北京大学
PEKING UNIVERSITY

实验5.2：baidu screw & face mask detection

房宇轩

目 录

CONTENTS

任务简介

北京大学 PEKING UNIVERSITY

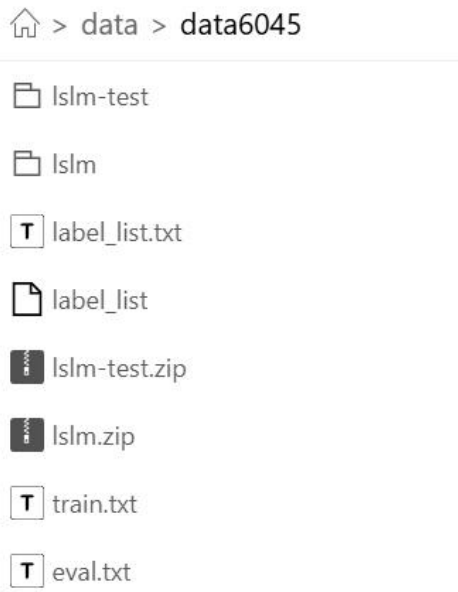
- 训练一个螺丝螺母/是否佩戴口罩目标检测的网络，并在到K210上联合LCD和摄像头运行
  - 目标检测网络：tiny-yolo
  - 训练平台：百度AI Studio平台、自己电脑搭建的训练环境

- 任务构成
  - 数据下载与处理
  - 配置tiny yolo模型的参数
  - 训练目标检测网络tiny-yolo
  - 测试所训练的网络
  - 模型转换
  - 编译和烧录程序

任务简介-数据集介绍

北京大学 PEKING UNIVERSITY

数据集（baidu screw）



label_list.txt 每一行一个类别，类别编号、类别名字

0 bolt（螺丝）　1 nut（螺母）

train.txt 每张图片图片名后多行，每一行一个样本（螺丝或螺母）
{"value":"bolt","coordinate":[[769.459,241.819],[947.546,506.167]]}

lstm/trainImageSet/xxx.jpg 训练图片

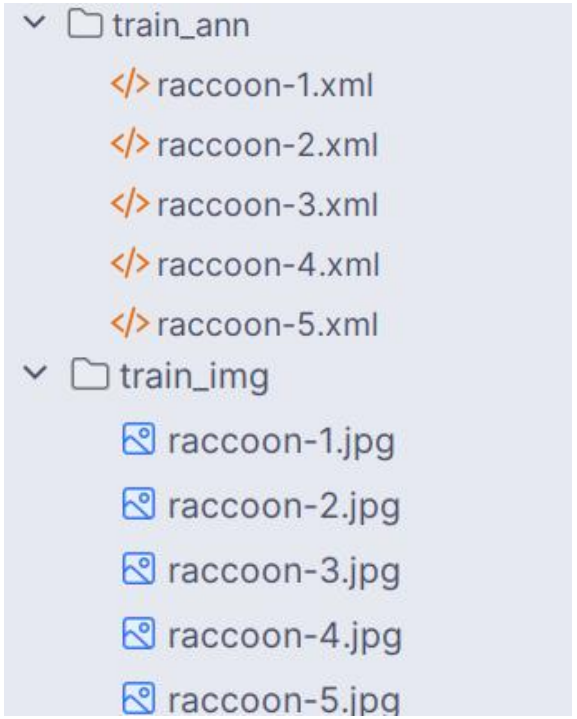
eval.txt 用于测试集，格式同 train.txt

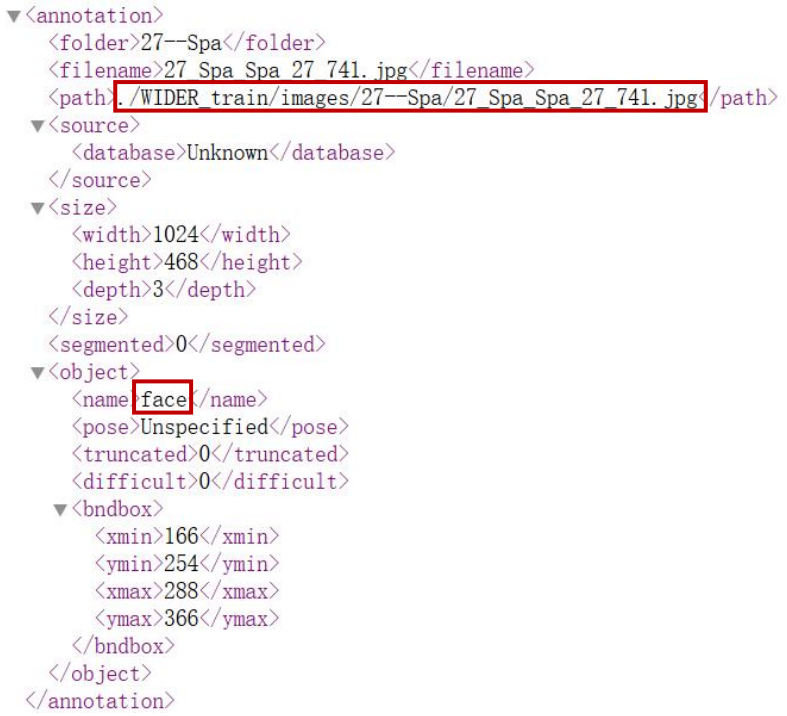
lstm-test/xxx.jpg 验证图片

任务简介-数据集介绍

北京大学 PEKING UNIVERSITY

- 数据集（face mask detection）





train_ann： 标注信息

train_img：训练图片，含人脸戴口罩和不带口罩

xml标注信息：

path：所标注图片的路径

size： 所标注图片的大小

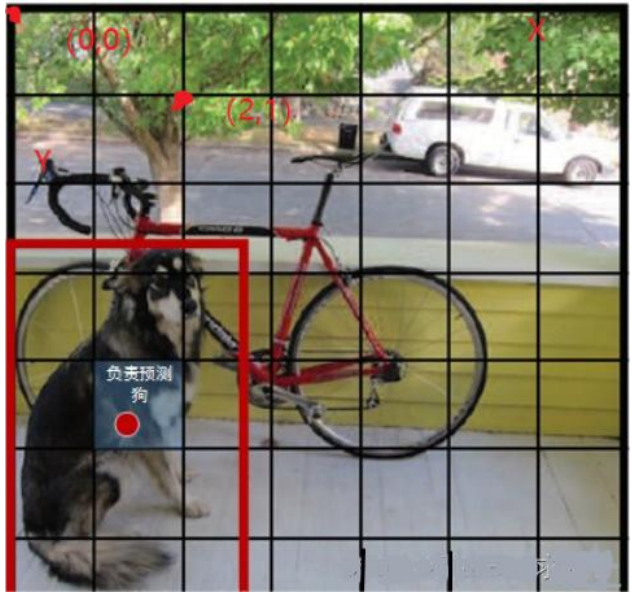
name：所标注检测框的类别

bndbox：bounding box，检测框左上、右下角坐标

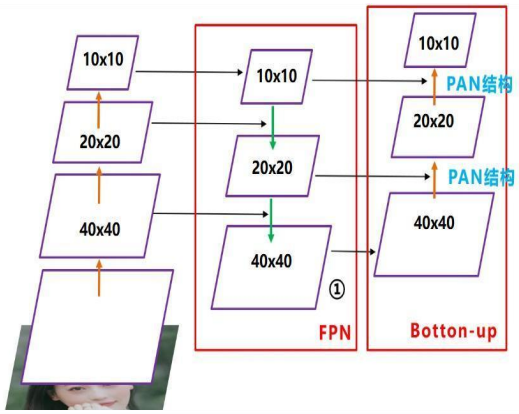
任务简介-YOLO-tiny

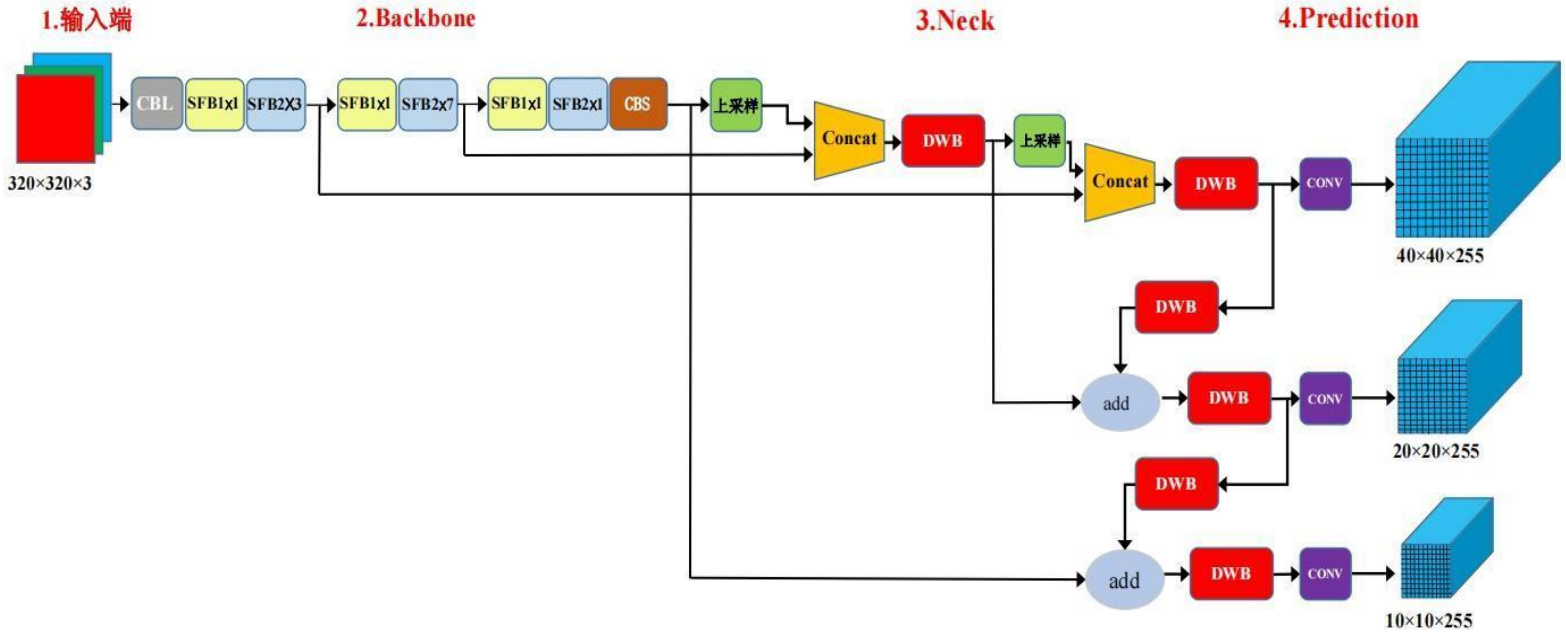
北京大学 PEKING UNIVERSITY

- YOLO 的核心思想就是利用整张图作为网络的输入，直接在输出层回归 bounding box（边界框） 的位置及其所属的类别。
- 将一幅图像分成 SxS 个网格（grid cell），如果某个 object 的中心落在这个网格中，则这个网格就负责预测这个 object。
- 每个 bounding box 要预测 (x, y, w, h) 和 confidence 共5个值，每个网格还要预测一个类别信息，记为 C 类。则 SxS个 网格，每个网格要预测 B 个 bounding box， 每个box中都有 C 个 classes对应的概率值。
- 预测出的S x S x B个目标窗口根据confidence阈值去除可能性较低目标窗口，并通过NMS去除冗余窗口
- 输出就是 S x S x （B*5+C) 的一个 tensor。5：一个bbox需预测5个参数



任务简介-YOLO-tiny





- ShuffleNet Backbone：轻量化特征提取
- FPN+PAN Neck：为了更好的提取融合特征。FPN层自顶向下传达强语义特征（High-Level特征），而特征金字塔则自底向上传达强定位特征（Low-Level特征），两两联手，从不同的主干层对不同的检测层进行特征聚合。
- prediction head：用于预测的3个尺度下特征图，采用GIoU_Loss做回归bbox参数的损失函数

实验例程-baidu screw

- YOLOv3目标检测网络参数配置

  可能需改参数：

  - data_dir\use_tiny\num_epochs\use_gpu\ignore_thresh\mode

- 定义两个类，分别代表 Yolo-v3 和 Yolo-v3-tiny 两个模型。跟随其后的是模型选择函数，根据配置使用不同的模型
- 初始化参数，初始化日志的便利函数
- 图像增强处理的系列函数


```
"data_dir": "data/data6045/",
"file_list": "train.txt",
"class_dim": -1,
"label_dict": {},
"image_count": -1,
"continue_train": False,# 是否加载前一次的训练参数，接着训练
"pretrained": False,
"pretrained_model_dir": "./pretrained-model",
"save_model_dir": "./yolo-model",
"model_prefix": "yolo-v3",
"use_tiny": True,          # 是否使用 裁剪 tiny 模型
"max_box_num": 20,          # 一幅图上最多有多少个目标
"num_epochs": 120,
"train_batch_size": 5,      # 对于完整 yolov3，每一批的训练样本不能太多，内存会炸掉
"use_gpu": False,
"yolo_cfg": {
    #"input_size": [3, 608, 608],
    #"anchors": [10,13,16,30,33,23,30,61,62,45,59,119,116,90, 156,198, 373, 326],
    "input_size": [3, 416, 416],
    "anchors": [7,9,11,21,23,16,21,42,43,38,41,82,80,62,107,136,256,224],
    "anchor_mask": [[6, 7, 8], [3, 4, 5], [0, 1, 2]] },
"yolo_tiny_cfg": {
    "input_size": [3, 256, 256],
    "anchors": [6, 8, 13, 15, 22, 34, 48, 50, 81, 100, 205, 191],
    "anchor_mask": [[3, 4, 5], [0, 1, 2]] },
"ignore_thresh": 0.7,
"mean_rgb": [127.5, 127.5, 127.5],
"mode": "train",
"multi_data_reader_count": 0,
"apply_distort": True,
"valid_thresh": 0.01,
"nms_thresh": 0.45,
"image_distort_strategy": {
    "expand_prob": 0.5,
    "expand_max_ratio": 4,
    "hue_prob": 0.5,
    "hue_delta": 18,
    "contrast_prob": 0.5,
    "contrast_delta": 0.5,
    "saturation_prob": 0.5,
    "saturation_delta": 0.5,
    "brightness_prob": 0.5,
    "brightness_delta": 0.125 },
"rsm_strategy": {
    "learning_rate": 0.001,
    "lr_epochs": [20, 40, 60, 80, 100],
    "lr_decay": [1, 0.5, 0.25, 0.1, 0.05, 0.01], },
"momentum_strategy": {
    "learning_rate": 0.1,
    "decay_steps": 2 ** 7,
    "decay_rate": 0.8 },
"early_stop": {
    "sample_frequency": 50,
    "successive_limit": 3,
    "min_loss": 2.5,
    "min_curr_map": 0.84
```

实验例程-baidu screw

- 自定义数据读取器，如果需要自定义数据，需要修改这段函数，以适配自定义数据的格式

- 定义数据读取、定义优化器
- 构建 program 和损失函数
- 训练
- 将模型固化:使参数不再更新，加速推理
- 使用固化的模型进行预测（模型测试）
- 模型压缩、量化
- 模型转换

```
def custom_reader(file_list, data_dir, input_size, mode):
    def reader():
        np.random.shuffle(file_list)
        for line in file_list:
            if mode == 'train' or mode == 'eval':
                ######################  以下可能是需要自定义修改的部分    ############################
                parts = line.split()
                image_path = parts[0]
                img = Image.open(image_path)
                if img.mode != 'RGB':
                    img = img.convert('RGB')
                im_width, im_height = img.size
                # bbox 的列表，每一个元素为这样
                # layout: label | x-center | y-cneter | width | height | difficult
                bbox_labels = []
                for object_str in parts[1:]:
                    bbox_sample = []
                    object = json.loads(object_str)
                    bbox_sample.append(float(train_parameters['label_dict'][object['value']]))
                    bbox = object['coordinate']
                    box = [bbox[0][0], bbox[0][1], bbox[1][0] - bbox[0][0], bbox[1][1] - bbox[0][1]]
                    bbox = box_to_center_relative(box, im_height, im_width)
                    bbox_sample.append(float(bbox[0]))
                    bbox_sample.append(float(bbox[1]))
                    bbox_sample.append(float(bbox[2]))
                    bbox_sample.append(float(bbox[3]))
                    difficult = float(0)
                    bbox_sample.append(difficult)
                    bbox_labels.append(bbox_sample)
                ######################  可能需要自定义修改部分结束    ############################
                if len(bbox_labels) == 0: continue
                img, sample_labels = preprocess(img, bbox_labels, input_size, mode)
                # sample_labels = np.array(sample_labels)
                if len(sample_labels) == 0: continue
                boxes = sample_labels[:, 1:5]
                lbls = sample_labels[:, 0].astype('int32')
                difficults = sample_labels[:, -1].astype('int32')
                max_box_num = train_parameters['max_box_num']
                cope_size = max_box_num if len(boxes) >= max_box_num else len(boxes)
                ret_boxes = np.zeros((max_box_num, 4), dtype=np.float32)
                ret_lbls = np.zeros((max_box_num), dtype=np.int32)
                ret_difficults = np.zeros((max_box_num), dtype=np.int32)
                ret_boxes[0: cope_size] = boxes[0: cope_size]
                ret_lbls[0: cope_size] = lbls[0: cope_size]
                ret_difficults[0: cope_size] = difficults[0: cope_size]
                yield img, ret_boxes, ret_lbls, ret_difficults
            elif mode == 'test':
                img_path = os.path.join(line)

                yield Image.open(img_path)
    return reader
```

实验例程-face mask detection

- 模型参数配置
- 配置文件主要定义数据相关参数、模型训练参数、路径等，在需根据模型和数据情况进行调整

```
13    argparser = argparse.ArgumentParser(
14        description='Train and validate YOLO_v2 model on any dataset')
15
16    argparser.add_argument(
17        *name_or_flags: '-c',
18        '--conf',
19        default="configs/from_scratch.json",
20        help='path to configuration file')
```

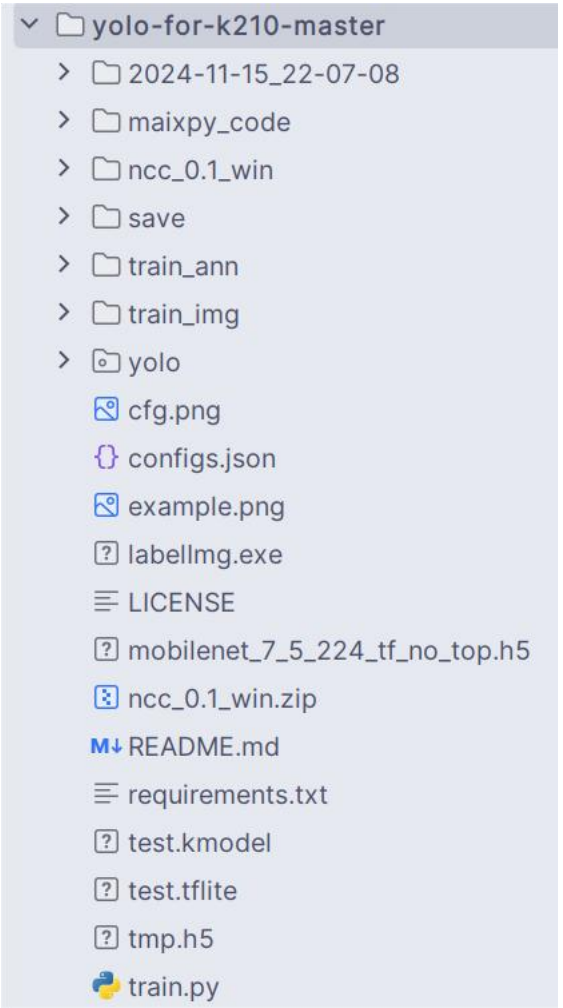
configs.json

```
"model" : {
    "architecture":         "MobileNet",
    "input_size":           224,
    "anchors":              [1, 1.2, 2, 3, 4, 3, 6, 4,
5, 6.5],
    "labels":               ["raccoon"],
    "coord_scale" :         1.0,
    "class_scale" :         1.0,
    "object_scale" :    5.0,
    "no_object_scale" :     1.0
},
"pretrained" : {
    "full":   ""
},
"train" : {
    "actual_epoch":         50,
    "train_image_folder":   "train_img",
    "train_annot_folder":   "train_ann",
    "train_times":          2,
    "valid_image_folder":   "",
    "valid_annot_folder":   "",
    "valid_times":          1,
    "batch_size":           16,
    "learning_rate":        1e-4,
    "saved_folder":        "save",
    "first_trainable_layer": "",
    "jitter":          true,
    "is_only_detect" :     false
}
```

实验例程-face mask detection

模型代码的整体目录：

├── configs.json：超参数的配置文件（这些文件（yaml文件）是用来配置训练集和测试集还有验证集的路径的，其中还包括目标检测的种类数和种类的名称）；
├── yolo：里面主要是一些网络构建的函数。
├── ncc：存放模型转换工具。
├── 以时间命名的文件：放置训练好的权重参数。
├── detect.py：利用训练好的权重参数进行目标检测，可以进行图像、视频和摄像头的检测。
├── train.py：训练自己的数据集的函数。
├── train_ann：训练集xml标注文件
├── train_img：图像文件
├──requirements.txt：这是一个文本文件，里面写着使用yolov项目的环境依赖包的一些版本，可以利用该文本导入相应版本的包。



实验例程-face mask detection

- 数据转换：txt->xml

  从TXT（YOLO格式）转换为XML：
  解析TXT文件： 使用文件操作读取TXT文件的每一行，每行代表一个目标。
  类别映射逆转： 将TXT中的类别索引逆转回目标类别名称，以便在XML中使用。
  坐标逆转换： 将TXT中的YOLO格式的边界框坐标进行逆转换，将相对图像尺寸的坐标转换回绝对坐标。
  生成XML： 使用XML创建库（如xml.etree.ElementTree）创建XML树。为每个目标创建XML元素和子元素，包含类别、边界框坐标等信息。
  保存XML： 将创建的XML树保存为XML文件。

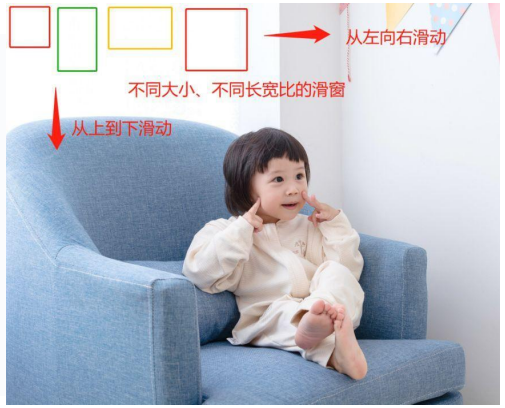
- 记录图片路径的文件路径可能需要修改，images，labels，xmllabels必须在一个文件目录下，images存放图片，labels存放txt文件

```
import cv2
import os

labels = ['mask', 'no_mask']  # 数据集类别名
xml_head = '''<annotation>
    <folder>images</folder>
    <!--文件名-->
    <filename>{}</filename>.
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>{}</width>
        <height>{}</height>
        <depth>{}</depth>
    </size>
    <segmented>0</segmented>
    '''
xml_obj = '''
    <object>
        <name>{}</name>
        <pose>Unspecified</pose>
        <!--是否被裁减，0表示完整，1表示不完整-->
        <truncated>0</truncated>
        <!--是否容易识别，0表示容易，1表示困难-->
        <difficult>0</difficult>
        <!--bounding box的四个坐标-->
        <bndbox>
            <xmin>{}</xmin>
            <ymin>{}</ymin>
            <xmax>{}</xmax>
            <ymax>{}</ymax>
        </bndbox>
    </object>
    '''
xml_end = '''
</annotation>'''
cnt = 0
```

```
with open('train.txt', 'r') as train_list:#其中包含图片路径
    for lst in train_list.readlines():
        lst = lst.strip()
        jpg = lst  # image path
        txt1 = lst.replace('images', 'labels')
        txt = lst.replace('.jpg', '.txt')  # yolo label
txt path
        xml_path1 = jpg.replace('images', 'xmllabels')
        xml_path = jpg.replace('.jpg', '.xml')
        # xml保存路径，此时images，labels，xmllabels必须在一
个文件目录下，images存放图片，labels存放txt文件。

        obj = ''

        img = cv2.imread(jpg)
        img_h, img_w = img.shape[0], img.shape[1]
        head = xml_head.format(str(jpg), str(img_w),
str(img_h))
        with open(txt, 'r') as f:
            for line in f.readlines():
                yolo_datas = line.strip().split(' ')
                label = int(float(yolo_datas[0].strip()))
                center_x =
round(float(str(yolo_datas[1]).strip()) * img_w)
                center_y =
round(float(str(yolo_datas[2]).strip()) * img_h)
                bbox_width =
round(float(str(yolo_datas[3]).strip()) * img_w)
                bbox_height =
round(float(str(yolo_datas[4]).strip()) * img_h)

                xmin = str(int(center_x - bbox_width / 2))
                ymin = str(int(center_y - bbox_height / 2))
                xmax = str(int(center_x + bbox_width / 2))
                ymax = str(int(center_y + bbox_height / 2))

                obj += xml_obj.format(labels[label], xmin,
ymin, xmax, ymax)
        with open(xml_path, 'w') as f_xml:
            f_xml.write(head + obj + xml_end)
        cnt += 1
        print(cnt)
```

实验例程-face mask detection

```
58  #if LOAD_KMODEL_FROM_FLASH
59  uint8_t model_data[KMODEL_SIZE];
60  #else
61  #if ANCHOR_NUM == 5
62  INCBIN(model, "detect_5.kmodel");
63  #endif
64  #if ANCHOR_NUM == 9
65  INCBIN(model, "detect_9.kmodel");
66  #endif
67  #endif
```



- 是否从Flash读取kmodel，默认为否，如果是则创建model_data进行存储
- 识别时的anchor数，默认为5

```
72  void camera_switch(void)
73  {
74      g_camera_no = !g_camera_no;
75      g_camera_no ? open_gc0328_0() : open_gc0328_1();
76
77      int enable = g_camera_no ? 1 : 0;
78      pwm_set_enable(1, 1, enable);
79  }
```

功能：切换摄像头并控制pwm

具体实现：g_camera_no取反；如果g_camera_no为1打开摄像头0，否则打开摄像头1；如果g_camera_no为1使能pwm，否则禁用；控制pwm 1的1通道

```
87  static int dvp_irq(void *ctx)
88  {
89      if (dvp_get_interrupt(DVP_STS_FRAME_FINISH))
90      {
91          dvp_config_interrupt(DVP_CFG_START_INT_ENABLE | DVP_CFG_FINISH_INT_ENABLE, 0);
92          dvp_clear_interrupt(DVP_STS_FRAME_FINISH);
93          g_dvp_finish_flag = 1;
94      }
95      else
96      {
97          dvp_start_convert();
98          dvp_clear_interrupt(DVP_STS_FRAME_START);
99      }
100     return 0;
101 }
```

功能：dvp摄像头接口模块中断程序，dvp支持把摄像头输入图像数据转发给 AI 模块或者内存

具体实现：是否接受到图像结束采集中断，如果是则在开始采集到结束都禁用中断；当采集结束清除结束中断；设置结束标志为1；如果为否则开始采集图像；清除开始中断；

实验例程-face mask detection

```
103  static void io_init(void)
104  {
105      /* Init DVP IO map and function settings */
109      fpioa_set_function(DVP_PWDN_PIN, FUNC_CMOS_PWDN);
110      fpioa_set_function(DVP_XCLK_PIN, FUNC_CMOS_XCLK);
111      fpioa_set_function(DVP_VSYNC_PIN, FUNC_CMOS_VSYNC);
112      fpioa_set_function(DVP_HREF_PIN, FUNC_CMOS_HREF);
113      fpioa_set_function(DVP_PCLK_PIN, FUNC_CMOS_PCLK);
114      fpioa_set_function(DVP_SCCB_SCLK_PIN, FUNC_SCCB_SCLK);
115      fpioa_set_function(DVP_SCCB_SDA_PIN, FUNC_SCCB_SDA);
116
117      /* Init SPI IO map and function settings */
118      fpioa_set_function(LCD_DC_PIN, FUNC_GPIOHS0 + LCD_DC_IO);
119      fpioa_set_function(LCD_CS_PIN, FUNC_SPI0_SS3);
120      fpioa_set_function(LCD_RW_PIN, FUNC_SPI0_SCLK);
121      fpioa_set_function(LCD_RST_PIN, FUNC_GPIOHS0 + LCD_RST_IO);
122
123      sysctl_set_spi0_dvp_data(1);
124
125      /* LCD Backlight */
126      fpioa_set_function(LCD_BLIGHT_PIN, FUNC_GPIOHS0 + LCD_BLIGHT_IO);
127      gpiohs_set_drive_mode(LCD_BLIGHT_IO, GPIO_DM_OUTPUT);
128      gpiohs_set_pin(LCD_BLIGHT_IO, GPIO_PV_LOW);
129
130  #if (BOARD_VERSION == BOARD_V1_3)
131      /* KEY IO map and function settings */
132      fpioa_set_function(KEY_PIN, FUNC_GPIOHS0 + KEY_IO);
133      gpiohs_set_drive_mode(KEY_IO, GPIO_DM_INPUT_PULL_UP);
134      gpiohs_set_pin_edge(KEY_IO, GPIO_PE_FALLING);
135      gpiohs_irq_register(KEY_IO, 1, key_trigger, NULL);
136
137       /* pwm IO.*/
138      fpioa_set_function(LED_IR_PIN, FUNC_TIMER0_TOGGLE2 + 1 * 4);
139       /* Init PWM */
140      pwm_init(1);
141      pwm_set_frequency(1, 1, 3000, 0.3);
142      pwm_set_enable(1, 1, 0);
143  #endif
144  }
```

功能：初始化IO引脚

具体实现：

- 初始化DVP IO引脚：43：关闭dvp；46：dvp系统时钟；42：帧同步线；44：行同步线；47：像素时钟；41：串行相机时钟；40：串行相机数据
- 初始化SPI IO引脚：37：GPIO速度配置2；38：片选，SPI0的3；36：SPI串行时钟；39（复位）：复位速度；使能dvp传数据
- 初始化LCD背景亮度：设置亮度为17；设置为输出模式并置低
- 初始化按键：速率设为8；设为输入上拉；下降沿触发；当按时出发中断
- 初始化PWM：pwm频率3000占空比0.3；使能

实验例程-face mask detection

```
153  #if (CLASS_NUMBER > 1)
154  typedef struct
155  {
156      char *str;
157      uint16_t color;
158      uint16_t height;
159      uint16_t width;
160      uint32_t *ptr;
161  } class_lable_t;
```

- 如果识别类别数多于1
- 建立目标标签结构体

```
165  class_lable_t class_lable[CLASS_NUMBER] =
166  {
167      { "no_mask", NAVY },
168      { "mask", DARKGREEN }
169  };
```

- 定义要识别的类别对应的边框颜色
- 没口罩：蓝
- 有口罩：绿

```
174  static void lable_init(void)
175  {
176  #if (CLASS_NUMBER > 1)
177      uint8_t index;
178
179      class_lable[0].height = 16;
180      class_lable[0].width = 8 * strlen(class_lable[0].str);
181      class_lable[0].ptr = lable_string_draw_ram;
182      lcd_ram_draw_string(class_lable[0].str, class_lable[0].ptr, BLACK, class_lable[0].color);
183      for (index = 1; index < CLASS_NUMBER; index++) {
184          class_lable[index].height = 16;
185          class_lable[index].width = 8 * strlen(class_lable[index].str);
186          class_lable[index].ptr = class_lable[index - 1].ptr + class_lable[index - 1].height * class_lable[index - 1].width / 2;
187          lcd_ram_draw_string(class_lable[index].str, class_lable[index].ptr, BLACK, class_lable[index].color);
188      }
189  #endif
190  }
```

功能：初始化检测框

具体实现：

- 定义类别0初始边框高16，款8*字符数，指向字符存储地址；画出黑色的字符边框背景蓝；
- 定义类别0初始边框高16，款8*字符数，指向字符存储地址（去除类别0占用空间）；画出黑色的字符边框背景绿；

实验例程-face mask detection

```
192  static void drawboxes(uint32_t x1, uint32_t y1, uint32_t x2, uint32_t y2, uint32_t class, float prob)
193  {
194      if (x1 >= 320)
195          x1 = 319;
196      if (x2 >= 320)
197          x2 = 319;
198      if (y1 >= 240)
199          y1 = 239;
200      if (y2 >= 240)
201          y2 = 239;
202
203  #if (CLASS_NUMBER > 1)
204      lcd_draw_rectangle(x1, y1, x2, y2, 2, class_lable[class].color);
205      lcd_draw_picture(x1 + 1, y1 + 1, class_lable[class].width, class_lable[class].height, class_lable[class].ptr);
206  #else
207      lcd_draw_rectangle(x1, y1, x2, y2, 2, RED);
208  #endif
209  }
```

功能：画检测框

具体实现：

- 规范边界坐标
- 如果类别多于1：根据类别确定颜色，线宽为2；将类别以图像显示在LCD上；
- 否则只显示红色边界框

实验例程-face mask detection

```
266  int main(void)
267  {
268      /* Set CPU and dvp clk */
269      sysctl_pll_set_freq(SYSCTL_PLL0, PLL0_OUTPUT_FREQ);
270      sysctl_pll_set_freq(SYSCTL_PLL1, PLL1_OUTPUT_FREQ);
271      sysctl_clock_enable(SYSCTL_CLOCK_AI);
272      uarths_init();
273      plic_init();
274      io_set_power();
275      io_init();
276      lable_init();
277      /* flash init */
278      printf("flash init\n");
279      w25qxx_init(3, 0);
280      w25qxx_enable_quad_mode();
281  #if LOAD_KMODEL_FROM_FLASH
282      w25qxx_read_data(0xA00000, model_data, KMODEL_SIZE, W25QXX_QUAD_FAST);
283  #endif
284      /* LCD init */
285      printf("LCD init\n");
286      lcd_init();
287      lcd_set_direction(DIR_YX_RLDU);
288      lcd_clear(BLACK);
289      lcd_draw_string(136, 70, "DEMO", WHITE);
290      lcd_draw_string(104, 150, "face mask detection", WHITE);
291      /* DVP init */
292      printf("DVP init\n");
293      dvp_init(8);
294      dvp_set_xclk_rate(24000000);
295      dvp_enable_burst();
296      dvp_set_output_enable(0, 1);
297      dvp_set_output_enable(1, 1);
298      dvp_set_image_format(DVP_CFG_RGB_FORMAT);
299      dvp_set_image_size(320, 240);
302  #elif (BOARD_VERSION == BOARD_V1_3)
303      gc0328_init();
304      open_gc0328_1();
305  #endif
```

- 初始化CPU和dvp时钟：设置时钟频率；使能时钟；初始化uarths，时钟源为PLL0；初始化外部中断；初始化IO；初始化检测框；初始化flash
- 如果从flash中读权重，执行读操作
- 初始化LCD：初始化lcd；设置显示朝向；清屏；显示初始字符
- 初始化DVP：初始化dvp，寄存器长8；设置时钟频率；使能突发传输模式；使能输出；设置图像格式RGB；图像大小（320，240）
- 初始化摄像头；打开摄像头1

实验例程-face mask detection

北京大学 PEKING UNIVERSITY

```
307     kpu_image.pixel = 3;
308     kpu_image.width = 320;
309     kpu_image.height = 256;
310     image_init(&kpu_image);
311     display_image.pixel = 2;
312     display_image.width = 320;
313     display_image.height = 240;
314     image_init(&display_image);
315     memset(kpu_image.addr, 127, kpu_image.pixel * kpu_image.width * kpu_image.height);
316     dvp_set_ai_addr((uint32_t)(kpu_image.addr + 8 * 320), (uint32_t)(kpu_image.addr + 320 * 256 + 8 * 320),
317     (uint32_t)(kpu_image.addr + 320 * 256 * 2 + 8 * 320));
318     dvp_set_display_addr((uint32_t)display_image.addr);
319     dvp_config_interrupt(DVP_CFG_START_INT_ENABLE | DVP_CFG_FINISH_INT_ENABLE, 0);
320     dvp_disable_auto();
321     /* DVP interrupt config */
322     printf("DVP interrupt config\n");
323     plic_set_priority(IRQN_DVP_INTERRUPT, 1);
324     plic_irq_register(IRQN_DVP_INTERRUPT, dvp_irq, NULL);
325     plic_irq_enable(IRQN_DVP_INTERRUPT);
326     /* init obj detect model */
327     if (kpu_load_kmodel(&obj_detect_task, model_data) != 0)
328     {
329         printf("\nmodel init error\n");
330         while (1);
331     }
332     obj_detect_rl.anchor_number = ANCHOR_NUM;
333     obj_detect_rl.anchor = anchor;
334     obj_detect_rl.threshold = 0.7;
335     obj_detect_rl.nms_value = 0.4;
336     obj_detect_rl.classes = CLASS_NUMBER;
337     region_layer_init(&obj_detect_rl, 10, 8, (4 + 2 + 1) * ANCHOR_NUM, kpu_image.width, kpu_image.height);
338     /* enable global interrupt */
339     sysctl_enable_irq();
340
341 #if (BOARD_VERSION == BOARD_V1_3)
342     tick_init(TICK_NANOSECONDS);
343 #endif
```

- 初始化kpu图像；初始化显示图像
- 写入kpu图像
- 设置 AI 存放图像的地址：红在kup图像地址基础上增加8*320，绿在kup图像地址基础上增加图像大小（红）和8*320，蓝加2个（红绿）图像大小和8*320
- 设置采集图像在内存中的存放地址，dvp中断为禁用状态，禁用自动接收图像，是指dvp中断优先级1并使能
- 初始化模型：初始化检测anchor数，识别置信度阈值，nms阈值，类别数，初始化输出层
- 使能系统中断，初始化定时器

实验例程-face mask detection

```
345     /* system start */
346     printf("System start\n");
347     while (1)
348     {
349 #if (BOARD_VERSION == BOARD_V1_3)
350         if (KEY_PRESS == key_get())
351         {
352             camera_switch();
353         }
354 #endif
355         g_dvp_finish_flag = 0;
356         dvp_clear_interrupt(DVP_STS_FRAME_START | DVP_STS_FRAME_FINISH);
357         dvp_config_interrupt(DVP_CFG_START_INT_ENABLE | DVP_CFG_FINISH_INT_ENABLE, 1);
```

```
358     while (g_dvp_finish_flag == 0)
359         ;
360     /* run obj detect */
361     g_ai_done_flag = 0;
362     kpu_run_kmodel(&obj_detect_task, kpu_image.addr, DMAC_CHANNEL5, ai_done, NULL);
363     while(!g_ai_done_flag);
364     float *output;
365     size_t output_size;
366     kpu_get_output(&obj_detect_task, 0, (uint8_t **)&output, &output_size);
367     obj_detect_rl.input = output;
368     region_layer_run(&obj_detect_rl, &obj_detect_info);
369     /* display pic*/
370     lcd_draw_picture(0, 0, 320, 240, display_image.addr);
371     g_dvp_finish_flag = 0;
372
373     /* draw boxs */
374     region_layer_draw_boxes(&obj_detect_rl, drawboxes);
```

- 按按键执行切换摄像头
- 清除摄像头接口中断标志
- 在kpu中运行kmodel进行目标检测
- 获取 KPU处理得到的模型结果，任务目标检测，存到output中
- 将结果赋给检测框结构体
- 得到边框结果
- 显示获得的图像
- 显示得到的检测框和类别结果

项目代码更改-baidu screw

```python
def multi_process_custom_reader(file_path, data_dir, num_workers, input_size, mode):
    file_path = os.path.join(data_dir, file_path)
    readers = []
    images = [line.strip() for line in open(file_path)]
    # n = int(math.ceil(len(images) // num_workers))
    # image_lists = [images[i: i + n] for i in range(0, len(images), n)]
    image_lists = images
    # for l in image_lists:
        # readers.append(paddle.batch(custom_reader(l, data_dir, input_size, mode),
        #                             batch_size=train_parameters['train_batch_size']))
    readers.append(paddle.batch(custom_reader(image_lists, data_dir, input_size, mode),
                                batch_size=train_parameters['train_batch_size']))
    return paddle.reader.multiprocess_reader(readers, False)
```

```python
                loss = fluid.layers.yolov3_loss(
                    x=out,
                    gt_box=gt_box,
                    gt_label=gt_label,
                    anchors=model.get_anchors(),
                    anchor_mask=model.get_anchor_mask()[i],
                    class_num=model.get_class_num(),
                    ignore_thresh=train_parameters['ignore_thresh'],
                    downsample_ratio=downsample_ratio)
```

```python
train_parameters = {
    "data_dir": "data/data6045/",
    "file_list": "train.txt",
    "class_dim": -1,
    "label_dict": {},
    "image_count": -1,
    "continue_train": False,      # 是否加载前一次的训练参数，接着训练
    "pretrained": False,
    "pretrained_model_dir": "./pretrained-model",
    "save_model_dir": "./yolo-model",
    "model_prefix": "yolo-v3",
    "use_tiny": True,          # 是否使用 裁剪 tiny 模型
    "max_box_num": 20,          # 一幅图上最多有多少个目标
    "num_epochs": 120,
    "train_batch_size": 5,      # 对于完整 yolov3，每一批的训练样本不能太多，内存会炸掉
    "use_gpu": False,
    "yolo_cfg": {
        #"input_size": [3, 608, 608],
        #"anchors": [10, 13,  16, 30,  33, 23,  30, 61,  62, 45,  59, 119,  116, 90,  156, 198,  373, 326],
        "input_size": [3, 416, 416],
        "anchors": [7, 9,  11, 21,  23, 16,  21, 42,  43, 38,  41, 82,  80, 62,  107, 136,  256, 224],
        "anchor_mask": [[6, 7, 8], [3, 4, 5], [0, 1, 2]]
    },
    "yolo_tiny_cfg": {
        "input_size": [3, 256, 256],
        "anchors": [6, 8, 13, 15, 22, 34, 48, 50, 81, 100, 205, 191],
        "anchor_mask": [[3, 4, 5], [0, 1, 2]]
    },
    "ignore_thresh": 0.7,
    "mean_rgb": [127.5, 127.5, 127.5],
    "mode": "train",
    "multi_data_reader_count": 0,
    "apply_distort": True,
    "valid_thresh": 0.01,
    "nms_thresh": 0.45,
```

项目代码更改-baidu screw

北京大学 PEKING UNIVERSITY

imaug新建终端安装

```
(yolo) [root@qingteng K210_Yolo_framework-master]# conda install -c conda-forge imgaug
```

make_voc_list.py

```
6   import skimage
7   from skimage import io
23              np.array(io.imread(image_path_list[i]).shape[0:2])]
```

```
75                  loss = fluid.layers.yolov3_loss(
76                      x=out,
77                      gt_box=gt_box,
78                      gt_label=gt_label,
79                      anchors=model.get_anchors(),
80                      anchor_mask=model.get_anchor_mask()[i],
81                      class_num=model.get_class_num(),
82                      ignore_thresh=train_parameters['ignore_thresh'],
83                      downsample_ratio=downsample_ratio)
```

```
28  train_parameters = {
29      "data_dir": "data/data6045/",
30      "file_list": "train.txt",
        "class_dim": -1,
32      "label_dict": {},
33      "image_count": -1,
34      "continue_train": False,      # 是否加载前一次的训练参数，接着训练
35      "pretrained": False,
36      "pretrained_model_dir": "./pretrained-model",
37      "save_model_dir": "./yolo-model",
38      "model_prefix": "yolo-v3",
39      "use_tiny": True,          # 是否使用 裁剪 tiny 模型
40      "max_box_num": 20,          # 一幅图上最多有多少个目标
41      "num_epochs": 120,
42      "train_batch_size": 5,      # 对于完整 yolov3，每一批的训练样本不能太多，内存会炸掉
43      "use_gpu": False,
44      "yolo_cfg": {
45          #"input_size": [3, 608, 608],
46          #"anchors": [10, 13,  16, 30,  33, 23,  30, 61,  62, 45,  59, 119,  116, 90,  156, 198,  373, 326],
47          "input_size": [3, 416, 416],
48          "anchors": [7, 9,  11, 21,  23, 16,  21, 42,  43, 38,  41, 82,  80, 62,  107, 136,  256, 224],
49          "anchor_mask": [[6, 7, 8], [3, 4, 5], [0, 1, 2]]
50      },
51      "yolo_tiny_cfg": {
52          "input_size": [3, 256, 256],
53          "anchors": [6, 8, 13, 15, 22, 34, 48, 50, 81, 100, 205, 191],
54          "anchor_mask": [[3, 4, 5], [0, 1, 2]]
55      },
56      "ignore_thresh": 0.7,
57      "mean_rgb": [127.5, 127.5, 127.5],
58      "mode": "train",
59      "multi_data_reader_count": 0,
60      "apply_distort": True,
61      "valid_thresh": 0.01,
62      "nms_thresh": 0.45,
```

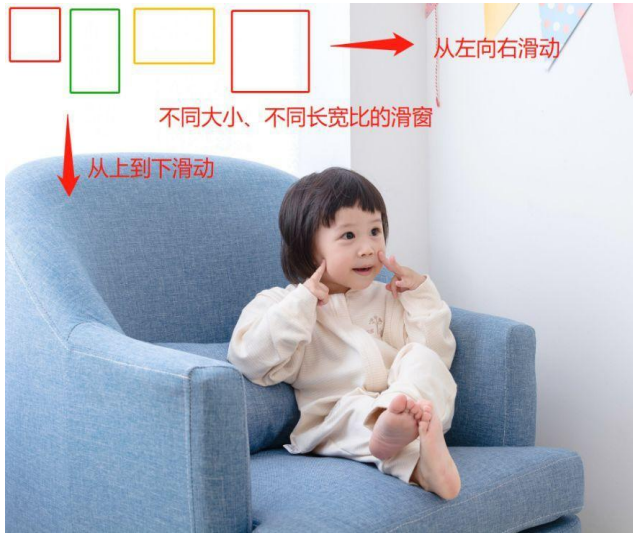
项目代码更改-face mask detection

```
{
    "model" : {
        "architecture":         "MobileNet",
        "input_size":           224,
        "anchors":              [1, 1.2, 2, 3, 4, 3, 6, 4, 5, 6.5],
        "labels":               ["raccoon"],
        "coord_scale" :         1.0,
        "class_scale" :         1.0,
        "object_scale" :    5.0,
        "no_object_scale" :     1.0
    },
    "pretrained" : {
        "full":   ""
    },
    "train" : {
        "actual_epoch":         50,
        "train_image_folder":   "train_img",
        "train_annot_folder":   "train_ann",
        "train_times":          2,
        "valid_image_folder":   "",
        "valid_annot_folder":   "",
        "valid_times":          1,
        "batch_size":           16,
        "learning_rate":        1e-4,
        "saved_folder":         "save",
        "first_trainable_layer": "",
        "jitter":           true,
        "is_only_detect" :      false
    }
}
```

- architecture：此代码支持MobileNet（实际是YOLO_v2）
- labels：待识别物体的类别
- pretrained：如果有预训练模型可以写上预训练模型的路径
- train_image_folder：自己数据集的训练集图片路径
- train_annot_folder：生成或下载的xml格式标注的路径
- saved_folder：结果保存路径

```
#define ANCHOR_NUM 5  //5 or 9
#if ANCHOR_NUM == 5
INCBIN(model, "detect_5.kmodel");
#endif
#if ANCHOR_NUM == 9
INCBIN(model, "detect_9.kmodel");
#endif
#endif
```

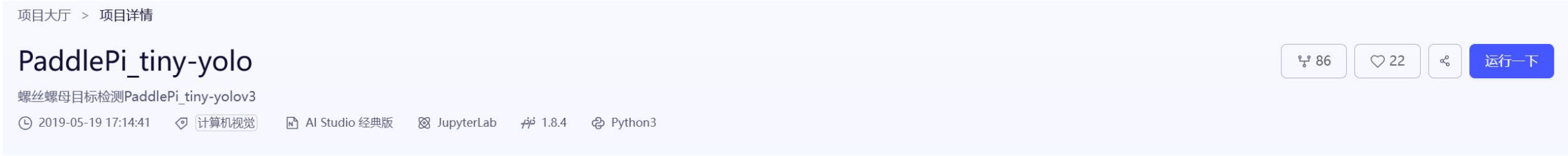
- ANCHOR_NUM：在识别过程中定义的窗口形状数量
- detect_5.kmodel：改成自己训练转换得到的模型的名称



操作流程

北京大学 PEKING UNIVERSITY

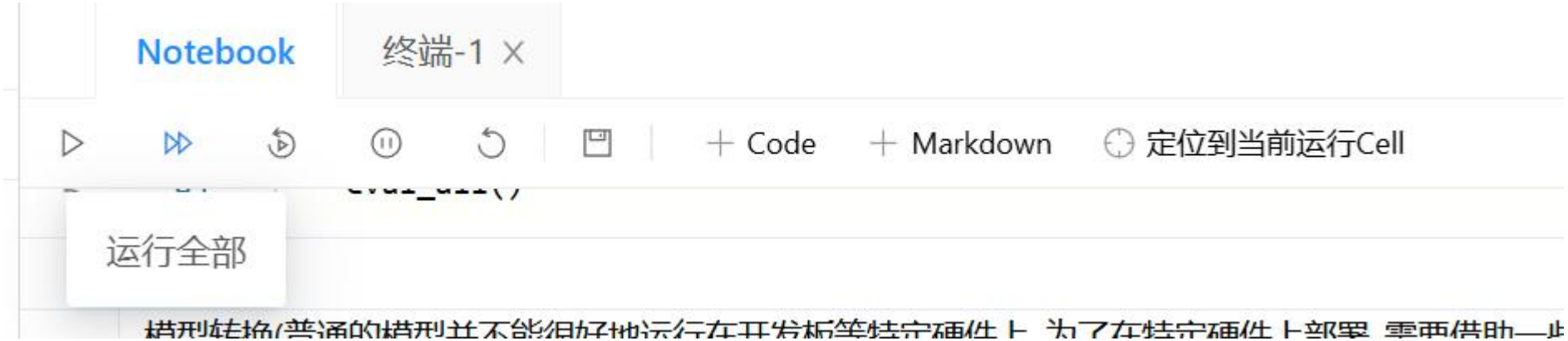
- 进入网站：https://aistudio.baidu.com/projectdetail/61305



- 开始训练模型：开始运行->启动环境->基础版->进入jupyter notebook界面

操作流程

- 运行全部



- 等待模型训练、测试、转换完成

work
data
logs
yolo-model

```
                    logger.info(
                        "Pass {0}, trainbatch {1}, loss {2} time {3}".format(pass_id, batch_id, loss, "%2.2f sec" % period))
                # 采用简单的定时采样停止办法，可以调整为更精细的保存策略
                if total_batch_count % 100 == 0:
                    logger.info("temp save {0} batch train result".format(total_batch_count))
                    fluid.io.save_persistables(dirname=train_parameters['save_model_dir'],
                                               main_program=train_program,
                                               executor=exe)
            except fluid.core.EOFException:
                train_reader.reset()

        logger.info("training till last epcho, end training")
        fluid.io.save_persistables(dirname=train_parameters['save_model_dir'], main_program=train_program, executor=exe)


if __name__ == '__main__':
```

```
2024-11-11 22:13:27,932-INFO: Pass 0, trainbatch 40, loss 264.3001708984375 time 1.16 sec
2024-11-11 22:13:27,932 - 1511849593.py[line:52] - INFO: Pass 0, trainbatch 40, loss 264.3001708984375 time 1.16 sec
2024-11-11 22:13:39,979-INFO: Pass 0, trainbatch 50, loss 195.87741088867188 time 1.05 sec
```

操作流程-baidu screw

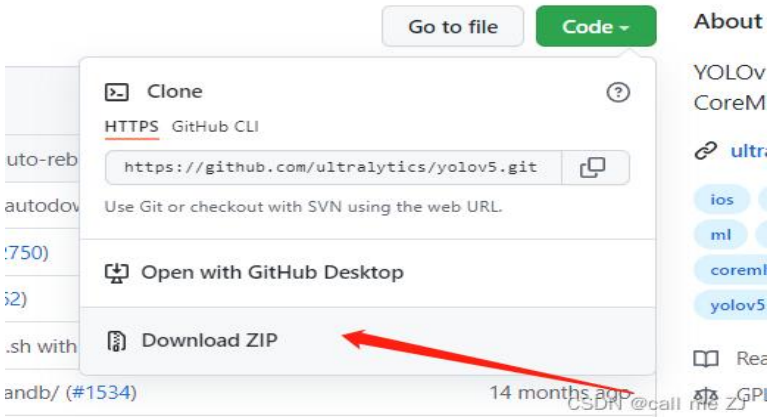
北京大学 PEKING UNIVERSITY

- 将AI Studio上训练的模型下载，拷贝到示例代码中进行编译并烧录程序。

交叉编译环境和SDK 参考文档《PaddlePi-K210命令行开发环境搭建指南》
1）下载示例程序baidu_screw，将 baidu_screw 目录放到kendryte-standalone-sdk的src目录下
2）下载在AI Studio上训练好的模型tiny_yolo.kmodel到 baidu_screw 目录下
3）在kendryte-standalone-sdk（请使用develop分支的代码）目录新建build文件夹。如：
D:\Documents\kendryte-standalone-sdk\build
4）编译
编译完成在build目录生成baidu_screw.bin
5）烧录程序
K-flash.exe安装参考文档《PaddlePi-K210命令行开发环境搭建指南》
开发板插上USB Type-C，上电进入ISP模式。运行K-flash.exe，选择 Device 、设置波特率、选择
baidu_screw.bin，点击 Flash 开始下载。
6）运行测试
烧录完成后自动重启，使用图片进行测试（可使用AI Studio里面的数据集图片进行测试）。

操作流程-face mask detection

1. 环境的安装：Anaconda 的安装、Pytorch环境安装、 pip install -r requirements.txt
2. 项目克隆，GitHubyolov5官网：
   https://gitcode.com/gh_mirrors/yo/yolov5/tree/v5.0?utm_source=csdn_github_accelerator&isLogin=1



3. 解压并从项目根目录打开

操作流程-face mask detection

北京大学 PEKING UNIVERSITY

4. 数据集的准备
5. 项目克隆，GitHubyolov5官网：https://gitcode.com/gh_mirrors/yo/yolov5/tree/v5.0?utm_source=csdn_github_accelerator&isLogin=1
6. 下载工具,下载ncc工具箱,将[ncc_0.1_win.zip]放置在工程根目录，解压到当前文件夹
   网盘下载：https://pan.baidu.com/s/1NT2tG4Rv2YJyjOKRh-3t4w 提取码：z9fr
7. 开始训练:python train.py -c configs.json
8. 模型转换：ncc_0.1_win\ncc test.tflite test.kmodel -i tflite -o k210model --dataset train_img
9. 转换完成根目录会出现test.kmodel，kmodel模型放到face-mask-detection目录下
10. 编译：编译完成在build目录生成.bin文件
11. 烧录程序：运行K-flash.exe，选择 Device 、设置波特率、选择.bin，点击 Flash 开始下载
12. 运行测试：烧录完成后自动重启，使用图片进行测试（可使用数据集里的图片进行测试）。

效果展示