Semantic segmentation of cells from clinical images using Deep learning

Project 9. Segmentation of cell images.

Sameer Agarwal (S192274), Mikkel Lehmann Nielsen (S153048), Nicklas Leander Lund (S145129)

Supervised by Peter Jensen, CTO of Cellari.io

02456 Deep learning, DTU Compute, Technical University of Denmark

Introduction and summary

Introduction

From basic biological questions and diagnosis to clinical trials, cell morphology provides key quantitative feedback that drive research. Unfortunately, this is most commonly a manual task and can be time intensive and prone to human error. The task is made even more difficult due to overlapping cells and poor imaging quality among other factors.

One such task is the accurate segmentation of glands to obtain reliable morphological statistics for cancer research. A deep learning approach comparing two state of the art Convolutional Neural Network architectures – U-Net [2] (2015) and Mask-RCNN [3] (2018) is demonstrated to automate this process of segmenting glands in a cell image.

Summary

- Build a memory efficient batch generator from scratch
- Build a data augmenter to expand the training set to a given size and make the model more robust
- Implemented CNN architectures viz. U-Net and Mask-RCNN for solving the pixelwise image segmentation problem
- Benchmarked and compared the results with other implementations against the same problem across different measurement metrics

Data & preprocessing

Data

Data from the "gland segmentation in histology images" challenge 2015 [6] is used. The data has images of Hematoxylin and Eosin (H&E) stained slides, consisting of a variety of histologic grades. The dataset is provided together with ground truth annotations by expert pathologists. It is used in the detection of Colorectal cancer.

There are 85 images in the training data set along with the ground truth masks as well as 80 images for testing, split into A (60) and B (20). A sample image can be seen in the next section.

Preprocessing

Data Augmentation is used to make the model more robust and invariant to the change in location and shapes of cells as well as also to expand the dataset as we only have 80 training samples.

A composite pipeline of vertical flip, horizontal flip and random cropping, each with a probability of 50 percent is created for augmentation. All images are reshaped to the size 512x512. Below is an example of an augmented image and the corresponding mask.

Figure 1: Data Augmentation

Batch generator

While working with image datasets, efficient use of memory is always a challenge. Hence, designing an efficient batch generator became significantly important to be able to effectively train the model.

The batch generator is designed such that it only loads and augments one batch at a time, instead of saving all images in memory simultaneously. This significantly reduces memory load and allow us to train larger batches.

Model 1: U-Net[2]

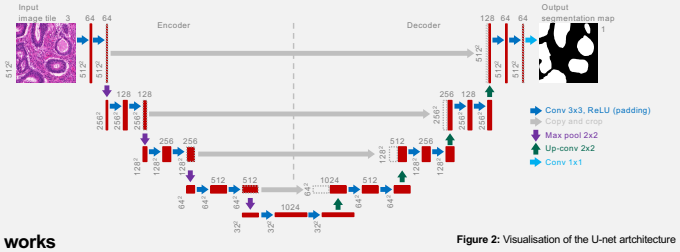

Figure 2: Visualisation of the U-net artchitecture

How it works

The original goal of the U-net is to efficiently learn pixel-wise classification on only a few samples. This makes it a good choice in this project, as both data and compute is limited. The model works as follows: In the encoder part, the cell-image is down-sampled. Here the network learns context and features at multiple levels, essentially learning how the cells look. In the decoder part, the cell-features are up-sampled. Here the network learns how to map the low-resolution features to a high-resolution pixel-space. It does so by up-sampling and concatenating previous layers across, and in the end, making a binary classification of each pixel as either being a cell or background. The result is a model that can find cells in images.

Tuning and training

To tune the model, first a broad random search was run, then, based on the best results, a narrower random search was run. Then, the best model was chosen and trained. After training, the new model was used as foundation for further training as would be done with transfer-learning. The tuning technique chosen here was babysitting. The validation loss went from ~0.69 to ~0.42 to ~0.29 following these steps.

Design choices and parameters

Vanilla network with no dropout. Zero padding to preserve image size. SGD with high learning rate (0.1) and L2 regularization (1e-5) initially for 900 epochs, then Adam with low learning rate (0.00001) and L2 regularization (0.001) for 50 epochs. Both using BCELoss as criterion and batch-size of 6. Compared with the original U-net, no post-processing was performed, and our model is, in many ways, simpler.

Sample results

Figure 3: Sample results from U-net on test set B. Showing Best, worst and average prediction

Model 2: MaskRCNN[3]

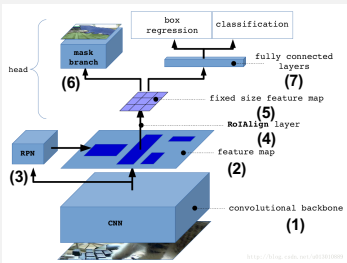

Figure 4: Visualization[4] of the MaskRCNN architecture.

How it works

The purpose of the MaskRCNN is to perform object instance segmentaion which is the task of simultaneous object- and semantic segmentation. The model has shown good performance in more complicated tasks like instance segmentation and it is therefore expected that the model will perform well in this project. The model is by construction flexible and easy to generalize to new types of tasks, for instance semantic segmentation of cells as will be done in this project.

Referring to figure 4, in **(1)** A fully convolutional network learns features of the image on different levels of abstraction which is outputted as a feature map in **(2)**. In **(3)** a region proposal network learns where the areas of interests are on the feature map **(2)**. In **(4)** a region of interest align layer then extracts a fixed-size feature map, from each region of interest on the feature map. Essentially only learning cell-related features and not learning noise. The regions of interest are then passed to a mask branch, **(6)**, and a fully convolutional network, **(7)**. In **(6)**, pixel-wise predictions are made to construct a mask. In **(7)**, bounding boxes are predicted, and classification of instances is performed.

Design Choises and parameters

The MaskRCNN is implemented via the PyTorch sub-package torchvision.models. The CNN backbone is a ResNet50[7] pre-trained on the COCO-dataset[8] and the size of the hidden convolution layer for the mask branch is 256. SGD with momentum at 0.9, a learning rate of 5e-3 and small L2 reguralization of 5e-4 was found to provide the best results. Additionally a small batch size of 2 was used and the model was trained for 500 epochs.

Tuning and training

The model was tuned and trained by babysitting. Starting from the values of the implementation in the PyTorch example sensible experimentations were done trying to lower the BCEWithLogitLoss of the predicted mask from the model a resulting validation loss of ~0.54 was obtained for the best model.

Sample results

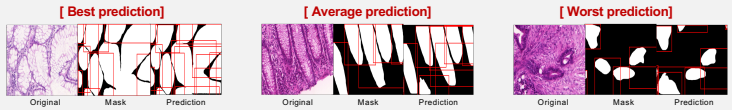

Figure 5: Sample results from MaskRCNN on test set A. Showing Best, worst and average prediction

Benchmarks[1]

Method	F1 Score				Object dice				Object Hausdorff				Sum of ranks
	Part A		Part B		Part A		Part B		Part A		Part B		
	Score	Rank	Score	Rank	Score	Rank	Score	Rank	Score	Rank	Score	Rank	
DTU-MASKRCNN	0.922	1	0.906	1	0.905	1	0.872	1	37.529	1	74.287	1	6
CUMedVision2	0.912	2	0.716	5	0.897	2	0.781	7	45.418	2	160.347	8	26
ExB1	0.891	5	0.703	6	0.882	5	0.786	4	57.413	7	145.575	3	30
ExB3	0.896	3	0.719	4	0.886	3	0.765	8	57.350	6	159.873	7	31
Freiburg2	0.870	6	0.695	7	0.876	6	0.786	5	57.093	4	148.463	5	33
CUMedVision1	0.868	7	0.769	3	0.867	8	0.800	2	74.596	8	153.646	6	34
DTU-UNET	0.822	9	0.862	2	0.723	11	0.791	3	126.685	11	138.294	2	38
ExB2	0.892	4	0.686	8	0.884	4	0.754	9	54.785	3	187.442	10	38
Freiburg1	0.834	8	0.605	9	0.875	7	0.783	6	57.194	5	146.607	4	39
LIB	0.777	10	0.306	12	0.781	9	0.617	11	112.706	10	190.447	11	63
CVML	0.652	11	0.541	10	0.644	12	0.654	10	155.433	12	176.244	9	64
vision4GlaS	0.635	12	0.527	11	0.737	10	0.610	12	107.491	9	210.105	12	66

DTU-UNET: *TestLossA: 0.403 TestLossB: 0.380 CountA: 0.211 CountB: 0.127*
DTU-MASKRCNN *TestLossA: 0.540 TestLossB: 0.540 CountA: 0.825 CountB: 0.842*

Results and conclusion:

The benchmark table contains results from the GlaS contest[1] which ran officially in 2015, around the same time when the U-Net was first proposed. Our implementation of the U-Net is comparable to the peers of its time, finishing on a 7th place out of 12 models.

Even though we extensively tuned and trained the U-Net, whilst only doing a limited amount of babysitting on the MaskRCNN, the MaskRCNN still outperforms all previous models on all categories of the benchmark. Investigating the sample results from figure 3 and 5, similar conclusions can be made. While the U-Net performs good on the best prediction, it lacks robustness and performs very poorly on others. The MaskRCNN however, almost makes perfect predictions, even on its worst samples, and is therefor very robust in its generalization. This is because it reduces the multi-object problem, to single-object problems by using the RPN.

The MaskRCNN, was first proposed in 2018[3] and shows how much the architecture for semantic segmentation have improved in just three years. Therefore, when considering semantic segmentation of cells from clinical images, the results just 5 years ago would not be robust enough for real use, however, today the results seems so robust, that real use-cases are realistic if implementing state-of-the-art models, such as the MaskRCNN.

References: *(1): https://arxiv.org/abs/1603.00275 (2) https://arxiv.org/abs/1505.04597 (3) https://arxiv.org/pdf/1703.06870.pdf*
(4) https://ronjian.github.io/blog/2018/05/16/Understand-Mask-RCNN (5) https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html (6) https://warwick.ac.uk/fac/sci/dcs/research/tia/glascontest/ (7) https://arxiv.org/pdf/1512.03385.pdf (8) https://cocodataset.org/#home