Modeling for 3D printing defect (spaghetti) detection

Michal Raczkowski

15-01-2024

Contents

1 Introduction

This document outlines training and configuration of YOLOv8 [1] model for detecting "spaghetti" problem in 3D printing

2 Explanation

YOLO, which stands for *"You Only Look Once,"* is an object detection algorithm with the following key features:

1. **Single Pass Detection:** YOLO analyzes the entire image in a single pass, making it significantly faster than methods that scan the image multiple times.
2. **Image Division:** The image is divided into a grid. Each grid cell is responsible for detecting objects within its bounds.
3. **Bounding Boxes and Class Predictions:** Each cell predicts bounding boxes and class probabilities for these boxes. Each bounding box comes with a confidence score reflecting the likelihood of an object being present and its class.
4. **Filtering and Combining Boxes:** YOLO filters out bounding boxes with low confidence scores. It also combines overlapping boxes for the same object into a single box.
5. **Output:** The final output is the image with bounding boxes drawn around detected objects, each labeled with the class of the object.

This approach, where the entire image is processed at once using a single neural network, allows YOLO to perform object detection tasks quickly, suitable for real-time applications.

3 Dataset

To train YOLO model for object detection of "spaghetti" issue we need dataset containing images and their labels which describe where on specific image is "spaghetti" defect, whole process and sourcing is described in document called "Data provisioning.pdf"
Dataset structure should look like this:

```
data/
├── images/
│   ├── train/
│   │   └── ...
│   └── val/
│       └── ...
└── labels/
    │   └── ...
    ├── train/
    │   └── ...
    └── val/
```

4 Configuration

For effective training, a configuration file is required for the model. This file provides information about the dataset's location, training and validation images, and specifies which classes correspond to the labels. All this information is stored in the `config.yaml` file.
The `config.yaml` file should have the following structure::

```
path: Repos/OpenWeekProject/data
train: images/train
val: images/train

names:
  0: spaghetti
```

Explanation:

- `path`::describe path of dataset main directory
- `train`::describe path for train images, it is relative to `path`
- `val`::describe path for validation images, it is relative to `path`
- `names`::describe classes used in model

  `0`: describe number of class

  `"spaghetti"` is name of the class, it could be any custom name

5 Code

Python code for training is very straight forward

```
from ultralytics import YOLO

model = YOLO("yolov8n.yaml")

results = model.train(data="config.yaml", epochs=20)
```

Explanation:

- `from ultralytics import YOLO`: imports the `YOLO` class from the `ultralytics` library, which provides functionalities for YOLO object detection models.
- `model = YOLO("yolov8n.yaml")`: Initializes a YOLO model using the configuration specified in `yolov8n.yaml`, setting up the model's architecture and parameters.
- `results = model.train(data="config.yaml", epochs=20)`: This line trains the YOLO model on a dataset specified in `config.yaml` and performs the training for 20 epochs. An `epoch` refers to one complete pass through the entire training dataset. In this case, the model will go through the dataset 20 times, each time updating its parameters to improve its accuracy and performance in object detection. The results of the training process are stored in the variable `results`.

6 Results

After executing the code mentioned in the previous section, a new directory named `runs` should appear in the directory where our script is located. The structure of this directory should be as follows:

```
runs/
└── detect/
    ├── train/
    │   ├── weights/
    │   └── ...
    └── ...
```

In the `train` directory, all results are stored, including images of confusion matrices, ROC curves, precision curves, F1 curves, etc. This directory also contains images showing prediction results on the validation set.

References

[1] You Only Look Once: YOLOv8:
https://github.com/ultralytics/ultralytics