usam múltiplas camadas de processamento para descobrir padrões e estruturas em conjuntos de dados muito grandes. Cada camada aprende um conceito a partir dos dados sobre os quais as camadas subsequentes se baseiam; quanto maior o nível, mais abstratos são os conceitos aprendidos. O *deep learning* não depende de um processamento de dados prévio e extrai recursos automaticamente. Para usar um exemplo simples, uma rede neural profunda encarregada de interpretar formas aprenderia a reconhecer arestas simples na primeira camada e então adicionaria o reconhecimento das formas mais complexas compostas por essas arestas nas camadas subsequentes.” (RUSK, 2016)

2.2 Rede neural convolucional (CNN)

Similares às tradicionais Redes Neurais Artificiais ANNs (sigla para *Artificial Neural Networks*), as Redes Neurais Convolucionais (CNNs) são compostas por unidades conectadas entre si, chamadas de neurônios artificiais, que ajustam seus pesos sinápticos de maneira retropropagada realizando assim o aprendizado de um determinado padrão. Cada neurônio ainda irá realizar uma operação com base no valor de entrada recebido.

> “A única diferença notável entre as CNNs e as ANNs tradicionais é que as CNNs são usadas principalmente no campo de reconhecimento de padrões em imagens. Isso nos permite codificar recursos específicos de imagens na arquitetura, tornando a rede mais adequada para tarefas focadas em imagens ao mesmo tempo que reduz ainda mais os parâmetros necessários para configurar o modelo.” (O’SHEA & NASH, 2015)

2.3 Aplicações

Com diversas aplicações na solução de problemas reais, as tecnologias voltadas à visão computacional têm se proliferado em diversas áreas. (BRANDIZZI, 2020).

Segundo Neves, Neto e Gonzaga (2012, p. 6) as aplicações de visão computacional estão presentes em diversos segmentos tecnológicos que envolvem análise de imagens, reconhecimento de padrões e controle inteligente, abrangendo múltiplas áreas do conhecimento, tais como agronomia, astronomia, biologia, biometria, medicina e muitas outras.

No campo da medicina, a visão computacional está sendo utilizada na análise de Raios X, impressões de MRI e estruturas celulares. Métodos de análise para histologia mamária para evitar câncer também estão sendo utilizados, assim, o que antes era necessário um patologista com anos de experiência, pode agora ser feito rapidamente por uma máquina (ROSEBROOK, ADRIAN, 2016 apud GONZAGA, 2017, p. 12).

Outras aplicações da visão computacional estão na área da segurança. Análise de vídeos de segurança para procurar suspeitos após um roubo, detectar movimento suspeito em uma área. (GONZAGA, 2017, p. 11)

3. Tecnologias de desenvolvimento

Esta seção abordará as ferramentas e métodos utilizados para o desenvolvimento do sistema para o reconhecimento facial de expressões de sonolência.

3.1 Python

Python é uma linguagem de programação interpretada, de alto nível, multiparadigma e dinamicamente tipada, isto é, não exige a declaração explícita de tipos de dados nas variáveis, pois a própria linguagem é capaz de definir os tipos a serem utilizados durante a execução do programa, com base nos valores atribuídos a essas variáveis.

Python foi criado no início da década de 1990 por Guido van Rossum na *Stichting Mathematisch Centrum*, nos Países Baixos, como o sucessor de uma linguagem chamada ABC. (PYTHON SOFTWARE FOUNDATION, 2023)

3.2 NumPy

NumPy é um projeto de código aberto que permite a computação numérica com Python. Foi criada em 2005 com base nos primeiros trabalhos das bibliotecas Numeric e Numarray. (NUMPY, 2023)

> "É uma biblioteca Python que fornece um objeto de matriz multidimensional, vários objetos derivados (como matrizes e matrizes mascaradas) e uma variedade de rotinas para operações rápidas em matrizes, incluindo matemática, lógica, manipulação de forma, classificação, seleção, [...] álgebra linear básica, operações estatísticas básicas, simulação aleatória etc." (NUMPY, 2023)

3.3 PyTorch

PyTorch é uma biblioteca de aprendizado de máquina de código aberto, desenvolvida inicialmente pelo grupo de pesquisa em IA da *Meta* (antiga *Facebook*), que possibilita o desenvolvimento e o treinamento de modelos de *deep learning* baseados em redes neurais.

3.4 OpenCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de software de código aberto de visão computacional e aprendizado de máquina, que foi construída para fornecer uma infraestrutura comum para aplicações de visão computacional e para acelerar o uso da percepção de máquina nos produtos comerciais. (OPENCV, 2024)

A biblioteca tem mais de 2.500 algoritmos otimizados, que inclui um conjunto abrangente de algoritmos clássicos e de última geração de visão computacional e aprendizado de máquina. (OPENCV, 2024)

3.5 YOLO

YOLO (*You Only Look Once*), um modelo popular de detecção de objetos e segmentação de imagens, foi desenvolvido por Joseph Redmon e Ali Farhadi na Universidade de Washington. Lançado em 2015, o YOLO rapidamente ganhou popularidade por sua alta velocidade e precisão. (ULTRALYTICS, 2023)

Redmon *et al*. (2015) descreve os desafios de realizar detecções de forma otimizada com outros métodos:

> "Abordagens mais recentes, como R-CNN, usam métodos de proposta de região para primeiro gerar potenciais caixas delimitadoras em uma imagem e, em seguida, executar um classificador nessas caixas propostas. Após a classificação, o pós-processamento é usado para refinar as caixas delimitadoras, eliminar detecções duplicadas e remarcar as caixas com base em outros objetos na cena. Esses processos complexos são lentos e difíceis de otimizar porque cada componente individual deve ser treinado separadamente."

YOLO é agradavelmente simples, como mostra a Figura 1. Uma única rede convolucional prevê simultaneamente múltiplas caixas delimitadoras e prováveis classes para essas caixas. O YOLO treina imagens completas e otimiza diretamente o desempenho da detecção. (REDMON *et al*., 2015)

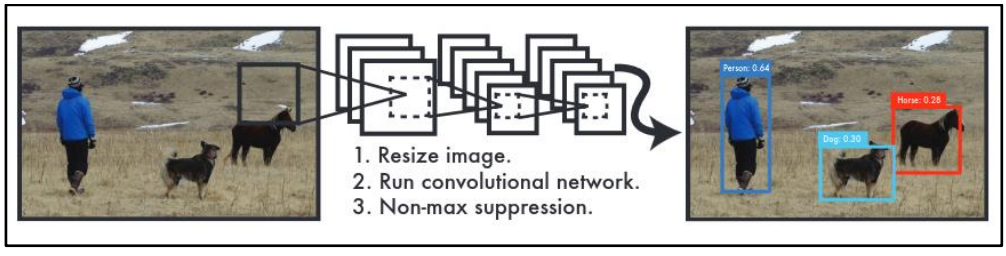

Figura 1: O sistema de detecção YOLO.
Fonte: Redmon *et al*., 2015

3.6 Google Colab

O Colab é um serviço do Jupyter Notebook hospedado que não requer configuração para uso e oferece acesso gratuito a recursos de computação, incluindo GPUs e TPUs. (GOOGLE, 2017). Trata-se de um ambiente de desenvolvimento que permite executar código Python pelo navegador, através de documentos chamados *notebooks*.

Um *notebook* é um documento compartilhável que combina código de computador, descrições em linguagem simples, dados, visualizações ricas como modelos 3D, gráficos e figuras, e controles interativos. (JUPYTER, 2015)

O Colab é adequado principalmente para aprendizado de máquina, ciência de dados e educação. (GOOGLE, 2017)

4. Desenvolvimento do sistema

Para o desenvolvimento deste trabalho, foi preciso treinar um modelo de aprendizado supervisionado usando imagens de um *dataset* personalizado. O aprendizado supervisionado é uma categoria de aprendizado de máquina que usa conjuntos de dados rotulados para treinar algoritmos para prever resultados e reconhecer padrões. (GOOGLE CLOUD, 2024). Esses conjuntos de dados são conhecidos como dataset.

Ao contrário da aprendizagem não supervisionada, os algoritmos de aprendizagem supervisionada recebem treinamento rotulado para aprender a relação entre a entrada e as saídas. (GOOGLE CLOUD, 2024) O modelo YOLO precisa ser treinado com dados rotulados para aprender as classes de objetos presentes nesses dados.

4.1 Rotulagem de dados

O *dataset* deste projeto contém um total de 252 imagens, divididas igualmente entre 126 imagens de expressões de uma pessoa acordada e 126 imagens de uma pessoa sonolenta, capturadas por uma webcam e salvas no formato JPEG. Para a rotulagem de cada imagem, duas classes são definidas: *acordado* e *sonolento*. A construção do *dataset* foi feita com o auxílio do software Roboflow, uma plataforma de desenvolvimento de projetos de *machine learning* e criação de *datasets*, que permitiu a manipulação e edição das imagens com caixas delimitadoras e a rotulagem de cada uma delas, além de mudanças de propriedades como brilho, contraste, entre outras. A Figura 2 mostra duas imagens de exemplo inseridas no *dataset*, representando a expressão de uma pessoa acordada e de uma pessoa sonolenta, respectivamente.

Figura 2: Uma pessoa acordada e uma pessoa dormindo
Fonte: Autoria própria

O software Roboflow suporta uma ampla gama de formatos de entrada e saída, tornando fácil a importação de *datasets* em diferentes bibliotecas de machine learning. (SHANDILYA, S. K. *et al*, 2023)

Das imagens presentes no *dataset*, 228 foram separadas para treinar o modelo e 16 para a validação, isto é, para avaliar a performance do modelo enquanto os parâmetros são ajustados durante o treinamento. Além disso, mais 8 imagens foram separadas para teste, ou seja, elas serão usadas apenas para a avaliação final do modelo após o treinamento. Observe a Figura 3.

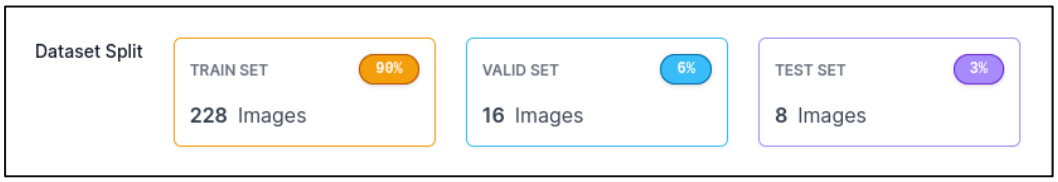

Figura 3: Divisão do dataset entre dados para treino, validação e teste.
Fonte: Autoria própria

Após incluir as imagens e rótulos ao *dataset*, foi feita a sua exportação para o formato YOLO e seu carregamento para o projeto no Google Colab. A Figura 4 mostra o arquivo YAML com as informações do *dataset* importado do Roboflow (linhas 5 a 10), as classes usadas como rótulos para as imagens (linhas 2 e 3) e os diretórios das imagens para teste, treinamento e validação do modelo (linhas 11 a 13).

```yaml
names:
- acordado
- sonolento
nc: 2
roboflow:
  license: CC BY 4.0
  project: drowsiness-dotzm
  url: https://universe.roboflow.com/fatec-ofqxt/drowsiness-dotzm/dataset/4
  version: 4
  workspace: fatec-ofqxt
test: ../test/images
train: Drowsiness-4/train/images
val: Drowsiness-4/valid/images
```

Figura 4: Documento YAML do *dataset*
Fonte: Autoria própria

4.2 Treinamento do modelo

A biblioteca PyTorch foi utilizada para carregar o modelo YOLO. Após importar o *dataset* para o projeto, deu-se início ao treinamento do modelo. O YOLO possui, por padrão, seu próprio conjunto de dados para a detecção de diversos objetos. Portanto, foi preciso substituí-lo pelo *dataset* importado do Roboflow, contendo os parâmetros necessários para detectar as expressões faciais.

O modelo foi treinado no ambiente de desenvolvimento Google Colab com recursos computacionais fornecidos pela própria plataforma, como a GPU T4, uma unidade de processamento gráfico de alto desempenho, que é especialmente adequada para operações de *deep learning* e processamento de grandes conjuntos de dados. Esse conjunto de ferramentas e recursos permitiu um processo de treinamento mais eficiente em menos tempo, eliminando a dependência de hardware local robusto.

O modelo foi treinado por 150 *epochs*, o que significa que ele passou por todo o *dataset* 150 vezes durante o treinamento. Cada passagem completa pelo conjunto de dados, conhecida como época (*epoch*), permitiu ao modelo ajustar seus parâmetros e melhorar seu desempenho.

4.3 Captura de imagens e inferência

Após o treinamento, o modelo está pronto para ser usado em imagens totalmente novas e não rotuladas. Essa etapa é chamada de inferência. No campo da inteligência artificial, a inferência é o processo que um modelo de aprendizado de máquina treinado usa para tirar conclusões a partir de dados totalmente novos. (CLOUDFLARE, 2024)

```python
# carregando o modelo YOLO personalizado
model = torch.hub.load('ultralytics/yolov5',
                       'custom',
                       path='drowsiness_model/exp2/best.pt')

```

Figura 5: Código para carregar o modelo treinado
Fonte: Autoria própria

A biblioteca OpenCV possui uma classe chamada *VideoCapture*, que permite o uso de câmeras ou arquivos de vídeo para captura de imagens, como mostra a Figura 6.

Na linha 11 do código, um objeto do tipo *VideoCapture* é instanciado com o índice 0, para usar a webcam do computador. As linhas 12 e 13 representam as dimensões da janela da aplicação.

```
10  # propriedades da janela da webcam
11  webcam = cv2.VideoCapture(0)
12  webcam.set(3, 640)
13  webcam.set(4, 480)
```

Figura 6: Trecho de código para usar a webcam
Fonte: Autoria própria

Na Figura 7, o código Python dá início a um loop na linha 19. Durante este loop, cada frame do vídeo capturado pela câmera é processado pelo modelo (linhas 20 e 21). Em seguida, a imagem processada é exibida na tela (linhas 23 e 24), mostrando a caixa delimitadora, o rótulo e a precisão, caso o modelo faça alguma detecção. Este processo se repete até que o usuário encerre a aplicação.

```
15  # Iniciando webcam
16  if not webcam.isOpened():
17      print("Não foi possível abrir a webcam.")
18      exit()
19  while True:
20      ret, img = webcam.read() # fazendo a leitura de um frame
21      detection = model(img) # o modelo processa o frame e guarda o resultado
22
23      cv2.imshow('Teste de captura',
24                 np.squeeze(detection.render())) # o resultado é renderizado
25
26      if cv2.waitKey(1) == ord('q'): # Aperte Q para encerrar a aplicação
27          break
28
29  webcam.release()
30  cv2.destroyAllWindows()
```

Figura 7: Código para detectar expressões faciais pelas imagens da câmera
Fonte: Autoria própria

5. Resultados

A Figura 8 abaixo mostra o desempenho do modelo durante seu treinamento com gráficos de perda. Perda é a penalidade para uma previsão ruim. Ou seja, perda é um número que indica quão ruim foi a previsão do modelo em um único exemplo. Se a previsão do modelo for perfeita, a perda será zero. Caso contrário, a perda será maior. (GOOGLE DEVELOPERS, 2024)

Os gráficos de treinamento (*train*) e validação (*val*) representam a perda associada às previsões das caixas delimitadoras (*box_loss*), detecção de objetos (*obj_loss*) e classificação dos objetos (*cls_loss*) no conjunto de dados do treino. O eixo vertical do gráfico indica a taxa de perda e o eixo horizontal indica as épocas (*epochs*). Observa-se um declínio nos gráficos, indicando que a taxa de perda está diminuindo e que o modelo está aprendendo a detectar melhor as expressões faciais, apesar das oscilações dos resultados nos gráficos de validação.

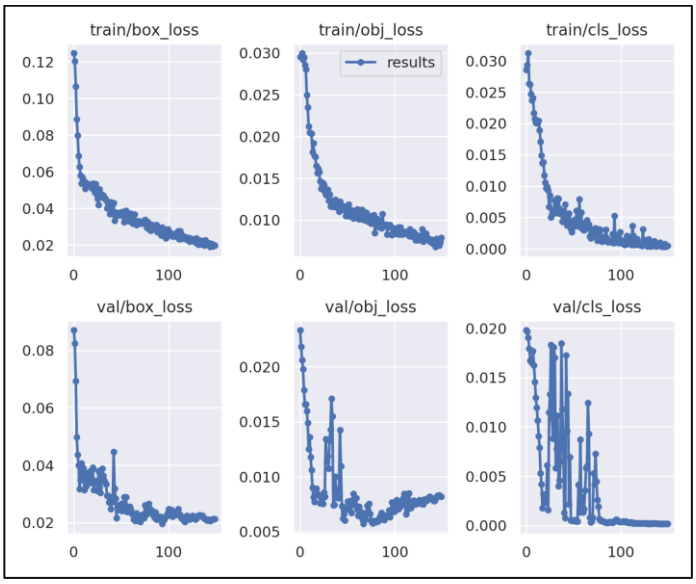

Figura 8: Resultados do treinamento do modelo
Fonte: Autoria própria

As imagens abaixo são detecções feitas pelo modelo por uma webcam. Estas imagens não estão incluídas no *dataset* e o modelo as classificou em tempo real.

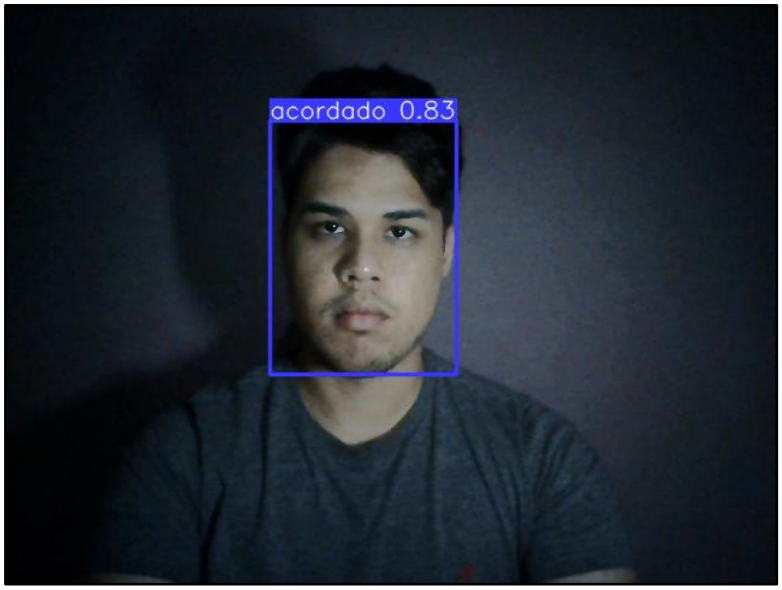

Figura 9: Detecção facial de uma pessoa acordada
Fonte: Autoria própria

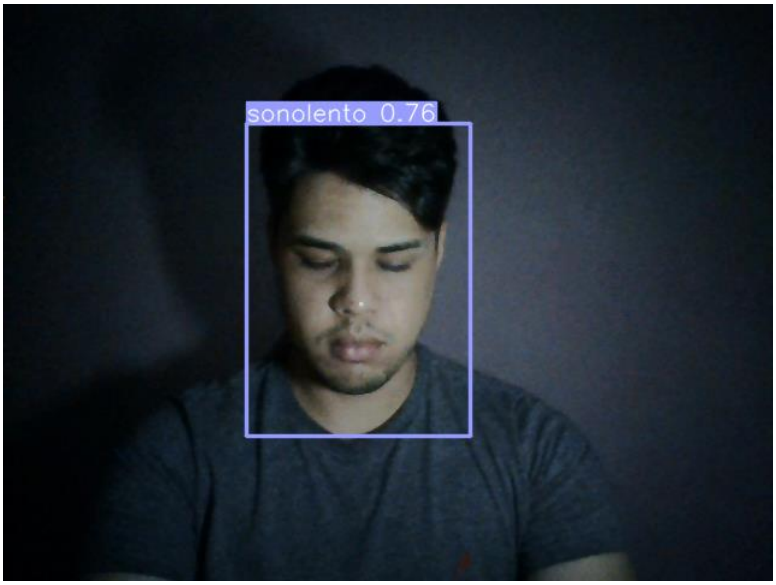

Figura 10: Detecção facial de uma pessoa dormindo
Fonte: Autoria própria

6. Conclusão

O objetivo inicial do presente trabalho foi atingido ao demonstrar uma forma de combater o problema da condução sonolenta no trânsito, fazendo uso de ferramentas de aprendizado de máquina e processamento de imagens para reconhecer quando um motorista estiver se sentindo sonolento. O sistema conseguiu detectar expressões de cansaço de uma pessoa em tempo real, apresentando uma precisão alta em diferentes exemplos durante os testes.

O trabalho representa a fase inicial do desenvolvimento do sistema, portanto, ele não está pronto para ser implementado em cenários do mundo real. A aplicação pode ser aprimorada futuramente com novas formas de classificar imagens, rastreando outros aspectos do rosto humano, como olhos e movimento da boca; um conjunto de dados mais robusto, com maior variedade de exemplos; tomada de decisões após detectar expressões de cansaço, como emitir um alerta para o condutor ou recomendar uma parada. Essas e outras melhorias visam contribuir para a segurança no trânsito e o bem-estar dos motoristas, proporcionado uma condução mais segura e confortável, reduzindo o número de acidentes causados por fadiga e, consequentemente, salvando vidas.

Referências

BIBLIOTECA VIRTUAL EM SAÚDE (1999) Descritores de Ciências da Saúde. BIREME. Disponível em: https://decs.bvsalud.org/.

BRANDIZZI, Loreane. Visão Computacional: O que é? Como funciona?. Disponível em: https://www.serpro.gov.br/menu/noticias/noticias-2020/o-que-eh-visao-computacional. Acesso em 23 ago. 2023.

CAMARGOS, A. S. et al. (2019) Jornal Medicina de Tráfego - Volume 2. 02. Ed. São Paulo: Associação Brasileira de Medicina de Tráfego. Disponível em: https://www.abramet.com.br/repo/public/commons/Jornal%20Medicina%20de%20Tr%C3%A1fego%20-%20Abril.pdf. Acesso em 4 jun. 2024.

CLOUDFLARE (2024). AI inference vs. training: What is AI inference?. Disponível em: https://www.cloudflare.com/learning/ai/inference-vs-training/

COSTA, D. J. R. J. da. (2014) A fadiga na condução. Cascais, Portugal: ANSR. Disponível em: http://www.ansr.pt/SegurancaRodoviaria/ArtigosTecnicos/Documents/Trabalho%20fadiga%20com%20logotipo%20ANSR.pdf. Acesso em 9 ago. 2023.

GOOGLE (2017) Colaboratory: Frequent Asked Questions. Google. Disponível em: https://research.google.com/colaboratory/faq.html. Acesso em 5 jun. 2024.

GOOGLE CLOUD (2024) What is Supervised Learning?. Google. Disponível em: https://cloud.google.com/discover/what-is-supervised-learning. Acesso em 30 mai 2024.

GOOGLE DEVELOPERS (2024) Descending into ML: Training and Loss. Google. Disponível em: https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss.

JUPYTER TEAM (2015) Project Jupyter Documentation. Jupyter. Disponível em: https://docs.jupyter.org/en/latest/. Acesso em 5 jun. 2024.

LUCAS AUGUSTO, Gonzaga (2017) Aplicações da Visão Computacional Utilizando Python. Uberlândia: Universidade Federal de Uberlândia. p. 11-12. Disponível em: https://repositorio.ufu.br/handle/123456789/25519. Acesso em 30 set. 2023.

NEVES, L. A. P.; NETO, H. V.; GONZAGA, Adilson (2012) Avanços em Visão Computacional. Curitiba: Omnipax. Disponível em: https://repositorio.utfpr.edu.br/jspui/handle/1/895. Acesso em 23 ago. 2023.

NUMPY DEVELOPERS (2023) What is NumPy?. NumPy. Disponível em: https://numpy.org/devdocs/user/whatisnumpy.html. Acesso em 9 out. 2023.

NUMPY TEAM (2023) NumPy – About Us. NumPy. Disponível em: https://numpy.org/about/. Acesso em 9 out. 2023.

O'SHEA, Keiron; NASH, Ryan (2015) An Introduction to Convolutional Neural Networks. arXiv. Disponível em: https://arxiv.org/pdf/1511.08458. Acesso em 30 abr 2024.

OPENCV (2024) About - OpenCV. Disponível em: https://opencv.org/about/.

PYTHON SOFTWARE FOUNDATION (2023) History and License. Python. Disponível em: https://docs.python.org/3/license.html. Acesso em 5 out. 2023.

REDMON, Joseph et al (2015) You Only Look Once: Unified, Real-Time Object Detection. arXiv. Disponível em: https://arxiv.org/abs/1506.02640. Acesso em 4 jun 2024.

RUSK, Nicole (2016) Deep learning. Nature Methods. Disponível em: https://www.nature.com/articles/nmeth.3707. Acesso em 8 set. 2023.

SALLES, Álvaro (2022) O que é Visão Computacional e para que serve?. Santo Digital. Disponível em: https://santodigital.com.br/o-que-e-visao-computacional-e-para-que-serve/. Acesso em: 15 ago. 2023.

SHANDILYA, S. K. et al (2023) YOLO-based segmented dataset for drone vs. bird detection for deep and machine learning algorithms. Data in Brief. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352340923004742. Acesso em 4 jun 2024.

ULTRALYTICS (2023) YOLO: A Brief History. Ultralytics. Disponível em: https://docs.ultralytics.com/#yolo-a-brief-history. Acesso em 9 out. 2023.

WESSEL, Rosalie (2022) What are driver drowsiness detection systems and how do they work?. TomTom. Disponível em: https://www.tomtom.com/newsroom/explainers-and-insights/driver-drowsiness-detection-systems/. Acesso em: 15 ago. 2023.