YOLO Finetuned Model

In this notebook we prepare the Sohas_weapon-Detection dataset (https://github.com/ari-dasci/OD-WeaponDetection) for training it for the YOLOv8 Object detection model. The annotations that come with the dataset do not fit the YOLO format, because they used an XML format and therefore, we needed to convert the annotations into the YOLO format. The YOLO object detection format specifies annotations in text files, with one file per image, containing the class ID and normalized bounding box coordinates (center x, center y, width, height) for each object in the image.

Since, we already used an algorithm to transform the XML annotations into the coco.json format, we reused that in this pre-processing notebook aswell and then transform the json into the YOLO format. After we have successfully transformed the annotation format we can use the dataset for training the object detection model.

Libraries

First we have to pip install ultralytics in order to import the YOLO object detection model. We also import a number of other libraries that we will need later in the project.

```
!pip install ultralytics
```

```
Requirement already satisfied: ultralytics in ./.venv/lib/python3.9/site-packa
Requirement already satisfied: matplotlib>=3.3.0 in ./.venv/lib/python3.9/site
Requirement already satisfied: opencv-python>=4.6.0 in ./.venv/lib/python3.9/s
Requirement already satisfied: pillow>=7.1.2 in ./.venv/lib/python3.9/site-pac
Requirement already satisfied: pyyaml>=5.3.1 in ./.venv/lib/python3.9/site-pac
Requirement already satisfied: requests>=2.23.0 in ./.venv/lib/python3.9/site-
Requirement already satisfied: scipy>=1.4.1 in ./.venv/lib/python3.9/site-pack
Requirement already satisfied: torch>=1.8.0 in ./.venv/lib/python3.9/site-pack
Requirement already satisfied: torchvision>=0.9.0 in ./.venv/lib/python3.9/sit
Requirement already satisfied: tqdm>=4.64.0 in ./.venv/lib/python3.9/site-pack
Requirement already satisfied: psutil in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: py-cpuinfo in ./.venv/lib/python3.9/site-packag
Requirement already satisfied: pandas>=1.1.4 in ./.venv/lib/python3.9/site-pac
Requirement already satisfied: seaborn>=0.11.0 in ./.venv/lib/python3.9/site-p
Requirement already satisfied: ultralytics-thop>=0.2.5 in ./.venv/lib/python3
Requirement already satisfied: contourpy>=1.0.1 in ./.venv/lib/python3.9/site-
Requirement already satisfied: cycler>=0.10 in ./.venv/lib/python3.9/site-pack
Requirement already satisfied: fonttools>=4.22.0 in ./.venv/lib/python3.9/site
Requirement already satisfied: kiwisolver>=1.3.1 in ./.venv/lib/python3.9/site
Requirement already satisfied: numpy>=1.23 in ./.venv/lib/python3.9/site-packa
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.9/site-p
Requirement already satisfied: pyparsing>=2.3.1 in ./.venv/lib/python3.9/site-
Requirement already satisfied: python-dateutil>=2.7 in ./.venv/lib/python3.9/s
Requirement already satisfied: importlib-resources>=3.2.0 in ./.venv/lib/pytho
```

```
Requirement already satisfied: pytz>=2020.1 in ./.venv/lib/python3.9/site-pacl
Requirement already satisfied: tzdata>=2022.7 in ./.venv/lib/python3.9/site-pa
Requirement already satisfied: charset-normalizer<4,>=2 in ./.venv/lib/python3
Requirement already satisfied: idna<4,>=2.5 in ./.venv/lib/python3.9/site-pacl
Requirement already satisfied: urllib3<3,>=1.21.1 in ./.venv/lib/python3.9/sit
Requirement already satisfied: certifi>=2017.4.17 in ./.venv/lib/python3.9/sit
Requirement already satisfied: filelock in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: typing-extensions>=4.8.0 in ./.venv/lib/python3
Requirement already satisfied: sympy in ./.venv/lib/python3.9/site-packages (f
Requirement already satisfied: networkx in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: jinja2 in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: fsspec in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: zipp>=3.1.0 in ./.venv/lib/python3.9/site-packa
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.9/site-packages
Requirement already satisfied: MarkupSafe>=2.0 in ./.venv/lib/python3.9/site-p
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in ./.venv/lib/python3.9/s

[notice] A new release of pip is available: 24.0 -> 24.1.2
[notice] To update, run: pip install --upgrade pip
```

```
from ultralytics import YOLO
import os
import torch
from PIL import Image
import cv2
import matplotlib.pyplot as plt
from tqdm import tqdm
print(f"PyTorch version: {torch.__version__}")
```

```
/Users/davidpichler/Documents/Uni/DigiEcon22/4 Semester/AI II/Projekt/noteboo
  warnings.warn(
PyTorch version: 2.3.1
```

```
print(f"Is MPS built? {torch.backends.mps.is_built()}")
print(f"Is MPS available? {torch.backends.mps.is_available()}")

# Set device
device = "mps" if torch.backends.mps.is_available() else "cpu"
print(f"Using device: {device}")
```

```
Is MPS built? True
Is MPS available? True
Using device: mps
```

Load Data

First we load some pictures and inspect the pictures

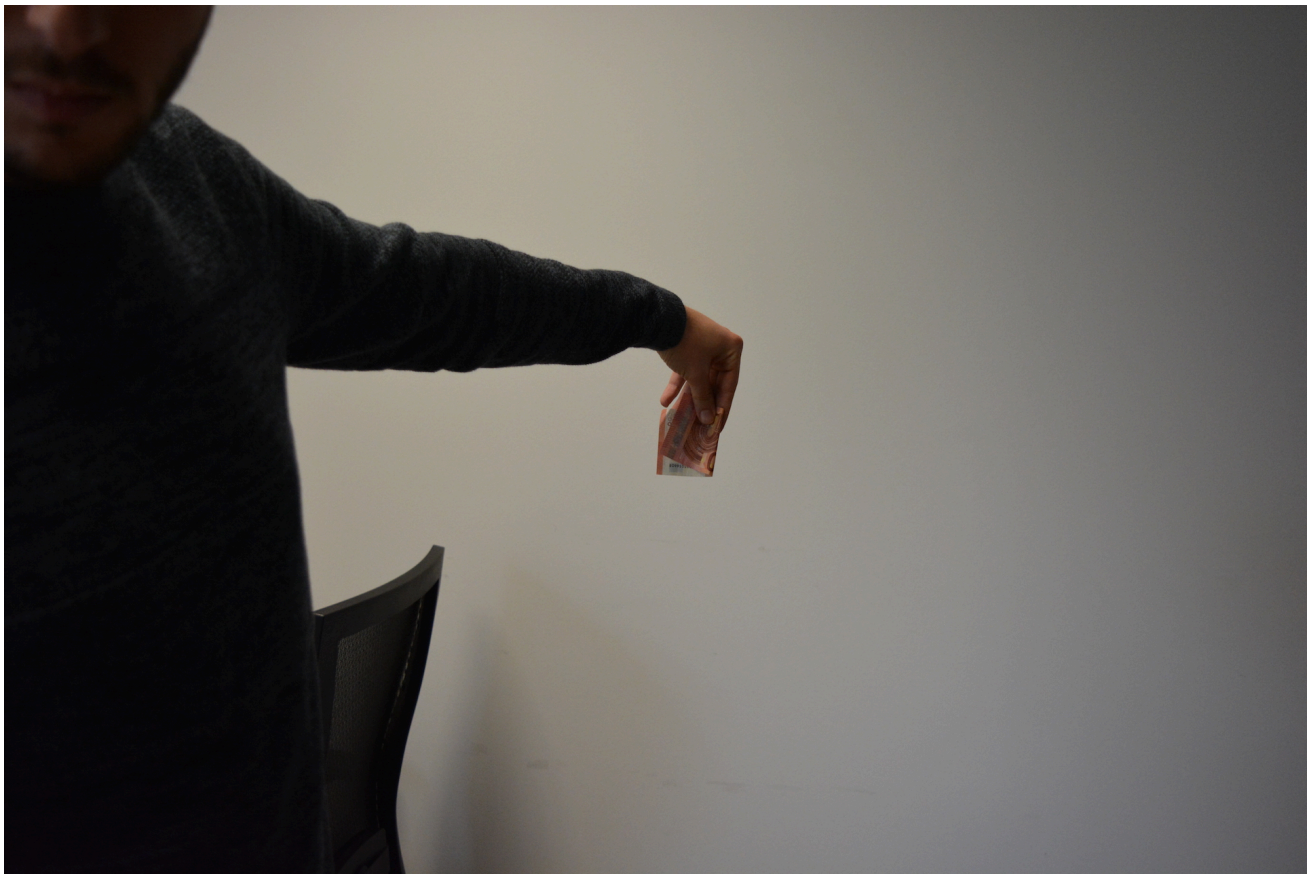
```
# Directory containing your images
directory = "data/Sohas_weapon-Detection"

# List all files in the directory
files = os.listdir(directory)
print(files)

# Load images
images = []
for file in files:
    if file.endswith(".jpg") or file.endswith(".png"):  # Add more extensions if
        image_path = os.path.join(directory, file)
        image = Image.open(image_path)
        images.append(image)
```

```
['billete_2341.jpg', 'pistol_0696.jpg', 'annotations_test', '.DS_Store', 'tes
```

```
images[0]
```



XML Coco convertion

In this section we convert the XML annotations into the COCO Json Format as we did for the DetrRes50 aswell.

Create Coco test annotation

```
# Directory containing your images
image_directory = "data/Sohas_weapon-Detection/test/images"
xml_directory = "data/Sohas_weapon-Detection/test/annotation"
```

```
import os
from xml.etree import ElementTree as ET
def extract_category_mapping(xml_directory):
  """Extracts unique object names (categories) from all XML files in a directory

  Args:
      xml_directory: Path to the directory containing the XML annotation files.

  Returns:
      A dictionary mapping unique object names (categories) to consecutive intege
  """
  category_mapping = {}
  category_id = 1
  for filename in tqdm(os.listdir(xml_directory)):
    if filename.endswith(".xml"):
      xml_path = os.path.join(xml_directory, filename)
      tree = ET.parse(xml_path)
      root = tree.getroot()

      for obj in root.findall('object'):
        category_name = obj.find('name').text
        if category_name not in category_mapping:
          category_mapping[category_name] = category_id
          category_id += 1

  return category_mapping


category_mapping = extract_category_mapping(xml_directory)

print(f"Category mapping: {category_mapping}")
```

```
100%|██████████| 866/866 [06:33<00:00,  2.20it/s]Category mapping: {'smartphon
```

```
import os
from xml.etree import ElementTree as ET
from collections import Counter


def convert_to_coco(xml_directory, category_mapping, output_filename="coco_annota
  """Converts XML annotation files in a directory to COCO format and saves it to

  Args:
      xml_directory: Path to the directory containing the XML annotation files.
      category_mapping: Dictionary mapping unique object names (categories) to th
      output_filename: Filename for the output COCO annotations JSON file (defaul
  """
  coco_annotations = {"images": [], "annotations": []}
  image_id = 1  # Counter for image IDs

  xml_files = sorted(os.listdir(xml_directory))

  for filename in tqdm(xml_files):
    if filename.endswith(".xml"):
      xml_path = os.path.join(xml_directory, filename)
      image_filename = os.path.splitext(filename)[0] + ".jpg"  # Extract filename

      # Print debug information
      print(f"Processing image: {filename}")
      #print(f"Expected image filename: {image_filename}")

      tree = ET.parse(xml_path)
      root = tree.getroot()

      annotations = []
      for obj in root.findall('object'):
        name = obj.find('name').text
        category_id = category_mapping.get(name)
        if category_id is None:
          print(f"Warning: Category '{name}' not found in mapping. Skipping objec
          continue

        xmin = int(obj.find('bndbox/xmin').text)
        ymin = int(obj.find('bndbox/ymin').text)
        xmax = int(obj.find('bndbox/xmax').text)
        ymax = int(obj.find('bndbox/ymax').text)
        width = xmax - xmin
        height = ymax - ymin
        area = width * height

        annotation = {
            "image_id": image_id,
            "file_name": image_filename,
            "id": len(coco_annotations["annotations"]) + 1,  # Unique ID for each
            "category_id": category_id,
            "name": name,
            "bbox": [xmin, ymin, width, height],
            "area": area
        }
```

```
        annotations.append(annotation)

      # Add image information
      coco_annotations["images"].append({
            "image_id": image_id,
            "file_name": image_filename
      })

      # Add annotations for this image
      coco_annotations["annotations"].append(annotations)
      #coco_annotations["annotations"] = annotations
      #print(coco_annotations)
      image_id += 1

  # Save COCO annotations to JSON file
  with open(output_filename, "w") as f:
    json.dump(coco_annotations, f)
  print("Successfully written annotation json file")
  return coco_annotations



# Example usage (assuming you have the category_mapping dictionary)
coco_annotations = convert_to_coco(xml_directory, category_mapping, "data/Sohas_w
```

```
  0%|          | 0/866 [00:00<?, ?it/s]Processing image: ABbframe00145.xml
Processing image: ABbframe00289.xml
Processing image: ABbframe00322.xml
Processing image: ABbframe00325.xml
Processing image: ABbframe00331.xml
Processing image: ABmframe00154.xml
Processing image: ABmframe00262.xml
Processing image: ABmframe00280.xml
Processing image: ABmframe00334.xml
Processing image: ABmframe00346.xml
Processing image: ABsframe00010.xml
Processing image: ABsframe00121.xml
Processing image: ABsframe00157.xml
Processing image: ABsframe00193.xml
Processing image: ABsframe00196.xml
Processing image: ABsframe00208.xml
Processing image: DSC_00021.xml
Processing image: DSC_0010.xml
Processing image: DSC_00131.xml
Processing image: DSC_00321.xml
Processing image: DSC_00361.xml
Processing image: DSC_00591.xml
Processing image: DSC_0069.xml
Processing image: DefenseAndSurvive14.xml
Processing image: DefenseKnifeAttack0103.xml
Processing image: DefenseKnifeAttack0155.xml
Processing image: DefenseKnifeAttack0163.xml
Processing image: DefenseKnifeAttack0168.xml
Processing image: DefenseKnifeAttack0169.xml
Processing image: DefenseKnifeAttack0172.xml
Processing image: DefenseKnifeAttack0183.xml
Processing image: DefenseKnifeAttack0184.xml
```

```
Processing image: DefenseKnifeAttack0218.xml
Processing image: DefenseKnifeAttack0315.xml
Processing image: DefenseKnifeAttack0320.xml
Processing image: DefenseKnifeAttack0376.xml
Processing image: DefenseKnifeAttack0399.xml
Processing image: DefenseKnifeAttack0443.xml
Processing image: DefenseKnifeAttack0451.xml
Processing image: DefenseKnifeAttack0476.xml
Processing image: DefenseKnifeAttack0690.xml
Processing image: DefenseKnifeAttack0734.xml
Processing image: DefenseKnifeAttack0784.xml
Processing image: DefenseKnifeAttack0798.xml
Processing image: DefenseKnifeAttack1072.xml
Processing image: DefenseKnifeAttack1115.xml
Processing image: DefenseKnifeAttack1223.xml
Processing image: DefenseKnifeAttack1224.xml
Processing image: HBbframe00145.xml
Processing image: HBbframe00181.xml
Processing image: HBbframe00316.xml
Processing image: HBbframe00361.xml
Processing image: HBmframe00151.xml
Processing image: HBmframe00163.xml
Processing image: HBmframe00172.xml
Processing image: HBmframe00256.xml
```

```
with open("data/Sohas_weapon-Detection/test/coco_annotations.json", "rb") as f:
    coco_annotations = json.load(f)
```

```
xml_files = sorted(os.listdir(xml_directory))
len(xml_files)
```

```
866
```

```
len(coco_annotations["images"]), len(coco_annotations["annotations"])
```

```
(866, 866)
```

```
coco_annotations["images"][500], coco_annotations["annotations"][500][0]
```

```
({'image_id': 501, 'file_name': 'knife_408.jpg'},
 {'image_id': 501,
  'file_name': 'knife_408.jpg',
  'id': 501,
  'category_id': 2,
  'name': 'knife',
  'bbox': [69, 377, 326, 412],
  'area': 134312})
```

as you can see here the output in coco format for an object looks like this

({'image_id': 501, 'file_name': 'knife_408.jpg'}, {'image_id': 501, 'file_name': 'knife_408.jpg', 'id': 501, 'category_id': 2, 'name': 'knife', 'bbox': [69, 377, 326, 412], 'area': 134312})

Creat Coco train annotation

```
image_directory = "data/Sohas_weapon-Detection/train/images"
xml_directory = "data/Sohas_weapon-Detection/train/annotation"
cocojson_save_path = "data/Sohas_weapon-Detection/train/coco_annotations.json"


category_mapping = extract_category_mapping(xml_directory)

print(f"Category mapping: {category_mapping}")
```

```
100%|██████████| 5076/5076 [44:14<00:00,  1.91it/s]   Category mapping: {'mon
```

```
coco_annotations = convert_to_coco(xml_directory, category_mapping, cocojson_save
```

```
  0%|          | 0/5076 [00:00<?, ?it/s]Processing image: ABbframe00154.xm
  0%|          | 1/5076 [00:00<37:38,  2.25it/s]Processing image: ABbframe
Processing image: ABbframe00166.xml
Processing image: ABbframe00169.xml
  0%|          | 4/5076 [00:00<16:41,  5.07it/s]Processing image: ABbframe
  0%|          | 5/5076 [00:01<21:39,  3.90it/s]Processing image: ABbframe
  0%|          | 6/5076 [00:01<25:41,  3.29it/s]Processing image: ABbframe
Processing image: ABbframe00271.xml
Processing image: ABbframe00274.xml
Processing image: ABbframe00277.xml
Processing image: ABbframe00280.xml
Processing image: ABbframe00283.xml
Processing image: ABbframe00286.xml
Processing image: ABbframe00292.xml
Processing image: ABbframe00295.xml
Processing image: ABbframe00298.xml
Processing image: ABbframe00301.xml
Processing image: ABbframe00304.xml
Processing image: ABbframe00313.xml
Processing image: ABbframe00316.xml
Processing image: ABbframe00319.xml
Processing image: ABbframe00328.xml
Processing image: ABbframe00346.xml
Processing image: ABbframe00349.xml
Processing image: ABbframe00370.xml
Processing image: ABbframe00421.xml
  1%|          | 26/5076 [00:02<04:22, 19.21it/s]Processing image: ABbfram
  1%|          | 28/5076 [00:02<06:05, 13.80it/s]Processing image: ABbfram
Processing image: ABmframe00127.xml
Processing image: ABmframe00133.xml
Processing image: ABmframe00142.xml
Processing image: ABmframe00157.xml
Processing image: ABmframe00163.xml
Processing image: ABmframe00172.xml
Processing image: ABmframe00211.xml
Processing image: ABmframe00226.xml
Processing image: ABmframe00271.xml
  1%|          | 37/5076 [00:02<05:06, 16.44it/s]Processing image: ABmfram
Processing image: ABmframe00289.xml
```

```
  1%|          | 39/5076 [00:03<09:18,  9.01it/s]Processing image: ABmfram
Processing image: ABmframe00313.xml
  1%|          | 41/5076 [00:04<13:15,  6.33it/s]Processing image: ABmfram
Processing image: ABmframe00412.xml
Processing image: ABsframe00001.xml
Processing image: ABsframe00004.xml
Processing image: ABsframe00019.xml
  1%|          | 46/5076 [00:05<11:42,  7.16it/s]Processing image: ABsfram
Processing image: ABsframe00067.xml
Processing image: ABsframe00094.xml
Processing image: ABsframe00100.xml
Processing image: ABsframe00133.xml
Processing image: ABsframe00151.xml
  1%|          | 52/5076 [00:06<10:55,  7.66it/s]Processing image: ABsfram
Processing image: ABsframe00190.xml
Processing image: ABsframe00214.xml
Processing image: ABsframe00223.xml
  1%|          | 56/5076 [00:06<10:26,  8.01it/s]Processing image: ABsfram
```

```
with open("data/Sohas_weapon-Detection/train/coco_annotations.json", "rb") as f:
    coco_annotations = json.load(f)
```

```
xml_files = sorted(os.listdir(xml_directory))
len(xml_files)
```

```
866
```

```
len(coco_annotations["images"]), len(coco_annotations["annotations"])
```

```
(866, 866)
```

```
coco_annotations["images"][500], coco_annotations["annotations"][500][0]
```

```
({'image_id': 501, 'file_name': 'knife_408.jpg'},
 {'image_id': 501,
  'file_name': 'knife_408.jpg',
  'id': 501,
  'category_id': 2,
  'name': 'knife',
  'bbox': [69, 377, 326, 412],
  'area': 134312})
```

Coco YOLO convertion

```
import json
import cv2
import os
import matplotlib.pyplot as plt
import shutil
```

object within the picture (center x, center y, width, height)

```
import json
import os

def find_image_path(image_base_path):
    """
    Helper function to find the correct image path considering different extensions
    """
    possible_extensions = ['.jpg', '.JPG', '.jpeg', '.JPEG', '.png', '.PNG']
    for ext in possible_extensions:
        image_path = image_base_path + ext
        if os.path.exists(image_path):
            return image_path
    return None

def convert_coco_to_yolo(coco_json_path, output_dir):
    # Load COCO JSON
    with open(coco_json_path, 'r') as f:
        coco_data = json.load(f)

    # Create a dictionary to map image ids to filenames
    image_id_to_filename = {image['image_id']: image['file_name'] for image in coco

    # Create the output directory if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    # Process each annotation
    for annotation in coco_data['annotations']:
        image_id = annotation[0]['image_id']
        bbox = annotation[0]['bbox']
        category_id = annotation[0]['category_id']

        # COCO format: [top left x, top left y, width, height]
        # YOLO format: [x_center, y_center, width, height] normalized
        x_tl, y_tl, width, height = bbox
        x_center = x_tl + width / 2
        y_center = y_tl + height / 2

        # Normalize coordinates by the dimensions of the image
        image_filename = image_id_to_filename[image_id]
        image_base_path = os.path.join(os.path.dirname(coco_json_path), "images/",
        image_path = find_image_path(image_base_path)

        if image_path is None:
            print(f"Warning: Image {image_base_path} not found with expected extens
            continue
```