Detecção de veículos próximo ao Hospital de Clinicas HC CWB

UniBrasil – Curitiba/PR

Prática profissional em Aprendizado de Máquina – Mozart Hasse

Bruno Gonçalves e Karla Valeria

# O Projeto

Desenvolver um sistema que seja capaz de detectar e contar veículos de imagens aéreas auxiliando em diversas áreas, como: estacionamentos, controle interno de veículos para condomínios, otimização de tráfego, entre outras.

# Melhorias Futuras

Realizar um dashboard de acompanhamento por tempo, para verificar quantidade de vagas disponíveis com a integração de um front-end para melhor visualização do usuário.

# Etapas

O seguinte projeto foi dividido em 5 etapas:

1. Aquisição de dados e download de dados semelhantes ao escopo.
2. Organização de *dataset* para treinamento e validação
3. Treinamento
4. Validação de discussão sobre resultados
5. Detecção de veículos
6. Detecção e contagem de veículos

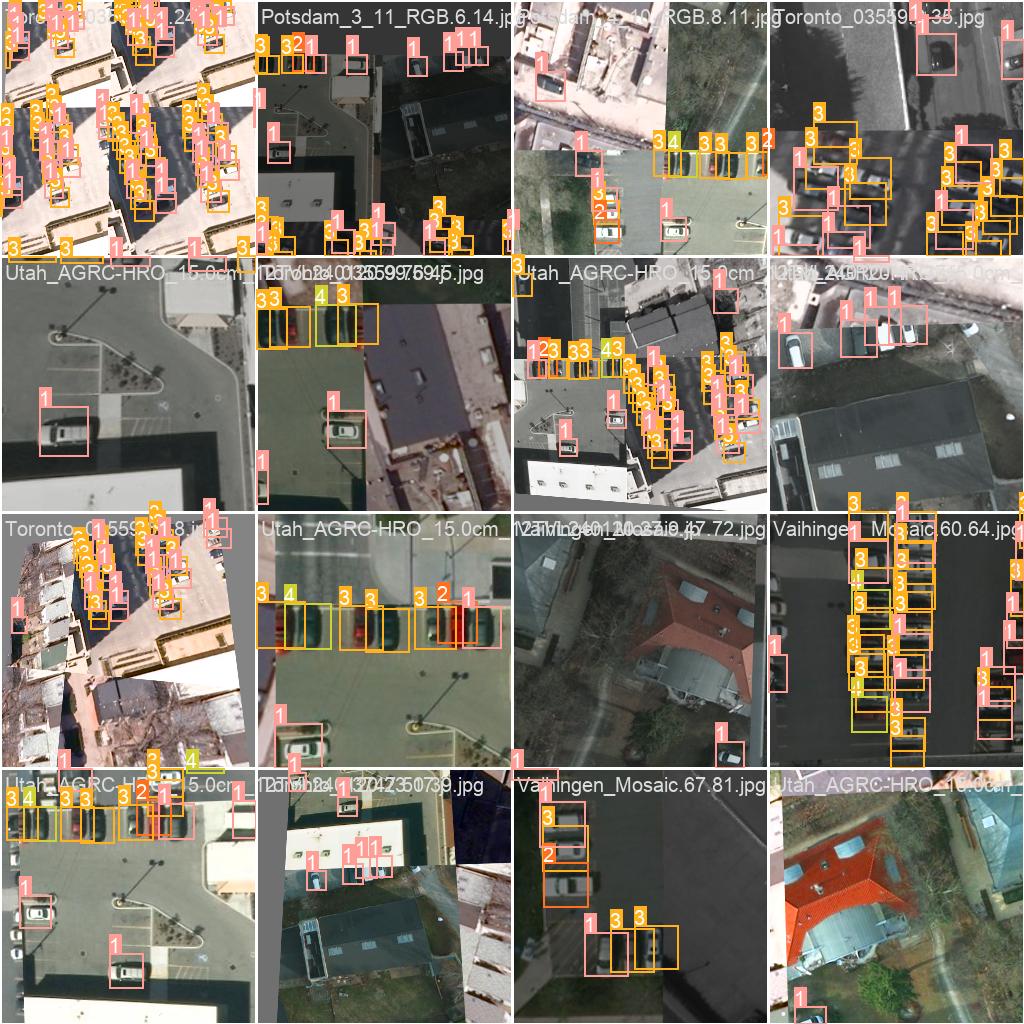
Após a pesquisa e conhecimento de diversas versões da arquitetura YOLO ( you only look once ) e por se tratar de uma técnica de detecção de objetos que se destaca pela sua rapidez e eficiência, trata como um problema de regressão direta que mapeia a imagem de entrada para uma série de caixas delimitadoras e classes associadas. No qual foi baseado em uma arquitetura FCN ( Fully Convolutional Network) que diferentemente das CNNs tradicionais que usam camadas densas (Fully connected layers) as FCNs usam apenas camadas convolucionais, permitindo que a rede produza um mapa de características que mantém as relações espaciais da imagem de entrada.

No decorrer das pesquisas, chegamos a seguinte url: <https://pjreddie.com/darknet/yolo/>, contendo diversas versões da arquitetura proposta por Joseph Redmon, chegando a mais atualizada ultralytics/yolov9.

## Aquisição de dados e download de dados semelhantes ao escopo

Ao realizar a extração do dataset cowc-m-everything, o mesmo continha outras pastas dentro, ambas contendo a imagem original, a imagem com os carros selecionados e um arquivo txt contendo as coordenadas dos veículos. Um total de 29,549.00 imagens, 143,580.00 marcações de carros.

Link para download do dataset: <https://gdo152.llnl.gov/cowc/download/>



## Organização de *dataset* para treinamento e validação

Após a extração do *dataset*, realizamos a identificação e separação das imagens e *labels* da seguinte forma:

DATASET DE TREINO:

* Dataset/train/images
* Dataset/train/labels

DATASET DE VALIDAÇÃO:

* Dataset/validation/images
* Dataset/validation/labels

Aqui identificamos os *labels* e imagens e separamos:

imagens = []

labels = []

for folder in folders:

for i in glob.glob(folder + "\\*.txt"):

temp = os.path.dirname( i ) + '\\'+ os.path.splitext( os.path.basename( i ) ) [0] + ".jpg"

if not os.path.exists(temp) :

raise Exception("deu pau")

imagens.append( temp )

labels.append( i )

df = pd.DataFrame( data = {'image\_id': imagens, "labels": labels } )

label\_total = pd.DataFrame()

for i, row in df.iterrows():

label = pd.read\_csv( row.labels , sep = " ", names = ['cls','a' , 'b', 'c', 'd'])

label['path'] = row.labels

label\_total = pd.concat( [label\_total, label ] , ignore\_index=True )

print(i, "\\", df.shape[0], end= "\r")

for i in label\_total.cls.unique().tolist():

print( label\_total[ label\_total.cls == i ].iloc[0] )

gkf = GroupKFold(n\_splits = 5)

df['fold'] = -1

for fold, (train\_idx, val\_idx) in enumerate(gkf.split(df, groups = df.image\_id.tolist())):

df.loc[val\_idx, 'fold'] = fold

val\_df = df[df['fold']==4]

val\_df.head()

Aqui criamos as pastas para sepração:

def create\_folder(df, labels\_path, images\_path):

for index, row in tqdm(df.iterrows(), total=len(df)):

shutil.copy( row.image\_id , os.path.join( images\_path, os.path.basename( row.image\_id ) ) )

shutil.copy( row.labels , os.path.join( labels\_path, os.path.basename( row.labels ) ) )

create\_folder( df, r"C:\Users\Bruno\yolo bruno\dataset\train\labels", r"C:\Users\Bruno\yolo bruno\dataset\train\images" )

create\_folder( val\_df, r"C:\Users\Bruno\yolo bruno\dataset\validation\labels", r"C:\Users\Bruno\yolo bruno\dataset\validation\images" )

Aqui realizamos a separação por classes:

classes = [ 'car1', 'car2', 'car3', 'car4', 'car5' ]

data = dict(

train = r"C:\Users\Bruno\yolo bruno\dataset\train",

val = r"C:\Users\Bruno\yolo bruno\dataset\validation",

nc = 5,

names = classes

)

with open('./yolo\_bruno.yaml', 'w') as outfile:

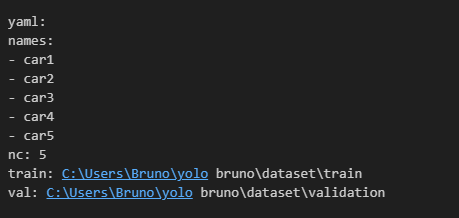
yaml.dump(data, outfile, default\_flow\_style=False)

f = open('./yolo\_bruno.yaml', 'r')

print('\nyaml:')

print(f.read())

No qual ficou definido o arquivo yaml:



## Treinamento

Para realizar o treinamento do modelo, conforme o arquivo *train.ipynb* no código *model.train*, foram definidos os seguintes parâmetros:

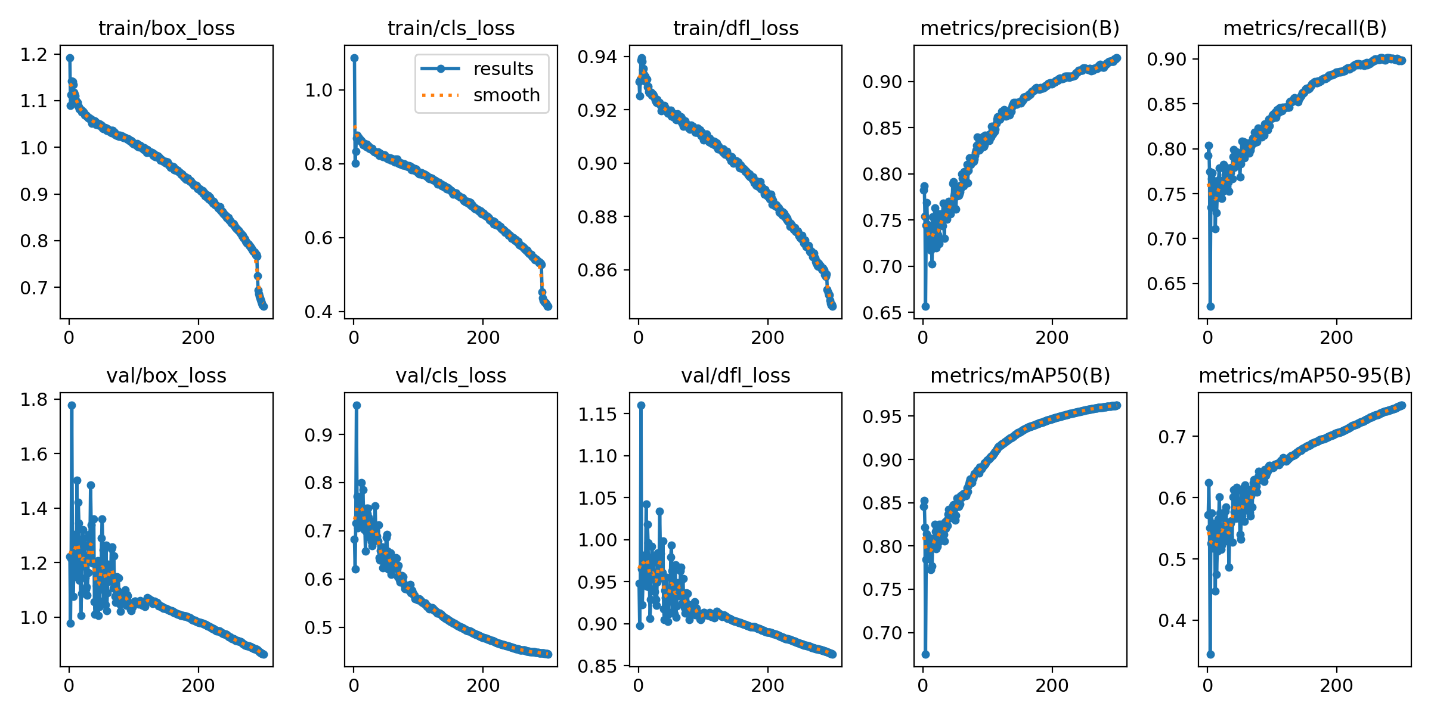
* epochs=300 - após alguns testes, decidimos que 300 épocas mostrou o melhor resultado;
* imgsz=256 - definição do tamanho da imagem para 256x256 pixels;
* batch=260 - foi definido o tamanho do batch em 260;
* resume=True - para retomar a partir do ponto de verificação anterior;
* save\_period = 1 foi definido a frequência com que os pontos de verificação serão salvos durante o treinamento;
* save=True – indica que os pontos de verificação devem ser salvos;
* workers=12 - número de subprocessos usados para carregar os dados;
* hsv\_v = 0.6 - fator de aumento da variação do valor no espaço de cores HSV (Hue, Saturation, Value);
* hear = 0.4 - definido o grau de cisalhamento;
* perspective = 0.001 - definido o grau de transformação de perspectiva aplicada como parte do aumento de dados.

O sistema rodou em um ambiente que possui 16GB de RAM DDR5, uma GPU nvidia RTX 3060 de 12GB, processador Ryzen 5 7600X, após pequenos treinamentos observamos que o batch\_size confortável por questões de memória RAM e memória da GPU, configuramos o batch\_size para 260, executando a etapa então de treinamento por 11:30 horas.

Abaixo um gráfico que representa a matrix de confusão do modelo finalInterface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

## Validação e discussão sobre resultados

Aqui temos os gráficos que mostram os seguintes resultados:

Abaixo temos a descrição dos valores:

**No treinamento (train/):**

* train/box\_loss = referência de perda associada a precisão das caixas delimitadoras ( bouding boxes) previstas pelo modelo. É uma medida de quão bem o modelo está ajustando as caixas

delimitadoras aos objetos no treinamento.

* Train/cls\_loss = referência à perda de classificação. É a medida de quão bem o modelo está classificando os objetos nas suas respectivas categorias durante o treinamento.
* Train/dfl\_loss = é uma métrica específica usada em algumas versões do YOLO. É uma perda adicional que ajuda a melhorar a precisão das caixas delimitadoras, considerando a distribuição dos valores previstos.

**No métricas (metrics/):**

- metrics/precision(B) = é a proporção de verdadeiros positivos em relação ao total de positivos previstos. Uma alta precisão indica poucas falsos positivos.

- metrics/recall(B) = é a proporção de verdadeiros positivos em relação ao total de positivos reais (objetos presentes nas imagens). Um alto recall indica poucas falsos negativos.

**Durante a validação (val/):**

- val/box\_loss = Semelhante ao train/box\_loss, mas avaliado nos dados de validação. Mede a precisão das caixas delimitadoras previstas pelo modelo durante a validação.

- val/cls\_loss = Semelhante ao train/cls\_loss, mas avaliado nos dados de validação. Mede a precisão da classificação do modelo durante a validação.

- val/dfl\_loss = Semelhante ao train/dfl\_loss, mas avaliado nos dados de validação. É a perda de DFL durante a validação.

**Nas métricas de validação (metrics/):**

- metrics/mAP50(B) = Média da precisão média (mean Average Precision) com um threshold de IoU (Intersection over Union) de 50%. Mede a precisão da detecção de objetos considerando um nível de sobreposição de 50%.

- metrics/mAP50-95(B) = Média da precisão média (mean Average Precision) calculada em vários níveis de IoU (de 50% a 95%, com incrementos de 5%). Isso fornece uma visão mais completa da performance do modelo em diferentes níveis de sobreposição.

## Detecção de veículos

Nessa etapa, chegamos no resultado do nosso projeto. De acordo com o dataset que treinamos o nosso modelo; por ele possuir diversas imagens para validação, chegamos a um resultado de 92% de acerto, conforme imagem abaixo:

Desenho de um prédio

Descrição gerada automaticamente com confiança baixa

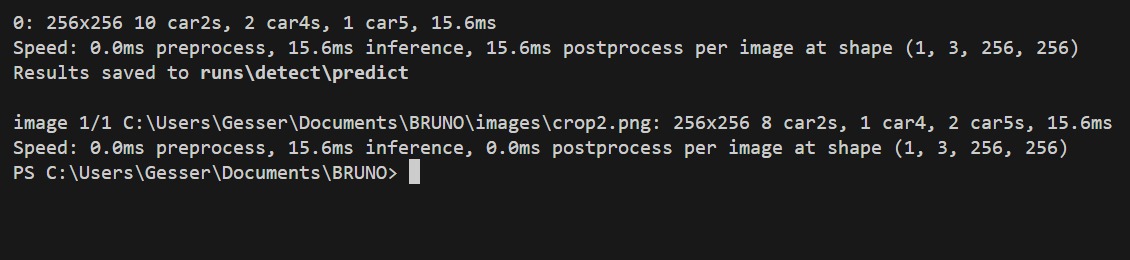
Nessa imagem o sistema detectou:

car2 = 8

car4 = 1

car5 = 2

## Contagem de veículos



# Resultados

Os resultados apresentados no decorrer da documentação demonstraram a eficácia de nossos esforços trazendo um resultado de 92% de precisão no reconhecimento de carros presentes na imagem. Esse resultado foi possível por meio de uma máquina com uma configuração excelente que permitiu a execução de 300 épocas com 260 batchs. Porém, até chegar nesse resultado, foram ocorrendo diversos imprevistos como o Early stopping sem uma previsão correta, e com isso fomos ajustando os parâmetros até chegar em um resultado compatível ao nosso objetivo. Sendo assim, vale ressaltar que o projeto foi um sucesso.