UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Dmytro Urukov

Improving Microscopy Image Segmentation with Object Detection

Master's Thesis (30 ECTS)

Supervisor: Mikhail Papkov, MSc

Tartu 2021

Improving Microscopy Image Segmentation with Object Detection

Abstract:
Automated analysis of microscopy images is an essential part of modern biological research. Recent advances in deep learning have greatly improved its quality and helped decrease the amount of time-consuming manual work during the experiments. Biologists are interested not only in the accurate detection of various objects (whole cells, cell organelles, tissue structures, *etc*.) but also in the high-quality segmentation of their shape. In this work, we address the problem of obtaining realistic instance segmentation masks from images with high object density. We show that combining segmentation and detection methods into a single image analysis pipeline helps efficiently separate overlapping objects and improves the segmentation quality. To reduce the complexity of this pipeline, we propose a novel CenterUNet multi-task neural network architecture that simultaneously performs object detection and semantic segmentation. We evaluate the performance of this architecture across several microscopy image domains and conduct a thorough ablation study to identify the necessary and sufficient combination of detection subtasks to solve the segmentation problem.

We believe that the results of our research provide valuable insights and can help individual practitioners as well as the image analysis industry. Our developed model may improve microscopy image segmentation pipelines at virtually zero computational cost and little integration efforts.

Keywords:
deep learning, computer vision, convolutional neural networks, object detection, instance segmentation, multi-task learning, digital microscopy

CERCS: P176 - Artificial intelligence; T111 - Imaging, image processing; B110 - Bioinformatics, medical informatics, biomathematics, biometrics

Mikroskoopiapiltide segmenteerimise parandamine objektituvastusega

Lühikokkuvõte:
Automaatne mikroskoopiapiltide analüüs on kaasaegsete bioloogiaalaste uuringute oluline osa. Selle kvaliteeti on aidanud parandada viimaste aastate arengud sügavõppes, mis muuhulgas on vähendanud ka ajamahuka manuaalse töö hulka eksperimentide läbiviimisel. Lisaks mikroskoopiapiltidel olevate objektide tuvastamisele (tervikrakud, rakuorganellid, koestruktuurid jne) pakub bioloogidele huvi ka nende kõrgekvaliteediline segmenteerimine. Käesolevas magistritöös uurime, kuidas suure objektitihedusega piltidelt realistlikke objektipõhiseid segmenteerimismaske saada. Näitame, et segmenteerimis- ja tuvastamismeetodite kombineerimine üheks töövooks aitab efektiivselt eraldada osaliselt kattuvaid objekte ning parandada segmenteerimise kvaliteeti. Sellise tööriista keerukuse vähendamiseks pakume välja uudse CenterUNet mitme-eesmärgilise arhitektuuri, mis tegeleb samaaegselt objektituvastuse ja segmenteerimisega. Me hindame selle võimekust mitmetes mikroskoopiapiltide domeenides ning viime läbi põhjaliku komponentuuringu, et identifitseerida vajalik ja piisav kombinatsioon tuvastamise alamosadest, et lahendada segmenteerimisülesanne.

Me usume, et meie uuring pakub väärt teadmisi ning aitab nii praktikuid kui ka pildianalüüsitööstust. Meie arendatud mudel võib mikroskoopiapiltide segmenteerimise töövoogude tulemusi parandada nullilähedase lisanduva arvutusliku keerukusega ja vähese integreerimisele kuluva tööga.

Võtmesõnad:
sügavõpe, tehisnägemine, konvolutsioonilised neurovõrgud, objektituvastus, objektipõhine segmenteerimine, mitme-eesmärgiline õpe, digitaalmikroskoopia

CERCS: P176 - Tehisintellekt; T111 - Pilditehnika; B110 - Bioinformaatika, meditsiiniinformaatika, biomatemaatika, biomeetrika

Contents

1 Glossary

Cell — an elementary structural and functional block of all living organisms.

Cell line — an immortalized population of cells derived from a single cell, used for biological and drug research.

Nucleus — a cell organelle present in most cells of multicellular organisms that contains the cell's genome and regulates its activities.

Glomerulus — a capillary tuft of a ball shape inside a kidney that takes part in blood filtration to produce urine.

Brightfield microscopy — the simplest form of microscopy, in which the light is either transmitted through or reflected from a biological sample.

Fluorescence microscopy — a form of microscopy, in which certain cell structures are marked with molecules that can emit light, which is captured by the microscope.

Histology — a field of biology that studies tissues.

2 Introduction

Most modern biological research methods of clinical diagnostics and drug development rely on the analysis of microscopy data [XXS+18, CRG+19]. When this analysis is done manually, biologists may need to spend a huge amount of time studying a large set of collected microscopy images. Not only is this process extremely time- and labor-consuming, but it is also vulnerable to human bias. Two researchers with a similar qualification level may produce greatly different analysis results [HHW+18]. Consequently, automated image analysis methods are becoming increasingly popular in the field as they allow processing thousands of images at a time and thus let the researchers focus more on designing new experiments [MBK+19].

The advances in artificial intelligence methods have triggered the development of numerous machine learning approaches to automated microscopy image analysis during the last decade [SG13, SPB+18]. There are now a lot of professional software tools recognized by the community (CellProfiler [MGC+18], Ilastik [BKK+19], *etc*.) that allow biologists with no computer vision background to conduct high-throughput microscopy data analysis with an accessible graphical user interface, where the user is asked to annotate a few images manually to let the software extract meaningful features from the annotated objects (*e.g*. size, intensity) and train a model. However, the latest research in deep learning has dramatically pushed the industry forward. At the moment, most of the state-of-the-art biological image analysis models are based on Convolutional Neural Networks [RFB15]. They help avoid the step of manual feature engineering and can extract and learn meaningful visual features of biological images automatically from previously annotated raw imaging data [MBK+19].

2.1 Problem

Image segmentation (separating objects from the background) is one of the essential steps in analyzing digital microscopy images. The researchers analyze the morphology and shape of the segmented objects (cells, nuclei, tissue structures, *etc*.) in order to understand their behavior, biological properties, or reaction to different treatments [CRG+19]. However, the task of accurate cell segmentation remains challenging because of frequent object clustering and overlapping, variability in shape and signal level, and the presence of artifacts on the image. So, the segmentation must not only separate the objects (cells, nuclei, *etc*.) from the background properly but also separate the cells from each other with a clear and realistic border. In the end, biologists care more about the quality of distinguishing individual cells rather than the meticulous pixel-wise labeling of the image [VBJ+19].

Biologists may also need to conduct a quantitative analysis of microscopy images. In this case, they are interested only in calculating a number of objects on an image and, perhaps, the position of their centers [XNZ18]. Modern approaches to cell de-

tection and counting also involve deep learning methods, namely the object detection architectures [HHW+18].

2.2 Motivation

Our primary motivation in this research comes from the microscopy image processing industry side. Modern domain-specific image analysis software is usually quite heavy and complicated, and including plenty of deep learning models for each specific task can make this software even heavier. In our research, we investigate the possibility of combining object detection and image segmentation models into a single multi-purpose model capable of doing both tasks well.

2.3 Contribution

In this work, we explore the methods of using object detection predictions to produce instance segmentation and separate touching and overlapping objects. We present a novel neural network architecture that performs both object detection and semantic segmentation. Moreover, its object detection output includes the orientation angle of the objects, which improves the visual interpretability of the results. Finally, we investigate whether training a network to perform both segmentation and detection tasks inside one model improves the segmentation quality.

2.4 Outline

Background gives a general overview of deep learning, image segmentation, and object detection and describes related work on these topics.

Data and methods describes the datasets used in the experiments, characterizes used deep learning methods, and explains proposed architectures and algorithms.

Experiments and results describes the conducted experiments and reports the results on all datasets.

Discussion systematically analyzes the completed work, critically reflects on the results, and lists potential directions for future work.

Conclusion shortly summarizes the contribution and key results.

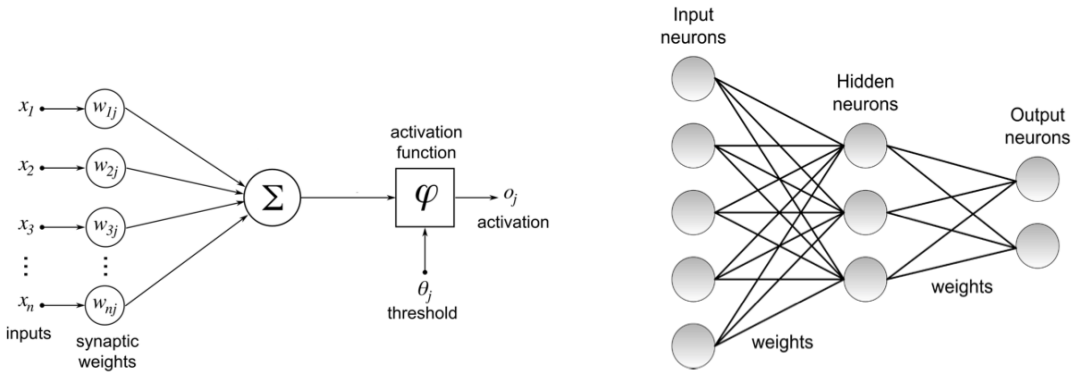
References contains a list of used literature sources.

Appendix includes supplementary figures and a public license.

3 Background

In this section, we provide an overview of deep learning for computer vision and describe the tasks of image segmentation and object detection as well as the related work in these fields.

3.1 Deep learning for computer vision



(a) Artificial neuron model.

(b) Multi-layer perceptron.

Figure 1. Perceptron [CMLBSG19].

The initial findings in the deep learning field date back to the middle of the 20th century. The first Artificial Neural Networks (ANNs) were inspired by the structure of the human brain. The neurons in our brain accumulate the input signals (electrical impulses) from other neurons; if the sum of these inputs exceeds a certain threshold, the neuron fires with an output signal and passes it further to the nearby cells. This principle was implemented in the McCulloch-Pitts model of an artificial neuron for simple computations [MP43].

The perceptron algorithm proposed by Frank Rosenblatt [Ros58] utilizes the ideas behind the McCulloch-Pitts neuron model to create a neuron as a computing unit, initially designed for a binary classification problem. In essence, perceptron receives multiple values as an input, which are multiplied with certain weights. The result is summed up and passed to an activation function, which produces the output (Figure 1a). A single-unit perceptron can deal only with linearly separable data, so the neurons can be connected and grouped into layers, forming a multi-layer perceptron (MLP) that is able to work with linearly non-separable data and learn more complex patterns in data. The layers are fully connected, meaning that each output of the current layer is connected with each input of the next layer (Figure 1b). The weights of the MLP are based on the training data. They are calculated during an iterative training process using a backpropagation

algorithm [RHW86], which includes two phases - a forward pass and a backward pass. During the forward pass, the input is passed through the network till the end using current weights and bias terms, and an output is produced. A loss is calculated between a predicted value and a ground truth value using a task-specific loss function. During the backward pass, the error is propagated back through the network. Each weight or bias value is adjusted using the value of the loss partial derivative with respect to this weight or bias to minimize the error value. This process is called gradient descent.

The aim of deep learning in computer vision tasks is to simulate human vision. However, unlike our eyes, fully connected networks like MLP do not capture any spatial features on an image. Moreover, the number of connections between the neurons of fully connected layers required to process an image of a decent size is extremely big, which makes these networks computationally expensive and hard to train when applied to images. However, Convolutional Neural Networks (CNNs) solve this issue by using a set of small trainable filters, which are used when an image is passed through a number of convolution operations. Stacking multiple layers of convolutional filters makes the network notice particular spatial structures on an image. These structures become more and more complex with the network depth increasing, starting from simple lines and edges to more complex features and spatial patterns.

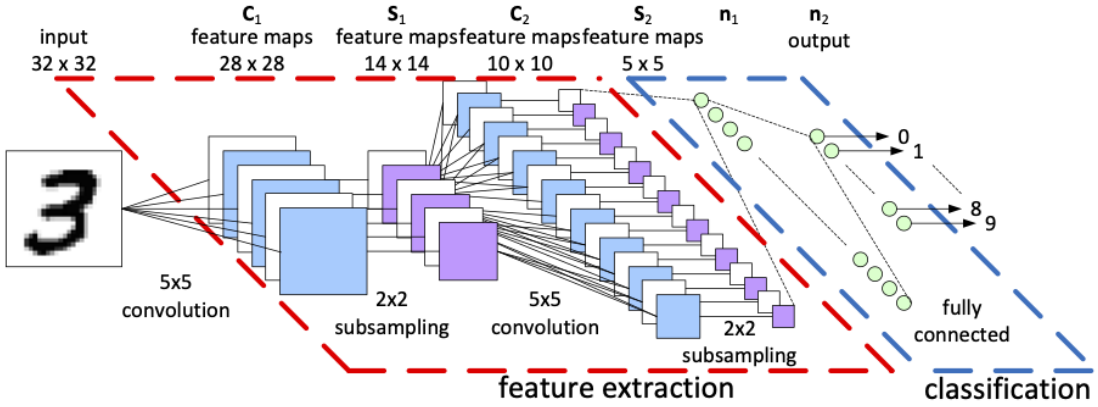


Figure 2. Convolutional neural network architecture for handwritten digit classification task [PMC11].

A typical CNN usually consists of convolutional, normalization, pooling, and activation layers. Convolutional layers extract meaningful features from an image. Pooling layers gradually decrease the dimensionality of the network and "distill" the extracted features. Normalization layers center and standardize the intermediate state of the network to improve convergence. Activation layers add non-linearity to the training. We refer to this part of the architecture as "feature extractor", "body", or "backbone", it is usually task-independent. The final layers (the network's "head") are explicitly designed

for each task. For example, when performing image classification, convolved and pooled features are flattened into a fully connected (dense) layer and connected to another dense layer with the number of nodes equal to the number of predicted classes. A sample CNN architecture for image classification is shown inFigure 2.

One of the first CNN architectures similar to the ones used nowadays was proposed back in 1998 by Yann LeCun to recognize handwritten characters [LBBH98]. At that time, the processing units required to train CNNs were yet too slow and expensive. However, things have changed to better a decade later with the spread of consumer graphical processing units (GPUs), capable of running thousands of operations in parallel. An introduction of AlexNet [KSH12] architecture for image classification in 2012 led to an explosive growth of research in neural networks for computer vision. This network was trained on an extensive ImageNet [DDS+09] dataset and introduced several new techniques like dropout layers [SHK+14] and various data augmentations [SK19] to decrease overfitting. Modern architectures also incorporate the ideas of residual blocks [HZRS15] and inception modules [SLJ+14] which let the networks improve the performance by adding more layers. As a result, nowadays CNNs usually contain tens of stacked layers with millions of trainable parameters and easily outperform humans in many computer vision tasks.

Convolutional neural networks are used to solve different tasks in computer vision, some of the most researched ones include classification, object detection, and image segmentation (Figure 3).

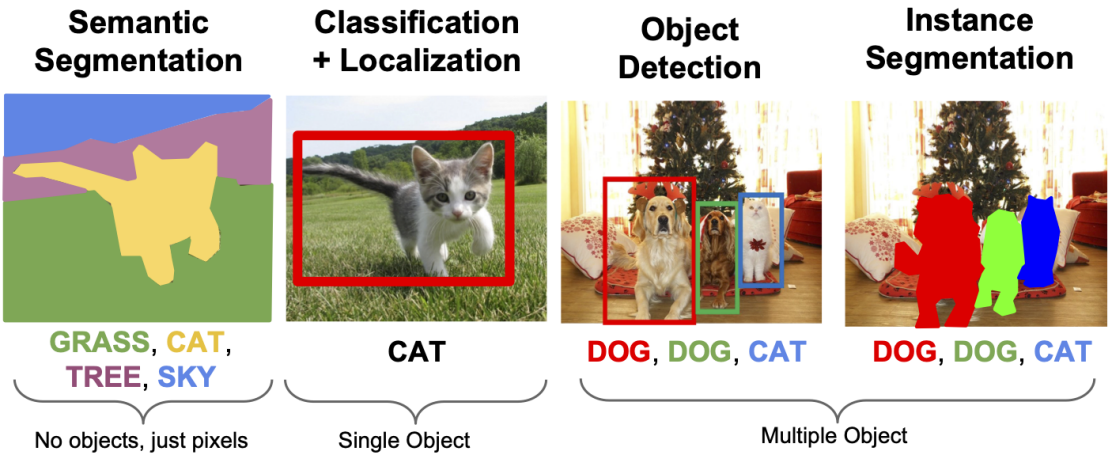


Figure 3. Popular tasks in modern computer vision [LJY17].

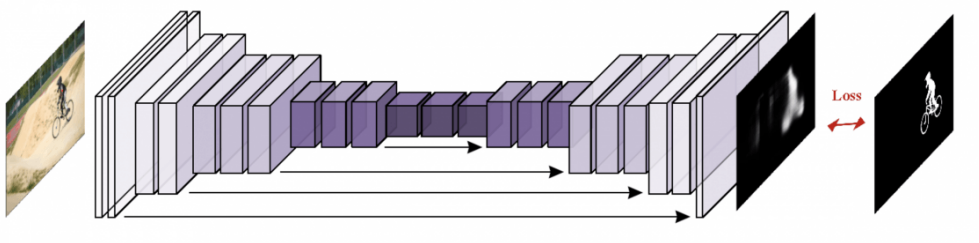


Figure 4. Fully convolutional architecture for semantic segmentation [MC19, WSS18].

3.2 Image segmentation

Image segmentation is a task of assigning to each pixel of the image a certain class value (Figure 3). Semantic segmentation is used to separate a foreground from a background. If there are multiple objects of the same class on an image, their pixels will be labeled with the same value. If all objects on an image belong to the same class (*e.g.* cells of a single cell line), they will be labeled with a single value producing a binary mask. By contrast, instance segmentation is used to distinguish individual objects on an image and label each object uniquely.

Image segmentation is used in many research areas, including autonomous driving, remote sensing, aerial and satellite image analysis, and medical imaging [MBP+20].

CNNs have become a basis for research in automated biological image processing during the last decade and quickly outperformed the traditional morphological and machine learning methods for cell nuclei analysis [SPB+18]. The research was greatly boosted by the introduction of Fully Convolutional Networks (FCN) [LSD15] , which can take images of arbitrary size as an input and produce pixel-wise segmentation masks of the same size as an output (Figure 4).

The FCN models became a core of conventional pipelines for cell nuclei segmentation, but they still require a large amount of training data to achieve a good segmentation quality. However, acquiring more training data for biological research requires conducting complex experiments and is extremely labor-consuming both for the generation of the raw images and expert annotation compared to the outside world images. In essence, there is no public dataset of the size comparable to ImageNet [DDS+09] available for biological researchers nowadays. This obstacle was addressed by introducing U-Net [RFB15] architecture that was designed specifically for biomedical image segmentation. As the name suggests, the network consists of a downsampling encoder and an upsampling decoder connected in a U-like shape. It also incorporated the idea of skip-connections [DVC+16] between the encoder and decoder used to increase the network's segmentation performance.

3.3 Object detection

Object detection is a multi-task research problem, which combines both a task of predicting a bounding box around each target object (regression problem) and a task of predicting the class of the object in a bounding box (classification problem).

Most of the neural network architectures used in object detection can be divided into two groups: one-stage detectors and two-stage detectors. In general, two-stage detectors are known to have higher object detection metrics values and a huge number of parameters, while one-stage detectors are usually lighter in size and number of parameters and thus are performing faster in terms of inference speed [LOW+19].

3.3.1 Two-stage detectors

Two-stage detectors, as their name suggests, perform the object detection in two sequential iterations (Figure 5). In the first stage, the network produces a set of class-agnostic region proposals (bounding boxes). In the second stage, the network predicts the classes for these region proposals and performs the final bounding boxes regression [JZL+19]. The most popular representatives of two-stage detectors are the Region-based CNNs, which evolved from R-CNN [GDDM14] to Fast R-CNN [Gir15] and Faster R-CNN [RHGS16].

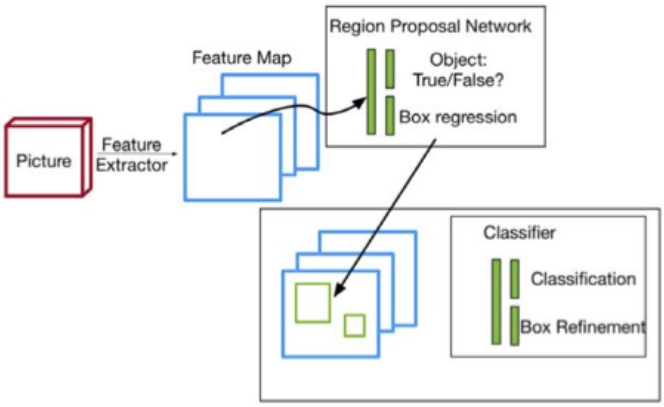


Figure 5. General architecture of two-stage detectors [NBG19].

3.3.2 One-stage detectors

On the contrary, one-stage detectors [JZL+19] perform bounding box regression and object class prediction from the input image in a single pass (Figure 6). The detection speed can be so high that one-stage detectors can even be used for the real-time video stream processing of 30+ frames per second (*e.g.* a stream from a digital microscope to detect and count cells on the fly [FC20]). As these models are usually more lightweight, the

inference can be performed even on small devices like smartphones [JYG+20]. YOLO (You Only Look Once) [RDGF16] family of architectures is one of the most popular one-stage detectors known for its high inference speed. The network outputs a list of object box proposals, which is passed to a Non-Maximum Suppression (NMS) algorithm [RDGF16] that selects most probable boxes based on objectness score. The YOLO architecture has gradually evolved into YOLOv4 [BWL20] and YOLOv5 [JSB+20] networks that demonstrate increased object detection performance and incredible inference speed.

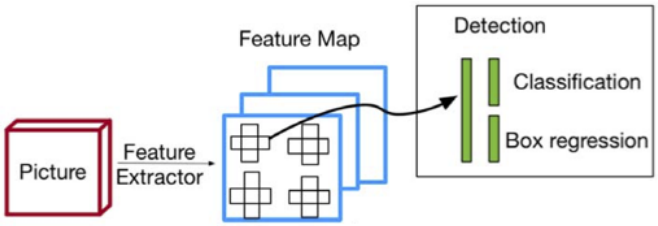


Figure 6. General architecture of one-stage anchor-based detectors [NBG19].

However, using NMS-like algorithms as a postprocessing step over model prediction results means that the network is not trained fully end-to-end. Moreover, computing anchor boxes and filtering them out requires excessive computational resources and may not guarantee high accuracy if the objects in the dataset have very different shapes [YFG20]. For these reasons, the anchor-free one-stage detectors like CornerNet [LD19] and CenterNet [ZWK19] have recently gained popularity. Instead of the anchor box approach, CornerNet uses top-left and bottom-right corners to represent the bounding box and predicts the position of these two corners using a heatmap approach. Alternatively, CenterNet uses the center of the object as the detection center and regresses the object width and height from it (Figure 7).

keypoint heatmap [C] local offset [2] object size [2]

Figure 7. Outputs of different heads of CenterNet [ZWK19].

3.3.3 Rotated object detection

(a) Axis-aligned bounding boxes (b) Rotated bounding boxes

Figure 8. Two types of bounding boxes [TZD+17].

Most of the traditional object detection methods use bounding boxes and encode the object position on an image using four variables: center point coordinates (x, y), object width, and object height (Figure 8a). These axis-aligned boxes are well suitable to display objects that usually appear in the same orientation, *e.g.* people and traffic signs on the streets are usually aligned vertically from an autonomous driving car viewpoint.

However, the axis-aligned representation poses a problem when dealing with target objects that can appear on an image in arbitrary orientation, like satellite (Figure 8), aerial or digital microscopy photos. In this case, four parameters are not enough to describe a target object with high precision. The size of a predicted bounding box does not reflect a physical size of a rotated target object on such images. Moreover, a background takes more than half of the box area when the target object is rotated at about 45 degrees. When the target objects are located densely, the predicted bounding boxes may also include parts of other objects, making them hard to distinguish and analyze (Figure 8a). Moreover, it can make training of a model less stable, as the neural network will be confused by the high presence of background and other objects in a ground truth bounding box.

A concept of a rotated bounding box formulated in [LPL17] addresses the mentioned problems by adding an angle parameter to a definition of a bounding box to identify its orientation. Compared to an axis-aligned box, the size of a rotated box reflects a physical size of a target object in all orientations, includes fewer background pixels, and can help separate tightly located objects like cars on a parking lot (Figure 8b).

Modern rotated object detection architectures are mostly based on traditional axis-aligned object detectors and modify them to detect arbitrarily-rotated objects. For example, SCRDet [YYY+19] is based on two-stage Faster-RCNN [RHGS16], while

more novel R^3Det [YYFH20] is based on one-stage anchor-based RetinaNet [LGG+18].

3.4 Multi-task learning

A concept of multi-task learning can be formulated as training a model to perform multiple high-level tasks in parallel and produce multiple results during the inference. An example of such learning can be combining object detection and instance segmentation in one model. For each detected bounding box produced by the detection branch, the segmentation branch produces a pixel-wise instance segmentation of an object inside this box. Some models like Mask-RCNN [HGDG18] use this approach but focus on instance segmentation only. However, adding a segmentation head to a popular one-stage anchor-based object detector RetinaNet [LGG+18] during the training can help improve the object detection quality [FSB19]. One-stage CentripetalNet [DLL+20] also gains an improvement of object detection quality after adding an instance segmentation head to an anchor-free detector. It is hypothesized that ground truth pixel-wise annotation masks and bounding boxes give the model more information where to extract the features from compared to using bounding boxes only.

4 Data and methods

In this section, we describe the microscopy image datasets used in our experiments and the preprocessing steps applied to them. Then, we explain the methods used for microscopy image segmentation and object detection, including the detection of rotated objects. Finally, we discuss our approaches to combine semantic segmentation and object detection in one pipeline and describe a proposed neural network architecture that enables such a combination.

4.1 Datasets

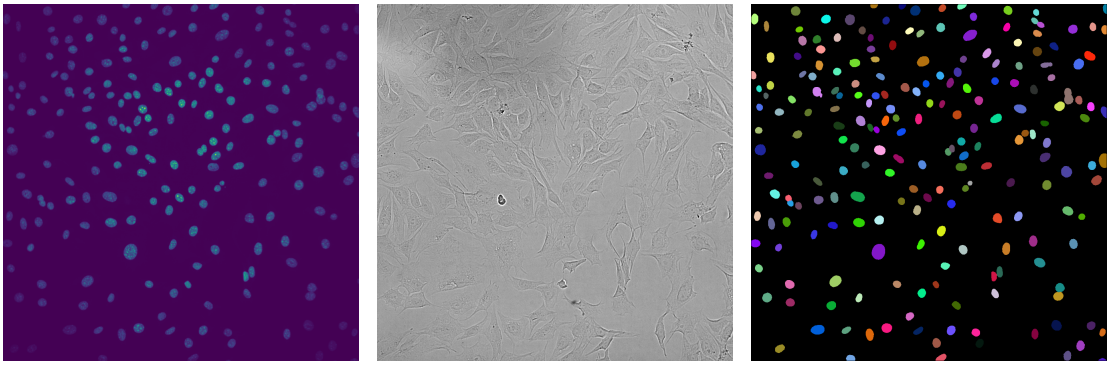
We have used four datasets from different microscopy image domains. The first two datasets were provided by PerkinElmer, Inc. and contain cell images in brightfield and fluorescent modalities along with annotated nuclei masks. We futher denote these datasets as Seven cell lines Brightfield and Seven cell lines Fluorescent respectively. The third dataset was collected from a Kaggle "Data Science Bowl 2018" competition, it contains fluorescent images from multiple diverse experiments. The fourth dataset comes from a Kaggle "HuBMAP - Hacking the Kidney" competition and includes kidney histology images with annotated glomeruli masks.

4.1.1 PerkinElmer Seven cell lines

PerkinElmer Seven cell lines (7CL) dataset consists of 3024 microscope images of 1080x1080 pixels provided to the University of Tartu by PerkinElmer Inc. for research and experiments during an ongoing cooperation project. The images contain cells of different cell lines: A549, HT1080, HeLa, HepG2, MCF7, MDCK, NIH-3T3. Each image contains cells of exactly one cell line, and there are 432 images of each cell line. The average number of cells on each image is different, with HepG2 images being the most densely populated, having lots of touching and overlapping nuclei (Figure 10).

Each image is presented in two modalities: brightfield (Figure 9b), which contains human-visible cells and fluorescent (Figure 9a), which contains cell nuclei stained with a Hoechst 33342 dye. The ground truth object-wise segmentation masks (Figure 9c) were obtained from fluorescent modality images using PerkinElmer Harmony software [Mas15]. We treat the images in fluorescent and brightfield modalities as individual datasets and conduct all experiments with then separately. We use identical ground truth nuclei masks for both datasets as a target for training.

We have randomly split the original annotated dataset into training (2/3, 2016 images), validation (1/6, 504 images), and testing (1/6, 504 images) datasets.



(a) Fluorescent modality (b) Brightfield modality (c) Ground truth mask

Figure 9. Example image from Seven cell lines dataset.

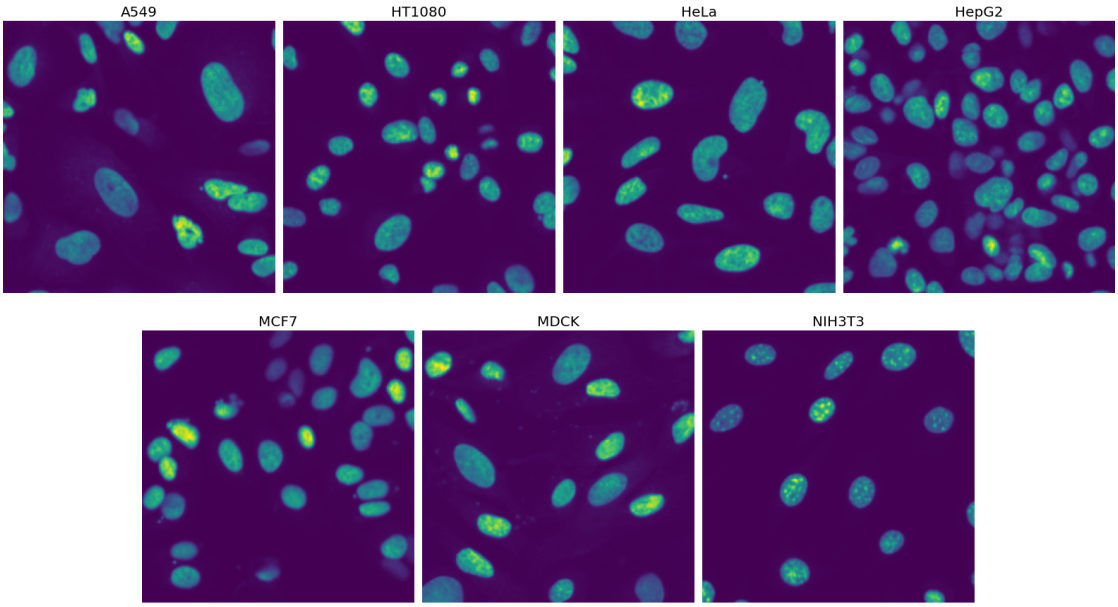
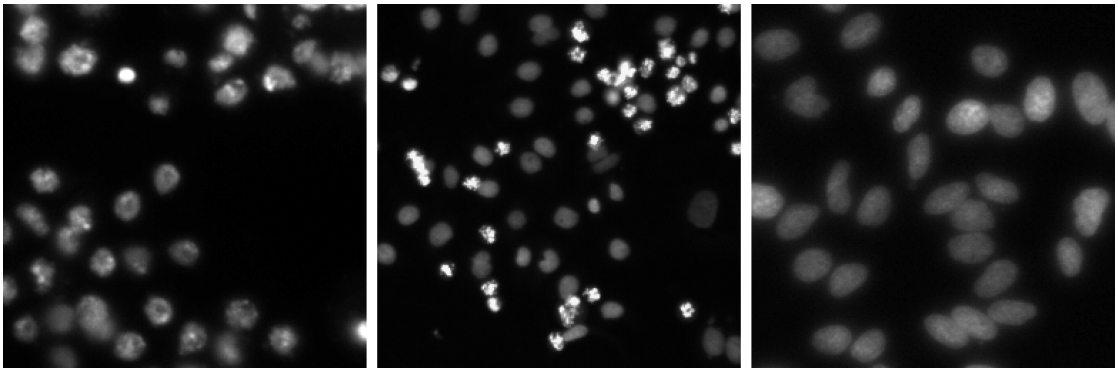


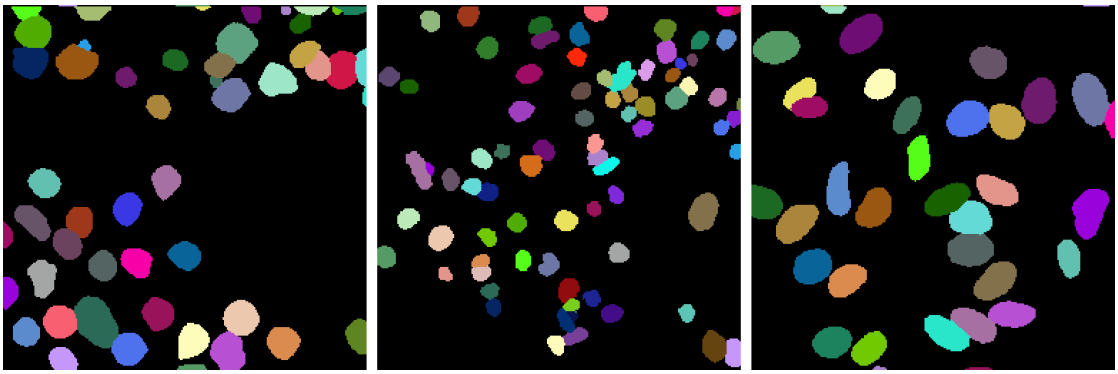
Figure 10. Zoomed examples of images that contain cells of different cell lines from PerkinElmer Seven cell lines Fluorescent dataset.

4.1.2 Data Science Bowl 2018

Cell nuclei Data Science Bowl 2018 (DSB2018) dataset was collected from the training and testing datasets of the "Data Science Bowl 2018" Kaggle competition [CGK+19] on nuclei segmentation. The dataset contains images and annotated object-wise segmentation masks from 30 different experiments with various resolutions, cell types, image modalities, and magnifications (Figure 11). We use only fluorescent images for our



(a) Images



(b) Ground truth masks

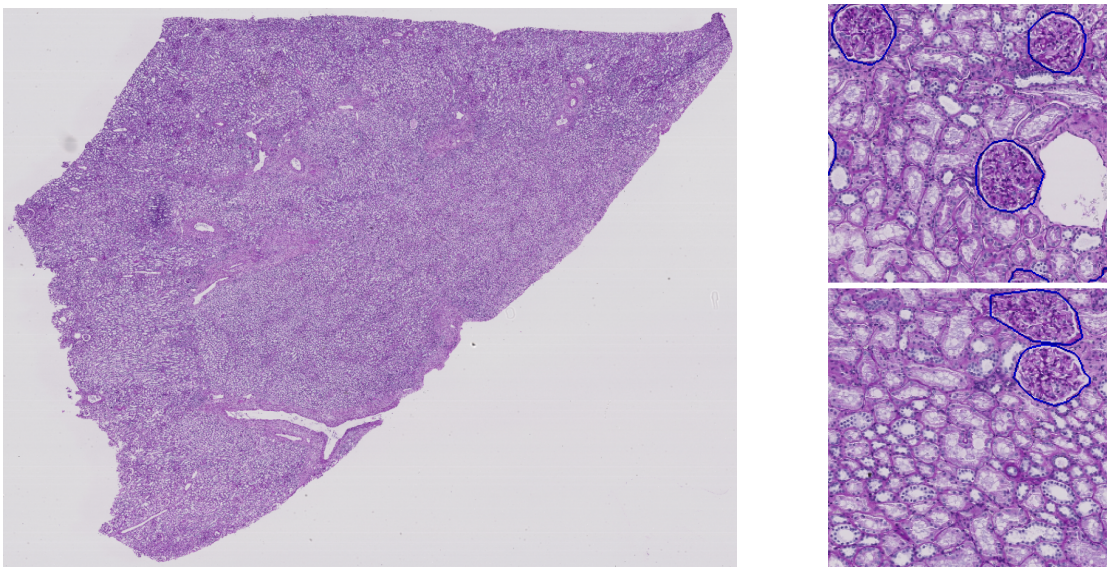
Figure 11. Example fluorescent images and respective object-wise masks from Data Science Bowl 2018 dataset.

experiments, which constitute a majority of the images in this dataset (81%).

The original dataset consists of 599 fluorescent images for training and 53 images for testing. We have used a subset of the training set as a validation set, resulting in a split of the original dataset into training (546 images), validation (53 images), and testing (53 images) datasets.

4.1.3 HuBMAP Kidney glomeruli

Images of the HuBMAP Kidney glomeruli dataset come from a "HuBMAP - Hacking the Kidney" Kaggle competition about segmenting glomeruli in human kidney tissue images. The dataset includes 15 kidney images in a very high resolution with object-wise segmentation masks of glomeruli provided. The file sizes of the images are huge (500MB - 5GB), so, in order to process them efficiently, we downscale them to 25% of the original resolution and crop the downscaled images into 256×256 px patches, obtaining 9580



(a) (b)

Figure 12. (a) Example full resolution image from HuBMAP Kidney glomeruli dataset. (b) Small patches containing glomeruli (contoured in blue).

small images from the tissue regions of 15 original images (Figure 12).

We have used 11 original full resolution images for training, 2 for validation, and 2 for testing.

4.1.4 Preprocessing

We have applied some simple data preprocessing techniques to our data to make further neural network training work properly. Firstly, we have scaled image pixel values from integer 0-255 range into a float 0-1 range. Secondly, we have applied standard scaling separately by subtracting the channel-wise mean and dividing by the channel-wise standard deviation computed on the training subset of each dataset.

When training on large images, we randomly crop 256×256 px (DSB2018) or 512×512 px (7CL) regions from different regions of the original images in order to fit more various samples into the batch and thus make the training more stable. The selected size of the crop depends on the original image size.

4.2 Microscopy image segmentation

Following the general trend of numerous medical segmentation papers and competitions [ZRC19, CGK+19], we have chosen the U-Net architecture [RFB15] to segment medical images.

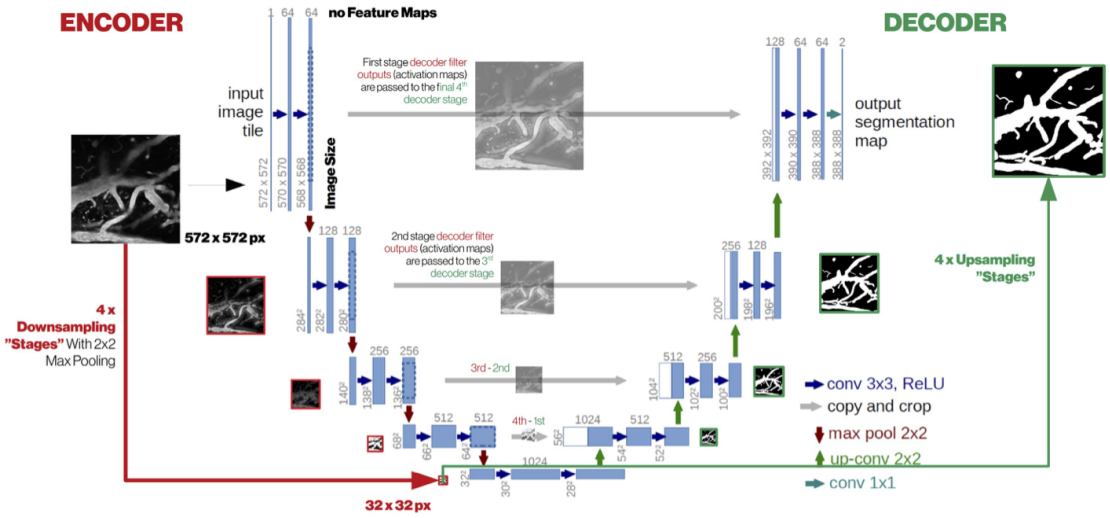


Figure 13. Original U-Net architecture [Tei19, RFB15].

The original U-Net architecture was designed specifically for precise semantic segmentation in dense scenes and was first used to segment biomedical images. The network has symmetric U-like shape and consists of two parts: the contracting path (encoder) on the left and the expansive path (decoder) on the right (Figure 13). The encoder part consists of traditional convolutional blocks (Conv 3x3 - ReLU - Conv 3x3 - ReLU-MaxPool 2x2) to downsample the image and extract the features into low-resolution feature maps. However, the decoder part replaces MaxPool 2x2 operations with an UpConv 2x2 ("up-convolutions") to gradually upscale the feature maps and produces a segmentation mask. In order to help recover small details from the downscaled feature maps, each layer of the decoder is connected to a respective layer of the encoder with a skip connection (concatenation).

U-Net is a fully-convolutional network, so it can process images of arbitrary size, given that their dimensions are divisible by 2^k, where k is equal to the "depth" (number of downsampling layers) of the network. It is usually achieved by padding the input image to the nearest valid size with zeros or using reflective padding.

Modern researchers do not usually implement the U-Net architecture exactly as it is shown inFigure 13. One of the most frequent modifications is a use of custom encoder, *e.g.* ResNet [HZRS15] or EfficientNet [TL20] without the last fully-connected classification layer. Using an encoder with a lower number of parameters makes the model lighter while using a heavier encoder can help extract features better and gain a higher segmentation quality. These modifications help achieve a trade-off between model weight and segmentation quality and incorporate the latest findings in neural network architectures.

We have built our U-Net segmentation pipeline using PyTorch [PGM+19] deep learning framework for Python and Segmentation Models PyTorch [Yak20] library that contains a high-level API for building custom segmentation models and includes implementations of various encoders and decoders. We use a ResNet50 [HZRS15] encoder as motivated by an encoder ablation study (Section 5.4) and a conventional U-Net decoder of depth 5. We initialize an encoder with weights pre-trained on ImageNet as this is known to benefit model performance [RZKB19], even though the medical image domain is far from the natural images domain in the ImageNet dataset.

We use a Dice loss function [SLV+17] for training:

$$\mathcal{L}_{segm} = 1 - \frac{2 \times TP}{(TP + FP) + (TP + FN)} \tag{1}$$

4.3 Object detection

Here we describe two neural network architectures we have used for object detection on microscopy images: a one-stage anchor-based YOLO model and a one-stage anchor-free CenterNet model. We briefly explain their structure and key principles, discuss their advantages and disadvantages and describe the implementations we have used.

4.3.1 YOLO

YOLO (You Only Look Once) [RDGF16] is a popular one-stage anchor-based object detection model, capable of doing real-time detection even on mobile devices.

Generally, YOLO architecture can be divided into two main components: feature extractor and detector (Figure 14). For example, YOLOv3 uses a Darknet-53 feature extractor, which has 53 3x3 and 1x1 convolutional layers and skip-connections in between, inspired by ResNet [HZRS15]. The encoder is followed by 53 more layers of a detector, which gradually upscales the feature map. The detection is done on multiple scales (Figure 14): at layer 82 (where feature map is 1/32 size of the original image), layer 94 (1/16 size), and final layer 106 (1/8 size), which helps detect objects of large, medium and small size respectively.

In order to predict bounding box coordinates and a class probability of the object on one of 3 scales, the image is divided into a grid of cells. Each cell predicts a number of bounding boxes (*e.g.* three) using so-called anchor boxes, which are allocated on a pre-defined grid with various aspect ratios that incorporate prior knowledge about the objects in the dataset. The predicted values for each box are its center coordinates (x, y), size (width & height), confidence score (probability that there is an object inside the box), and a list of class probabilities. For example, if our input image is of size 416x416, then the size of the grid at layer 82 is 13x13. If we use 3 anchor boxes for every scale,

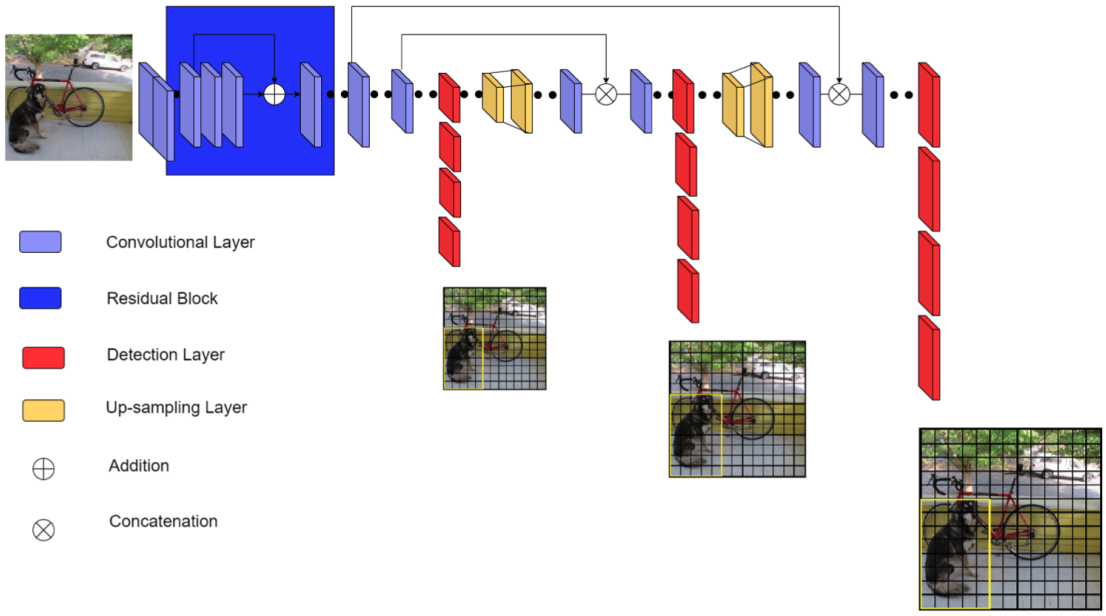


Figure 14. YOLOv3 architecture [Val20].

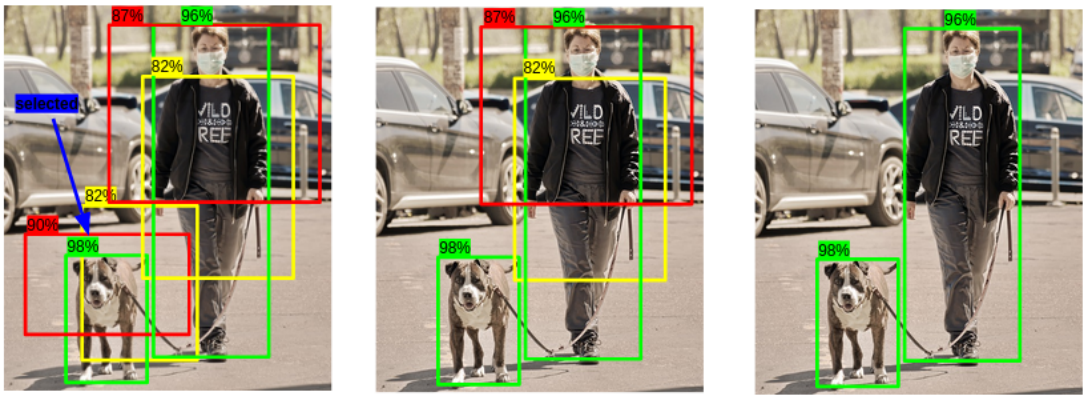


Figure 15. Non-Maximum Suppression applied to YOLO predictions [Sin20].

and our dataset has 80 classes (*e.g.* COCO [LMB+15]), then, the output shape of the detection is $(13, 13, 3 \cdot (5 + 80))$.

Given that YOLO predicts bounding boxes for each cell in the grid, there is a

high chance that there will be multiple overlapping bounding boxes predicted for the same object. In order to find the best bounding boxes, YOLO uses a Non-Maximum Suppression (NMS) algorithm [RDGF16]. First, it ignores all predicted boxes with a confidence score below a certain threshold. Then, the remaining boxes are sorted by a confidence score. Finally, for each object, the algorithm filters out the boxes that have a non-maximum confidence score and overlap with the most probable box more than a specified threshold (Figure 15).

We have used Ultralytics YOLOv5 [JSB+20] implementation in PyTorch, which is an improved version of YOLOv3 [JYG+20] with a more advanced encoder. We chose the YOLOv5l ("large") model among four available models (small, medium, large, extra-large). It contains 47M parameters and offers a good trade-off between inference speed and detection quality.

4.3.2 CenterNet

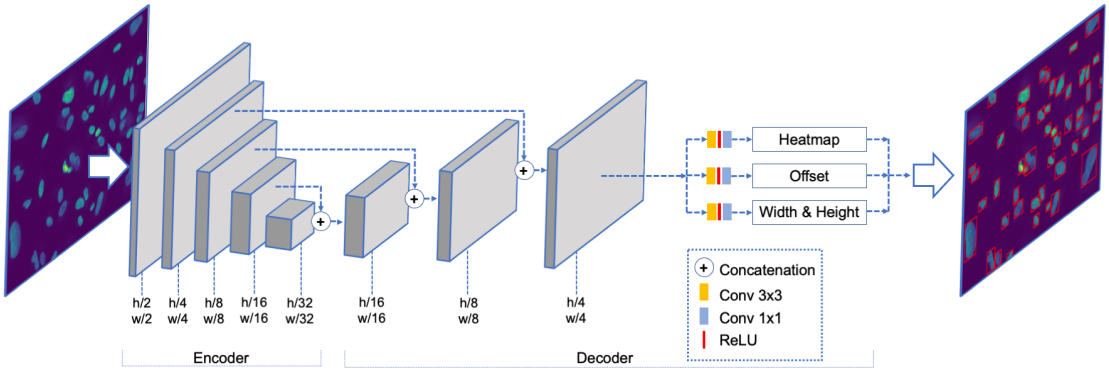
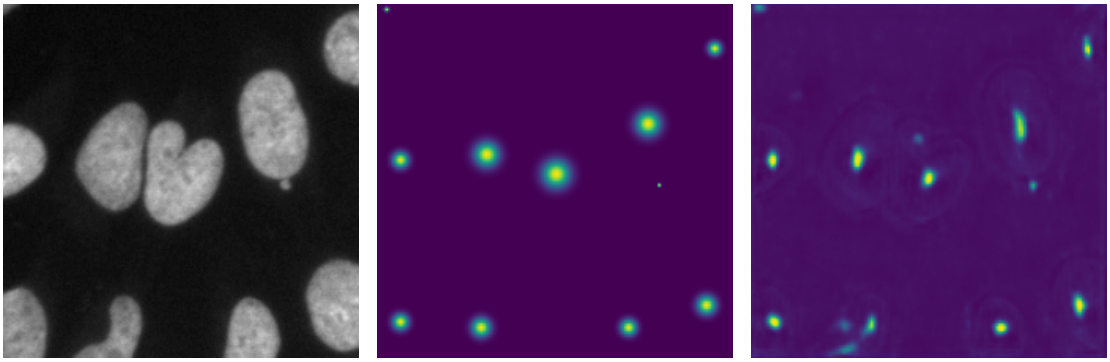


Figure 16. CenterNet architecture.

The anchor-based neural networks like YOLO have a very good performance on medium and large objects but may have problems detecting small and tiny objects like cell nuclei. For example, RetinaNet has a minimum anchor size of 32×32 px, meaning that smaller objects may be undetected. YOLOv3 implementation allows calculating best anchor box sizes for a given dataset prior to training a model but is still very sensitive to these values.

CenterNet [ZWK19] architecture uses a different approach: it treats objects as points and formulates object detection as a task of detecting the center point of an object on a heatmap (Figure 17). The peak values on the heatmap indicate the centers of the objects, and other parameters of the bounding box (width, height, and center offset) are regressed directly from each predicted center. The values in the peaks of the heatmap are used as confidence scores, which can be thresholded later on.



(a) (b) (c)

Figure 17. (a) Cropped raw image from Data Science Bowl 2018 dataset. (b) Ground-truth center heatmap. (c) Predicted center heatmap.

A CenterNet architecture consists of an encoder and a decoder (Figure 16). An encoder can be a classical feature extractor like ResNet50 without the last fully connected layer. A decoder upscales the extracted features and produces a feature map with a size of 1/4 of the input image (which speeds up the training and inference without losing much of a performance). A feature map is passed into three separate heads:

- heatmap head: used to detect object centers from peaks;
- width/height head: used to regress object size out of heatmap peaks;
- offset head: contains offset for center coordinates used to compensate 4x heatmap downscaling.

One of the features of CenterNet is a use of deformable convolutions [DQX+17] in a decoder. Unlike the traditional convolutional layers, which use a static kernel as a sliding window during a convolution operation, deformable convolutions utilize a trainable matrix of location offsets, which specify the locations of pixels to sample from (Figure 18).

The heatmap center prediction (Figure 17c) is trained to match a ground truth mask (Figure 17b) generated with a Gaussian kernel described in [LD19]. The training is performed using a focal loss [LGG+18]:

$$\mathcal{L}_{center} = -\frac{1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} & \\ \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases} \tag{2}$$

where α and β are loss hyperparameters, and N is a number of keypoints (centers) on an image. We choose $\alpha = 2$ and $\beta = 4$.

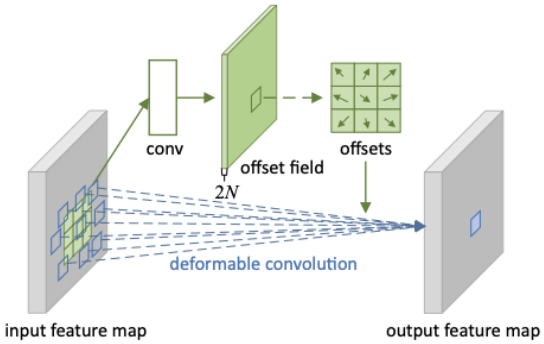


Figure 18. Illustration of 3x3 deformable convolution [DQX+17].

The regression of object properties (width, height, center offset) is performed using an L1 loss at heatmap center points:

$$\mathcal{L}_{size} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{S}_{p_k} - s_k \right| \tag{3}$$

The general training objective is a sum of all losses with different weights, formulated as:

$$\mathcal{L}_{det} = L_{center} + \lambda_{size} L_{size} + \lambda_{off} L_{off} \tag{4}$$

We set width & height loss weight $\lambda_{size} = 0.2$ and center offset loss weight $\lambda_{off} = 1$.

We use CenterNet-better [Wan20] implementation of CenterNet in PyTorch [PGM+19], which is a slightly enhanced and well-structurized version of the original CenterNet [ZWK19] as a basis for our experiments. We use a ResNet50 [HZRS15] encoder, as motivated by an ablation study described in Section 5.4.

4.4 Detecting rotated objects with CenterNet

In Section 3.3.3 we have explained why detecting rotated boxes enhances the visual quality and usability of the predictions compared to detecting classical axis-aligned boxes. This method has recently gained a lot of popularity for aerial and satellite image analysis, and we applied it to microscopy image analysis for similar reasons. Firstly, nuclei can be located very densely on the image, and axis-aligned boxes cannot accurately separate them without overlapping (Figure 20). Secondly, nuclei can appear in arbitrary orientations, and some axis-aligned boxes may contain too many background pixels (Figure 21). Using rotated bounding boxes decreases the number of background pixels and improves the visual interpretability of the results, especially in images with crowded objects.

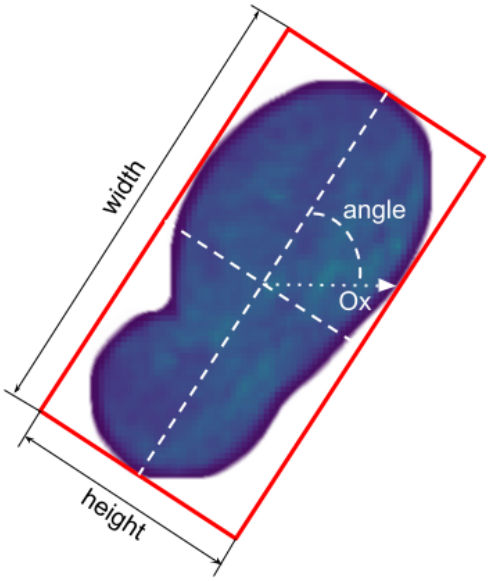


Figure 19. Rotated bounding box width, height and angle coding.

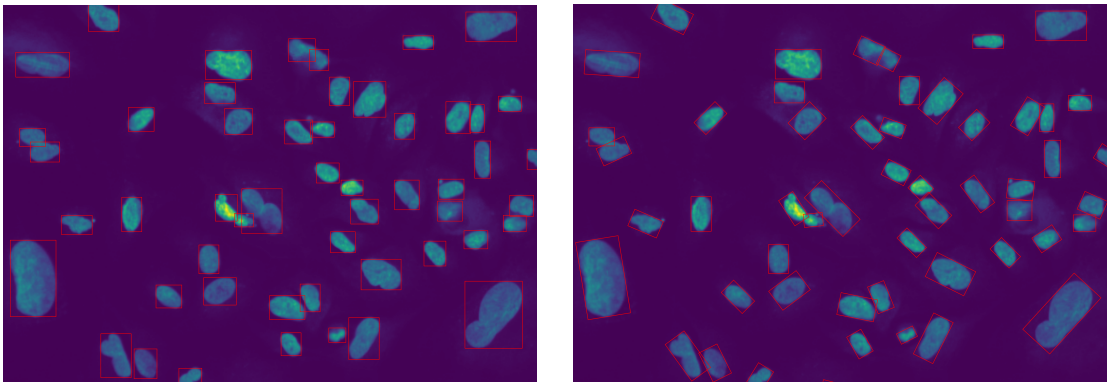
In order to detect rotated objects on digital microscopy images, we modify the CenterNet [Wan20] architecture described in Section 4.3.2. One of the key changes is a modified generation of ground truth bounding boxes. For each object in the dataset, we use a ground truth segmentation mask and calculate a minimum rotated rectangle (rectangle with a minimum area that includes object mask) using Shapely [G$^+$07] library for manipulation with geometric objects. Instead of using width and height along the axes, we take the longest dimension of the minimum rotated rectangle as width and the shortest as height. We calculate an object rotation angle as an angle between the horizontal axis Ox and the longest axis of the rectangle (Figure 19). The new width and height are stored in pixels, and the rotation angle is stored in radians within $[0, \pi)$.

We have added an additional head for rotation angle regression analogous to the width & height head and predict a single value of the rotation angle in radians for each object (Figure 22). Analogous to Equation (3), an L1 loss is used to train an angle prediction head:

$$\mathcal{L}_{angle} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{\Theta}_{p_k} - \theta_k \right| \tag{5}$$

The general training objective is now formulated as:

$$\mathcal{L}_{rdet} = L_{center} + \lambda_{size} L_{size} + \lambda_{off} L_{off} + \lambda_{angle} L_{angle} \tag{6}$$



(a) (b)

Figure 20. (a) Axis-aligned GT bounding boxes. (b) Rotated GT bounding boxes. Zoomed sample from 7CL dataset, fluorescent modality.

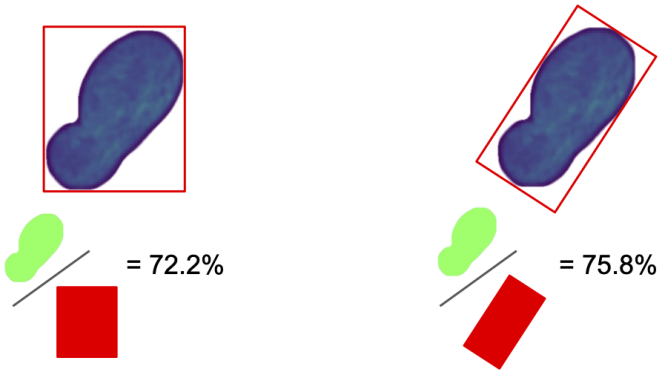


Figure 21. Mean ratio between GT mask area and axis-aligned/rotated bounging box calculated over all objects of PerkinElmer 7CL Fluorescent dataset.

We set the center offset loss weight $\lambda_{off} = 1$, width & height loss weight $\lambda_{size} = 2$ and angle loss weight $\lambda_{size} = 2$.

4.5 From semantic to instance segmentation

U-Net model described in Section 4.2 takes a raw image as an input and produces a semantic segmentation of the objects on it, separating the nuclei from the background. However, as we mentioned in Section 3.2, biologists are also interested in separating individual nuclei that are touching and form a clump. Ignoring clumps may lead to an undercounting and improper analysis of nuclei morphological features on the images with a high density of nuclei, like the images of HepG2 cell line in PerkinElmer 7CL

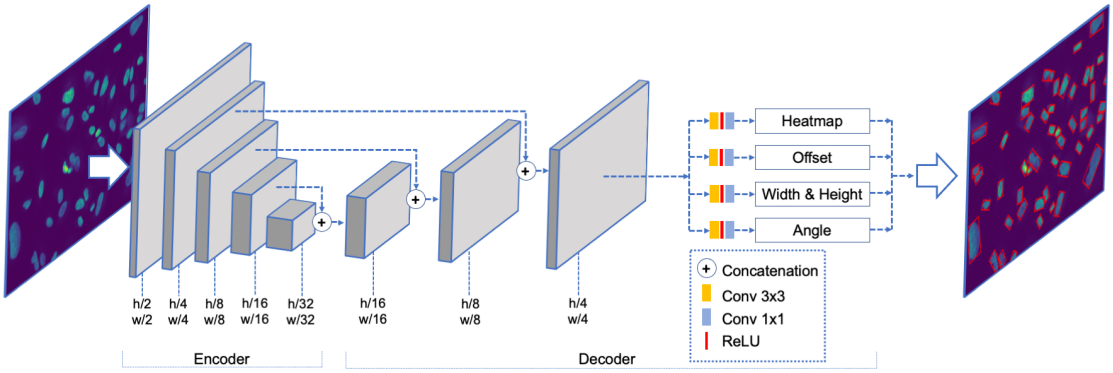


Figure 22. CenterNet architecture with added object angle regression head.

dataset (Figure 10).

Most of the modern methods of separating the touching nuclei are based on a watershed algorithm [RM01], which treats an image as a sloping terrain with hills and valleys, and the height of the terrain depends on the pixel/mask values. The "water" is poured from the starting points (markers) provided as input until the water flows meet each other and denote separating lines. The classic approach to select the markers is by calculating a negative distance map of a binary segmentation mask and "flooding" from regional minima. However, there were many modifications that build upon it and improve its quality [AKWJ19, KŻS+20]. A totally alternative approach of separating nuclei is by predicting a separate output channel with nuclei borders and subtracting it from the nuclei segmentation channel during the postprocessing stage [CQY+17].

In our implementation, we separate the touching nuclei by using an output of both semantic segmentation and object detection models, inspired by [YGF+20]. Our object detection evaluation (Section 5.5) show that object detection models perform well at detecting individual nuclei even in nuclei clumps. Thus, we hypothesize that using these models' output can significantly improve the instance segmentation and let us separate individual nuclei in the scenes with high object density.

The process of combining models' predictions is illustrated inFigure 23. We train the models separately and run the inference in parallel. In the first step, both object detection (YOLO or CenterNet) and instance segmentation model (U-Net) take a raw image as an input and make independent predictions. In the second step, coordinates of nuclei centers are derived from the bounding boxes produced by the object detection model. On the last step, these center coordinates are used as seeds (markers) for a watershed algorithm [RM01] which draws borders between touching cells and returns a labeled mask with individual nuclei. We use a scikit-image [vdWSN+14] implementation of watershed algorithm in our experiments.

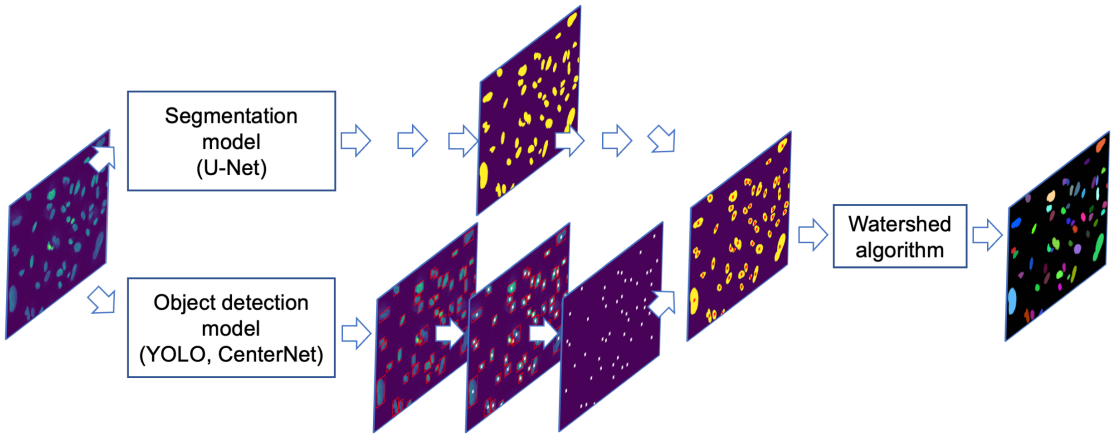


Figure 23. Combining models' predictions and separating individual nuclei with watershed algorithm.

4.6 CenterUNet: adding segmentation to CenterNet

The method of using two separate models to produce instance segmentation described above has an evident drawback: we have to train and run two networks separately. Even though the inference speed does not decrease too much when adding an object detector like YOLOv5 into the segmentation pipeline, we still have to keep both networks in GPU memory. Therefore, it is clear that it would be more optimal to train a single network capable of doing both object detection and instance segmentation in a single pass.

One of the approaches used nowadays to combine both object detection and instance segmentation in a single model was proposed by the authors of Mask R-CNN [HGDG18], where they add a segmentation branch on top of a two-stage Faster R-CNN [RHGS16] object detector. Each proposed bounding box is treated as a Region of Interest (RoI), and its coordinates are used to crop (pool) a small region out of a feature map using a RoIAlign operation [HGDG18], which also resizes the pooled region of an arbitrary rectangular size into a square of a fixed size (*e.g.* 14×14 px). This resized RoI is passed into a small segmentation head, which produces a segmentation mask of the object inside the region and resizes it back to the shape of the original RoI. Some novel architectures are built on top of single-stage anchor-free object detectors [DLL$^+$20, FSB19], which report good instance segmentation performance on COCO [LMB$^+$15] dataset.

U-Net (Figure 13) and CenterNet (Figure 22) architectures are very similar and are built on shared principles. They have the same encoder design with five layers and a very similar decoder except that U-Net has five layers in the decoder, while CenterNet has only three. The output of the CenterNet decoder, which is connected to detection heads, has the size of 1/4 of the input image: as mentioned in Section 4.3.2, it helps speed up the inference significantly. However, to produce full-size segmentation masks, we add two

more layers to the decoder and connect its output to a segmentation head that generates semantic segmentation masks (Figure 24). Added decoder layers have the same structure as the existing ones, including deformable convolutions described in Section 4.3.2 inside them. We hypothesize that segmentation and detection heads will not disrupt each other during the training. As mentioned in Section 3.4, we expect exactly the opposite: to see an improvement of segmentation quality due to the effects of multi-task learning, since we give the network more information compared to training separate models.

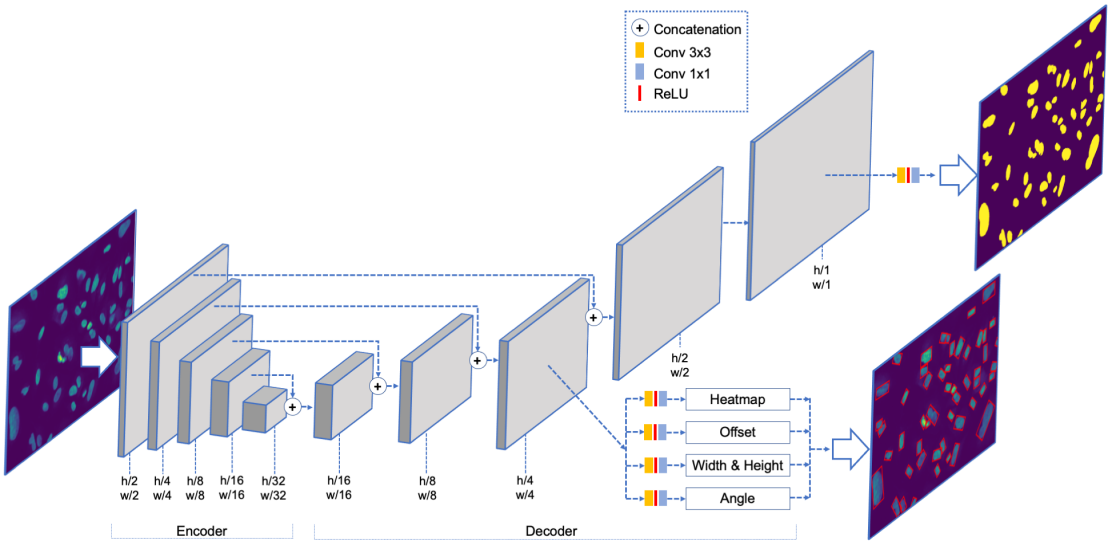


Figure 24. CenterUNet architecture.

We use a Dice loss as in (1) to train a segmentation branch, and object detection branch is trained to optimize a complex detection loss (6).

A general training objective of CenterUNet is formulated as:

$$\mathcal{L}_{comb} = L_{center} + \lambda_{size} L_{size} + \lambda_{off} L_{off} + \lambda_{angle} L_{angle} + \lambda_{segm} L_{segm} \tag{7}$$

where $\lambda_{size} = 2$, $\lambda_{angle} = 2$, $\lambda_{off} = 1$, $\lambda_{segm} = 1$.

The segmentation head of CenterUNet produces a binary semantic segmentation mask, while the detection head produces rotated bounding box coordinates. We follow the same approach of combining segmentation and detection predictions described in Section 4.5, and generate a labeled instance segmentation mask using a watershed algorithm. The complete architecture of CenterUNet with watershed postprocessing is shown inFigure 25.

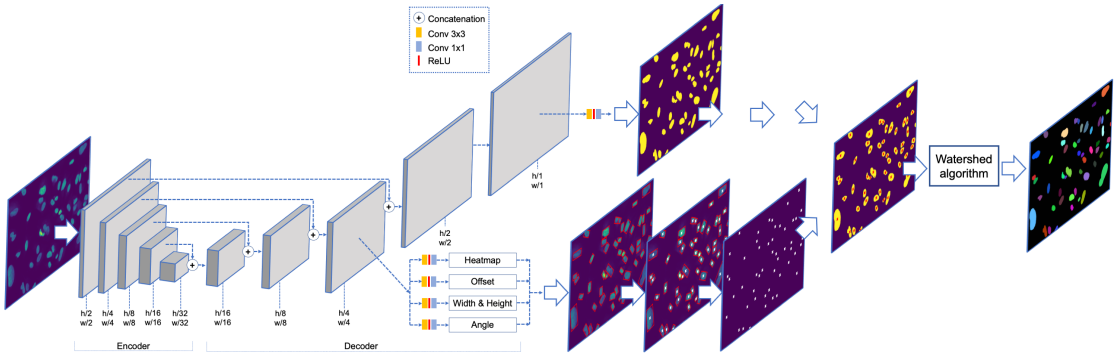


Figure 25. CenterUNet architecture with watershed postprocessing.

5 Experiments and results

In this section, we describe the neural network training parameters, define the detection and segmentation metrics used for evaluation, motivate a choice of an architecture for the encoder and report the results for rotated object detection and instance segmentation tasks.

5.1 Neural network training

U-Net We trained the model until convergence using a Dice loss and an Adam optimizer with 3×10^{-4} learning rate (reduced gradually on plateau up to 1×10^{-8}) and 1×10^{-4} weight decay. We used batch size 16 for PerkinElmer 7CL Fluorescent and Brightfield datasets and batch size 32 for Data Science Bowl and HuBMAP Kidney glomeruli datasets.

YOLO The model training follows the default training strategy available in the implementation [JSB+20] with GIoU and binary cross-entropy losses. We used SGD optimizer with 1×10^{-2} initial learning rate and 5×10^{-4} weight decay. The other parameters matched the default configuration as in the original YOLOv5 implementation [JSB+20].

CenterNet & CenterUNet We trained the models until convergence using focal loss for center heatmap head, L1 losses for center offset, width/height and angle heads, and Dice loss for segmentation head (CenterUNet). We used initial learning rate of 5×10^{-3} with scheduled decrease to 5×10^{-6} and a 1×10^{-4} weight decay. Analogous to training U-Net, we used batch size 16 for PerkinElmer 7CL Fluorescent and Brightfield datasets and batch size 32 for Data Science Bowl and HuBMAP Kidney glomeruli datasets.

The number of training batches for all of the models did not exceed 20000. We have used Weights & Biases [Bie20] to track the training experiments and compare their results. The experiments and models' training was conducted using the resources of HPC Center of the University of Tartu and their NVIDIA Tesla P100 32GB GPU.

5.2 Detection metrics

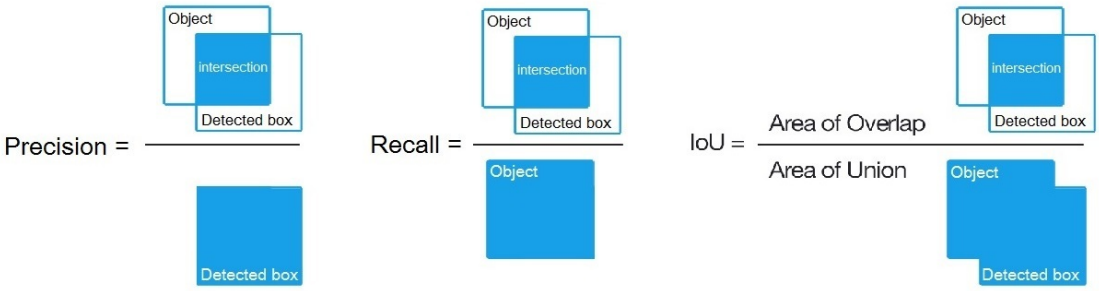


Figure 26. Precision, recall and IoU visualization [BWL20].

We evaluate the prediction of object detection models based on specific metrics, which evaluate how close the predicted bounding boxes are to the ground-truth boxes. A basis of these metrics is a value of Intersection over Union (Figure 26), which indicates the degree of overlap between the boxes and is defined as:

$$IoU = \frac{A \cap B}{A \cup B} = \frac{prediction \cap target}{prediction \cup target} \tag{8}$$

We use Shapely [G+07] library to compute the IoU between predicted and ground-truth rotated boxes numerically.

In order to compute the object detection metrics, we iterate over the ground-truth bounding boxes and compare them with predicted bounding boxes, counting the number of true positives (TP; predicted and GT boxes overlap with an IoU value above a certain threshold), false positives (FP; the predicted box does not overlap with the GT box or the IoU value is too low), false negatives (FN; the GT object does not overlap with the predicted object or the value is too low). The higher the IoU threshold is used for evaluation, the lower number of true positives will be calculated for the image (Figure 27).

We can compute the precision and recall values using the obtained statistics as:

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

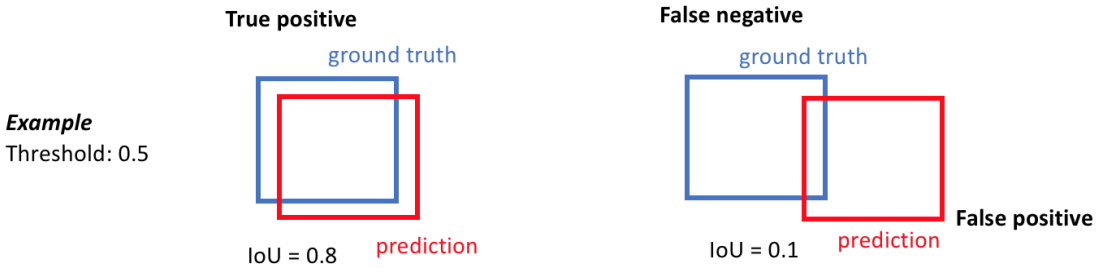


Figure 27. IoU thresholding for object detection evaluation [Jor18].

In practice, using a single IoU threshold value to calculate TP, FP, and FN values for precision and recall is not enough to evaluate the quality of the object detection model. In our experiments, we use a popular COCO Average Precision (AP) metric [LMB+15], computed by calculating the precision using 11 different thresholds from 0.5 to 0.95 with step 0.05 and averaging the result. We also report AP values with a fixed 0.5 IoU threshold (AP50) and a 0.75 IoU threshold (AP75).

5.3 Segmentation metrics

We evaluate the segmentation performance of the models using specific metrics that show how close the predicted segmentation mask is to the ground-truth segmentation masks.

Firstly, we evaluate a semantic segmentation performance of the model, *i.e.* its ability to separate the foreground and background. We use a set of pixel-wise metrics, meaning that TP, FP and FN values are computed per each pixel on the masks. We report a pixel-wise intersection over union (PW IoU) value calculated analogous to (8) as:

$$IoU = \frac{prediction \cap target}{prediction \cup target} = \frac{TP}{TP + FP + FN} \tag{11}$$

We also report a pixel-wise F1 (PW F1) score calculated as:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{12}$$

Secondly, we evaluate an instance segmentation performance of the model — its ability to separate foreground and background and also separate individual objects. To do that, we first compute TP, FP, and FN values for each object, using 11 different IoU thresholds (0.5...0.95) analogous to the calculation of COCO AP for the object detection. Using these values, we report the object-wise IoU value (OW IoU) computed analogous to (11) and object-wise F1 score (OW F1) analogous to (12) for all IoU thresholds and averaging the results.

5.4 Encoder choice

As described in Section 4.2, an encoder part of a traditional U-Net is often customized and replaced with a pretrained backbone, namely by one of the popular classification architectures, which is used as a feature extractor. In order to choose the best encoder for further experiments, we have compared the performance of a few popular encoder architectures available in the Segmentation Models Pytorch [Yak20] library. We evaluate the encoders on the PerkinElmer 7CL Brightfield dataset (Section 4.1.1), as it appears to be the toughest for the models on both tasks [FSM+19].

We have evaluated a rotated object detection quality using an output of a rotated object detection branch of CenterUNet (Figure 24) described in Section 4.4. We report rotated bounding box AP, AP50 and AP75 values for different encoder architectures (Table 1).

Table 1. Rotated object detection results of CenterUNet on PerkinElmer 7CL Brightfield dataset.

Encoder	Params	AP	AP50	AP75
ResNet50 [HZRS15]	23M	24.40	71.55	6.65
SE-ResNeXt50 [HSA+19]	25M	24.64	72.08	7.11
EfficientNet-B2 [TL20]	7M	16.82	58.99	2.71
MobileNetV2 [SHZ+19]	2M	20.41	64.63	4.61

We have also evaluated segmentation quality using an output of a segmentation branch of CenterUNet (Figure 25) with chosen encoders. To ensure a correct comparison, we do not use watershed postprocessing to combine the detection and segmentation predictions described in Section 4.5. We report the PW IoU, PW F1, OW IoU, OW F1 and the inference time of the models with these encoders (Table 2)

Table 2. Segmentation results of CenterUNet on PerkinElmer 7CL Brightfield dataset (without watershed postprocessing).

Encoder	PW IoU	PW F1	OW IoU	OW F1	Inference, ms
ResNet50 [HZRS15]	0.592	0.742	0.267	0.330	71
SE-ResNeXt50 [HSA+19]	0.588	0.738	0.261	0.313	80
EfficientNet-B2 [TL20]	0.512	0.675	0.196	0.225	70
MobileNetV2 [SHZ+19]	0.551	0.709	0.222	0.265	56

The primary goal of our experiments is to improve the segmentation performance, so for our further experiments, we have chosen a ResNet50 [HZRS15] encoder that

performs best in a segmentation task and is second-best in a rotated object detection task, being slightly behind the SE-ResNeXt50 [HSA+19] encoder.

5.5 Rotated object detection

We have evaluated the performance of the rotated object detection branch of CenterUNet on different datasets and reported the AP, AP50, and AP75 values (Table 3). We have also visualized the predicted rotated bounding boxes vs. ground truth rotated bounding boxes (Figure 28).

Table 3. Rotated object detection results on different datasets.

Dataset	AP	AP50	AP75
PerkinElmer 7CL Fluorescent	53.46	97.46	51.10
PerkinElmer 7CL Brightfield	24.40	71.55	6.65
Data Science Bowl 2018	17.53	51.04	9.00
Hubmap Kidney glomeruli	53.63	88.63	58.36

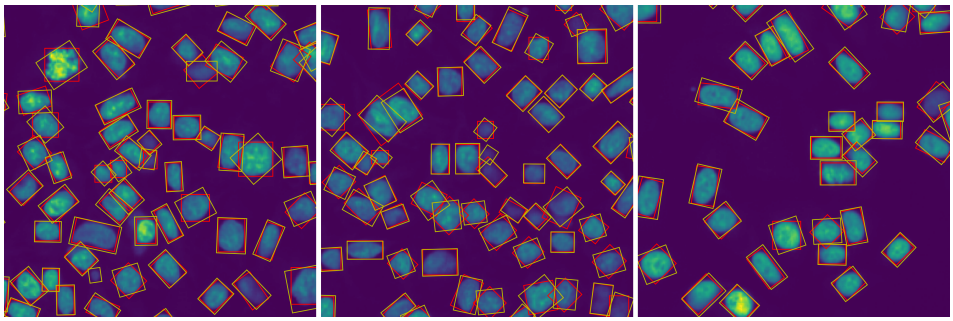
5.6 Improving segmentation with object detection

In this section, we describe our instance segmentation experiments, report the evaluation results, and investigate the multi-task learning effects inside CenterUNet.

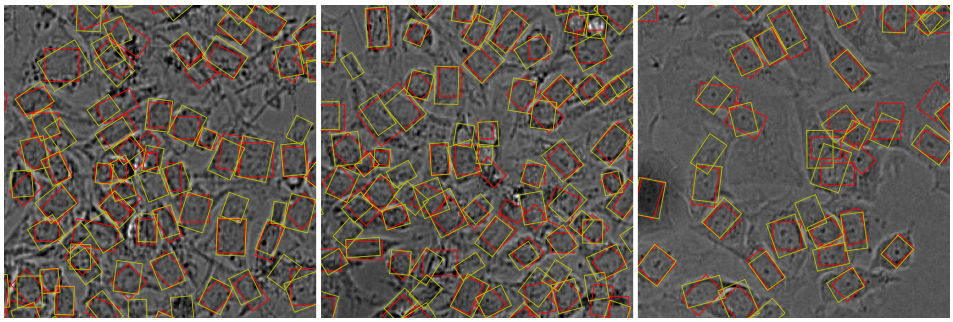
5.6.1 Instance segmentation results

We have evaluated different methods of combining semantic segmentation and object detection for the task of instance segmentation. As a baseline, we have used an experimental setup when only U-Net (Section 4.2) is used, and the instance segmentation is produced by using a label function from scikit-image [vdWSN+14], which labels pixels of individual objects different integer value for each object. The second experiment setup included combining the predictions of U-Net and YOLO models using a watershed algorithm (Section 4.5), and in the third experiment, we combined U-Net and CenterNet in the same way. The fourth experiment setup included combining predictions from segmentation and detection branches of CenterUNet. We have reported the results of these experiments for all datasets: PerkinElmer 7CL Fluorescent (Table 4), PerkinElmer 7CL Brightfield (Table 6), Data Science Bowl 2018 (Table 8), and HuBMAP Kidney glomeruli (Table 10).

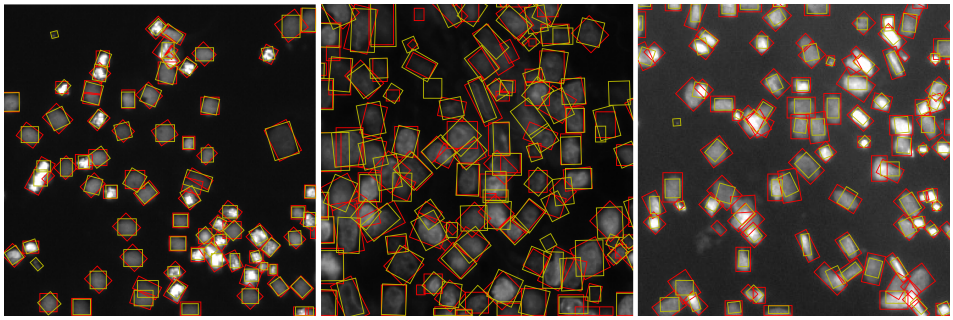
We have also evaluated whether training an object detection branch of CenterUNet (Section 4.6) improves the instance segmentation quality. In our first experimental setup,



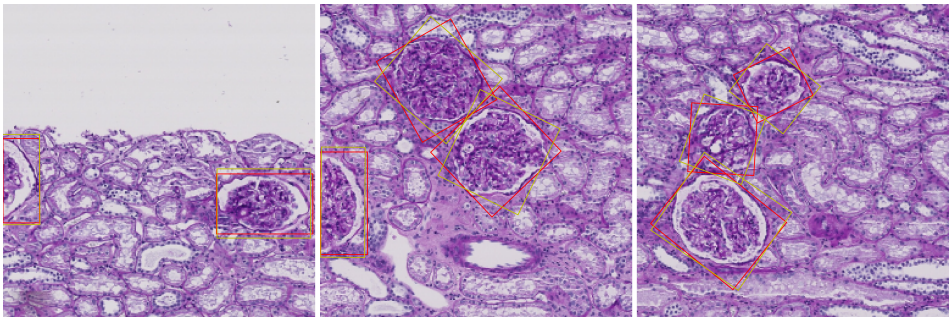
(a) PerkinElmer 7CL Fluorescent dataset



(b) PerkinElmer 7CL Brightfield dataset



(c) Data Science Bowl 2018 dataset



(d) HuBMAP Kidney glomeruli dataset

Figure 28. Rotated object detection results on different datasets. Zoomed crops from test images. Red bounding boxes - ground truth, yellow bounding boxes - predictions.

we evaluated CenterUNet with a segmentation branch only, "turning off" a training detection branch using zero loss weights. In the second experiment, we have turned it back on, and in the third experiment, we have combined predictions using a watershed algorithm. We have reported the instance segmentation metrics for all datasets: PerkinElmer 7CL Fluorescent (Table 5), PerkinElmer 7CL Brightfield (Table 7), Data Science Bowl 2018 (Table 9), and HuBMAP Kidney glomeruli (Table 11).

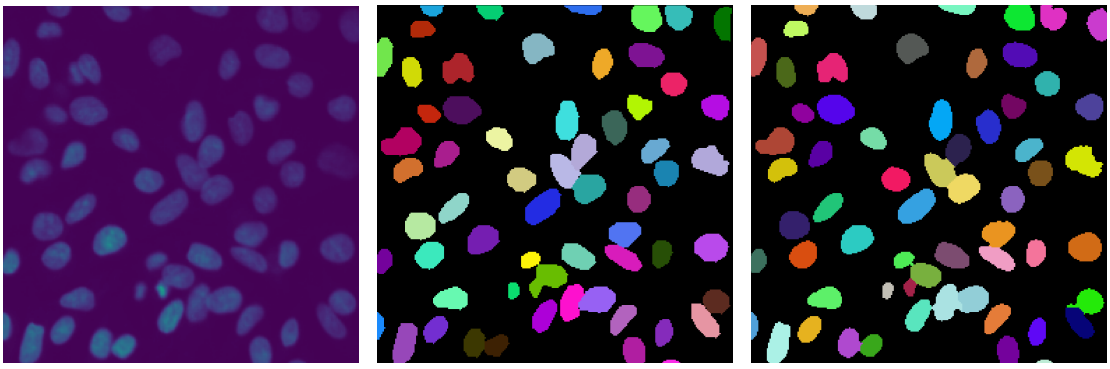
Finally, we have visualized the predicted instance segmentation masks vs. ground-truth masks for all datasets: PerkinElmer 7CL Fluorescent (Figure 29), PerkinElmer 7CL Brightfield (Figure 30), Data Science Bowl 2018 (Figure 31), and HuBMAP Kidney glomeruli (Figure 32).

Table 4. Segmentation results on PerkinElmer 7CL Fluorescent dataset.

Model	Watershed	PW IoU	PW F1	OW IoU	OW F1	Inference, ms
U-Net		0.958	0.967	0.806	0.827	39
U-Net + YOLO	✓	0.958	0.967	0.825	0.843	75
U-Net + CenterNet	✓	0.958	0.967	0.832	0.858	94
CenterUNet	✓	0.930	0.963	0.827	0.867	70

Table 5. CenterUNet performance on PerkinElmer 7CL Fluorescent dataset.

Model	Detection	Watershed	PW IoU	PW F1	OW IoU	OW F1
CenterUNet			0.932	0.965	0.788	0.845
CenterUNet	✓		0.931	0.964	0.787	0.845
CenterUNet	✓	✓	0.930	0.963	0.827	0.867



(a) Raw cropped image (b) Ground truth mask (c) Predicted mask

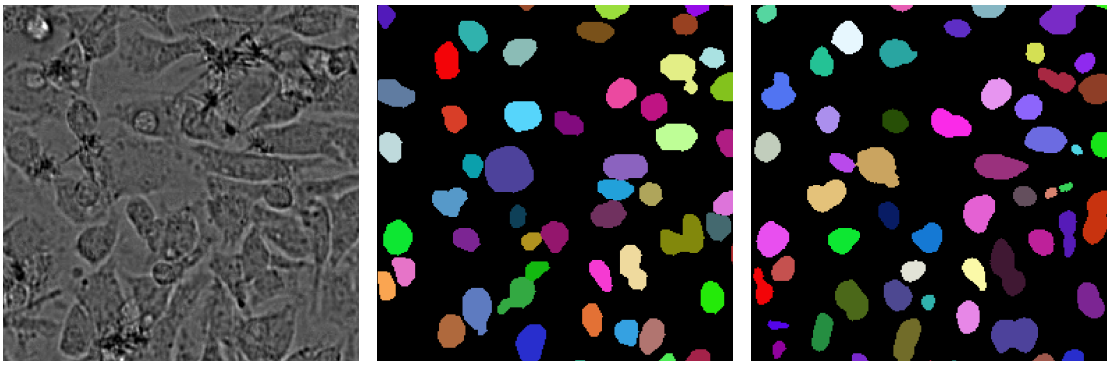
Figure 29. Zoomed example of CenterUNet segmentation predictions with watershed postprocessing on PerkinElmer 7CL Fluorescent dataset.

Table 6. Segmentation results on PerkinElmer 7CL Brightfield dataset.

Model	Watershed	PW IoU	PW F1	OW IoU	OW F1	Inference, ms
U-Net		0.635	0.763	0.320	0.398	40
U-Net + YOLO	✓	0.635	0.763	0.329	0.405	75
U-Net + CenterNet	✓	0.635	0.763	0.317	0.401	94
CenterUNet	✓	0.592	0.742	0.269	0.325	71

Table 7. CenterUNet performance on PerkinElmer 7CL Brightfield dataset.

Model	Detection	Watershed	PW IoU	PW F1	OW IoU	OW F1
CenterUNet			0.569	0.723	0.239	0.298
CenterUNet	✓		0.592	0.742	0.267	0.320
CenterUNet	✓	✓	0.592	0.742	0.269	0.325



(a) Raw cropped image (b) Ground truth mask (c) Predicted mask

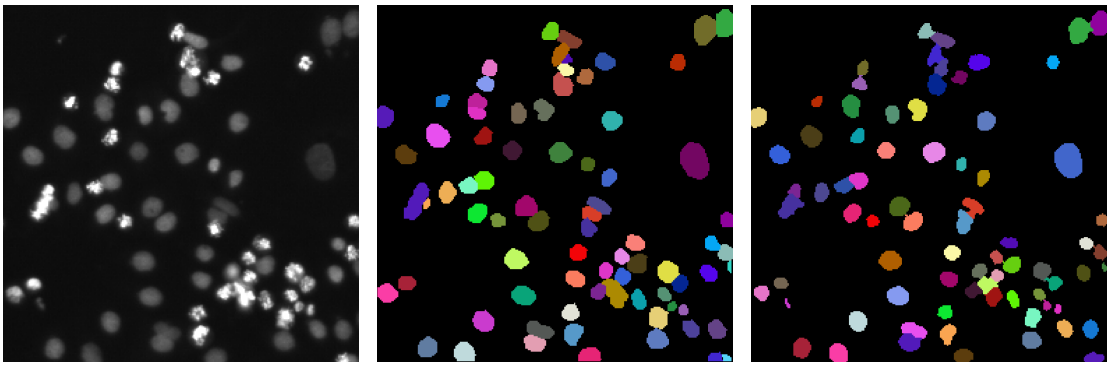
Figure 30. Zoomed example of CenterUNet segmentation predictions with watershed postprocessing on PerkinElmer 7CL Brightfield dataset.

Table 8. Segmentation results on Data Science Bowl 2018 dataset.

Model	Watershed	PW IoU	PW F1	OW IoU	OW F1	Inference, ms
U-Net		0.761	0.848	0.409	0.468	14
U-Net + YOLO	✓	0.761	0.848	0.397	0.483	32
U-Net + CenterNet	✓	0.761	0.848	0.391	0.477	41
CenterUNet	✓	0.739	0.839	0.390	0.471	13

Table 9. CenterUNet performance on Data Science Bowl 2018 dataset.

Model	Detection	Watershed	PW IoU	PW F1	OW IoU	OW F1
CenterUNet			0.788	0.879	0.376	0.418
CenterUNet	✓		0.739	0.839	0.343	0.426
CenterUNet	✓	✓	0.739	0.839	0.390	0.471



(a) Raw cropped image (b) Ground truth mask (c) Predicted mask

Figure 31. Zoomed example of CenterUNet segmentation predictions with watershed postprocessing on Data Science Bowl 2018 dataset.

Table 10. Segmentation results on HuBMAP Kidney glomeruli dataset.

Model	Watershed	PW IoU	PW F1	OW IoU	OW F1	Inference, ms
U-Net		0.823	0.880	0.342	0.370	13
U-Net + YOLO	✓	0.823	0.880	0.351	0.383	33
U-Net + CenterNet	✓	0.823	0.880	0.348	0.379	41
CenterUNet	✓	0.804	0.837	0.339	0.366	11

Table 11. CenterUNet performance on HuBMAP Kidney glomeruli dataset.

Model	Detection	Watershed	PW IoU	PW F1	OW IoU	OW F1
CenterUNet			0.846	0.876	0.341	0.381
CenterUNet	✓		0.804	0.837	0.330	0.358
CenterUNet	✓	✓	0.804	0.837	0.339	0.366

(a) Raw cropped image (b) Ground truth mask (c) Predicted mask

Figure 32. Zoomed example of CenterUNet segmentation predictions with watershed postprocessing on HuBMAP Kidney glomeruli dataset.

5.6.2 Multi-task learning

We have investigated which components of the object detection branch of CenterUNet (Figure 24) influence the segmentation branch performance as a result of multi-task learning. We report the results of different training experiments with different object heads switched on or off (Table 12) on the PerkinElmer 7CL Brightfield dataset.

Table 12. CenterUNet performance on PerkinElmer 7CL Brightfield dataset. The ✓sign indicates that the loss weight of the appropriate head was positive, while the absence of ✓sign indicates that the loss weight of a head was set to 0 and the head was not learning anything during the training.

Segm.	Center	W&H	Angle	PW IoU	PW F1	OW IoU	OW F1
✓				0.569	0.723	0.239	0.298
✓	✓			0.599	0.748	0.271	0.337
✓	✓	✓		0.593	0.743	0.266	0.328
✓	✓	✓	✓	0.592	0.742	0.267	0.330

6 Discussion

We have explored different deep learning methods of performing object detection and semantic segmentation on digital microscopy images. We have also investigated ways to combine them to produce high-quality instance segmentation masks of microscopic objects like cell nuclei. Moreover, our goal was to investigate how exactly the segmentation task can benefit directly from object detection via multi-task learning. We have evaluated our approaches on four different datasets, which include fluorescent (PerkinElmer 7CL Fluorescent, Data Science Bowl2018), brightfield (PerkinElmer 7CL Brightfield), and histological (HuBMAP Kidney glomeruli) microscopy images.

Firstly, we have chosen an encoder for U-Net semantic segmentation architecture. We have demonstrated that the model can produce high-quality segmentation masks on fluorescent and histology images. However, the target objects (nuclei) on brightfield images are much harder to segment even manually by human experts, so the metric values on this dataset are predictably lower.

Secondly, we have trained two different object detection models. We have selected an anchor-based one-stage YOLOv5 as a first model for its high inference speed, together with a good detection performance. We have also chosen a one-stage CenterNet architecture as the most popular anchor-free architecture, eliminating many disadvantages of anchor-based models and capable of detecting even tiny and clumped objects.

We modified CenterNet by adding an object angle prediction head. The model was now able to produce a rotated bounding box around an object on an image, with the box width and height reflecting the shape of real objects irrespective of their orientation. Moreover, most of the target objects in the datasets we use (nuclei and glomeruli) are elongated, so rotated bounding boxes around them contain fewer background pixels and elements of neighboring objects. Compared to axis-aligned object detectors, our CenterNet with object angle head was able to produce more useful and interpretable predictions, even in nuclei clumps (Figure 28). Numeric results of rotated object detection evaluation (Table 3) indicate that the performance of the detector is very high on PerkinElmer 7CL Fluorescent (Figure 28a) and HuBMAP Kidney glomeruli (Figure 28d) datasets. Analogous to the segmentation task, lower performance for the PerkinElmer 7CL Brightfield dataset (Figure 28b) can be explained by the complexity of identifying nuclei width and height from a brightfield signal. Weaker performance on Data Science Bowl 2018 dataset (Figure 28c) can be attributed to the lower generalization of the model caused by an incredible diversity of this dataset and the presence of nuclei of all various sizes and types in it.

Thirdly, we have implemented a method of combining predictions of semantic segmentation (U-Net) and object detection (YOLO / CenterNet) models to produce instance segmentation masks. To do that, we extract the centers of predicted bounding boxes during the postprocessing stage and use them as markers for a watershed algorithm that separates the touching objects on a binary mask. We have shown that this method im-

proves object-wise segmentation metrics for fluorescent and brightfield datasets, meaning that the quality of distinguishing individual nuclei is increased.

However, using two separate models to perform instance segmentation has several disadvantages, including the doubled amount of computational resources required to train them and keep them in GPU memory when making an inference. Moreover, the measurements of inference time indicate that an inference of object detector adds an overhead time to the inference process. To eliminate this problem, we propose CenterUNet architecture (Figure 25), which can produce rotated object detection and semantic segmentation in one pass. We have noticed that U-Net (Figure 13) and CenterNet (Figure 22) have a very similar encoder-decoder architecture, so we modified the decoder of CenterNet to also produce full-size segmentation masks. In this way, detection and segmentation branches of CenterUNet share an encoder and a part of the decoder, and the whole network is trained end-to-end simultaneously for both tasks. Analogous to the architecture with two models, the outputs of segmentation and detection branches are combined to produce an instance segmentation using a watershed algorithm. We also evaluate the performance of this method and report an improvement in object-wise metrics compared to using segmentation only. Moreover, we observe that CenterUNet inference time is lower than the inference time of U-Net + YOLO or U-Net + CenterNet combinations.

Next, we have investigated how a presence of an object detection branch in CenterUNet (Figure 24) influences the performance of a segmentation branch. For each dataset, we have trained a CenterUNet with an object detection branch either being trained or not (using a zero loss weight). Evaluation has shown that the presence of a detection branch (without watershed postprocessing) has improved the instance segmentation on the PerkinElmer 7CL Brightfield dataset. On the other hand, the segmentation performance on the PerkinElmer 7CL Fluorescent dataset has not changed, indicating that the segmentation branch is capable of segmenting nuclei pixels correctly on its own. As for Data Science Bowl 2018 and HuBMAP Kidney glomeruli datasets, training an object detection branch has decreased segmentation performance. The degradation of segmenting glomeruli on histology images might have been caused by the fact that there are a lot of truncated glomeruli on sampled 256×256 px patches (Figure 28d). It means that glomeruli centers on ground-truth binary masks of these patches may not match real glomeruli centers, which may not even be present on a patch. This effect could have been mitigated by a more sophisticated patch sampling from the original high-resolution images (Figure 12a) to reduce the number of truncated glomeruli processed by the network.

Finally, we have investigated how different components of the CenterUNet detection branch influence the segmentation branch performance. We have trained several models with different object detection heads (center heatmap, width/height, rotation angle) turned on or off. The evaluation metrics on the PerkinElmer 7CL Brightfield dataset indicate that the most significant gain to segmentation performance comes from predicting a

center heatmap, while additionally regressing object width, height, and rotation angle slightly decreases the performance.

In general, object detection seems to impact semantic segmentation positively for some types of microscopy images when combined into one model like CenterUNet and trained using multi-task learning. However, it requires validation on each particular dataset and a meticulous choice of loss weights to ensure that detection does improve the segmentation. Nevertheless, when it comes to combining object detection and semantic segmentation using watershed postprocessing, the impact of this postprocessing is positive for all types of microscopy data, as separating the touching objects inevitably leads to better instance segmentation both visually and in terms of metrics.

6.1 Limitations

The results of our experiments have shown that instance segmentation performance on various medical imaging domains is very different. While the instance segmentation quality of fluorescent images is of high quality, the performance on brightfield images leaves much room for improvement. It means that the performance of our method should be evaluated individually for each new dataset it is applied to.

6.2 Future work

There exist several directions for potential improvement that can be implemented in the future to improve the instance segmentation quality of microscopy images. Firstly, U-Net architecture is not an only choice for a segmentation architecture anymore. There is a variety of other architectures that can be tested, namely Feature Pyramid Network [LDG+17], Pyramid Attention Network [LXAW18], DeepLabV3+ [CZP+18] and others.

Secondly, it is possible to train a neural network to predict an additional mask channel with object borders. This channel can be subtracted from the object mask in order to separate touching nuclei [CQY+17]. Adding this method to the proposed pipeline by combining object detection and semantic segmentation using watershed postprocessing can further improve the instance segmentation on datasets with high object density.

The proposed CenterUNet architecture produces rotated bounding boxes as part of the output. However, we use only the centers of these boxes during watershed postprocessing to separate touching objects. Further research can be directed to finding a way to incorporate the orientation of these objects into the object separation process.

7 Conclusion

Microscopy image analysis has dramatically benefited from the development of deep learning methods in computer vision. Biologists and pharmacists can now locate and segment microscopic objects on images for further analysis with high throughput and precision. We have shown that the instance segmentation quality of such objects can be improved by combining segmentation and object detection methods into one pipeline, which can help distinguish individual objects in microscopy images better. We have presented a novel neural network architecture that combines object detection and semantic segmentation capabilities in a single model and achieves good results on both tasks with low computational overhead. Moreover, we have demonstrated a positive impact of object detection on semantic segmentation directly when trained simultaneously inside one network using multi-task learning. We have also shown the advantages and the great potential of using rotated object detection methods in digital microscopy. We conclude that our model can be integrated into various cell segmentation, detection, and counting pipelines inside modern microscopy image analysis software.

8 Acknowledgements

This work was funded by PerkinElmer, Inc. and the Estonian Ministry of Foreign Affairs Development Cooperation and Humanitarian Aid funds. The computational resources were provided by the High Performance Computing Center of the University of Tartu. I want to thank Kaupo Palo for his continuous support from the industry side. Also, I am grateful to my supervisor Mikhail Papkov for his invaluable advice, insightful feedback and endless patience. Finally, I would like to acknowledge my research group colleagues - Leopold Parts, Dmytro Fishman, Sten-Oliver Salumaa, Mohammed Ali for their support.

References

[AKWJ19] Mahmoud Abdolhoseini, Murielle G. Kluge, Frederick R. Walker, and Sarah J. Johnson. Segmentation of heavily clustered nuclei from histopathological images. *Scientific Reports*, 9(1):4551, Mar 2019.

[Bie20] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[BKK+19] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N. Straehle, Bernhard X. Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, Kemal Eren, Jaime I. Cervantes, Buote Xu, Fynn Beuttenmueller, Adrian Wolny, Chong Zhang, Ullrich Koethe, Fred A. Hamprecht, and Anna Kreshuk. ilastik: interactive machine learning for (bio)image analysis. *Nature Methods*, September 2019.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[CGK+19] Juan C. Caicedo, Allen Goodman, Kyle W. Karhohs, Beth A. Cimini, Jeanelle Ackerman, Marzieh Haghighi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, Mohammad Rohban, Shantanu Singh, and Anne E. Carpenter. Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nature Methods*, 16(12):1247–1253, Dec 2019.

[CMLBSG19] Luis Camuñas-Mesa, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic spiking neural networks and their memristor-cmos hardware implementations. *Materials*, 12:2745, 08 2019.

[CQY+17] Hao Chen, Xiaojuan Qi, Lequan Yu, Qi Dou, Jing Qin, and Pheng-Ann Heng. Dcan: Deep contour-aware networks for object instance segmentation from histology images. *Medical Image Analysis*, 36:135–146, 2017.

[CRG+19] Juan Caicedo, Jonathan Roth, Allen Goodman, Tim Becker, Kyle Karhohs, Matthieu Broisin, Csaba Molnar, Claire McQuin, Shantanu Singh, Fabian Theis, and Anne Carpenter. Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. *Cytometry Part A*, 95, 07 2019.

[CZP+18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[DLL+20] Zhiwei Dong, Guoxuan Li, Yue Liao, Fei Wang, Pengju Ren, and Chen Qian. Centripetalnet: Pursuing high-quality keypoint pairs for object detection, 2020.

[DQX+17] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks, 2017.

[DVC+16] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In Gustavo Carneiro, Diana Mateus, Loïc Peter, Andrew Bradley, João Manuel R. S. Tavares, Vasileios Belagiannis, João Paulo Papa, Jacinto C. Nascimento, Marco Loog, Zhi Lu, Jaime S. Cardoso, and Julien Cornebise, editors, *Deep Learning and Data Labeling for Medical Applications*, pages 179–187, Cham, 2016. Springer International Publishing.

[FC20] Martin Fränzl and Frank Cichos. Convolutional neural networks for real-time localization and classification in feedback digital microscopy, 2020.

[FSB19] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C. Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free, 2019.

[FSM+19] Dmytro Fishman, Sten-Oliver Salumaa, Daniel Majoral, Samantha Peel, Jan Wildenhain, Alexander Schreiner, Kaupo Palo, and Leopold Parts. Segmenting nuclei in brightfield images with neural networks. *bioRxiv*, 2019.

[G+07] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[Gir15] Ross Girshick. Fast R-CNN, 2015.

[HGDG18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN, 2018.

[HHW+18] Henning Höfener, André Homeyer, Nick Weiss, Jesper Molin, Claes Lundström, and Horst Hahn. Deep learning nuclei detection: A simple approach can deliver state-of-the-art results. *Computerized Medical Imaging and Graphics*, 70:43–52, 12 2018.

[HSA+19] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Jor18] Jeremy Jordan. Evaluating image segmentation models. `https://www.jeremyjordan.me/evaluating-image-segmentation-models/`, 2018. Accessed: 2021–05-06.

[JSB+20] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, Lorenzomammana, Tkianai, YxNONG, AlexWang1900, Laurentiu Diaconu, , Marc, Wanghaoyang0106, Ml5ah, , Doug, Hatovix, Jake Poznanski, Lijun Yu, Changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, Hopesala, Pritul Dave, and Yzchen. ultralytics/yolov5: v3.0, 2020.

[JYG+20] Glenn Jocher, Yonghye Kwon, Guigarfr, Josh Veitch-Michaelis, Perry0418, Ttayu, , Marc, Gabriel Bianconi, Fatih Baltacı, Daniel Suess, Idow09, WannaSeaU, Wang Xinyu, Timothy M. Shead, Thomas Havlik, Piotr Skalski, NirZarrabi, LukeAI, LinCoce, Jeremy Hu, IlyaOvodov, GoogleWiki, Francisco Reveriano, , Falak, and Dustin Kendall. ultralytics/yolov3: 43.1map@0.5:0.95 on coco2014, 2020.

[JZL+19] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[KŻS+20] Marek Kowal, Michał Żejmo, Marcin Skobel, Józef Korbicz, and Roman Monczak. Cell nuclei segmentation in cytological images using convolutional neural network and seeded watershed algorithm. *Journal of Digital Imaging*, 33(1):231–242, Feb 2020.

[LBBH98] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[LD19] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints, 2019.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[LGG+18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[LJY17] Fei-Fei Li, Justin Johnson, and Serena Yeung. Lecture 11: Detection and segmentation. `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf`, 2017. Accessed: 2021–04–01.

[LMB+15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[LOW+19] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey, 2019.

[LPL17] Lei Liu, Zongxu Pan, and Bin Lei. Learning a rotation invariant detector with rotatable bounding box, 2017.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[LXAW18] Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. Pyramid attention network for semantic segmentation, 2018.

[Mas15] Andrew J. Massey. Multiparametric cell cycle analysis using the operetta high-content imager and harmony software with PhenoLOGIC. *PLOS ONE*, 10(7):e0134306, July 2015.

[MBK+19] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. *Nature Methods*, 16(12):1233–1246, Dec 2019.

[MBP+20] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

[MC19] Wojciech Mormul and Paweł Chmielak. Satellite images semantic segmentation with deep learning. `https://deepsense.ai/satellite-images-semantic-segmentation-with-deep-learning/`, 2019. Accessed: 2021–04-18.

[MGC+18] Claire McQuin, Allen Goodman, Vasiliy Chernyshev, Lee Kamentsky, Beth A. Cimini, Kyle W. Karhohs, Minh Doan, Liya Ding, Susanne M. Rafelski, Derek Thirstrup, Winfried Wiegraebe, Shantanu Singh, Tim Becker, Juan C. Caicedo, and Anne E. Carpenter. CellProfiler 3.0: Next-generation image processing for biology. *PLOS Biology*, 16(7):e2005970, July 2018.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[NBG19] Vanessa Ndonhong, Anqi Bao, and Olivier Germain. Wellbore schematics to structured data using artificial intelligence tools. 04 2019.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PMC11] Mcj Maurice Peemen, B. Mesman, and H. Corporaal. Speed sign detection and recognition by convolutional neural networks. 2011.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[RHGS16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks, 2016.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[RM01] Jos BTM Roerdink and Arnold Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, 41:187–228, 2001.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[RZKB19] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging, 2019.

[SG13] Christoph Sommer and Daniel W. Gerlich. Machine learning in cell biology – teaching computers to recognize phenotypes. *Journal of Cell Science*, 126(24):5529–5539, 2013.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[SHZ+19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[Sin20] Aishwarya Singh. Selecting the right bounding box using non-max suppression (with implementation). `https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-impleme` 2020. Accessed: 2021–04–22.

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[SLV+17] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Lecture Notes in Computer Science*, page 240–248, 2017.

[SPB+18] Kevin Smith, Filippo Piccinini, Tamas Balassa, Krisztian Koos, Tivadar Danka, Hossein Azizpour, and Peter Horvath. Phenotypic image analysis software tools for exploring and understanding big image data from cell-based assays. *Cell Systems*, 6(6):636 – 653, 2018.

[Tei19] Petteri Teikari. Multiphoton vasculature segmentation 5: U-net. `https://www.slideshare.net/PetteriTeikariPhD/multiphoton-vasculature-segmentation-5-unet`, 2019. Accessed: 2021–04–15.

[TL20] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[TZD+17] Tianyu Tang, Shilin Zhou, Zhipeng Deng, Lin Lei, and Huanxin Zou. Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks. *Remote Sensing*, 9:1170, 11 2017.

[Val20] Paolo Valdez. Apple defect detection using deep learning based object detection for better post harvest handling, 2020.

[VBJ+19] Tomas Vicar, Jan Balvan, Josef Jaros, Florian Jug, Radim Kolar, Michal Masarik, and Jaromir Gumulec. Cell segmentation methods for label-free contrast microscopy: review and comprehensive comparison. *BMC bioinformatics*, 20(1):360–360, Jun 2019. 31253078[pmid].

[vdWSN+14] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[Wan20] Feng Wang. Centernet-better. `https://github.com/FateScript/CenterNet-better`, 2020.

[WSS18] Wenguan Wang, J. Shen, and L. Shao. Video salient object detection via fully convolutional networks. *IEEE Transactions on Image Processing*, 27:38–49, 2018.

[XNZ18] Weidi Xie, J. Alison Noble, and Andrew Zisserman. Microscopy cell counting and detection with fully convolutional regression networks.

Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 6(3):283–292, 2018.

[XXS+18] Fuyong Xing, Yuanpu Xie, Hai Su, Fujun Liu, and Lin Yang. Deep learning in microscopy image analysis: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4550–4568, 2018.

[Yak20] Pavel Yakubovskiy. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2020.

[YFG20] He Yang, Beibei Fan, and Lingling Guo. Anchor-free object detection with mask attention. *EURASIP Journal on Image and Video Processing*, 2020(1), July 2020.

[YGF+20] Linfeng Yang, Rajarshi P. Ghosh, J. Matthew Franklin, Simon Chen, Chenyu You, Raja R. Narayan, Marc L. Melcher, and Jan T. Liphardt. NuSeT: A deep learning tool for reliably separating and analyzing crowded cells. *PLOS Computational Biology*, 16(9):e1008193, September 2020.

[YYFH20] Xue Yang, Junchi Yan, Ziming Feng, and Tao He. R3det: Refined single-stage detector with feature refinement for rotating object, 2020.

[YYY+19] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Sun Xian, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects, 2019.

[ZRC19] Tongxue Zhou, Su Ruan, and Stéphane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3-4:100004, 2019.

[ZWK19] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points, 2019.

Appendix

I. Supplementary figures

(a) Zoomed sample from PerkinElmer 7CL Brightfield dataset

(b) Zoomed sample from PerkinElmer 7CL Brightfield dataset

(c) Sample from Data Science Bowl 2018 dataset

Figure 33. Examples of bad instance segmentation performance of CenterUNet. Raw image (left), ground-truth segmentation (center), prediction (right).

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Dmytro Urukov**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Improving Microscopy Image Segmentation with Object Detection,
   (title of thesis)

   supervised by Mikhail Papkov.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Dmytro Urukov
14/05/2021