NTNU
Norwegian University of
Science and Technology

Detecting and Localizing Cell Nuclei in Medical Images.

Johan Scott Loudon

Master of Science in Computer Science
Submission date: July 2018
Supervisor: Kerstin Bach, IDI

Norwegian University of Science and Technology
Department of Computer Science

This master thesis is dedicated to the fantastic free education system of Norway and everyone who has fought to make and keep it that way, which has allowed me to get a good degree of almost completely my own choice practically free of charge.

Summary

In this master thesis we have adapted and implemented Mask R-CNN (He et al., 2017) to the task of detecting and localizing nuclei in medical imaging. Mask R-CNN, which does instance segmentation, was chosen as the architecture to implement, based on a literature review. Our best Mask R-CNN model achieved a F1-score of 0.385 on the validation set and 0.460 on the test set. This thesis was inspired by and a part of the Kaggle 2018 Data Science Bowl[1]. We did not complete our implementation in time to enter the competition, but if we had, this score would have lead to 291st place out of 738 participating individuals or teams in the Kaggle competition. The differences between our implementation and the second-placed implementation, which also implemented Mask R-CNN, were mainly a different backbone and heavy data augmentation. This shows that our approach was competitive, and with modifications could have been competing for the top placements in the competition. The code is available at: `https://github.com/jolohan/detectron2.git`

[1]`https://www.kaggle.com/c/data-science-bowl-2018`

Sammendrag

I denne masteroppgaven har vi adaptert og implementert Mask R-CNN (He et al., 2017) for å detektere og lokalisere cellekjerner på medisinske bilder. Vi valgte å implementere Mask R-CNN, som utfører forekomst-segmentering, på bakgrunn av en litteraturstudie på dette feltet. Vår beste Mask R-CNN modell oppnådde en F1-score på 0.385 på valideringssettet og 0.460 på testsettet. Denne oppgaven ble inspirert av og var en del av Kaggle 2018 Data Science Bowl[2]. Vi fullførte ikke implementasjonen vår i tide til å bli med i konkurransen, men hvis vi hadde det, ville vår score ha holdt til 291. plass ut av 738 deltakende individer og lag. Forskjellene mellom vår løsning og 2. plass' løsning, som også implementerte Mask R-CNN, var hovedsakelig en annen og dypere modell av Mask R-CNN, og mye dataforbedring. Dette viser at vår tilnærming var konkurransedyktig, og med modifikasjoner kunne ha konkurrert om de øverste plassene i denne konkurransen. Koden er tilgjengelig på: `https://github.com/jolohan/detectron2.git`

[2]`https://www.kaggle.com/c/data-science-bowl-2018`

Preface

This thesis was written in the spring semester of 2018 for the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The subject for this thesis was defined in cooperation with my supervisor Kerstin Bach. I would like to thank my supervisor Kerstin Bach for giving very helpful feedback and assistance.

Table of Contents

List of Tables

List of Figures

Abbreviations

RoI	Region of Interest
IoU	Intersection over Union
CS	Compressed Sensing or Computer Science
CNN	Convolutional Neural Network
ENet	Efficient Neural Network
MTL	Multi-Task Learning
WSI	Whole-Slide Images
FCN	Fully Convolutional Network
HNM	Hard Negative Mining
CAD	Computer-Aided Diganosis
DCAN	Deep Countor-Aware Network
HoG	Histogram of oriented Gradients
WBCs	White Blood Cells
PGSA	Plant Growth Simulation Algorithm
TP	True Positive
FP	False Positive
FN	False Negative
RRC	Rolling Recurrent Convolution
RPN	Region Proposal Network

Chapter 1

Introduction

1.1 Motivation

Detecting and localizing cell nuclei in medical images is an important tool for diagnosing many diseases like heart disease, cancer and diabetes, to name a few. This work can be time consuming and difficult to do accurately. It would therefore be beneficial to automate this process, to reduce workload and increase accuracy.

Visual computing has made big leaps forward in recent years inspired by machine learning and convolutional networks. The more traditional approach of extracting features and feeding them into a classifier has come up short when faced with complex shapes, colours and backgrounds. With the advancement of different architectures based on CNNs, these problems no longer require the same amount of expertise in selecting which features to extract. CNNs implicitly encode patterns, shapes and other features and reduce the need for selecting features.

Due to this, the task of automatically locating nuclei is one of many tasks that now have become suited for machine learning. This is why this spring, the consulting company Booz Allen Hamiltion as organizers in cooperation with the host Kaggle, have made this the task of the Kaggle 2018 Data Science Bowl[1]. Arranging a competition for solving this problem provides Kaggle with many different solutions. Crowdsourcing the problem and offering prizes to the best solution(s) can often lead to state-of-the-art solutions. An open dataset lets people and teams from all over the world compete and try to design the best solution. In addition, their discussion forum is a helpful resource which facilitates sharing of advice and ideas.

1.2 Research questions

The overall goal is to implement a competitive system for localizing nuclei in medical imaging. This problem can often be very sparse. The target nuclei often occupy a small

[1] https://www.kaggle.com/c/data-science-bowl-2018

part of the image. This leads to class imbalance between the foreground and background of the images that contain the content to be classified. This overall goal will lead the literature search. The state-of-the-art method(s) from the literature search will be adapted and applied on solving the nuclei localization problem. The results from this implementation will be evaluated and discussed. Our implementation and its results will then be compared to the best implementations in the Kaggle competition.

In order to achieve the overall goal the following four research questions were formulated:

1. What are the state-of-the-art methods for detecting and localizing nuclei in medical imaging?
2. How can the state-of-the-art methods be applied on the task of localizing nuclei?
3. How will the chosen method perform in the described task?
4. How will the chosen method compare to the best implementations in *Kaggle's Data Science Bowl 2018: Localizing nuclei in medical imaging*?

1.3 Research methods

To find the state-of-the-art methods we have conducted a literature search which is described in detail in chapter 2. After finding the state-of-the-art methods we have modified and implemented them. Then, we have evaluated their performance in respect to a set of metrics that are described in section 4.2.

1.4 Thesis structure

Chapter 2 will describe our literature search and its results. Chapter 3 will contain some background theory in addition to a description of the data and implementation, before we report the evaluation and results in chapter 4. Thereafter the results will be discussed in chapter 5. And in the end we will summarize the thesis and describe future work that can be done in chapter 6.

Chapter 2

Literature Review

2.1 Search methodology

To find the state-of-the-art method(s) for detecting and localizing nuclei in medical imaging we have conducted a semi-structured literature review. To conduct this review we first had to come up with search terms to find papers that are relevant to the research topic of this thesis. The search terms were found by googling the problem and picking the most relevant terms in consideration to both recall and precision. The search term suggestions were split into three categories in table 2.1: problem-related; medicine and the two method-related categories; computer vision and artificial intelligence.

The search results in the next section had to be filtered for non-relevant papers. To filter efficiently we read the title, evaluated if it was about detection, segmentation or localization of 2D images. Many papers were filtered out because of their focus on classification or 3D images. If it was not filtered out by the title, we skimmed the paper for pictures of what kind of images they had worked with and read the abstract to find out if it dealt with instance segmentation of small objects. Many papers do not specify if they focus on separating foreground instances from each other or only separate foreground from background. Others only handle large objects that occupy most of the pixel space. Also, the images in the paper often show very precisely what they are hoping to accomplish, and can be very helpful if there are many advanced medical terms in the paper. Based on this filtering we get the results in the *Relevant* column of tables 2.2, 2.3, 2.4. Classifying it as relevant, only means that based on title and abstract, it most likely can *not* be classified as *non-relevant*, not that it definitely is relevant.

In addition, we have found some papers through other sources. This has not been part of the structured search. The papers found through other sources that have been deemed relevant are three papers and have been summarized in subsection 2.2.4.

Medicine	**AI**	**Computer vision**
cellular subunits	deep learning	detection
imaging	convolution	segmentation
nuclei	ensemble	pixel space
cell	LSTM	compressed sensing
medical	image recognition	image processing
		localization

Table 2.1: Search term proposals

Table 2.2: Google scholar search results

Search input	Year		Relevant
	No filter	2016-2018	
nuclei detection imaging deep learning	48 000	19 500	N/A
nuclei detection localization imaging deep learning	32 500	16 000	N/A
”nuclei detection” ”localization” ”image recognition” ”convolutional”	33	31	8

2.1.1 Google scholar search

The first domain that was searched was Google Scholar[1]. The search terms were *nuclei, detection, imaging, deep learning, localization, image recognition, convolutional.* The results from these search terms can be found in table 2.2. Here the focus was shifted from field-related to method-related, since we want to find the state-of-the-art method. This led to the results in table 2.2. The reason the search was limited to papers from 2016 and onward is the huge steps computer vision has taken recently because of CNNs. We then read the titles or/and abstracts of every one of these to determine if they were relevant to the thesis. After this filtering, 8 reports were left. These will be summarized in the next section.

2.1.2 CVPR17 search

The most prominent and renown conference, *CVPR17* in the field of visual computing was included in this literature review. We searched for the search terms over the paper titles. Then filtering by title, and sometimes abstract and images in the paper, was applied to determine what papers could be relevant. Table 2.3 shows the number of hits and the hits that were potentially relevant. Most of the papers that went through filtering were not relevant.

[1] `https://scholar.google.com`

Table 2.3: CVPR17 search results

Search term	**Hits**	**Filter by title/abstract**
localization	20	0
cell	0	-
nuclei	0	-
detect	67	0
recognition	57	0
convolution	58	1
ensemble	2	0
LSTM	5	0
Medical	4	0
Sum	213	1

Table 2.4: AI conferences search results

Search term	**Hits**	**Filter by title/abstract**
localization	5	0
cell	0	-
nuclei	0	-
detect	38	2
recognition	36	0
convolution	23	0
ensemble	7	0
LSTM	7	0
Medical	3	0
Sum	119	2

2.1.3 AI conferences search

Two of the biggest AI conferences, *IJCAI 2017* and *AAAI 2017*, were also included in this literature review. After going through titles and abstracts of the first results, it was discovered that a large percentage of the papers were not relevant for this thesis. Therefore the vast majority of papers were only filtered by the titles to decide if they could potentially be relevant. The remaining papers were filtered using the same filtering procedure as described in section 2.1. We get similar results in table 2.4 as in table 2.3.

2.2 Summaries of literature

2.2.1 Google Scholar search

In (Khoshdeli and Parvin, 2018) the authors tackle nuclei segmentation in stained histology sections by using a combination of colour decomposition, fusion of three CNNs (ENets) and Watershed (Beucher and Meyer, 1992). Watershed is an edge detector algorithm that simulates a flooding process of the image where the intensity of a pixel represent its height. The three ENets have different tasks. The region- and boundary-ENets segment foreground/background and mark the borders of the nuclei. Then the third ENet combines the output of the two first ENets to separate joined instances of nuclei.

The same researchers show in (Khoshdeli and Parvin, 2018) that using a feature-based representation as input to a shallow CNN can be effective at detecting nuclei. The feature-based representation improves colour decomposition by using the Laplacian of Gaussian (LoG) which focuses on blob-shaped objects. Their best F1-score is achieved with the LoG of the non-negative matrix factorization (NMF) of the LoG fed as input to the shallow CNN.

In (Kainz et al., 2017) the authors attempt to segment and classify colon glands. Their method is two distinct CNNs; one for separating foreground and background and one for separating joined foreground instances. This second CNN is their big contribution. They use a quite old CNN implementation, LeNet-5 (LeCun et al., 1998). Our opinion is that replacing the LeNet-5 with a more advanced backbone would increase their accuracy.

(Xie et al., 2018) achieve better than state-of-the-art results in locating cell centroids. Their method is a fully residual CNN with structured regression that outputs a proximity map with higher values for pixels closer to centroids. It performs well partly because it combines the structured regression with a proximity map that is not down-scaled. Because of this, they do not suffer any loss of accuracy due to loss of spatial information, which they would suffer with max pooling.

Others have focused on detection and classification in stead of centroid localization. In (Wang et al., 2016) the authors achieve state-of-the art scores in 2016 for localizing tumors and classifying them in Whole-Slide Images (WSI) in order to identify metastatic breast cancer. They combine two deep models, one with high sensitivity to get candidates and one with low sensitivity to select candidates. The highlight of this paper is that they do not get the same errors as the pathologist, so when they combine their system with a pathologist the error rate is reduced by 85 %.

(Lin et al., 2017a) also focus on localizing and classifying tumors in WSI, in order to detect metastatic breast cancer. Their method is a Fully Convolutional Network (FCN) combined with asynchronous sample prefetching and hard negative mining (HNM). HNM is their chosen method for handling the class imbalance in this problem. Their implementation uses the predicted false positives from the previous trained classifier as negative training examples for the current one. They achieve state-of-the-art scores in tumor localization.

To segment object instances in histology images novel, (Chen et al., 2017) propose a deep contour-aware network (DCAN) combined with unified MTL. Their best results are with the contour-aware component, but they impressively receive superior to state-of-the-art results even without it.

In (Tareef et al., 2017) they attempt to accurately segment cervical nuclei and cytoplasm in pap smear images with DL and dynamic shape modeling. They achieve performance competitive to that of state-of-the-art methods, with a CNN combined with dynamic shape prior. They perform especially well in separating highly overlapping cells.

2.2.2 CVPR17 search

Single-stage detectors are simpler to apply and require less parameter tuning and expertise but have been lacking in performance, mainly because they do not produce good enough bounding boxes. In (QiongYan and LiXu, 2017) this issue is addressed by using Rolling Recurrent Convolution (RRC). RRC aggregates contextual information across feature maps and achieved state-of-the-art results in KITTI detection, becoming the first single-stage-detector to do so.

2.2.3 AI conferences search

Even though CNNs have become the most common method for computer vision problems, there are other AI methods; e.g. genetic algorithms like Plant Growth Simulation Algorithm (PGSA). (Bhattacharjee and Paul, 2017) apply a bio-inspired genetic algorithm, PGSA, for detecting white blood cells (WBCs). The authors solve it as a circle detection problem and the fitness is computed by comparing candidate solutions to the edge pixels found in the pre-processing step. It is the first use of PGSA applied to detection of WBCs, and it achieves 98.3 % TP rate and 1.7 % FP rate.

Traditionally, edge detection and object detection have been separate tasks. (Lu and Shapiro, 2017) try to unify these problems. This led to improvements in both tasks. Initial object proposals from the first stage are used as input to stage two to improve boundaries, which in turn, improve the object proposals in stage three, and this process is repeated until convergence. They achieve close to state-of-the-art results even though their method is unsupervised.

2.2.4 Other

(He et al., 2017) won best paper award at one of the most renown visual computing conferences, *ICCV 2017*. They add a small change to Faster R-CNN (Ren et al., 2015) and get state-of-the-art results in instance segmentation, bounding box object detection and person keypoint detection. The small change is a branch for predicting a object mask on each Region of Interest (RoI) in parallel with classification and bounding box regression. In order for this to work the spacial alignment has to be preserved, and therefore a quantization-free layer, RoIAlign, is introduced. RoIAlign has a big impact and improves mask accuracy by 10 - 50 %, with bigger gains for higher IoU thresholds. They also define L_{mask} so that masks across classes do not compete.

(Lin et al., 2017b) won the best student paper award at the same conference. They focus on why single-stage detectors have not matched the accuracy of two-stage detectors, especially in dense object detection. They discover that the reason for this is the class imbalance between foreground and background cases. This imbalance causes training to focus to heavily on easy classifiable examples. To rectify this, they change the loss function so that it penalizes miss-classified examples proportionally to how ”hard” they are.

In (Xue and Ray, 2017) the authors attempt to detect and localize cells in microscopy images by marking the centroid of each cell. Their method is compressed sensing (CS) combined with a CNN. They use CS to address the class imbalance issue they face, because the cell centroids represent a very small part of each image. They implement CS as random projections of the output space, which contains target cells. CS makes the method easy to ensemble, and their implementation is robust to system prediction errors because the resulting encoding has redundant values. They also implement Multi-Task Learning (MTL) to combine cell detection and cell counting to further increase performance.

2.3 Discussion of the state of the art

In this section we will discuss the relationship between the findings of the literature search and the objectives of this thesis.

2.3.1 General instance segmentation and object detection

In (Lin et al., 2017b) they design a detector they call RetinaNet. This model is included in Detectron (Girshick et al., 2018) which is the main framework in this thesis. The target cells often occupy a small portion of the image leading to class imbalance between background and foreground. Since it addresses class imbalance, this would make this architecture suited to this thesis. The problem with running this architecture in this thesis is that it is an object detector and therefore only outputs bounding boxes. This is not sufficient when the desired output is masks.

Similarily to (Lin et al., 2017b), (Bhattacharjee and Paul, 2017) deals with detection, but it focuses on detection of WBCs which is highly relevant for this thesis. Since the PGSA starts with candidate solutions, it could be combined with another approach and used as post-processing. By feeding candidate masks from e.g. a CNN to PGSA, PGSA could be used to separate TPs and FPs.

Like in (He et al., 2017), (Lu and Shapiro, 2017) performs well by combining bounding box detections and edge detection. The method here could be added as a post-processing step to e.g. a CNN, by feeding candidate masks in as initial object proposals. This could refine the masks further and lead to a higher accuracy, especially for high IoU thresholds.

Contrary to many of the other papers, (He et al., 2017) deal with instance segmentation which the Kaggle competition is an example of. That is why this is the paper we have chosen our model from. The backbone that performs best with Mask R-CNN is ResNeXt-101-FPN (He et al., 2017). Since it is effective in most tasks, it generalizes well. This should make it able to perform on localizing nuclei, which are smaller objects than the data, COCO dataset (Lin et al., 2014), often used in comparison metrics for more general architectures. Their layer RoIAlign, improves accuracy, especially when there are higher IoU thresholds.

2.3.2 Instance segmentation of large objects in medical imaging

(Lin et al., 2017a) uses HNM to boost performance. This could be implemented in this thesis by using false positives with no overlap with ground truths as negative examples.

The results from (Wang et al., 2016) are old and have been beaten since in (Lin et al., 2017a). Therefore this paper is not very relevant any more. This goes to show that filtering by year is a necessity in a literature review in this field.

2.3.3 Instance segmentation of small objects in medical imaging

(Tareef et al., 2017)s work of separating highly overlapping cells is interesting. They achieve this with dynamic shape prior, which is something that could be implemented in this thesis.

In (Chen et al., 2017) they achieve very good results on the same task as in this thesis. The unified MTL that uses contour information and complementary appearance information could be of relevance to this thesis.

Another method that could be relevant is the combining of CNNs that have separate subtasks into one instance segmentor. This shows promising results in (Khoshdeli and Parvin, 2018).

2.3.4 Cell centroid localization in medical imaging

(Xue and Ray, 2017) demonstrates that CNNs can be combined with compressed sensing for a potential performance boost. Since the data used for this thesis also is sparse in target locations and therefore faces class imbalance issues, compressed sensing could be relevant to experiment with in this thesis.

(Khoshdeli and Parvin, 2018)s feature-based representation could be implemented as pre-processing in this thesis. Taking the LoG og the NMF of the LoG for accentuating blob-shaped objects, could make it easier to distinguish nuclei from the background. This could then be fed as input to the backbone model of this thesis.

In (Xie et al., 2018) they achieve superior to state-of-the-art results in cell centroid localization. The dense proximity map they output could be relevant if it could be combined with a model that outputs masks. The proximity map could be used to evaluate positives as true or false at inference time.

Chapter 3

Method

3.1 General overview of field

Identifying the nuclei and their borders is an instance segmentation problem. Most of the earlier approaches for identifying nuclei have been based on locating cell centroids. Traditionally, classical image processing techniques have been the method for solving this. Classical image processing techniques, like intensity thresholding and feature detection, are an intuitive way of detecting edges. HoG transform (McConnell, 1986) was also used for edge detection, and LoG (Kong et al., 2013) advanced blob detection results. Traditionally these simple feature extractors have been used in combination with a classifier that based on the values in the feature vector, determines if a region is a cell or not.

The feature extractor combined with a classifier is intuitive, but suffers from a lack of generality. Also, it often demands a certain level of expertise in the field of the problem to pick the correct feature extractor(s). Because of this, artificial neural nets and more specifically CNNs have become the common methods. The advantage of neural nets are that they can extract the features implicitly themselves, so expertise in the field is not as essential, and they are good at generalizing, because of the implicit feature extraction. An advantage of CNNs is that they are good at recognizing the same object even if it is a different size or in a different place in the image.

Some of the most recent advances have combined CNNs (Kainz et al., 2017; Khoshdeli and Parvin, 2018; Khoshdeli and Parvin, 2018; Wang et al., 2016) that have different subtasks to achieve better performance on the main task. Others have used different methods, like compressed sensing (Xue and Ray, 2017), colour decomposition (Khoshdeli and Parvin, 2018), LoG of NMF (Khoshdeli and Parvin, 2018), to transform the input before feeding it into a CNN.

3.2 Mask R-CNN

The state-of-the-art method in instance segmentation is Mask-RCNN from (He et al., 2017). It generalizes well. This is an important reason for picking it and adapting it to nuclei segmentation. Mask R-CNN is a type of Region-based CNN and more specifically an extension of Faster R-CNN, which in turn is an improvement on Fast R-CNN (Girshick, 2015). As described in (He et al., 2017), the difference between it and Faster R-CNN is the addition of "(..) a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression. The mask branch is a small FCN (Long et al., 2015) applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner". This transforms Faster R-CNN which uses pooling, into a model that has pixel-to-pixel alignment between input and output in the network. Mask R-CNN achieves this with a quantization-free layer, RoIAlign. This is critical when the evaluation metric has high IoU thresholds and is therefore a big reason for Mask R-CNNs superior results. Another change from Faster R-CNN is their decoupling of mask and class prediction. For every RoI, they predict one mask per class independently from the other masks.

Mask R-CNN is built on Faster R-CNN, which consists of two stages: a Region Proposal Network (RPN) and basically Fast R-CNN. The firste stage, RPN, outputs candidate object bounding boxes. In Faster R-CNN, the second stage takes these as input and predicts classes and bounding boxes in parallel. While in Mask R-CNN, this second stage also predicts masks in parallel. This leads to MTL during training with the loss being defined as in equation 3.1.

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.1}$$

$\mathrm{L_{cls}}$ and $\mathrm{L_{box}}$ are unchanged from Fast R-CNN. $\mathrm{L_{mask}}$ is new in Mask R-CNN and is defined only for the mask that corresponds to the ground truth class for the ground truth mask. $\mathrm{L_{mask}}$ is the average binary cross-entropy loss of the per-pixel sigmoid of the masks pixels. This is what makes Mask R-CNN able to decouple mask and class prediction, so that masks across classes do not compete.

To counter the spatial aggregation problems RoIPool introduces, Mask R-CNN adds a layer called RoIAlign. This layer, in contrast to RoIPool, does not quantisize the RoI boundaries or bins. It uses bilinear interpolation (Jaderberg et al., 2015) to compute the input feature values.

Mask R-CNN works well with several backbone architectures, but especially with ResNet-FPN. During training, an RoI is considered positive and $\mathrm{L_{mask}}$ is defined, if the IoU with a ground-truth box is 0.5 or higher. The training is image-centric (Girshick, 2015).

Our contribution will be applying Mask R-CNN to the problem of instance segmentation of nuclei.

3.3 Data

The image set BBBC038v1 was used for training and testing. It is available from the Broad Bioimage Benchmark Collection (Ljosa et al., 2012), and was the standard dataset for the Kaggle competition. The dataset is annotated by humans, so there are errors there, but

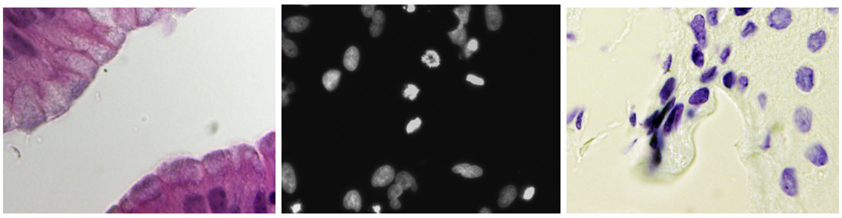


Figure 3.1: Three arbitrarily picked images from stage1_test, showing the diversity in colour and shape the model needs to handle.

there should not be any big systematic errors. All the images in the dataset have a height between 256 and 524 pixels and a width of between 161 and 696 pixels. The number of nuclei varies greatly; from only a couple to several hundred. The images have been captured under different conditions. They vary in color and lighting, and the cells vary in type of cell. This can be seen in figure 3.1.

The dataset consists of a training set of 670 images. On average there are 29461/670 = 44 nuclei per training image. This means that even though 670 images is not a lot, there are close to 30 000 training cases before doing any data augmentation.

Since the Kaggle competition was split into two stages, the test set for stage 1 was used as a validation test set. This set consists of 65 images with 4152 nuclei for an average of 64 nuclei per image. So this set had almost 50 % more nuclei per image than the training set. A common size for a validation set is 10 % of the training set, which this set is.

The test set for stage 2 of the competition consists of 3019 images. This will be used as the test set for this thesis. Since Kaggle have not released the solutions for this set the only way to get a score on this test set is to run results through the submission page on Kaggle[1]. Therefore it is unknown how many nuclei there are on average in the test set. A result we submitted online at Kaggle which got a score of 0.455, had 81167 nuclei proposals, which makes for an average of 27 nuclei proposals per image. Equation 4.5 puts an upper bound at 27/0.455=59 and a lower bound at 27*0.455=12 of how many nuclei there are per image in the test set.

In order to use Detectron (Girshick et al., 2018) the data needed to be converted to COCO-.json format. The data was re-structured into the correct directory tree. Then we edited pycococreator (waspinator, 2018) and ran it, to create .json-annotation files for the data set: one file for training, one for testing stage 1, the validation set, and one for testing stage 2, the test set.

3.4 Models

The best model available for this task is Detectrons Mask R-CNN with Bells & Whistles; e2e_mask_rcnn_X-152-32x8d-FPN-IN5k_1.44x. The model with pre-trained weights is available from the overview of the models[2]. When trying to train it from scratch or run inference on the pre-trained model, it runs out of memory on the GPU used in this thesis.

[1]`https://www.kaggle.com/c/data-science-bowl-2018/submit`
[2]`https://github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md`

This is a know error[3], since the model needs to run on a 16 GB GPU, even though this is not specified in Detectrons overview of the models. For the experiments in this thesis, we have only used a 12 GB GPU.

Therefore we have run the experiment on our second-choice model: the end-to-end version of e2e_mask_rcnn_X-101-64x4d-FPN_1x. This is according to the same overview of the models, the model with the second-best performance on mask prediction. All models have been tested on (Lin et al., 2014), and Mask R-CNN with Bells & Whistles achieves a mask AP score of 41.5, while our second-choice receives a score of 37.5. This means that the model with Bells & Whistles is close to 11 % better. The chosen model has been adapted to the problem, including changing the number of classes to two, one for nuclei and one for background. Some modifications have also been applied to make the model more suited for nuclei detection and localization. These are described in the next subsections.

3.4.1 Training

The training is very similar to training Faster R-CNN (Ren et al., 2015). The difference is that it predicts candidate masks in parallel with classification and bounding boxes. This leads to the loss in equation 3.1. For every image batch, the proposals for masks, classes and bounding boxes are evaluated against the ground truth and the loss is calculated. Then this loss is propagated back trough the network and the hyper-parameters weight the learning.

The learning rate is set to 0.0025 with Gamma as 0.1. We train for 180 000 iterations on a single GPU and weight decay is 0.0001. Since we have less than 700 training images, we have decided to go with pre-trained weights. The pre-trained weights[4] from the overview of the models have been used as start weights when training.

We have trained the same model twice, but with some adjustments. The two models' configurations files can be found in Appendix B 6.2. The model is adjusted to better adapt to the specific problem of nuclei segmentation. The model with adjustments will be referred to as the modified model, while the model without adjustments will be referred to as the standard model. The main differences between the models are the increase of importance of the L_{mask} by 20 % by adding MRCNN.WEIGHT_LOSS_MASK: 1.2 to the configuration file, and the increase in number of detections per image from 100 to 500. We have also added different scales during training, so that the training images are re-sized so the shorter side varies between 700, 800, 900 and 980 pixels. This is effectively a way to data augment.

3.4.2 Inference and post-processing

In addition, since the Kaggle competition does not allow overlapping instances, some post-processing had to be done. We used a script from (gangadher, 2018) and modified it. This

[3]https://github.com/facebookresearch/Detectron/issues/35

[4]https://s3-us-west-2.amazonaws.com/detectron/36494496/12_2017_baselines/e2e_mask_rcnn_X-101-64x4d-FPN_1x.yaml.07_50_11.fkwVtEvg/output/train/coco_2014_train%3Acoco_2014_valminusminival/generalized_rcnn/model_final.pkl

script removes overlapping pixels from the last added mask. It also removes all masks under a certain certainty threshold, masks that are too small or masks that overlap with all other masks more than a certain threshold. The certainty threshold, describing how certain the model is that the mask actually is a nucleus, is set to 0.5. The size threshold is set to 15, so that all masks that are smaller than 15 pixels are discarded. The mask-overlap threshold is set to 0.3, so at least 70 % of the mask needs to be non-overlapping with the union of all the other masks.

3.5 Frameworks

In this thesis we have used Facebooks Detectron (Girshick et al., 2018). It is built on caffe2 (caffe2, 2018) and is primarily an object detection tool. Caffe2 is a deep learning framework that works with a python API. The rest of the scripts we have written for this thesis are python and shell scripts. In order to run Detectron on the remote GPU and maintain all its dependencies, Docker (Merkel, 2014) was used.

The full overview is presented in figure 3.2. The model runs a type of Mask R-CNN and takes as input a configuration file specifying settings. During training the model takes as input the training data. When it is finished training, it takes the test data as input and runs inference on the images in the dataset. Inference is run by feeding the images into the model. The model outputs candidate segmentations in a .json-file. These segmentations are then post-processed to separate instances and remove unlikely candidates as described in subsection 3.4.2. The results from the post-processing are then evaluated against the ground truth using the evaluation metrics from section 4.2, leading to a F1-score which tells us the accuracy of the model.

If the model is run on the test set for the Kaggle leaderboard, there is no ground truth file available for evaluating the results locally. The only way to get an evaluation is to submit the post-processed results file online at Kaggle. The results, F1-score, are calculated as described in section 4.2.

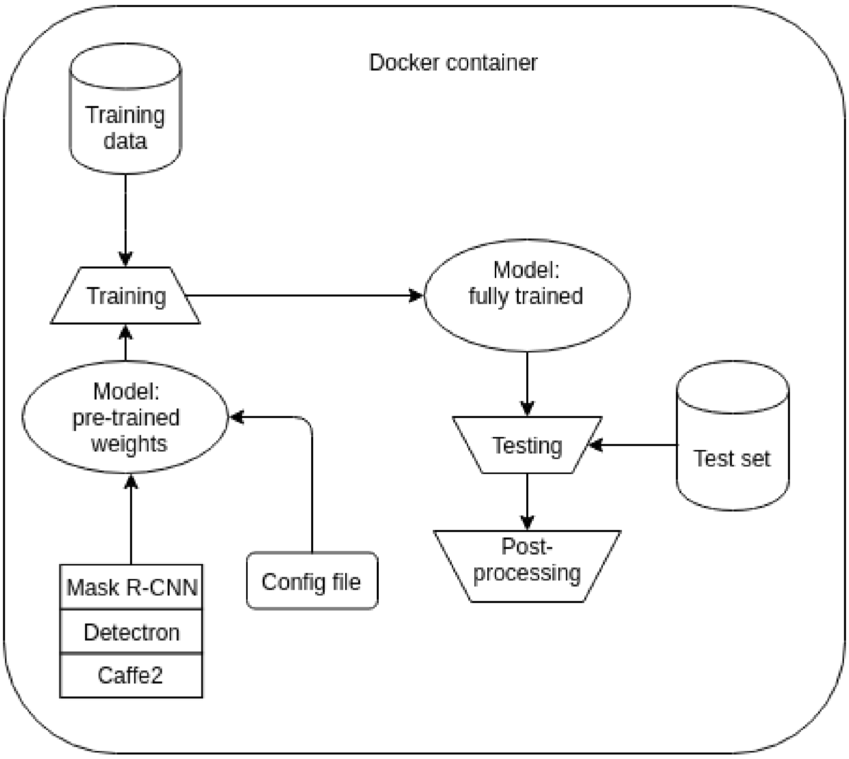


Figure 3.2: Framework implemented and tested on Kaggles 2018 Data Science Bowl

Chapter 4

Experiment

4.1 Setup

The experiments were run on a GeForce GTX TITAN X with 12 GB memory. Since this was run on a shared machine, the model was run inside a Docker container. To be able to edit the code locally and then quickly run it, the Detectron source code with models was located on the shared machine and mounted locally for editing. A guide for how to do this can be found here[1].

4.2 Evalutation

4.2.1 Evaluation metrics

The evaluation used in this thesis is the same one as the one used in the Kaggle competition, which is described at the competition web page[2]. The evaluation is based on a average of F1-scores at different Intersection over Union (IoU) thresholds. To evaluate how well a single mask does at marking the ground truth its IoU score is computed in equation 4.1. This penalizes both marking pixels as false positives and false negatives equally. Thereby it balances recall and precision. The same is true for equation 4.2.

$$IoU(A,B) = \frac{A \cap B}{A \cup B} \tag{4.1}$$

The thresholds range from *0.5* to *0.95* with a step size of *0.05*. For every threshold *t* the score from equation 4.2 is calculated by finding the number of true positives, false positives and false negatives.

[1]https://www.digitalocean.com/community/tutorials/how-to-use-sshfs-to-mount-remote-file-systems-over-ssh

[2]`https://www.kaggle.com/c/data-science-bowl-2018#evaluation`

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{4.2}$$

A predicted mask is considered a true positive when it matches a ground truth mask with an IoU above the threshold. False positives are true positives subtracted from all positive results 4.3. False negatives are true positives subtracted from all ground truths 4.4.

$$FP(t) = \mathit{AllPositives} - TP(t) \tag{4.3}$$

$$FN(t) = \mathit{AllGroundTruths} - TP(t) \tag{4.4}$$

The total score for an image then becomes the average of the F1-score at all the different thresholds in equation 4.5.

$$\frac{1}{|thresholds|} \sum_t \frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{4.5}$$

Then to get the F1-score for the whole image set, the average of all the individual image scores is computed.

4.2.2 Evaluation script

In order to get the results from the stage1 test set, we wrote our own evaluation script following the same metrics. To speed this up, the floating point mask scores were converted to integers that correspond to at how many thresholds they would be considered a hit 4.6. E.g. a score of 0.69 would translate to 4 hits, since it would be a hit at thresholds *0.5*, *0.55*, *0.6* and *0.65*.

$$number\_of\_hits\_score = \left\lceil \frac{(score - 0.5)}{0.05} \right\rceil \tag{4.6}$$

Then, instead of looping through each threshold, the *number_of_hits_score* for each mask were added up and counted as the number of hits. This meant that the number of all positives and number of ground truths had to be multiplied by 10, the total number of thresholds, for equation 4.3 and equation 4.4 to still be valid. Equation 4.5 then becomes equation 4.7.

$$\frac{\sum_t TP(t)}{\sum_t TP(t) + \sum_t FP(t) + \sum_t FN(t)} \tag{4.7}$$

4.3 Results

The training of the modified and standard model took around 50 h each on our setup 4.1. Both models were trained for 180 000 iterations. The reason for there being not that many datapoints, is that Detectron does not do any testing while training. Instead it saves the model at certain steps, which can be specified in its configuration file. Then after training

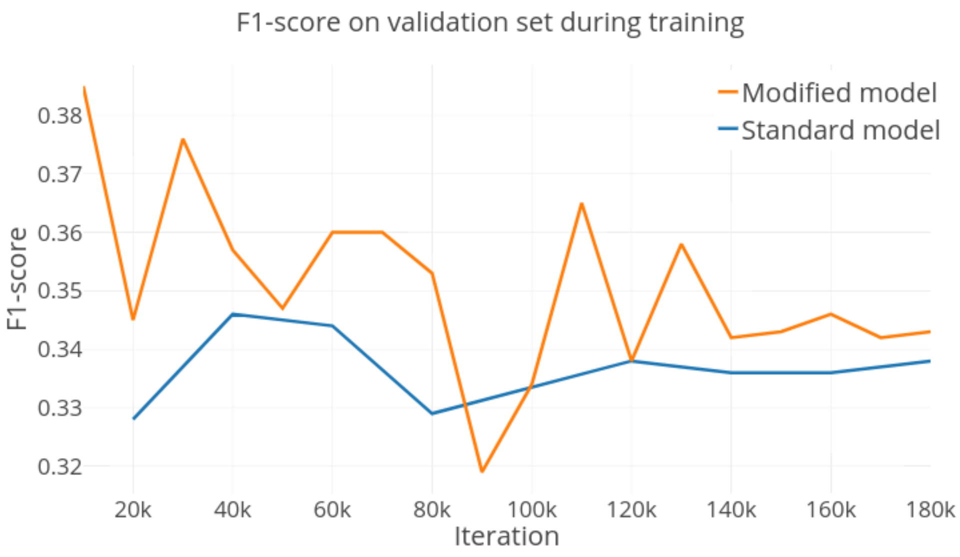


Figure 4.1: Results from test set: F1-score

inference can be run on the saved models to get the accuracy on the validation set at every selected step of the training. The standard in Detectron is to save a model every 20 000 iterations. For the modified model we changed this to every 10 000 iterations.

4.3.1 Validation set

Figure 4.1 shows that the score from the validation set converges to 0.34-0.35 for the modified model and 0.33-0.34 for the standard model. The best score on the validation set, 0.385, is the result from the modified model after 10 000 iterations. The standard models best score is 0.346. On average for all the saved models, the modified model with an average F1-score of 0.351, performs 0.014 better than the standard model, which has an average F1-score of 0.337. This is a 4 % increase in performance.

4.3.2 Test set: Kaggle

The ground truth of the test set used for the Kaggle competition is not publicly available. The only way to get a performance score is to submit results online and let Kaggle do the evaluation of your results. The model that had the best performance on the validation set was the modified model saved after 10 000 iterations. When we run inference on the test set with this model and submit online at Kaggle, we achieved an F1-score of 0.460. We did not complete our implementation in time to enter the competition, but if we had, this score would put us at 291st place of the 738 participating teams and individuals that entered the leaderboard for stage 2. For stage 1, there were more than 2000 participants, but most of them probably skipped stage 2 due to subpar performance on stage 1. Also a lot of individual competitors joined a team of competitors, reducing the number of leaderboard entries.

Chapter 5

Discussion

5.1 Literature review

The literature search showed that there are several different state-of-the art methods for detecting and localizing nuclei. (Lin et al., 2017b) and (Bhattacharjee and Paul, 2017) perform very well on detection, but do not output masks. (Lu and Shapiro, 2017) demonstrate the effectiveness of combining object proposals and edge detection, but they utilize unsupervised training, so we are unsure if it would have been competitive to implement in this thesis. Others (Lin et al., 2017a) have focused on detecting tumors and classifying them, while this thesis does not focus on classification.

The feature-based representation utilized in (Khoshdeli and Parvin, 2018) shows promise as a pre-processing method, but the most difficult problem in constructing good masks for the data in this thesis is separating overlapping nuclei. Therefore we are unsure if this feature-based representation would help with this issue, since it is mainly a tool for accentuating nuclei from the background. (Xie et al., 2018; Xue and Ray, 2017) both focus on cell detection by outputting a cell centroid and not a mask. Therefore they would require adaption if they were to be used for this thesis.

However, we discovered when finished with the literature review, that this problem is an instance segmentation problem. If we were to do the literature review again, we would add the term ”instance segmentation” and maybe remove some of the less fitting terms, like detection and localization. This could have lead to a more precise search with less non-relevant papers.

There are several state-of-the-art methods for instance segmentation in medical imaging. (Tareef et al., 2017) and (Khoshdeli and Parvin, 2018) focus on overlapping cells. To solve this, (Tareef et al., 2017) utilize dynamic priors, while (Khoshdeli and Parvin, 2018) fuse deep CNNs. They have annotated their own dataset which they use for evaluation, so there are no comparable results. (Chen et al., 2017) design a deep contour-aware network. Their method and (Tareef et al., 2017)s method achieve state-of-the-art results on their respective datasets. The difference between these and (He et al., 2017) is that Mask R-CNN does not focus on medical imaging and is not as heavily engineered. Mask R-CNN is a single model and is therefore easier to adapt and implement. It is the state-of-the-art for more general instance segmentation. In addition, it is interesting to experiment with adapting a general instance segmentation model to the specific problem of instance segmentation of nuclei in medical imaging. The results from (Tareef et al., 2017; Khoshdeli and Parvin, 2018; Chen et al., 2017; He et al., 2017) are difficult to compare since they have not been evaluated on the same datasets. We chose Mask R-CNN as our model since it generalizes well, is easy to adapt and has, to our knowledge, not been experimented with for solving instance segmentation of nuclei. In the next subsections we will show that the chosen model performs well, is competitive and could have been competitive against the best performing approaches in the Kaggle competition with some modifications.

5.2 Method

To apply Mask R-CNN in this thesis some modifications had to be done, including changing the number of classes to two, one for nuclei and one for background. Other modifications were made to the standard model to make the modified model more suited for the problem in this thesis. These were mainly increasing the weighting of L_{mask} by 20 % and the number of detections per image from 100 to 500. The increased weight of L_{mask} also implicitly decreases the importance of $L_{classification}$. This can increase performance, since the classification branch of Mask R-CNN is not very important for performance when segmenting instances of only one type.

5.3 Results

The reason the scores are so low for this problem, is the way the score is calculated in section 4.2. Since the evaluation only counts a mask overlapping 50 % with a ground truth value as a hit at one threshold, the evaluation metric severely punishes bad separation of overlapping nuclei and small inaccuracies in mask borders. It also punishes both false positives and false negatives, which also contributes to F1-scores being below what is considered ”normal” accuracy values in other ML problems.

In figure 4.1 we see that the modified model performs better than the standard model. On average, the modified model performs 4 % better than the standard model. This indicates that our modifications led to a performance increase, but only a very slight one. Since the performance increase is small, we can not rule out that there is a probability of the increase being a coincidence.

The modified model seems to overfit the data a little bit as the training proceeds. Its highest F1-scores are 0.385 and 0.376 from the 10k and 30k iterations models, and the score converges to below 0.35 after around 140k iterations. The standard model does not seem to overfit, but also converges, to a score of almost 0.34. Since the modified model does some data augmentation, it should be less susceptible to overfitting. With this in mind, and the standard model seemably not overfittig, it might be just a coincidence that the modified model achieves its best scores after 10k and 30k iterations.

We are missing the 100 k iterations model for the standard model, due to memory failure while saving the model during training, and we have saved a model twice as often during training of the modified model. This is why we have 19 datapoints for the modified model and 9 for the standard. If we had collected the same amount of data points for the modified model (at 20k, 40k, 60k, ...), the graph would be smoother, with higher lows and lower highs. This also supports that it is a coincidence that the modified model seems to be overfitting.

The results also indicate that we do not need to train for 180 k iterations, when our best results are achieved after 10 k and 30 k iterations. This is probably the standard number of iterations due to the models normally being trained on bigger datasets.

5.4 Comparison to Kaggle

We chose the model that performed best on the validation set as the model we ran the test set on. The modified model trained for 10 k iterations was the top performer and achieved an F1-score of 0.460 on the test set when submitted and evaluated online at Kaggle. We did not complete our implementation in time to enter the competition, but if we had, this score would place us 291st out of 738 participating individuals and teams on the leaderboard for stage 2 of the competition. This demonstrates that our chosen method is competitive.

The top performing model scored 0.385 on the validation set and 0.460 on the test set. This is an increase of 19 % and shows that the validation set probably is harder than the test set. If the only goal was to get the highest score on the test set, we could run inference on the test set with all the saved models. The problem with this is that it would practically be training on the test set by trying different models and choosing the one that performs best. This is why we have only run, and reported the scores from, the test set on the 10 k iterations model.

5.4.1 Winner of competition

The winner of Kaggles Data Science Bowls solution can be found at the competition discussion forum[1]. On the stage 2 leaderboard their solution achieved an F1-score of 0.631[2], beating the second-placed entry by impressive 0.017. A screenshot of the leaderboard can be found in appendix A 6.2. They describe their approach as UNet on steroids. They use a encoder-decoder architecture based on UNet with encoders pretrained on ImageNet. Their choice of encoders includes DPN-92, Resnet-152, InceptionResnetV2 and Resnet101. They do the same preprocessing as on ImageNet, since they use pretrained models. The output from the encoder is post-processed by LightGBM (Ke et al., 2017). They separate candidate nuclei by looking at features of the candidates: e.g. solidity, area, neighbours median area and others. In addition, they do some of the same post-processing as we have done in this thesis, removing candidates that are below a certain certainty threshold.

They discovered quite early that one of the most challenging aspects of this problem is separating overlapping nuclei. This led to them predicting the borders of the nuclei separately from predicting the masks. They do this by training a network to predict the

[1]`https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741`
[2]`https://www.kaggle.com/c/data-science-bowl-2018/leaderboard`

borders. The network that predicts the borders and the network that predicts the masks are then combined and fed as input to watershed post-processing.

Due to the training set being relatively small they have done a lot of data augmentation. The full list is available at the competition forum[3], and include gaussian noise, greyscaling, inverting and more complicated techniques like channel shuffle and nuclei copying on images. They believe that the last two techniques were important, the last one due to it focusing the learning more on the hard part of the problem: separating overlapping nuclei.

In addition they do some small test-time augmentation: standard flips and rotations of 90, 180 and 270 degrees. Another way they compensated for the lack of training data, was adding other datasets and using them in training as well. They added parts of other datasets including: the janowczyk (Janowczyk, 2018) and nucleisegmentationbenchmark (Kumar et al., 2017) datasets. They do not do any ensembling except averaging masks before post-processing.

Watershed post-processing is a powerful technique that if added to our implementation could have boosted performance. It could especially have helped with the difficult task of separating overlapping nuclei. Another technique that we could have explored further is data augmentation. Since the training set is small, different data augmentation techniques could have compensated for this. We implemented some data augmentation techniques, but could have explored more techniques.

5.4.2 Other top placements in the competition

Most of the other teams that placed 10 or higher on the leaderboard, used some kind of Mask R-CNN. This indicates that our choice of Mask R-CNN, even if not absolutely optimal, was definitely a competitive choice. The difference between our method and their methods is mainly that they use our first-choice backbone architecture: X-152-32x8d-FPN. In addition their implementations have been heavily engineered. They have added a lot of pre- and post-processing and also done more data augmentation. With another graphics card, we could have used our first-choice model. It is reasonable to assume that this alone would have increased our score with around 10 %, leading to a score of 0.504, just outside of the top 100 on the leaderboard.

One of the joint second-place solutions has been documented at the competition forum[4]. They achieved a score of 0.614. They use Mask R-CNN from Matterport(matterport, 2018), which is another framework other than Detectron. Other than being another framework, the two Mask R-CNN architectures are similar. Their solution is very straightforward and utilizes only a single model. The most notable part of their implementation is that they have done a lot of data augmentation. The data augmentation consists of a lot of scaling augmentation; zooming in and out and changing aspect ratio, and also 15 different test time augmentations including rotations, scaling and channel color shifting. They have also made some changes to the parameters that are described in their post.

[3]`https://www.kaggle.com/c/data-science-bowl-2018/discussion/54741`
[4]`https://www.kaggle.com/c/data-science-bowl-2018/discussion/56393`

5.4.3 Comparison of winner and 2nd place

The main differences between the winner of the competition and the 2nd place, were the backbone architecture and post-processing. The winners chose UNet combined with Watershed post-processing, and the 2nd place chose Mask R-CNN. Mask R-CNN is a single model and an architecture that is easy to operate. In contrast, UNet combined with Watershed post-processing requires more engineering and has more hyper-parameters to tune. The advantage of the winners implementation is that they had a separate network for predicting the borders. This helped them separate overlapping nuclei, which was a critical part of the competition. Mask R-CNN also does classification, so its full strength was not utilized here. If the competition included classifying the segmented nuclei, this Mask R-CNN implementation or one of the other top Mask R-CNN implementations, probably would have won.

Chapter 6

Conclusion and future work

6.1 Conclusion

In this thesis we have conducted a literature review for finding the state-of-the-art methods for detecting and localizing nuclei in medical imaging. We have implemented and applied one of the state-of-the-art methods, Mask R-CNN (He et al., 2017) to this task. We trained the same model twice, first without modifications, then with some adjustments. The adjustments were made to adapt the model to the problem of detecting and localizing nuclei, and included increasing the weight of L_{mask}. The modified model performed 4 % better on average than the non-modified model. The implementation that performed best on the validation set with a score of 0.385, achieved an F-1 score of 0.460 on the test set. We did not complete our implementation in time to enter the competition, but if we had, this score would have lead to 291st place out of 738 participating individuals or teams in Kaggles 2018 Data Science Bowl[1]. Even though we were 0.171 behind the winning approach which scored 0.631, we implemented the same single model as the second-place approach. The differences between our implementation and the second-placed implementation were mainly a different backbone and heavy data augmentation. This shows that our approach was competitive, and with modifications could have been competing for the top placements in the competition.

6.2 Future work

Mask R-CNN also does classification, on top of solving the problem in this thesis. Therefore it could be applied to segmenting the nuclei into masks and classifying the nuclei into type of nuclei. Being a very general instance segmentation framework, it could do instance segmentation of almost any type of data within visual computing.

One of the simplest ways to increase performance would be more tuning to discover which data augmentation techniques are effective, and doing a hyper-parameter search.

[1]`https://www.kaggle.com/c/data-science-bowl-2018`

Another method to increase performance would be training several Mask R-CNN models on different augmentations of the training data, and then combining them by ensemble methods. A more advanced addition would be adding compressed sensing as done in (Xue and Ray, 2017) as pre-processing. This would increase precision in separating overlapping cells, due to the redundant information CS gives. Since all the Mask R-CNN approaches were beaten by Unet with Watershed post-processing in the Kaggle competition, it could increase performance to combine Mask R-CNN with Watershed as post-processing.

Bibliography

Beucher, S., Meyer, F., 1992. The morphological approach to segmentation: the watershed transformation. Optical Engineering-New York-Marcel Dekker Incorporated- 34, 433–481.

Bhattacharjee, D., Paul, A., 2017. A leukocyte detection technique in blood smear images using plant growth simulation algorithm. In: AAAI. pp. 17–23.

caffe2, 4 2018. caffe2. [accessed: 27th of May 2018].
URL `https://github.com/caffe2/caffe2.git`

Chen, H., Qi, X., Yu, L., Dou, Q., Qin, J., Heng, P.-A., 2017. Dcan: Deep contour-aware networks for object instance segmentation from histology images. Medical image analysis 36, 135–146.

gangadher, 5 2018. Object detectron nuclei. [accessed: 27th of May 2018].
URL `https://github.com/gangadhar-p/NucleiDetectron.git`

Girshick, R., 2015. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448.

Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K., 2018. Detectron. `https://github.com/facebookresearch/detectron`, [accessed: 27th of May 2018].

He, K., Gkioxari, G., Dollár, P., Girshick, R., Mar. 2017. Mask R-CNN. ArXiv e-prints.

Jaderberg, M., Simonyan, K., Zisserman, A., et al., 2015. Spatial transformer networks. In: Advances in neural information processing systems. pp. 2017–2025.

Janowczyk, A., 4 2018. Janowczyk medical imaging database. [accessed: 7th of July 2018].
URL `http://www.andrewjanowczyk.com/deep-learning/`

Kainz, P., Pfeiffer, M., Urschler, M., 2017. Segmentation and classification of colon glands with deep convolutional neural networks and total variation regularization. PeerJ 5, e3874.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.-Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems. pp. 3146–3154.

Khoshdeli, M., Parvin, B., Feb. 2018. Deep Learning Models Delineates Multiple Nuclear Phenotypes in HE Stained Histology Sections. ArXiv e-prints.

Khoshdeli, M., Parvin, B., 2018. Feature-based representation improves color decomposition and nuclear detection using a convolutional neural network. IEEE Transactions on Biomedical Engineering 65 (3), 625–634.

Kong, H., Akakin, H. C., Sarma, S. E., 2013. A generalized laplacian of gaussian filter for blob detection and its applications. IEEE transactions on cybernetics 43 (6), 1719–1733.

Kumar, N., Verma, R., Sharma, S., Bhargava, S., Vahadane, A., Sethi, A., 2017. A dataset and a technique for generalized nuclear segmentation for computational pathology. IEEE transactions on medical imaging 36 (7), 1550–1560.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324.

Lin, H., Chen, H., Dou, Q., Wang, L., Qin, J., Heng, P.-A., Jul. 2017a. ScanNet: A Fast and Dense Scanning Framework for Metastatic Breast Cancer Detection from Whole-Slide Images. ArXiv e-prints.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., Aug. 2017b. Focal Loss for Dense Object Detection. ArXiv e-prints.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., Dollár, P., May 2014. Microsoft COCO: Common Objects in Context. ArXiv e-prints.

Ljosa, V., Sokolnicki, K. L., Carpenter, A. E., 2012. Annotated high-throughput microscopy image sets for validation. Nat Methods 9 (7), 637.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440.

Lu, Y., Shapiro, L. G., 2017. Closing the loop for edge detection and object proposals. In: AAAI. pp. 4204–4210.

matterport, 6 2018. Matterports mask r-cnn. [accessed: 6th of July 2018].
URL `https://github.com/matterport/Mask_RCNN`

McConnell, R. K., Jan. 28 1986. Method of and apparatus for pattern recognition. US Patent 4,567,610.

Merkel, D., Mar. 2014. Docker: Lightweight linux containers for consistent development and deployment. Linux J. 2014 (239).
URL `http://dl.acm.org/citation.cfm?id=2600239.2600241`

QiongYan, J., LiXu, Y., 2017. Accurate single stage detector using recurrent rolling convolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99.

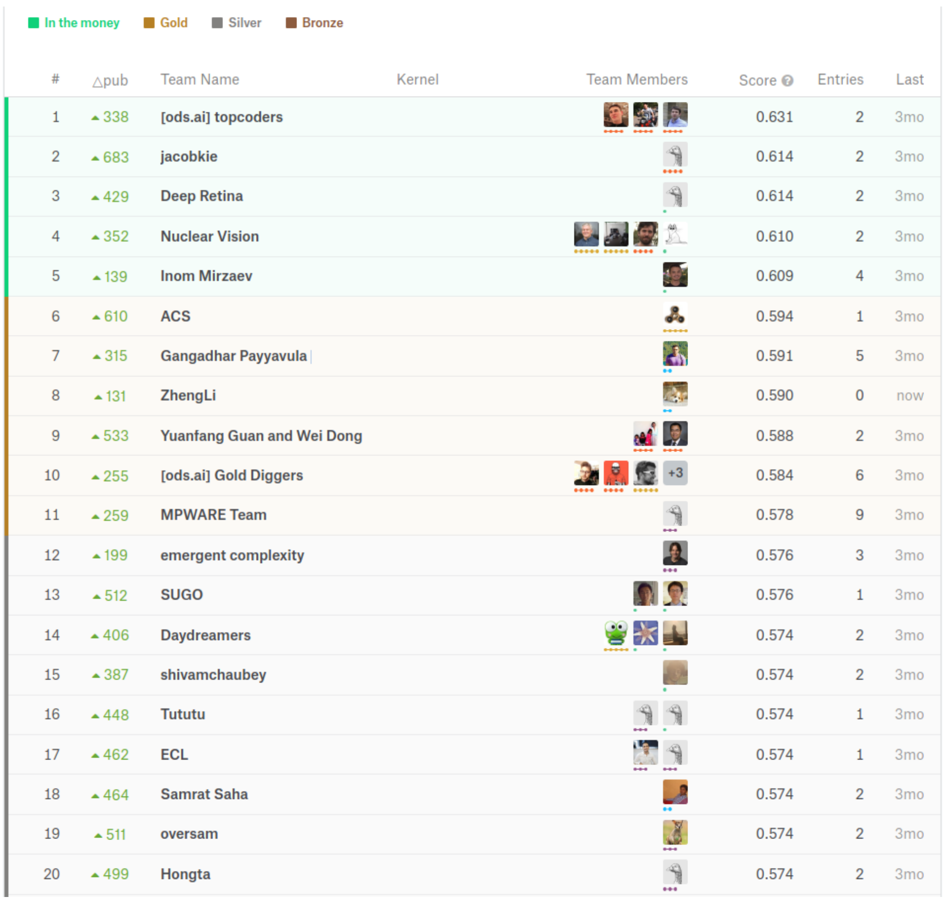
Tareef, A., Song, Y., Huang, H., Wang, Y., Feng, D., Chen, M., Cai, W., 2017. Optimizing the cervix cytological examination based on deep learning and dynamic shape modeling. Neurocomputing 248, 28–40.

Wang, D., Khosla, A., Gargeya, R., Irshad, H., Beck, A. H., Jun. 2016. Deep Learning for Identifying Metastatic Breast Cancer. ArXiv e-prints.

waspinator, 5 2018. Pycoco creator. [accessed: 3rd of June 2018].
URL `https://github.com/waspinator/pycococreator.git`

Xie, Y., Xing, F., Shi, X., Kong, X., Su, H., Yang, L., 2018. Efficient and robust cell detection: A structured regression approach. Medical image analysis 44, 245–254.

Xue, Y., Ray, N., Aug. 2017. Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing. ArXiv e-prints.



🟩 In the money 🟧 Gold ⬜ Silver 🟫 Bronze

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲338	[ods.ai] topcoders			0.631	2	3mo
2	▲683	jacobkie			0.614	2	3mo
3	▲429	Deep Retina			0.614	2	3mo
4	▲352	Nuclear Vision			0.610	2	3mo
5	▲139	Inom Mirzaev			0.609	4	3mo
6	▲610	ACS			0.594	1	3mo
7	▲315	Gangadhar Payyavula			0.591	5	3mo
8	▲131	ZhengLi			0.590	0	now
9	▲533	Yuanfang Guan and Wei Dong			0.588	2	3mo
10	▲255	[ods.ai] Gold Diggers		+3	0.584	6	3mo
11	▲259	MPWARE Team			0.578	9	3mo
12	▲199	emergent complexity			0.576	3	3mo
13	▲512	SUGO			0.576	1	3mo
14	▲406	Daydreamers			0.574	2	3mo
15	▲387	shivamchaubey			0.574	2	3mo
16	▲448	Tututu			0.574	1	3mo
17	▲462	ECL			0.574	1	3mo
18	▲464	Samrat Saha			0.574	2	3mo
19	▲511	oversam			0.574	2	3mo
20	▲499	Hongta			0.574	2	3mo

Figure 6.1: The top 20 entries on the stage 2 leaderboard of Kaggles Data Science Bowl 2018

Appendix A

Appendix A contains the leaderboard shown in figure 6.1, of Kaggles Data Science Bowl 2018 at 7th of July.

Appendix B

This appendix contains the two configuration files for the two models we have trained in this thesis. The first configuration file is for the unmodified model and is detailed below:

```
MODEL:
  TYPE: generalized_rcnn
  CONV_BODY: FPN.add_fpn_ResNet101_conv5_body
  NUM_CLASSES: 2
  FASTER_RCNN: True
  MASK_ON: True
NUM_GPUS: 1
SOLVER:
  WEIGHT_DECAY: 0.0001
  LR_POLICY: steps_with_decay
  # 1x schedule (note TRAIN.IMS_PER_BATCH: 1)
  BASE_LR: 0.001
  GAMMA: 0.1
  MAX_ITER: 180000
  STEPS: [0, 120000, 160000]
FPN:
  FPN_ON: True
  MULTILEVEL_ROIS: True
  MULTILEVEL_RPN: True
RESNETS:
  STRIDE_1X1: False
  TRANS_FUNC: bottleneck_transformation
  NUM_GROUPS: 64
  WIDTH_PER_GROUP: 4
FAST_RCNN:
  ROI_BOX_HEAD: fast_rcnn_heads.add_roi_2mlp_head
  ROI_XFORM_METHOD: RoIAlign
  ROI_XFORM_RESOLUTION: 7
  ROI_XFORM_SAMPLING_RATIO: 2
MRCNN:
  ROI_MASK_HEAD: mask_rcnn_heads.mask_rcnn_fcn_head_v1up4convs
  RESOLUTION: 28  # (output mask resolution) default 14
  ROI_XFORM_METHOD: RoIAlign
  ROI_XFORM_RESOLUTION: 14  # default 7
  ROI_XFORM_SAMPLING_RATIO: 2  # default 0
  DILATION: 1  # default 2
  CONV_INIT: MSRAFill  # default GaussianFill
TRAIN:
  WEIGHTS: configs/pre-trained-weights/ \
           e2e_mask_rcnn_X-101-64x4d-FPN_1x \
           ___pre-trained-weights.pkl
  DATASETS: ('dsb18_train', 'dsb18_val')
  SCALES: (800,)
  MAX_SIZE: 1333
  IMS_PER_BATCH: 1
  BATCH_SIZE_PER_IM: 512
  RPN_PRE_NMS_TOP_N: 2000  # Per FPN level
TEST:
  FORCE_JSON_DATASET_EVAL: True
  DATASETS: ('dsb18_stage1_test',)
  SCALE: 800
  MAX_SIZE: 1333
  NMS: 0.5
  RPN_PRE_NMS_TOP_N: 1000  # Per FPN level
  RPN_POST_NMS_TOP_N: 1000
OUTPUT_DIR: .
```

The second configuration file is for the modified model and is detailed below:

```
MODEL:
  TYPE: generalized_rcnn
  CONV_BODY: FPN.add_fpn_ResNet101_conv5_body
  NUM_CLASSES: 2
  FASTER_RCNN: True
  MASK_ON: True
NUM_GPUS: 1
SOLVER:
  WEIGHT_DECAY: 0.0001
  LR_POLICY: steps_with_decay
  # 1x schedule (note TRAIN.IMS_PER_BATCH: 1)
  BASE_LR: 0.0025
  GAMMA: 0.1
  MAX_ITER: 180000
  STEPS: [0, 120000, 160000]
FPN:
  FPN_ON: True
  MULTILEVEL_ROIS: True
  MULTILEVEL_RPN: True
RESNETS:
  STRIDE_1X1: False
  TRANS_FUNC: bottleneck_transformation
  NUM_GROUPS: 64
  WIDTH_PER_GROUP: 4
FAST_RCNN:
  ROI_BOX_HEAD: fast_rcnn_heads.add_roi_2mlp_head
  ROI_XFORM_METHOD: RoIAlign
  ROI_XFORM_RESOLUTION: 7
  ROI_XFORM_SAMPLING_RATIO: 2
MRCNN:
  ROI_MASK_HEAD: mask_rcnn_heads.mask_rcnn_fcn_head_\
                 v1up4convs
  RESOLUTION: 28  # (output mask resolution) default 14
  ROI_XFORM_METHOD: RoIAlign
  ROI_XFORM_RESOLUTION: 14  # default 7
  ROI_XFORM_SAMPLING_RATIO: 2  # default 0
  DILATION: 1  # default 2
  CONV_INIT: MSRAFill  # default GaussianFill
  # changed
  WEIGHT_LOSS_MASK: 1.2

TRAIN:
  WEIGHTS: configs/pre-trained-weights/ \
           e2e_mask_rcnn_X-101-64x4d-FPN_1x \
            ___pre-trained-weights.pkl
  DATASETS: ('dsb18_train', 'dsb18_val')
  # Augment
  SCALES: (700, 800, 900, 980)
  MAX_SIZE: 1333
  IMS_PER_BATCH: 1
  BATCH_SIZE_PER_IM: 512
  RPN_PRE_NMS_TOP_N: 2000  # Per FPN level
  SNAPSHOT_ITERS: 10000

TEST:
  FORCE_JSON_DATASET_EVAL: True
  DATASETS: ('dsb18_stage1_test',)
  SCALE: 800
  MAX_SIZE: 1333
  NMS: 0.5
  RPN_PRE_NMS_TOP_N: 1000  # Per FPN level
  RPN_POST_NMS_TOP_N: 1000
  # add detections per image
  DETECTIONS_PER_IM: 500

OUTPUT_DIR: .
```