Prototype development

- **Objective:** ***use AI approaches to detect and assess the state of EDP electronic boxes***
- *What's in scope?*
  - Detection of EDP boxes from photos
  - Classification of their states
- *What's not in scope?*
  - Detection of specific instance of alterations (crack, oxidation, lock broken)

Detection of the boxes

Photos

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
•
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Data

```
•
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

 # 22650 aug photos
 # train: 16987
 # valid: 5663
```

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Data augmentation

```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Augmentation

Data

```
.
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

# 22650 aug photos
# train: 16987
# valid: 5663
```

Object detection

YOLOv4: Optimal Speed and Accuracy of Object Detection.
Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao
arXiv, april 2020

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
0 0.410495 0.507119 0.125286 0.160234
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

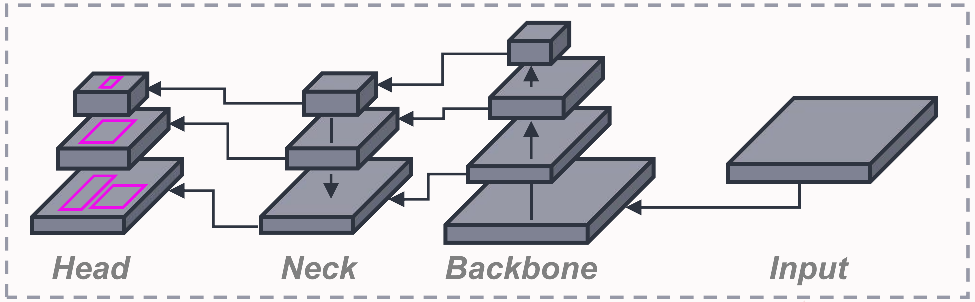
```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Data

```
.
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

# 22650 aug photos
# train: 16987
# valid: 5663
```

Object detection

Architecture



YOLOv4: Optimal Speed and Accuracy of Object Detection.
Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao
arXiv, april 2020

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
0 0.410495 0.507119 0.125286 0.160234
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

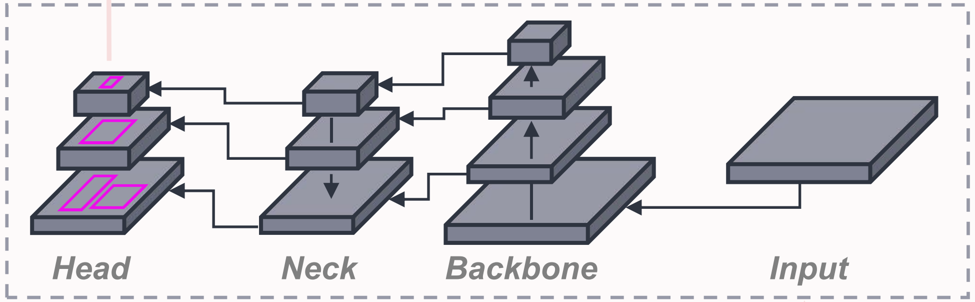
```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Data

```
.
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

# 22650 aug photos
# train: 16987
# valid: 5663
```

Predictions & metrics

```
mAP ─┐
     ├─▶ AP@50
IoU ─┘
```



Object detection

Architecture

YOLOv4: Optimal Speed and Accuracy of Object Detection.
Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao
arXiv, april 2020

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
•
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

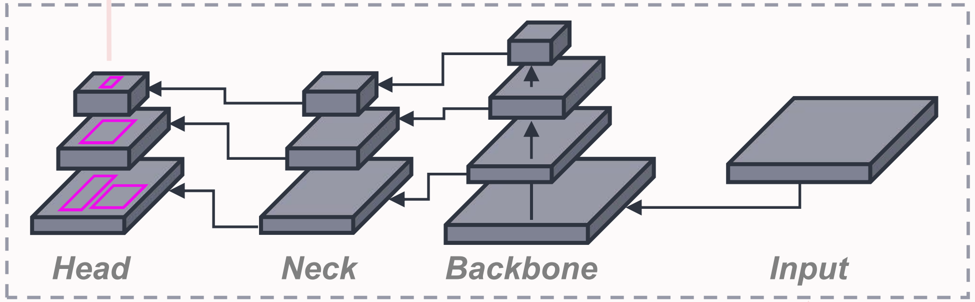
```
augmentation_pipeline = A.Compose(
     A.Resize(416, 416),
     A.Equalize(by_channels=True),
     A.RGBShift(
         r_shift_limit=(-30, 30),
         g_shift_limit=(-30, 30),
         b_shift_limit=(-30, 30),
         p=0.25
     ),
     A.HorizontalFlip(p=0.35),
     A.VerticalFlip(p=0.35),
     A.ShiftScaleRotate(p=0.35),
     A.RandomSnow(
         brightness_coeff=2.0,
         p=0.2
     )],
     A.BboxParams(
         'yolo',
         ['class_labels']
     )
)
```

Data

```
•
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

# 22650 aug photos
# train: 16987
# valid: 5663
```

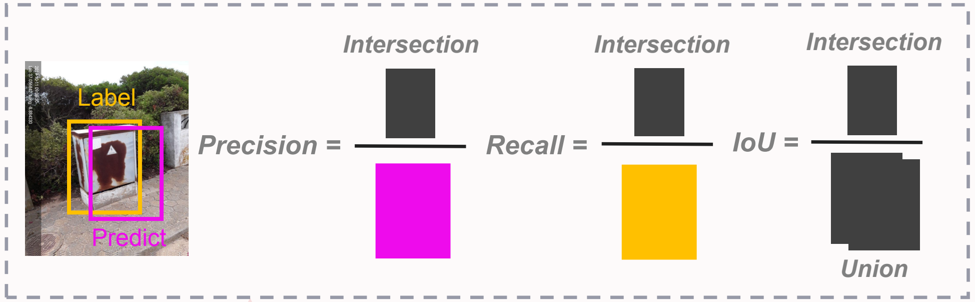
Object detection

Architecture



Predictions & metrics

```
mAP ─┐
     ├─▶ AP@50
IoU ─┘
```



YOLOv4: Optimal Speed and Accuracy of Object Detection.
Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao
arXiv, april 2020

Detection of the boxes

Photos

Labeling

CVAT YOLO output

```
.
├── obj.data
├── obj.names
├── obj_train_data
└── train.txt
```

```
0 0.450688 0.540651 0.305322 0.421997
```

```
0 0.410495 0.507119 0.125286 0.160234
```

```
0 0.529792 0.439717 0.321354 0.336973
```

Augmentation

Data augmentation

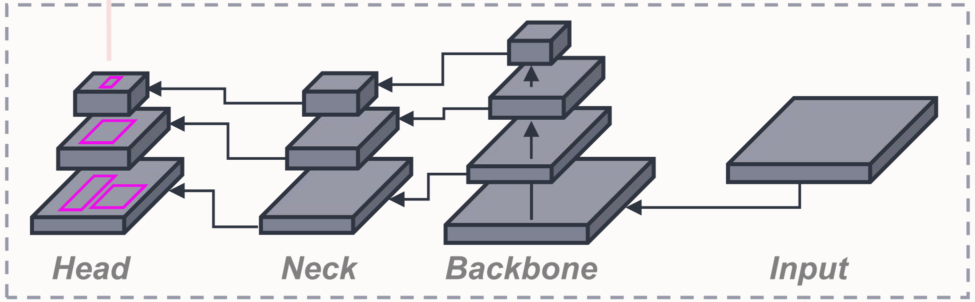
```
augmentation_pipeline = A.Compose(
    A.Resize(416, 416),
    A.Equalize(by_channels=True),
    A.RGBShift(
        r_shift_limit=(-30, 30),
        g_shift_limit=(-30, 30),
        b_shift_limit=(-30, 30),
        p=0.25
    ),
    A.HorizontalFlip(p=0.35),
    A.VerticalFlip(p=0.35),
    A.ShiftScaleRotate(p=0.35),
    A.RandomSnow(
        brightness_coeff=2.0,
        p=0.2
    )],
    A.BboxParams(
        'yolo',
        ['class_labels']
    )
)
```

Data

```
.
├── obj.data
├── obj.names
├── obj
├── test
├── train.txt
└── test.txt

# 22650 aug photos
# train: 16987
# valid: 5663
```

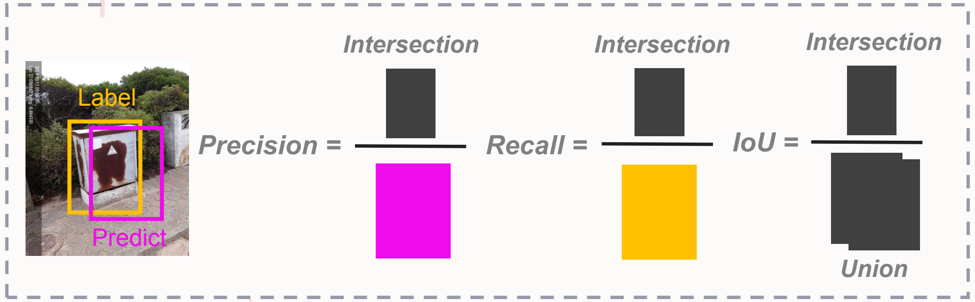
Object detection

Architecture



Predictions & metrics

```
mAP ─┐
     ├──▶ AP@50
IoU ─┘
```

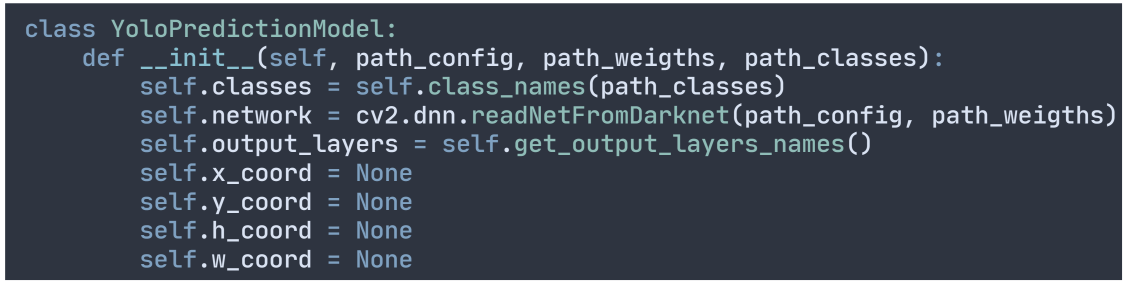


Results

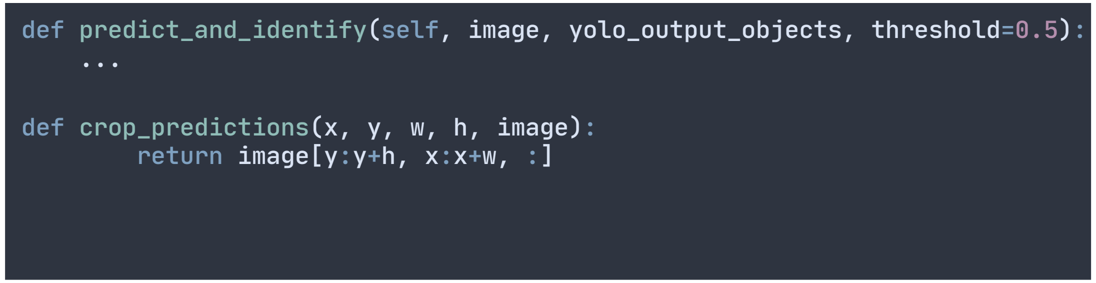
```
./darknet detector map [meta] [cfg] [weigth]

class_id = 0, name = box-closed
TP = 5669, FP = 81, FN = 52
precision = 0.99, recall = 0.99, F1-score = 0.99
ap = 99.73%
average IoU   = 87.53 %
mean average precision (mAP@0.50) = 99.73 %
```

YOLOv4: Optimal Speed and Accuracy of Object Detection.
Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao
arXiv, april 2020

Testing the model and extracting the EDP Boxes

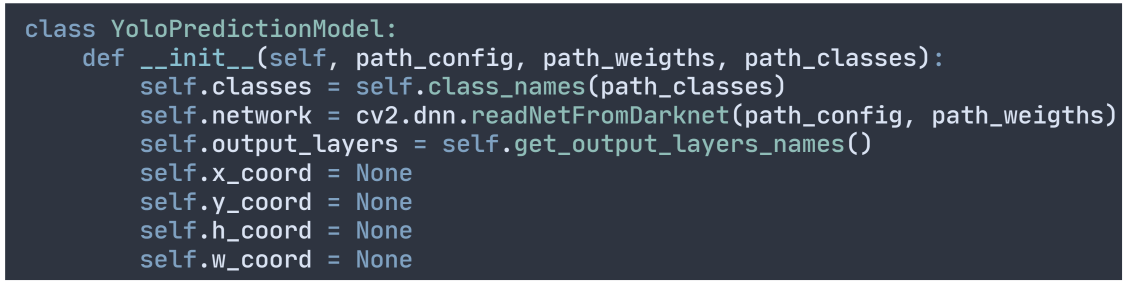


```
class YoloPredictionModel:
    def __init__(self, path_config, path_weigths, path_classes):
        self.classes = self.class_names(path_classes)
        self.network = cv2.dnn.readNetFromDarknet(path_config, path_weigths)
        self.output_layers = self.get_output_layers_names()
        self.x_coord = None
        self.y_coord = None
        self.h_coord = None
        self.w_coord = None
```

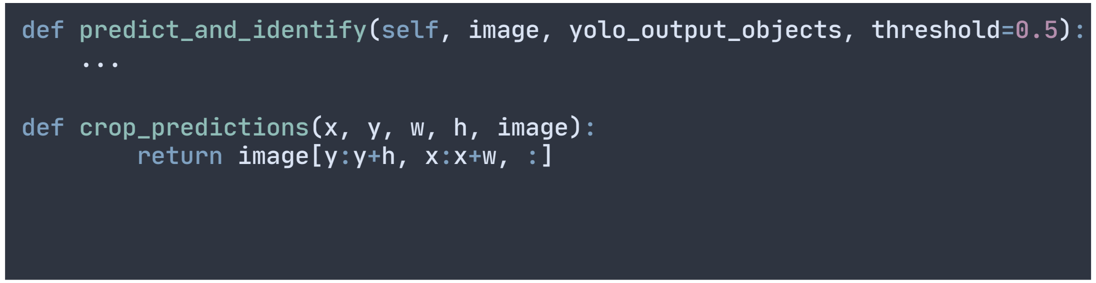


```
def predict_and_identify(self, image, yolo_output_objects, threshold=0.5):
    ...

def crop_predictions(x, y, w, h, image):
        return image[y:y+h, x:x+w, :]
```

Testing the model and extracting the EDP Boxes

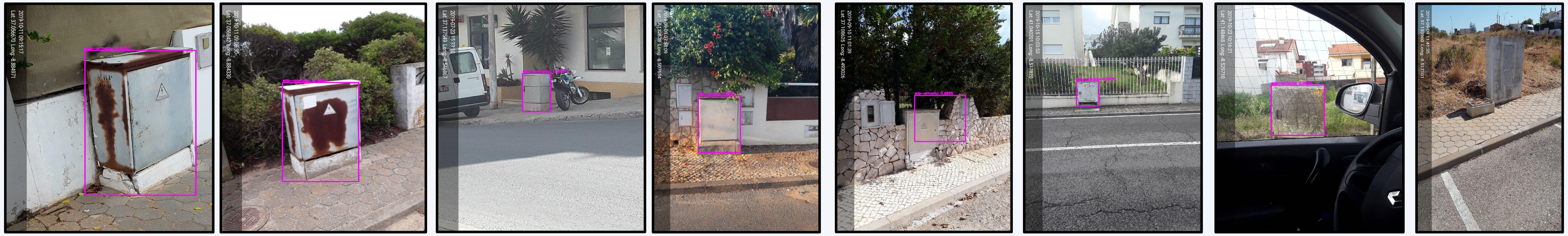


```
class YoloPredictionModel:
    def __init__(self, path_config, path_weigths, path_classes):
        self.classes = self.class_names(path_classes)
        self.network = cv2.dnn.readNetFromDarknet(path_config, path_weigths)
        self.output_layers = self.get_output_layers_names()
        self.x_coord = None
        self.y_coord = None
        self.h_coord = None
        self.w_coord = None
```

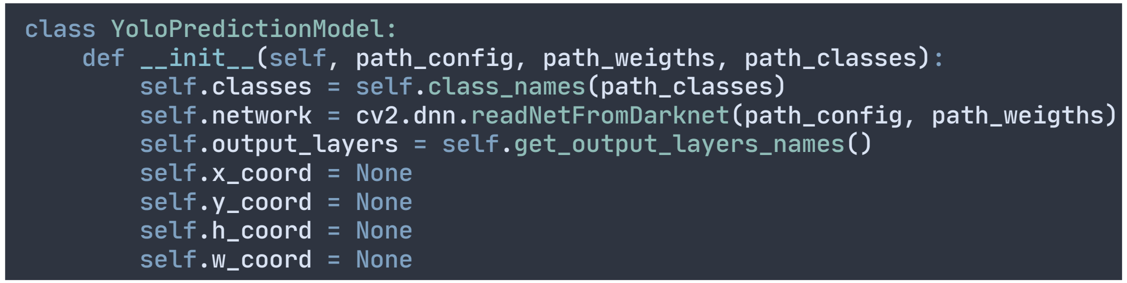


```
def predict_and_identify(self, image, yolo_output_objects, threshold=0.5):
    ...

def crop_predictions(x, y, w, h, image):
        return image[y:y+h, x:x+w, :]
```

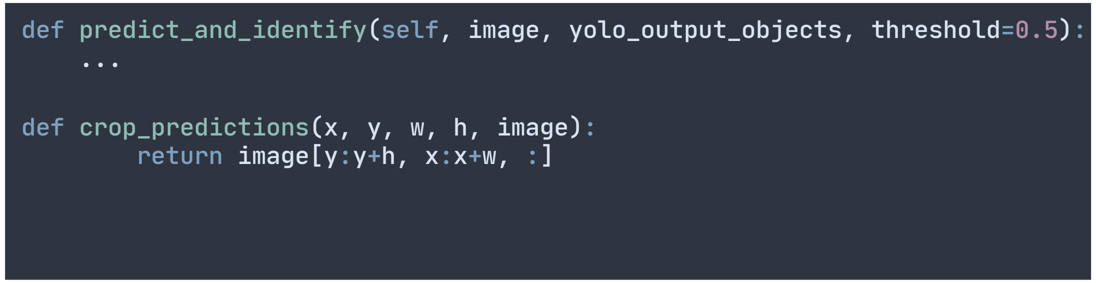
Predictions



Testing the model and extracting the EDP Boxes

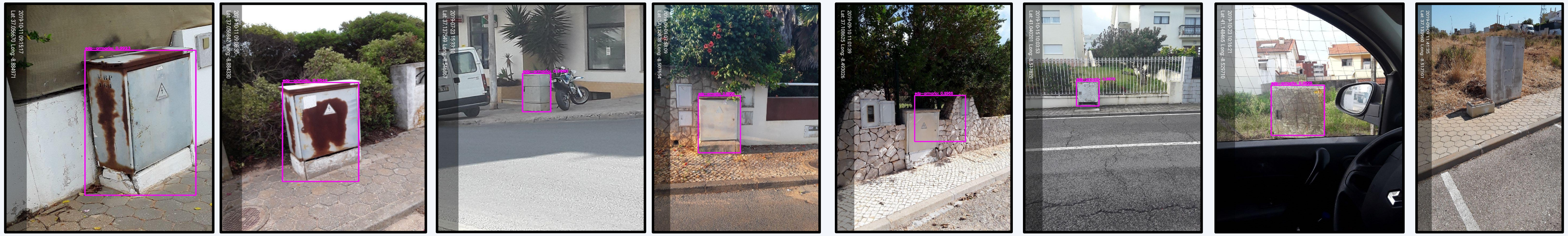


```
class YoloPredictionModel:
    def __init__(self, path_config, path_weigths, path_classes):
        self.classes = self.class_names(path_classes)
        self.network = cv2.dnn.readNetFromDarknet(path_config, path_weigths)
        self.output_layers = self.get_output_layers_names()
        self.x_coord = None
        self.y_coord = None
        self.h_coord = None
        self.w_coord = None
```



```
def predict_and_identify(self, image, yolo_output_objects, threshold=0.5):
    ...

def crop_predictions(x, y, w, h, image):
        return image[y:y+h, x:x+w, :]
```

Predictions



Scrapping