



BIANCA PAES DE ANDRADE SOARES

WENDERSON JUVENAL LOPES DOS SANTOS

**PROJETO DE *MACHINE LEARNING* E INTELIGÊNCIA ARTIFICIAL
ANALISANDO O ESTADO DE SEMÁFOROS**

RECIFE

2023

1. Introdução

O presente relatório tem como objetivo apresentar o projeto de um modelo de aprendizado de máquina, trabalho voltado para inteligência artificial, no qual envolve o treinamento de um modelo para identificação do estado de semáforos. O modelo se baseia na coloração do semáforo indicado na imagem enviada para determinar seu estado e retornar o resultado ao usuário.

2. Motivação

A ideia do modelo surgiu diante da necessidade atual de otimizar o tráfego urbano. Com isso, a implementação de um modelo de inteligência artificial capaz de analisar imagens e determinar o estado de semáforos contidos nela pode oferecer diversas aplicações práticas ao mercado. Por exemplo, ao integrar esse modelo a um sistema de transporte de carros autônomos em fase inicial de desenvolvimento, os carros inteligentes passam a ter a capacidade de interpretar o status dos semáforos em tempo real e reagir de forma precisa a essas informações, ajustando velocidade e planejando trajetórias de forma segura. Além disso, essa funcionalidade pode ser incorporada a aplicativos de navegação, proporcionando aos usuários uma maior precisão do tempo estimado do percurso e incluir rotas alternativas com base no estado atual de semáforos, podendo ser ajustados para otimizar a escolha de caminhos que minimizem o tempo de espera nos semáforos, possibilitando um deslocamento mais rápido e eficiente.

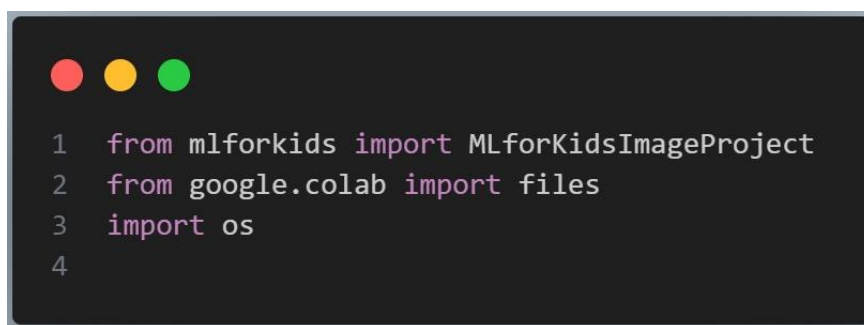
Visando uma aplicação de teste para linhas de comando, foi desenvolvido um algoritmo no qual o usuário envia uma imagem e lhe é retornado uma mensagem de acordo com a condição do semáforo, essa condição é determinada por meio de um código elaborado em Python e testado no Google Colab, no qual utiliza o modelo de aprendizagem de máquina criado no site *Machine Learning for Kids*. Com isso, para o treinamento do modelo, foram utilizadas diversas imagens de semáforos do *dataset* (base de dados) “*Traffic Light Detection Dataset*” da plataforma *Kaggle*, utilizado em conformidade com sua disponibilização, pela licença CC0 de Domínio Público; onde o script reconhece os padrões encontrados nas imagens de treinamento para tentar entender o que seria um sinal aberto (verde), um sinal fechado (vermelho) ou um sinal de atenção (amarelo) e armazenar esses dados para tomar decisões futuras.

Com o modelo treinado, o script analisa a cor indicada nos semáforos da imagem solicitada e a classifica como semáforo liberado ou não liberado, exibindo uma mensagem para revelar o resultado. O script informa também a taxa de confiança em sua resposta dada.

3. Scripts do Projeto:

Logo abaixo serão descritos todos os blocos do script desenvolvido, bem como as saídas geradas em cada linha do algoritmo. No entanto, antes de executar o código em Python, foi necessário instalar as dependências necessárias, sendo elas: “*NumPy*” (1.23.5), “*Pillow*” (9.3.0), “*SciPy*” (1.9.3), “*TensorFlow*” (2.10.1) e “*TensorFlow Hub*” (0.12.0); todas instaladas utilizando o comando *pip install* no terminal.

Figura 1 - Importando as bibliotecas



```
1  from mlforkids import MLforKidsImageProject
2  from google.colab import files
3  import os
4
```

Fonte: Os autores.

Iniciamos o código pela importação das bibliotecas, visualizado na Figura 1. Como utilizamos o modelo de treinamento criado no site *Machine Learning for Kids*, foi necessário utilizar o módulo “*mlforkids*”, sendo mais específico a classe “*MLforKidsImageProject*”, que é a classe referente ao treinamento de modelo com imagens. Além disso, foi importado o módulo “*files*” da biblioteca “*google.colab*”, visto que a utilizamos para importar as imagens ao Google Colab na execução do código. Ademais, foi importado ainda, a biblioteca “*os*” para utilizarmos algumas funcionalidades do sistema operacional.

Vale ressaltar que decidimos utilizar o Google Colab na execução do projeto pois as versões de algumas bibliotecas utilizadas pelo “*mlforkids.py*”, como *TensorFlow*, são muito antigas e por isso necessitam de uma versão mais antiga do Python, não sendo possível executar o script nas versões atuais da linguagem.

Figura 2 - Código principal do script

```
1
2 # mlforkids key
3 key = "c6740a40-937e-11ee-98c7-c79c85a0194ea582b4a0-1349-40ec-b4d9-aca12f697527"
4
5 print("Semáforo Livre ou Não?\n\n")
6
7 # training model
8 myproject = MLforKidsImageProject(key)
9 myproject.train_model()
10
11 print("\nEnvie uma imagem de sinal de trânsito e descubra se ele está liberado ou não.")
12
13
14 while True:
15     print("\n1. Enviar imagem.")
16     print("2. Teste")
17     print("3. Sair\n")
18
19     respMenu = input()
20     if respMenu=="1":
21         filenameUploaded = uploadImg()
22         result = analisarImg(filenameUploaded)
23         print(f"\n{filenameUploaded} : Semáforo {result[0]} - {result[1]}% de confiança!")
24     elif respMenu=="2":
25         test()
26     else:
27         break
```

Fonte: Os autores.

No código principal, visualizado na Figura 2, foi declarada a variável referente a chave do projeto no *Machine Learning for Kids* e realizado o treinamento do modelo de aprendizado de máquina com os comandos: “*MLforKidsImageProject(key)*” e “*myproject.train_model()*”, podendo ser verificadas as saídas do script nas figuras 3, 4, 5 e 6.

Buscando facilitar a interação código-usuário, foi desenvolvido um menu simples, na figura 7, com 3 opções para executar o código:

- Enviar imagem: O script pede que o usuário carregue uma imagem, ela é processada e o resultado é mostrado. Nessa etapa são utilizadas as funções definidas previamente: “*uploadImg*” e “*analisarImg*”.
- Teste: O script solicita a função “*test*” e itera sobre as imagens pré-definidas do diretório de testes, e ao final exibe a precisão do modelo.
- Sair: O script é finalizado.

Output:

Figura 3 - Script realiza o *download* das informações do projeto no site *Machine Learning for Kids*

```
Semáforo Livre ou Não?  
  
MLFORKIDS: Baixando informações sobre o projeto de aprendizado de máquina  
MLFORKIDS: Baixando imagens de exemplos no mlForKids para treinar o modelo... (isso vai demorar um pouco)
```

Fonte: Os autores.

Figura 4 - Script informa que baixou as imagens de treinamento das classes “green”, “red” e “yellow”

```
00789.jpg : imagem baixada! - green  
00792.jpg : imagem baixada! - green  
00815.jpg : imagem baixada! - green  
00825.jpg : imagem baixada! - green  
00844.jpg : imagem baixada! - green  
00867.jpg : imagem baixada! - green  
00868.jpg : imagem baixada! - green  
00891.jpg : imagem baixada! - green  
00001.jpg : imagem baixada! - red  
00068.jpg : imagem baixada! - red  
00072.jpg : imagem baixada! - red  
00091.jpg : imagem baixada! - red  
00127.jpg : imagem baixada! - red  
00128.jpg : imagem baixada! - red  
00129.jpg : imagem baixada! - red  
00302.jpg : imagem baixada! - red
```

Fonte: Os autores.

Figura 5 - Script informa que iniciou o processo de treinamento

```
MLFORKIDS: Definindo camadas a serem incluídas na rede neural  
MLFORKIDS: Iniciando treinamento do modelo... (isso também vai demorar)
```

Fonte: Os autores.

Figura 6 - Script informa que o treinamento do modelo foi finalizado

```
MLFORKIDS: Treinamento completo!
```

Fonte: Os autores.

Figura 7 - Script imprime o menu do usuário

```
Envie uma imagem de sinal de trânsito e descubra se ele está liberado ou não.  
  
1. Enviar imagem.  
2. Teste  
3. Sair
```

Fonte: Os autores.

Função *uploadImg()*:

Figura 8 - Código da função que realiza o *upload* das imagens

```
1 def uploadImg():  
2     uploaded = files.upload()  
3  
4     for filename in uploaded.keys():  
5         return filename  
6
```

Fonte: Os autores.

Quando digitado no menu a opção 1 de enviar imagem, foi solicitada a função “*uploadImg*”, com código encontrado na figura 8, onde utilizamos a função “*upload*” da biblioteca “*google.colab*” para carregar ao código no Google Colab as imagens solicitadas, conforme a figura 9.

Output:

Figura 9 - Script solicita o *upload* da imagem

```
Escolher arquivos 00097-green.jpg  
• 00097-green.jpg(image/jpeg) - 372426 bytes, last modified: 04/12/2023 - 100% done  
Saving 00097-green.jpg to 00097-green.jpg  
00097-green.jpg : Semáforo NÃO LIBERADO - 88% de confiança!
```

Fonte: Os autores.

Função *analisarImg()*:

Figura 10 - Código da função que analisa a imagem

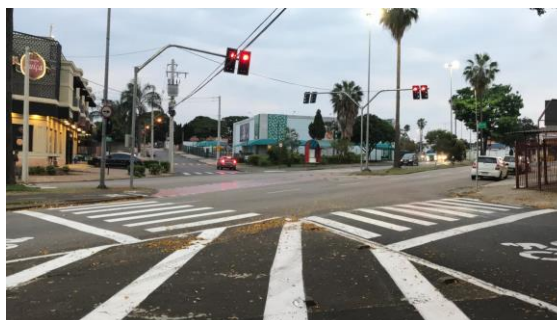
```
1 def analisarImg(filename):
2     # get image
3     original_stdout = sys.stdout
4     sys.stdout = io.StringIO()
5     demo = myproject.prediction(filename)
6     sys.stdout = original_stdout
7
8     label = demo["class_name"]
9     confidence = int(demo["confidence"])
10
11     if label=="green":
12         label = "LIBERADO"
13     else:
14         label = "NÃO LIBERADO"
15
16     return [label, confidence]
```

Fonte: Os autores.

Quando o script solicita a função “*analisarImg*”, código da figura 10, o programa recebe a imagem “*filename*” e o modelo de inteligência artificial faz a predição de acordo com o treinamento realizado pelo *Machine Learning for Kids*, e assim, atribui àquela imagem sua classe correspondente no modelo, bem como a taxa de confiança em sua resposta. Sendo a classificação “*green*” ou verde, é retornado ao script a informação de que naquela imagem o sinal está liberado, caso contrário, retorna-se que o semáforo não está liberado.

Exemplos de Teste:

Figura 11 – Imagem de exemplo com semáforo vermelho



Fonte: Prefeitura de Indaiatuba, 2021.

Output: “Semáforo NÃO LIBERADO – 99% de confiança!”

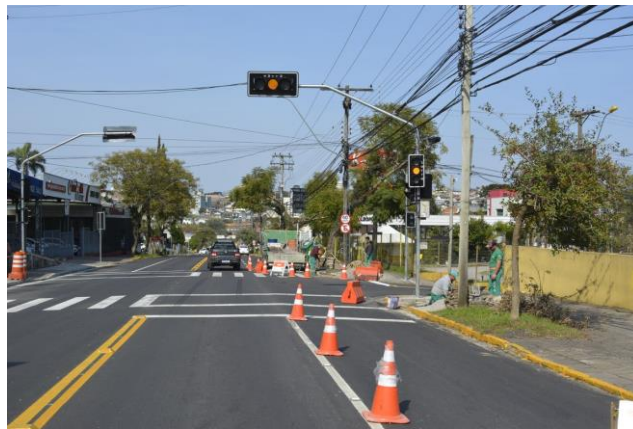
Figura 12 – Imagem de exemplo com semáforo vermelho



Fonte: Prefeitura de Indaiatuba, 2023.

Output: “Semáforo NÃO LIBERADO – 95% de confiança!”

Figura 13 – Imagem de exemplo com semáforo amarelo



Fonte: Prefeitura de Caxias do Sul, 2019.

Output: “Semáforo NÃO LIBERADO – 78% de confiança!”

Figura 14 – Imagem de exemplo com semáforo verde



Fonte: Prefeitura de Curitiba, 2023.

Output. “Semáforo LIBERADO – 78% de confiança!”

Função *test()*:

Figura 15 - Código da função que realiza o teste de imagens pré-definidas

```
1 def test():
2     testImages = os.listdir("/content/test_images")
3
4     points = 0
5     print("TESTE: analisando imagens do diretório test_images")
6
7     for img in testImages:
8         imgName = img.split("-")[0] + ".jpg"
9         situation = img.split("-")[1].split(".")[0]
10
11         if situation=="green":
12             label = "LIBERADO"
13         else:
14             label = "NÃO LIBERADO"
15
16         result = analisarImg("/content/test_images/" + img)
17         if label==result[0]:
18             print(f"{imgName} : Esperado - {label} ; Obtido - {result[0]} ({result[1]}%) ; SUCESSO!")
19             points+=1
20         else:
21             print(f"{imgName} : Esperado - {label} ; Obtido - {result[0]} ({result[1]}%) ; FALHA!")
22
23     print(f"\n\nAnalisando as {len(testImages)} imagens enviadas, a precisão final do algoritmo foi de {points/len(testImages)*100}%")
24
```

Fonte: Os autores.

Quando digitado a opção 2 no menu, é solicitada a função “*test*”, código encontrado na figura 15. Nesta função, utilizando a biblioteca “*os*”, o código lista as imagens de teste pré-definidas presentes no diretório “*test_imagens*”, no qual está contido na raiz do projeto. Em seguida, itera sobre cada imagem (todas padronizadas

por “nome-cor.jpg”), identifica a cor exibida no nome, guarda o status do semáforo “LIBERADO” ou “NÃO LIBERADO” na variável “*situation*”, de acordo com a cor, e executa a função “*analisarImagem*” armazenando na variável “*result*” o resultado da predição do modelo. Logo depois, o algoritmo realiza uma comparação entre a cor identificada nas duas variáveis para cada imagem do diretório, e imprime na tela o resultado esperado e o obtido pela inteligência artificial, avaliando o resultado com “SUCESSO!” ou “FALHA!” e emitindo a taxa de confiança dada na resposta do modelo. No final, o código imprime a precisão do modelo na predição em relação a todas as imagens analisadas.

Output:

Figura 16 - Script mostra o resultado da análise das imagens de teste

```
TESTE: analisando imagens do diretório test_images
00290.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (99%) ; SUCESSO!
02536.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (84%) ; FALHA!
02600.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (86%) ; FALHA!
01961.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (96%) ; SUCESSO!
02070.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (99%) ; SUCESSO!
02494.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (82%) ; FALHA!
00604.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (84%) ; FALHA!
02710.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (72%) ; FALHA!
02520.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (77%) ; SUCESSO!
02490.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (97%) ; SUCESSO!
01212.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (98%) ; SUCESSO!
02240.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (90%) ; FALHA!
02517.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (98%) ; SUCESSO!
02936.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (66%) ; SUCESSO!
02855.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (79%) ; SUCESSO!
02728.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (97%) ; SUCESSO!
02642.jpg : Esperado - NÃO LIBERADO ; Obtido - LIBERADO (93%) ; FALHA!
00376.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (98%) ; SUCESSO!
01873.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (99%) ; SUCESSO!
01826.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (99%) ; SUCESSO!
00060.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (90%) ; SUCESSO!
00955.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (91%) ; SUCESSO!
02696.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (91%) ; SUCESSO!
02448.jpg : Esperado - LIBERADO ; Obtido - NÃO LIBERADO (92%) ; FALHA!
02462.jpg : Esperado - NÃO LIBERADO ; Obtido - NÃO LIBERADO (87%) ; SUCESSO!
02344.jpg : Esperado - LIBERADO ; Obtido - LIBERADO (77%) ; SUCESSO!
```

Fonte: Os autores.

Figura 17 – Script exibe a precisão final do algoritmo

```
Analisando as 46 imagens enviadas, a precisão final do algoritmo foi de 73%
```

Fonte: Os autores.

4. Discussão dos Resultados

Um dos pontos a ser destacado é a alta precisão das respostas por parte do modelo de aprendizado de máquina nas imagens do diretório de teste, que ficou em torno de 73%. No entanto, inicialmente esse valor estava aproximadamente 60%, e buscando entender o motivo, verificamos que usávamos apenas duas classes: “verde” e “vermelho”, com isso acrescentamos ao modelo de treinamento a classe “amarelo”, o que fez com que a precisão do modelo aumentasse. Apesar de utilizarmos na identificação as cores das classes “verde”, “vermelho” e “amarelo”, optamos por manter apenas as classificações “liberado” ou “não liberado” para que a tomada de decisão do modelo ficasse mais objetiva e minimizasse possíveis erros.

Outro ponto importante trata-se das imagens, acreditamos que outra fonte de erro pudesse ter sido causada pela quantidade de cores nas imagens da base de dados utilizada. O que pode ter levado a uma certa confusão no momento de diferenciar os objetos e encontrar as cores do semáforo. Entretanto, não encontramos outra base de dados com imagens que ofereçam maior foco aos semáforos em si, e que estejam de forma acessível para utilizarmos. Em relação as imagens de exemplo testadas: figuras 11, 12, 13 e 14, observamos que em geral o modelo emitiu uma avaliação correta e com alta taxa de confiança. Nas figuras 13 e 14 a confiança ficou abaixo de 80%, mas é válido ressaltar que as imagens utilizadas não têm vínculo algum com a base de dados utilizada no treinamento, tratando-se de imagens reais fotografadas de semáforos de cidades brasileiras.

Referências

Machine Learning for Kids. Disponível em: <https://machinelearningforkids.co.uk/>.

Traffic Light Detection Dataset. Disponível em: <https://www.kaggle.com/datasets/wjybuqi/traffic-light-detection-dataset>.

Imagens selecionadas para o projeto - Diretório do Projeto: Disponível em: <https://www.cin.ufpe.br/~wnjls/projeto%20IC/>.

Prefeitura de Indaiatuba, 2021. **Semáforo no cruzamento da avenida Conceição.** Disponível em: <https://www.indaiatuba.sp.gov.br/relacoes-institucionais/impressa/noticias/30208/>.

Prefeitura de Indaiatuba, 2023. **Semáforo no cruzamento da avenida Presidente Kennedy.** Disponível em: <https://pirapop.com.br/indaiatuba-implanta-sistema-semaforo/>.

Prefeitura de Caxias do Sul, 2019. **Semáforo na avenida São Leopoldo.** Disponível em: <https://caxias.rs.gov.br/noticias/2019/08/semaforo-em-esquina-da-avenida-sao-leopoldo-entra-em-operacao-neste-sabado>.

Prefeitura de Curitiba, 2023. **Semáforo na rua Brigadeiro Franco.** Disponível em: <https://transito.curitiba.pr.gov.br/noticias/prefeitura/zelo-pela-cidade-setran-instala-semaforos-em-3-esquinas-movimentadas/69951>.