Introdução

A detecção e classificação são cruciais na visão computacional, o seu principal propósito é a identificação de objetos em uma imagem. Na classificação a imagem é passada a uma única classe. Já na detecção localiza a posição e o tamanho de todos os objetos de interesse em uma imagem.

Quais são as principais bibliotecas e frameworks para pré-processamento de imagens?
Dentre elas destacam-se a **OpenCV**, **TensorFlow** e **PyTorch.**

OpenCV: A biblioteca de visão computacional mais utilizada, oferecendo um amplo conjunto de ferramentas para diversas tarefas, como leitura, escrita, transformação e análise de imagens.

TensorFlow e PyTorch: Frameworks de deep learning que incluem módulos para pré-processamento de imagens, como normalização e aumento de dados.

Aplicação

A detecção e classificação amplamente utilizados em sistemas de reconhecimento facial, sistemas de vigilância, na área de saúde e na indústria. Neste estudo o modelo **EfficientDet-D0** é baixado do *TensorFlow Hub*. Esse modelo já foi treinado em um grande conjunto de dados e pode identificar diversos objetos. A imagem é carregada, convertida para o formato RGB e transformada em um tensor. A função **model(input_tensor)** executa a detecção, retornando um dicionário com informações sobre os objetos detectados. As informações relevantes, como caixas delimitadoras, classes, são extraídas do dicionário retornado pela função de detecção. A função **plot_detections** cria um gráfico, plota a imagem original e desenha as caixas delimitadoras ao redor dos objetos detectados.

```python
import tensorflow as tf
import cv2
import numpy as np
import tensorflow_hub as hub # Import tensorflow_hub

# Carregar o modelo pré-treinado EfficientDet-D0 usando tensorflow_hub
model = hub.load("https://tfhub.dev/tensorflow/efficientdet/d0/1")

# Carregar a imagem
image_path = "panda.png"  # Substitua por o caminho da sua imagem
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
input_tensor = tf.convert_to_tensor(image)
input_tensor = input_tensor[tf.newaxis, ...]

# Realizar a detecção
detections = model(input_tensor)

# Extrair as caixas delimitadoras e as classes
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections] for key, value in detections.items()}
boxes = detections['detection_boxes']
classes = detections['detection_classes']
scores = detections['detection_scores']
```

```python
# Visualizar os resultados
import matplotlib.pyplot as plt
%matplotlib inline

def plot_detections(image, boxes, classes, scores):
  fig,ax = plt.subplots(1)
  ax.imshow(image)

  for i in range(num_detections):
    if scores[i] < 0.5:
      continue
    # Cria uma caixa delimitadora..
    ymin, xmin, ymax, xmax = boxes[i]
    width, height = image.shape[1], image.shape[0]
    (left, right, top, bottom) = (xmin * width, xmax * width, ymin * height, ymax * height)
    # Cria um patch retangular.
    rect = plt.Rectangle((left, top), right - left, bottom - top, fill=False, edgecolor='red', linewidth=2)
    # Adiciona o patch aos eixos
    ax.add_patch(rect)

  plt.show()

plot_detections(image, boxes, classes, scores)
```