Roteiro de Modelos com Imagens

Este roteiro apresenta o modelo YOLO – You Only Look Once, um dos modelos mainstream para produtização das principais tarefas que vimos na teoria: Detecção, Segmentação, Classificação, OBB e Pose Estimation. Vamos executar cada uma das tarefas usando o Colab e o YOLO para familiarização com as execuções e comandos. Também será realizado um processo completo de criação de dataset, desde a rotulagem das imagens, criação do dataset, inclusão no modelo YOLO, treinamento para ajuste fino e análise de resultados.

**Pacotes Python**

Nesta prática usaremos os seguintes pacotes:

Ultralytics, IPython, math

**Instalação do ultralytics**

!pip install ultralytics -q

**Importando os pacotes**

import ultralytics

from Ipython.display import display, Image

**Execuções possíveis para YOLO**

**Modo CLI**

!yolo {demais argumentos}

**Exemplo:**

!yolo task=detect mode=predict model=yolo11n.pt conf=0.25 source='image.jpg' save=True

#**task:** detect, segment, classify, pose, obb

#**mode:** train, val, predict, export, track

#**model:** nome do modelo, com extensão .pt

Outros parâmetros são necessários em cada conjuntos *task* e *mode*.

**Modo SDK Python**

model = YOLO("model.pt") #trocar model pelo arquivo do modelo!

results = model.predict(source='image.jpg', conf=0.25, save=True)

Os mesmos parâmetros são usados, porém com uma sintaxe diferente.

**Modelos para referência**

|  |  |
| --- | --- |
| **YOLOv8** | **YOLOv11** |
|  |  |

**Execuções iniciais**

Utilizando o modo SDK Python, vamos executar todas as tasks disponíveis e ver algumas saídas.

Utilize os comandos abaixo para executar a predição:

model = YOLO("model.pt") #trocar model pelo arquivo do modelo (ver tabela acima)

results = model.predict(source='image.jpg/vídeo.mp4', conf=0.25, save=True)

Utilize o comando abaixo para apresentar o resultado:

Image('image.jpg', height=300)

Video('video.mp4', height=300)

**Imagens**

Para detecção, segmentação, classificação, utilizar **meteor.jpg**

Para estimação de pose, utilizar **running.jpg**

Para oriented bounding box, utilizar **aerial.jpg**

Fique livre para alterar as imagens. Podem utilizar imagens que já tenham ou buscar na internet para entender diferentes resultados.

**Acessando os resultados**

**Detecção**

results

results[0].boxes.xywh # center-x, center-y, width, height

results[0].boxes.xyxy # top-left-x, top-left-y, bottom-right-x, bottom-right-y

results[0].boxes.conf

results[0].boxes.cls

**Segmentação**

results

results[0].masks.xy

results[0].masks.data

**OBS:** apresente o results[0].masks.data utilizando o plt.imshow().

**Classificação**

results

results[0].probs.top1

results[0].probs.top1conf

**Pose Estimation**

results

results[0].keypoints.xy

results[0].keypoints.data

**OBB – *Oriented Bounding Box***

results

results[0].obb.xywhr # center-x, center-y, width, height, angle (radians)

results[0].obb.conf

results[0].obb.cls

results[0].names

**Tracking**

Para a tarefa de *tracking* é necessário que uma sequência de imagens ou *frames* sejam apresentados ao modelo. Por isso, vamos utilizar um vídeo. Este vídeo pode estar no Youtube ou qualquer outra plataforma, ou também no diretório local.

model = YOLO("model.pt") # trocar model pelo arquivo do modelo (ver tabela pág.2)

results = model.track("vídeo.mp4", show=True, conf=0.5, save=True)

Para esta tarefa, usar o vídeo **meteor.mp4**

Os resultados também são acessados via *results*, como box, classes e confiança. Para visualizar o resultado, faça o download do vídeo do Colab e execute na sua máquina.

**Atividade para Treinamento de um modelo**

Para esta atividade, vamos fazer o uso do ultralytics no Colab e também de uma aplicação chamada Roboflow para criação do *dataset* rotulado.

**Conjunto de dados não rotulado**

Utilizar os vídeos **vid\_01.mp4** a **vid\_17.mp4**

**Adicionando rótulos às imagens**

Como vamos utilizar os vídeos, cada frame será convertido em uma imagem. Estas imagens deverão ser rotuladas utilizando a aplicação Roboflow, através do link: <https://app.roboflow.com/>

Seguir os passos:

1. New project
2. Project Name, Project Type
   1. Object Detection
3. Drag and Drop files
   1. Os arquivos de vídeo
4. Annotate
   1. Realizar o processo de rotulagem nas imagens
5. Add to Dataset
6. Create Dataset version
7. Usar Dataset version criado no colab para treinamento

**Realizando o treinamento**

A atividade de treinamento que será utilizada aqui é de ajuste fino. Este conceito é também atribuído ao termo *Transfer Learning*. Vamos precisar de um modelo de base e então fazer o treinamento a partir deste modelo.

Usar o modo SDK Python para esta atividade. Importante: o modo CLI pode ser usado também.

model = YOLO(“model.pt) # carrega um modelo pré-treinado

results = model.train(data="config.yaml", epochs=10, imgsz=640)

Abra o arquivo **data.yaml** e analise o conteúdo!

Aguardar o processo de treinamento ser finalizado.

**Verificando o resultado**

Image(filename='confusion\_matrix.png', width=600)

Image(filename='results.png', width=600)

**Validando o modelo**

IMPORTANTE: Usar apenas um dos comandos abaixo:

model = YOLO("model.pt") # para carregar um modelo oficial (pág. 2)

model = YOLO("best.pt") # para carregar um modelo treinado customizado

metrics = model.val()

metrics.box.map #map50-95

metrics.box.map50 #map50

metrics.box.map75 #map75

metrics.box.maps #map50-95 de cada categoria

**Utilizando o modelo treinamento e customizado para avaliar saídas**

Usar a imagem **trucks.png**

model = YOLO("model.pt")

results = model("imagem.jpg")

O *results* aqui é o mesmo já utilizado nos testes iniciais deste roteiro. Faça as análises novamente, considerando que estamos trabalhando com uma tarefa *Detect*.

Usar um dos vídeos **vid\_x.mp4** (de preferência um que não foi usado para treinamento) e realizar o processo de *tracking* no vídeo.