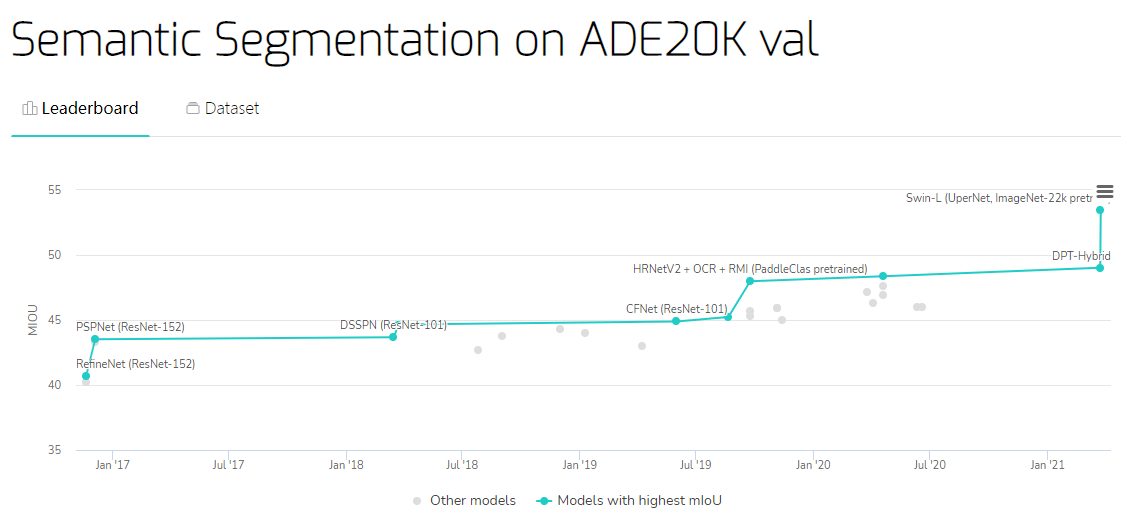
1. **作业**
2. 下载并阅读语义分割的文献，最好下载代码，训练并预测结果

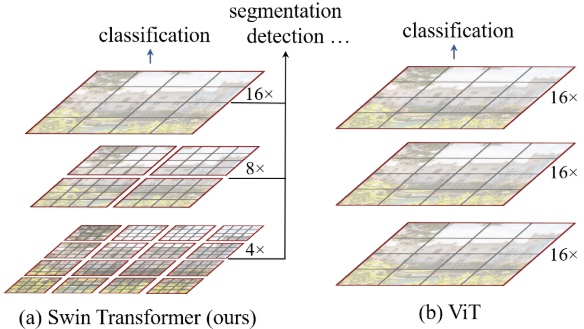
写出设计思想和训练方法

Swin Transformer 2021 ICCV Best Paper

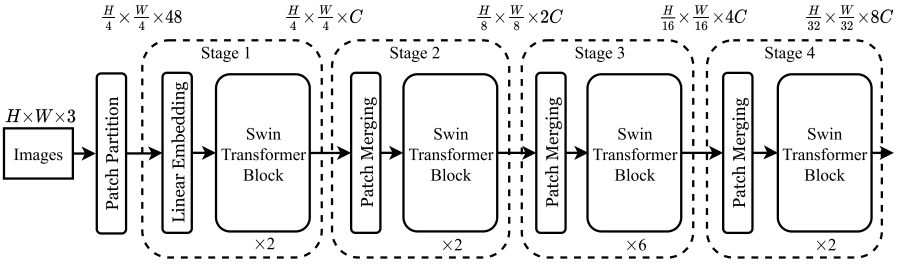


Swin Transformer在语义分割中真的可以为所欲为

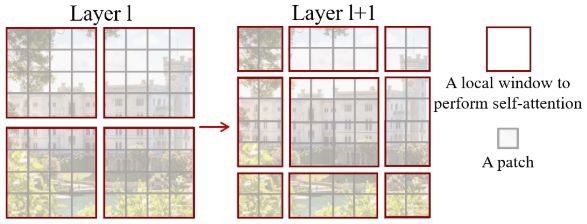
动机：Transformer从NLP迁移到CV上没有大放异彩主要有两点原因：1. 两个领域涉及的scale不同，NLP的scale是标准固定的，而CV的scale变化范围非常大。2. CV比起NLP需要更大的分辨率，而且CV中使用Transformer的计算复杂度是图像尺度的平方，这会导致计算量过于庞大。为了解决这两个问题，Swin Transformer相比之前的ViT做了两个改进：1.引入CNN中常用的层次化构建方式构建层次化Transformer 2.引入locality思想，对无重合的window区域内进行self-attention计算。



相比于ViT，Swin Transfomer计算复杂度大幅度降低，具有输入图像大小线性计算复杂度。Swin Transformer随着深度加深，逐渐合并图像块来构建层次化Transformer，可以作为通用的视觉骨干网络，应用于图像分类、目标检测和语义分割等任务。

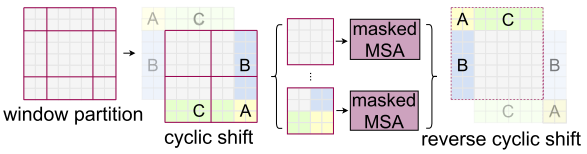


Swin Transformer和ViT划分patch的方式类似，Swin Transformer也是先确定每个patch的大小，然后计算确定patch数量。不同的是，随着网络深度加深ViT的patch数量不会变化，而Swin Transformer随着网络深度的加深数量会逐渐减少并且每个patch的感知范围会扩大，这个设计是为了方便Swin Transformer的层级构建，并且能够适应视觉任务的多尺度。



W-MSA将输入图片划分成不重合的windows，然后在不同的window内进行self-attention计算。

W-MSA虽然降低了计算复杂度，但是不重合的window之间缺乏信息交流，于是作者进一步引入shifted window partition来解决不同window的信息交流问题，在两个连续的Swin Transformer Block中交替使用W-MSA和SW-MSA。以上图为例，将前一层Swin Transformer Block的8x8尺寸feature map划分成2x2个patch，每个patch尺寸为4x4，然后将下一层Swin Transformer Block的window位置进行移动，得到3x3个不重合的patch。移动window的划分方式使上一层相邻的不重合window之间引入连接，大大的增加了感受野。



代码看看就行了，跑的话没卡，溜了溜了

git clone <https://github.com/SwinTransformer/Swin-Transformer-Semantic-Segmentation.git>

然后在<https://github1s.com/SwinTransformer/Swin-Transformer-Semantic-Segmentation>中开发

用dockerfile创建容器

ARG PYTORCH="1.6.0"

ARG CUDA="10.1"

ARG CUDNN="7"

FROM pytorch/pytorch:${PYTORCH}-cuda${CUDA}-cudnn${CUDNN}-devel

ENV TORCH\_CUDA\_ARCH\_LIST="6.0 6.1 7.0+PTX"

ENV TORCH\_NVCC\_FLAGS="-Xfatbin -compress-all"

ENV CMAKE\_PREFIX\_PATH="$(dirname $(which conda))/../"

RUN apt-get update && apt-get install -y git ninja-build libglib2.0-0 libsm6 libxrender-dev libxext6 \

 && apt-get clean \

 && rm -rf /var/lib/apt/lists/\*

# Install mmsegmentation

RUN conda clean --all

RUN pip install mmcv-full==latest+torch1.6.0+cu101 -f https://download.openmmlab.com/mmcv/dist/index.html

RUN git clone https://github.com/open-mmlab/mmsegmenation.git /mmsegmentation

WORKDIR /mmsegmentation

RUN pip install -r requirements/build.txt

RUN pip install --no-cache-dir -e .

之后python tools/test.py ./upernet\_swin\_tiny\_patch4\_window7\_512x512\_160k\_ade20k.py ./upernet\_swin\_small\_patch4\_window7\_512x512.pth --eval mIoU

日志如下

文本

描述已自动生成

结果

{"mode": "train", "epoch": 127, "iter": 159450, "lr": 0.0, "memory": 9004, "data\_time": 0.00819, "decode.loss\_seg": 0.18561, "decode.acc\_seg": 74.17458, "aux.loss\_seg": 0.116, "aux.acc\_seg": 71.31023, "loss": 0.30161, "time": 0.49021}

{"mode": "train", "epoch": 127, "iter": 159500, "lr": 0.0, "memory": 9004, "data\_time": 0.00698, "decode.loss\_seg": 0.18536, "decode.acc\_seg": 74.01761, "aux.loss\_seg": 0.11908, "aux.acc\_seg": 70.92154, "loss": 0.30443, "time": 0.47604}

{"mode": "train", "epoch": 127, "iter": 159550, "lr": 0.0, "memory": 9004, "data\_time": 0.00777, "decode.loss\_seg": 0.19661, "decode.acc\_seg": 74.90445, "aux.loss\_seg": 0.12626, "aux.acc\_seg": 71.75307, "loss": 0.32288, "time": 0.49018}

{"mode": "train", "epoch": 127, "iter": 159600, "lr": 0.0, "memory": 9004, "data\_time": 0.01033, "decode.loss\_seg": 0.19508, "decode.acc\_seg": 75.03985, "aux.loss\_seg": 0.12427, "aux.acc\_seg": 71.95869, "loss": 0.31935, "time": 0.47125}

{"mode": "train", "epoch": 127, "iter": 159650, "lr": 0.0, "memory": 9004, "data\_time": 0.01099, "decode.loss\_seg": 0.19951, "decode.acc\_seg": 75.2465, "aux.loss\_seg": 0.12869, "aux.acc\_seg": 71.99489, "loss": 0.3282, "time": 0.48476}

{"mode": "train", "epoch": 127, "iter": 159700, "lr": 0.0, "memory": 9004, "data\_time": 0.00945, "decode.loss\_seg": 0.1931, "decode.acc\_seg": 73.37125, "aux.loss\_seg": 0.122, "aux.acc\_seg": 70.46385, "loss": 0.3151, "time": 0.48621}

{"mode": "train", "epoch": 127, "iter": 159750, "lr": 0.0, "memory": 9004, "data\_time": 0.00822, "decode.loss\_seg": 0.18035, "decode.acc\_seg": 74.55851, "aux.loss\_seg": 0.11159, "aux.acc\_seg": 71.86345, "loss": 0.29194, "time": 0.48014}

{"mode": "train", "epoch": 127, "iter": 159800, "lr": 0.0, "memory": 9004, "data\_time": 0.00866, "decode.loss\_seg": 0.18698, "decode.acc\_seg": 74.05446, "aux.loss\_seg": 0.11713, "aux.acc\_seg": 71.17066, "loss": 0.30411, "time": 0.48956}

{"mode": "train", "epoch": 127, "iter": 159850, "lr": 0.0, "memory": 9004, "data\_time": 0.01056, "decode.loss\_seg": 0.18598, "decode.acc\_seg": 73.51646, "aux.loss\_seg": 0.1202, "aux.acc\_seg": 70.40939, "loss": 0.30618, "time": 0.47983}

{"mode": "train", "epoch": 127, "iter": 159900, "lr": 0.0, "memory": 9004, "data\_time": 0.00876, "decode.loss\_seg": 0.19023, "decode.acc\_seg": 73.67225, "aux.loss\_seg": 0.12165, "aux.acc\_seg": 70.65798, "loss": 0.31188, "time": 0.49436}

{"mode": "train", "epoch": 127, "iter": 159950, "lr": 0.0, "memory": 9004, "data\_time": 0.01006, "decode.loss\_seg": 0.18031, "decode.acc\_seg": 75.47766, "aux.loss\_seg": 0.11307, "aux.acc\_seg": 72.64909, "loss": 0.29338, "time": 0.48057}

{"mode": "val", "epoch": 127, "iter": 160000, "lr": 0.0, "mIoU": 0.4764, "mAcc": 0.5878, "aAcc": 0.8245}

1. 下载并阅读目标识别的文献，最好下载代码，训练并预测结果

写出设计思想和训练方法

目标识别也用swin transformer 这个网络真的可以为所欲为

Git clone <https://github.com/SwinTransformer/Swin-Transformer-Object-Detection.git>

python tools/test.py ./mask\_rcnn\_swin\_tiny\_patch4\_window7\_mstrain\_480-800\_adamw\_1x\_coco.py ./mask\_rcnn\_swin\_tiny\_patch4\_window7\_1x.pth --eval bbox segm

结果

{"mode": "train", "epoch": 12, "iter": 7000, "lr": 0.0, "memory": 7909, "data\_time": 0.0287, "loss\_rpn\_cls": 0.02407, "loss\_rpn\_bbox": 0.03625, "loss\_cls": 0.16821, "acc": 93.70386, "loss\_bbox": 0.22095, "loss\_mask": 0.22871, "loss": 0.67819, "time": 0.42421}

{"mode": "train", "epoch": 12, "iter": 7050, "lr": 0.0, "memory": 7909, "data\_time": 0.02878, "loss\_rpn\_cls": 0.02483, "loss\_rpn\_bbox": 0.0356, "loss\_cls": 0.16539, "acc": 94.01221, "loss\_bbox": 0.20966, "loss\_mask": 0.21481, "loss": 0.65029, "time": 0.42718}

{"mode": "train", "epoch": 12, "iter": 7100, "lr": 0.0, "memory": 7909, "data\_time": 0.03283, "loss\_rpn\_cls": 0.02601, "loss\_rpn\_bbox": 0.03555, "loss\_cls": 0.17287, "acc": 93.5791, "loss\_bbox": 0.22184, "loss\_mask": 0.22332, "loss": 0.6796, "time": 0.42121}

{"mode": "train", "epoch": 12, "iter": 7150, "lr": 0.0, "memory": 7909, "data\_time": 0.03052, "loss\_rpn\_cls": 0.02787, "loss\_rpn\_bbox": 0.03533, "loss\_cls": 0.16987, "acc": 93.73462, "loss\_bbox": 0.22121, "loss\_mask": 0.22283, "loss": 0.67711, "time": 0.42191}

{"mode": "train", "epoch": 12, "iter": 7200, "lr": 0.0, "memory": 7909, "data\_time": 0.02705, "loss\_rpn\_cls": 0.02502, "loss\_rpn\_bbox": 0.03233, "loss\_cls": 0.15884, "acc": 94.13232, "loss\_bbox": 0.21204, "loss\_mask": 0.21345, "loss": 0.64169, "time": 0.42012}

{"mode": "train", "epoch": 12, "iter": 7250, "lr": 0.0, "memory": 7909, "data\_time": 0.0286, "loss\_rpn\_cls": 0.02801, "loss\_rpn\_bbox": 0.03797, "loss\_cls": 0.17216, "acc": 93.65112, "loss\_bbox": 0.22962, "loss\_mask": 0.22394, "loss": 0.6917, "time": 0.43742}

{"mode": "train", "epoch": 12, "iter": 7300, "lr": 0.0, "memory": 7909, "data\_time": 0.02828, "loss\_rpn\_cls": 0.02654, "loss\_rpn\_bbox": 0.03631, "loss\_cls": 0.17673, "acc": 93.51904, "loss\_bbox": 0.22787, "loss\_mask": 0.22634, "loss": 0.6938, "time": 0.43469}

{"mode": "val", "epoch": 12, "iter": 7330, "lr": 0.0, "bbox\_mAP": 0.437, "bbox\_mAP\_50": 0.666, "bbox\_mAP\_75": 0.477, "bbox\_mAP\_s": 0.285, "bbox\_mAP\_m": 0.47, "bbox\_mAP\_l": 0.573, "bbox\_mAP\_copypaste": "0.437 0.666 0.477 0.285 0.470 0.573", "segm\_mAP": 0.398, "segm\_mAP\_50": 0.633, "segm\_mAP\_75": 0.427, "segm\_mAP\_s": 0.242, "segm\_mAP\_m": 0.431, "segm\_mAP\_l": 0.546, "segm\_mAP\_copypaste": "0.398 0.633 0.427 0.242 0.431 0.546"}