Step 1. Install Necessary Packages and CUDA Extension

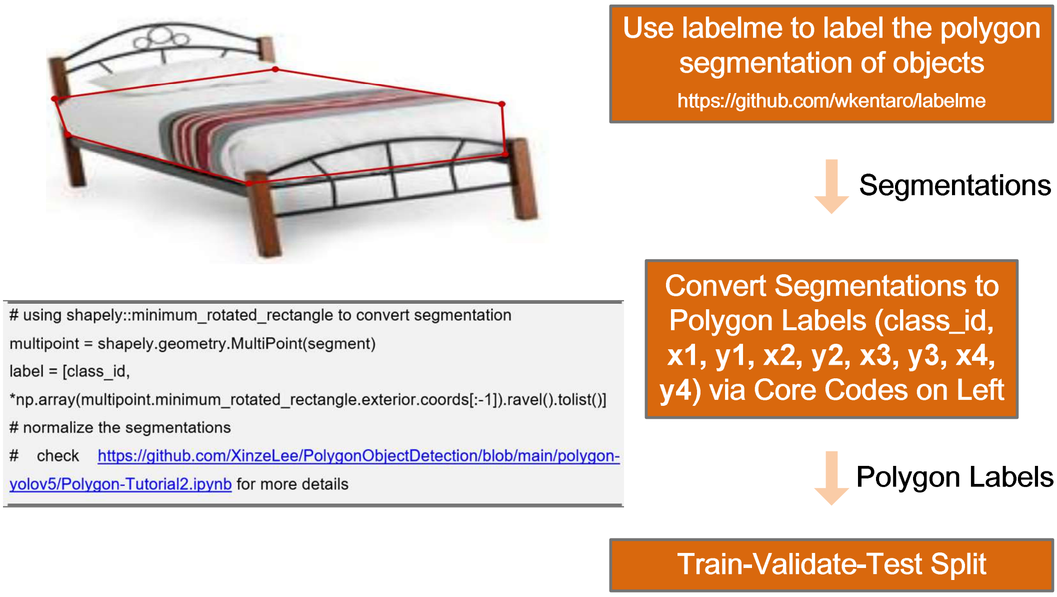
Follow the steps below to install the requirements and the CUDA extension for the computation of IOU (intersection over union) of polygon boxes.

INSTALLATION OF CUDA EXTENSION (COMMAND LINE)

```
# cd to the directory of polygon-yolov5
cd polygon-yolov5
# install python package requirements
# if want Albumentation augmentation, check requirements.txt
pip install -r requirements.txt
# install CUDA extensions
cd utils/iou_cuda
python setup.py install
cd .. && cd ..
```

Step 2. Label Data and Train-Validate-Test Split

This step aims to get the polygon-labeled data (class_id, x1, y1, x2, y2, x3, y3, x4, y4) for us to train, validate and final test. You can use any tools or apps to label the data. Take one note in mind: **ensure four corners are in sequence** (either clockwise or anti-clockwise).



Use labelme to label the polygon segmentation of objects
https://github.com/wkentaro/labelme

Segmentations

Convert Segmentations to Polygon Labels (class_id, x1, y1, x2, y2, x3, y3, x4, y4) via Core Codes on Left

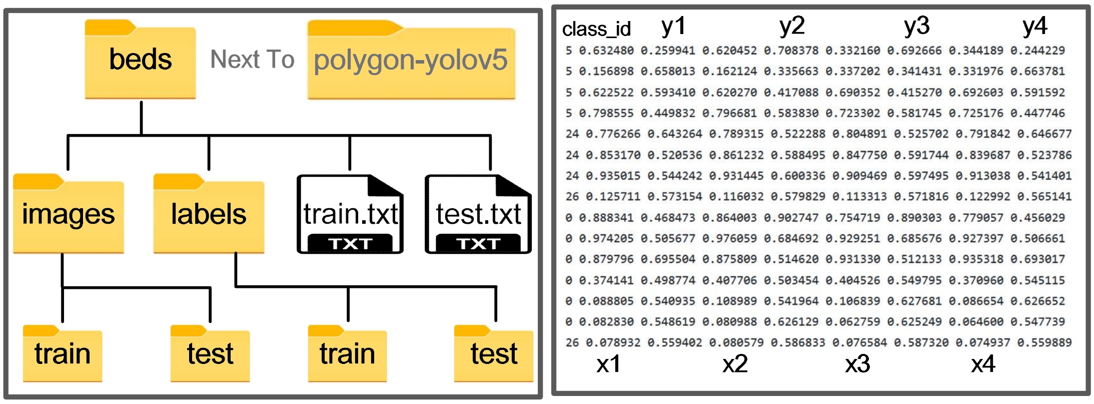
Polygon Labels

Train-Validate-Test Split

```
# using shapely::minimum_rotated_rectangle to convert segmentation
multipoint = shapely.geometry.MultiPoint(segment)
label = [class_id,
*np.array(multipoint.minimum_rotated_rectangle.exterior.coords[:-1]).ravel().tolist()]
# normalize the segmentations
# check https://github.com/XinzeLee/PolygonObjectDetection/blob/main/polygon-yolov5/Polygon-Tutorial2.ipynb for more details
```

EXEMPLAR FLOWCHART OF GENERATING POLYGON-LABELED DATA VIA FIRST METHOD

To generate polygon-labeled data, there are two ways in general: First is to label the segmentation of objects and then convert the segmentations to polygon labels; Second is to directly label four corners. Above flowchart gives you an example of the first method. The dataset directory should be set as the following figure. Please note that the final polygon labels for training, validating and testing should be (class_id, x1, y1, x2, y2, x3, y3, x4, y4), where x1 to y4 are normalized coordinates.



```
5 0.632480 0.259941 0.620452 0.708378 0.332160 0.692666 0.344189 0.244229
5 0.156898 0.658013 0.162124 0.335663 0.337202 0.341431 0.331976 0.663781
5 0.622522 0.593410 0.620270 0.417088 0.690352 0.415270 0.692603 0.591592
5 0.798555 0.449832 0.796681 0.583830 0.723302 0.581745 0.725176 0.447746
24 0.776266 0.643264 0.789315 0.522288 0.804891 0.525702 0.791842 0.646677
24 0.853170 0.520536 0.861232 0.588495 0.847750 0.591744 0.839687 0.523786
24 0.935015 0.544242 0.931445 0.600336 0.909469 0.597495 0.913038 0.541401
26 0.125711 0.573154 0.116032 0.579829 0.113313 0.571816 0.122992 0.565141
0 0.888341 0.468473 0.864003 0.902747 0.754719 0.890303 0.779057 0.456029
0 0.974205 0.505677 0.976059 0.684692 0.929251 0.685676 0.927397 0.506661
0 0.879796 0.695504 0.875809 0.514620 0.931330 0.512133 0.935318 0.693017
0 0.374141 0.498774 0.407706 0.503454 0.404526 0.549795 0.370960 0.545115
0 0.088805 0.540935 0.108989 0.541964 0.106839 0.627681 0.086654 0.626652
0 0.082830 0.548619 0.080988 0.626129 0.062759 0.625249 0.064600 0.547739
26 0.078932 0.559402 0.080579 0.586833 0.076584 0.587320 0.074937 0.559889
```

Dataset Directory

Exemplar Polygon Labels

DATASET DIRECTORY AND EXEMPLAR POLYGON LABELS

Step 3. Modify Configuration Files

There are several configuration files that you need to customize for your own dataset:

- Optimizer configuration file “data/hyp.scratch.yaml”;
- Dataset information file “data/polygon_your_dataset.yaml”;
- Model configuration file “model/polygon_yolov5.yaml”.

In “data/hyp.scratch.yaml”, you can specify optimizer hyper-parameters such as learning rate, momentum, weight decay, etc. You can also specify the data augmentation effects for your own dataset, such as translate, scale, rotation, shear, etc.

In “data/polygon_your_dataset.yaml”, you have to change the dataset path, number of classes and class names.

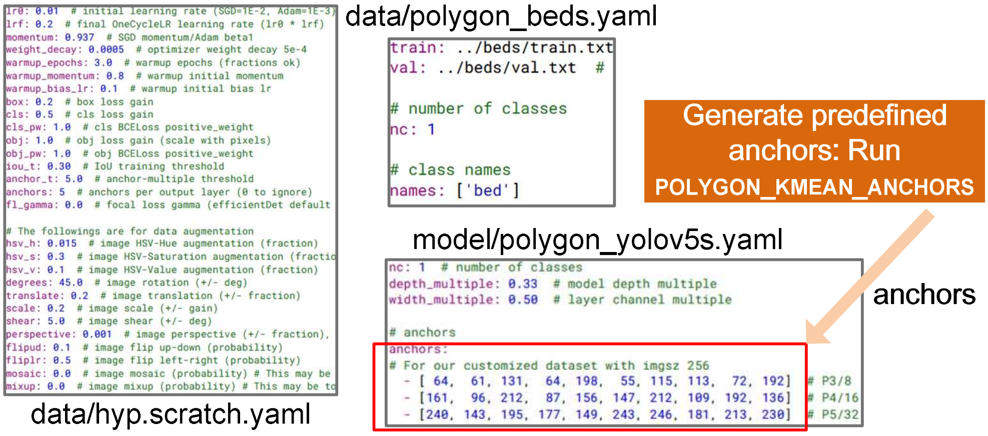
In “model/polygon_yolov5.yaml”, you have to choose the specific network structure for your dataset. You also need to change the number of classes. Afterwards, please go to the tutorial “**Polygon-Tutorial1.ipynb**”, **run** the polygon_kmean_anchors as the following to generate predefined anchors for your dataset, and **copy** the generated anchors to “model/polygon_yolov5.yaml”. In this step, you need to ensure that the image size “img_size” is the suitable and the same one for training, testing and detecting, and to ensure that the anchor threshold value “thr=5.” is the same as the “anchor_t” in optimization configuration file “data/hyp.scratch.yaml”.

RUN POLYGON_KMEAN_ANCHORS (PYTHON)

```
from utils.autoanchor import polygon_kmean_anchors

nl = 3 # number of anchor layers
na = 5 # number of anchors
img_size = 640 # image size for training and testing

datacfg = "data/ polygon_your_dataset.yaml"
anchors = polygon_kmean_anchors(datacfg, n=nl*na, gen=3000, img_size=img_size, thr=5.)
```



data/hyp.scratch.yaml

```
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.2  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.30  # IoU training threshold
anchor_t: 5.0  # anchor-multiple threshold
anchors: 5  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default

# The followings are for data augmentation
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.3  # image HSV-Saturation augmentation (fractio
hsv_v: 0.1  # image HSV-Value augmentation (fraction)
degrees: 45.0  # image rotation (+/- deg)
translate: 0.2  # image translation (+/- fraction)
scale: 0.2  # image scale (+/- gain)
shear: 5.0  # image shear (+/- deg)
perspective: 0.001  # image perspective (+/- fraction),
flipud: 0.1  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 0.0  # image mosaic (probability) # This may be
mixup: 0.0  # image mixup (probability) # This may be to
```

data/polygon_beds.yaml

```
train: ../beds/train.txt
val: ../beds/val.txt  #

# number of classes
nc: 1

# class names
names: ['bed']
```

Generate predefined anchors: Run POLYGON_KMEAN_ANCHORS

model/polygon_yolov5s.yaml

```
nc: 1  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
# For our customized dataset with imgsz 256
  - [ 64,  61, 131,  64, 198,  55, 115, 113,  72, 192]  # P3/8
  - [161,  96, 212,  87, 156, 147, 212, 109, 192, 136]  # P4/16
  - [240, 143, 195, 177, 149, 243, 246, 181, 213, 230]  # P5/32
```

anchors

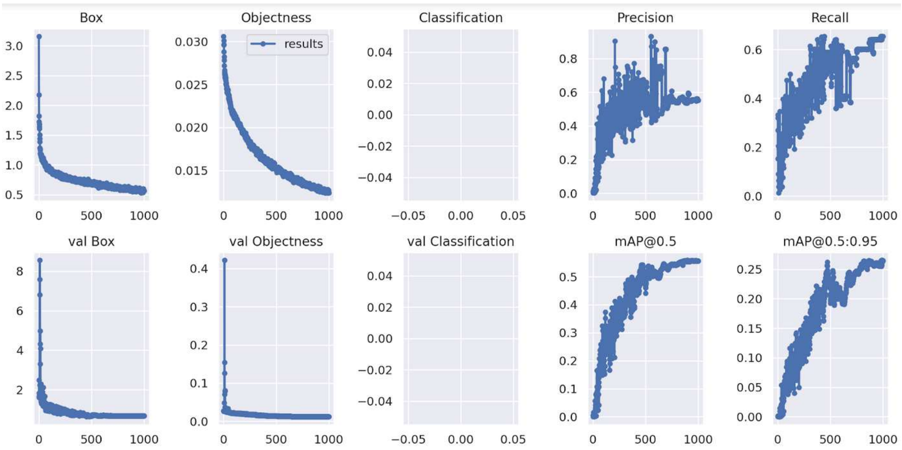
EXEMPLAR CONFIGURATION FILES

Step 4. Train Your Model

This is the time to train your model. Please use below code to train. In the beginning, you might want to choose a large training epoch for the model to overfit first, and then change the training epoch to the suitable value.

TRAIN YOUR MODEL (COMMAND LINE)

```
python polygon_train.py --weights “” --cfg polygon_yolov5.yaml \
    --data polygon_your_dataset.yaml --hyp hyp.scratch.yaml --img-size 640 \
    --epochs 400 --batch-size 16 --noautoanchor --polygon --cache
```



Exemplar Training Process on Custom Dataset

Step 5. Test and Detect via Trained Model

Please use the following codes to test and detect the trained model.

Test the Trained Model (Command Line)

```
python polygon_test.py --weights 'runs/train/exp/weights/polygon_best.pt' \
    --data polygon_your_dataset.yaml --img 640 --iou 0.4 --task val
```

Detect via the Trained Model (Command Line)

```
python polygon_detect.py --weights 'runs/train/exp/weights/polygon_best.pt' \
    --img 640 --conf 0.5 --iou-thres 0.4 \
    --source 'you_source_file_or_source'
```