Coin Counter Project

INTRODUCTION

This project is a coin counter project made for real-time coin detection. The architecture used was Yolov8n for the detection and MobilnetV2 for the classification and the models were trained based on two datasets of coins.

INSTALLATION

After installing the zip folder and unzipping it, there are 3 files that are needed to be run in the following order.

1. data.yaml
   This file includes the location of the coinDatasetForDetector folder, and the path should be replaced with your path of that folder

2. DetectorTrainer.py

   This is the python file that trains the detector based on one of the datasets to detect the coins. The variables used can be altered, but these are good values for the model to run decent. After this runs, a .pt file should be created

3. classifierTrainer.py

   This python file trains the classifier that decides what coins to put in each category (ex. the 2 euro coin is classified as a 2.00 etc.). Again the variables used can be tweaked, but these worked the best. After this runs a .pth file should be created

4. CoinDetector.py

This file is the main file of all, that uses the previous two files and runs the real-time camera footage, showing the coins that it detects and the total. The paths that need to be altered for the code to run are the following:

I) **yolo_path** (this should be the path of the .pt file that the classifierTrainer.py will create)

II) **video_path** (this should be the path of the .mp4 or .mov etc. file that can be used in case of running the coin detector on a video and not real-time. If the code is run real-time this can be ignored)

III) **classifier.load_state_dict(torch.load("PATH", map_location=device))** (PATH should be replaced with the name of the .pth file that the classifierTrainer.py will create)

(THE DATASETS USED WERE FOUND ON THE FOLLOWING WEBSITES AND ARE NOT OURS)

DATASET 1: https://universe.roboflow.com/a1-rbm94/coin-detector-wzimv/dataset/3

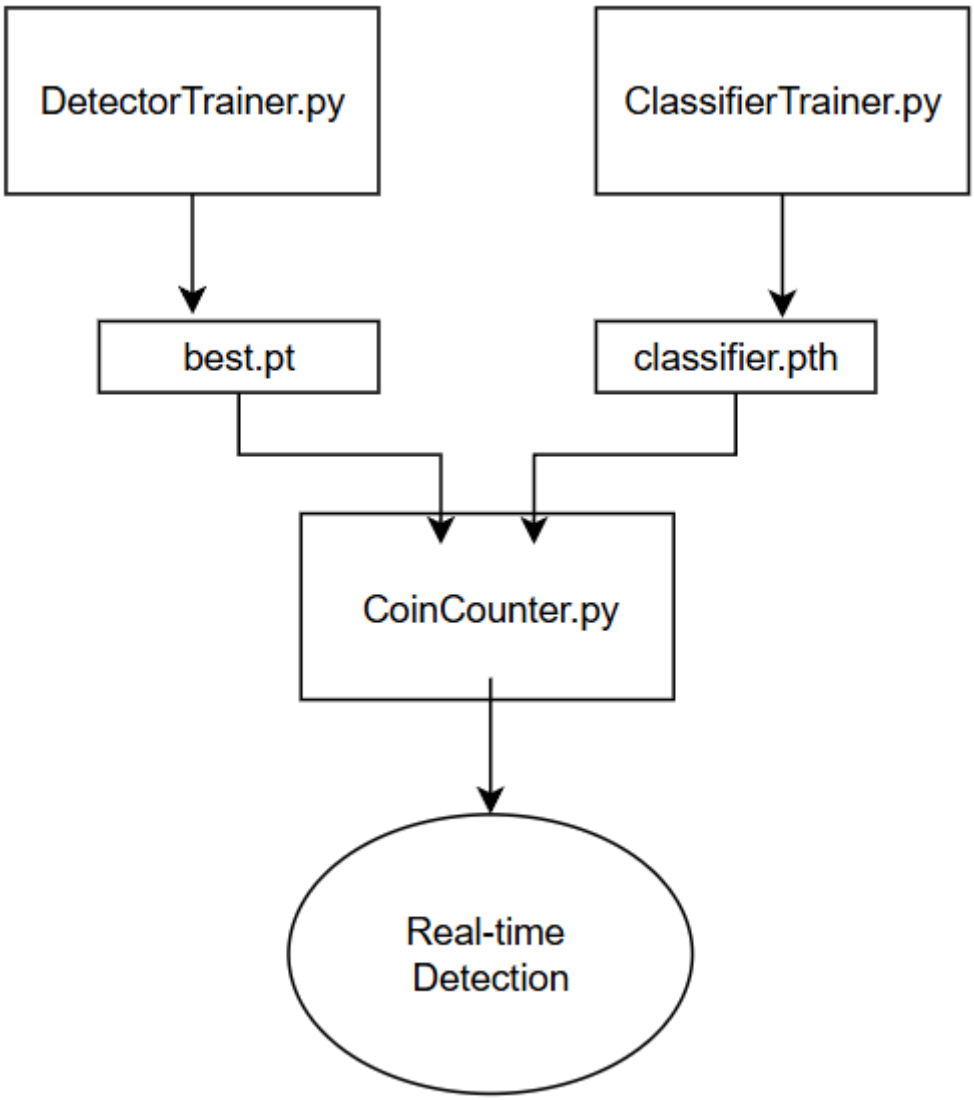
This dataset was used for the detection of coins in the frame

DATASET 2: https://github.com/kaa/coins-dataset

This dataset was used to identify every coin and classify it as 1 euro, 2 euro etc. It needs to be split in two folders (train and val) for the proper function of the classifier. We split it in a 80/20 percentile between train/val.

Pictures of the main sections

How the model works:



Detector.py

```
# --- Training with early stopping ---
model.train(
    data=r"C:\Users\kokmo\Desktop\CoinCounter\data.yaml", # path to your dataset yaml
    epochs=150,              # max epochs
    imgsz=640,               # image size
    batch=16,                # batch size
    device=device,           # GPU or CPU
    workers=4,               # data loading speed
    name='coin_detector',    # output folder
    exist_ok=True,           # overwrite if exists
    patience=20              #stop after 10 loops of not improving
)
```

Variables used in training

Classifier.py

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize(224),  # μεγαλώνουμε τη μικρότερη πλευρά
        transforms.RandomResizedCrop( size: 224, scale=(0.7, 1.0)),
        transforms.RandomRotation(360),
        transforms.RandomHorizontalFlip(),
        transforms.RandomVerticalFlip(),
        transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3, hue=0.1),
        transforms.ToTensor(),
        transforms.Normalize( mean: [0.485, 0.456, 0.406],
                              std: [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),  # μόνο center crop
        transforms.ToTensor(),
        transforms.Normalize( mean: [0.485, 0.456, 0.406],
                              std: [0.229, 0.224, 0.225])
    ]),
}
```

Data augmentation of the dataset used

CoinCounter.py

```
# DETECTION (YOLO)
results = detector(frame, stream=True, verbose=False, conf=0.3)
```

Yolo detection with threshold (conf)

```
# Results and filtering
for i in range(len(indices)):
    idx = indices[i].item()
    conf = confidences[i].item()
    x1, y1, x2, y2 = boxes_coords[i]

    if conf >= CONFIDENCE_THRESHOLD:
        val = coin_values[idx]
        label = coin_labels[idx]
        current_frame_total += val
        color = (0, 255, 0)
        text = f"{label} EUR ({conf*100:.1f}%)"
    else:
        color = (0, 0, 255)
        text = "Uncertain"
```

MobilenetV2 filtering if confidence>threshold