Обучение и дообучение модели

Исходные данные:

Датасет: https://www.kaggle.com/datasets/johannesbayer/cghd1152

Модель: YOLOv5

1. Подготовка данных

Проще всего загрузить данные в Google Colab непосредственно с Kaggle. Вот здесь представлена подробная инструкция, как это сделать:

https://www.kaggle.com/general/74235

После загрузки данных и разархивирования нужно переложить картинки и аннотации так, чтобы в дальнейшем их можно было использовать для обучения модели. Код для подготовки был использован отсюда:

https://blog.paperspace.com/train-yolov5-custom-data/#custom-network-architecture

Рекомендуется сначала ознакомиться со статьей, поскольку в ней довольно подробно объясняется каждое действие.

Основной код, используемый далее, приведён ниже:

```
import torch
from IPython.display import Image  # for displaying images
import os
import random
import shutil
from sklearn.model_selection import train_test_split
import xml.etree.ElementTree as ET
from xml.dom import minidom
from tqdm import tqdm
from PIL import Image, ImageDraw
import numpy as np
import matplotlib.pyplot as plt
import re
```

```
!mkdir /content/images /content/annotations
```

```
images = []
```

```
annotations = []
for i in range(1, 24):
    path = f'/content/drafter_{i}/'
    images.extend([os.path.join(path+'images', x) for x in
os.listdir(path+'images')])
    annotations.extend([os.path.join(path+'annotations', x) for x in
os.listdir(path+'annotations')])
```

```
def move_files_to_folder(list_of_files, destination_folder):
    for f in list_of_files:
        try:
            shutil.move(f, destination_folder)
        except:
            print(f)
            assert False

move_files_to_folder(images, '/content/images/')
move_files_to_folder(annotations, '/content/annotations/')
```

```
rm -r /content/cghd1152.zip
```

```
def extract_info_from_xml(xml_file):
    root = ET.parse(xml_file).getroot()

    # Initialise the info dict
    info_dict = {}
    info_dict['bboxes'] = []

    # Parse the XML Tree
    for elem in root:
        # Get the file name
        if elem.tag == "filename":
            info_dict['filename'] = elem.text

        # Get the image size
        elif elem.tag == "size":
            image_size = []
            for subelem in elem:
                image_size.append(int(subelem.text))

            info_dict['image_size'] = tuple(image_size)

        # Get details of the bounding box
        elif elem.tag == "object":
            bbox = {}
            for subelem in elem:
                if subelem.tag == "name":
                    bbox["class"] = subelem.text

                elif subelem.tag == "bndbox":
                    for subsubelem in subelem:
                        bbox[subsubelem.tag] = 
int(subsubelem.text)
            info_dict['bboxes'].append(bbox)

    return info_dict
```

```
cd /content
```

```
class_name_to_id_mapping = {
  "__background__": 0,
  "text": 1,
  "junction": 2,
  "crossover": 3,
  "terminal": 4,
  "gnd": 5,
  "vss": 6,
  "voltage.dc": 7,
  "voltage.ac": 8,
  "voltage.battery": 9,
  "resistor": 10,
  "resistor.adjustable": 11,
  "resistor.photo": 12,
  "capacitor.unpolarized": 13,
  "capacitor.polarized": 14,
  "capacitor.adjustable": 15,
  "inductor": 16,
  "inductor.ferrite": 17,
  "inductor.coupled": 18,
  "transformer": 19,
  "diode": 20,
  "diode.light_emitting": 21,
  "diode.thyrector": 22,
  "diode.zener": 23,
  "diac": 24,
  "triac": 25,
  "thyristor": 26,
  "varistor": 27,
  "transistor.bjt": 28,
  "transistor.fet": 29,
  "transistor.photo": 30,
  "operational_amplifier": 31,
  "operational_amplifier.schmitt_trigger": 32,
  "optocoupler": 33,
  "integrated_circuit": 34,
  "integrated_circuit.ne555": 35,
  "integrated_circuit.voltage_regulator": 36,
  "xor": 37,
  "and": 38,
  "or": 39,
  "not": 40,
  "nand": 41,
  "nor": 42,
  "probe.current": 43,
  "probe.voltage": 44,
  "switch": 45,
  "relay": 46,
  "socket": 47,
  "fuse": 48,
  "speaker": 49,
  "motor": 50,
  "lamp": 51,
  "microphone": 52,
  "antenna": 53,
  "crystal": 54,
  "mechanical": 55,
  "magnetic": 56,
  "optical": 57,
  "block": 58,
  "unknown": 59
}
class_name_to_id_mapping
```

```
# Convert the info dict to the required yolo format and write it to disk
def convert_to_yolov5(info_dict):
    print_buffer = []

    # For each bounding box
    for b in info_dict["bboxes"]:
        try:
            class_id = class_name_to_id_mapping[b["class"]]
        except KeyError:
            print("Invalid Class. Must be one from ",
class_name_to_id_mapping.keys())

        # Transform the bbox co-ordinates as per the format required by
YOLO v5
        b_center_x = (b["xmin"] + b["xmax"]) / 2
        b_center_y = (b["ymin"] + b["ymax"]) / 2
        b_width    = (b["xmax"] - b["xmin"])
        b_height   = (b["ymax"] - b["ymin"])
        # Normalise the co-ordinates by the dimensions of the image
        image_w, image_h, image_c = info_dict["image_size"]
        b_center_x /= image_w
        b_center_y /= image_h
        b_width    /= image_w
        b_height   /= image_h

        #Write the bbox details to the file
        print_buffer.append("{} {:.3f} {:.3f} {:.3f}
{:.3f}".format(class_id, b_center_x, b_center_y, b_width, b_height))

    # Name of the file which we have to save
    filename = info_dict["filename"]
    old_format = image_format.findall(filename)[0]
    filename = filename.replace(old_format, '.txt')

    save_file_name = os.path.join("/content/annotations", filename)
    print(f'info_dict {filename}\n save_file_name {save_file_name}')

    # Save the annotation to disk
    print("\n".join(print_buffer), file= open(save_file_name, "w"))
```

```
# Get the annotations
annotations = [os.path.join('/content/annotations', x) for x in
os.listdir('/content/annotations') if x[-3:] == "xml"]
annotations.sort()
```

```
# Convert and save the annotations
for ann in tqdm(annotations):
    info_dict = extract_info_from_xml(ann)
    convert_to_yolov5(info_dict)
annotations = [os.path.join('/content/annotations', x) for x in
os.listdir('/content/annotations') if x[-3:] == "txt"]
```

Проверить, все ли правильно отображается, можно с помощью следующего кода:

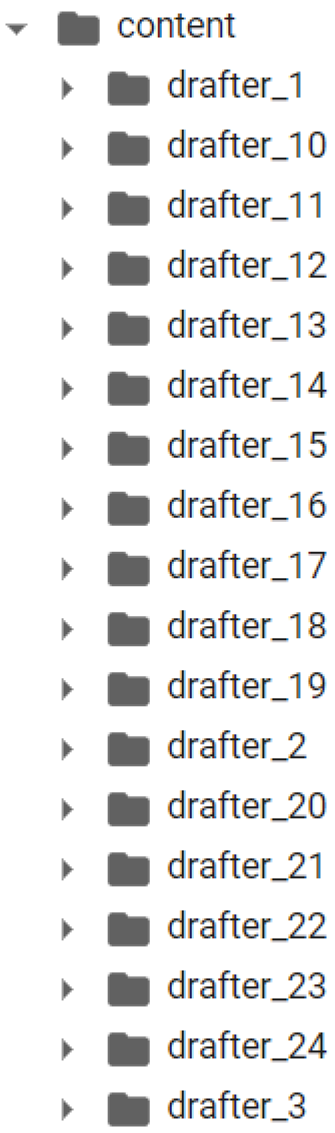
```
class_id_to_name_mapping = dict(zip(class_name_to_id_mapping.values(),
class_name_to_id_mapping.keys()))

def plot_bounding_box(image, annotation_list):
    annotations = np.array(annotation_list)
    w, h = image.size

    plotted_image = ImageDraw.Draw(image)
    transformed_annotations = np.copy(annotations)
    transformed_annotations[:,[1,3]] = annotations[:,[1,3]] * w
    transformed_annotations[:,[2,4]] = annotations[:,[2,4]] * h

    transformed_annotations[:,1] = transformed_annotations[:,1] -
(transformed_annotations[:,3] / 2)
    transformed_annotations[:,2] = transformed_annotations[:,2] -
(transformed_annotations[:,4] / 2)
    transformed_annotations[:,3] = transformed_annotations[:,1] +
transformed_annotations[:,3]
    transformed_annotations[:,4] = transformed_annotations[:,2] +
transformed_annotations[:,4]

    for ann in transformed_annotations:
        obj_cls, x0, y0, x1, y1 = ann
        plotted_image.rectangle(((x0,y0), (x1,y1)), width=8)

        plotted_image.text((x0, y0 - 10),
class_id_to_name_mapping[(int(obj_cls))])

    plt.imshow(np.array(image))
    plt.show()


# Get any random annotation file

random.seed(20)
annotation_file = random.choice(annotations)
annotation_file
```

```
with open(annotation_file, "r") as file:
    annotation_list = file.read().split("\n")[:-1]
    annotation_list = [x.split(" ") for x in annotation_list]
    annotation_list = [[float(y) for y in x ] for x in annotation_list]
annotation_list
```

```
finding = annotation_file[21:].replace('.txt', '')
for im in os.listdir('/content/images'):
    if im.find(finding) != -1:
        image_file = '/content/images/'+im
image_file
```

```
image = Image.open(image_file)
#Plot the Bounding Box
plot_bounding_box(image, annotation_list)
```

На этом этапе папки выглядят примерно так:



content
drafter_1
drafter_10
drafter_11
drafter_12
drafter_13
drafter_14
drafter_15
drafter_16
drafter_17
drafter_18
drafter_19
drafter_2
drafter_20
drafter_21
drafter_22
drafter_23
drafter_24
drafter_3

После этого переносим изображения и аннотации, делим на обучающую, валидационую и тестовую выборки:

```
# Read images and annotations
images = [os.path.join('/content/images', x) for x in
os.listdir('/content/images')]
annotations = [os.path.join('/content/annotations', x) for x in
os.listdir('/content/annotations') if x[-3:] == "txt"]

images.sort()
annotations.sort()

# Split the dataset into train-valid-test splits
train_images, val_images, train_annotations, val_annotations =
train_test_split(images, annotations, test_size = 0.2, random_state = 1)
```

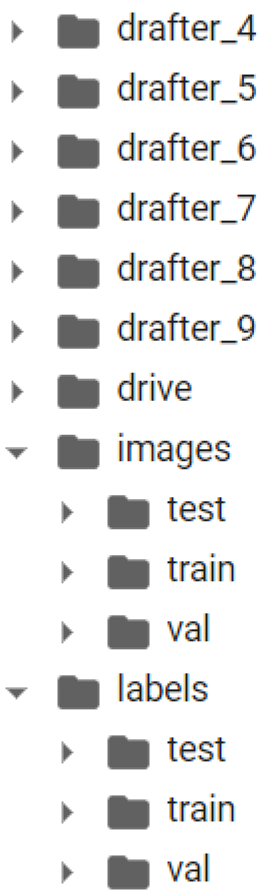
```
val_images, test_images, val_annotations, test_annotations =
train_test_split(val_images, val_annotations, test_size = 0.5,
random_state = 1)
```

```
!mkdir /content/images/train /content/images/val /content/images/test
/content/annotations/train /content/annotations/val
/content/annotations/test
```

```
#Utility function to move images
def move_files_to_folder(list_of_files, destination_folder):
    for f in list_of_files:
        try:
            shutil.move(f, destination_folder)
        except:
            print(f)
            assert False

# Move the splits into their folders
move_files_to_folder(train_images, '/content/images/train')
move_files_to_folder(val_images, '/content/images/val/')
move_files_to_folder(test_images, '/content/images/test/')
move_files_to_folder(train_annotations, '/content/annotations/train/')
move_files_to_folder(val_annotations, '/content/annotations/val/')
move_files_to_folder(test_annotations, '/content/annotations/test/')
```

```
mv /content/annotations /content/labels
```

```
cd /content/
```



```
drafter_4
drafter_5
drafter_6
drafter_7
drafter_8
drafter_9
drive
images
    test
    train
    val
labels
    test
    train
    val
```

2. Загрузка Yolov5

В той же статье объясняется, как работать с моделью.

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install
```

```
pwd
```

В текстовом редакторе создаем новый файл, вставляем следующий текст:

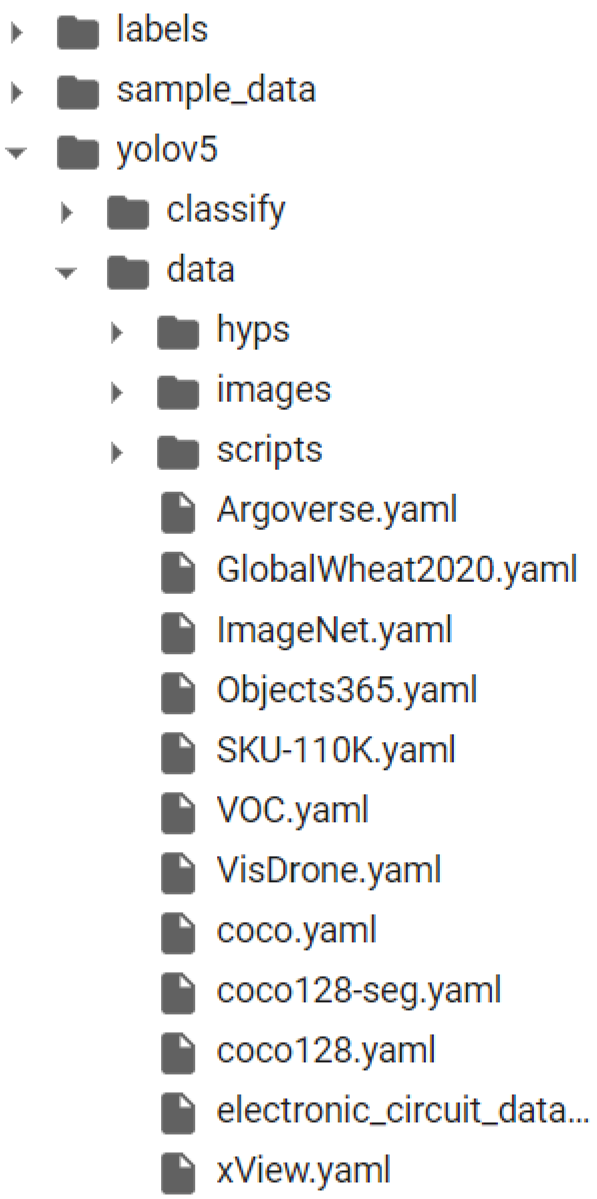
```
path: ../content/
train: ../images/train/
val: ../images/val/
test: ../images/test/
# number of classes
nc: 60
# class names
```

```
names: ["__background__", "text", "junction", "crossover", "terminal", "gnd", "vss",
"voltage.dc", "voltage.ac", "voltage.battery", "resistor", "resistor.adjustable",
"resistor.photo", "capacitor.unpolarized", "capacitor.polarized", "capacitor.adjustable",
"inductor", "inductor.ferrite", "inductor.coupled", "transformer", "diode",
"diode.light_emitting", "diode.thyrector", "diode.zener", "diac", "triac", "thyristor", "varistor",
"transistor.bjt", "transistor.fet", "transistor.photo", "operational_amplifier",
"operational_amplifier.schmitt_trigger", "optocoupler", "integrated_circuit",
"integrated_circuit.ne555", "integrated_circuit.voltage_regulator", "xor", "and", "or", "not",
"nand", "nor", "probe.current", "probe.voltage", "switch", "relay", "socket", "fuse", "speaker",
"motor", "lamp", "microphone", "antenna", "crystal", "mechanical", "magnetic", "optical",
"block", "unknown"]
```

Этот файл сохраняем под именем `electronic_circuit_data.yaml`

Заносим файл в папку `yolov5/data` . Это можно сделать вручную или с помощью следующего кода:

```
%cp /content/drive/My\ Drive/Colab\ Notebooks/electronic_circuit_data.yaml
/content/yolov5/data
```



```
▸ labels
▸ sample_data
▾ yolov5
    ▸ classify
    ▾ data
        ▸ hyps
        ▸ images
        ▸ scripts
          Argoverse.yaml
          GlobalWheat2020.yaml
          ImageNet.yaml
          Objects365.yaml
          SKU-110K.yaml
          VOC.yaml
          VisDrone.yaml
          coco.yaml
          coco128-seg.yaml
          coco128.yaml
          electronic_circuit_data...
          xView.yaml
```

Обучение модели запускается следующим кодом:

```
!python train.py --img 640 --cfg yolov5s.yaml --hyp hyp.scratch-low.yaml
--batch 32 --epochs 5 --data electronic_circuit_data.yaml --weights
yolov5s.pt --workers 24 --name yolo_electronic_circuit_det
```

Стоит отметить, что модель лучше обучать при подключенном GPU. Мне этого хватало на обучение на 10-15 эпох в день. Лучше обучить модель на 5-8 эпохах, а затем дообучать по 5 эпох за раз. В противном случае GPU при перерасходе отключится и удалит все файлы.

После каждого дообучения скачивайте файл с весами на компьютер. Например, так:

```
%cp /content/yolov5/runs/train/yolo_electronic_circuit_det/weights/best.pt /content/drive/My\ Drive/Colab\ Notebooks/
```

При дообучении модели достаточно указать, какие веса использовать при обучении:

```
!python train.py --img 640 --cfg yolov5s.yaml --hyp hyp.scratch-low.yaml --batch 32 --epochs 5 --data electronic_circuit_data.yaml --weights /content/drive/MyDrive/Colab\ Notebooks/best.pt --workers 24 --name yolo_electronic_circuit_det
```

Замечание: если запускать обучение несколько раз в день и не менять название папки, куда сохранять результат, то результаты будут сохраняться в папку с названием «исходное_название2», «исходное_название3» и так далее. Не забывайте об этом при загрузке весов.