

Yolo & Flutter

Costruiamo un rilevatore di magia

Chi sono



Simone Bonfrate

XR Engineer @ Wideverse

AI & Data Science Student @ Politecnico di Bari

Community Lead @ GDG Bari

Runner, D&D Player, Manga Reader...

**Ci arriva una
lettera**

**Ci arriva una
lettera**

(Non da Hogwarts)

Definiamo il Task

Il pubblico ministero babbano ci ha chiesto di progettare un applicazione per rilevare la magia

**Cosa abbiamo a
disposizione?**

Definiamo il Task

Per rilevare la bacchetta possiamo sfruttare la fotocamera.

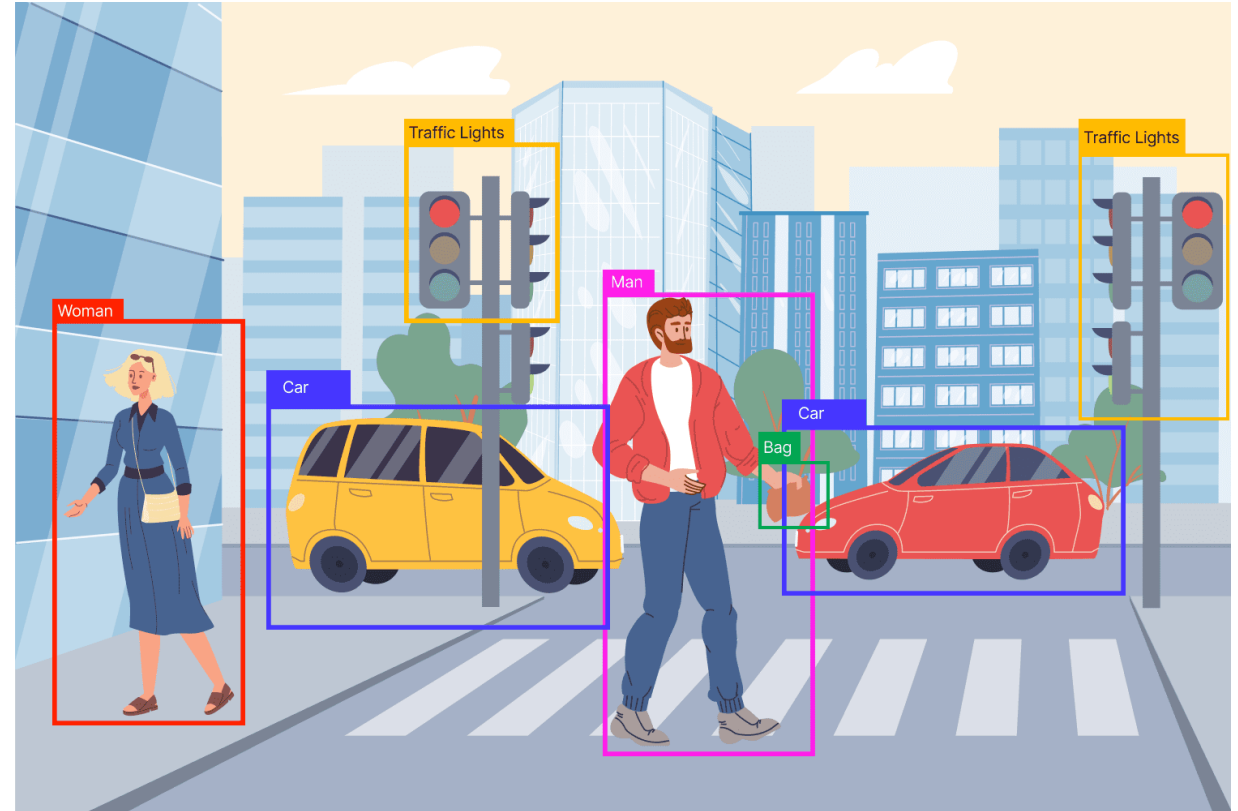
La foto può essere usata per un task di object detection.



Object Detection

Object Detection: che cos'è?

E' una tecnica di computer vision che, dato in input un'immagine, permette di distinguere la presenza o meno di oggetti di una determinata classe.



YOLO (You Only Look Once)

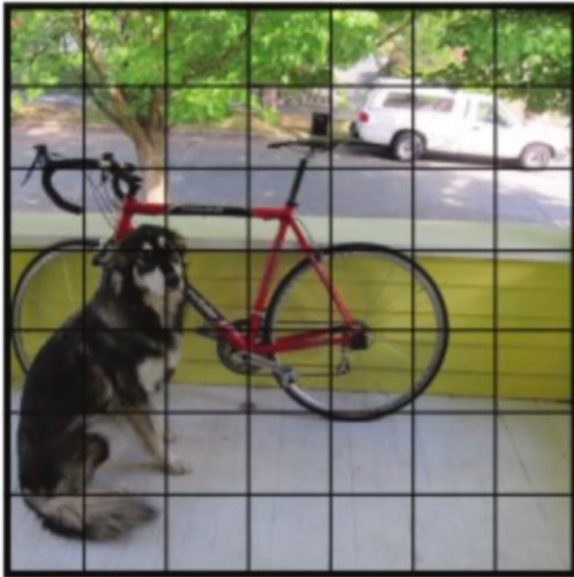
YOLO è un modello di Object detection capace di predire dall'intera immagine una sola volta la posizione e la classe degli oggetti.

Questo approccio consente a YOLO di ottenere risultati in tempo reale. Utile per applicazioni che richiedono una rapida elaborazione.

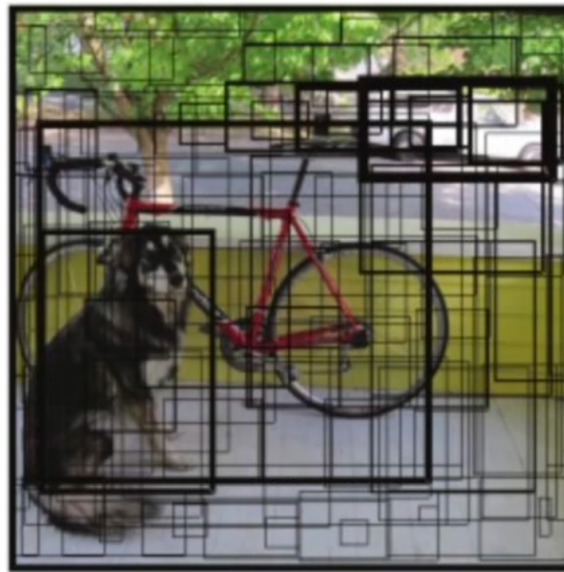


YOLO (You Only Look Once)

YOLO divide l'immagine in una griglia di celle e per ciascuna ricava sia le bounding box che la classe .



S x S grid of cells



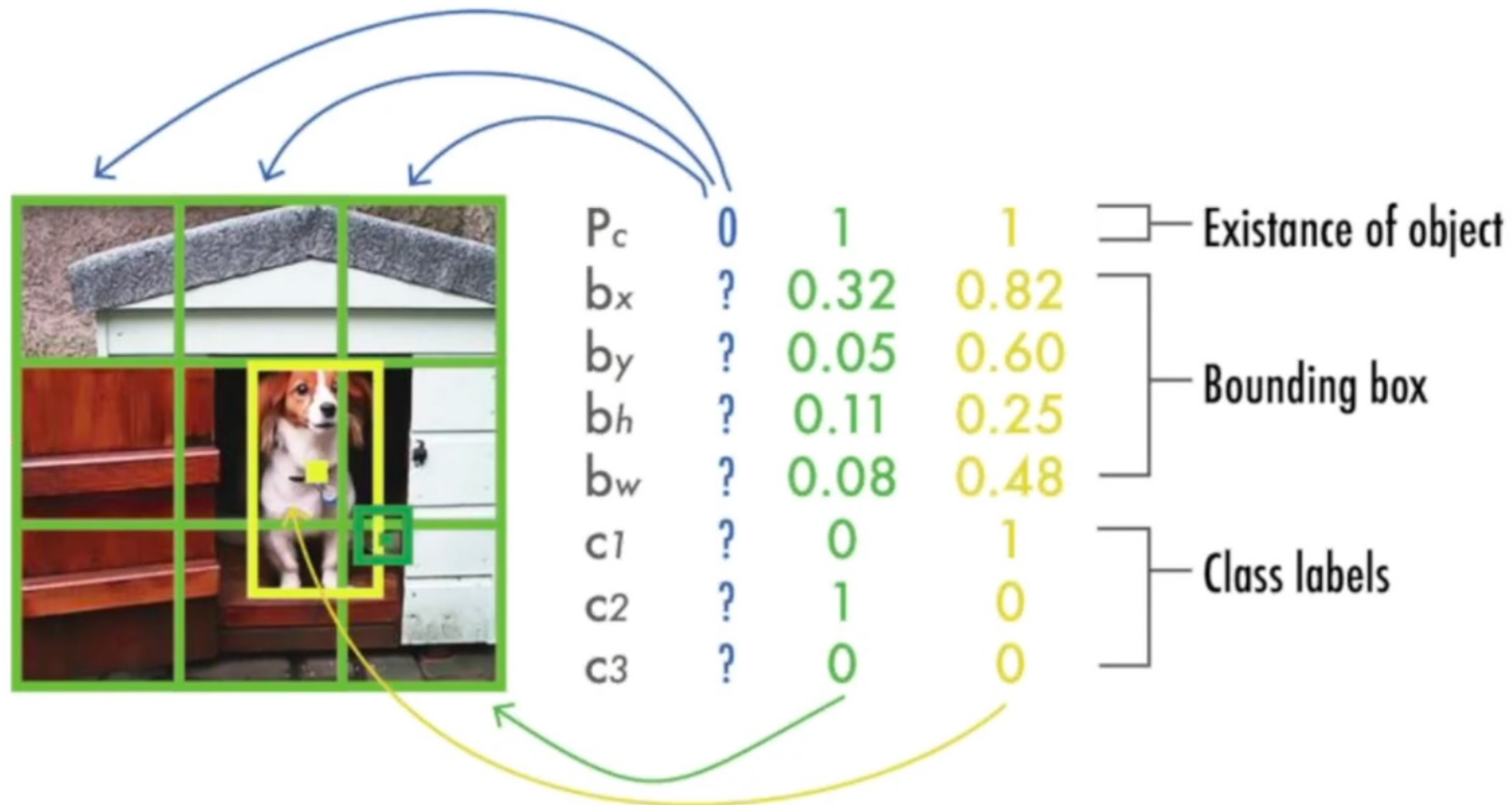
Bounding boxes and confidence



Class probability map

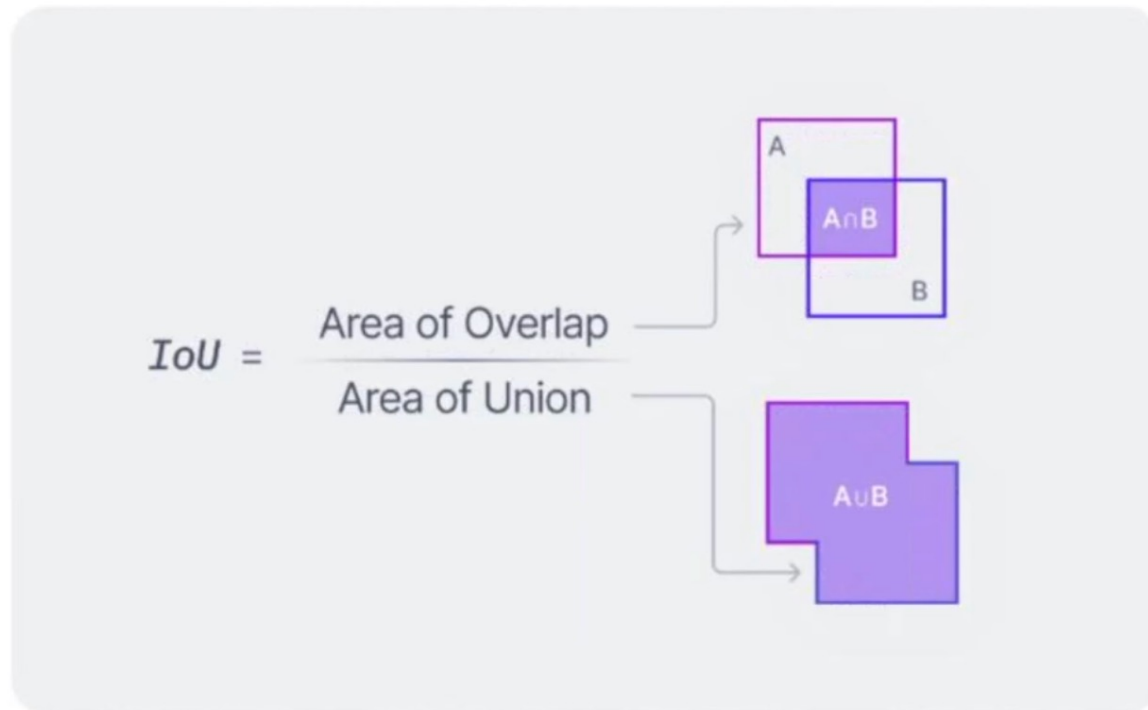
YOLO (You Only Look Once)

L'unione delle classi predette e delle bounding box genera un output vettoriale nel seguente formato



YOLO (You Only Look Once)

Per stimare le performance si usa l'Intersection over Union



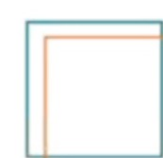
$$\text{Intersection over Union } (IoU) = \frac{|A \cap B|}{|A \cup B|}$$

IoU= 0.35



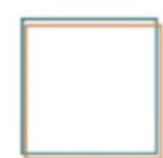
Poor

IoU= 0.74



Good

IoU= 0.93

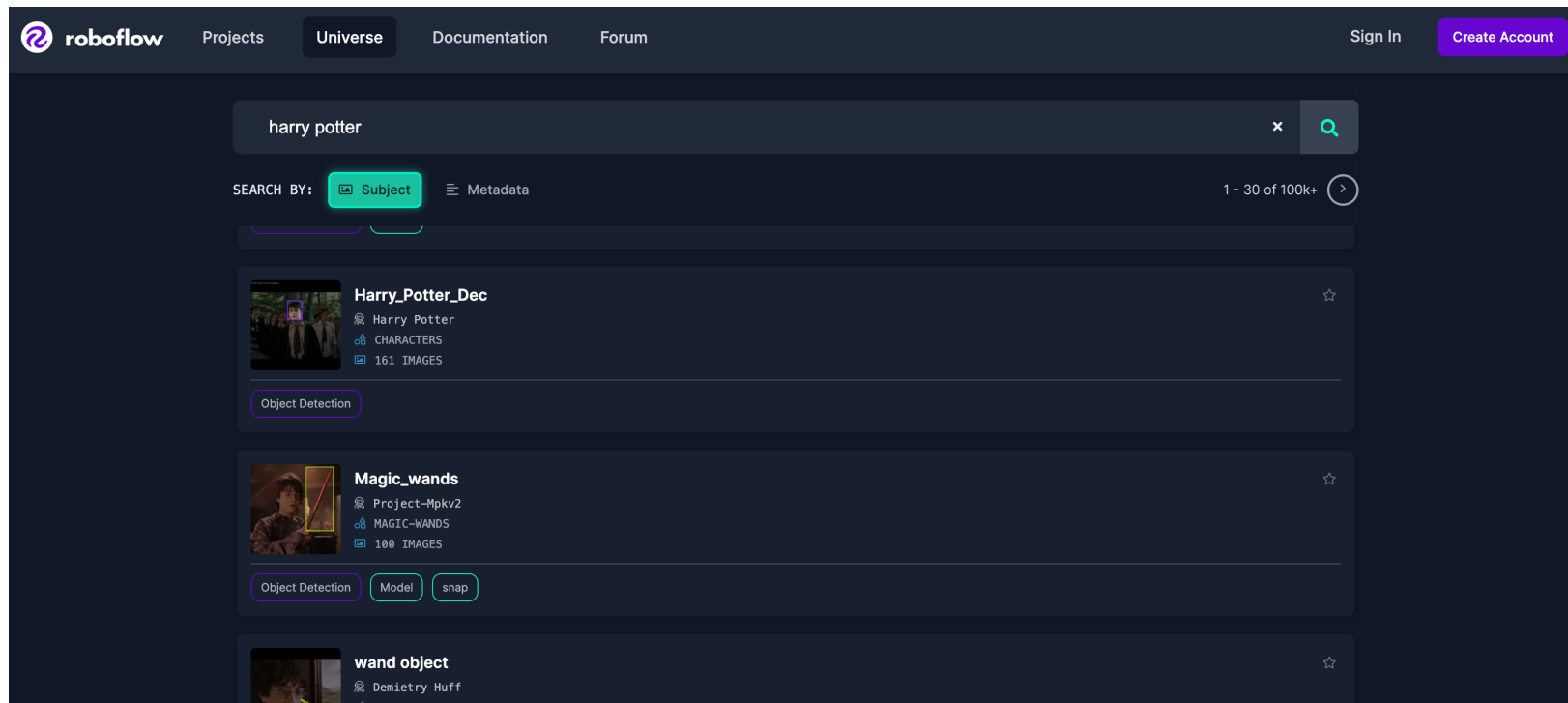


Excellent

Scegliamo il dataset

Il dataset è la parte più importante per la realizzazione di un'applicazione di computer vision perché permette di migliorare il modello affinché possa generalizzare e rilevare le classi di oggetti definite a prescindere dell'ambiente in cui è posizionato o l'oggetto osservato.

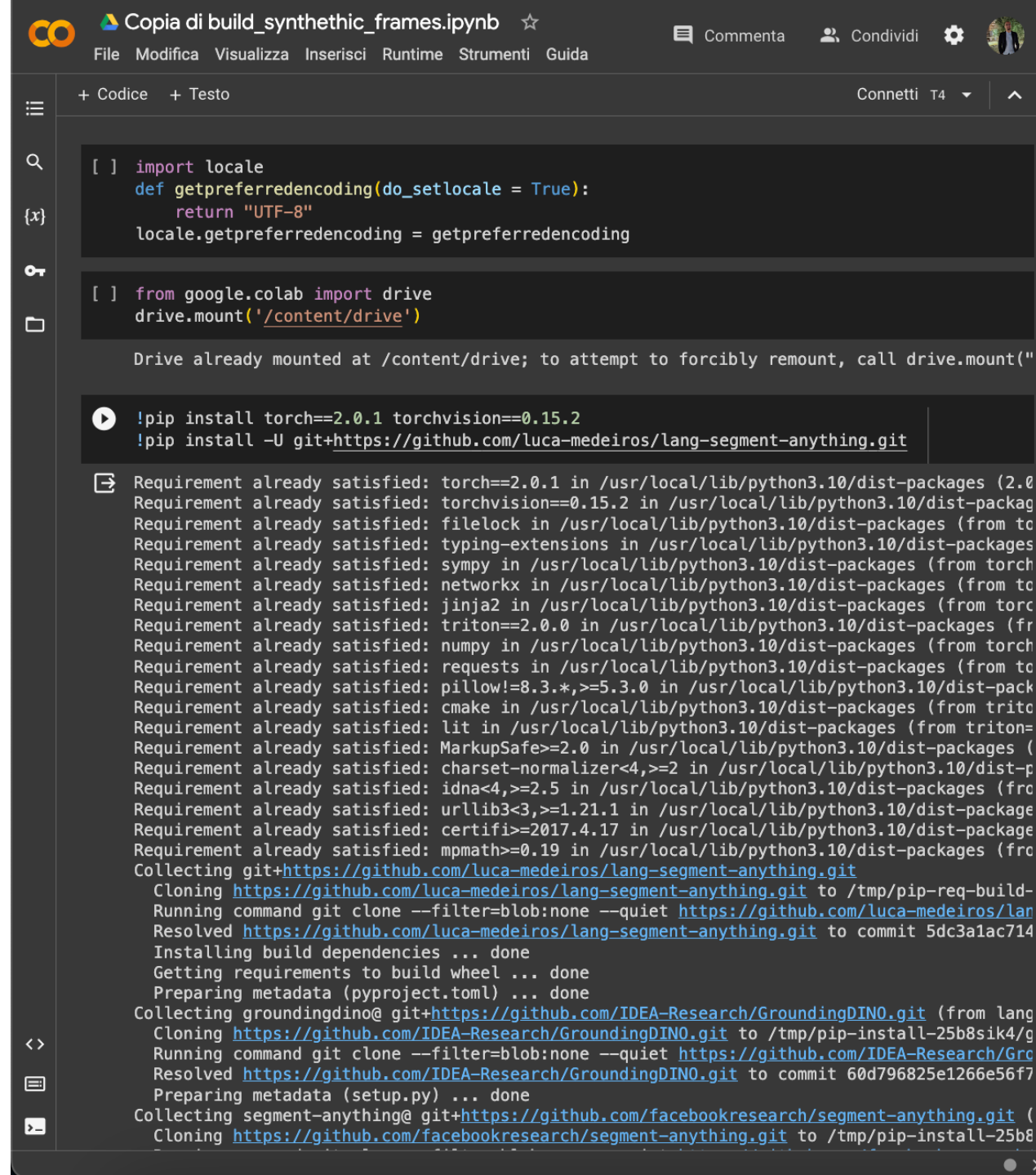
Per scaricare il dataset possiamo sfruttare Roboflow e cercare il progetto più adatto a noi



Prepariamo l'environment (Python)

Per effettuare il training di YOLO è necessaria molta potenza grafica.

Tramite Google Colab è possibile accedere ad ambienti Python pronti per lo sviluppo dell'IA gratuitamente



```
Copia di build_synthetic_frames.ipynb
File Modifica Visualizza Inserisci Runtime Strumenti Guida

+ Codice + Testo Connetti T4

[ ] import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding

[ ] from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("

!pip install torch==2.0.1 torchvision==0.15.2
!pip install -U git+https://github.com/luca-medeiros/lang-segment-anything.git

Requirement already satisfied: torch==2.0.1 in /usr/local/lib/python3.10/dist-packages (2.0
Requirement already satisfied: torchvision==0.15.2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from to
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from to
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from torch
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torch
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from to
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from trito
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton=
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (fro
Collecting git+https://github.com/luca-medeiros/lang-segment-anything.git
  Cloning https://github.com/luca-medeiros/lang-segment-anything.git to /tmp/pip-req-build-
  Running command git clone --filter=blob:none --quiet https://github.com/luca-medeiros/lan
  Resolved https://github.com/luca-medeiros/lang-segment-anything.git to commit 5dc3a1ac714
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting groundingdino@ git+https://github.com/IDEA-Research/GroundingDINO.git (from lang
  Cloning https://github.com/IDEA-Research/GroundingDINO.git to /tmp/pip-install-25b8sik4/g
  Running command git clone --filter=blob:none --quiet https://github.com/IDEA-Research/Grc
  Resolved https://github.com/IDEA-Research/GroundingDINO.git to commit 60d796825e1266e56f7
  Preparing metadata (setup.py) ... done
Collecting segment-anything@ git+https://github.com/facebookresearch/segment-anything.git (
  Cloning https://github.com/facebookresearch/segment-anything.git to /tmp/pip-install-25b8
```

Installiamo le dipendenze

```
!pip install torch==2.0.1 torchvision==0.15.2
```

```
!git clone https://github.com/ultralytics/yolov5 # clone repo
```

```
%cd yolov5
```


Scarichiamo il dataset

```
!pip install -q roboflow
```

```
%cd /content/yolov5
```

```
from roboflow import Roboflow  
rf = Roboflow(api_key="<API_KEY>")  
project = rf.workspace("<WORKSPACE>").project("<PROJECT>")  
dataset = project.version(1).download("yolov5")
```

Configuriamo il dataset per YOLO

```
# define number of classes based on YAML
import yaml

with open(dataset.location + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])

#this is the model configuration we will use for our tutorial
%cat /content/yolov5/models/yolov5s.yaml

from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```

Configuriamo il dataset per YOLO

```
%writetemplate /content/yolov5/models/custom_yolov5s.yaml
```

```
# parameters
```

```
nc: {num_classes} # number of classes
```

```
depth_multiple: 0.33 # model depth multiple
```

```
width_multiple: 0.50 # layer channel multiple
```

```
# anchors
```

```
anchors:
```

```
...
```

```
# YOLOv5 backbone
```

```
backbone:
```

```
...
```

```
# YOLOv5 head
```

```
head:
```

```
...
```

Alleniamo il modello

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416
                  --batch 16
                  --epochs 20
                  --data {dataset.location}/data.yaml
                  --cfg ./models/custom_yolov5s.yaml
                  --weights 'yolov5s.pt'
                  --name yolov5s_results
                  --cache
```

Testiamo il modello

```
# when we ran this, we saw .007 second inference time. That is 140
FPS on a TESLA P100!
# use the best weights!
%cd /content/yolov5/
!python detect.py
    --weights runs/train/yolov5s_results/weights/best.pt
    --img 416
    --conf 0.4
    --source {dataset.location}/test/images
```

Esportiamo il modello per Flutter

```
%cd /content/yolov5/  
!python export.py  
    --weights runs/train/yolov5s_results/weights/best.pt  
    --include torchscript  
    --img 416  
    --optimize  
    --conf-thres 0.7
```

Realizziamo il componente per Flutter

```
class ObjectDetectionView{
  File? _imageFile;
  late ModelObjectDetection _objectModel;
  String? _imagePrediction;
  List? _prediction;
  File? _image;
  ImagePicker _picker = ImagePicker();
  bool objectDetection = false;
  List<ResultObjectDetection?> objDetect = [];

  Future loadModel() async {
    String pathObjectDetectionModel = "assets/models/yolov5s.torchscript";

    try {
      _objectModel = await FlutterPytorch.loadObjectDetectionModel(
        pathObjectDetectionModel, 1, 416, 416,
        labelPath: "assets/labels/labels.txt");
    } catch (e) {
      if (e is PlatformException) {
        print("only supported for android, Error is $e");
      } else {
        print("Error is $e");
      }
    }
  }
}
```

Realizziamo il componente per Flutter

```
@override
void setState(){
  super.setState();
  loadModel();
}

Widget build(BuildContext context){
  return Column(
    children: <Widget>[
      Container(
        height: 150,
        width: 300,
        child: objDetect.isNotEmpty
          ? _image == null
            ? Text('No image selected.')
            : _objectModel!.renderBoxesOnImage(_image!, objDetect)
          : _image == null
            ? Text('No image selected.')
            : Image.file(_image!),
      ),
      Center(
        Button(
          onTap: () => runObjectDetection(),
          child: Text('Select Image')
        )
      )
    ],
  );
}
```


Realizziamo il componente per Flutter

```
return Column(  
  children: <Widget>[  
    Container(  
      height: 150,  
      width: 300,  
      child: objDetect.isNotEmpty  
        ? _image == null  
          ? Text('No image selected.')  
          : _objectModel!.renderBoxesOnImage(_image!, objDetect)  
        : _image == null  
          ? Text('No image selected.')  
          : Image.file(_image!),  
    ),  
  
    Center(  
      Button(  
        onTap: () => runObjectDetection(),  
        child: Text('Select Image')  
      )  
    ),  
  ],  
);  
}
```

Realizziamo il componente per Flutter

```
Future runObjectDetection() async {
  //pick an image

  final XFile? image = await _picker.pickImage(
    source: ImageSource.gallery, maxWidth: 200, maxHeight: 200);

  objDetect = await _objectModel.getImagePrediction(
    await File(image!.path).readAsBytes(),
    minimumScore: 0.1,
    IOUThershold: 0.3);

  objDetect.forEach((element) {
    print({
      "score": element?.score,
      "className": element?.className,
      "class": element?.classIndex,
      "rect": {
        "left": element?.rect.left,
        "top": element?.rect.top,
        "width": element?.rect.width,
        "height": element?.rect.height,
        "right": element?.rect.right,
        "bottom": element?.rect.bottom,
      },
    },
  ),
}
```

Realizziamo il componente per Flutter

```
IOUThershold: 0.3);

objDetect.forEach((element) {
  print({
    "score": element?.score,
    "className": element?.className,
    "class": element?.classIndex,
    "rect": {
      "left": element?.rect.left,
      "top": element?.rect.top,
      "width": element?.rect.width,
      "height": element?.rect.height,
      "right": element?.rect.right,
      "bottom": element?.rect.bottom,
    },
  });
});

setState(() {
  _image = File(image!.path);
});
}
}
```

Demo

Come posso migliorare il dataset?

Per migliorare le prestazioni del modello (accuratezza), è necessario migliorare il dataset, inserendo più immagini con caratteristiche diverse.

Esistono diversi approcci:

- Augmentation
- Creazione di un dataset sintetico
- Scraping delle immagini e labelling automatico

Augmentation

L'augmentation crea più versioni della stessa immagine con alcune modifiche

Augmentations create new training examples for your model to learn from.

IMAGE LEVEL AUGMENTATIONS

- Flip
- 90° Rotate
- Crop
- Rotation
- Shear
- Grayscale
- Hue
- Saturation
- Brightness
- Exposure
- Blur
- Noise
- Cutout
- Mosaic

BOUNDING BOX LEVEL AUGMENTATIONS ?

- Flip
- 90° Rotate
- Crop
- Rotation
- Shear
- Brightness
- Exposure
- Blur
- Noise

Cancel

Dataset sintetico

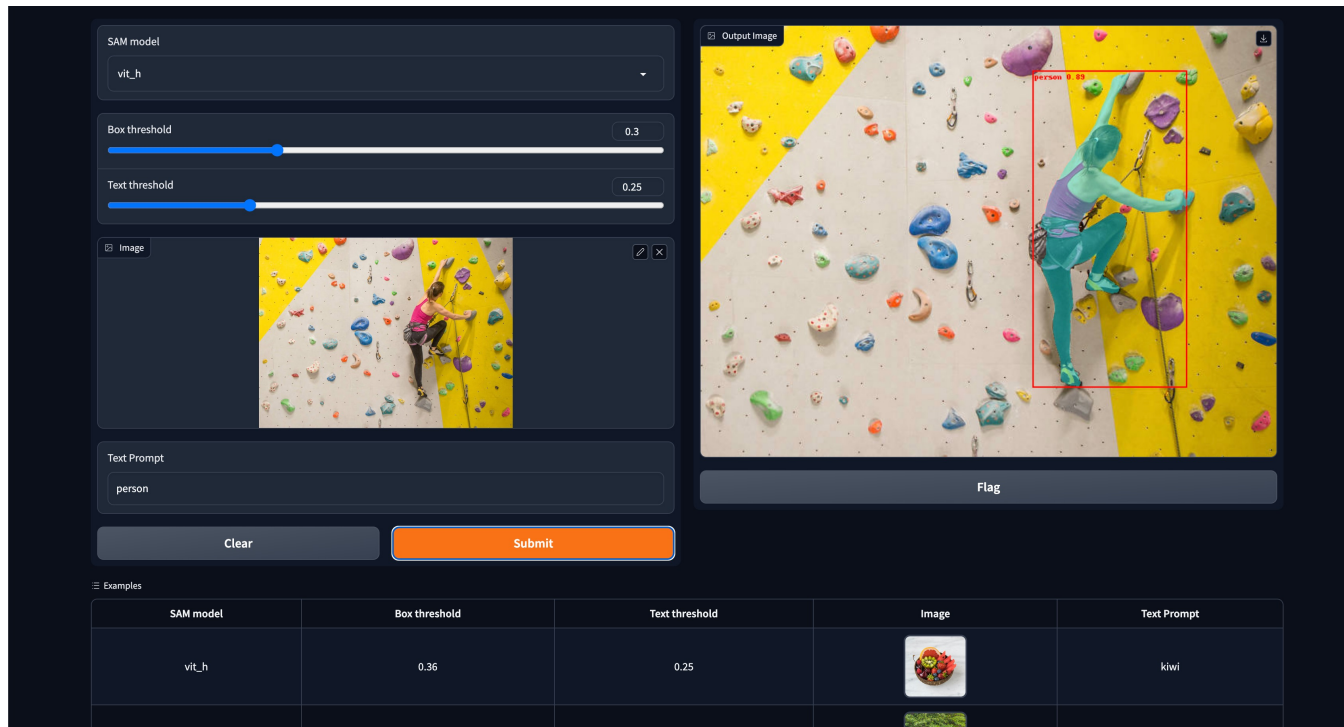
I dataset sintetici sono composti da dati creati ad hoc in un contesto virtuale controllato il più fedele possibile ai dati reali.





Autolabeling

Una tecnica che permette di etichettare gli oggetti in un immagine utilizzando modelli di deep learning.

Un esempio è LangSAM, un progetto composto da SAM e GroundingDINO e genera le bounding box partendo da un prompt



The screenshot displays the LangSAM web interface. On the left, there are controls for the SAM model (set to 'vit_h'), Box threshold (0.3), and Text threshold (0.25). Below these is an 'Image' input field showing a person climbing a wall. A 'Text Prompt' field contains the word 'person'. At the bottom of the controls are 'Clear' and 'Submit' buttons. On the right, the 'Output Image' shows the same climbing scene with a red bounding box around the climber, labeled 'person 0.93'. Below the output image is a 'Flag' button. At the bottom of the interface, there is an 'Examples' section with a table.

SAM model	Box threshold	Text threshold	Image	Text Prompt
vit_h	0.36	0.25		kiwi
				

Domande?

FlutterHEROES



23 February 2024

See you in person or online!

brought to you by [/synesthesia](#) **events**

Grazie per l'attenzione

 bonfry

 Simone Bonfrate

Riferimenti

Roboflow Youtube channel

Ultralytics LCC



Lascia un feedback

