Lec10_Detection&Segmentation

Yolo(Yolo (You Only Look Once)

- single convolutional neural network
- divide into grids & predict class prob & bounding boxes of each grid
- S×S grid, B bounding boxes, Each box is with a confidence, C class probabilities
- S×S×(B*5+C) tensor, S×S×B bounding boxes

UNet

- architecture-“fully convolutional network” → solve problem: medical img segmentation]
- left part - down-sampling(向下取样)
  - typical convolutional network → repeated application of convolutions, each followed by a rectified ReLU(rectified linear unit) & max pooling.
  - spatial info reduced while feature info increased/
- right part - upsampling
  - combines feature & spatial info through a sequence of up convolutions & concatenations with hight-resolution feature from left

R CNN to Mask RCNN

RCNN(take too long & consume too much space)

1. input img
2. extract region proposal((1k-2k candidates regions generated from each img, selective search generates regional recommendation based on objectives
3. Feature extraction: deep network(AlexNet, VGG, other CNN)
4. SVM classification: send features into a SVM classifier
5. Bounding box regression based on region proposals of the SVM

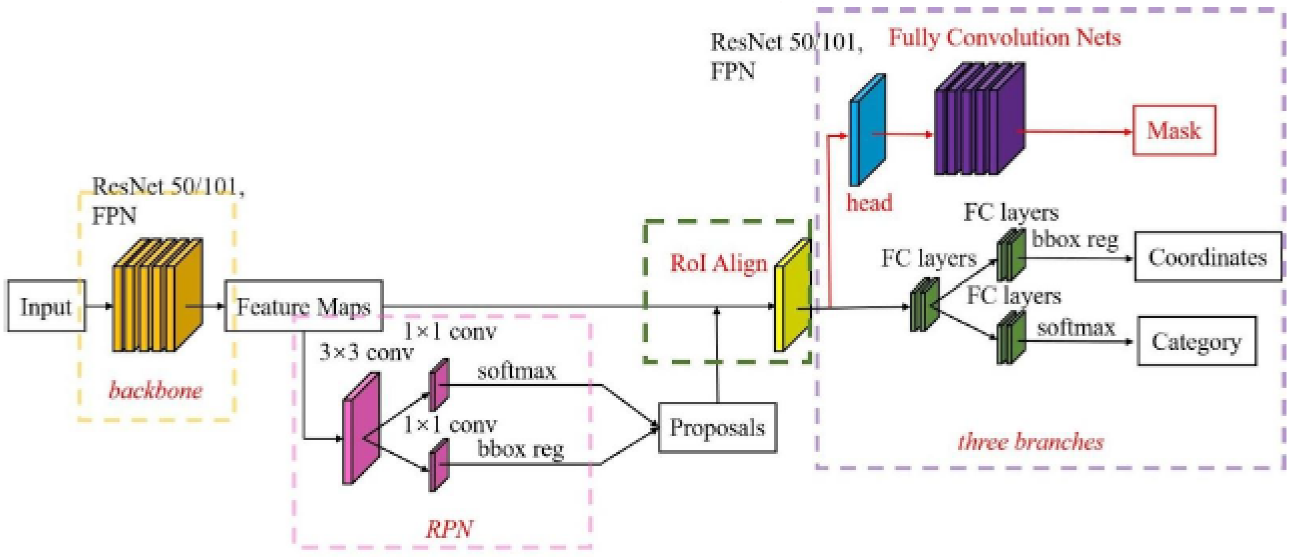
Fast RCNN (selective seach method takes too long)

1. Feed the whole img into CNN
2. Extract feature for each region from feature map directly
3. border regression 边界回归 embedded into the training of the network

Faster RCNN: region proposal Network(RPN) used to generate the region finding candidate boxes is integrated(object detection algorithm

Mask RCNN is sematic segmentation algorithm(latest RCNN)

- ROI align is to replace ROI pooling to improve the accuracy of instance segment
- VGG in Faster RCNN replaced with ...(detailed detection)
  - ResNet(combine low-level feature w/ high-level feature)
  - FPN(mine multi-scale info)
- FCN layer(MASK layer) added for sematic segment



Tutorial

```
# **Yolov5**
%cd /content/yolov5
print("if you want to use the default images, just click `Cancel upload`")
uploaded = files.upload()
if len(uploaded) != 0:
  fnames = list(uploaded.keys())
  for fname in fnames:
    !mv $fname ../images/
!python detect.py --weights yolov5s.pt --img 256 --conf 0.25 --source ../images

# **Dlib**
## conda/pip install dlib
%cd /content
print("if you want to use the default images, just click `Cancel upload`")
uploaded = files.upload()
if len(uploaded) != 0:
  fnames = list(uploaded.keys())
  for fname in fnames:
    !mv $fname ../images/
fnames = [os.path.join('images', fname) for fname in os.listdir('images')]
%cd /content
'''
dlib.get_frontal_face_detector() is a function that returns a pre-trained object
detector
for detecting frontal faces in an image. The detector is based on a
Histogram of Oriented Gradients (HOG) feature descriptor and a
linear support vector machine (SVM) classifier.
'''
detector = dlib.get_frontal_face_detector()
for fname in fnames:
  img = cv2.imread(fname, cv2.IMREAD_COLOR)[..., ::-1]
  dets = detector(img, 1) # 1 means no upscaling
  for rec in dets:
    fig, ax = plt.subplots(1, 1)
    ax.imshow(img)
    l, r, u, b = np.maximum(rec.left(), 0), np.minimum(rec.right(), img.shape[1]),
\
                 np.maximum(rec.top(), 0), np.minimum(rec.bottom(), img.shape[0])
    ax.add_patch(patches.Rectangle((l, u), r-l, b-u, color=(1,1,1,0.5)))
    plt.show()
```

Segmentation

```
# DeepLab v3+
%cd /content
!git clone https://github.com/VainF/DeepLabV3Plus-Pytorch
%cd DeepLabV3Plus-Pytorch
%pip install -r requirements.txt
os.makedirs('checkpoints', exist_ok=True)
if not os.path.isfile('./checkpoints/all.zip'):
  !wget -O ./checkpoints/all.zip
https://www.dropbox.com/sh/w3z9z8lqpi8b2w7/AAB0vkl4F5vy6HdIhmRCTKHSa?dl=1
  !unzip ./checkpoints/all.zip -d ./checkpoints/

%cd /content/DeepLabV3Plus-Pytorch
print("if you want to use the default images, just click `Cancel upload`")
uploaded = files.upload()
if len(uploaded) != 0:
  fnames = list(uploaded.keys())
  for fname in fnames:
    !mv $fname ../images/
!python predict.py --input ../images --model deeplabv3plus_mobilenet --ckpt
checkpoints/best_deeplabv3plus_mobilenet_voc_os16.pth --save_val_results_to
test_results
```

Face Parsing

```
%cd ..
%rm -rf ./face-parsing.PyTorch
%cd /content
!git clone https://github.com/Awenbocc/face-parsing.PyTorch
%cd face-parsing.PyTorch
os.makedirs('res/cp', exist_ok=True)
if not os.path.isfile('./res/cp/79999_iter.pth'):
  !wget -O /content/face-parsing.PyTorch/res/cp/79999_iter.pth
https://drive.google.com/u/0/uc?
id=154JgKpzCPW82qINcVieuPH3fZ2e0P812&export=download

%cd /content/face-parsing.PyTorch
print("if you want to use the default images, just click `Cancel upload`")
uploaded = files.upload()
if len(uploaded) != 0:
  fnames = list(uploaded.keys())
```

```
    for fname in fnames:
      !mv $fname ../images/

  !python test.py
```