Research Log

Stage 1: Environment preparation:

- **Install/check anaconda**

  conda -V

```
[x    j@node0209 ~]$ conda -V
conda 4.12.0
```

- **Create new virtual environment:**
  conda create -n yolov5 python=3.9

- **Activate the conda virtual environment:(palmetto)**
  source activate yolov5 # for linux

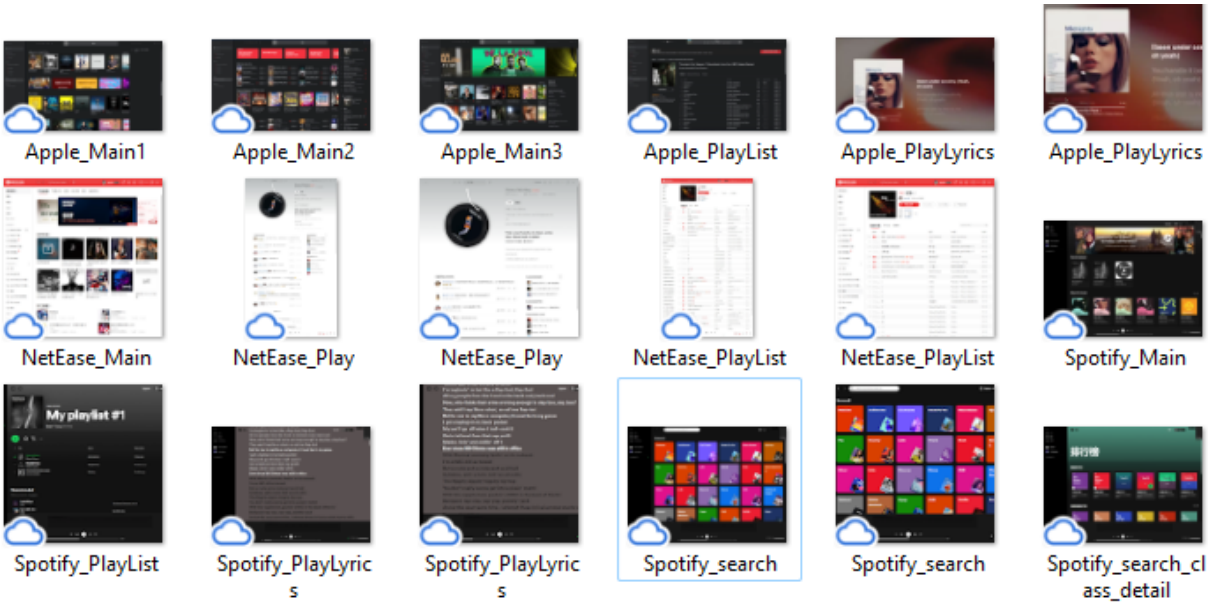
- **Download the Yolo-v5 code:**
  git clone https://github.com/ultralytics/yolov5 # clone

- **Goto the yolov5 folder:**
  cd [path_to_yolov5]

- **Install the required packages:**
  pip install -r requirement.txt

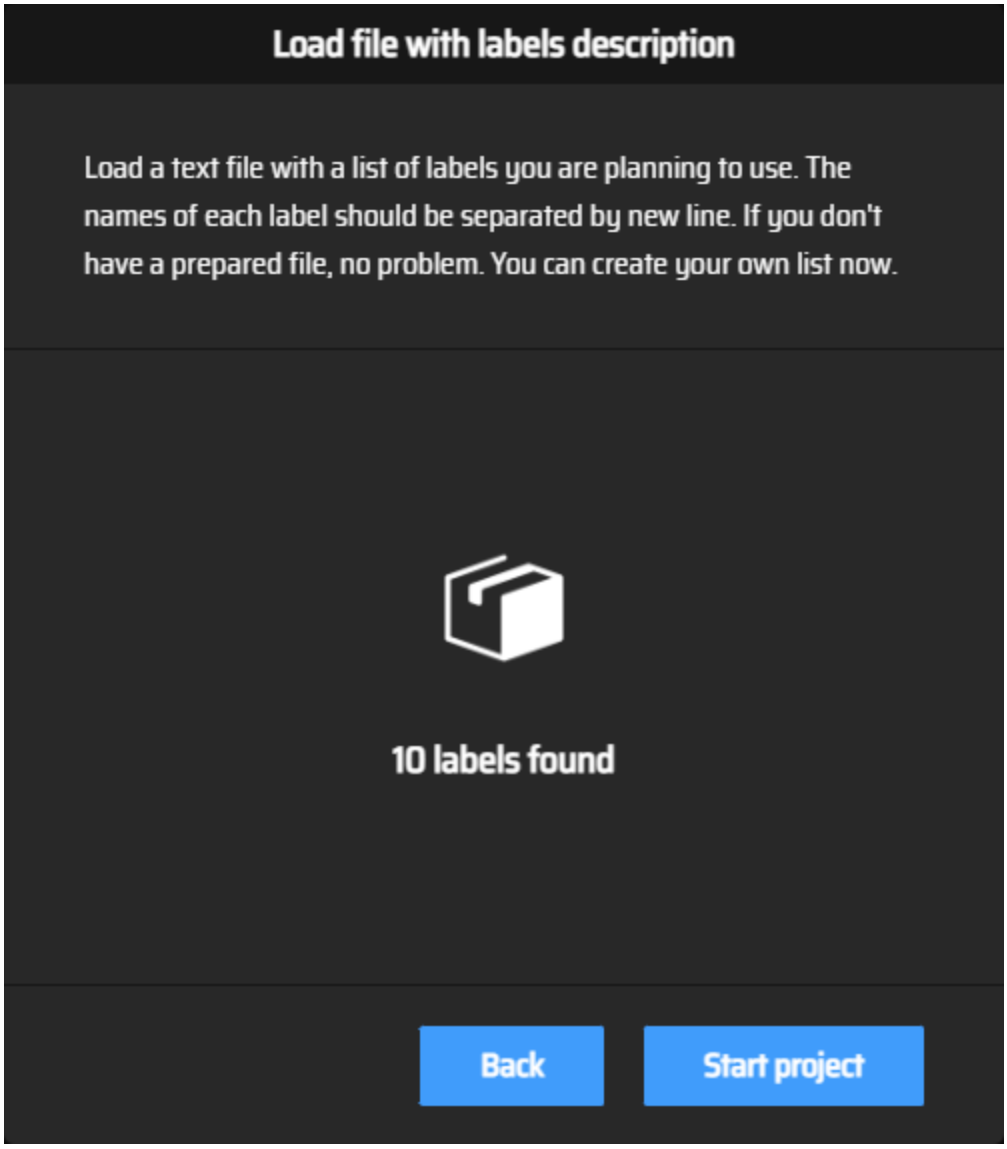
Stage 2: Labels and picture preparation:

- **Prepare the pictures:**



- **Label the picture on image labeling tools: [WebSite]**

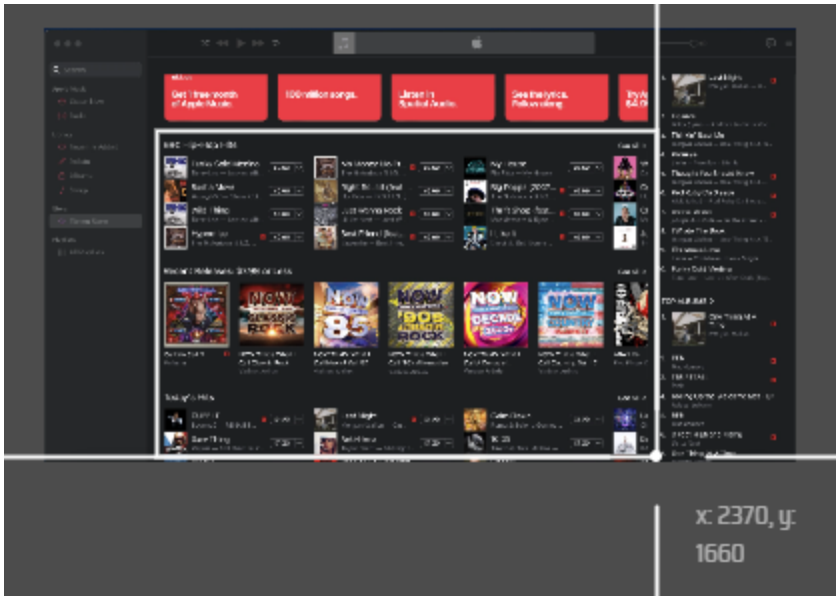
- **Load the pictures and load the labels from file:**



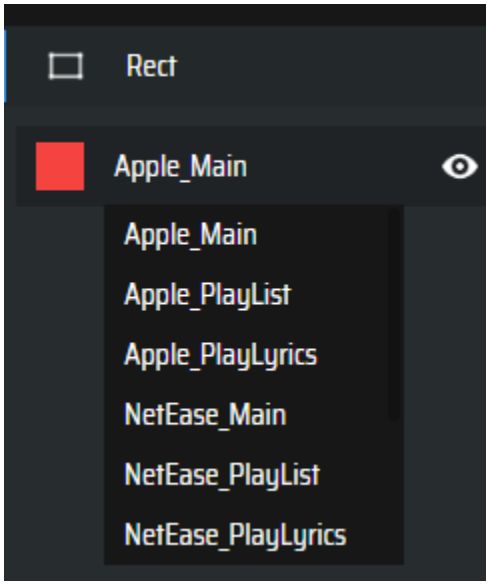
labels.txt

```
Apple_Main
Apple_PlayList
Apple_PlayLyrics
NetEase_Main
NetEase_PlayList
NetEase_PlayLyrics
NetEase_Comments
Spotify_Main
Spotify_PlayList
Spotify_PlayLyrics
```

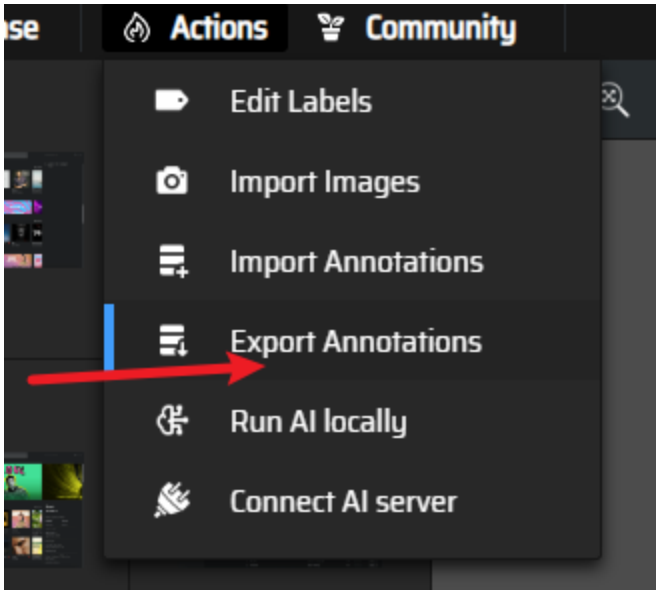
- **Start Labeling picture:**

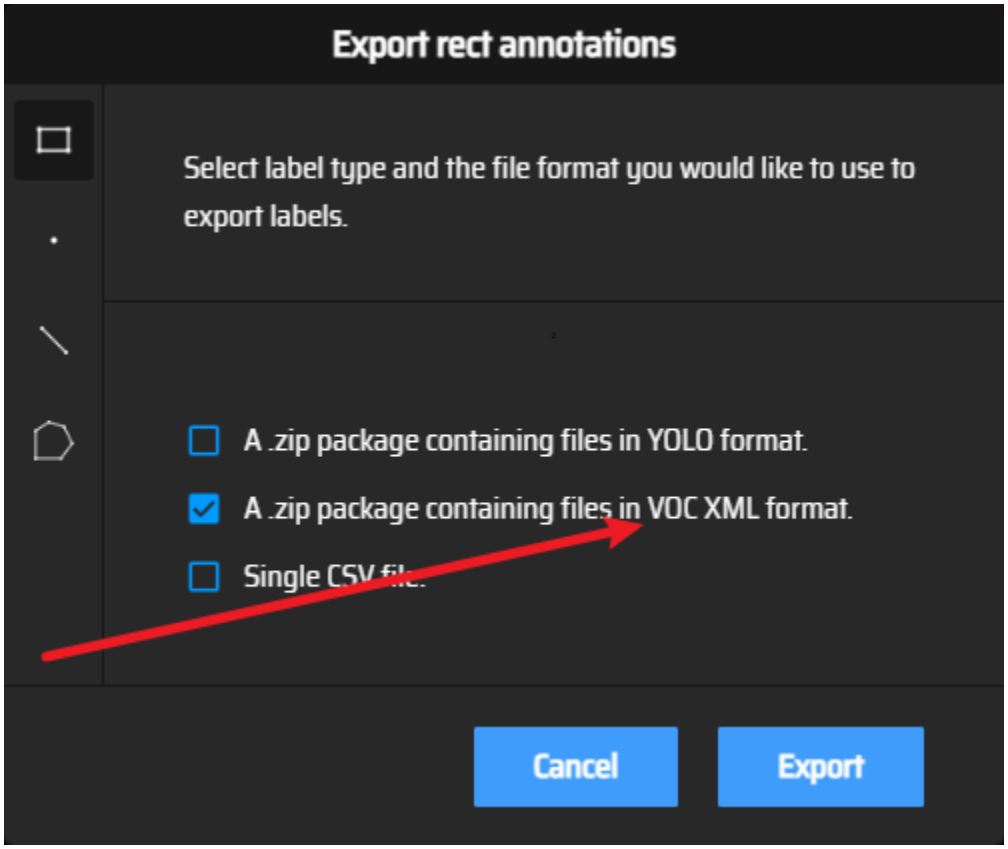


- **Select label:**

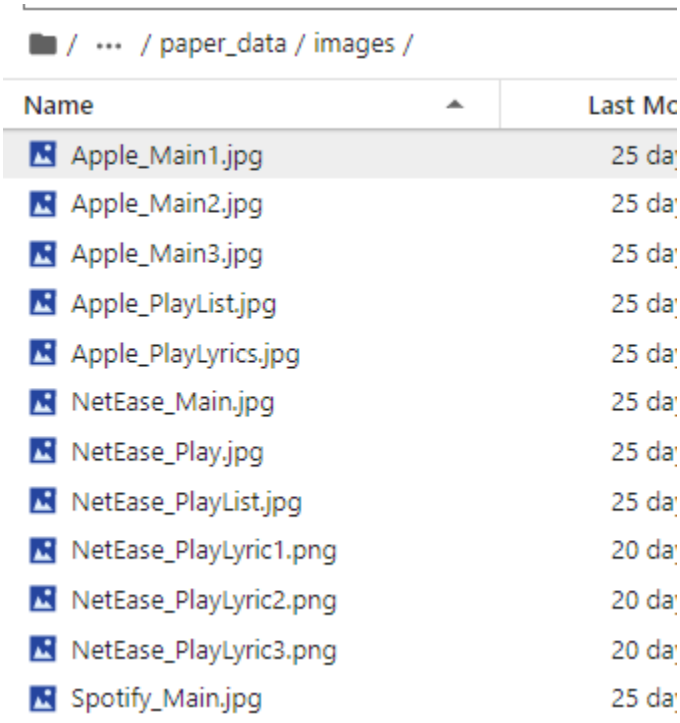


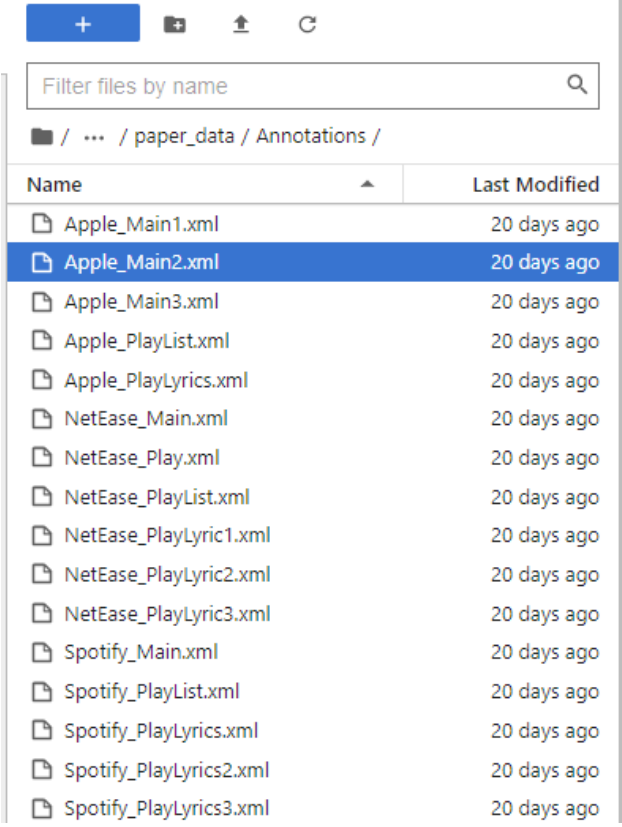
- **Export labels:**

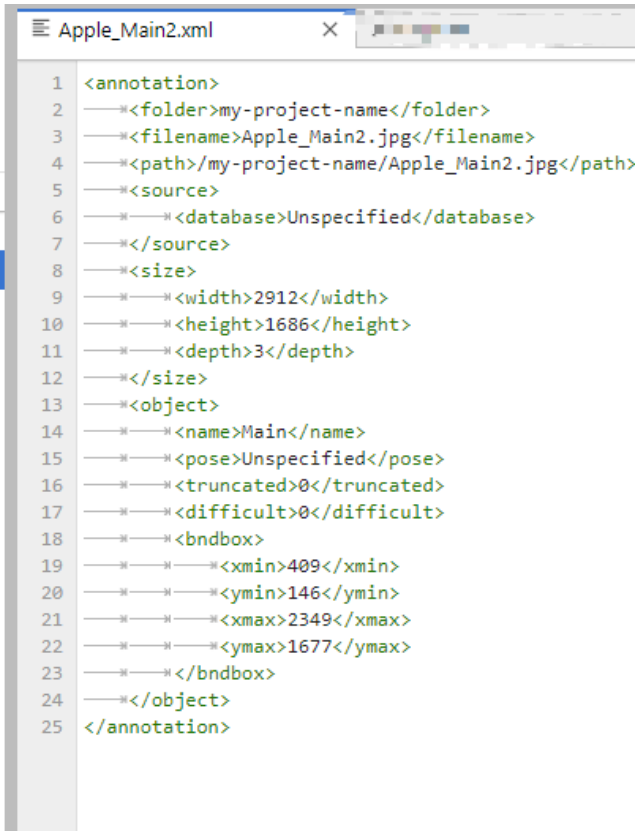

Actions
Community
Edit Labels
Import Images
Import Annotations
Export Annotations
Run AI locally
Connect AI server

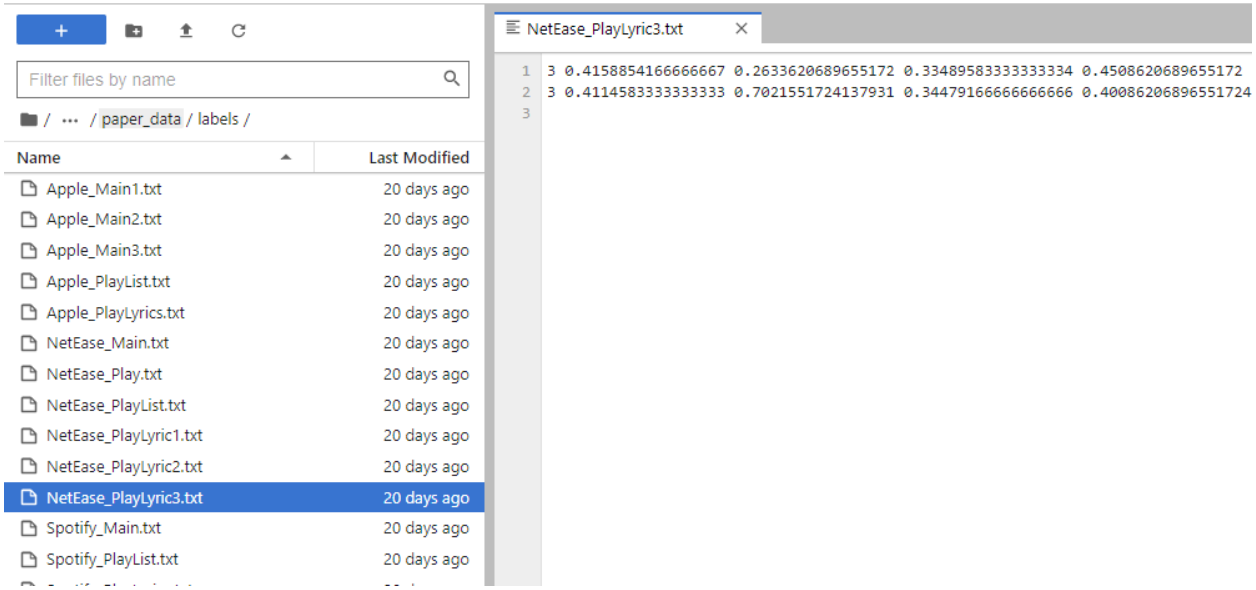

Export rect annotations
Select label type and the file format you would like to use to export labels.
A .zip package containing files in YOLO format.
A .zip package containing files in VOC XML format.
Single CSV file.
Cancel
Export

- **Put the image file and the label files(txt and XML) under the yolov5 folder:**








Filter files by name
/ ... / paper_data / labels /
Name
Last Modified
Apple_Main1.txt 20 days ago
Apple_Main2.txt 20 days ago
Apple_Main3.txt 20 days ago
Apple_PlayList.txt 20 days ago
Apple_PlayLyrics.txt 20 days ago
NetEase_Main.txt 20 days ago
NetEase_Play.txt 20 days ago
NetEase_PlayList.txt 20 days ago
NetEase_PlayLyric1.txt 20 days ago
NetEase_PlayLyric2.txt 20 days ago
NetEase_PlayLyric3.txt 20 days ago
Spotify_Main.txt 20 days ago
Spotify_PlayList.txt 20 days ago
NetEase_PlayLyric3.txt
1 3 0.4158854166666667 0.2633620689655172 0.33489583333333334 0.4508620689655172
2 3 0.4114583333333333 0.7021551724137931 0.34479166666666666 0.40086206896551724
3

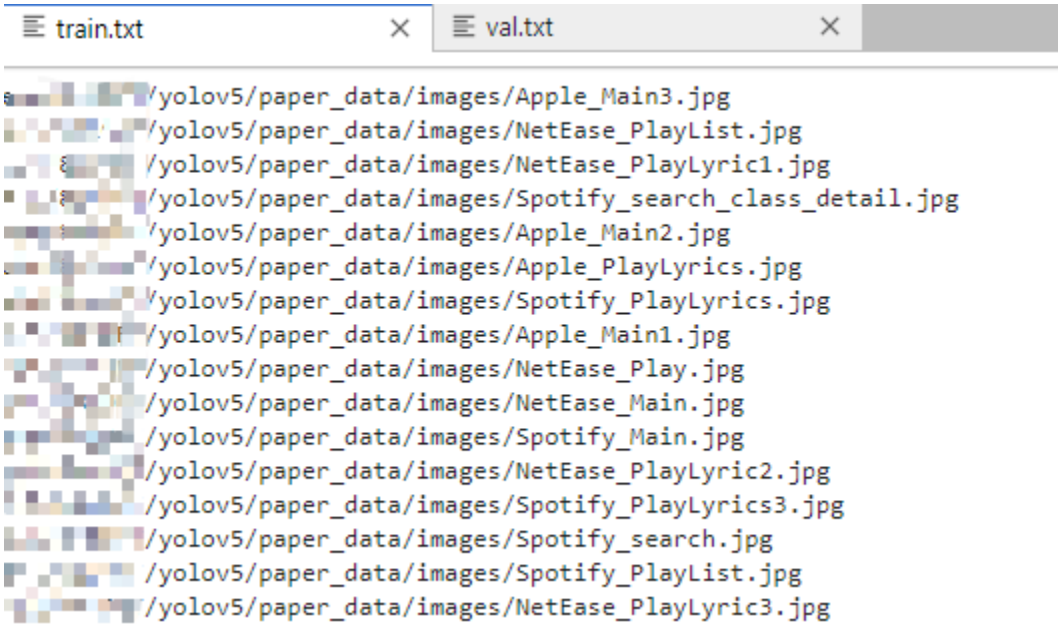
Stage 3: Dataset preparation:

- **Train test split with python:**

split_train_val.py | train.txt | val.txt

```python
# coding:utf-8
import os
import random
import argparse

parser = argparse.ArgumentParser()
# address of the xml file, modified according to your own data xml is generally stored under Annotations
parser.add_argument('--xml_path', default='/home[redacted]/yolov5/paper_data/Annotations', type=str, help='input xml label path')
# parser.add_argument('--xml_path', default='paper_data/Annotations', type=str, help='input xml label path')
#Division of datasets, address selection of own data under ImageSets/Main
# parser.add_argument('--txt_path', default='paper_data/ImageSets/Main', type=str, help='output txt label path')
parser.add_argument('--txt_path', default='/home[redacted]/yolov5/paper_data/ImageSets/Main', type=str, help='output txt label path')
opt = parser.parse_args()

trainval_percent = 1.0
train_percent = 0.9
xmlfilepath = opt.xml_path
txtsavepath = opt.txt_path
total_xml = os.listdir(xmlfilepath)
if not os.path.exists(txtsavepath):
    os.makedirs(txtsavepath)

num = len(total_xml)
print('total_xml',num)
list_index = range(num)
tv = int(num * trainval_percent)
tr = int(tv * train_percent)
trainval = random.sample(list_index, tv)
train = random.sample(trainval, tr)

file_trainval = open(txtsavepath + '/trainval.txt', 'w')
file_test = open(txtsavepath + '/test.txt', 'w')
file_train = open(txtsavepath + '/train.txt', 'w')
file_val = open(txtsavepath + '/val.txt', 'w')

for i in list_index:
    name = total_xml[i][:-4] + '\n'
    if i in trainval:
        file_trainval.write(name)
        if i in train:
            file_train.write(name)
        else:
            file_val.write(name)
    else:
        file_test.write(name)

file_trainval.close()
file_train.close()
file_val.close()
```

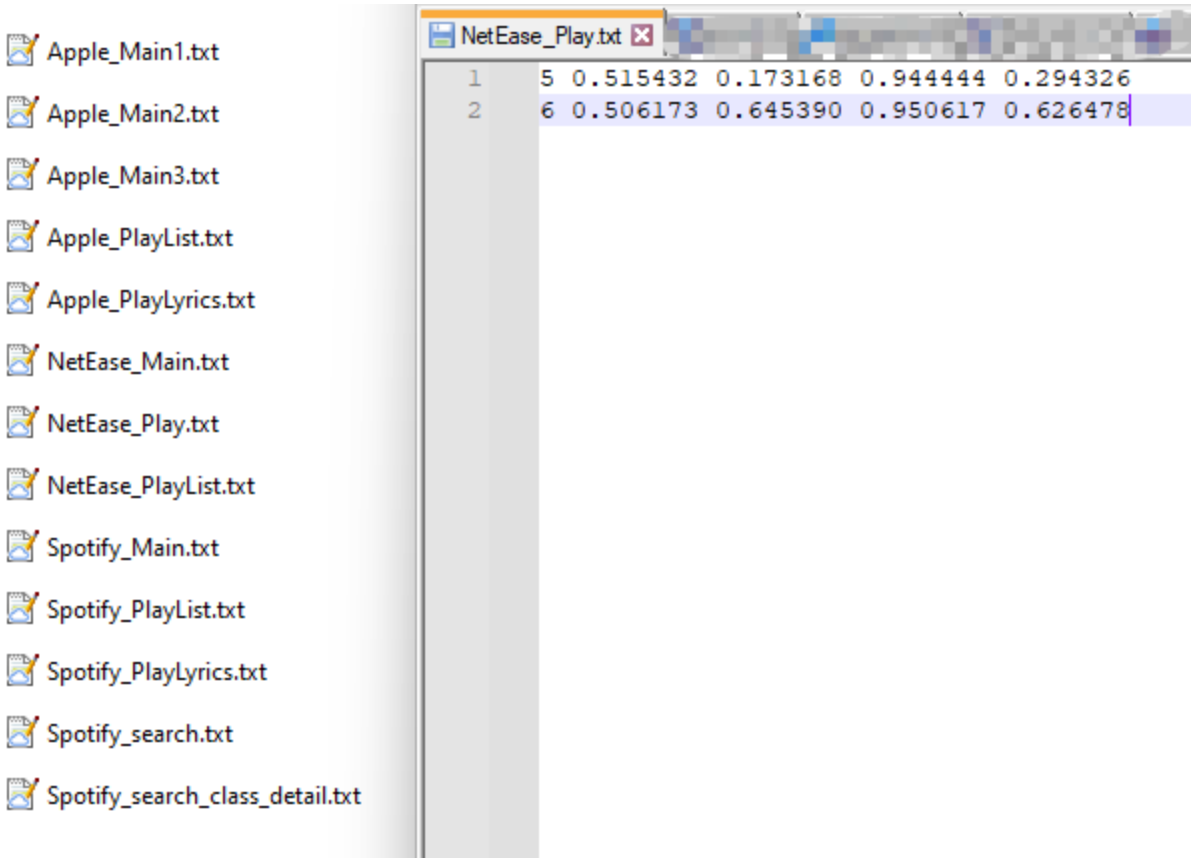
- **Run the code: python3 split_train_val.py**
- **Result: train set and validation set.**



- **Create the label files:**
- **python3 voc_label.py**

```python
# -*- coding: utf-8 -*-
import xml.etree.ElementTree as ET
import os
from os import getcwd

sets = ['train', 'val', 'test']
# classes = ["Apple_Main","Apple_PlayList","Apple_PlayLyrics","NetEase_Main","NetEase_PlayList",
#            "NetEase_PlayLyrics","NetEase_Comments","Spotify_Main","Spotify_PlayList","Spotify_PlayLyrics"]   # my own labels
classes = ["Main","PlayList","PlayLyrics","Comments"]   # my own labels
abs_path = os.getcwd()
print(abs_path)

def convert(size, box):
    dw = 1. / (size[0])
    dh = 1. / (size[1])
    x = (box[0] + box[1]) / 2.0 - 1
    y = (box[2] + box[3]) / 2.0 - 1
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x * dw
    w = w * dw
    y = y * dh
    h = h * dh
    return x, y, w, h

def convert_annotation(image_id):
    in_file = open('/home/[redacted]/yolov5/paper_data/Annotations/%s.xml' % (image_id), encoding='UTF-8')
    out_file = open('/home[redacted]/yolov5/paper_data/labels/%s.txt' % (image_id), 'w')
    tree = ET.parse(in_file)
    root = tree.getroot()
    size = root.find('size')
    w = int(size.find('width').text)
    h = int(size.find('height').text)
    for obj in root.iter('object'):
        difficult = obj.find('difficult').text
        # difficult = obj.find('Difficult').text
        cls = obj.find('name').text
        if cls not in classes or int(difficult) == 1:
            continue
        cls_id = classes.index(cls)
        xmlbox = obj.find('bndbox')
        b = (float(xmlbox.find('xmin').text), float(xmlbox.find('xmax').text), float(xmlbox.find('ymin').text),
             float(xmlbox.find('ymax').text))
        b1, b2, b3, b4 = b
        # 标注越界修正
        if b2 > w:
            b2 = w
        if b4 > h:
            b4 = h
```

Labels



- **Setup the yaml settings: the path and the labels**

```yaml
path: /home/[redacted]/yolov5/paper_data/  # dataset root dir
train: /home/[redacted]/yolov5/paper_data/images
val: /home/[redacted]/yolov5/paper_data/images
test:  # test images (optional)

# number of class
nc: 10


classes: ['Main','PlayList','PlayLyrics','Comments']
# # class names

CLASS_NAMES:  ['Main','PlayList','PlayLyrics','Comments']   # my own labels
# Classes
names:
  0: Main
  1: PlayList
  2: PlayLyrics
  3: Comments
```

Stage 4: Options preparation:

- **Setting the training parameters:**

Weights(which model), settings(yaml), epochs, batch size etc.

```
def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5m.pt', help='initial weights path')
    parser.add_argument('--cfg', type=str, default='models/yolov5m.yaml', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/eyeTrack.yaml', help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=10, help='total training epochs')
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument('--noval', action='store_true', help='only validate final epoch')
    parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
    parser.add_argument('--noplots', action='store_true', help='save no plot files')
    parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache', type=str, nargs='?', const='ram', help='image --cache ram/disk')
    parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
    parser.add_argument('--device', default='0', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
    parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
    parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')
    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
    parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')
    parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')
    parser.add_argument('--name', default='exp', help='save to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--quad', action='store_true', help='quad dataloader')
    parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
    parser.add_argument('--patience', type=int, default=100, help='EarlyStopping patience (epochs without improvement)')
    parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers: backbone=10, first3=0 1 2')
    parser.add_argument('--save-period', type=int, default=-1, help='Save checkpoint every x epochs (disabled if < 1)')
    parser.add_argument('--seed', type=int, default=0, help='Global training seed')
    parser.add_argument('--local_rank', type=int, default=-1, help='Automatic DDP Multi-GPU argument, do not modify')

    # Logger arguments
    parser.add_argument('--entity', default=None, help='Entity')
    parser.add_argument('--upload_dataset', nargs='?', const=True, default=False, help='Upload data, "val" option')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval')
    parser.add_argument('--artifact_alias', type=str, default='latest', help='Version of dataset artifact to use')

    return parser.parse_known_args()[0] if known else parser.parse_args()
```

- **Training the model:**

python train.py --img 640 --batch 8 --epoch 501 --data data/eyeTrack.yaml --cfg models/yolov5m.yaml --weights weights/yolov5m.pt --device '0'

```
-img 640   --device cpuyaml  --data data/myvoc.yaml --epoch 200 --batch-size 8 --
train: weights=weights/yolov5s.pt, cfg=models/yolov5s.yaml, data=data/myvoc.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=200, batch_size=8, imgsz=640, rect=False, resume=False, nosave=False, noval=False,
 noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False, device=cpu, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs/train, name=e
xp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
github: up to date with https://github.com/ultralytics/yolov5 ✅
Traceback (most recent call last):
  File "/home/xiangyj/PythonProject/8810ET/yolov5/train.py", line 640, in <module>
    main(opt)
  File "/home/xiangyj/PythonProject/8810ET/yolov5/train.py", line 504, in main
    check_file(opt.data), check_yaml(opt.cfg), check_yaml(opt.hyp), str(opt.weights), str(opt.project)  # checks
  File "/home/xiangyj/PythonProject/8810ET/yolov5/utils/general.py", line 491, in check_file
    assert len(files), f'File not found: {file}'  # assert file was found
AssertionError: File not found: data/myvoc.yaml
 --img 640   --device cpuml  --data data/eyeTrack.yaml --epoch 20 --batch-size 8

train: weights=weights/yolov5m.pt, cfg=models/yolov5m.yaml, data=data/eyeTrack.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=20, batch_size=8, imgsz=640, rect=False, resume=False, nosave=False, noval=Fals
e, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False, device=cpu, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs/train, name
=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=lates
t
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 📌v7.0-117-g85f6019 Python-3.9.16 torch-1.13.1+cu117 CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4
.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5 📌in ClearML
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 📌runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
2023-03-09 10:40:07.065450: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in
performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

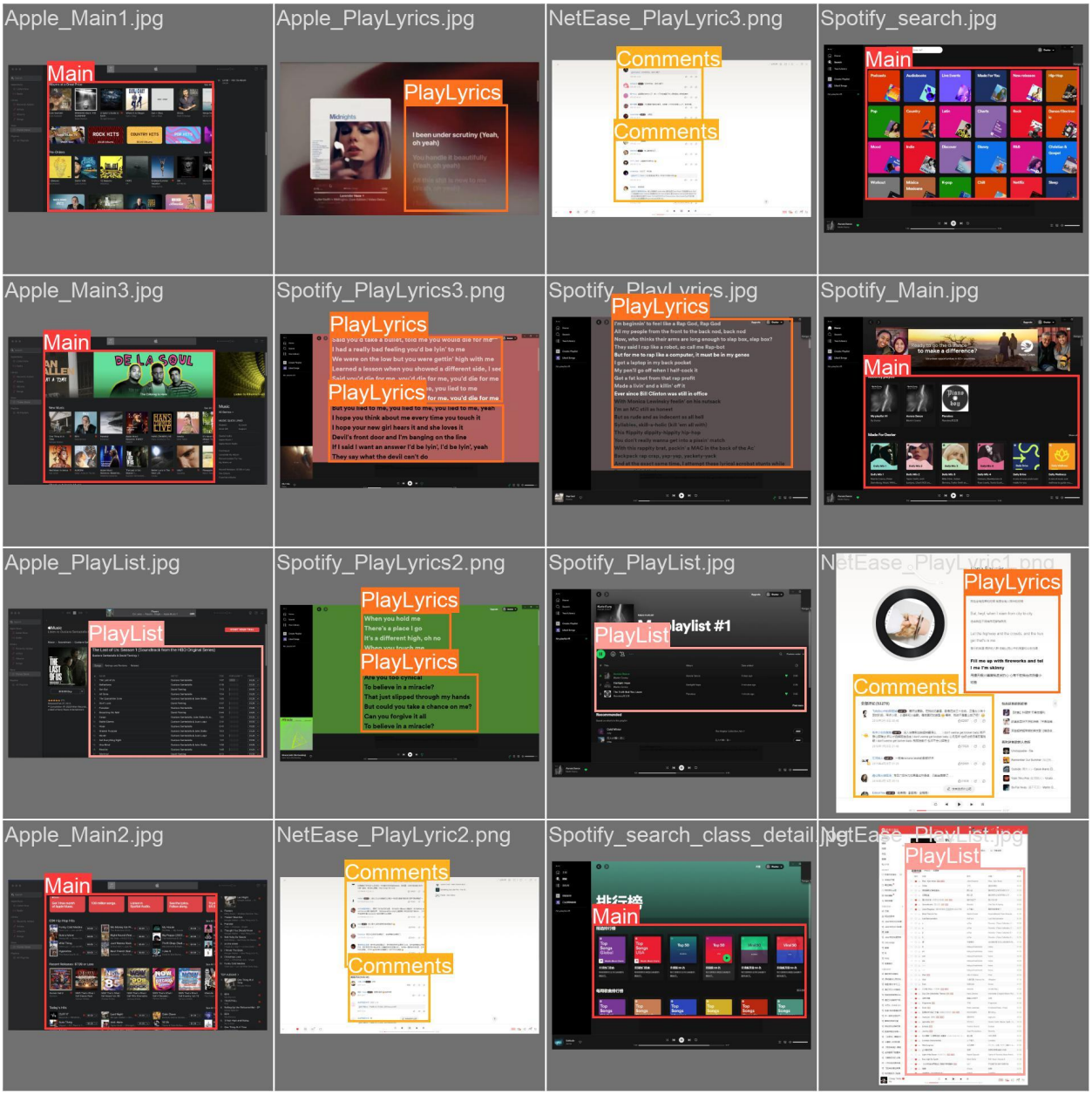
```
               Class     Images  Instances          P          R      mAP50
                 all         13         14      0.747      0.667      0.829      0.691

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
    299/299       3.3G    0.01993    0.01809    0.03063         18        640: 1
               Class     Images  Instances          P          R      mAP50
                 all         13         14      0.747      0.667      0.829      0.691

300 epochs completed in 0.135 hours.
Optimizer stripped from runs/train/exp12/weights/last.pt, 42.3MB
Optimizer stripped from runs/train/exp12/weights/best.pt, 42.3MB

Validating runs/train/exp12/weights/best.pt...
Fusing layers...
YOLOv5m summary: 212 layers, 20889303 parameters, 0 gradients, 48.0 GFLOPs
               Class     Images  Instances          P          R      mAP50
                 all         13         14       0.76      0.667      0.884      0.741
           Apple_Main         13          3      0.618          1      0.995      0.785
       Apple_PlayList         13          1      0.732          1      0.995      0.895
     Apple_PlayLyrics         13          1      0.716          1      0.995      0.697
         NetEase_Main         13          1      0.823          1      0.995      0.995
     NetEase_PlayList         13          1          1          0      0.995      0.895
   NetEase_PlayLyrics         13          1          1          0      0.995      0.796
     NetEase_Comments         13          1          1          0      0.332      0.298
         Spotify_Main         13          3       0.44          1       0.83      0.698
     Spotify_PlayList         13          2       0.51          1      0.828      0.613
Results saved to runs/train/exp12
(yolov5) [xiangyj@node0214 yolov5]$
```

- **Above is the first version of the yolov5 model, I put too many labels in this version. And the performance was bad, only 75% acc, and it will have bad result in the real classification task.**

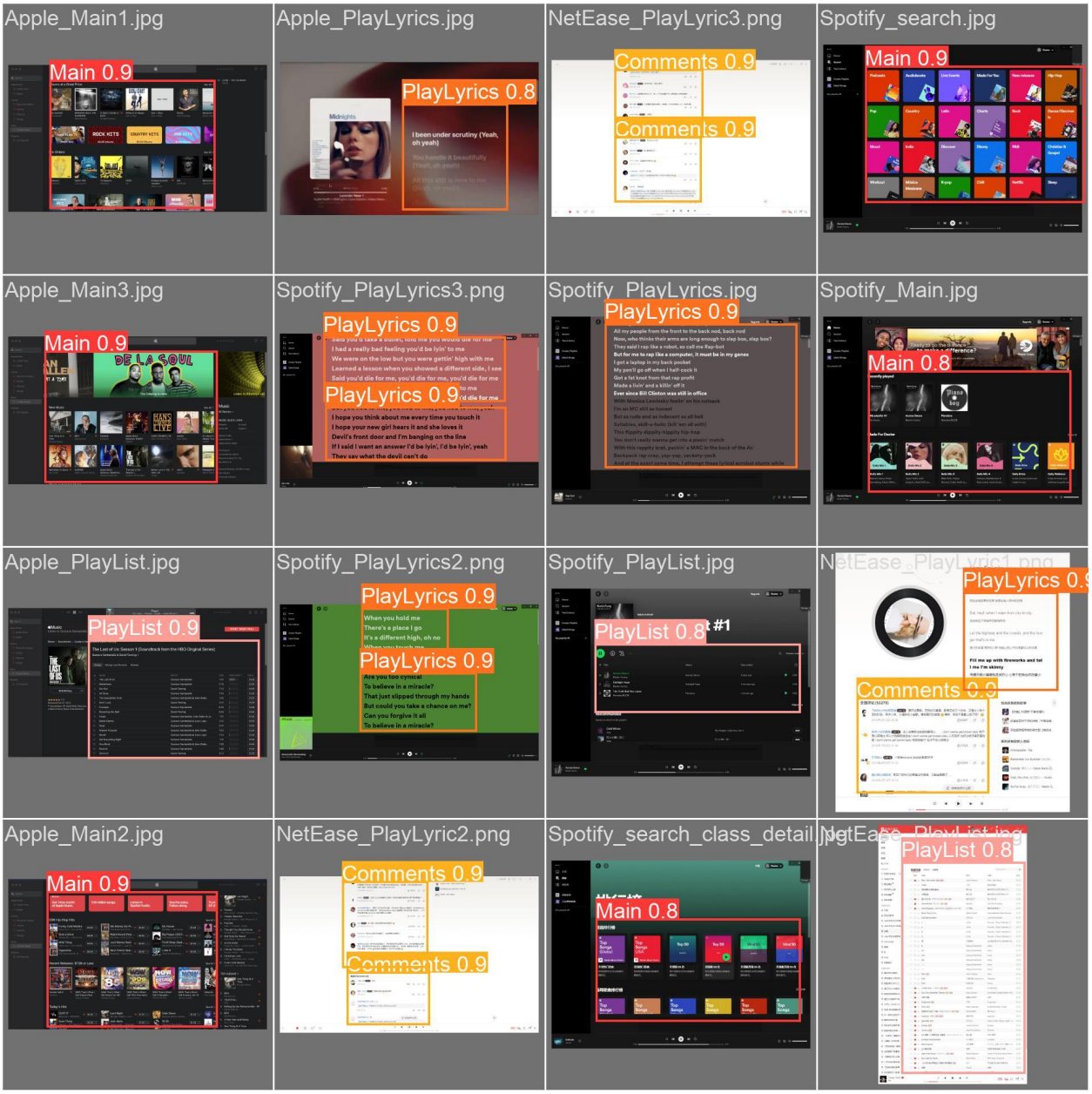
- **In the second version I re-labeled the data into 4 classes and added some extra samples, the result was good.(99% acc)**

```
    500/500      3.6G    0.03126    0.01943   0.006795         9        640: 100%|██████████| 3/3 [00:00<00:00, 11.53it/s]
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 2/2 [00:00<00:00, 15.21it/s]
                 all         18         25      0.985          1      0.995      0.891

501 epochs completed in 0.205 hours.
Optimizer stripped from runs/train/exp17/weights/last.pt, 42.1MB
Optimizer stripped from runs/train/exp17/weights/best.pt, 42.1MB

Validating runs/train/exp17/weights/best.pt...
Fusing layers...
YOLOv5m summary: 212 layers, 20865057 parameters, 0 gradients, 47.9 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 2/2 [00:00<00:00, 11.50it/s]
                 all         18         25      0.986          1      0.995      0.902
                Main         18          8      0.992          1      0.995       0.93
            PlayList         18          3      0.984          1      0.995      0.895
          PlayLyrics         18          8      0.987          1      0.995       0.91
            Comments         18          6       0.98          1      0.995      0.874
Results saved to runs/train/exp17
[xiangyj@node0055 yolov5]$
```

- **The labels:**


Apple_Main1.jpg
Main
Apple_PlayLyrics.jpg
PlayLyrics
NetEase_PlayLyric3.png
Comments
Comments
Spotify_search.jpg
Main
Apple_Main3.jpg
Main
Spotify_PlayLyrics3.png
PlayLyrics
PlayLyrics
Spotify_PlayLyrics.jpg
PlayLyrics
Spotify_Main.jpg
Main
Apple_PlayList.jpg
PlayList
Spotify_PlayLyrics2.png
PlayLyrics
PlayLyrics
Spotify_PlayList.jpg
PlayList
NetEase_PlayLyric1.png
PlayLyrics
Comments
Apple_Main2.jpg
Main
NetEase_PlayLyric2.png
Comments
Comments
Spotify_search_class_detail.jpg
Main
NetEase_PlayList.jpg
PlayList

- **The prediction result:**



- **Now I'm working on the video object detection and extraction of the bounding box into tabular data, and the eye tracking data extraction.**

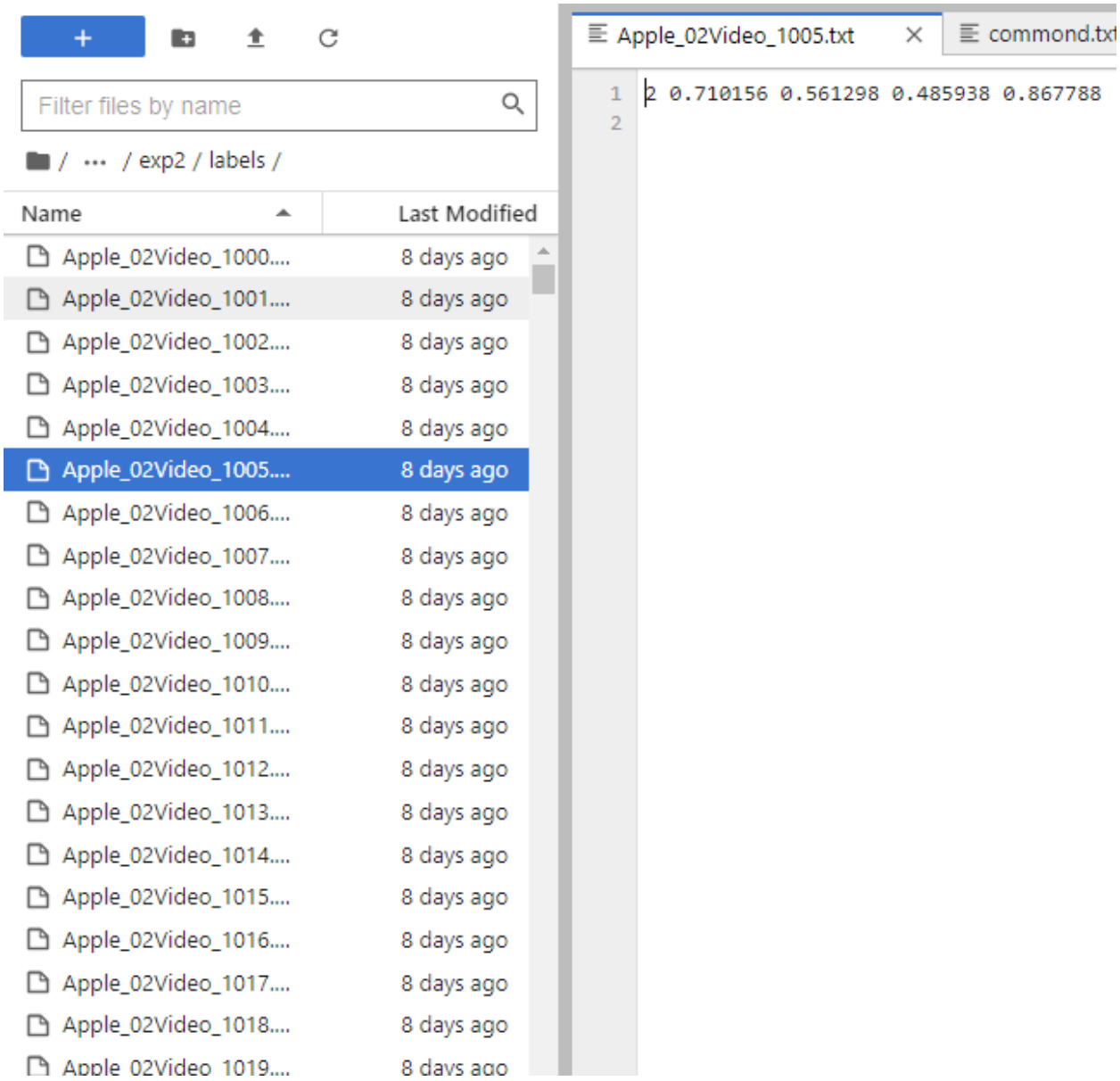
videoDetect.py

```python
import cv2
import numpy as np
from yolov5.yolov5 import YOLOv5

# Load the trained YOLO model
model = YOLOv5(weights='yolov5s.pt', conf_thres=0.5, iou_thres=0.5)

# Read category labels
with open('data.yaml', 'r') as f:
    data = yaml.load(f, Loader=yaml.FullLoader)
classes = data['names']

# Open video stream
# cap = cv2.VideoCapture(0)   # Use the default camera
cap = cv2.VideoCapture('/home/ [redacted] /yolov5/paperdata/video.mp4')
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Perform target detection
    results = model.detect(frame)

    # Draw bounding boxes and category labels
    for result in results:
        x1, y1, x2, y2, conf, cls = result
        cls_name = classes[int(cls)]
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(frame, cls_name, (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        #todo: output the bouding boxex labels data into csv files.

    # Display the processed frames
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) == ord('q'):
        break

# Release resources
cap.release()
cv2.destroyAllWindows()
```

Video Detection worked

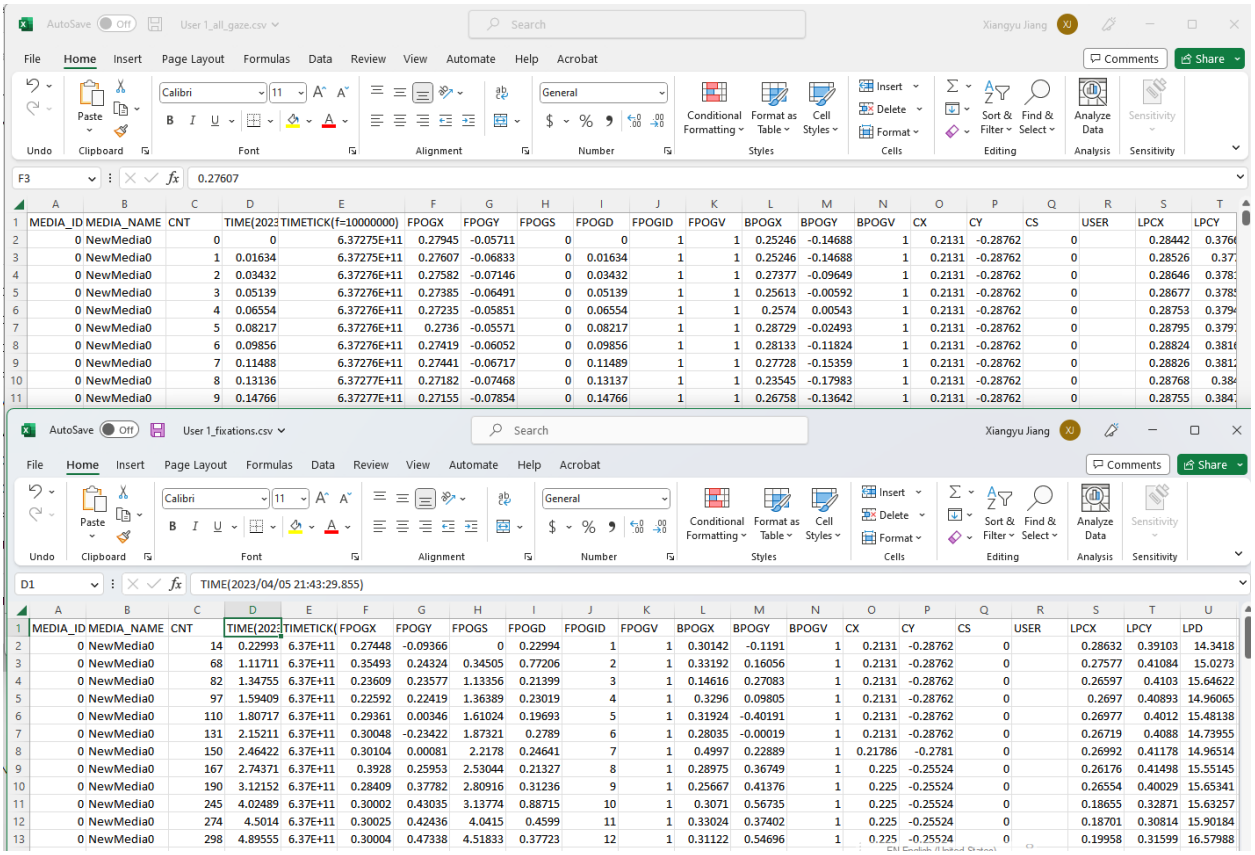
```
[redacted] yolov5]$ python detect.py --weights runs/train/exp17/weights/best.pt --source paper_data/Videos/Apple_02Video.mp4 --save-txt
detect: weights=['runs/train/exp17/weights/best.pt'], source=paper_data/Videos/Apple_02Video.mp4, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, devi
rue, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickn
conf=False, half=False, dnn=False, vid_stride=1
YOLOv5 🚀 v7.0-117-g85f6019 Python-3.9.12 torch-1.13.1+cu117 CPU

Fusing layers...
YOLOv5m summary: 212 layers, 20865057 parameters, 0 gradients, 47.9 GFLOPs
video 1/1 (1/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 103.3ms
video 1/1 (2/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.6ms
video 1/1 (3/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 67.2ms
video 1/1 (4/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 72.5ms
video 1/1 (5/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.6ms
video 1/1 (6/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.7ms
video 1/1 (7/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 63.2ms
video 1/1 (8/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 63.3ms
video 1/1 (9/2105) /hc[redacted] yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.2ms
video 1/1 (10/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.3ms
video 1/1 (11/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 83.5ms
video 1/1 (12/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.1ms
video 1/1 (13/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.8ms
video 1/1 (14/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 60.6ms
video 1/1 (15/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 60.6ms
video 1/1 (16/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 152.8ms
video 1/1 (17/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 68.0ms
video 1/1 (18/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 64.1ms
video 1/1 (19/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.6ms
video 1/1 (20/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 75.9ms
video 1/1 (21/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 76.8ms
video 1/1 (22/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.1ms
video 1/1 (23/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.8ms
video 1/1 (24/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 63.2ms
video 1/1 (25/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.7ms
video 1/1 (26/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 62.6ms
video 1/1 (27/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 63.2ms
video 1/1 (28/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 61.2ms
video 1/1 (29/2105) /h[redacted] /yolov5/paper_data/Videos/Apple_02Video.mp4: 416x640 (no detections), 63.0ms
```

The speed is about 15 frames per second.



Here is the detection result, each class of label in each frame and the position.

For the Eye-tracking part, I have tested and output the Eye-tracking data with GazePoint Analysis.



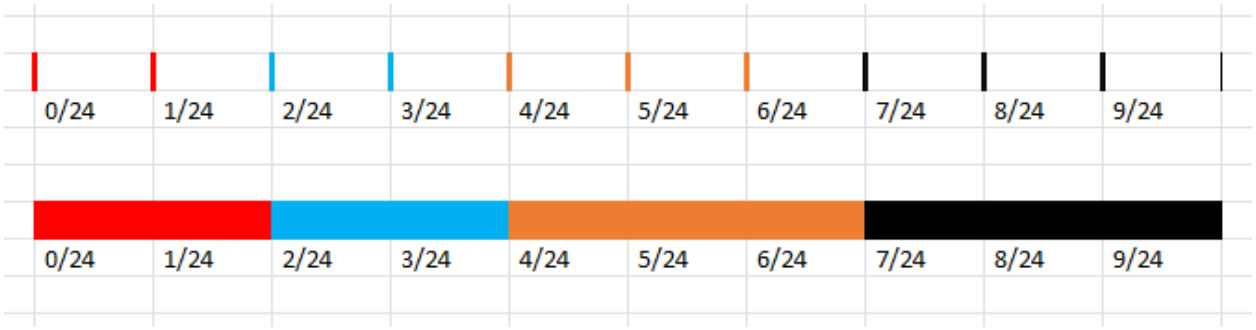
MEDIA_ID	MEDIA_NAME	CNT	TIME(2023	TIMETICK(f=10000000)	FPOGX	FPOGY	FPOGS	FPOGD	FPOGID	FPOGV	BPOGX	BPOGY	BPOGV	CX	CY	CS	USER	LPCX	LPCY
0	NewMedia0	0	0	6.37275E+11	0.27945	-0.05711	0	0	1	1	0.25246	-0.14688	1	0.2131	-0.28762	0		0.28442	0.376
0	NewMedia0	1	0.01634	6.37275E+11	0.27607	-0.06833	0	0.01634	1	1	0.25246	-0.14688	1	0.2131	-0.28762	0		0.28526	0.37
0	NewMedia0	2	0.03432	6.37276E+11	0.27582	-0.07146	0	0.03432	1	1	0.27377	-0.09649	1	0.2131	-0.28762	0		0.28646	0.378
0	NewMedia0	3	0.05139	6.37276E+11	0.27385	-0.06491	0	0.05139	1	1	0.25613	-0.00592	1	0.2131	-0.28762	0		0.28677	0.378
0	NewMedia0	4	0.06554	6.37276E+11	0.27235	-0.05851	0	0.06554	1	1	0.2574	0.00543	1	0.2131	-0.28762	0		0.28753	0.379
0	NewMedia0	5	0.08217	6.37276E+11	0.2736	-0.05571	0	0.08217	1	1	0.28729	-0.02493	1	0.2131	-0.28762	0		0.28795	0.379
0	NewMedia0	6	0.09856	6.37276E+11	0.27419	-0.06052	0	0.09856	1	1	0.28133	-0.11824	1	0.2131	-0.28762	0		0.28824	0.381
0	NewMedia0	7	0.11488	6.37276E+11	0.27441	-0.06717	0	0.11489	1	1	0.27728	-0.15359	1	0.2131	-0.28762	0		0.28826	0.381
0	NewMedia0	8	0.13136	6.37277E+11	0.27182	-0.07468	0	0.13137	1	1	0.23545	-0.17983	1	0.2131	-0.28762	0		0.28768	0.384
0	NewMedia0	9	0.14766	6.37277E+11	0.27155	-0.07854	0	0.14766	1	1	0.26758	-0.13642	1	0.2131	-0.28762	0		0.28755	0.384

MEDIA_ID	MEDIA_NAME	CNT	TIME(2023	TIMETICK(	FPOGX	FPOGY	FPOGS	FPOGD	FPOGID	FPOGV	BPOGX	BPOGY	BPOGV	CX	CY	CS	USER	LPCX	LPCY	LPD
0	NewMedia0	14	0.22993	6.37E+11	0.27448	-0.09366	0	0.22994	1	1	0.30142	-0.1191	1	0.2131	-0.28762	0		0.28632	0.39103	14.3418
0	NewMedia0	68	1.11711	6.37E+11	0.35493	0.24324	0.34505	0.77206	2	1	0.33192	0.16056	1	0.2131	-0.28762	0		0.27577	0.41084	15.0273
0	NewMedia0	82	1.34755	6.37E+11	0.23609	0.23577	1.13356	0.21399	3	1	0.14616	0.27083	1	0.2131	-0.28762	0		0.26597	0.4103	15.64622
0	NewMedia0	97	1.59409	6.37E+11	0.22592	0.22419	1.36389	0.23019	4	1	0.3296	0.09805	1	0.2131	-0.28762	0		0.2697	0.40893	14.96065
0	NewMedia0	110	1.80717	6.37E+11	0.29361	0.00346	1.61024	0.19693	5	1	0.31924	-0.40191	1	0.2131	-0.28762	0		0.26977	0.4012	15.48138
0	NewMedia0	131	2.15211	6.37E+11	0.30048	-0.23422	1.87321	0.2789	6	1	0.28035	-0.00019	1	0.2131	-0.28762	0		0.26719	0.4088	14.73955
0	NewMedia0	150	2.46422	6.37E+11	0.30104	0.00081	2.2178	0.24641	7	1	0.4997	0.22889	1	0.21786	-0.2781	0		0.26992	0.41178	14.96514
0	NewMedia0	167	2.74371	6.37E+11	0.3928	0.25953	2.53044	0.21327	8	1	0.28975	0.36749	1	0.225	-0.25524	0		0.26176	0.41498	15.55145
0	NewMedia0	190	3.12152	6.37E+11	0.28409	0.37782	2.80916	0.31236	9	1	0.25667	0.41376	1	0.225	-0.25524	0		0.26554	0.40029	15.65341
0	NewMedia0	245	4.02489	6.37E+11	0.30002	0.43035	3.13774	0.88715	10	1	0.3071	0.56735	1	0.225	-0.25524	0		0.18655	0.32871	15.63257
0	NewMedia0	274	4.5014	6.37E+11	0.30025	0.42436	4.0415	0.4599	11	1	0.33024	0.37402	1	0.225	-0.25524	0		0.18701	0.30814	15.90184
0	NewMedia0	298	4.89555	6.37E+11	0.30004	0.47338	4.51833	0.37723	12	1	0.31122	0.54696	1	0.225	-0.25524	0		0.19958	0.31599	16.57988

For the result analysis part:

The Video Frame and the EyeTracking are not in the same frequency(24FPS and 60Hz), and the timeline does not match, and the timing and location of the samples are subject to error. To address these issues, two possible solutions were proposed. The first solution is to use the KNN method with K = 3 and 5 to smooth the distribution of the video object position data and eye tracking fixation data respectively. This approach can help to reduce the noise and errors in the data and improve the accuracy of the analysis. The second solution is to use a time-based comparison method to measure the position of the object for each time segment. This method can help to compensate for the timing and location errors caused by the differences in the frequency of the Video Frame and the EyeTracking.

video filter		fixation filter
1/3		1/5
1/3		1/5
1/3		1/5
		1/5
		1/5



Eye Tracking data analysis

Eye Tracking data analysis

pdf

```
!pwd
```

[1]:
```
import pandas as pd
import numpy as np
def loadData(fileName):
    Dataset = pd.read_table("~/PythonProject/8810ET/Detect/data/{fileName}.csv".format(fileName = fileName),sep = ',')
    return Dataset
```

Load Gaze Data

[2]:
```
FileName = 'Netease_User 0_all_gaze'
basePath = '/home/xiangyj'
# Dataset = pd.read_table("~/PythonProject/8810ET/Detect/data/{FileName}.csv".format(FileName = FileName),sep = ',')
# Dataset.head()
Dataset = loadData(FileName)
Dataset.head()
```

[2]:

	MEDIA_ID	MEDIA_NAME	CNT	TIME(2023/04/24 19:08:39.611)	TIMETICK(f=10000000)	FPOGX	FPOGY	FPOGS	FPOGD	FPOGID	...	TTL0	TTL1	TTLV	PIXS	PIXV	AOI	SACCADE_MAG	SACCADE_DIR	VID_FRAME	Unnamed: 51
0	0	NewMedia0	0	0.00000	544366653480	0.41083	0.21048	0.0	0.00000	1	...	1	1	0	0.0	0	NaN	0.0	0.0	0	NaN
1	0	NewMedia0	1	0.01652	544366818723	0.40611	0.21883	0.0	0.01652	1	...	1	1	0	0.0	0	NaN	0.0	0.0	0	NaN
2	0	NewMedia0	2	0.03286	544366982070	0.40026	0.22444	0.0	0.03286	1	...	1	1	0	0.0	0	NaN	0.0	0.0	0	NaN
3	0	NewMedia0	3	0.04949	544367148372	0.39652	0.23085	0.0	0.04949	1	...	1	1	0	0.0	0	NaN	0.0	0.0	0	NaN
4	0	NewMedia0	4	0.06588	544367312254	0.39548	0.23786	0.0	0.06588	1	...	1	1	0	0.0	0	NaN	0.0	0.0	0	NaN

5 rows × 52 columns

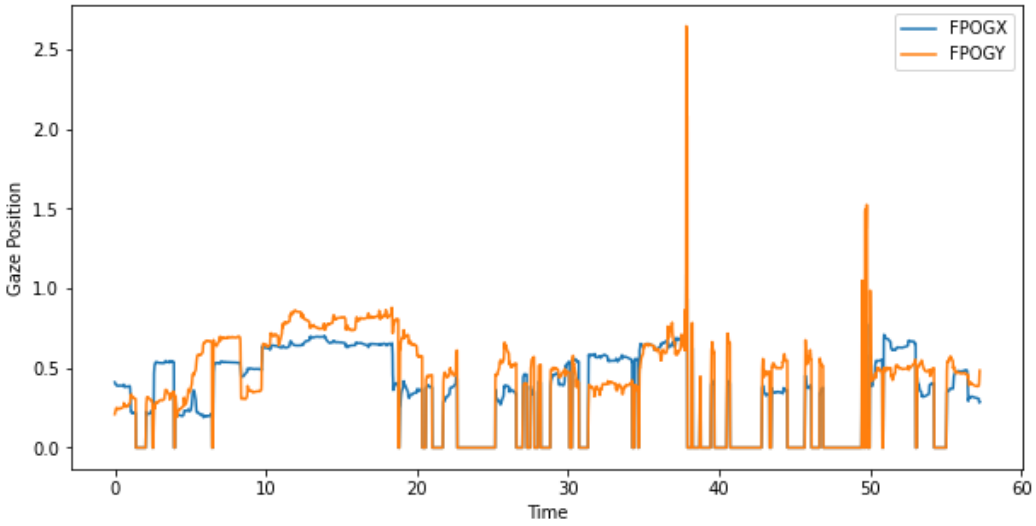
[3]:
```
times = Dataset.iloc[:, 3]

X= Dataset.iloc[:, 5]  # 5:FPOGX, 11 BPOGX
Y= Dataset.iloc[:, 6]  # 6:FPOGY, 12 BPOGY
times.tail()
```

[3]:
```
3480    57.17117
3481    57.18732
3482    57.20366
3483    57.21998
3484    57.23638
Name: TIME(2023/04/24 19:08:39.611), dtype: float64
```

Original data

data preview & pre-process

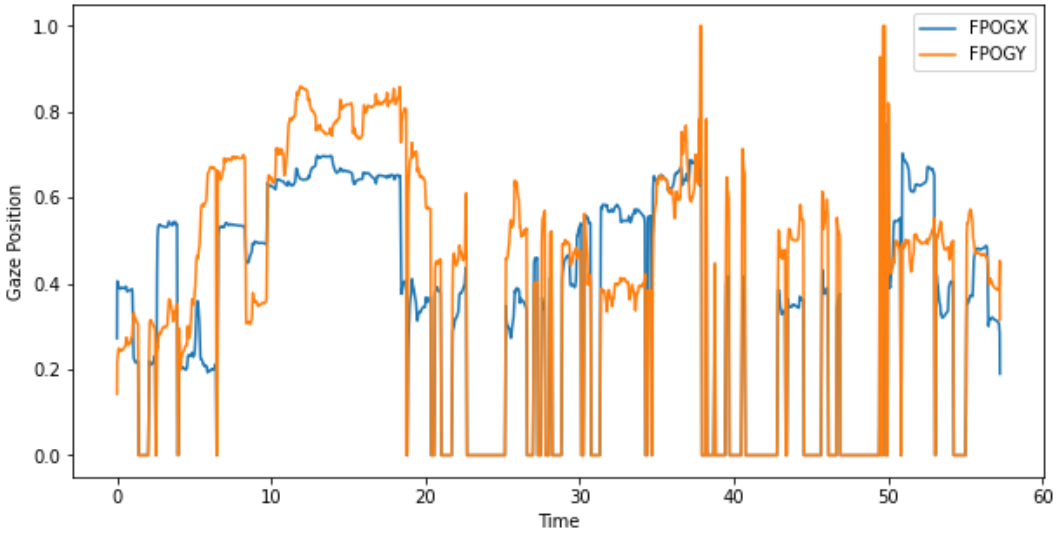
```
# create line plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,5))
ax.plot(times, X, label='FPOGX')
ax.plot(times, Y, label='FPOGY')
ax.legend()
ax.set_xlabel('Time')
ax.set_ylabel('Gaze Position')
plt.savefig(FileName+'Gazedata_kernel1.pdf',bbox_inches = 'tight', format='pdf')
plt.show()
```



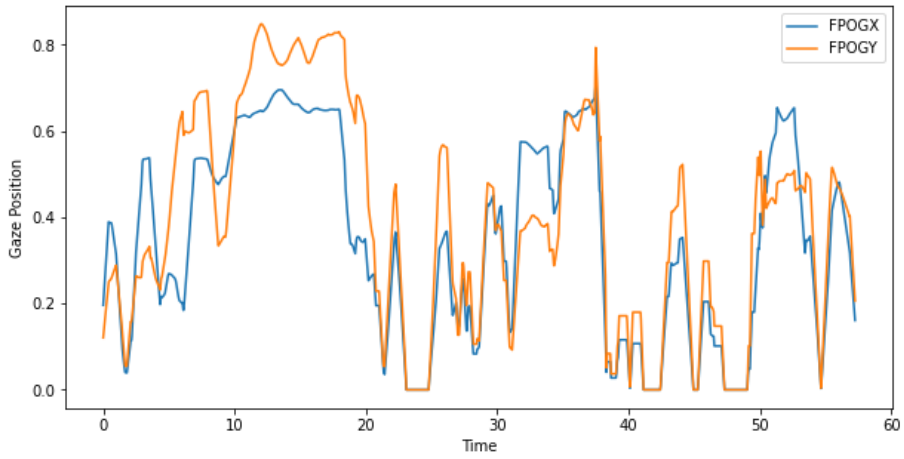
Filtered data

difference kernel size(3 and 50)

[5]:
```
# Define the convolution kernel
kernel = np.ones((3,))/3

# Convolve the X and Y data with the kernel
X_conv = np.convolve(X, kernel, mode='same')
Y_conv = np.convolve(Y, kernel, mode='same')

# Apply thresholding
X_conv[X_conv > 1] = 1
Y_conv[Y_conv > 1] = 1
fig, ax = plt.subplots(figsize=(10,5))
ax.plot(times, X_conv, label='FPOGX')
ax.plot(times, Y_conv, label='FPOGY')
ax.legend()
ax.set_xlabel('Time')
ax.set_ylabel('Gaze Position')
plt.savefig(FileName+'Gazedata_kernel'+str(len(kernel))+'.pdf',bbox_inches = 'tight', format='p
plt.show()
```



[6]:
```
# Define the convolution kernel
kernel = np.ones((50,))/50

# Convolve the X and Y data with the kernel
X_conv1 = np.convolve(X, kernel, mode='same')
Y_conv1 = np.convolve(Y, kernel, mode='same')

# Apply thresholding
X_conv1[X_conv1 > 1] = 1
Y_conv1[Y_conv1 > 1] = 1
fig, ax = plt.subplots(figsize=(10,5))
ax.plot(times, X_conv1, label='FPOGX')
ax.plot(times, Y_conv1, label='FPOGY')
ax.legend()
ax.set_xlabel('Time')
ax.set_ylabel('Gaze Position')
plt.savefig(FileName+'Gazedata_kernel'+str(len(kernel))+'.pdf',bbox_inches = 'tight', format='pdf')
plt.show()
```



Load the txt files into data frame

Load Frame Data

[7]:
```
#read files from folder, the files are named as 'Name0001.txt' to 'Name0100.txt'
#the files are in the same folder as this script
#the files are in the VOC format

import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def txt_to_dataframe(path):
    txt_list = []
    file_list = []
    for txt_file in glob.glob(path + '/*.txt'):
        # print(txt_file)
        with open(txt_file, 'r') as f:
            # txt_list.append({txt_file:f.read()})
            txt_list.append(f.read())
            file_list.append(txt_file)
     # for loop to add the files into a dataframe and split the columns into multiple columns
    txt_df = pd.DataFrame(txt_list)
    all_data = pd.DataFrame()
    i = 0
    for txt in txt_list[:]:
        # print(txt)
        txt = txt.split('\n')
        txt = [x for x in txt if x]
        txt = [x.split(' ') for x in txt]
        df = pd.DataFrame(txt,columns = ['class','x','y','w','h'])
        # print(df)
        df['frame'] = file_list[i].split('/')[-1].split('_')[1].split('.')[0]
        # string to int
        df['frame'] = df['frame'].astype(int)
        all_data = all_data.append(df)
        i = i+1
    all_data = all_data.sort_values(by=['frame'])
    return txt_df,all_data
```

[8]:
```
videoName = '[redacted]'
txtfiles,alldata = txt_to_dataframe(basePath+'/[redacted]/yolov5/runs/detect/exp4/labels/')
# txtfiles.head(),
frames = int(alldata['frame'].iloc[-1])
print(frames,'frames')
alldata.tail()
# txtfiles.sort(key=lambda x: int(x.split('/')[-1].split('_')[1].split('.')[0]))
```

```
1711 frames
```

[8]:

	class	x	y	w	h	frame
0	0	0.55365	0.443773	0.695796	0.727695	1707
0	0	0.55365	0.443773	0.695796	0.727695	1708
0	0	0.55365	0.443773	0.695796	0.727695	1709
0	0	0.55365	0.443773	0.695796	0.727695	1710
0	0	0.55365	0.443773	0.695796	0.727695	1711

Match the object in txt and gaze data

Measure the Object

[9]:
```
fps = frames/times.iloc[-1]#30
classes = pd.DataFrame()
i = 0
# for i in np.arange(len(X_conv)):
for time in times[:-1]:
    # time = times[i]
    frame = int(time * fps)+1
    # frame = 1495
    # print(frame)
    lists = alldata[alldata['frame'] == frame]
    for num in range(len(lists)):
        item1 = lists.iloc[num]
        x,y,w,h = float(item1['x']),float(item1['y']),float(item1['w']),float(item1['h'])
        x0 = x-0.5*w -0.05
        x1 = x+0.5*w +0.05
        y0 = y-0.5*h -0.05
        y1 = y+0.5*h +0.05
        # print(X[i],Y[i])
        # print(x0,x1,y0,y1)
        if X_conv[i]>=x0 and X_conv[i]<=x1 and Y_conv[i]>=y0 and Y_conv[i]<=y1:
            cla = pd.DataFrame({'frame':[frame],'class':[item1['class']]})
        else:
            cla = pd.DataFrame({'frame':[frame],'class':[4]})
        classes = pd.concat([classes, cla], axis=0)
    i = i+1
print(classes)
```

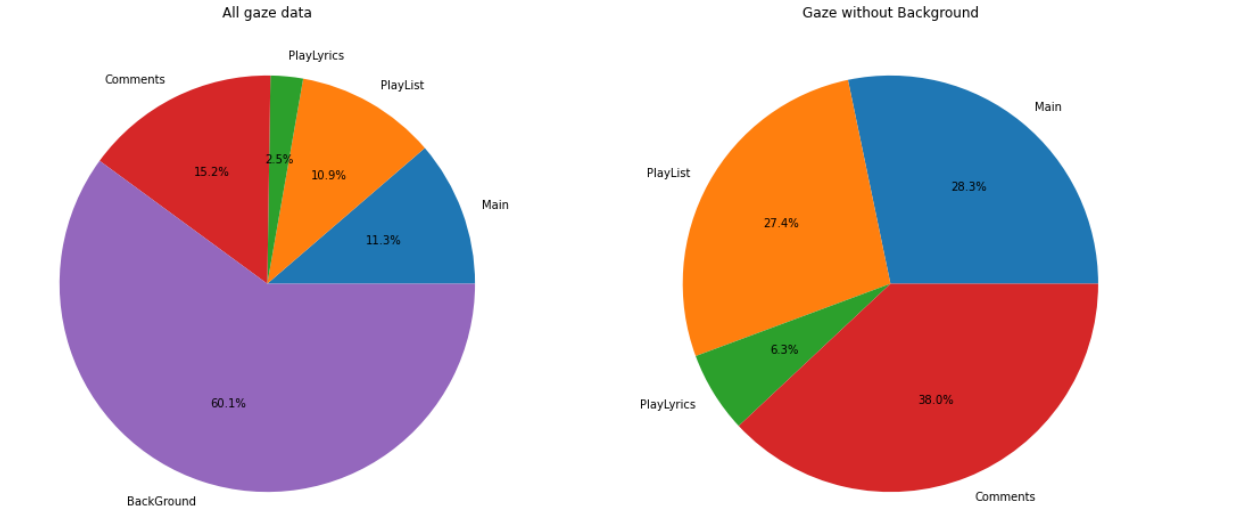
```
    frame class
0       1     1
0       1     1
0       1     1
0       2     1
0       2     1
..    ...   ...
0    1709     0
0    1710     0
0    1710     0
0    1711     0
0    1711     0

[3567 rows x 2 columns]
```

[10]:
```
values = classes['class'].astype(int).to_numpy()
print(values)
```

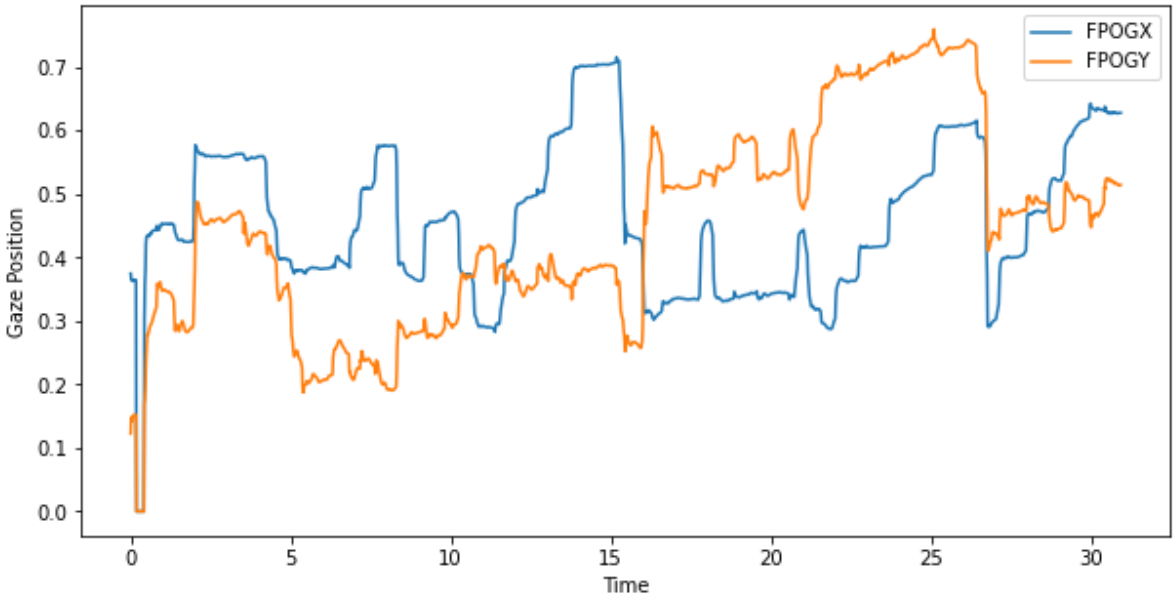
```
[1 1 1 ... 0 0 0]
```

Result analysis

```
names = ['Main','PlayList','PlayLyrics','Comments','BackGround']
def show_pie(_classes,saveName):
    values = _classes['class'].astype(int).to_numpy()
    unique_elements, counts = np.unique(values, return_counts=True)
    uniq_names = [name for index, name in enumerate(names) if index in unique_elements]

    fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(18, 9))

    # First pie chart
    axs[0].pie(counts, labels=uniq_names, autopct='%1.1f%%')
    axs[0].set_title('All gaze data')

    # Second pie chart
    axs[1].pie(counts[:-1], labels=uniq_names[:-1], autopct='%1.1f%%')
    axs[1].set_title('Gaze without Background')
    plt.savefig(saveName+'Gaze.pdf',bbox_inches = 'tight', format='pdf')
    plt.show()
show_pie(classes,FileName)
```



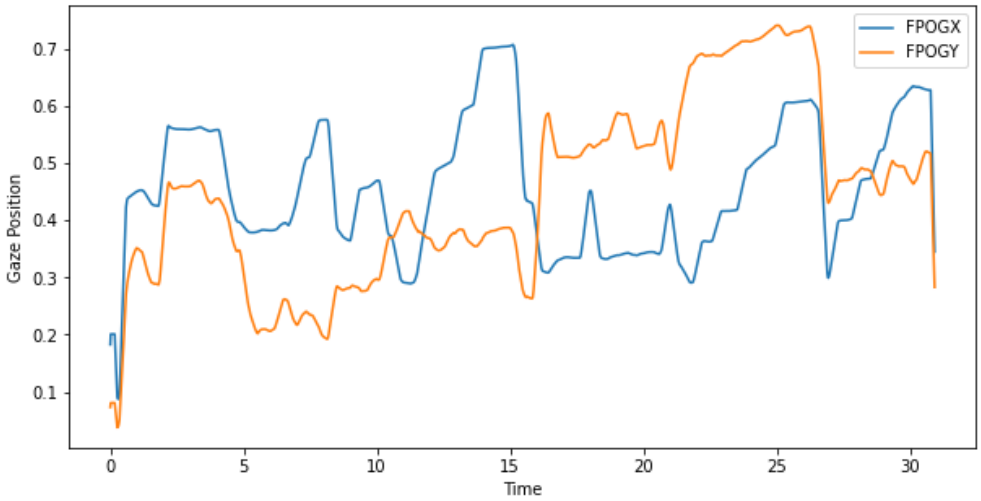
Dataset2 and smoothed Dataset2




```
549
483 frames None
15.613599930433816 fps
```

]:

	class	x	y	w	h	frame
0	0	0.576026	0.583119	0.795709	0.545103	479
0	0	0.576026	0.583119	0.795709	0.545103	480
0	0	0.576026	0.583119	0.795709	0.545103	481
0	0	0.576026	0.583119	0.795709	0.545103	482
0	0	0.576026	0.583119	0.795709	0.545103	483




```
1884 1884
```

Dataset2 result analysis

```
show_pie(classes0,FileName0)
```