HW3 Instance Segmentation

Selected Topics in Visual Recognition using Deep Learning

GitHub link

https://github.com/samuelyutt/Selected-Topics-in-Visual-Recognition-using-Deep-Learning-course/tree/hw3/hw3-InstanceSegmentation

Utilities and reference

- Swin Transformer: https://github.com/microsoft/Swin-Transformer
- Swin Transformer Object Detection: https://github.com/SwinTransformer/Swin-Transformer-Object-Detection
- mmdection: https://github.com/open-mmlab/mmdetection

Introduction

Swin Transformer is based on mmdection, and was initially described in arxiv. It is capable of serving as a general purpose backbone for computer vision. To address the problem in adapting a transformer between language and vision, Swin Transformer utilizes a hierarchical transformer. The representation is computed with shifted windows, and the name Swin comes from Shifted WINdow. The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection.

Methodology

Prepare.py

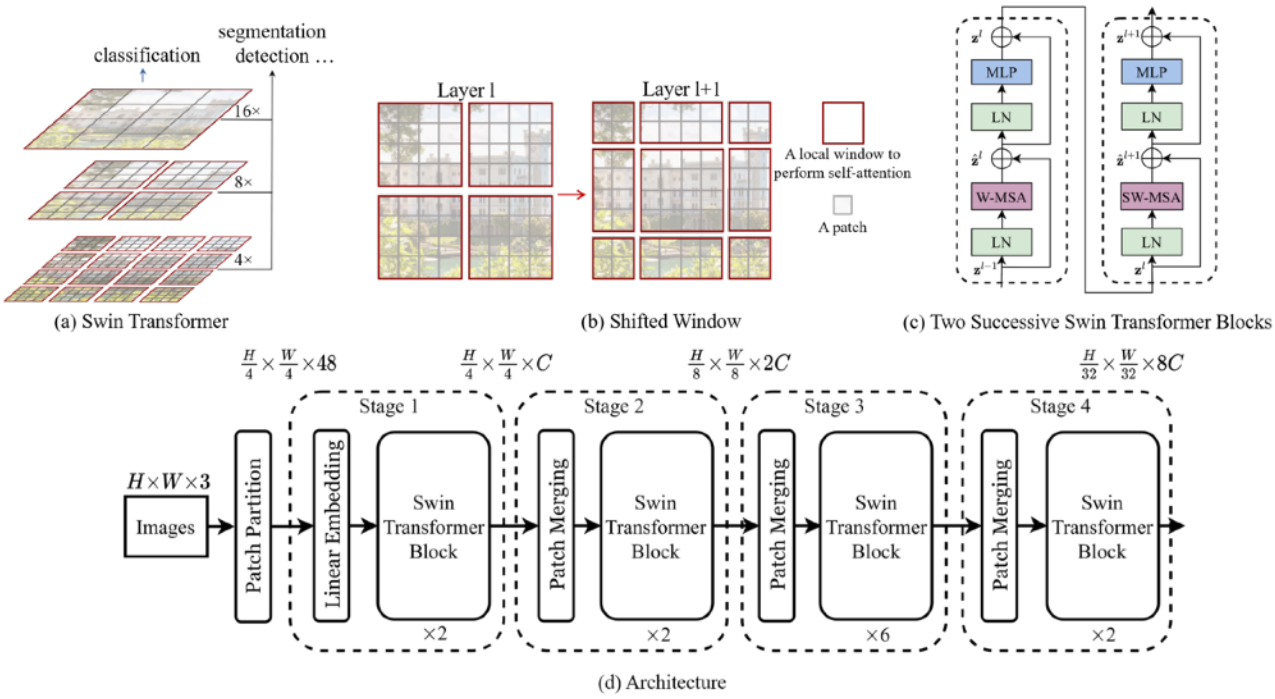
Prepare.py in src/ assumes that the train and test datasets are stored in datasets/. After executing prepare.py, train datasets will be separated into 2 parts, for training and validating purpose. Also, the mask will be parsed and transform into json format files.

Data pre-process

In the input configuration of Swin Transform, random flipped is used with a flip ratio at 0.5. Additionally, data augmentation will be applied by resizing and random cropping. Finally, the augmented inputs will be normalized.

Model architecture

Referring to the paper, there are mainly 4 stages in the Swin Transform architecture. Each contains one Swin Transform block. An overview of the Swin Transformer architecture is presented in Figure (d).



(a) Swin Transformer (b) Shifted Window (c) Two Successive Swin Transformer Blocks

(d) Architecture

Swin Transformer block

A Swin Transformer block consists of a shifted window based MSA module, followed by a 2-layer MLP with GELU non-linearity in between. A LayerNorm layer is applied before each MSA module and each MLP, and a residual connection is applied after each module. It is illustrated as in Figure (c).

Hyperparameters

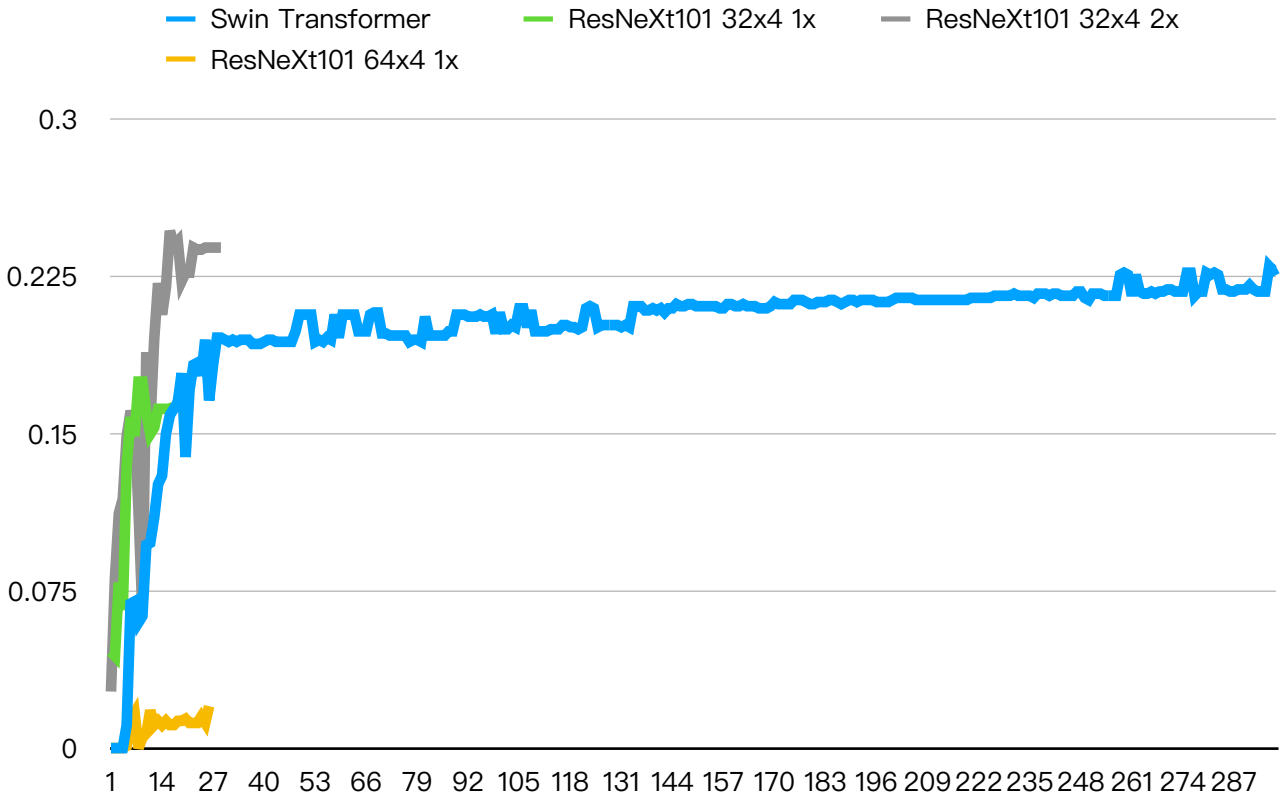
Hyperparameters such as num_classes and max_epochs are written in configs/ such as nuclei/mask_rcnn_swin_s.py.

Swin Transformer provides a lot of default values and backbones. However, some of them did not fit the datasets. Therefore, we have to create our custom configurations and to override some inherited base configurations.

Checkpoint of weights are stored in work_dirs/ during each epoch. Meanwhile, a validation test will be performed as well.

Results

Table of experiment results



The above is the mean average precision (mAP) on validating data of each epoch during training. Only Swin Transformer can train up to about 300 epochs. All of the other models will fail during training.

Analyze your result

All methods except for Swin Transformer seem to allocate too much space on the GPU and thus the training process is aborted due to out of memory.

In order to prevent CUDA out of memory problem, we will need to set gpu_assign_thr to 1. Additionally, we also need to set with_cp to True if ResNet backbone is utilized.

However, I still encounter a lot of memory problem even though the above methods are applied. As a result, I chose to use Swin Transformer models as my final model since it is more stable to use.

Summary

In this homework, I have learned to train custom datasets using Swin Transform based on mmdetection to perform instance segmentation. I have trained mmdetection model of ResNet50 and ResNeXt101 backbone, as well as small Swin transformer model. All of their performance did not meet my expectation. Last but not least, comparing to YOLOv5 I used in the previous homework, I personally don't consider mmdetection as a friendly tool since I encountered lots of environmental issue and out of memory error and had to modify the configurations to solve the problems.