Otimização da Colheita de Palmas: Modelo de Aprendizado de Máquina para Identificação de Folhas para Corte

Euler Lima, Gustavo Nunes, José Júnio, Rogério Chaves

***Abstract*—This report describes the steps involved in training a model to classify the possibility of cutting elephant ear palm. The algorithm was developed in python and the images for the dataset were acquired via field research.**

***Index Terms*—Palmas, Colheira, Automação, Inteligência artificial**

I. Introdução

ESTE artigo apresenta um projeto focado no desenvolvimento de um dataset para treinar um modelo de Aprendizado de Máquina na identificação de palmas "orelha de elefante" aptas para corte. A iniciativa visa aprimorar a eficiência e segurança na colheita, aproveitando os avanços tecnológicos para minimizar desperdícios.

A região semiárida do Nordeste do Brasil, com sua grande concentração de pequenos agricultores, é o cenário propício para a implementação dessa solução. Ao modelar um algoritmo preciso e construir uma base de dados robusta, o projeto não apenas promete otimizar a produção, mas também contribuir significativamente para melhorar as condições de trabalho e segurança dos agricultores, alinhando-se aos princípios do desenvolvimento sustentável.

A. Etapas do desenvolvimento

1) ***Fotografar****:* No dia 16 de outubro de 2023 na Universidade Estadual do Sudoeste da Bahia foi realizada uma pesquisa de campo com a supervisão dos professores Geraldo Pereira Rocha Filho e Hélio Lopes Santos com o intuito de aprender como se comportam as palmas e aprender sobre o comportamento das mesmas. Nessa pesquisa de campo foram apresentados a palma de característica miúda e de característica orelha de elefante. Os estudantes fizeram uso dos seus smartphones para realizar a captura e todos compartilharam em uma pasta do google drive para que todos os grupos pudessem ter acesso e a partir disso desenvolver o dataset.

2) ***Tratamento das imagens****:* As imagens foram tratadas a partir das seguintes etapas:

- Remoção de artefatos usando o Photoshop para todas as imagens;
- Normalização das imagens para o padrão RGB;
- Redimensionamento das imagens para 640x640;
- Normalização das imagens do para a escala de 0 a 1;
- Augmento de dados.

Para o desenvolvimento da primeira etapa (remoção de artefatos), realizou-se a identificação do artefato: O primeiro passo, é observar a imagem e detectar possíveis artefatos que possam interferir nos pixels.

Fig. 1. encontrada uma aranha que poderia afetar o padrão de cores da planta.

Posteriormente utilizou a ferramenta máscara para corte (caneta) do Photoshop, que realiza a seleção do artefato e após concluir o contorno cria uma máscara de seleção na qual pode ser removida da imagem.

Fig. 2. Uso da máscara do photoshop.

E por fim, após exclusão do artefato no passo anterior, será selecionado uma parte da imagem que tenha cores coerentes com a parte faltante, efetua-se a cópia, e posicionada a camada copiada atrás da camada da imagem original. E para suavizar a demarcação deixada pelo recorte do artefato, utilizamos a ferramenta "borrar" do Photoshop para suavizar e integrar o degradê das cores.

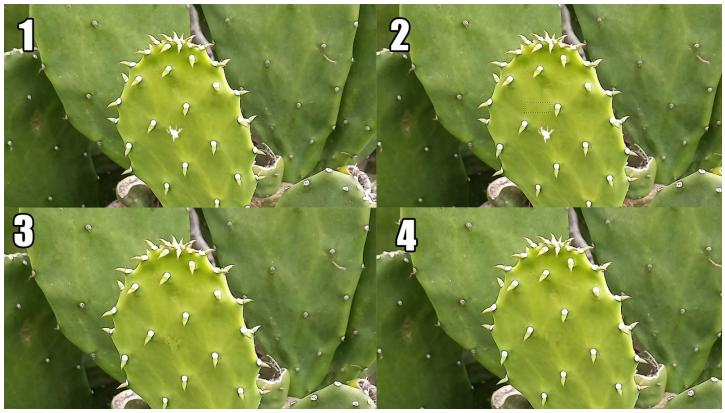

Fig. 3. Mesclagem de camadas.

Para o desenvolvimento da segunda (padrão RGB) e terceira etapa (redimensionamento), primeiro criou-se um diretório para destinar as imagens. Após isso, carregou-se as imagens e posteriormente fez o uso de um laço de repetição com o objetivo de converter as cores de ”BGR” para ”RGB”. Posteriormente, dentro do mesmo laço de repetição, utilizou-se o método ”resize” da biblioteca ”cv2” que tem como argumentos a imagem a ser alterada e as dimensões finais e assim foram feitas as etapas de redimensionamento e mudança no padrão de cor.

Algorithm 1 Importando e tratando as imagens

```
import Augmentor
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os
import shutil

elefante_imagens = os.listdir('Orelha de elefante editada')
datasets_dir = 'imagens_redimensionadas_normalizadas
/Orelha_elefante'
if not os.path.exists(datasets_dir):
  os.makedirs(datasets_dir)  ▷ Criação do diretório que guardará as imagens modificadas.
for elefante_imagem in elefante_imagens:
  image = cv2.imread(f'Orelha de elefante editada/{elefante_imagem}')
  image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  ▷ Conversão das cores BGR para RGB
  image = cv2.resize(image, (640, 640))  ▷ Redimensiosamento para 640x640
  image = image / 255.0  ▷ Normalização para imagem para escala de 0 a 1
  plt.imsave(fname=f'{datasets_dir}/{elefante_imagem}', arr=image)  ▷ Salvar imagem
```

Para o desenvolvimento da quarta etapa (Normalização das imagens do para a escala de 0 a 1) é necessário aumentar a variedade do dataset, e isso pode ser feito por meio de técnicas como rotação, espelhamento e zoom.

Algorithm 2 Escalando o dataset

```
pasta_redimensionadas_normalizadas = 'imagens_redimensionadas_normalizadas/Orelha_elefante'
p = Augmentor.Pipeline(pasta_redimensionadas_normalizadas)
p.rotate(probability=0.7, max_left_rotation=10, max_right_rotation=10)
p.zoom_random(probability=0.7, percentage_area=0.8)
p.flip_left_right(probability=0.5)
p.flip_top_bottom(probability=0.5)
p.sample(100)
p.process()
os.rename('imagens_augmentadas', 'dataset')  ▷ Renomeação do diretório com as imagens preprocessadas
```

*3) **Anotação das imagens**:* Após a etapa de tratamento das imagens, juntamente com a augmentação das mesmas, realizou-se a anotação das imagens. Cada palma presente nas imagens foi analisada e caracterizada como adequada ou inadequada para corte. Para essa tarefa, utilizou-se a ferramenta online de código aberto chamada ”Make Sense”. Dentro dessa ferramenta, foi possível examinar cada imagem e criar uma marcação (box) para categorizar cada planta de palma quanto à sua viabilidade de colheita, conforme representado na imagem de exemplo abaixo:

Fig. 4. Imagem das palmas previamente tratada porem sem a anotação feita no Make Sense

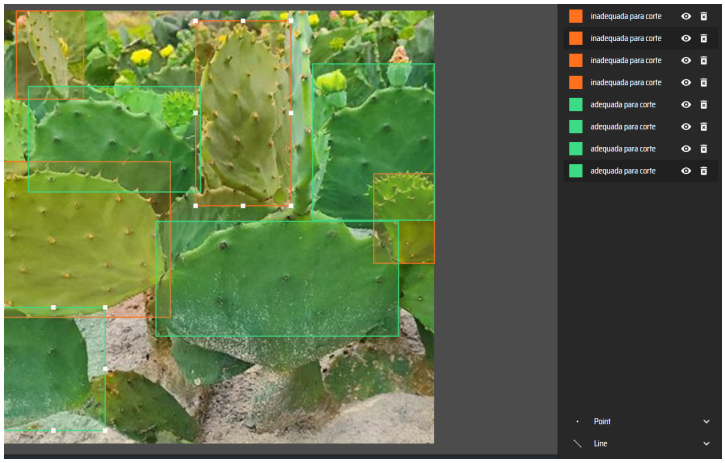

Fig. 5. Anotações feitas em uma imagem de exemplo, utilizando a ferramenta Make Sense.

Para assegurar um padrão em todas as anotações, realizou-se uma pesquisa, observando as características das palmas consideradas adequadas e inadequadas para cortes. Essas características incluem:

- Tamanho e Maturidade: Considerou-se o tamanho das palmas detectadas, sendo estabelecido um tamanho mínimo para cortar as palmas maduras.
- Coloração: A coloração das palmas foi utilizada como um indicador de maturidade. Palmas maduras podem apresentar cores mais escuras ou diferentes.
- Textura: Analisou-se a textura da pele das palmas, sendo que a textura pode variar entre palmas maduras e jovens.
- Formato: O formato da palma foi considerado, observando que palmas maduras podem ter um formato mais arredondado ou oval, enquanto as jovens podem apresentar um formato mais alongado.
- Densidade de Espinhos: A densidade de espinhos nas palmas foi avaliada como um indicador de idade, pois palmas maduras tendem a ter menos espinhos do que as jovens. A contagem de espinhos foi utilizada como um indicador adicional.

*4) **Construção do dataset, separando-o em teste, treino e validação**:* Para a etapa de construção do dataset, realizou-se a distribuição das imagens, previamente classificadas, da seguinte maneira, 70/100 para o conjunto de treinamento, 20/100 para o conjunto de teste e 10/100 para o conjunto de validação. O total de imagens utilizado foram de 156. Desse total, 31 imagens foram reservadas para o conjunto de teste, 16 imagens foram alocadas para o conjunto de validação e o restante para treino.

A distribuição dessas imagens foi estrategicamente planejada, considerando a utilização de imagens com maiores extremidades relacionado a classificação de ”inadequadas para corte” e ”adequadas para corte” no conjunto de teste, enquanto as imagens restantes foram destinadas ao conjunto de validação. As imagens que abrangiam de maneira mais equilibrada as categorias de ”adequadas para corte” e ”inadequadas para corte” foram designadas para o conjunto de treinamento, visando uma representação abrangente e equitativa no processo de treinamento do modelo. Este procedimento foi adotado para garantir uma análise robusta do desempenho do modelo diante de diversas condições e tipos de imagens.

Algorithm 3 Construção do dataset

```
from sklearn.model_selection import train_test_split
import numpy as np
import os
import shutil
PATH = 'dataset/Orelha_elefante' ▷ Obter uma lista com os nomes dos arquivos.
filenames = os.listdir(PATH)
print(f'Número de arquivos: len(filenames)')
filenames = list(map(lambda name: name.split('.')[0], filenames)) #Remover a extensão .jpg e .txt.
filenames = list(set(filenames))  ▷ Remover as duplicações.
train, rest = train_test_split(filenames, test_size= 0.3, random_state=42)
test, val = train_test_split(rest, test_size= 0.5, random_state=42)  ▷ Separação entre treino, validação e teste.
print(f'Número de arquivos de treino: len(train)')
print(f'Número de arquivos de validação: len(val)')
print(f'Número de arquivos de teste: len(test)')
```

Após carregar as imagens de teste, treino e validação de forma aleatória, foram criado os diretórios de treino, teste e validação e enviado as imagens para a mesma.

Algorithm 4 Para o treino

```
train_dir = 'final_dataset/train'  ▷ Criar diretório de treino.
if not os.path.exists(train_dir):
    os.makedirs(train_dir)
for train_file in train:  ▷ Mover arquivos de treino para o diretório de treino.
    img = f'PATH/train_file.jpg'
    annotation = f'PATH/train_file.txt'
    shutil.move(img, train_dir)
    shutil.move(annotation, train_dir)
```

Algorithm 5 Para o teste

```
test_dir = 'final_dataset/test'  ▷ Criar diretório de teste.
if not os.path.exists(test_dir):
    os.makedirs(test_dir)
for teste_file in test:  ▷ Mover arquivos de teste para o diretório de teste.
    img = f'PATH/teste_file.jpg'
    annotation = f'PATH/teste_file.txt'
    shutil.move(img, test_dir)
    shutil.move(annotation, test_dir)
```

Algorithm 6 Para a validação

```
val_dir = 'final_dataset/val' ▷ Criar diretório de validação
if not os.path.exists(val_dir):
    os.makedirs(val_dir)
for val_file in val: ▷ Mover arquivos de validação para o diretório de validação.
    img = f'PATH/val_file.jpg'
    annotation = f'PATH/val_file.txt'
    shutil.move(img, val_dir)
    shutil.move(annotation, val_dir)
```

Após a separação, foi percebido que necessitaria revisar essas imagens de modo que no diretório de treino validação fossem compostas por imagens que possuam anotações com uma predominância extrema seja de "adequada para corte" ou "não adequada para corte" para averiguar melhores resultados e assim foi feito o remanejo dessas imagens.

*5) **Treinamento**:* Antes de iniciar o treinamento, é necessário instalar algumas dependências, como o ultralytics que é um modelo de última geração (SOTA) que se baseia no sucesso das versões anteriores do YOLO e apresenta novos recursos e aprimoramentos para aumentar ainda mais o desempenho e a flexibilidade. Além disso, foi instalado também o yolov8 que foi projetado para ser rápido, preciso e fácil de usar, o que o torna uma excelente opção para uma ampla gama de tarefas de detecção e rastreamento de objetos, segmentação de instâncias, classificação de imagens e estimativa de pose.

Algorithm 7 Instalando dependências

```
!pip install ultralytics
!wget https://github.com/ultralytics/assets/releases/download/v0.0.0/y
```

Posteriormente foram realizada as importações necessárias para realizar o treinamento.

Algorithm 8 Importações necessárias para o treinamento.

```
from IPython.display import display
from PIL import Image
from ultralytics import YOLO
import os
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
model = YOLO('yolov8m.pt')
results = model.train(data='palmas.yaml', epochs=100, imgsz=640)
```

Algorithm 9 Treinamento

```
test_images_path = os.listdir('final_dataset/palmas/images/test')
images_dir = 'images_result if notos.path.exists(images_dir) :
    os.makedirs(images_dir)
  for image in test_images_path:
    results = model(f'final_dataset/palmas/images/test/image', conf=0.5) ▷ Predição na imagem.
  for r in results: ▷ Mostrar os resultados.
    im_array = r.plot() # plot a BGR numpy array of predictions
    im = Image.fromarray(im_array[..., ::-1])
     #RGB PIL image.
    image_name = image.split('/')[-1]
    im.show()
     #Mostrar a imagem.
    im.save(f'images_dir/image_name') ▷ Salvar a imagem
```

B. Resultados

A análise dos dados obtidos revela insights valiosos sobre o desempenho do modelo em relação à precisão em diferentes conjuntos de dados. A seguir, apresentamos os resultados por meio de gráficos, proporcionando uma visão mais clara da relação entre a quantidade de imagens e a precisão alcançada.

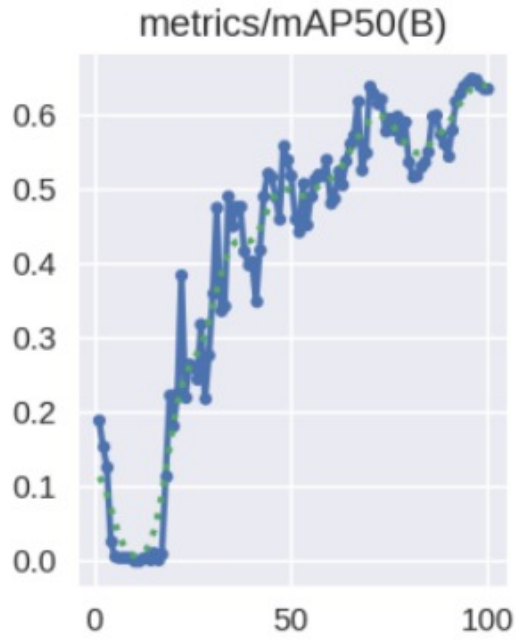

Fig. 6. Área sobre a curva AUC onde threshold da IOU é igual ou maior que 50.

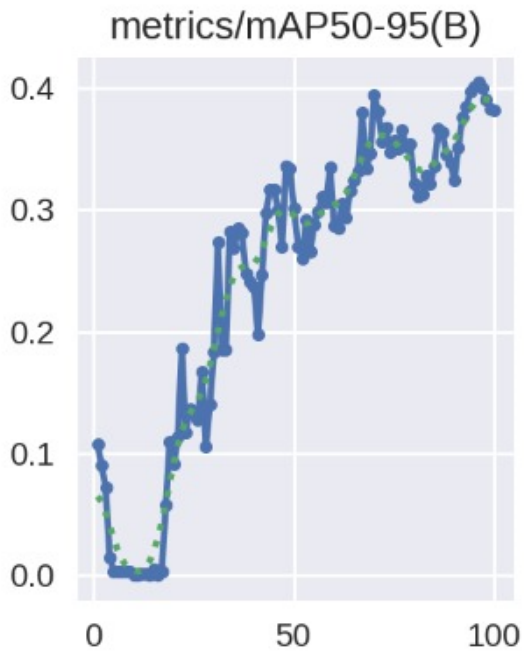

Fig. 7. Área sobre a curva AUC onde threshold da IOU é maior que 50 e menor que 90.

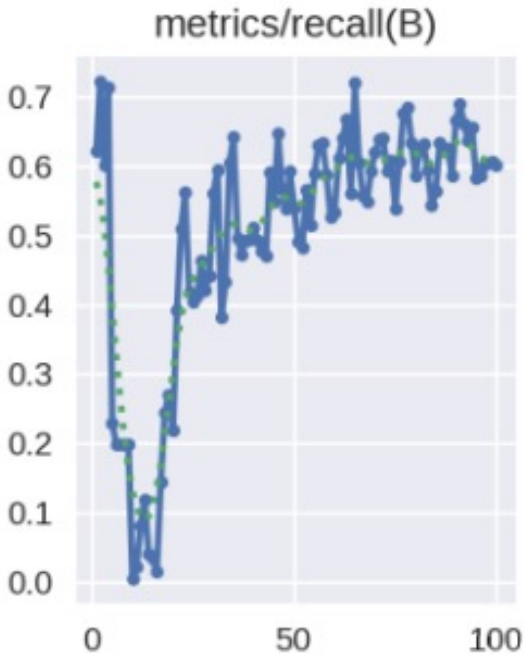

Fig. 9. Medir o encontro de todos os aspectos positivos (adequadas para corte).

Fig. 8. Medir a precisão de suas previsões. Ou seja, a porcentagem de suas previsões que está correta.

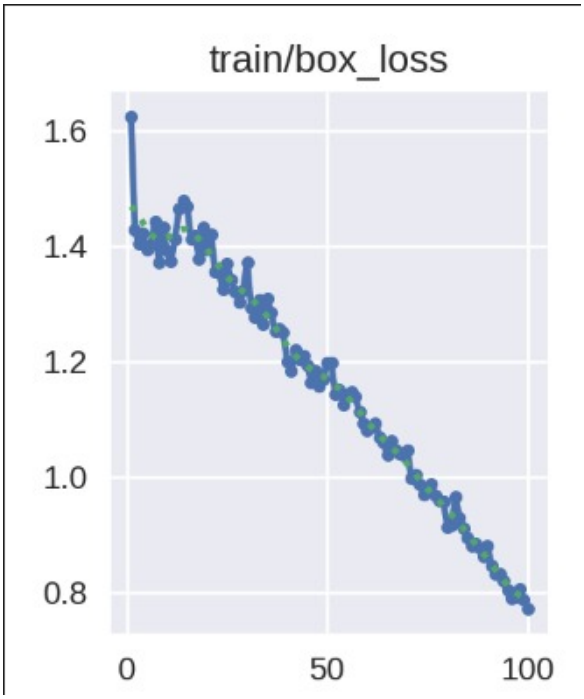

Fig. 10. Perda que mede o quão próximas as caixas delimitadoras previstas estão para o objeto de verdade no conjunto de treino.

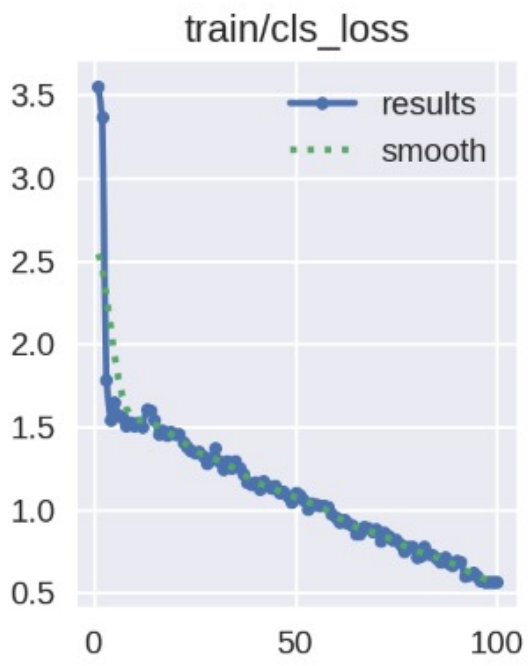

Fig. 11. Uma perda do conjunto de treino que mede a correção do classificação de cada caixa delimitadora prevista. Essa perda é geralmente chamada de perda de entropia cruzada.

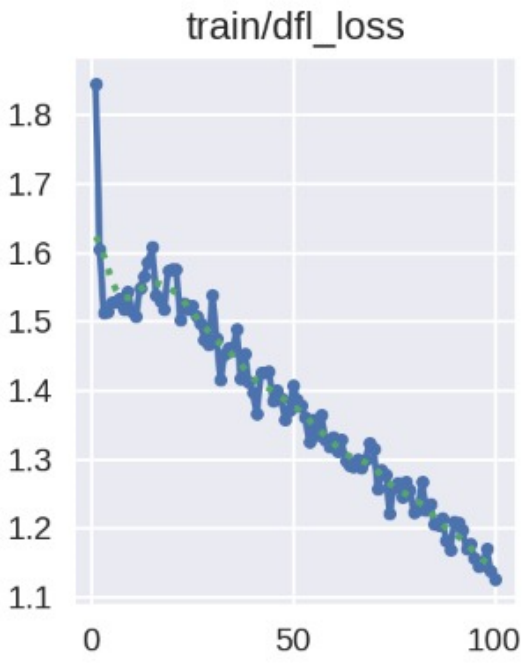

Fig. 12. O processo de aprendizagem da distribuição espacial das características para o conjunto de treino.

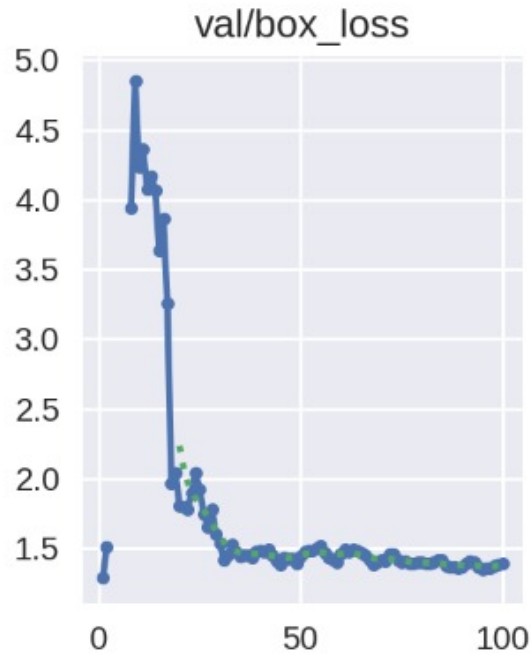

Fig. 13. Perda que mede o quão próximas as caixas delimitadoras previstas estão para o objeto de verdade no conjunto de validação.

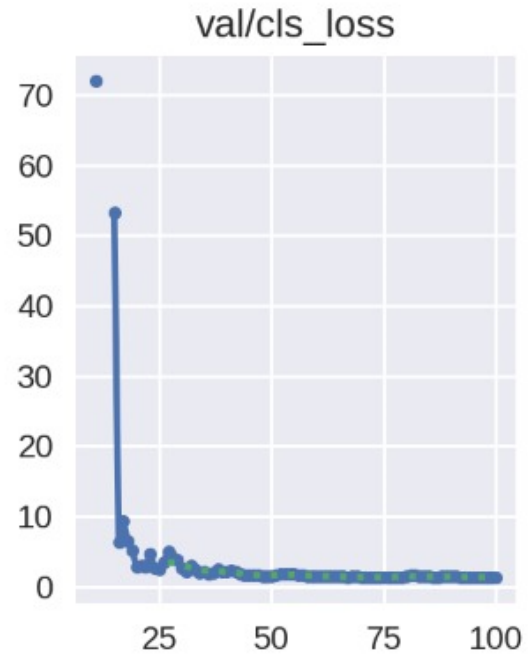

Fig. 14. Uma perda do conjunto de validação que mede a correção do classificação de cada caixa delimitadora prevista. Essa perda é geralmente chamada de perda de entropia cruzada..

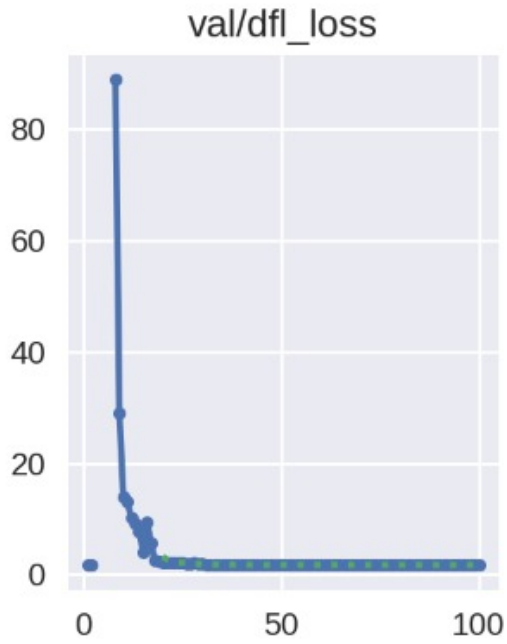

Fig. 15. O processo de aprendizagem da distribuição espacial das características para o conjunto de validação.

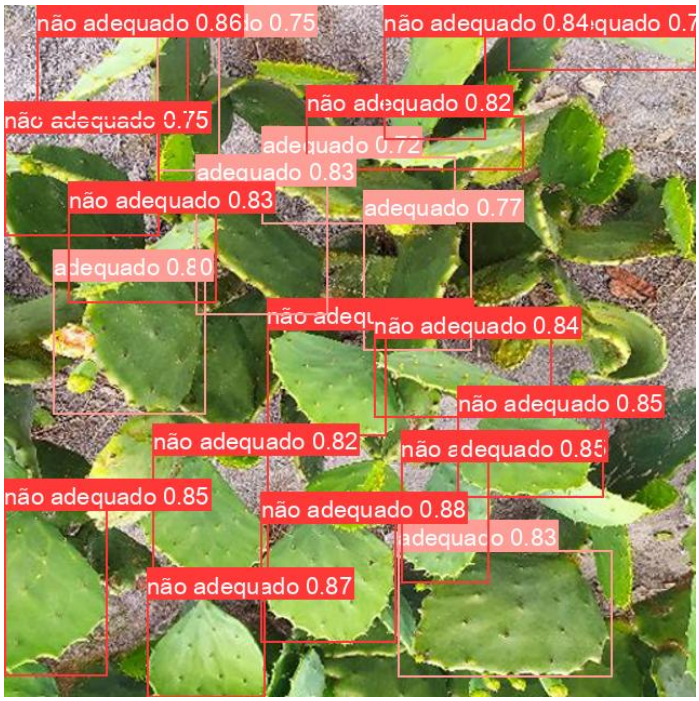

Fig. 16. Um dos resultados no pós treinamento.

AGRADECIMENTOS

Os autores gostariam de agradecer ao professor Geraldo Rocha Pereira Filho por ter proposto este trabalho de modo que obtiveram experiência em um trabalho em campo, no trabalho colaborativo e em um trabalho desafiador.

REFERENCES

[1] Ultralytics YOLO - Model Training, https://docs.ultralytics.com/modes/train/