DIP 2021 spring Final Project — Object Detection

— 0712238 Yan-Tong Lin

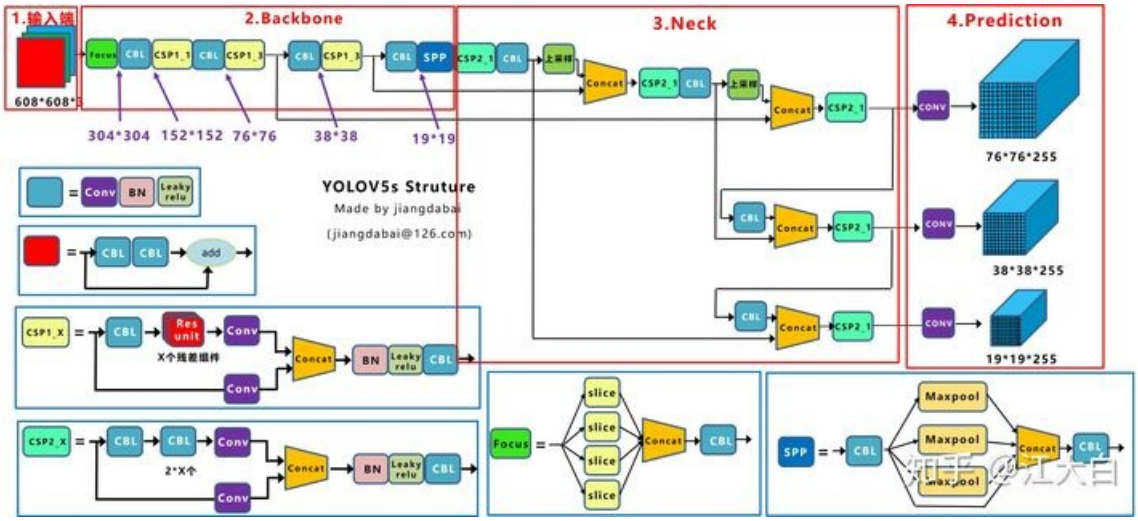
Links of Codes and Results

- the online version of this report
- GitHub repo link
- video link (Google drive)
- video link (Youtube)

Method

I adopt `yolov5`, a deep learning model, for this project. Though being not an official new version of YOLO and controversial, it is quite efficient and accurate (yolov5x AP_{test}=50.4) for the task of multiscaled object detection. (The SOTA of multiscaled objective detection is swin transformer with AP_{test}=58.7 based on paperswithcode) A brief introduction to YOLOv5 is given.

YOLO v5



> from https://zhuanlan.zhihu.com/p/172121380

YOLOs is a series of models designed manually to specialize in real time multi-scaled object detection. The architecture of YOLO v5, similar to its v3, v4 counterparts, can be decomposed to 4 parts — Input, Backbone, Neck, and Head.
A variety of strategies and improvements is applied throughout the 4 building blocks of the model.

- Input (data enhancement)
  - Mosaic data enhancement
  - adaptive anchoring
  - etc.
- Backbone (feature extraction)
  - Focus
  - CSPDarknet
    - solves the problems of repeated gradient information in large-scale backbones and

integrates the gradient changes into the feature map.

- Neck (feature fusion)
  - PAN+FPN
    - enhanced bottom-up path, which improves the propagation of low-level features.
- Head (prediction)
  - Yolo Layer
    - different sizes of feature maps to achieve multi-scale prediction.
  - GIOU_Loss

Source Code — `my_detect.py`

To meet the requirements (display my student id and the object counts), I modify `detect.py` of `yolov5` project to `my_detect.py`. The source code of `my_detect.py` is attached below.
If the source code is cut due to the limitation of page width, please visitthe online version of this report or the GitHub repo.

```
import argparse
import time
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow, non_max_s
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path, save_one_b
from utils.plots import colors, plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized


# 0712238, for counting with defaultdict(int)
from collections import defaultdict


@torch.no_grad()
def detect(opt):
    source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, opt.view_im
    save_img = not opt.nosave and not source.endswith('.txt')  # save inference image
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswi
        ('rtsp://', 'rtmp://', 'http://', 'https://'))

    # Directories
    save_dir = increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok)  #
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = opt.half and device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size
    names = model.module.names if hasattr(model, 'module') else model.names  # get cl
    if half:
        model.half()  # to FP16

    # Second-stage classifier
```

```
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101', n=2)  # initialize
        modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device

    # Set Dataloader
    vid_path, vid_writer = None, None
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride)

    # Run inference
    if device.type != 'cpu':
        model(torch.zeros(1, 3, imgsz, imgsz).to(device).type_as(next(model.parameter
    t0 = time.time()
    for path, img, im0s, vid_cap in dataset:
        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Inference
        t1 = time_synchronized()
        pred = model(img, augment=opt.augment)[0]

        # Apply NMS
        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, opt.classes,
                                   max_det=opt.max_det)
        t2 = time_synchronized()

        # Apply Classifier
        if classify:
            pred = apply_classifier(pred, modelc, img, im0s)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if webcam:  # batch_size >= 1
                p, s, im0, frame = path[i], f'{i}: ', im0s[i].copy(), dataset.count
            else:
                p, s, im0, frame = path, '', im0s.copy(), getattr(dataset, 'frame', 0

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # img.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'im
            s += '%gx%g ' % img.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if opt.save_crop else im0  # for opt.save_crop
            # 0712238
            cnt = defaultdict(int)
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
```

```
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
                        line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or opt.save_crop or view_img:  # Add bbox to image
                        c = int(cls)  # integer class
                        ## 0712238, filter out irrelevant results
                        if names[c] in opt.cls:
                            cnt[names[c]] += 1
                            label = None if opt.hide_labels else (names[c] if opt.hid
                            plot_one_box(xyxy, im0, label=label, color=colors(c, True
                            if opt.save_crop:
                                save_one_box(xyxy, imc, file=save_dir / 'crops' / nam

            # 0712238, print counts
            text = '0712238\n'
            for c in opt.cls:
                if c == 'person':
                    text += f'the number of people detected: {cnt[c]}\n'
                else:
                    text += f'the number of {c}s detected: {cnt[c]}\n'
            y0, dy = 100, 50
            for i, txt in enumerate(text.split('\n')):
                y = y0+i*dy
                # image, text, coord, font, size, color, thickness, anti-aliasing
                cv2.putText(im0, txt, (100, y), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0,

            # Print time (inference + NMS)
            print(f'{s}Done. ({t2 - t1:.3f}s)')

            # Stream results
            if view_img:
                cv2.imshow(str(p), im0)
                cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path != save_path:  # new video
                        vid_path = save_path
                        if isinstance(vid_writer, cv2.VideoWriter):
                            vid_writer.release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                            save_path += '.mp4'
                        vid_writer = cv2.VideoWriter(save_path, cv2.VideoWriter_fourc
                    vid_writer.write(im0)

    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir
        print(f"Results saved to {save_dir}{s}")

    print(f'Done. ({time.time() - t0:.3f}s)')


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default='yolov5s.pt', help=
    parser.add_argument('--source', type=str, default='data/images', help='source')
```

```
    parser.add_argument('--img-size', type=int, default=640, help='inference size (pi
    parser.add_argument('--conf-thres', type=float, default=0.25, help='object confid
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold
    parser.add_argument('--max-det', type=int, default=1000, help='maximum number of
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3
    parser.add_argument('--view-img', action='store_true', help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.tx
    parser.add_argument('--save-conf', action='store_true', help='save confidences in
    parser.add_argument('--save-crop', action='store_true', help='save cropped predic
    parser.add_argument('--nosave', action='store_true', help='do not save images/vid
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --cl
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic N
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default='runs/detect', help='save results to pro
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/nam
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box t
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hi
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision
    ## 0712238, add option to filter out irrelevant classes and show only the desired
    parser.add_argument('--cls', nargs='+', default=[], help='filter by the names of
    opt = parser.parse_args()
    print(opt)
    check_requirements(exclude=('tensorboard', 'thop'))

    if opt.update:  # update all models (to fix SourceChangeWarning)
        for opt.weights in ['yolov5s.pt', 'yolov5m.pt', 'yolov5l.pt', 'yolov5x.pt']:
            detect(opt=opt)
            strip_optimizer(opt.weights)
    else:
        detect(opt=opt)
```

A Step by Step Guide to Reproduce the Result

- prepare your video
- clone the yolov5 project
  - `git clone https://github.com/ultralytics/yolov5.git`
- use `my_detect.py` to get the result
  - `python my_detect.py --source {your video} --cls {desired classes} --weights {yolo weight}`
- the project hierarchy

```
final/
    - yolov5/
        - my_detect.py
        - ...
    - sample_videos/
        - person_bicycle.mp4
        - ...
```

- the parameters for my result
  - `python my_detect.py --source ..\sample_videos\person_bicycle.mp4 --cls person bicycle --weights yolov5x.pt`

A Better Performance

A better performance is illustrated in the following screen shots.
The adopted model successfully detects the pair of person and bicyble in the back of another pair as well as avoids the misclassification in the middle of the picture.

- baseline (YOLOv3)

- this work (YOLOv5)

YOLOv5 is a relatively new model compared to the baseline (YOLOv3), so it is not surprising that it outperforms YOLOv3. Also, I adopt the `yolov5x` weight, which is the largest model available, to do the inference.

Possibilities for Further Improvement

Swin Transformer

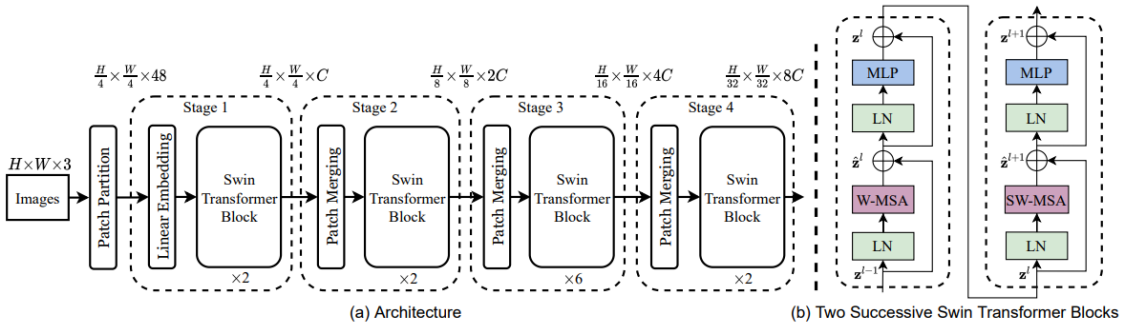


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

> from https://arxiv.org/pdf/2103.14030.pdf

According to paperswithcode, the SOTA of multiscaled objective detection is based on swin transformer with AP_{test}=58.7. Swin transformer is a successful attempt to apply transformers (which I am familiar with due to my experience during the summer internship at IIS NLU lab) to computer vision tasks. It utilizes “Shifted Window based Multihead Self Attention” and other features to preserve the advantages of CNNs (locality, translation invariance, hierarchical), incorporate the advantages of transformers (computational power), and remain efficient enough.

Finetuning

Since the assignment requires only the detection of people and bicycles, if we finetune the model with a dataset of people and bicycles and a loss function that ignores other classes, we may get a better result. Unfortunately, I do not have easy access to powerful GPU to do finetuning now.

Resources

- https://github.com/ultralytics/yolov5
- https://github.com/microsoft/Swin-Transformer
- https://arxiv.org/pdf/2103.14030.pdf
- technical details of yolov5 model
  - https://www.researchgate.net/publication/349299852_A_Forest_Fire_Detection_System_Based_on_Ensemble_Learning
  - https://zhuanlan.zhihu.com/p/172121380
  - https://www.233tw.com/algorithm/28664
- YOLO 1-5
  - https://zhuanlan.zhihu.com/p/334961642