National University of Sciences and Technology
School of Electrical Engineering and Computer Science
Department of Computing

CS893: Advanced Computer Vision Spring 2022

Semester Project

Instance Segmentation

Instructor: Dr. Muhammad Moazam Fraz

Submitted by: Muhammad Salman Akhtar
Registration No: 398895

1 Introduction

Segmentation is a task of segregating the objects from the background and holds an important place in computer vision domain. While semantic segmentation is aimed at classifying each pixel in an image to its corresponding class, instance segmentation goes a step further and also identifies the individual instance of an object. It can perhaps be looked as a combination of object detection (location and classification) and semantic segmentation (pixel-wise identification). Thus, instance segmentation can be regarded as one of the most challenging tasks in computer vision. Instance segmentation outputs a more rich and dense output as compared to both object detection and semantic segmentation. Instance segmentation finds its usage in diverse set of fields such as enabling autonomous vehicles to understand their environment, aerial crop monitoring and augmenting medical image diagnostics (X-ray and MRI scans). This project also deals with the development of deep learning-based instance segmentation algorithm for isolating nuclei in medical images.

2 Dataset

The dataset is composed of two separate datasets. The first one is known as CoNSeP (Colorectal Nuclear Segmentation and Phenotypes) introduced in HoVer-Net (1). It consists of 41 images divided into 27 for training and remaining 14 for testing purpose. Each image is of 1000x1000 dimension which can be regarded as a high-resolution image. Moreover, most of the images exhibit a very high density of nuclei with highest value of 2282, mean 577 and lowest 21. The high density of nuclei posses two-fold challenge: one being difficult to isolate the bunched together nuclei instance while the second one is related to the computational complexity which requires one mask for each instance of the object. Thus, this dataset offers a significant challenge with regards to instance segmentation. Figure 1 shows the sample picture from this dataset.

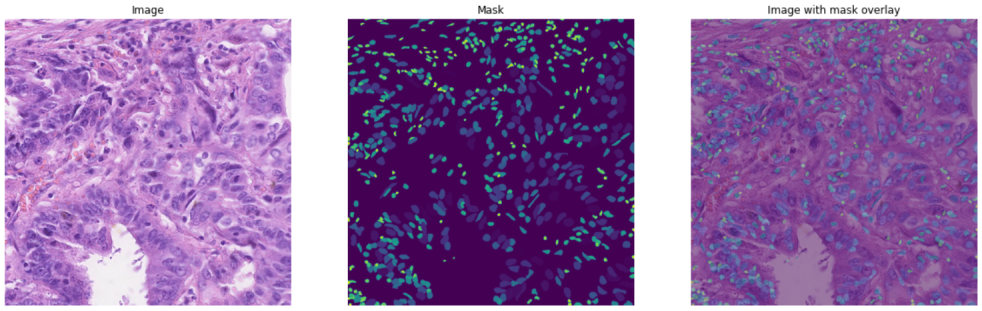

Figure 1: CoNSeP dataset sample

The second dataset is known as CPM17. It consists of 64 images divided into 32 for training and 32 for testing purpose. Few of the images are of are of size 600x600 while others have dimensions of 500x500. This dataset is relatively less dense and contains highest value of nuclei to be 283, mean of 106 and lowest value of 38. This dataset is relatively less challenging as compared to CoNSeP dataset. Figure 2 shows the sample picture from this dataset.

3 Methods for Instance Segmentation

Since, the dawn of deep learning era, a number of approaches have been devised to take up the task of instance segmentation. Table 1 groups the algorithms commonly used for this task.

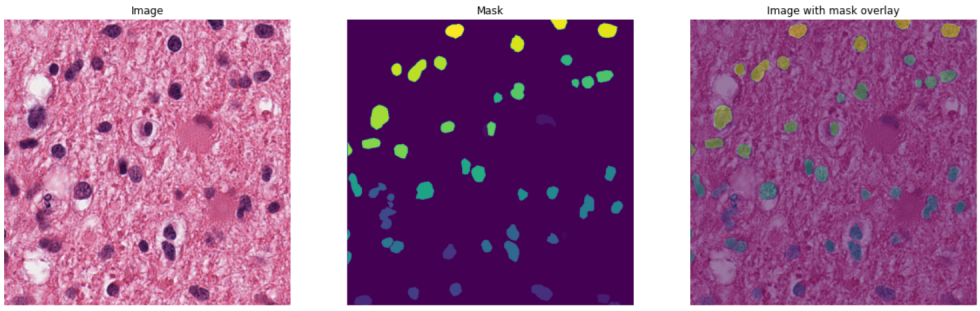

Figure 2: CPM17 dataset sample

Table 1: Instance Segmentation Methods

Group	Technique
Classification of mask proposals	RCNN, Fast RCNN, Faster RCNN
Detection followed by segmentation	HTC, PANet, Mask RCNN, Mask Scoring RCNN, MPN, YOLACT
Labelling pixels followed by clustering	Deep Watershed Transform, Instance Cut
Dense sliding window methods	Deep Mask, Instance FCN, Tensor Mask

4 Model Architecture

For this project, I have adopted Mask RCNN which is placed into detection followed by segmentation group. I used two variations of Mask RCNN which are explained in the following paragraphs.

4.1 Architecture 1

The first architecture is composed of Feature Pyramid Network (FPN) with a backbone of ResNet 50. The output of FPN at all 5 scales are fed to the Region Proposal Network (RPN) which identifies the possible regions (nuclei) and based on these proposals, ROI Align layer extracts these regions from feature maps and passed it to separate three network heads. The upper two heads are for object detection and composed of a classifier and a regressor. While the bottom head is the mask head which is used for instance segmentation. Figure 3 shows the architecture for this model.

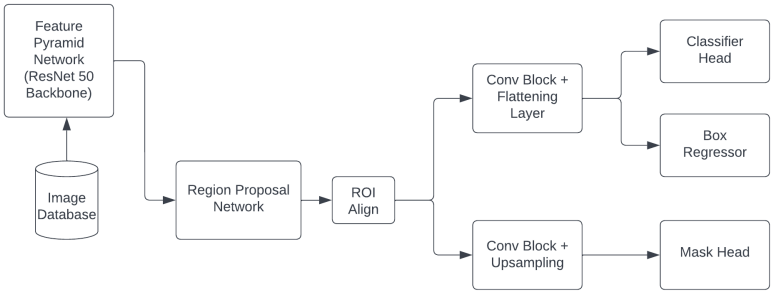

Figure 3: Architecture 1 (Resnet50 backbone)

4.2 Architecture 2

In architecture 2, the Feature Pyramid Network (FPN) was changed. First of all, the backbone has been upgraded to ResNet 101. Moreover, at the bottleneck Atrous Spatial Pyramid Pooling (ASPP) has been added. The block is composed of three parallel atrous convolution layers with different rates (6, 12 and 18) along with a simple convolution layer. All these layers are concatenated and after more convolutions passed the decoder part of the network. The ASPP block has been added to make the model more capable of handling multi-scale objects. Moreover, the simple up-sampling layers of decoder have been replaced by transposed convolutions. Unlike, up-sampling layers the transposed convolution layers have learnable parameters and considered to give better results. The rest of the blocks of the model remain same as with architecture one. The modified FPN for architecture2 has been shown in figure 4.

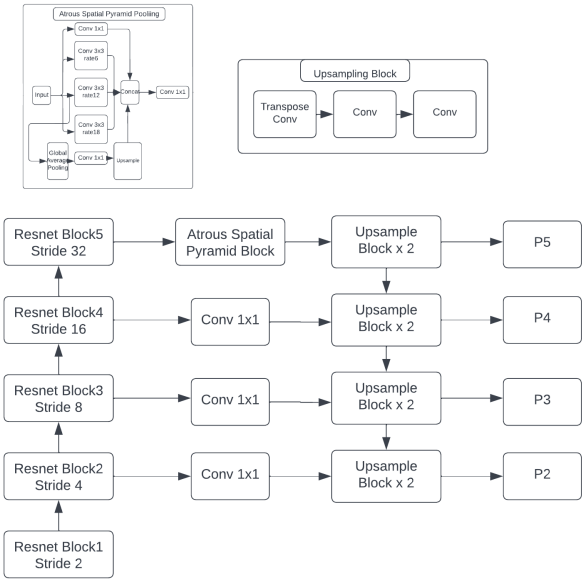

Figure 4: Modified FPN

5 Model Training

Both models are trained on Google Colab platform on allotted GPU.

5.1 Data Augmentation

A data generator from (2) has been repurposed by incorporating requisite changes for this dataset. It is capable of generating dataset batches (images, masks) of desired sizes. Moreover, this dataset generator is also capable of image augmentation on the fly. For image augmentation 'imgaug' library has been used that incorporated random rotation, flip, shifting and brightness variation. Table 2 summarizes the augmentation operations that were applied to the training dataset.

Table 2: Augmentation Parameters

Rotation	Shift	Vertical Flip)	Brightness	Horizontal Flip
90, 270	0.8 to 1.2	0.5	0.8 to 1.2	0.5

5.2 Loss Function

Five different losses have been used to calculate the cumulative loss. All losses have been given the same weightage and defined by equation 1. The rpn_boxloss and box_regressor_loss are the calculated using the absolute difference between ground truth and predictor. For calculating rpn_class_loss, classfier_head_loss and mask_loss binary cross entropy loss has been used.

$$loss = rpn\_class\_loss + rpn\_box\_loss + classfier\_head\_loss + box\_regressor\_loss + mask\_loss \quad (1)$$

6 Transfer Learning

As pointed out earlier resnet50 and resnet101 backbones have been used for Feature Pyramid Network. For both these models, pretrained weights on "ImageNet" dataset have been used (3). Moreover, the models were then trained on a nuclei dataset from 2018 Data Science Bowl link for 16 epochs. This dataset contains 657 images of various sizes along with the ground masks of nuclei instances. However, the nuclei density for this dataset is significantly less as compare to the datasets of this project.

7 Challenges

The dataset is composed of high-density objects (nuclei) and therefore poses a computational challenge. The cpm17 dataset was relatively easily trained since its density is significantly lower than the CoNSeP dataset. The CoNSeP dataset is composed of 1000x1000 dimension with some images containing very high-density regions of nuclei. The first attempt of training was failed as Colab environment crashed due to insufficient resources. As a way forward random crop of sizes 512x512 and then 256x256 was tried but also failed with crash. Then a compromised scheme was devised to tackle this situation. Each image was divided into 16 smaller images of size 250x250 thus resulting into 27x16 = 432 images. From these 432 images, only those images are retained which contain less than 700 nuclei instances and remaining were discarded. This resulted into 257 images which could be used to train the model in Colab environment.

Table 3 shows the training parameters while figure 5 and figure 6 depicts the training graphs for architecture 1 and architecture 2 respectively.

8 How to run the code

The models have been implemented using python programming language in Jupyter notebook and corresponding "MaskRCNNv1.ipynb" can be accessed from github link. It is pertinent to mention here that Mask R-CNN for object detection and instance segmentation (Matterport) (2) has been extensively used and tailored for this implementation. However, there are some changes incorporated in the original implementation such as modification of Feature Pyramid Network and dataset generator module. Therefore, this notebook requires "MaskRCNNv1.ipynb" requires the following python files for execution:-

Table 3: Training Parameters

Model	Batch Size	No of Epochs (Nuclei Dataset)	No of Epochs (CPM17)	No of Epochs (CoNSeP)	Learning Rate	Training timen
Architecture1	1	16	20	20	0.001 and reduced by a factor of 0.5	06 hours
Architecture2	1	16	24	14	0.001 and reduced by a factor of 0.5	06 hours

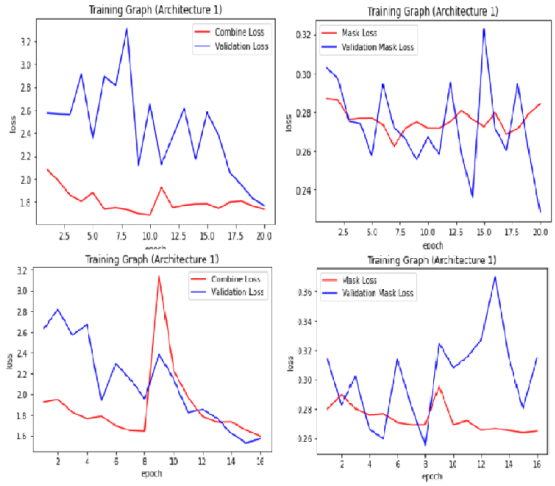

Figure 5: Training Graph (Architecture 1)

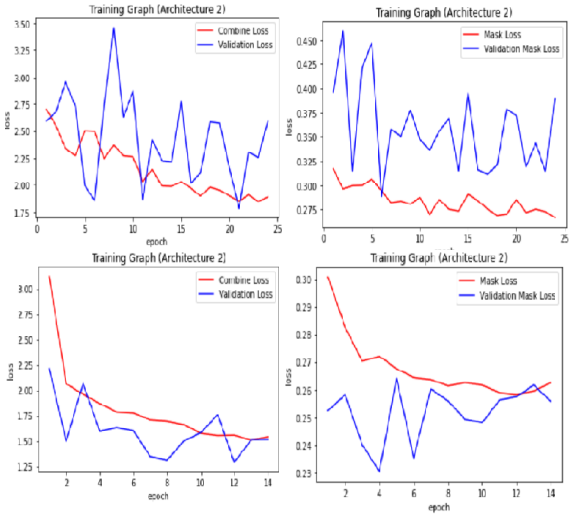

Figure 6: Training Graph (Architecture 2)

1. model.py (Modified)
2. config.py

3. customConfig.py (Implemented for dataset reading and model configuration)
4. utils.py
5. visualize.py

All these files can be accessed from the github link. The Mask RCNN implementation was carried out on mostly obsolete versions of python modules. Therefore, for smooth execution of code specific versions of modules are required to be installed (tensorflow-gpu==1.15.2, keras==2.2.5, h5py==2.10.0, scikit-image==0.16.2).

The metrics functions (dice coefficient, Aggregated Jaccard Score and Panoptic Quality) have been repurposed from (1). During programming, it was tried to follow a modular and object-oriented approach. The code expects that extracted dataset folder is also present in the same directory along with the code file. All the cells should be run sequentially from top to bottom as per instructions in the notebook. The notebook also includes the supporting functions that are required for training and testing the model.

9 Results and Discussion

Both models were evaluated based on number of metrics for both datasets. Table 4 shows the comparison of architecture 1 and architecture 2 for CPM17 dataset while table 5 shows the comparison for CoNSeP dataset.

Table 4: Result Comparison (CPM17)

Metric	Architecture 1	Architecture 2
mAP	0.2920	0.2834
Dice Coefficient	0.6686	0.5975
Aggregated Jaccard Index	0.4623	0.4004
Panoptic Quality	0.5096	0.4838
Detection Quality	0.6803	0.6245
Segmentation Quality	0.7452	0.7718

Table 5: Result Comparison (ConSeP)

Metric	Architecture 1	Architecture 2
mAP	0.2563	0.2592
Dice Coefficient	0.368	0.4504
Aggregated Jaccard Index	0.202	0.27
Panoptic Quality	0.1810	0.2446
Detection Quality	0.2458	0.3329
Segmentation Quality	0.6287	0.6414

Architecture 1 performed slightly better than architecture 2 on all the metrics except for segmentation quality on CPM17 dataset. However, on CoNSeP dataset architecture 2 outperformed architecture 1 with a significant margin on all metrics. This was expected since architecture 2 has a deeper and modified backbone with Atrous Spatial Pyramid Pooling layer. Moreover, its feature pyramid network also contains transposed convolution layers as compare to simple up-sampling layers.

Overall, both architectures performed better on CPM17 dataset as compared to CoNSeP dataset. The CoNCeP seems to be significantly challenging. Moreover, a number of images with densely populated nuclei could not be used for training due to computational limitations. This could the reason of poor performance of models on this dataset.

The qualitative results also portrays a similar picture. Figure 7 and figure 7 shows a sample detection response by architecture 1 on CPM17 and CoNSeP datasets. Similarly, figure 9 and figure 10 shows the sample detection response by architecture 2 on CPM17 and CoNSeP datasets.

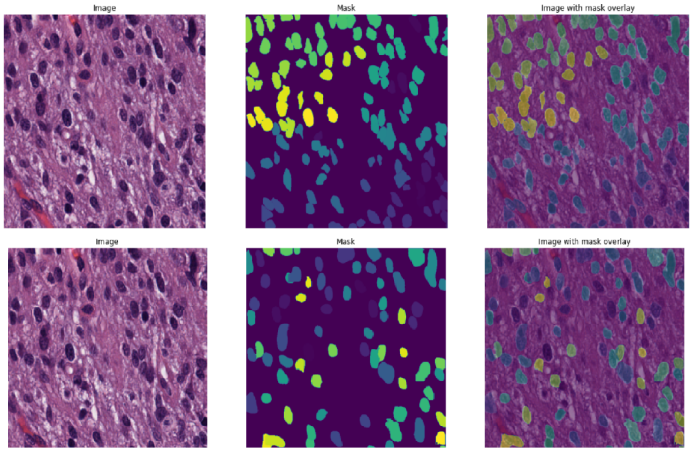

Figure 7: Prediction result (Architecture1 CPM17)

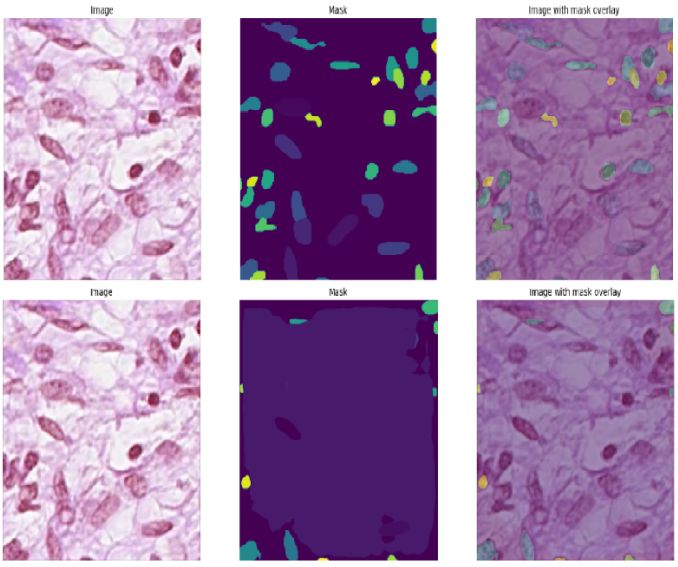

Figure 8: Prediction result (Architecture1 CoNSeP)

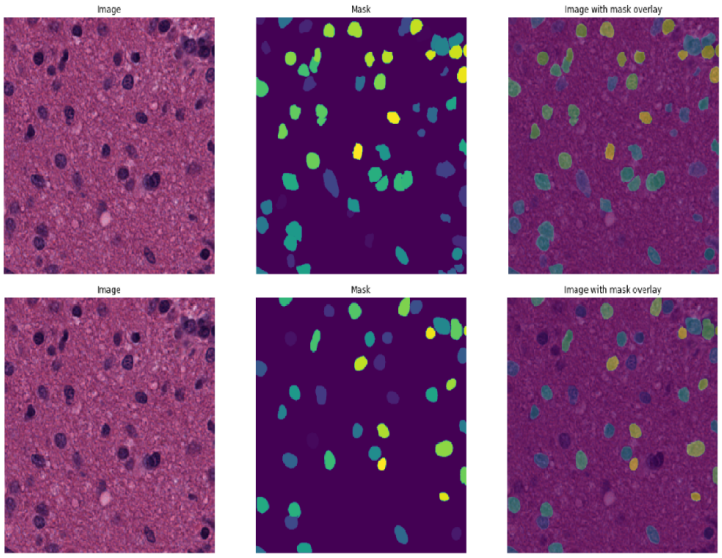

Figure 9: Prediction result (Architecture2 CPM17)

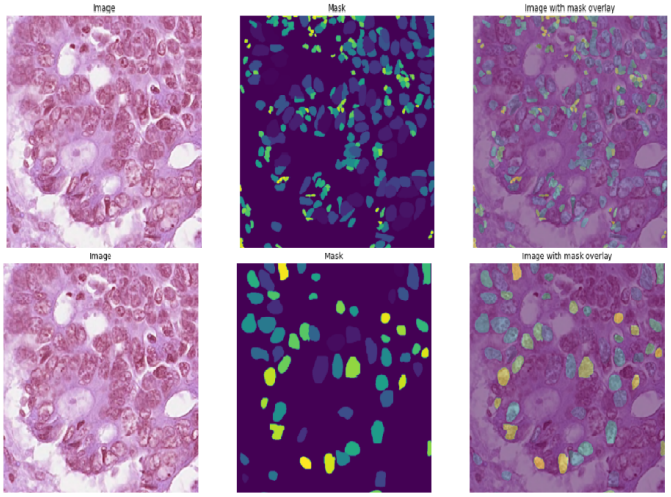

Figure 10: Prediction result (Architecture2 ConSeP)

10 Conclusion

In this project, two separate architectures of Mask RCNN (FPN with resnet50 and modified FPN with resnet101) pre-trained on imagenet dataset were implemented. The training for both models was augmented with additional nuclei dataset along with the provided datasets of CPM17 and CoNSeP. Training a densely populated nuclei dataset is significantly challenging with respect to both computational complexity as well as accurate segregation of closely placed nuclei. In order to overcome the computational bottleneck, CoNSeP dataset has been preprocessed to make it suitable for training on Colab environment. Both architectures performed better on CPM17 dataset, however, architecture 2 performed significantly better on CoNSeP dataset. The results were far from ideal and suggests that a lot of improvement would be required before it can be practically used for any medical diagnostic purpose.

References

[1] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot, "Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images," *Medical Image Analysis*, p. 101563, 2019.

[2] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[3] F. Chollet *et al.*, "Keras," https://keras.io, 2015.