# Обучение и дообучение модели

Исходные данные:

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

Модель: YOLOv5

## 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

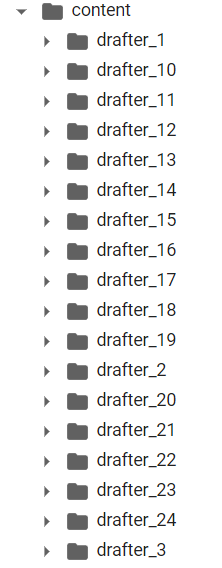
image\_file

image = Image.open(image\_file)

#Plot the Bounding Box

plot\_bounding\_box(image, annotation\_list)

На этом этапе папки выглядят примерно так:



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/

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

## 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](https://colab.research.google.com/drive/1V40PSZx9JfBnDlPDGBkxBXZINWlbchdM?authuser=1)/

val: ..[/images/val](https://colab.research.google.com/drive/1V40PSZx9JfBnDlPDGBkxBXZINWlbchdM?authuser=1)/

test: ..[/images/test](https://colab.research.google.com/drive/1V40PSZx9JfBnDlPDGBkxBXZINWlbchdM?authuser=1)/

# number of classes

nc: 60

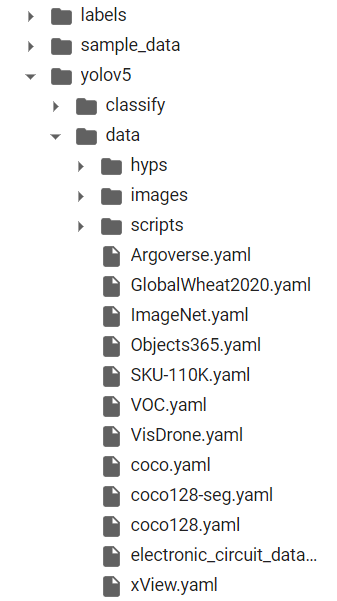
# class names

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



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.