DISSERTATION

submitted

to the

Combined Faculty for the Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Thomas Sebastian Wollmann, M.Sc.

Born in: Solingen, Germany

Oral examination: .....

Deep Learning for Detection and Segmentation in High-Content Microscopy Images

Advisor: PD Dr. Karl Rohr

Abstract

High-content microscopy led to many advances in biology and medicine. This fast emerging technology is transforming cell biology into a big data driven science. Computer vision methods are used to automate the analysis of microscopy image data. In recent years, deep learning became popular and had major success in computer vision. Most of the available methods are developed to process natural images. Compared to natural images, microscopy images pose domain specific challenges such as small training datasets, clustered objects, and class imbalance.

In this thesis, new deep learning methods for object detection and cell segmentation in microscopy images are introduced. For particle detection in fluorescence microscopy images, a deep learning method based on a domain-adapted Deconvolution Network is presented. In addition, a method for mitotic cell detection in heterogeneous histopathology images is proposed, which combines a deep residual network with Hough voting. The method is used for grading of whole-slide histology images of breast carcinoma. Moreover, a method for both particle detection and cell detection based on object centroids is introduced, which is trainable end-to-end. It comprises a novel Centroid Proposal Network, a layer for ensembling detection hypotheses over image scales and anchors, an anchor regularization scheme which favours prior anchors over regressed locations, and an improved algorithm for Non-Maximum Suppression. Furthermore, a novel loss function based on Normalized Mutual Information is proposed which can cope with strong class imbalance and is derived within a Bayesian framework.

For cell segmentation, a deep neural network with increased receptive field to capture rich semantic information is introduced. Moreover, a deep neural network which combines both paradigms of multi-scale feature aggregation of Convolutional Neural Networks and iterative refinement of Recurrent Neural Networks is proposed. To increase the robustness of the training and improve segmentation, a novel focal loss function is presented.

In addition, a framework for black-box hyperparameter optimization for biomedical image analysis pipelines is proposed. The framework has a modular architecture that separates hyperparameter sampling and hyperparameter optimization. A visualization of the loss function based on infimum projections is suggested to obtain further insights into the optimization problem. Also, a transfer learning approach is presented, which uses only one color channel for pre-training and performs fine-tuning on more color channels. Furthermore, an approach for unsupervised domain adaptation for histopathological slides is presented.

Finally, Galaxy Image Analysis is presented, a platform for web-based microscopy image analysis. Galaxy Image Analysis workflows for cell segmentation in cell cultures, particle detection in mice brain tissue, and MALDI/H&E image registration have been developed.

The proposed methods were applied to challenging synthetic as well as real

microscopy image data from various microscopy modalities. It turned out that the proposed methods yield state-of-the-art or improved results. The methods were benchmarked in international image analysis challenges and used in various cooperation projects with biomedical researchers.

Zusammenfassung

High-Content Mikroskopie führte zu vielen Fortschritten in der Biologie und Medizin. Diese Technologie hat die Zellbiologie in eine durch große Daten getriebene Wissenschaft transformiert. Computergestützte Bildanalyse wird genutzt, um mikroskopische Bilddaten automatisiert auszuwerten. In den letzten Jahren ist Deep Learning durch die Erfolge in der computergestützten Bildanalyse populär geworden. Die meisten eingesetzten Methoden wurden für die Anwendung an Bildern von natürlichen Szenen entwickelt. Im Vergleich dazu besitzen mikroskopische Bilddaten domänenspezifische Herausforderungen wie wenig Trainingsdaten, hohe Objektdichte und Klassenungleichgewicht.

In dieser Arbeit werden neue Deep Learning Methoden für Objekterkennung und Zellsegmentierung in Mikroskopiebildern vorgestellt. Es wurde eine Methode für Partikeldetektion in Fluoreszenzmikroskopiebildern auf Basis eines für diese Anwendung optimierten Deconvolution Network entwickelt. Weiterhin wurde eine Methode für die Detektion von mitotischen Zellen in heterogenen histopathologischen Bildern entwickelt, welche ein Deep Residual Network mit Hough Voting kombiniert. Die Methode wird für das Grading von Whole-Slide Histologiebildern genutzt. Darüber hinaus wurde eine Methode für sowohl Partikeldetektion als auch Zelldetektion basierend auf Objektzentroiden entwickelt, welche end-to-end trainiert werden kann. Die Methode umfasst ein Centroid Proposal Network, ein Layer für die Aggregation von Detektionshypothesen über alle Bildskalen und Anker, sowie eine Methode zur Regularisierung, die a-priori Anker gegenüber vorhergesagten Verschiebungen bevorzugt, und einen verbesserten Algorithmus für Non-Maximum Suppression. Eine neue Loss-Funktion basierend auf normalisierter Mutual Information wird vorgestellt, die mit starkem Klassenungleichgewicht umgehen kann.

Für die Zellsegmentierung wird ein Neuronales Netz mit vergrößertem rezeptiven Feld vorgestellt, um mehr semantische Informationen zu modellieren. Darüber hinaus wird ein Neuronales Netz vorgeschlagen, dass die Paradigmen von Multi-Skalen-Feature-Extraktion von Convolutional Neural Networks und iteratives Verfeinern mittels Recurrent Neural Networks verbindet. Für ein robusteres Training und eine verbessererte Segmentierung wurde eine Focal Loss basierte Loss-Funktion entwickelt.

Weiterhin wird ein Framework für Black-Box Hyperparameteroptimierung für biomedizinische Bildverarbeitungspipelines vorgestellt. Dieses Framework nutzt eine modulare Architektur, die Hyperparameterabtastung und Hyperparameteroptimierung trennt. Eine Visualisierung der Loss-Funktion, basierend auf Infimum Projektionen für die Analyse des Optimierungsprozesses, wird vorgeschlagen. Darüber hinaus wird eine Transfer Learning Technik vorgestellt, die Netzwerke, die mit einem Eingangskanal trainiert wurden, für Daten mit mehreren Eingangskanälen nutzbar macht. Zusätzlich wurde eine Methode für Unsupervised Domain Adaptation in histopathologischen Schnitten entwickelt.

Weiterhin wird Galaxy Image Analysis präsentiert, eine Platform für die web-

basierte Analyse von mikroskopischen Bildern. Galaxy Image Analysis Workflows für Zellsegmentierung in Zellkulturen, Partikeldetektion in Hirnschnitten von Mäusen, und MALDI/H&E Registrierung werden vorgestellt.

Die vorgestellten Methoden wurden für synthetische und reale Mikroskopiedaten mehrerer Modalitäten angewandt und erreichten Stand der Kunst oder bessere Performanz. Die Methoden wurden in internationalen Wettbewerben evaluiert und mit Kooperationspartnern in biomedizinischen Forschungsprojekten genutzt.

Acknowledgements

Undertaking my doctoral studies in computer science at the faculty for mathematics and computer science at Heidelberg University would not have been possible without the support and guidance that I received from many people.

First of all, I would like express my great thanks to PD Dr. Karl Rohr for supervising my work, but especially for his patience and always providing me with freedom to work on the topics I am interested in.

I want to express my gratitude to all the people with whom I had very productive collaborations. A special thanks goes to Dr. Björn Grüning (Backofen lab, University of Freiburg) for integrating me into the lovely Galaxy and Bioconda communities and always helping me out at the craziest times of the day. In addition, I would also like to thank my biomedical cooperation partners Charlotte Bold (Müller lab, Heidelberg University), Dr. Delia Braun (Rippe lab, DKFZ Heidelberg), PhD Karina Durso-Cain (Uprichard lab, Loyola University Chicago), Dr. Melanie Föll (Schilling lab, University of Freiburg), Dr. Manuel Gunkel (Erfle lab, Heidelberg University), Dr. Peter Kumberger (Graw lab, Heidelberg University), Dr. Alexandra Poos (König lab, University Hospital Jena), and all the others for letting me be part of their interesting research projects.

A big thanks goes to my colleague Christian Ritter for fruitful discussions, research collaboration and being a great companion during my doctoral studies. Moreover, I would like to thank the entire Biomedical Computer Vision Group for many discussions. Especially, I want to thank our secretaries Manuela Schäfer and Sabrina Wetzel for always keeping everything running smoothly and to take the university bureaucracy load off us. I want to thank my supervised interns, master students, and student assistants Danny Baltissen, Patrick Bernhard, Jan-Niklas Dohrke, Sophia Eijkman, Simone Gierlich, Julia Ivanova, Roman Spilger, Hao Tian, Niklas Vockert, and Kevin Walz for labeling and supporting me carrying out the experiments.

I am very grateful that Cornelia Wollmann, Dr. Gerhard Wollmann, Sarah Linnemeier, and Sanja Miskic found time for proof reading my thesis. My final thanks are reserved for my parents Cornelia and Gerhard and my girlfriend Sarah. I cannot thank my family enough for the continuous support they have given me throughout my life and carrying me through good as well as bad times.

I would also like to acknowledge the BMBF-funded Heidelberg Center for Human Bioinformatics (HD-HuB) within the German Network for Bioinformatics Infrastructure (de.NBI) #031A537C for financial support of my work.

Contents

List of Figures

List of Tables

Nomenclature

Mathematical symbols frequently used in this thesis.

$\odot$	Hadamard product
$*$	Convolution operator
$\mathbf{vec}(\mathbf{A})$	Vectorization of a matrix $\mathbf{A}$
$\mathcal{L}$	Loss function
$\mathbf{x}$	Input tensor of layer
$\mathbf{y}$	Output tensor of layer
$\mathbf{X}$	Prediction vector
$\mathbf{Y}$	Ground truth vector
$\mathbf{W}$	Weight tensor of layer
t	Iteration/time
i	Layer/Neuron index
b	Batch size
$w \times h$	Feature map width and height
p	Number of input feature maps
q	Number of output feature maps
k	Window size of convolution kernel

Publications

Major parts of this thesis have been published in peer-reviewed journals and peer-reviewed conference proceedings.

Peer-Reviewed Journal Articles

T. Wollmann and K. Rohr, "Deep Consensus Network: Aggregating predictions to improve object detection in microscopy images," *under review*, 2020

T. Wollmann, M. Gunkel, I. Chung, H. Erfle, and K. Rohr, "GRUU-Net: Integrated convolutional and gated recurrent neural network for cell segmentation," *Med. Image Anal.*, vol. 56, pp. 68–79, 2019

C. Ritter, **T. Wollmann**, P. Bernhard, M. Gunkel, D. M. Braun, J.-Y. Lee, J. Meiners, R. Simon, G. Sauter, H. Erfle, K. Rippe, R. Bartenschlager, and K. Rohr, "Hyperparameter optimization for image analysis: Application to prostate tissue images and live cell data of virus-infected cells," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 11, pp. 1847–1857, 2019

M. Veta, Y. J. Heng, N. Stathonikos, B. E. Bejnordi, F. Beca, **T. Wollmann**, K. Rohr, M. A. Shah, D. Wang, M. Rousson, *et al.*, "Predicting breast tumor proliferation from whole-slide images: The TUPAC16 challenge," *Med. Image Anal.*, vol. 54, pp. 111–121, 2019

M. C. Föll, L. Moritz, **T. Wollmann**, M. N. Stillger, N. Vockert, M. Werner, P. Bronsert, K. Rohr, B. A. Grüning, and O. Schilling, "Accessible and reproducible mass spectrometry imaging data analysis in Galaxy," *GigaScience*, vol. 8, no. 12, p. giz143, 2019

B. Grüning, R. Dale, A. Sjödin, B. A. Chapman, J. Rowe, C. H. Tomkins-Tinch, R. Valieris, A. Caprez, B. Batut, M. Haudgaard, T. Cokelaer, K. A. Beauchamp, B. S. Pedersen, Y. Hoogstrate, D. Ryan, A. Bretaudeau, G. L. Corguillé, C. Brueffer, D. Yusuf, S. Luna-Valero, R. Kirchner, K. Brinda, M. Raden, **T. Wollmann**, J. Köster, *et al.*, "Bioconda: A sustainable and comprehensive software distribution for the life sciences," *Nat. Methods*, vol. 15, pp. 475—-476, 2018

T. Wollmann, H. Erfle, R. Eils, K. Rohr, and M. Gunkel, "Workflows for microscopy image analysis and cellular phenotyping," *J. Biotechnol*, vol. 261, pp. 70–75, 2017

Peer-Reviewed Conference Proceedings

C. Ritter, **T. Wollmann**, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep learning particle detection for probabilistic tracking in fluorescence microscopy," in *Proc. ISBI*, IEEE, 2020

T. Wollmann, C. Ritter, J.-N. Dohrke, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "DetNet: Deep neural network for particle detection in microscopy images," in *Proc. ISBI*, pp. 517–520, IEEE, 2019

T. Wollmann, P. Bernhard, M. Gunkel, D. M. Braun, J. Meiners, R. Simon, G. Sauter, H. Erfle, K. Rippe, and K. Rohr, "Black-box hyperparameter optimization for nuclei segmentation in prostate tissue images," in *Proc. BVM*, pp. 345–350, Springer, 2019

T. Wollmann, C. S. Eijkman, and K. Rohr, "Adversarial domain adaptation to improve automatic breast cancer grading in lymph nodes," in *Proc. ISBI*, pp. 582–585, IEEE, 2018

T. Wollmann, J. Ivanova, and K. Rohr, "Multi-channel deep transfer learning for nuclei segmentation in glioblastoma cell tissue images," in *Proc. BVM*, pp. 316–321, Springer, 2018

R. Spilger, **T. Wollmann**, Y. Qiang, A. Imle, J.-Y. Lee, B. Müller, O. T. Fackler, R. Bartenschlager, and K. Rohr, "Deep Particle Tracker: Automatic tracking of particles in fluorescence microscopy images Using deep learning," in *Proc. MICCAI Workshop DLMIA*, pp. 128–136, Springer, 2018

D. Baltissen, **T. Wollmann**, M. Gunkel, I. Chung, H. Erfle, K. Rippe, and K. Rohr, "Comparison of segmentation methods for tissue microscopy images of glioblastoma cells," in *Proc. ISBI*, pp. 396–399, IEEE, 2018

T. Wollmann and K. Rohr, "Deep residual Hough voting for cell detection in histopathology images," in *Proc. ISBI*, pp. 341–344, IEEE, 2017

T. Wollmann and K. Rohr, "Automatic grading of breast cancer whole-slide histopathology images," in *Proc. BVM*, pp. 249–253, Springer, 2017

1 Introduction

1.1 Motivation

In biomedical research, information about physiological processes is often required to verify research hypotheses [18]. Microscopy imaging is one of the most important techniques to extract such information [19]. Since manual analysis is generally too slow, labour-intensive, and prone to errors, automatic analysis is required to process the constantly increasing amount of microscopy image data.

The complexity of acquired microscopy images poses many challenges for image analysis algorithms. In recent years, deep learning improved the state-of-the-art in many computer vision tasks [20, 21, 22]. Especially, advances in deep learning methods for object detection [23, 24, 25], semantic segmentation [26, 27, 28], and classification of images [1, 29, 30] lead to improved results. Object detection and segmentation are frequent tasks to analyse high-content microscopy images, and deep learning has been used for such kind of images (e.g., [31, 32]). However, most of the existing deep learning methods have been developed for images of natural scenes. Biomedical images and particularly microscopy images raise additional domain-specific challenges compared to images of natural scenes (e.g., small objects, low SNR). The images vary significantly due to the experimental setup, imaging workflows, and imaging modalities. In addition, large annotated training datasets like COCO [33] or ImageNet [34] for natural images are not available for biomedical data. Biomedical image datasets often suffer from low annotation standardization and significant label noise. Moreover, usage of cutting-edge computer vision methods and especially methods based on deep learning is currently quite complex. Utilization of high-performance computing (HPC) and cloud compute infrastructure is too cumbersome for most biomedical researchers.

This thesis addresses different challenges that are important for successfully using deep learning in high-content microscopy image analysis. The main topics of this thesis are outlined in Figure 1.1. The different chapters cope with *challenges posed by microscopy datasets*, introduce novel *deep learning methods*, and consider the *deployment* of image analysis workflows. Novel *deep learning methods* are proposed for main tasks of microscopy image analysis, namely *detection* and *segmentation*. In addition, automatic *optimization of the hyperparameters* of software pipelines, *transfer learning* for reusing trained networks, and *data augmentation* to cope with the lack of training data are investigated to address *dataset-specific challenges*. Furthermore, a *concept for web-based image analysis* and a system for *deployment of*

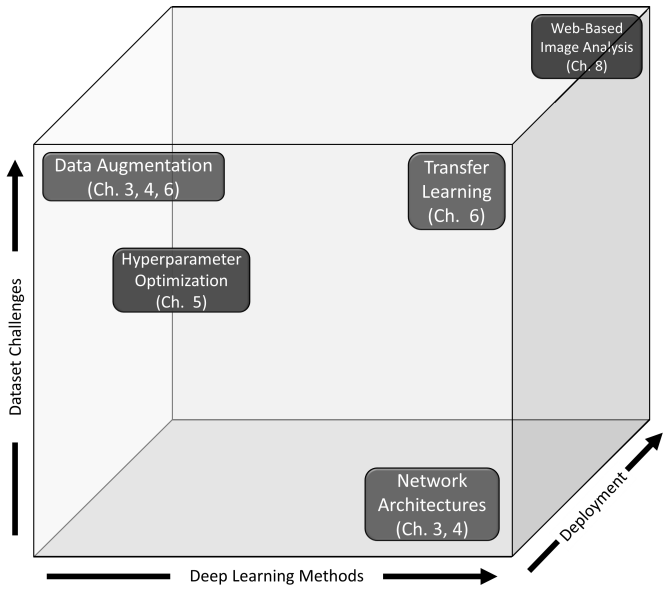


Figure 1.1: Main topics of this thesis.

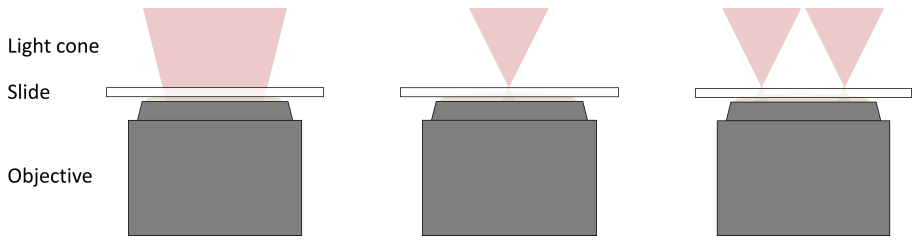
image analysis software in a research environment are presented.

1.1.1 Biomedical Microscopy Imaging

In medicine and biology, investigated structures like tissue microstructures, cells, viruses, or bacteria are too small to be seen with the naked eye. Microscopy techniques can provide magnified visual or photographic images of these structures [19]. Microscopes leverage light (optical microscope), electrons (electron microscope), or a scanning probe (scanning probe microscope) for capturing structures. This thesis focuses on optical microscopy modalities and considers an application of mass spectrometry imaging combined with optical microscopy. In the following, an overview of these modalities is given.

An optical microscope typically consists of an object slide containing the structures of interest, a system of lenses for magnification and filtering, which creates the image in the intermediate plane and is observable in the eyepiece or digitalized using an image sensor, and, depending on the technique, an illumination module [19]. An object is in focus when the light rays originating from the object specimen converge in the eyepiece or image sensor. In this thesis, datasets from translumination-based (bright-field, phase-contrast, differential interference contrast) and fluorescence-based (widefield, confocal, spinning disk) microscopes are analysed. *Translumination microscopy* can image tissue by transmitting light through the object slide [19]. In *bright-field microscopy*, the object slide is illuminated with white light and the absorption of the light creates contrast in the resulting image [19]. Stainings can be used to increase light absorption of certain structures, which accentuate them in the image. Unstained tissue is hardly visible in bright-field microscopes as it is mostly translucent. *Phase-contrast microscopy* can image translucent objects using phase shifts in light passing

through the object of interest. A phase-shift ring is used to shift the phase of the light by 90° or -90° that passed the slide. After filtering, the background light and the light scattered by the object overlay and the resulting constructive interference creates visual contrast. *Differential interference contrast (DIC) microscopy* is an alternative method to image translucent objects by interferometry. The light is first polarized and then separated into two rays, which are focused on the sample. After passing the sample, the rays are overlaid using a prism and contrast is created by constructive interference. *Fluorescence microscopy* can image fluorophore stained tissue which emits light when being illuminated [19]. The excitation spectrum is the required light which has to be emitted by the fluorescent light source to stimulate the fluorophore so that it emits light in its characteristic emission spectrum. There exist several fluorescence microscopy techniques like widefield, laser scanning, and spinning disk microscopy. The differences of the pattern of illumination is illustrated in Figure 1.2. In widefield microscopy, the whole slide is illuminated which then excites fluorophores [19]. The main disadvantage of widefield microscopy is that light emitted from the specimen out-of-focus interferes with the light emitted within focus, which reduces the maximum resolution in addition to the thickness of the specimen. Confocal microscopy uses the pinhole principle to only detect light from the image plane in focus. *Confocal laser scanning microscopy (CLSM)* uses a laser for illumination, which scans the slide in a raster pattern and uses a photomultiplier tube to detect the signal for each spot. Compared to widefield microscopy, single coordinates in 3D can be imaged. CLSM is comparably slow, since each coordinate has to be imaged sequentially. In *spinning disk confocal laser microscopy (SDCLM)*, multiple coordinates are illuminated simultaneously by leveraging multiplexing. Multiple pinholes are arranged on a mechanically spinning Nipkow disk in a specific pattern. A dichroic mirror is used to separate scattered/reflected light and laser light from the optics. Depending on the design, a second or the same Nipkow disk is used as light shade for each corresponding pinhole of the first Nipkow disk transit. In SDCLM, camera detectors are used instead of a photomultiplier tube. They have the advantage of a higher quantum efficiency. Therefore, images with a higher signal to noise ratio can be obtained compared to CLSM. Multiple operating modes can be combined in a single device like the microscope shown in Figure 1.3. A drawback of fluorescence microscopy is that fluorescent stains are phototoxic, invasive, and bleach



(a) Widefield (b) Laser scanning (c) Spinning disk

Figure 1.2: Illumination patterns in fluorescence microscopy

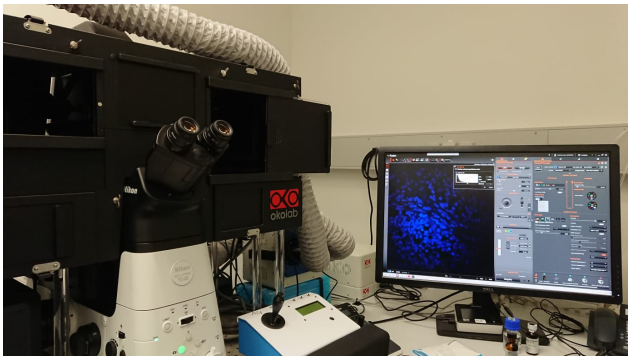


Figure 1.3: Optical table with Nikon Eclipse Ti2 microscope, which supports phase-contrast, DIC, CLSM, and SDCLM operating modes.

when being illuminated, which makes them more challenging for live cell imaging than other techniques [19].

Many microscopy techniques benefit from or require stained specimen to enhance contrast of structures of interest. Essential stains and dyes in biology and medicine are hematoxylin and eosin stain (H&E) and 4',6-diamidino-2-phenylindole (DAPI) [19]. *H&E staining* is usually used for bright-field microscopy in histology [19]. Hematoxylin binds to basophilic substances and appears as dark blue or violet in the image. Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are negatively charged and therefore acidic, which makes them basophilic. The chromosomes consisting of DNA are usually located in the cell nucleus, and RNA is highly concentrated in the ribosomes of the rough endoplasmic reticulum (Figure 1.4). The different cell states like cell division (mitotic phase), programmed cell death (apoptosis), or premature cell death (necrosis) manifest in different chromosome appearance. As opposed to hematoxylin, eosin binds to acidophilic substances and is seen in red or pink in the image. Amino acids and proteins that are amino acid complexes are basic, as the molecules are positively charged due to their arginine and lysine residues. Amino acids and proteins are highly concentrated in structures

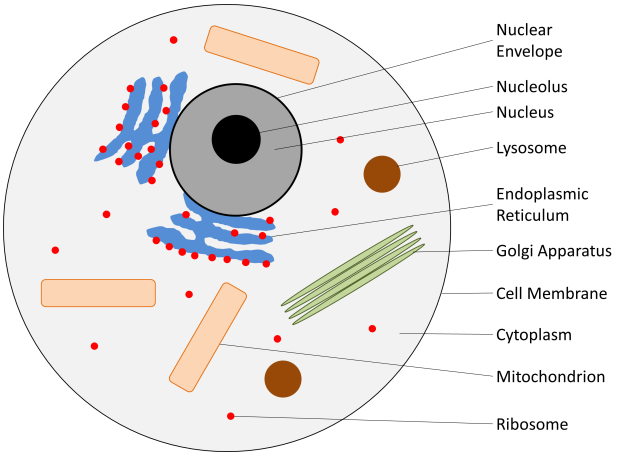


Figure 1.4: Eukaryotic cell in a simplified cutaway drawing

like the cytoplasm and cell organelles (e.g., mitochondria, erythrocytes, collagen, extracellular fibers). The Golgi apparatus, myelin, or adipocytes are hydrophobic structures and remain clear when using H&E staining as the stain is water-based. Immunostainings are used to detect specific proteins by exploiting the antibody mechanism to target the proteins of interest. *DAPI* is an essential stain in fluorescence microscopy [19]. It mainly binds to the regions rich in adenine-thymine in the DNA, dyeing the nucleus. DAPI also binds to RNA with a different emission wavelength. A main advantage of DAPI is that it can be combined with other popular dyes like GFP or CY3. However, cross-talk and bleed-through can occur when multiple stains are used. The cross-talk effect describes that dyes with overlapping excitation spectrum are illuminated at the same time. Bleed-through describes the effect that the emission spectra of two stains have an overlap so that the filters and detectors are not able to separate the signals. More specialized fluorescence stainings (e.g., GFP, CY3, FAM, Alexa Fluor) can be modified by using techniques like fluorescence in situ hybridization (FISH) to bind to specific structures (e.g., centromeres, telomeres, target genes).

A stain-free, but destructive microscopy method is spatially-resolved mass spectrometry [35]. *Matrix-assisted laser desorption/ionization (MALDI)* is an ionization technique where a matrix material is added to the sample (Figure 1.5b) [36]. A laser is used to ionize the sample and to excavate macro molecules. These ionized molecules are usually proteins that can be detected using by their time of flight (ToF) in a mass spectrometer (Figure 1.5a). When applying the laser grid-wise on a slide, spatially-resolved mass spectra can be obtained. MALDI-ToF can be applied to an already stained sample and can therefore be combined with optical microscopy [36].

The modalities, stainings, and dyes considered in this thesis are summarized in Table 1.1.

(a) Matrix sprayer (b) MALDI-ToF mass spectrometry

Figure 1.5: MALDI-ToF mass spectrometry hardware

Table 1.1: Overview of microscopy techniques considered in this thesis

Microscopy Principle	Contrast Technique	Modality	Staining & Dye
Optical	Translumination	Bright-field	H&E, Immuno
		Phase-contrast	-
		DIC	
	Fluorescence	Widefield	DAPI, GFP, CY3, FAM, Alexa Fluor
		CLSM	
		SDCLM	
Mass spectrometry	Desorption/Ionization	MALDI	-

1.1.2 Biomedical Microscopy Image Analysis

In a biological or medical research, where complex image data like high-content or high-throughput microscopy images are acquired using the imaging techniques described above, manual analysis is often not feasible. Automated image analysis can help coping with the data. A meta image analysis workflow of images in biology and medicine is illustrated in Figure 1.6 and consists of three main steps.

First, the images are *pre-processed* to improve the image quality and enhance meaningful content. Methods for pre-processing images are mostly based on filters (e.g., median rank filter, Gabor filter, histogram equalization). These filters can be applied within a sliding window or the whole image.

Second, specific features of interest are extracted using *image analysis* methods like object detection, segmentation, image classification, image registration, and object

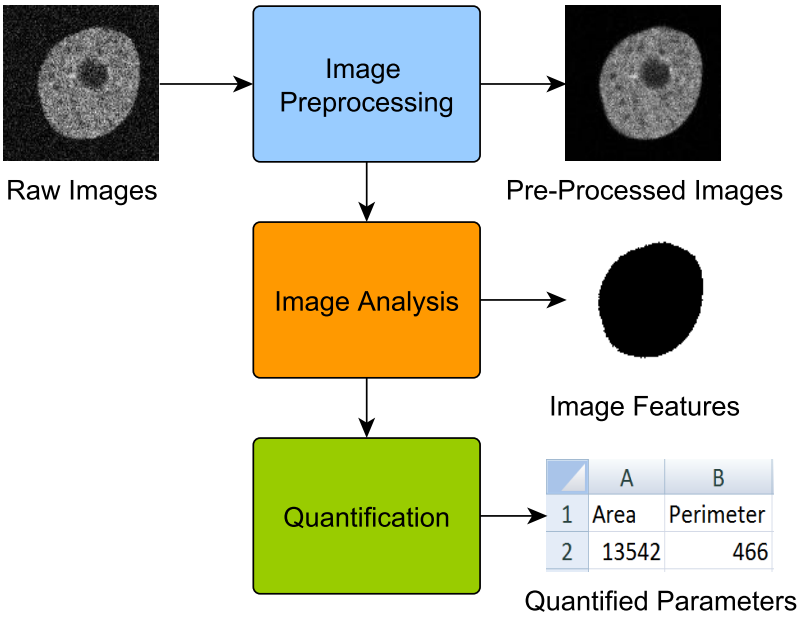


Figure 1.6: Meta image analysis workflow with an example of cellular phenotyping

tracking. Object detection can be performed by using local feature descriptors (e.g., SIFT, ORB) [37], which can be combined with a classifier (e.g., Logistic regression, Random Forrest, Support Vector Machine) [38]. Segmentation methods can be categorized by their definition of image segments. This includes methods without shape guidance based on histogram thresholding (e.g, Otsu's method) or clustering (e.g., K-Means, Mean-Shift, Hierarchical clustering) and with shape guidance using image regions (e.g., Region Growing) or model energy (e.g., Markov Random Fields, Level sets) [39]. Image classification can be conducted using global feature descriptors (e.g., Haralick features) [37] in combination with a classifier. Registration of images can be performed based on image intensity or features and a similarity measure (e.g., cross-correlation, mutual information) [39]. Object tracking can be based on object detection (tracking-by-detection). The detections can be linked into tracks (e.g., using a Kalman filter) [40]. Most traditional image analysis methods include hand-crafted features. Development of good feature extractors requires deep domain knowledge. Deep learning can be used to learn feature extractors without the need of explicitly modelling domain knowledge. These feature extractors can be used in various image analysis methods. Therefore, feature extraction and classification can be performed jointly by using deep learning. State-of-the-art in image analysis methods for detection and segmentation based on deep learning are described in Chapter 2.

Finally, the extracted features (e.g., cell count, cell elongation, mean stain intensity, tissue texture statistics, particle velocity) are quantified. Often, rule-based filtering is used for the quantified data to account for sup-optimal image analysis results. The resulting *readout* is used to reach a medical or biological conclusion along with the hypothesis.

Software can help biologists to use existing image analysis methods on their image data [41]. If no appropriate method for certain data is available or the required image analysis pipeline is too complicated, image analysis researcher are consulted. In these research projects, image analysis researchers and biologists collaborate closely. In Figure 1.7, a typical workflow of such a project is sketched. The medical or biological cooperation partner produces image *data* which is handed to the image analysis researcher. The image analysis researcher develops a method and a corresponding *software* and uses it to generate the desired *readout* (e.g., cell counts, cell phenotypes,

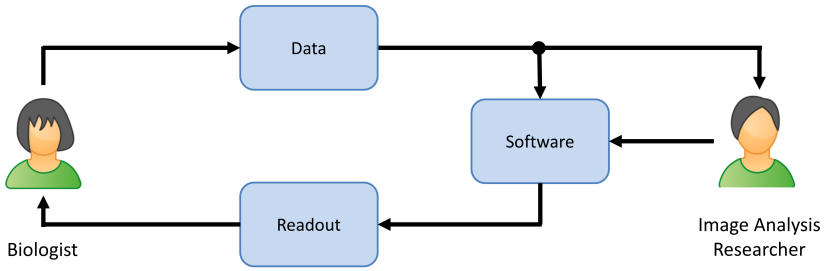


Figure 1.7: Workflow for general image analysis research projects

particle behavior). The readout is given to the cooperation partner to answer their biological or medical research hypothesis. Image acquisition and image analysis methods are continuously improved during the project. However, often the image analysis researcher has to run the software for a new dataset even if no changes to the software have been introduced, since most of the experimental image analysis pipelines are too cumbersome to use and therefore cannot be run by the biologist.

1.2 Contributions

This thesis proposes methods for easing frequent problems with biomedical microscopy datasets, improves deep learning models for biomedical computer vision, and presents a concept for deployment of cutting-edge algorithms in biomedical research projects. More specifically, the main contributions of this thesis are:

- **DetNet – Deep Neural Network for Particle Detection:** A new method for particle detection in microscopy images is proposed which uses deep learning and is based on a domain-adapted Deconvolution Network. Compared to standard deep neural network architectures, the number of parameters is significantly reduced. The method achieved better detection and localization results than previous methods.

- **Deep Residual Hough Voting for Mitotic Cell Detection in Histopathology Images:** A new method for mitotic cell detection in histopathology images is proposed which is based on a Deep Residual Network architecture combined with Hough voting. A voting layer for neural networks is proposed. Also, a novel loss function is introduced, which exploits polar coordinates and is invariant to the absolute magnitude of the voting error. The network is learned from scratch using cell centroids. In addition, a new method for grading whole-slide histology images of invasive breast carcinoma is proposed which is based on mitotic cell detection by a Deep Residual Network. The method combines a threshold-based attention mechanism and a deep neural network for mitotic cell detection and grading.

- **Deep Consensus Network for Particle and Cell Detection:** A new deep neural network named Deep Consensus Network (ConsensusNet) for particle and cell detection in microscopy images based on object centroids is introduced. The network is trainable end-to-end and comprises a Feature Pyramid Network-based feature extractor, a Centroid Proposal Network, and a layer for ensembling detection hypotheses over all image scales and anchors. Also, an anchor regularization scheme that favours prior anchors over regressed locations is suggested. In addition, an improved algorithm for Non-Maximum Suppression which significantly reduces the algorithmic complexity, is introduced. The method was applied to challenging data from the TUPAC16 mitosis detection challenge and the Particle Tracking

challenge and generally yielded better results than DetNet, Deep Residual Hough Voting, and previous methods.

- **Loss Function for Strong Class Imbalance in Object Detection:** For the Deep Consensus Network, a novel loss function based on Normalized Mutual Information is proposed. The loss can cope with strong class imbalance and is derived within a Bayesian framework.

- **ASPP-Net for Cell Segmentation:** A deep learning method leveraging atrous spatial pyramid pooling (ASPP) for cell segmentation is introduced. The ASPP increases the receptive field of the network to capture rich semantic information. The method is used in a workflow for large scale quantification of telomere length and PITX1 expression per cell.

- **GRUU-Net – Integrated Convolutional and Gated Recurrent Neural Network for Cell Segmentation:** The dominant paradigm in segmentation is using convolutional neural networks, less common are recurrent neural networks. A new deep learning method for cell segmentation is proposed which integrates convolutional neural networks and gated recurrent neural networks over multiple image scales to exploit the strength of both types of networks. The method was applied to images of cells from various modalities and yielded better results than previous methods.

- **Loss Function to Cope with Difficult Samples in Image Segmentation:** To increase the robustness of the training and improve segmentation, a novel focal loss function for GRUU-Net is introduced. A distributed scheme for optimized training of the integrated neural network is presented as well.

- **Hyperparameter Optimization:** A framework for zero-order black-box hyperparameter optimization called HyperHyper is presented which has a novel modular architecture that separates hyperparameter sampling and optimization. A visualization of the loss function based on infimum projection to obtain further insights into the optimization problem is also introduced.

- **Multi-Channel Deep Transfer Learning:** Two different approaches for transfer learning using fluorescence images of glioblastoma cell tissue with a different number of color channels are presented. The approaches exploit the similarity of source image channels and target image channels and are based on the ASPP-Net.

- **Unsupervised Domain Adaption for End-to-End Grading of Whole-Slide Images:** A novel deep learning method for domain adaption and classification of whole-slide images and patient level breast cancer grading is described. The proposed method is based on domain adaptation using a Cycle-Consistent Generative Adversarial Network (CycleGAN) in conjunction with a densely connected deep neural network.

- **Web-Based Microscopy Image Analysis:** The platform Galaxy Image Analysis for automated microscopy image analysis and cellular phenotyping within the Galaxy platform is introduced. Workflows for cell segmentation in cell culture images, particle detection in mice brain tissue data, and MALDI-/H&E image registration based on Galaxy Image Analysis are presented.

1.3 Organization of the Thesis

The thesis describes novel deep learning methods for object detection and segmentation. In Chapter 2, fundamentals of deep learning and previous work on object detection and segmentation are outlined. Chapter 3 introduces novel deep learning methods for detection of particles and cells. In Chapter 4, deep learning methods for segmentation of cells and an application to telomere quantification in tissue images are proposed. Chapter 5 introduces a framework for hyperparameter optimization of software pipelines in microscopy image analysis. Chapter 6 describes new methods for transfer learning for microscopy images. The evaluation of the proposed deep learning methods is presented in Chapter 7. A web-based framework for microscopy image analysis named Galaxy Image Analysis and applications are described in Chapter 8.

An overview of the developed methods, their connections, and a classification into the categories *Dataset Challenges*, *Deep Learning Methods*, and *Deployment* described above (cf. Figure 1.1) is given in Figure 1.8.

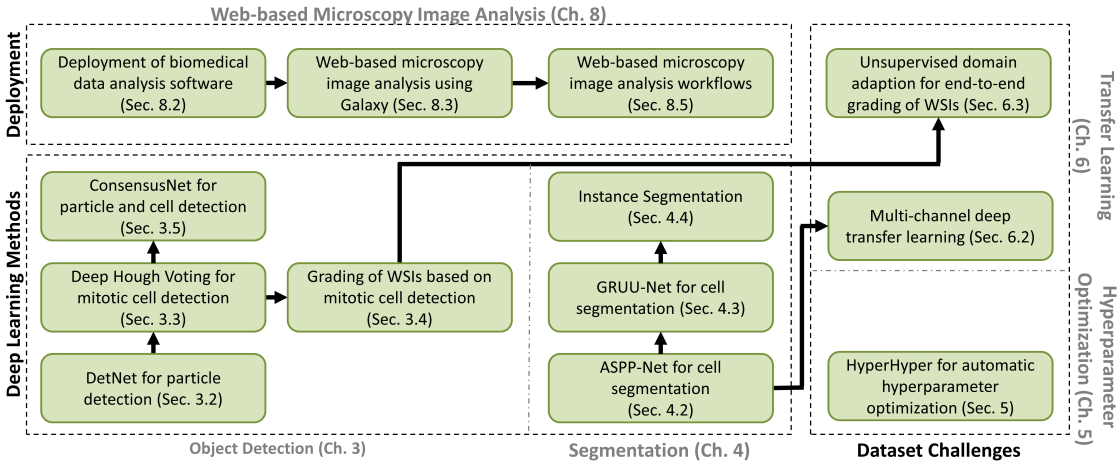


Figure 1.8: Connectivity of sections and chapters in this thesis. The main topics of the thesis are highlighted in bold. Connections between sections indicate that the described methods build on each other.

2 Foundations and Previous Work

In this chapter, fundamental concepts that are essential for this thesis are introduced. In particular, foundations of deep learning for computer vision and recent developments in the field of microscopy image analysis for detection and segmentation are reviewed. In this chapter, the fundamentals of deep learning for computer vision are presented. A review of state-of-the-art deep learning methods for object detection using bounding-boxes and centroids as well as semantic segmentation is conducted. Furthermore, applications and extensions of these methods for microscopy image analysis are elaborated on.

2.1 Deep Neural Networks for Computer Vision

Artificial Neural Networks (ANN) are a family of computation graphs loosely inspired by biological neural networks in animal brains. In an ANN, each layer of q neurons has the weight parameters $\mathbf{W} \in \mathbb{R}^{p \times q}$ and bias parameters $\mathbf{b} \in \mathbb{R}^q$, where p is the number of neuron activations of the previous layer. The activation $y_i \in \mathbb{R}$ of the i-th neuron is calculated by a weighted sum of the neuron activations of the previous layer $\mathbf{x} \in \mathbb{R}^{p \times 1}$ using the weights $\mathbf{w}_i \in \mathbb{R}^{p \times 1}$ and the bias $b_i \in \mathbb{R}$ followed by an activation function σ. Therefore, the activation $y_i \in \mathbb{R}$ of the i-th neuron in a layer is as follows:

$$y_i = \sigma((\mathbf{w}_i)^\intercal \mathbf{x} + b_i) \tag{2.1}$$

The *activation function* σ can be the identity function (linear function) or a non-linear function. Non-linear functions are required to capture the properties of complex data distributions. The orchestrated structure of multiple network components (e.g., layers, activations) is called network architecture. ANNs learn to perform a specific task by changing its parameters in the training phase using an optimization algorithm. The application of a trained network on data is called inference.

An ANN whose computation graph is directed and cycle-free is called *feed-forward neural network* (Figure 2.1). An early ANN is the Perceptron, which has one layer and the Heaviside step function as activation function [42]. If the network has at least one hidden layer, it is called multi-layer perceptron (MLP). More generally, an ANN with multiple consecutive hidden layers is called *Deep Neural Network* (DNN). The universal approximation theorem [43, 44] states that under mild assumptions (e.g., if σ is a non-constant, bounded, and a continuous function), a multi-layer feed-forward

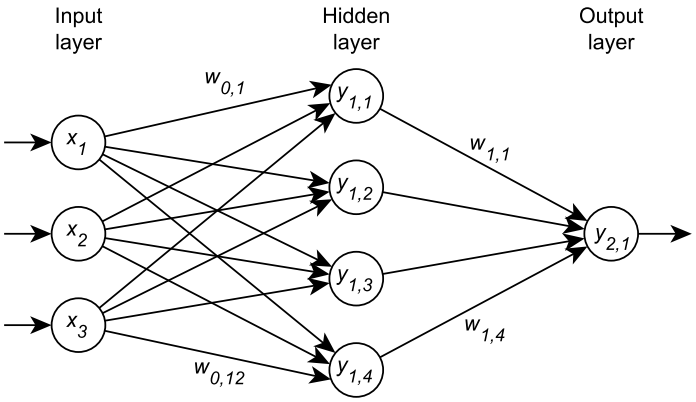


Figure 2.1: Feed-forward neural network

neural network can approximate any continuous function in a compact set $\Omega \in \mathbb{R}^n$ with an error bound which decreases with an increased number of neurons. Therefore, even a neural network with one hidden layer is an universal approximator. This raises the question how DNNs with a large number of parameters can be learned effectively in practice [45]. Recent theories state that an underlying hierarchical generation process exists in natural data, which can be decomposed into smaller problems by using the mutual information chain rule [46, 47]. Due to weight sharing within layers (see Section 2.1.1) and hierarchical composition of neurons, the underlying structure in data can be exploited to ease the network's training [47]. Moreover, the stochastic algorithms for training neural networks favour local optima with good generalization properties [48, 49, 50, 51].

Model Training

A popular algorithm for training DNNs is *stochastic gradient descent* (SGD) in combination with *backpropagation* [52]. Firstly, the current output of the network for a *batch* of samples is calculated, which is called *forward pass*. Next, an error scoring function (loss) $\mathcal{L} \in \mathbb{R}$ is calculated based on the corresponding reference. In the *backward pass*, the i-th weight $w_i \in \mathbb{R}$ of a neuron is updated in each iteration t of the optimization using the SGD update rule with learning rate $\eta \in \mathbb{R}^+$:

$$w_i^t \leftarrow w_i^{t-1} - \eta \nabla_{w_i^{t-1}} \mathcal{L} \tag{2.2}$$

Assuming that $\mathcal{L}$ is a function of y and y is a function of x, the gradient of the loss $\nabla\mathcal{L}$ for variables in lower layers can be back propagated using the chain rule of derivatives:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial x} \tag{2.3}$$

Therefore, unlike canonical gradient descent, SGD computes a parameter update for every batch instead for the whole dataset [53]. The incremental gradient descent of SGD introduces noise which helps escaping local minima and tends to more flatter minima than canonical gradient descent [48]. However, the surface of the loss, which depends on the data and the network's architecture as visualized in [49], can be flat or spiky in some regions. To exploit the shape of the loss surface in the optimizer's trajectory, an adaptive step size is advantageous. Therefore, the momentum of convergence is often used which also speeds up training and reduces oscillation of the loss. *RMSprop* is a popular extension to SGD, first presented by G. Hinton in a lecture of his Coursera class, which adaptively reduces the learning rate by exponentially decaying it by the squared derivative $(\nabla_{w_i^{t-1}}\mathcal{L})^2 \in \mathbb{R}$ [54]. The influence of the momentum can be changed by $\beta \in \mathbb{R}^+$, which is usually set to 0.9:

$$w_i^t \leftarrow w_i^{t-1} - \eta \frac{\nabla_{w_i^{t-1}}\mathcal{L}}{\sqrt{v_i^t + \epsilon}} \tag{2.4a}$$

$$v_i^t \leftarrow \beta v_i^{t-1} + (1-\beta)(\nabla_{w_i^{t-1}}\mathcal{L})^2 \tag{2.4b}$$

where ϵ is a small constant to avoid division by zero. *Adam* [55] is a more recent popular alternative to RMSprop, which in addition to the average of the squared derivative $v_i^t \in \mathbb{R}$ also uses the average of the derivative $m_i^t \in \mathbb{R}$. The influence of m_i^t and v_i^t can be changed by the parameters $\beta_1 \in \mathbb{R}^{[0,1]}$ and $\beta_2 \in \mathbb{R}^{[0,1]}$, which are usually set to 0.9 and 0.999 [55]:

$$w_i^t \leftarrow w_i^{t-1} - \eta \frac{\hat{m}_i^t}{\sqrt{\hat{v}_i^t + \epsilon}} \tag{2.5a}$$

$$\hat{m}_i^t = \frac{m_i^t}{1-\beta_1} \tag{2.5b}$$

$$\hat{v}_i^t = \frac{v_i^t}{1-\beta_2} \tag{2.5c}$$

$$m_i^t \leftarrow \beta_1 m_i^{t-1} + (1-\beta_1)\nabla_{w_i^{t-1}}\mathcal{L} \tag{2.5d}$$

$$v_i^t \leftarrow \beta_2 v_i^{t-1} + (1-\beta_2)(\nabla_{w_i^{t-1}}\mathcal{L})^2 \tag{2.5e}$$

Reddi et al. [56] pointed out that the convergence of Adam can be harmed, since $\hat{v}_i$ does not necessarily increase when the learning rate η is constant or decreasing. They proposed *AMSGrad*, which only keeps track of the maximum squared derivative, to

ensure a decreasing learning rate over the training iterations:

$$w_i^t \leftarrow w_i^{t-1} - \eta \frac{\hat{m}_i^t}{\sqrt{\hat{v}_i^t + \epsilon}} \tag{2.6a}$$

$$\hat{m}_i^t = \frac{m_i^t}{1 - \beta_1} \tag{2.6b}$$

$$\hat{v}_i^t = \frac{\max(v_i^t, \hat{v}_i^{t-1})}{1 - \beta_2} \tag{2.6c}$$

$$m_i^t \leftarrow \beta_1 m_i^{t-1} + (1 - \beta_1) \nabla_{w_i^{t-1}} \mathcal{L} \tag{2.6d}$$

$$v_i^t \leftarrow \beta_2 v_i^{t-1} + (1 - \beta_2) (\nabla_{w_i^{t-1}} \mathcal{L})^2 \tag{2.6e}$$

Activation Functions

Several *non-linear activation functions* for neural networks are described in the literature (e.g., [57, 58, 59, 22]). Due to the universal approximation theorem, their expressiveness in a DNN is similar when using enough neurons [43, 44]. Their main difference in practice is their effect on the gradient. A major challenge in training deep neural networks is the *vanishing gradient* problem which was identified by Hochreiter in 1991 [60]. Due to weight updates performed using backpropagation, the gradient is proportional to the partial derivatives of the loss function. Poor architectural choices can lead to contracting gradients which are accumulated by the chain rule of derivatives. Bounded functions (e.g., Sigmoid, Tanh), for example, also have a bounded gradient which can exponentially decrease the gradient when used in every layer. A similar effect can be observed with exploding gradients where gradients accumulate, resulting in large weight updates. These two effects lead to unstable training or underfitted models. The *Sigmoid function* squashes the input x to $[0, 1]$:

$$\text{Sigmoid}(x) = \begin{cases} \frac{1}{1+e^{-x}} & , x > 0 \\ \frac{e^x}{e^x+1} & , x \leq 0 \end{cases} \tag{2.7}$$

The output of the Sigmoid function is not zero-centred. This can be overcome with the hyperbolic tangent (Tanh), which squashes the input to $[-1, 1]$. Due to the bound of Sigmoid and Tanh, the optimization can be harmed. In 2011, Glorot et al. [57] proposed the *Rectified Linear Unit (ReLU)* as an unbound alternative to

previous activation functions:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.8}$$

When using ReLUs, the negative part of the neurons activation is not used. Therefore, neurons can be in a state where they become untrainable, since they have a derivative of zero. There are several variants of ReLUs that are designed to reduce this problem. The *Leaky Rectified Linear Unit (LReLU)* [58] makes use of a small negative component in its output by introducing a leakage parameter $a = 0.2$:

$$\mathrm{LReLU}(x, a = \mathrm{const}) = \begin{cases} x & , x > 0 \\ ax & , \text{otherwise} \end{cases} \tag{2.9}$$

The *Parametric Rectified Linear Unit (PReLU)* [59] is a generalization of LReLU, where the leakage parameter $a \in \mathbb{R}$ can be trained along with the network:

$$\mathrm{PReLU}(x, a) = \begin{cases} x & , x > 0 \\ ax & , \text{otherwise} \end{cases} \tag{2.10}$$

Network Weight Initialization

Weights in neural networks are initialized so that they keep a specific distribution over multiple layers, and additionally break symmetry so that each neuron can learn its distinct feature. Moreover, the activations should be zero-centered and have unit variance. State-of-the-art initialization schemes use values in the weight matrix $\mathbf{W}$ sampled from a scaled random uniform or Gaussian distribution. To ensure similar distributions of activations in the network, Glorot and Bengio [57] proposed to use a variance $\mathrm{Var}(\mathbf{W})$ of the sampling distribution based on the number of neurons n_{in} in the previous and the number of neurons n_{out} in the current layer:

$$\mathrm{Var}(\mathbf{W}) = \frac{2}{n_{\text{in}} + n_{\text{out}}} \tag{2.11}$$

When using activation functions like ReLU, half of the results are truncated. To overcome this, He et al. [59] proposed to only rely on the number of neurons in the previous layer:

$$\mathrm{Var}(\mathbf{W}) = \sqrt{\frac{2}{n_{\text{in}}}} \tag{2.12}$$

Model Regularization

State-of-the-art neural networks have millions of parameters and thus many degrees of freedom. They often have a learning capacity that exceeds the training dataset. In practice, the bias-variance tradeoff has to be considered to keep underfitting due

to high bias and overfitting due to high variance of the model in equilibrium [38]. By using regularization of the model, overfitting can be prevented. A common technique is to add ℓ_1 regularization:

$$\mathcal{L} = \mathcal{L}_{\text{obj}} + \lambda \sum_{i=1}^{N} |w_i| \tag{2.13}$$

or ℓ_2 regularization of the weight updates to the loss $\mathcal{L}_{\text{obj}}$:

$$\mathcal{L} = \mathcal{L}_{\text{obj}} + \lambda \sum_{i=1}^{N} (w_i)^2 \tag{2.14}$$

where N is the number of elements of $\mathbf{w}$. Regularization with ℓ_1 favours a sparse weight matrix and ℓ_2 favours a Gaussian distribution of the weights [38]. The influence of the regularization can be changed using the parameter $\lambda \in \mathbb{R}$. Another method of weight regularization using a norm is *weight decay*. Weight decay extends the SGD update rule (2.2) by adding a rescaling factor with a parameter λ to the weights:

$$w_i^t \leftarrow w_i^{t-1} - \eta \nabla_{w_i^{t-1}} \mathcal{L}(w_i^{t-1}) - \eta \lambda w_i^{t-1} \tag{2.15}$$

When using SGD, weight decay is similar to ℓ_2 regularization [61]. However, this does not hold for optimizers that use momentum. AdamW is an extension of Adam with weight decay [61]:

$$w_i^t \leftarrow w_i^{t-1} - \eta \frac{\hat{m}_i^t}{\sqrt{\hat{v}_i^t} + \epsilon} - \eta \lambda w_i^{t-1} \tag{2.16a}$$

$$\hat{m}_i^t = \frac{m_i^t}{1 - \beta_1} \tag{2.16b}$$

$$\hat{v}_i^t = \frac{v_i^t}{1 - \beta_2} \tag{2.16c}$$

$$m_i^t \leftarrow \beta_1 m_i^{t-1} + (1 - \beta_1)(\nabla_{w_i^{t-1}} \mathcal{L} + \lambda w_i^{t-1}) \tag{2.16d}$$

$$v_i^t \leftarrow \beta_2 v_i^{t-1} + (1 - \beta_1)(\nabla_{w_i^{t-1}} \mathcal{L} + \lambda w_i^{t-1}) \tag{2.16e}$$

Another method to prevent overfitting is *Dropout* [62], which adds Gaussian or Bernoulli noise to the layer's activations. Dropout can be interpreted as building random subgraphs within the neural network, which are robust to missing or wrong

inputs. Moreover, by adding noise to the activations, a Gaussian prior is put on the activations. In general, Dropout increases the training time, but improves the performance of the network. As the activation and weight distribution of neurons should not change throughout the network, the initial values of the parameters are initialized with a mean value of 0 and a unit variance of 1. However, in a forward pass, the distribution of activations can shift and scale, which also affects the weights as training progresses. This causes the weight distribution to degrade, slowing down training and triggering side-effects between layers. *Batch normalization (BN)* [63] is a simple, but effective technique which normalizes the activations in every layer and thus prevents the weight distributions from degradation. The shift $\beta \in \mathbb{R}$ and scale $\gamma \in \mathbb{R}$ of the activation distributions are learnable parameters of the architecture. The normalization is calculated for every batch with M samples using the mean value $\mu_b \in \mathbb{R}$ and the variance $\sigma_b^2 \in \mathbb{R}^+$. Therefore, these estimates are noisy, and batch normalization acts as a regularizer which often eliminates the need for Dropout [64]. During inference, the mean value $\mu \in \mathbb{R}$ and the variance $\sigma^2 \in \mathbb{R}^+$ of the training dataset are used instead of the batch wise μ_b and σ_b^2:

$$\mathrm{BN}(x_i) \equiv \gamma\, \hat{x}_i + \beta \tag{2.17a}$$

$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} \tag{2.17b}$$

$$\mu_b = \frac{1}{M} \sum_{m=1}^{M} x_m \tag{2.17c}$$

$$\sigma_b^2 = \frac{1}{M} \sum_{m=1}^{M} (x_m - \mu_b)^2 \tag{2.17d}$$

In [65], the normalization is performed for each instance of a batch individually. Moreover, *instance normalization* is also applied during inference without learning any shift or scale parameters. In some use cases where dataset-wide shift and scale parameters are hard to learn, instance normalization can still simplify the learning process [65, 66]. A simple but effective method to improve generalization of the model is data augmentation. *Data augmentation* is the process of distorting data to enforce the network to learn invariance to these distortions. Common augmentations are flipping, rotation, adding Gaussian or Poisson noise, color shifting, histogram stretching, and elastic deformations [22]. Moreover, crops of a larger image can be sampled to specifically analyze more important regions [67]. Controlling the sampling of difficult cases and easy cases is referred to as *hard negative mining*. If the order of the samples is following a specific scheme, it is referred to as curriculum learning

[68]. Data augmentation can be performed offline on the training dataset, and online during training and at test time [69]. *Test time augmentation* is usually performed to exploit the invariance of the model and boost the performance by averaging the predictions of several augmentations of the same input. In data augmentation, it is important that the augmentation steps do not alter the semantics of the input. In addition, excessive augmentation can lead to too much smoothing and therefore underfitting. In general, a combination of regularization of the model and data augmentation usually works best in practice [70]. By using proper regularization, not only overfitting is avoided, but fast training of the model can be observed, which is referred to as superconvergence [71].

Loss Functions

For neural networks, several training objectives (loss functions) are used in computer vision. For classification tasks, the *cross-entropy (*CE*)* is commonly used, which measures the average bits needed to encode an event drawn from probability distribution Q instead of the true distribution U, which is the sum of the entropy H of U and the Kullback-Leibler divergence $\mathrm{D_{KL}}$ of U with respect to Q where $\mathbb{E}$ is the expected value operator with respect to the distribution U [38, 22]:

$$\mathrm{CE}(U, Q) = \mathbb{E}_U[-\log(Q)] = \mathrm{H}(U) + \mathrm{D_{KL}}(U, Q) \tag{2.18}$$

In the discrete case, where $P(\mathbf{X}) \in \mathbb{R}^M$ are the predicted probabilities for the samples $\mathbf{X} = (X_1, ..., X_M)$ from Q and $P(\mathbf{Y}) \in \mathbb{R}^M$ the ground truth probabilities for the labels $\mathbf{Y} = (Y_1, ..., Y_M)$ from U, we can define a CE-based loss $\mathcal{L}_{\mathrm{CE}}$ for M samples:

$$\mathcal{L}_{\mathrm{CE}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{m=1}^{M} -P(Y_m) \log(P(X_m)) \tag{2.19}$$

CE is defined for discrete events. Thus, classes are usually encoded in an indicator vector (one-hot vector). Therefore, the output is one for the index of one class and zero for all other classes. The output of the network can then be interpreted as a probability distribution across the classes. Classification can also be viewed regarding set theory. Based on the Dice coefficient, a smooth approximation can be minimized over the sets $\mathbf{X}$ and $\mathbf{Y}$, where a small ϵ prevents a division by zero [72, 73]:

$$\mathcal{L}_{\mathrm{Dice}}(\mathbf{X}, \mathbf{Y}) = -\frac{2 \sum_{m=1}^{M} P(X_m)\ P(Y_m) + \epsilon}{\sum_{m=1}^{M} P(X_m) + \sum_{m=1}^{M} P(Y_m) + \epsilon} \tag{2.20}$$

The $\mathcal{L}_{\mathrm{Dice}}$ *loss* has the advantage of implicitly weighting the loss using the class imbalance in the ground truth and the number of activations of the network. The Jaccard similarity coefficient and the Cosine similarity can also be used analogously as training objective. For regression tasks, usually a ℓ_p norm is used [22]. The ℓ_1 norm

$\|\cdot\|_1$, which is also known as taxicab metric, should not be favored for regression, since it has no always a unique solution and is therefore hard to optimize [38]. However, it can be used to enforce sparse results, which is important in regularization for tasks such as dictionary learning [74]. The ℓ_2 norm $\|\cdot\|_2$ is also known as Euclidean norm, has a unique solution, and is therefore the most common objective for regression. To further improve the stability of the training process, the squared ℓ_2 norm is used in the $\mathcal{L}_{\ell_2}$ loss, which is also known as *mean squared error (MSE)*:

$$\mathcal{L}_{\ell_2}(\mathbf{v}, \mathbf{v}^{\mathrm{GT}}) = \frac{1}{M} \sum_{m=1}^{M} \|v_m - v_m^{\mathrm{GT}}\|_2^2 \tag{2.21}$$

where $\mathbf{v} \in \mathbb{R}^M$ denotes the predicted values and $\mathbf{v}^{\mathrm{GT}} \in \mathbb{R}^M$ the ground truth values. The $\mathcal{L}_{\ell_2}$ loss is prone to outliers since errors contribute quadratically to the total error. Huber [75] proposed a piecewise defined loss $\mathcal{L}_{\mathrm{H}}$ with hyperparameter $\delta \in \mathbb{R}$, which is more robust than the $\mathcal{L}_{\ell_2}$ loss:

$$\mathcal{L}_{\mathrm{H}}(\mathbf{v}, \mathbf{v}^{\mathrm{GT}}) = \frac{1}{M} \sum_{m=1}^{M} \begin{cases} \frac{1}{2}(v_m - v_m^{\mathrm{GT}})^2 & , |v_m - v_m^{\mathrm{GT}}| \leq \delta \\ \delta(|v_m - v_m^{\mathrm{GT}}| - \frac{1}{2}\delta) & , \text{otherwise} \end{cases} \tag{2.22}$$

Instead of explicitly formulating an objective, the loss function can also be learned along with the network [76] by using *adversarial training* [77]. In adversarial training, a discriminator learns to classify a dataset, but engineered adversarial examples are added to fool the discriminator. It has been shown that neural networks are vulnerable to adversarial examples [78]. This method can be used by training a discriminator network to distinguish adversarial or original samples. A generator network is trained concurrently to generate adversarial samples from noise and is updated alternately to the discriminator network. This zero-sum game between the two networks forms an actor-critic model [79] known from reinforcement learning. The described combination of a generator and discriminator network known as Generative Adversarial Network (GAN) was introduced by Goodfellow et al. [76].

The last layer of the ANN (see Figure 2.1) is the *output layer*. In a multi-class problem with C classes, and a class c is mutually exclusive, a generalization of the logistic function (Sigmoid) is used to squash the output vector $\mathbf{x} \in \mathbb{R}^C$ of the last layer into a probability distribution. The so called *SoftMax function* represents a categorical distribution and is defined by:

$$\mathrm{SoftMax}(\mathbf{x}) = \frac{\mathbf{e}^{\mathbf{x}}}{\sum_{c=1}^{C} \mathbf{e}^{x_c}} \tag{2.23}$$

Learning Schemes and Transfer Learning

In *supervised learning*, the model learns a function of an input sample to a ground truth label [38]. For this reason, labels have to be available during training. In

unsupervised learning, the model learns to detect frequent patterns from the training samples which is also known as clustering [38]. *Semi-supervised learning* combines supervised and unsupervised learning [80]. Training neural networks is computationally intensive. Moreover, obtaining curated and labeled datasets is cumbersome. Various techniques of transfer learning try to reuse models and data. In transfer learning, generalizable knowledge has to be extracted and negative knowledge transfer should be avoided [81]. *Transfer learning* methods can be categorized in inductive and transductive transfer [82]. In *inductive transfer*, knowledge is transferred to a new task. Inductive techniques comprise, for example, multi-task learning and self-taught learning. *Transductive transfer learning* can also be used to transfer knowledge to new domains (domain adaption). In transductive transfer learning, knowledge is transferred from a source domain to a target domain, where informed techniques require labeled data from the target domain and supervised techniques labeled data from the source domain [81]. Uninformed and unsupervised techniques do not need labeled data from the respective domains. A popular approach used in combination with neural networks is pre-training the model on a large dataset and fine-tuning the model on a target dataset with less labels using a lower learning rate. The pre-training dataset can also be obtained from a simulation or a generative model trained on real data in an unsupervised scheme. Moreover, it can be beneficial to fine-tune only the network's biases and scaling parameters, if batch normalization is used. The latter technique only compensates a covariate shift of the source to the target domain. An easy, but effective method is introducing a sample selection bias, where the model is trained on samples from the source and target domain. However, only source-domain samples that match the data distribution of the target domain are used, which is referred to as instance transfer. *Domain adaption* for deep neural networks is a very active research field. In recent works, several deep neural network components for domain adaption are proposed [83, 84, 85, 86]. In [83], Ganin et al. use an adversarial objective to learn domain-invariant features by predicting the domain of a sample in a separate branch. They propose a gradient reversal layer, which enforces that the features in the base-feature extractor are optimized to be domain invariant, while the features in the domain classifier are domain descriptive. For life-long learning over multiple domains, Bilen et al. [84] proposed to learn the same feature extractor over all available domains. However, the parameters for the batch normalization layers are multiplexed over all available domains. Therefore, the network has to learn just a few scale and shift parameters for each domain to compensate covariate shift. In [85], Tamaazousti et al. propose to learn a network on fine and coarse object categories to obtain very specific and generic features. They use a SVM and an automatic relabeling strategy to learn a new classifier with their pre-trained features.

2.1.1 Convolutional Neural Networks

A *Convolutional Neural Network (CNN)* is a neural network architecture with spatially tied parameters. Therefore, a layer in a CNN can be interpreted as a convolution ($*$) with a learnable filter $\mathbf{W} \in \mathbb{R}^{k \times k \times p \times q}$ followed by an element wise non-linear activation function σ:

$$\mathbf{y} = \sigma(\mathbf{W} * \mathbf{x} + \mathbf{b}) \tag{2.24}$$

where $k \times k$ is the window size of the convolutional kernel, p the number of input feature maps, and q the number of output feature maps. Like traditional neural networks, CNNs have multiple layers. To increase or decrease the spatial resolution of the feature maps, pooling operators are used. Applying a convolutional kernel on a pooled feature map has a higher receptive field than applying it on a non-pooled feature map. Thus, the convolutional layer can capture information in a larger receptive field. Common pooling operators are max pooling, average pooling, interpolation, and strided convolution. Max pooling and average pooling can only reduce the spatial resolution [22]. *Max pooling* on a regular grid of values $\mathbf{I} \in \mathbb{R}^{w \times h}$ with a window function $f(i, j) = I_{ij}$ of two times k at grid position i, j is defined by:

$$\text{MaxPool}(i, j) = \max_{\substack{\Delta i \in \{-k, \ldots, k\} \\ \Delta j \in \{-k, \ldots, k\}}} f(i + \Delta i, j + \Delta j) \tag{2.25}$$

Equivalently, *average pooling* is defined by:

$$\text{AVGPool}(i, j) = \frac{1}{(2k)^2} \sum_{\Delta i=-k}^{k} \sum_{\Delta j=-k}^{k} f(i + \Delta i, j + \Delta j) \tag{2.26}$$

As an alternative, interpolation (e.g., bilinear, bicubic) can be used to increase or decrease the resolution of a grid. Convolutions can also be used to increase or decrease the resolution by adding strides, which means that the filter kernel is not shifted by one, but by a larger or smaller number of pixels. To increase computational efficiency when increasing the spatial resolution, forward pass and backward pass can be swapped using transposed convolution, which is also known as deconvolution, or fractionally strided convolution in literature [87]. In contrast to pooling, dilated convolutions can be used to computationally capture information within an increased receptive more efficiently [88]. *Dilated convolutions*, which are also known as atrous convolutions, introduce holes into the convolutional kernel that increase the kernel without adding additional parameters [88].

Over the past years, the effectiveness of CNNs have been significantly improved by development of new architectures. Several popular CNN architectures are outlined in the next paragraph. CNN architectures typically consist of stacks of convolutional layers and pooling layers followed by a fully connected layer (Dense layer). The

described structure is also called *base architecture* [89, 90, 91]. Finally, additional application-specific layers are appended (e.g., a SoftMax layer for classification). The fully connected layer can also be replaced by global average or max pooling, which performs the pooling operation in a window with the same spatial size as the feature maps. In 2012, AlexNet [1] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by reducing the top-5 error from 26% to 15.3%, which lead to significant increase of the use of neural networks in computer vision. The architecture of *AlexNet* is outlined in Figure 2.2. AlexNet consists of a convolutional layer with 48 different 3×3 filters followed by max pooling with Local Response Normalization (LRN) in a window of 3×3 and a stride of two. In the second layer, 128 filters are extracted using 3×3 filters followed by max pooling with LRN. The next three convolutional layers extract 192, 192, and 128 feature maps using a 3×3 convolution followed by a ReLU activation function. Finally, two fully connected layers with 2048 neurons each, Dropout, and SoftMax are used to predict the class. In 2014, the Visual Geometry Group (VGG) was the runner up of the ILSVRC using a new architecture which gained increasing popularity as base architecture in many applications. The so called *VGG architecture* [92] replaces the convolutional layers of AlexNet with two convolutional layers, each using a ReLU activation function. Variants with a total of 11, 16, and 19 layers, respectively called VGG-11, VGG-16, and VGG-19, were proposed. However, the ILSVRC 2014 was won by *GoogLeNet* [93], which aimed at reducing the computational complexity, and achieved a top-5 error rate of 6.67%. Its layers had a variable receptive field due to their novel Inception layers. Inception layers perform multiple convolutions using different kernel sizes and concatenate the results into a feature stack. The *Inception architecture* was improved in [94, 95]. The idea of the Inception layer was further used in [28] in their atrous

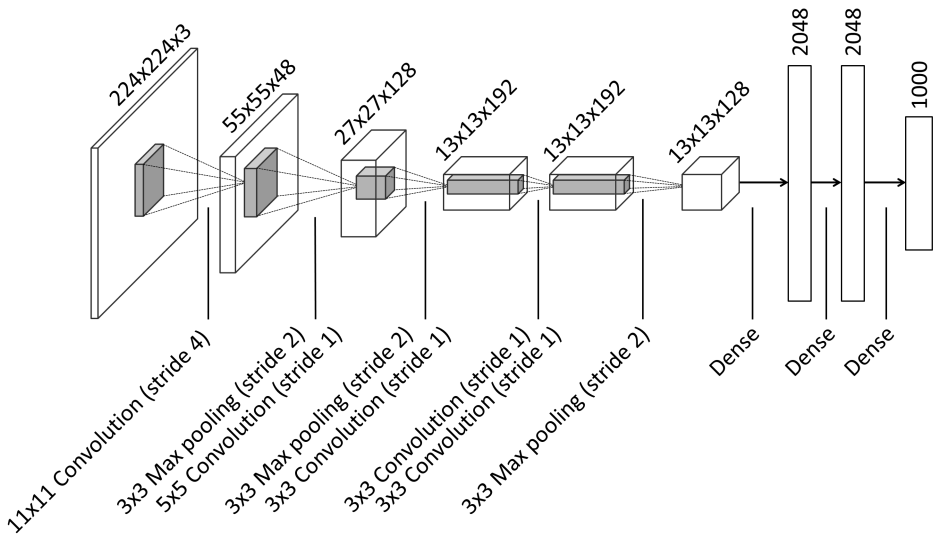


Figure 2.2: AlexNet architecture [1]. The feature maps tensor size is outlined at the top and the detailed layer configuration is delineated at the bottom.

spatial pyramid pooling (ASPP) blocks. Instead of using larger convolutional filters, dilated convolutions are used in ASPP blocks to reduce the number of parameters. The winner of ILSVRC 2015 was the *Residual Network architecture (ResNet)* [29] with a top-5 error rate of 3.57%. In ResNets, the input $\mathbf{x}_i \in \mathbb{R}^{w \times h \times p}$ is added to the output of a small subnetwork $\mathcal{F} \in \mathbb{R}^{w \times h \times q}$ with parameters $\mathbf{W}_i \in \mathbb{R}^{k \times k \times p \times q}$ to reduce gradient vanishing, where $m \times n$ is the spatial feature map size, p the number of input filters, q the number of output filters, and k the window size of the convolutional kernel:

$$\mathbf{y}_i = \mathbf{x}_i + \mathcal{F}(\mathbf{x}_i; \mathbf{W}_i) \tag{2.27}$$

Adding the input to the output of the residual is referred to as *skip connection*. Carefully designed recurrent units, which are explained further in Section 2.1.2, are capable of using a residual as shown in [96]. In 2017, the concept of residual connections was extended by the *Densely Connected Neural Network (DenseNet)* architecture [30]. The authors introduced Densely Connected blocks, where each layer has access to all feature maps of the previous layers. Therefore, layer i receives the concatenated feature maps $[\mathbf{x}_1; ...; \mathbf{x}_i]$ as input:

$$\mathbf{y}_i = \mathcal{F}(\mathbf{x}_1; ...; \mathbf{x}_i; \mathbf{W}_i) \tag{2.28}$$

2.1.2 Recurrent Neural Networks

A *Recurrent Neural Network (RNN)* is an ANN for processing sequential data. In 1982, John Hopfield combined previous ideas to propose the Hopfield network, which was one of the first RNNs [97]. The idea of an RNN is to use the same block on each input $\mathbf{x}_t \in \mathbb{R}^p$ of the sequence with $0 < t \leq T$ elements to produce the outputs $\mathbf{o}_t \in \mathbb{R}^q$, which is called unfolding (Figure 2.3). Therefore, an RNN is distinct from feedforward neural networks, since it forms a cycled computation graph. Information between consecutive steps are passed using an internal state $\mathbf{h}$ and forms a directed graph along the sequence.

The *Long Short-Term Memory (LSTM)* is a popular implementation of RNN units [98]. It was developed to handle exploding and vanishing gradients, which can occur in a naive RNN implementation. An LSTM uses the previous state $\mathbf{h}_{t-1}$, the previous memory cell $\mathbf{c}_{t-1}$, and the current input $\mathbf{x}_t$ to compute the output $\mathbf{o}_t$, the current

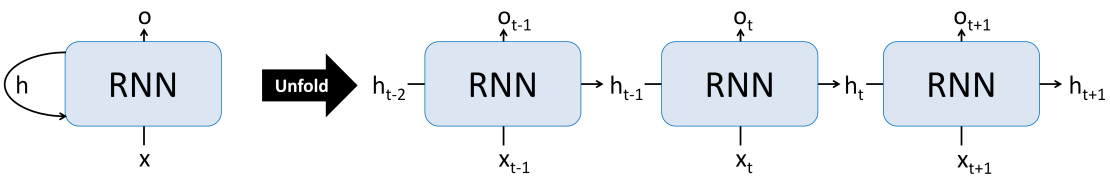


Figure 2.3: Illustration of unfolding RNN over observations $\mathbf{x}$.

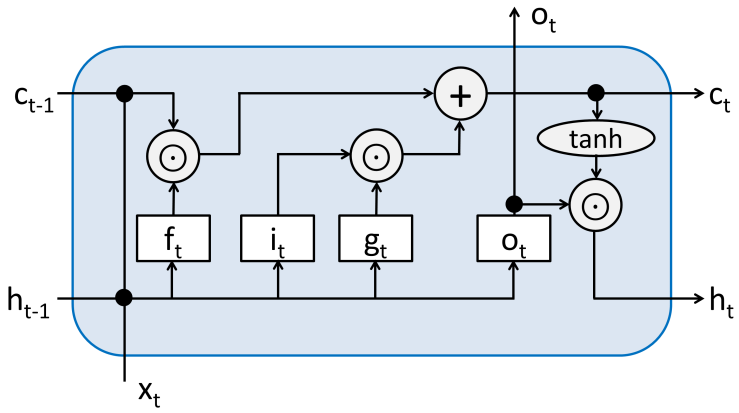


Figure 2.4: LSTM architecture

memory cell $\mathbf{c}_t$, and the current state $\mathbf{h}_t$ (Figure 2.4). First, the input $\mathbf{x}_t$ and the previous state $\mathbf{h}_{t-1}$ are weighted to calculate the forget gate $\mathbf{f}_t$ [99], which regulates reset of the memory cell, the input gate $\mathbf{i}_t$, which controls weighting of the input, and the output gate $\mathbf{o}_t$, which controls the computation of the output activation:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f^\intercal \mathbf{x}_t + \mathbf{U}_f^\intercal \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.29}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i^\intercal \mathbf{x}_t + \mathbf{U}_i^\intercal \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.30}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o^\intercal \mathbf{x}_t + \mathbf{U}_o^\intercal \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.31}$$

Afterwards, the candidate state $\mathbf{g}_t$ is calculated and fused with the previous memory cell $\mathbf{c}_{t-1}$ to calculate the current memory cell $\mathbf{c}_t$. The operator $\odot$ denotes the Hadamard product.

$$\mathbf{g}_t = \tanh(\mathbf{W}_g^\intercal \mathbf{x}_t + \mathbf{U}_g^\intercal \mathbf{h}_{t-1} + \mathbf{b}_g) \tag{2.32}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{2.33}$$

Finally, the new state $\mathbf{h}_t$ is calculated using the current memory cell:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.34}$$

A *Gated Recurrent Unit (GRU)* is an optimized RNN with similar performance as an LSTM with less parameters. The structure of a GRU is sketched in Figure 2.5. In contrast to an LSTM, a GRU has just a single output $\mathbf{h}_t$, which is concurrently the new state and output. First, the reset gate $\mathbf{r}_t$ and update gate $\mathbf{z}_t$ are calculated

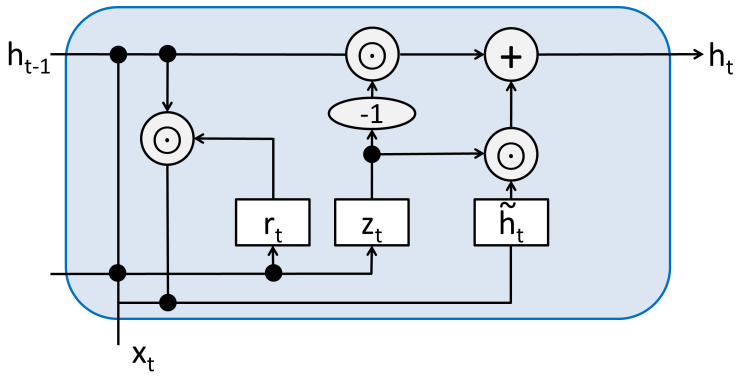


Figure 2.5: GRU architecture

using the input $\mathbf{x}_t$ and the parameters $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{b}_r$, $\mathbf{W}_z$, $\mathbf{U}_z$, and $\mathbf{b}_z$:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r^\intercal \mathbf{x}_t + \mathbf{U}_r^\intercal \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.35}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z^\intercal \mathbf{x}_t + \mathbf{U}_z^\intercal \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.36}$$

Then, the candidate state $\tilde{\mathbf{h}}_t$ is calculated using the parameters $\mathbf{W}_h$, $\mathbf{U}_h$, $\mathbf{b}_h$:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h^\intercal \mathbf{x}_t + \mathbf{U}_h^\intercal (\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{2.37}$$

Finally, the previous state $\mathbf{h}_{t-1}$ and the candidate state $\tilde{\mathbf{h}}_t$ are weighted to determine the new state $\mathbf{h}_t$:

$$\mathbf{h}_t = \mathbf{o}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \tag{2.38}$$

2.2 Object Detection

Object detection is the task of detecting semantic instances of specific objects within images. The detection result can be described using the bounding-box of an object or a location, such as the centroid. In this section, object detection using deep learning is introduced and specialized methods for microscopy images are presented.

2.2.1 Methods in General Computer Vision

One of the first works on *bounding box-based object detection* using deep neural networks was *R-CNN* by Girshick [100]. R-CNN uses the Selective Search algorithm [101] to extract region proposals from an image. These regions are cropped and rescaled. A CNN with VGG architecture is used to extract features from the cropped image patches and an SVM is used to classify object proposals into object categories. Hence, these networks are denoted as *multi-stage object detectors*. The author

extended his approach to *Fast R-CNN* [102], which computes the CNN features just once for the image before cropping and replacing the SVM with a classification network. The regression is performed using the Huber loss for the bounding-box and classification using the cross-entropy loss for the object categories. To learn both networks simultaneously, *Region of Interest Pooling (ROI Pooling)* is introduced, which can backpropagate the gradient through the rescaling step. In *Faster R-CNN*, Ren et al. replaced the region proposal algorithm with a *Region Proposal Network (RPN)*, which directly proposes object candidates from the base CNN [23]. The RPN predicts offsets to predefined bounding boxes with different aspect ratios (anchors) and corresponding scores. Greedy *non-maximum suppression (NMS)* is used to only pass boxes with high confidence and no overlap to the ROI Pooling step.

In the literature, the subnetwork that predicts object detections on top of extracted feature maps is often referred to as *detection head*. In 2015, Redmon et al. introduced *You Only Look Once (YOLO)* [24], which, in comparison to Faster R-CNN, is a *single stage detector* and uses the Darknet [24] instead of the VGG architecture. The YOLO approach was further improved in their follow-up works [103], [104]. YOLO directly performs regression on the offsets to predefined bounding-box anchors and corresponding confidences. NMS just has to be performed during inference. In general, YOLO has faster inference than Faster R-CNN [24]. *Single Shot MultiBox Detector (SSD)* introduced the detection of objects at different pooling stages of the network [105]. In their follow-up work, Redmon et al. introduced YOLOv3 [104], which also incorporated this technique. *Feature Pyramid Networks (FPN)* [106] is an extension of this technique by using a down- and upstream path to extract detections at different scales, similarly to a technique for semantic segmentation previously proposed by Ronneberger [27]. *RetinaNet* [107] introduces *Focal loss*, which scales the binary classification loss of predictions with low predicted confidence (hard examples), where $\gamma \geq 0$ is a modulating factor for negative mining:

$$\begin{aligned}\mathcal{L}_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{m=1}^{M} (&-(1 - P(X_m))^{\gamma} P(Y_m) \log(P(X_m)) \\ &- (x_m)^{\gamma} (1 - P(Y_m)) \log(1 - P(X_m)))\end{aligned} \tag{2.39}$$

2.2.2 Methods for Microscopy Image Data

Bounding box-based detection networks have been successfully applied to microscopy images. Rao [108] uses a modified Faster R-CNN for mitosis detection in H&E stained histological images. Akram et al. [109, 110] use a Faster R-CNN for cell tracking. The Faster R-CNN generates object proposals and a U-Net is used to segment cells. A graphical model is used to perform further cell tracking.

In comparison to bounding box-based object detection, centroid-based object detection tries to find only the centroid of an object without estimation of the bounding box. *Centroid-based object detection* is a frequent task in microscopy image

analysis, as the number of cells or particles and their distribution is often required by biologists or pathologists [111, 112, 113, 114, 115, 116, 117, 118]. Classification using a sliding window is a common approach for centroid-based object detection [111, 113, 114, 115, 116]. Cireşan et al. proposed using multiple, independently trained neural networks to predict the presence of a mitotic cell in the center of a sliding window [111]. They average the results to calculate their final prediction. Moreover, they use hard negative mining to improve their performance. Since mitotic and apoptotic cells look relatively similar, apoptotic cells are often mistaken as mitotic cells. Therefore, they train their network on mitotic cells (positive samples) only. Afterwards, they perform inference on their training dataset and extract false positive detections as negative samples and finally retrain their network using the extracted positive and negative samples. In [113], the proposed neural network predicts at each sliding window position M offset vectors $\mathbf{z}$, $m = 1, ..., M$, and corresponding confidences $\mathbf{h}$. A cross-entropy loss between $x_j \in \mathbb{R}$ and $y_j \in \mathbb{R}$ is used, where $\mathbf{c}_j \in \mathbb{R}^2$ is the j-th position in the predicted probability map, $\hat{\mathbf{z}}_m \in \mathbb{R}^2$ is the ground truth position of the m-th mitotic cell, $\mathbf{z}_m \in \mathbb{R}^2$ the predicted mitotic cell position, $h_m \in \mathbb{R}$ the corresponding confidence, and $d \in \mathbb{R}$ the maximum distance threshold:

$$x_j = \begin{cases} \left(\frac{1}{1+\left(\|\mathbf{c}_j - \mathbf{z}_m\|_2^2\right)/2} \right) h_m & , \forall m \neq m', \|\mathbf{c}_j - \mathbf{z}_m\|_2 \leq \|\mathbf{c}_j - \mathbf{z}_{m'}\|_2 \leq d \\ 0 & , \text{otherwise} \end{cases} \tag{2.40a}$$

$$y_j = \begin{cases} \frac{1}{1+\left(\|\mathbf{c}_j - \hat{\mathbf{z}}_m\|_2^2\right)/2} & , \forall m \neq m', \|\mathbf{c}_j - \hat{\mathbf{z}}_m\|_2 \leq \|\mathbf{c}_j - \hat{\mathbf{z}}_{m'}\|_2 \leq d \\ 0 & , \text{otherwise} \end{cases} \tag{2.40b}$$

Predictions of patches in a radius of $d = 4$ pixels are averaged to obtain the final ensemble prediction. The detection task can also be learned with density estimation as auxiliary task [112, 117, 119, 120, 121, 122, 118]. In [112, 117, 119, 120, 121, 122, 118], a density or distance map is predicted that describes the location of every object that is nearest to a pixel, respectively. However, the methods mainly differ in neural network architecture and training procedure. In addition to changing neural network architecture and training procedure, in [122] densities at multiple scales are predicted, and the average of the integrated density maps with respect to the number of objects in the image are optimized. Methods like [113, 118] perform a prediction for every pixel of the current sliding-window position. Predictions of overlapping pixels of all sliding-window positions are summarized into a weighted average for robustness and improved performance.

2.3 Semantic Segmentation

Semantic segmentation is the prediction of a class label for each pixel of an image. Deep learning for semantic segmentation has been widely used in computer vision [90]. In this section, semantic segmentation using deep learning is introduced and specialized methods for microscopy images are presented.

2.3.1 Methods in General Computer Vision

Classification networks can be used for operating in a sliding window to predict the class of the windows center pixel [123, 124]. However, features for neighboring pixels that are already computed cannot be reused, which makes these methods computationally ineffective. In 2015, Long et al. proposed the *Fully Convolutional Network (FCN)* for semantic segmentation [26]. It performs transposed convolution for upsampling on feature maps at multiple scales and fuses them into the prediction. In the *Deconvolution Network* [125], upsampling is performed gradually with intermediate convolutions in the expanding path of the introduced network. This architecture family is also known as *hourglass-shaped neural networks*. In Figure 2.6, an archetype of the hourglass-shaped neural network architecture family is shown. In the *U-Net* [27, 126, 127], long-range skip connections between the contracting and expanding path were added. The features in the expanding path are concatenated with the respective feature map from the contracting path. U-Net is therefore capable

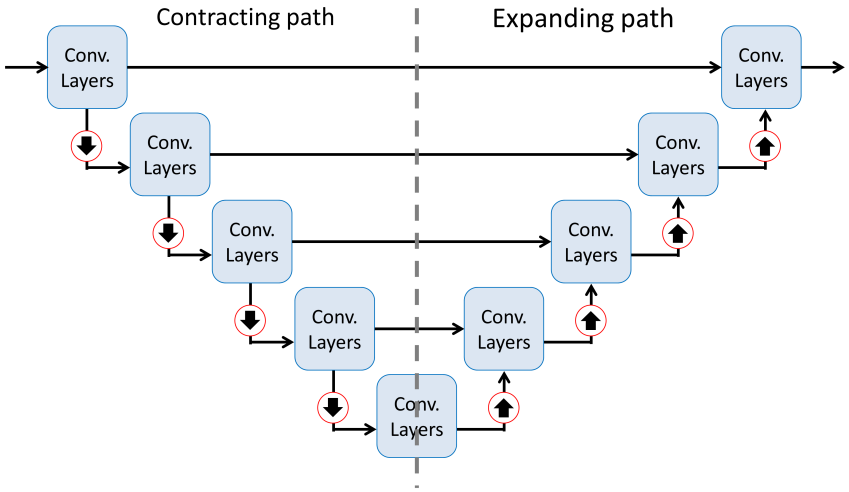


Figure 2.6: Generic hourglass-shaped neural network architecture. The original image is the input on the left and the output is generated on the right. Arrows pointing downwards denote pooling and arrows pointing upwards denote unpooling. The convolutional layer blocks perform feature extraction or fusion. The dotted line separates the contracting path and the expanding path of the network. More recently, architectures incorporate long skip connections between contracting and expanding path.

of retaining fine-grained details whilst incorporating context from a larger receptive field. *V-Net* is a 3D extension of U-Net[72], which uses additional short-range residual connections adapted from ResNet [29] in comparison to U-Net. Since algorithms are often evaluated using the Dice coefficient, they proposed to train the network with the Dice loss instead of the cross-entropy loss to improve performance. In [128], an additional convolution was added to the long-range skip connections. The authors also proposed to refactor the refinement step for a more efficient computation by replacing the concatenation of the feature maps and the subsequent convolution by a convolution on each set of feature maps by subsequent addition. In the *Attention U-Net* [129], an attention mechanism was added, which weighs the features from the skip connection with the features from the expanding path. Upsampling using transposed convolutions can lead to "checkerboard" artifacts [130] in the resulting feature maps, and therefore inaccurate predictions. This effect can be prevented by carefully choosing the stride factor or by using bilinear interpolation followed by a convolution. A more careful design of the refinement and throughout use of residual connections was presented in [131]. Jégou et al. showed in [132] that Densely Connected blocks can be incorporated instead of residual or plain convolutional layers to further boost performance. For more efficient use of parameters, in [133], the proposed capsules from [134] are used. Capsules perform expectation maximization (EM) to route the activations of the previous layer to the consecutive layer using corresponding predicted voting vectors. Therefore, they use agreement of multiple weak predictions similar to the idea of the Hough transform [135]. Milletari et al. [136, 137] proposed *Hough-CNN*, which predicts a vector to the centroid of the corresponding object for each pixel in the segmentation. Objects with high agreement are kept and the voting pixels form the segmentation mask. In contrast to applying individual functions at each scale in the expanding path, a convolutional RNN can be used to synthesize the segmentation at all scales [138]. The rational behind using an RNN is that the RNN is able to smooth single predictions and gradually combine them over all scales into a single prediction. To avoid the need for long-range skip connections and reducing and increasing the feature map resolution in the contracting and expanding path, dilated convolutions [88] can be used. Alternatively, *Segnet* [139] uses the pooling switches in the contracting path for unpooling in the expanding path, which is more memory-efficient than U-Net. The features from the long-range skip connections can also be aggregated at full resolution [140], which improves performance when training from scratch on small datasets. Methods like *ENet* [141] optimize inference speed by reducing the amount of parameters and computing steps using asymmetric convolutions. A convolution is factorized into two consecutive convolutions with dimensions $1 \times k$ and $k \times 1$, where k is the size of the convolutional kernel. These asymmetric convolutions were introduced in the Inception network in [94]. Moreover, Paszke et al. observed in [141] that when using PReLU activations, the first layers prefer a negative component, while consecutive layers prefer an activation close to zero. The proposed semantic segmentation networks lack the compliance with global

priors to avoid implausible segmentations. In [142], the global average pooled feature map is concatenated with all feature maps to add global context. The *Pyramid Scene Parsing Network* (PSPNet) [142] uses the output of an ASPP block [28] in a residual after the base network to incorporate global priors into the segmentation. In [143], the position of a pixel in the input image is added as feature map to enable the network to reason about spatial relationships. Classical methods like Conditional Random Fields (CRFs) [144] or Level Sets [145] were also reformulated as RNNs to incorporate global priors into the segmentation. They use iterative refinement of the feature maps to maintain consistency whilst improving the segmentation. In comparison to their non-deep learning counterparts, parts of the algorithm like the potential functions of the CRF are learned. To improve segmentation performance, detection can be performed prior to segmentation. In *Mask-CNN* [146], a Faster R-CNN is used for detection and the features of the detection head are passed to an expanding path of a Deconvolution Network to predict a segmentation for each detected object. In [73], a U-Net is used for coarse segmentation. Connected components are interpreted as detection and passed to a second U-Net which performs segmentation at higher resolution. The developers of the *DeepLab* neural networks evaluated several of the described methods and established state-of-the-art networks for semantic segmentation [147, 148, 28, 149]. In DeepLabV1 [147], they use a VGG-16 network to predict a coarse segmentation. To maintain spatial dimensions, they removed the pooling layers and compensated the loss in receptive field by adding appropriate dilation rates to the convolutional layers. The successor DeepLabV2 [28] uses a ResNet as base network. Moreover, an ASPP is introduced to incorporate global context, and a CRF is used to refine the predicted segmentation. In DeepLabV3 [148], several minor improvements on the layer configuration were performed and the use of CRFs was abandoned. The enhanced DeepLabV3+ [149] uses an hourglass-shaped architecture while maintaining the ASPP to capture context. Moreover, convolutions are replaced by more efficient depthwise separable convolutions from the Xception model [150]. Depthwise separable convolutions factorize the standard convolutional layer over k feature maps into a depthwise convolution and a consecutive pointwise convolution. The depthwise convolution applies a set of n convolutional kernels to each feature map independently, which results in kn feature maps. The pointwise convolution is a 1×1 convolution which combines all feature maps from the depthwise convolution. Note that factorized convolutional kernels can only represent a subset of possible convolutional kernels, since matrices $\mathbf{A}$ with $\text{rank}(\mathbf{A}) > 1$ are not always separable.

2.3.2 Methods for Microscopy Image Data

In microscopy image segmentation based on deep learning, several domain-specific adaptions and extensions were proposed (e.g., [123, 27, 151, 110, 152, 153]). U-Net [27] was developed for cell segmentation. In addition to the architecture, Ronneberger et al. proposed a weighing function for the cross-entropy loss, which enforces the

network to learn cell separation. Morphological operations are used to determine the distance of each pixel to the object border. The distance to the next two objects is used to reciprocally weigh the cross-entropy loss. Therefore, pixels that are close to two objects are weighted higher than pixels with more distant object borders. Drozdzal et al. [151] evaluated the effect of residual connections for microscopy images and showed that both long-range and short-range residuals are important to train very deep hourglass-shaped architectures. They also showed that the Dice loss is beneficial to successfully segment cell borders due to the intrinsic class balancing. In [154, 110], Akram et al. use a cascaded Faster R-CNN to extract object proposals and a U-Net to perform segmentation, which improves cell separation. For small datasets, Arbelle et al. [155] showed that adversarial training is beneficial for training with limited amount of annotated data. In [156], a third class for cell borders was added to the classes' foreground and background to use it for cell separation.

3 Detection in Microscopy Images

3.1 Overview and Task Description

Detection of prominent structures such as particles and cells in microscopy images is a frequent and important task in quantitative microscopy. The results are used to perform downstream tasks like nuclei density analysis or particle motion analysis.

Although many different types of methods for detection exist, in recent years deep learning methods dominate the field of computer vision. Deep learning has been successfully used for particle and mitosis detection (e.g., [114, 111, 157]). However, domain-specific challenges in object detection in microscopy images arise. Usually, many clustered objects of the same class have to be detected in very large images. Therefore, splitting of close objects and robust fusion of information from multiple image scales is important. To distinguish positive and negative samples which look very similar, specialized training procedures are needed. Moreover, to cope with the size of the images, the algorithms have to be fast and memory-efficient.

In this chapter, novel methods for detection of objects in microscopy images are presented. A domain-adapted architecture for particle detection is proposed. In addition, a novel neural network utilizing the benefits of the Hough transform for mitotic cell detection is presented. Moreover, a novel network with a differentiable consensus voting layer for object detection in microscopy images is proposed. The methods were published in Wollmann et al. [16, 10, 2].

3.2 DetNet: Deep Neural Network for Particle Detection in Fluorescence Microscopy Images

To gain insight on cellular processes, particle detection in fluorescence microscopy images is an important task and a prerequisite for particle tracking. Main challenges for particle detection are the small size of fluorescently labeled particles, low signal-to-noise ratio (SNR), and lack of prominent shape and appearance characteristics. Due to the large number of particles, manual detection is not feasible for many applications. In previous pork, different approaches for particle detection in fluorescence microscopy images were introduced (e.g., [158, 159, 160]) such as the spot-enhancing filter (SEF) [161], a H-Dome transform-based detector (H-Dome) [159], or adaptive thresholding with autoselected scale (ATLAS) [162]. SEF is often used for particle detection. In combination with probabilistic tracking methods, state-of-the-art results are obtained

[163, 40]. However, SEF assumes a relatively simple appearance model of particles, namely a Gaussian function. Another disadvantage of such classical methods is that several parameters need to be tuned. Recently, convolutional neural networks (CNNs) were used for particle detection (e.g., [114, 116]). However, these methods are based on a sliding window scheme or involve a relatively large number of parameters. In this section, a novel deep learning method for particle detection in fluorescence microscopy images using an hourglass-shaped deep neural network denoted as DetNet (Wollmann et al. [10]) is presented. The network is a domain-adapted Deconvolution Network and can cope with different particle shapes. DetNet is slim, fast, and can be trained with only a few ground truth annotations. In contrast to the CNN in [116], the method does not require a sliding window scheme, and all particles within an image are detected at once by sharing full-image convolutional features. Compared to the U-Net based approach in [114], the method has significantly less parameters and the network structure differs (e.g., a Deconvolution Network [125] is used instead of a U-Net as well as bilinear upsampling instead of transposed convolution). In addition, it is suggested to use a Dice loss and optimize the parameter of the sigmoid activation function to improve the performance.

The proposed deep neural network DetNet for particle detection is based on a Deconvolution Network [125] which has been adapted to the application domain. A Deconvolution Network is composed of a contracting (pooling) and an expanding (unpooling) path. The network handles objects at multiple scales naturally by the hourglass-shape of the network. A challenge in the application is that the objects (particles) are relatively small and lack complex shape information. Thus, data augmentation does not significantly increase the variability of the training data. In a setting with a limited number of training samples, overfitting is likely to occur. To enable accurate particle detection and efficient training, multiple adaptations to the Deconvolution Network architecture [125] are introduced. In particular, the number of parameters is significantly decreased by reducing the number of extracted feature maps. In addition, the size of the receptive field is reduced by employing pooling only two times instead of five times. Long range skip connections are not used as in [27], since in the application, detailed boundary information is not relevant, which further reduces the number of parameters. Moreover, the convolutional layers are replaced with residual blocks [29] and instance normalization [65] is used. This type of normalization is used because the batch normalization in the original formulation [29] needs a representative dataset to train moving averages, which is hardly available when using only few training samples. In the network residual, blocks are used. By using residual blocks, the problem of gradient vanishing is reduced, which improves the efficiency of training deep architectures. Instead of using transposed convolutions, bilinear upsampling is employed in the expanding path, which further reduces the number of parameters and avoids checkerboard artifacts. ReLU activations are used and the weights are initialized using HE initialization [59].

In total, the parameters were reduced to 17 k compared to a standard Deconvolution

Network with 1.1 M parameters, a U-Net with 1.9 M parameters, and the CNN in [114] with 400 k. The DetNet architecture is outlined in Figure 3.1. The network is trained using a Dice loss and early stopping with the AMSGrad optimizer [56] and a learning rate of $l_{\text{init}} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The Dice loss is a soft formulation of the standard Sørensen-Dice coefficient for the ground truth and prediction, which performs implicit class balancing and penalizes easy samples compared to the Cross-Entropy loss. It was found that the stability of the training improves by calculating the Dice loss over all N pixels in a batch instead of averaging the Dice loss over the single images. In contrast, when using a standard Cross-Entropy loss, training was not successful due to the heavy class imbalance. The training data is augmented using random flipping, rotation, and cropping.

In preliminary experiments, high precision and low recall for low SNR scenarios was observed. However, the aim was to balance precision and recall (as represented by the F1 score), and it was found that the F1 score can be significantly improved by optimizing the shift $a \in \mathbb{R}$ of the sigmoid function of the neural network

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-(x-a)}} \tag{3.1}$$

To optimize a, the *HyperHyper* hyperparameter optimization framework presented in Chapter 5 is used together with the other hyperparameters of the optimizer.

The detected particles can be used for subsequent tasks like particle tracking. Tracking particles in time-lapse microscopy image sequences is important to quantify dynamic behavior. Traditional tracking methods (e.g., [164, 165, 166, 167, 40, 168, 169]) use a handcrafted similarity measure to link particles between time steps. Methods based on deep learning have the potential to learn the similarity measure from data and improve performance. Previous methods using deep learning (e.g.,

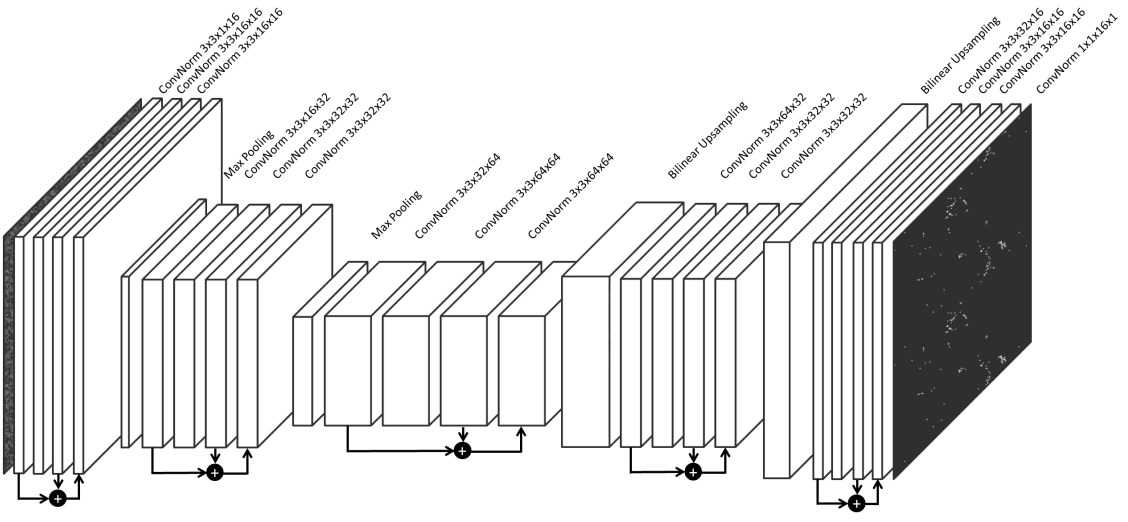


Figure 3.1: Deep neural network architecture of DetNet. The specific layer configuration is given above each layer.

[170, 171, 172]) are based on appearance features (e.g., in pedestrian tracking, cell tracking). However, particles appearance is less useful for linking than motion.

The Deep Particle Tracker (DPT) for particle tracking in time-lapse fluorescence microscopy images based on an RNN proposed by Spilger, Wollmann, et al. in [14] learns to determine assignment probabilities for correspondence finding, without requiring a handcrafted similarity measure. The network architecture of DPT is outlined in Figure 3.2. For each time step t and object i, the feature vector $\mathbf{x}_t^i = (x_t^i, y_t^i, s_t^i, \alpha_t^i)$ describes the objects position (x, y) and the speed and direction denoted by s and α (computed using the positions at two successive time points). For each object i, a fully-connected layer is used to compute an embedding $\mathbf{z}_t$ from $\mathbf{x}_{t-1}$. A sequence to sequence LSTM is used to calculate a sequence of hidden states $\mathbf{h}_t$ from the sequence of $\mathbf{z}_t$ to capture the motion of the object. For each time step, a fully-connected layer is used to compute an embedding $\mathbf{q}_t$ from the nearest neighbors of $\mathbf{x}_{t-1}$ in time step t. $\mathbf{q}_t$ and $\mathbf{h}_t$ are concatenated and passed into a the final fully-connected layer followed by SoftMax to predict the assignment probabilities $\mathbf{a} \in [0, 1]^{M+1}$ between the i-th object and the nearest detections M as well as the probability for a missing detection. For better training, the feature vectors $\hat{\mathbf{x}}_t^i$ are regressed by applying another fully-connected layer to $\mathbf{h}_t$. Gaussian dropout is applied after each layer. The computed assignment probabilities and the probabilities for missing detections (dummy detections in the probability matrix) are passed to the Hungarian algorithm for determining one-to-one correspondences. The network is trained using ground truth assignment probabilities (cross-entropy loss) and the cross-entropy loss and auxillary regression loss using the ground truth locations (mean squared error). Ground truth trajectories for microscopy image sequences of biological particles is hardly available and manual annotation is cumbersome. Therefore, synthetic data is used for training. Simulated trajectories of particles perform Brownian motion or directed motion.

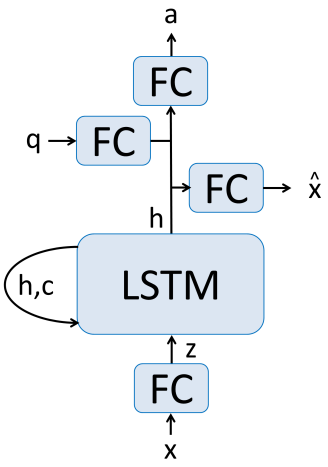


Figure 3.2: Deep neural network architecture of DPT.

3.3 Deep Residual Hough Voting for Mitotic Cell Detection in Histopathology Images

In the field of tissue microscopy, the detection and quantification of prominent cellular structures is a central task. Main challenges are data heterogeneity, image noise, and lack of training data. In addition, histology images typically have a very large size, a high density of image structures, and imbalanced object class occurrence. For the task of cell detection in histology images, several approaches exist (e.g., [111, 157, 173, 174]). Several advances of the architecture of ResNets have been made recently (e.g., [175, 141, 176]). On the other hand, the Hough transform is a robust method for object detection [177]. Due to the voting process in the Hough transform, single noisy votes hardly influence the result. It has been shown that CNNs can learn an implicit shape and texture representation [136, 113]. Therefore, CNNs are capable to predict the relative location of an object within an image patch, which can be exploited for Hough voting. However, existing approaches that combine CNNs and Hough voting (e.g., [136]) require discretization of the voting space, which leads to a rapidly growing voting space when using a larger number of bins or a larger voting region. Also, these approaches require prediction of an additional confidence score for each vote, which is an additional task for the network. Some methods (e.g., [111]) are based on an ensemble of networks which increases the computation time. In this section, a novel approach for mitotic cell detection in heterogeneous histopathology images is proposed, which combines a deep residual network with Hough voting. Also, a novel loss function is introduced, which exploits polar coordinates and is invariant to the magnitude of the voting error. The method was published in Wollmann et al. [16].

The method conducts fast feature extraction, multi-scale factor disentangling, and Hough voting in a deep neural network. The network has a new two branch architecture and is trained with a novel loss function. Training of the method is performed using cell centroids (e.g., by selecting bounding boxes) and the original image data. The architecture of the network is presented in Table 3.1. It consists of (i) a downsampling part, (ii) a factor disentangling part, and (iii) a pixel-wise classification part with two branches. The results can be upsampled if the exact positions of the detections are needed.

Fast downsampling is performed by the downsampling blocks, which contain a linear branch (no activation function) and a non-linear branch (non-linear activation function). The linear branch in the downsampling block performs average pooling and a 1x1 convolution to increase the feature maps. Within the non-linear branch a strided 3x3 convolution is used followed by a 1x1 convolution. Except for the first layer, the output of both branches are summed up and represent a residual block (res. block). All non-linear branches in the residual blocks of the factor disentangling part contain a 3x3 dilated convolution (DilConv) [88] to increase the receptive field, which is followed by a 1x1 convolution for mapping to the target feature space.

Table 3.1: Deep Residual Hough Voting architecture

Layer type		Output size
input		3 x N_x x N_y
downsampling block (non res.)		32 x $N_x/2$ x $N_y/2$
downsampling block		64 x $N_x/4$ x $N_y/4$
downsampling block		64 x $N_x/8$ x $N_y/8$
res. block		64 x $N_x/8$ x $N_y/8$
res. block		64 x $N_x/8$ x $N_y/8$
res. block		64 x $N_x/8$ x $N_y/8$
res. block (dilated 2)		64 x $N_x/8$ x $N_y/8$
res. block (dilated 2)		64 x $N_x/8$ x $N_y/8$
res. block (dilated 2)		64 x $N_x/8$ x $N_y/8$
res. block (dilated 4)		64 x $N_x/8$ x $N_y/8$
res. block (dilated 4)		64 x $N_x/8$ x $N_y/8$
res. block (dilated 4)		64 x $N_x/8$ x $N_y/8$
fconv. res. block	fconv. res. block	64 x $N_x/8$ x $N_y/8$
Dropout	Dropout	64 x $N_x/8$ x $N_y/8$
conv. (+sigmoid)	conv. (+sigmoid)	1 x $N_x/8$ x $N_y/8$
$\boldsymbol{\varphi}$	$\mathbf{r}$	1 x $N_x/8$ x $N_y/8$
voting		1 x $N_x/8$ x $N_y/8$
bilinear upsampling		1 x N_x x N_y

Using a 3x3 convolution instead of a 1x1 convolution resulted in smoother, but less accurate predictions. The linear branch does not perform a transformation on the input. Several dilated residual blocks with different dilations are used, which ensures a computational effective multi-scale feature aggregation. By employing dilated convolutions, skip connections and deconvolutions are avoided. Dilations larger than four pixels had no positive impact on the results. For all non-linear layers, the rectified linear unit (ReLU) is used as activation function. Reduction of covariate shift within the network is performed by batch normalization layers after each convolutional layer with a non-linear function. The Dropout probability [62] of data points was set to $p = 0.5$. Best results were achieved using Dropout of data points along all spatial and feature dimensions, compared to Dropout only along the spatial or feature dimensions, and any kind of Dropout within the residual blocks. Regularization of the weights using an ℓ_2-norm did not improve the performance. Weights in layers with ReLU activations are initialized utilizing HE initialization [59]. The other weights are initialized with Xavier initialization [178]. A fully convolutional residual block performs in their non-linear branch twice a 1x1 convolution, which is equivalent to applying a fully connected layer to each pixel. The resulting features are spatially weighted by a 3x3 convolution. Finally, a sigmoid function is used to scale the output to the range of polar coordinates, used for the Hough transform. Separate branches for radius and angle prediction is employed. Hence, features are shared between these branches, but the final non-linearities disentangle radius and

angle. A benefit of using two branches is that a factor of two more neurons in the network can be exploited. The results of the two branches are combined again in the voting layer. The voting layer can be applied at a lower resolution since cells cover a significant number of pixels there. This step further reduces the computation time. Afterwards, the result is upsampled with bilinear interpolation to match the original resolution.

In the proposed method, a formulation of the Hough transform in polar coordinates $(r,\, \varphi)$ is used:

$$v(i,j) = \int_{r=0}^{r_t} \int_{\varphi=0}^{2\pi} \delta(\begin{pmatrix} i \\ j \end{pmatrix} - \mathbf{g}(i + r \cdot \cos(\varphi), \\ j + r \cdot \sin(\varphi))) \; d\varphi \; dr \tag{3.2a}$$

$$\mathbf{g}(i,j) = \begin{pmatrix} i \\ j \end{pmatrix} + r_{ij} \cdot \begin{pmatrix} cos(\varphi_{ij}) \\ sin(\varphi_{ij}) \end{pmatrix} \tag{3.2b}$$

where $v \in \mathbb{R}$ is the voting function, $\mathbf{g} \in \mathbb{R}^2$ is the polar to Cartesian transform, δ the Dirac delta function, and $r_{ij} \in \mathbb{R}$ and $\varphi_{ij} \in \mathbb{R}$ are the predicted relative vote coordinates for each pixel with the indices i and j. The voting function can be efficiently implemented by separating $\mathbf{g}$ from the pixel-wise vote collection and masking with the predicted radius $\mathbf{r} \in \mathbb{R}^2$:

$$v(i,j) = \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \delta(\begin{pmatrix} i \\ j \end{pmatrix} - \mathbf{g}(x,y) \cdot \mathrm{H}(R_t - r_{xy})) \tag{3.3}$$

where $\mathrm{H} \in \{0,1\}$ denotes the Heaviside step function and r_{xy} is the radius with indices in Euclidean space. By using H, it is ensured that votes are collected only within a region of radius R_t in an image of dimensions $N_x \times N_y$.

The optimization of the weights is formulated as a two-task regression problem with loss function $\mathcal{L}$:

$$\mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} (\frac{1}{k} \|\Delta \mathbf{r}^i\|_2^2 - \frac{\lambda}{k^2} \left(\mathbf{vec}(\Delta \mathbf{r}^i) \cdot \mathbf{1_k}\right)^2 \\ + \frac{1}{k_\varphi^i} \|\Delta \boldsymbol{\varphi}^i\|_2^2 - \frac{\lambda}{{k_\varphi^i}^2} \left(\mathbf{vec}(\Delta \boldsymbol{\varphi}^i) \cdot \mathbf{1}_k\right)^2) \tag{3.4a}$$

$$k = N_x N_y \tag{3.4b}$$

$$k_\varphi^i = \epsilon + \mathbf{vec}(\mathbf{H}(r_t \cdot \mathbf{1}_{k \times k} - \mathbf{r}_{gt}^i)) \cdot \mathbf{1}_k \tag{3.4c}$$

$$\Delta \mathbf{r}^i = \mathbf{r}^i - \mathbf{r}^i_{gt} \tag{3.4d}$$

$$\Delta \boldsymbol{\varphi}^i = (\boldsymbol{\varphi}^i - \boldsymbol{\varphi}^i_{gt}) \odot \mathbf{H}(r_t \cdot \mathbf{1}_{k \times k} - \mathbf{r}^i_{gt}) \tag{3.4e}$$

where M is the number of samples in a mini-batch, k the number of pixels in input i, $\mathbf{1}_k$ a vector of length k of ones, $\mathbf{1}_{k \times k}$ a $k \times k$ matrix of ones, **vec** denotes the vectorization of a matrix, and λ is a hyperparameter. Since not all pixels contribute to the angle related loss, the normalization factor k^i_φ is introduced. To prevent a division by zero, a constant $\epsilon = 10^{-8}$ is used. The deviation matrices $\Delta \mathbf{r}^i$ and $\Delta \boldsymbol{\varphi}^i$ in (3.4d), (3.4e) of dimension $N_x \times N_y$ of the predictions for the radius $\mathbf{r}^i$ and angle $\boldsymbol{\varphi}^i$ to the ground truth $\mathbf{r}^i_{gt}$ and $\boldsymbol{\varphi}^i_{gt}$ are optimized using the scale-invariant mean squared error loss function in (3.4a). The loss function is a modification of the one in [179], which is invariant with respect to the global scale of the feature maps. To apply the binary mask represented by $\mathbf{H}$ for $\Delta \boldsymbol{\varphi}^i$, the Hadamard product ($\odot$) is used. For better convergence, the radius $\mathbf{r}$ is normalized to the unit circle. The model is trained using the Adam optimizer [55] with an initial learning rate $l_{init} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

3.4 Grading of Whole-Slide Images based on Mitotic Cell Counts

Breast cancer is the most common type of cancer and the primary cause for cancer deaths of women worldwide [180]. The progression of the disease is quantified by pathologists using whole-slide images (WSIs) of tumors and lymph nodes, which are stained with hematoxylin and eosin (H&E). Tumor growth is an important indicator for determining the prognosis of breast cancer patients. A higher proliferation rate is generally related to a worse prognosis due to increased probability for cancer relapse. Therefore, the quantification of the proliferation rate is an important biomarker to determine a suitable therapy. Currently, pathologists manually count mitotic cells in hematoxylin and eosin (H&E) stained histological slide preparations. Automation of this process is important and involves the detection of mitotic cells. Several approaches for cell detection in tissue microscopy images exist (e.g., [111, 173, 174]). Especially for mitosis detection several challenges have been conducted to compare available methods using image sections [181, 182, 157], but complete whole-slide images (WSI) were not used. In this section, a new approach is described, which combines a threshold-based attention mechanism with a deep neural network (DNN) for mitotic cell detection and grading of WSIs. The method was published in Wollmann et al. [17, 5]. Compared to previous approaches, the proposed method conducts fast feature extraction, multi-scale factor disentangling, and voting in a DNN. Training of the mitotic cell detection method uses only ground truth centroids and corresponding original images. Detection of mitotic cells is performed on

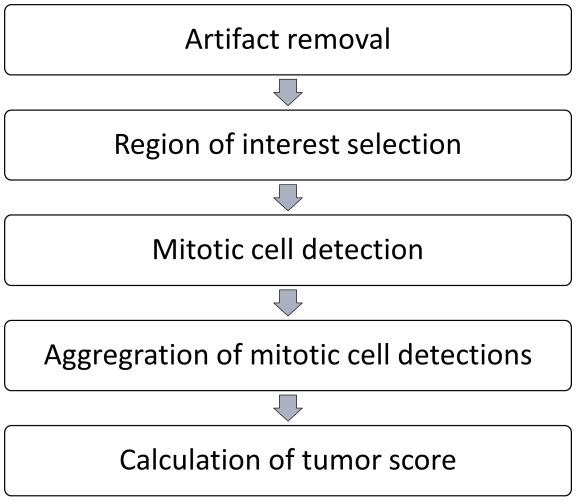


Figure 3.3: Workflow for grading breast cancer WSIs based on mitotic cell counts.

automatically determined regions of interest (ROIs) of WSIs. The detections are summarized over a selection of subregions within the ROI and classified into a proliferation score by a shallow decision tree. An additional dataset with annotated tumor grades is used to train the decision tree.

The proposed method uses a threshold-based attention mechanism to select a ROI with subregions of size $2\,mm^2$ from a whole-slide image (WSI). A machine learning method is used to predict the centroids of mitotic cells within each subregion. Using a shallow decision tree a tumour classification is obtained for the WSI. An overview of the workflow is shown in Figure 3.3.

ROI Selection

For selecting an appropriate region of interest (ROI), a multi-scale preprocessing approach is used. Preprocessing is performed on the highest image scale to remove artifacts like ink or non-flat tissue on the slide. In the first step, for removing ink artifacts, thresholding is applied to a ratio image (intensities of the red channel divided by the intensities of the green channel), which yields a mask. Within this mask the intensities are set to the maximum intensity in each of the three color channels of a WSI. Since the maximum intensity corresponds to background, the masked pixels are not considered in the subsequent analysis (Figure 3.4).

The second step of the preprocessing removes black structures. To this end, the intensities of the red and blue channel are added, and each pixel with an intensity below the 1 % percentile of the intensity histogram is replaced by the maximum intensity of the original image. This step removes black structures like flipped tissue. Afterwards the image is thresholded based on the mean intensity of the red and blue channel. For determining the threshold, the 2 % percentile of the histogram is used. The resulting image is smoothed with a Gaussian filter and downscaled so that $\sim$1.5 pixels represent $2\,mm^2$ (high power field) (Figure 3.5). For this lower scale image,

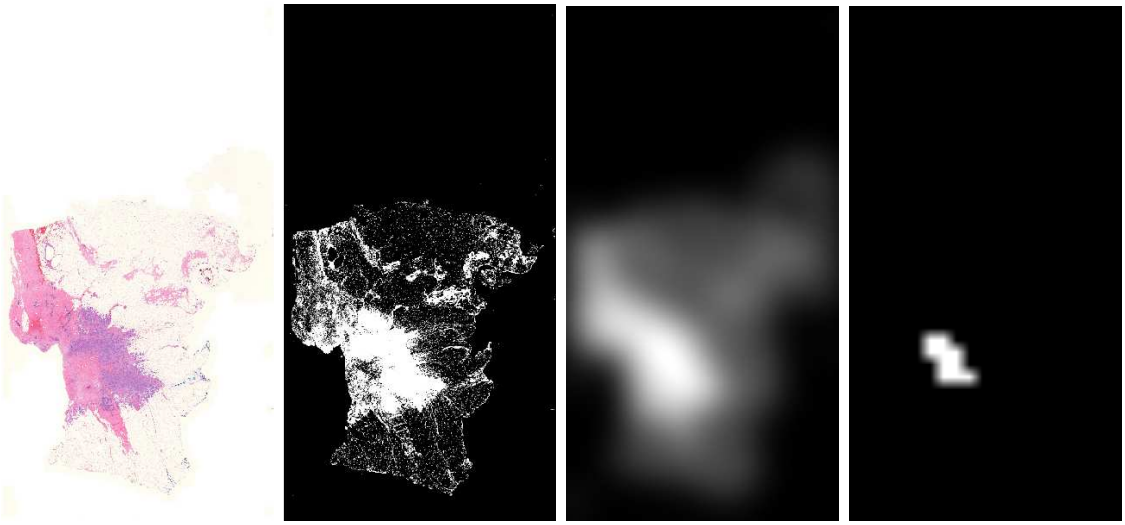
(a) Original WSI (b) Detected artifacts (white)

Figure 3.4: Example image demonstrating the artefact detection mechanism.

ten pixels with the highest intensities are selected and the corresponding $2\,mm^2$ subregions are extracted from the original high-resolution image. With this scheme the subregions have an overlap of up to 50 %. Finally, the subregions are rescaled to the pixel size the mitotic cell detector was trained on.

Mitotic Cell Detection and Counting

Mitotic cell detection is conducted using deep residual Hough voting presented in Section 3.3. The network consists of a downsampling section, a factor disentangling section, and a pixelwise classification section with two branches. Fast feature



(a) Original WSI (b) Segmented tissue (white) (c) Attention map for WSI (white) (d) Selected ROI within WSI (white)

Figure 3.5: Example demonstrating the attention mechanism.

extraction is achieved by strided convolutions. In the proposed method, a formulation of the Hough transform in polar coordinates is used, which includes a voting procedure that exploits the predicted relative vote coordinates for each pixel. The optimization is formulated as a two task regression problem. The model is trained using the Adam optimizer [55]. Deep residual Hough voting is presented in more detail in Section 3.3.

Mitotic Cell Count Thresholding

To determine the final score for the whole slide the 95 % percentile of the cell counts of the analysed subregions is calculated. This feature is used within a shallow decision tree with two thresholds to obtain a score between 1 and 3. The lower threshold turned out to be 6 and the upper was 10. These thresholds match the guideline for grading tumors of [183], which gives a score of 1 for a cell count below 6 cells per high power field and a score of 3 for a cell count above 10 cells per high power field. The selection of the 95 % percentile was determined by performing a grid search over different percentiles 70 % - 95 %, mean, median, and maximum of the detected mitotic cell counts per image using the ground truth from the training dataset. Corresponding thresholds were determined by a grid search between 0 and 30.

3.5 Deep Consensus Network for Particle and Cell Detection

To determine object counts or object density, usually a characteristic location such as the object's centroid is used (e.g., [111, 112, 113, 114, 115, 116, 117, 118]). Typically, classification networks (e.g., [92, 29]) are employed, that use a sliding window scheme to predict the presence of an object of interest within the window [111, 113, 114, 115, 116]. In contrast, centroid-based methods based on hourglass networks (e.g., [16, 120, 121, 122, 10]) do not rely on a sliding window scheme and are faster since multiple objects are detected at once by sharing full-image convolutional features computed in a single forward pass. In Section 3.2 a method for particle detection and in Section 3.3 a method for mitotic cell detection, which do not rely on a sliding window scheme, are presented. A more general method for object detection in microscopy images could be used for both particle detection and cell detection.

For images of natural scenes, the development of detection methods based on predicting the bounding box of objects was strongly driven by the PASCAL VOC [184] and MS COCO [33] challenges. In recent years, the leading methods were two-stage or one-stage detectors based on convolutional neural networks. Two-stage detectors like Faster-RCNN [23] use a region proposal network (RPN) on top of a classification network (backbone network) to propose detection hypotheses. Non-Maximum Suppression (NMS) is employed to filter the hypotheses, and a second

network is used on the cropped image patches to refine the prediction. Multiple bounding box priors called "anchors" enable the network to focus on confidence score prediction instead of bounding box regression, which showed improved performance [23]. The variations of the You Only Look Once (YOLO) network [24, 103, 104] that exploit a spatial grid can be trained faster, since they are one-stage schemes and do not require performing NMS during training. One-stage networks use a coarse spatial grid, where only one object hypothesis is obtained per bin. Gradients are backpropagated only to bins where an object is present in the ground truth. The Single Shot Detector (SSD) [105] network applies an RPN to multiple scales of their backbone network to extract object hypotheses. In the Feature Pyramid Network (FPN) [106] a contracting and expanding network path are combined with skip connections to retain fine spatial details. An RPN is applied to each scale to extract object hypotheses. RetinaNet [25] was the first one-stage network that outperformed two-stage detectors by using their proposed Focal loss. The Focal loss tackles the heavy class imbalance in object detection where most of the object hypotheses are negatives. Due to sparse gradients and heavy class imbalance, the training of detection networks from scratch is difficult. Therefore, detection networks are typically pre-trained on an auxiliary task like classification [23] or segmentation [185]. However, compared to object detection using bounding boxes much less work exists on *centroid-based object detection*. Since centroid-based object detection has similarities to object detection using bounding boxes, while the enclosing region of the object is not predicted, advances for object detection using bounding boxes (e.g., FPN, anchors, NMS) could be transformed and exploited to improve centroid-based object detection.

In this section, a novel deep neural network for centroid-based object detection is introduced, which relies on a consensus of object detection hypotheses and is denoted as Deep Consensus Network (ConsensusNet). Compared to previous approaches, object hypotheses for multiple centroid anchors are determined at multiple image scales using a novel Centroid Proposal Network (CPN). The hypotheses are aggregated in an differentiable voting space and a consensus is formed. A novel anchor regularization scheme is introduced, to increase the robustness of training. To retain fine spatial details, a modified FPN is used as backbone network. The method can be trained end-to-end without pre-training. During inference, a centroid-based NMS is used to remove conflicting hypotheses. An algorithmic improvement of NMS combined with consensus voting that requires much less computing time than a vanilla NMS is proposed. In addition, novel loss function is proposed, which is derived based on insights on existing loss functions. The novel loss function is based on Normalized Mutual Information (NMI) and emphasizes class balance and correlation. The method has been submitted for publication [2].

Overview of the Network Architecture

An overview of the proposed deep neural network architecture is given in Figure 3.6. The proposed Deep Consensus Network has a similar architecture as RetinaNet [25] and uses a Feature Pyramid Network (FPN) [106] to extract features at multiples scales from a ResNet-50 [29] backbone with the squeeze-and-excite mechanism [186] for the residuals. To enable detecting very small objects, a high-resolution variant of the FPN is used. For the high-resolution Deep Consensus Network, an additional upsampling layer to restore the same spatial resolution as the input is included. In addition, to retain spatial details, the first two max pooling layers of the network are removed and the filter size in the first layer was set to 3×3 pixel. Group normalization [187] and Leaky ReLU activation functions [58] are used. The extracted feature maps at each scale are forwarded to the proposed Centroid Proposal Network (CPN). In comparison to RetinaNet, the CPNs at each scale do not share weights to capture different representations at each scale since all images within the considered microscopy datasets have the same magnification. In contrast to an RPN which predicts bounding boxes, the CPN predicts a set of centroids $\mathbf{v}'$ with corresponding confidence scores $P(\mathbf{v}')$. Similar to the RPN in RetinaNet, anchors as priors, but with a different configuration, are used. Figure 3.7 shows the anchor configuration employed in the proposed network. For each spatial position, $N_a = 17$ anchors in total including one anchor without offset, eight anchors with length 0.5 pixels and eight anchors with length 1 pixel, are used. The offset vectors are rotated so that they well cover a unit disk. The i-th anchor $\mathbf{a}_i \in \mathbb{R}^2$ is applied to the i-th

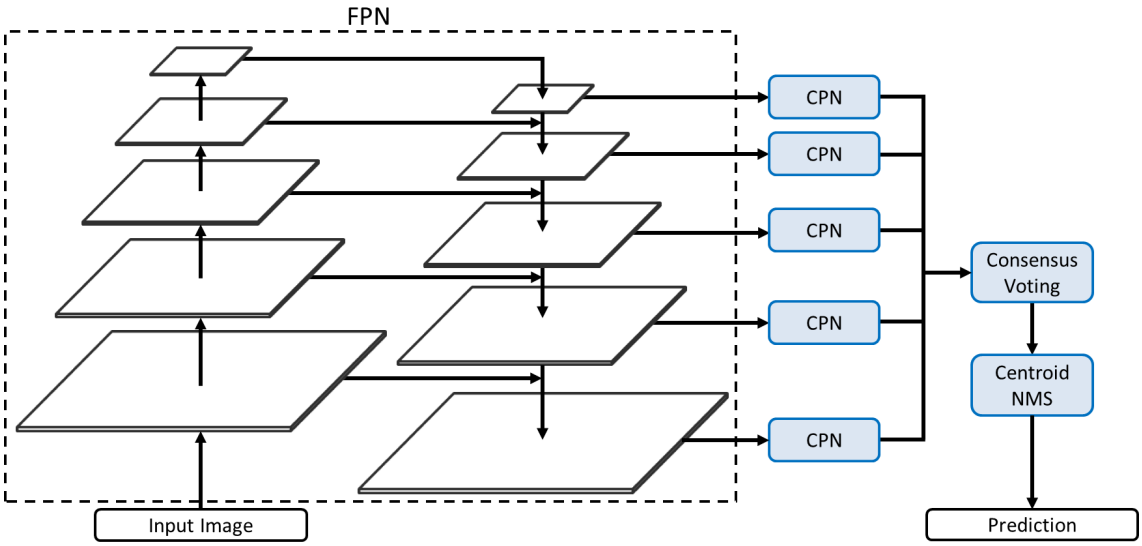


Figure 3.6: Deep Consensus Network architecture. A FPN is used for multi-scale feature extraction, CPNs for predicting object centroids, consensus voting for aggregation of predictions, and centroid-based NMS for eliminating conflicting proposals.

predicted offset vector $\mathbf{v}'_i \in \mathbb{R}^2$ yielding:

$$\mathbf{v}_i = s\ (\mathbf{v}'_i + \mathbf{a}_i) + \text{pos}(\mathbf{v}'_i)\ , \tag{3.5}$$

where $\text{pos}(\mathbf{v}'_i)$ is the spatial position of the offset vector and s denotes a scaling factor to normalize the distribution of $\mathbf{v}'_i$. To encourage the network to favour anchors pointing close to the object, the confidence scores $P(\mathbf{v}'_i)$ of large regression values for the magnitude of $\mathbf{v}'_i$ are penalized (regularized) to compute the confidence score of $\mathbf{v}_i$:

$$P(\mathbf{v}_i) = P(\mathbf{v}'_i)\ e^{-\ln(2)\left\|\mathbf{v}'_i\right\|} \tag{3.6}$$

For numerical reasons, the CPN predicts the logit (logarithm of the odds) $x'_i = \text{Logit}\left(P(\mathbf{v}'_i)\right) \in \mathbb{R}$ instead of directly predicting the confidence score $P(\mathbf{v}'_i) \in \mathbb{R}^{[0,1]}$. Thus, the regularization in (3.6) is performed in logit space yielding the logit:

$$\begin{aligned} x_i &= \text{Logit}\left(P(\mathbf{v}_i)\right) = \text{Logit}\left(P(\mathbf{v}'_i)\ e^{-\ln(2)\left\|\mathbf{v}'_i\right\|}\right) \\ &= \text{Logit}\left(P(\mathbf{v}'_i)\right) + \text{Logit}\left(e^{-\ln(2)\left\|\mathbf{v}'_i\right\|}\right) \\ &= x'_i + \ln\left(e^{-\ln(2)\left\|\mathbf{v}'_i\right\|}\right) - \ln\left(1 - e^{-\ln(2)\left\|\mathbf{v}'_i\right\|}\right) \\ &= x'_i - \ln(2)\left\|\mathbf{v}'_i\right\| - \ln\left(1 - e^{-\ln(2)\left\|\mathbf{v}'_i\right\|}\right) \end{aligned} \tag{3.7}$$

The feature extractor of the proposed CPN is different from the RPN in RetinaNet by concatenating the regressed N_o offsets $\mathbf{v}' \in \mathbb{R}^{N_o \times 2}$ with the features extracted from the FPN output $\mathbf{x}_{\text{in}} \in \mathbb{R}^{N_o \times p}$, with p feature maps, before predicting the logit vector $\mathbf{x}' \in \mathbb{R}^{N_o}$ of the confidence scores $P(\mathbf{v}')$ (see Figure 3.8). This has the advantage that the confidence score branch of the proposed network can condition confidence scoring on the regression results.

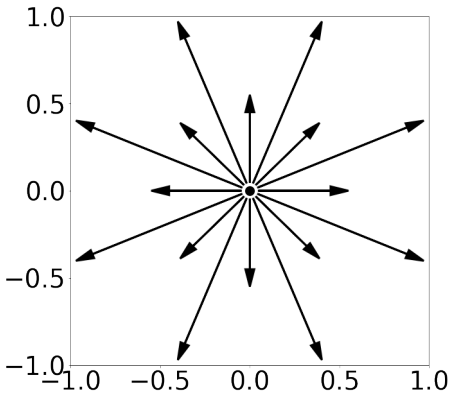


Figure 3.7: Anchors used as voting priors.

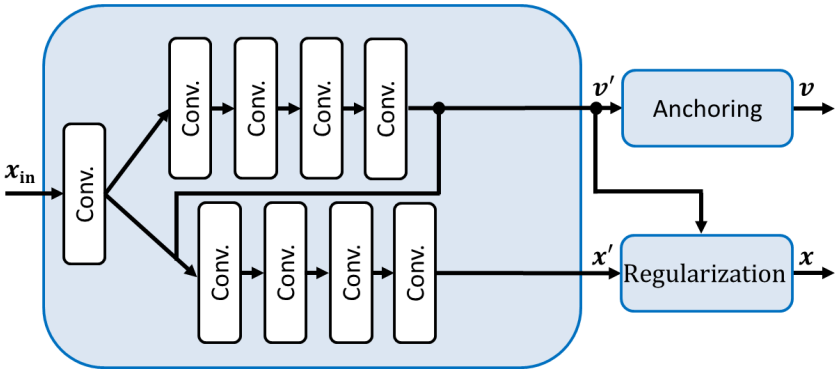


Figure 3.8: Centroid Proposal Network (CPN) and subsequent anchor and regularization steps. The offset vectors $\mathbf{v}'$ predicted by the CPN are combined with their corresponding anchors and the predicted confidence scores $P(\mathbf{v}')$ are regularized using the magnitude of the offset vectors $\mathbf{v}'$.

Deep Multi-Scale Consensus Voting

Previous deep learning methods for object detection (e.g., [23, 24, 105, 106, 25]) concatenate all proposals and use Non-Maximum Suppression (NMS) to remove conflicting proposals. Therefore, all information except the highest scoring proposals are discarded. Methods like [185] improve the results by ensembling conflicting proposals of multiple models. In the proposed approach, an extension of the voting-based approach in [16] for ensembling arbitrary many proposals from multiple voting anchors and image scales and is thus denoted as Deep Multi-Scale Consensus Voting is used. Compared to the previous approach the proposed novel approach is more general and trainable end-to-end, since the proposed voting scheme is differentiable. After consensus voting, NMS is performed on the few ensembled proposals to eliminate semantically conflicting detections.

By exploiting the spatial structure of an image, the number of proposals can be greatly reduced prior to NMS. An adaptive voting space with spatial bin size so that two neighboring detections within a minimum distance d do not fall in the same bin is leveraged. Therefore, increasing the minimum distance between objects decreases the number of bins of the voting space. Multiple votes and corresponding confidence scores calculated by the previous steps in (3.5) and (3.7) can fall into the same bin. Consensus Voting is used to ensemble the votes for each bin. Assume that the CPNs yield in total N votes over all image scales and anchors in a single bin of the voting space, where the i-th vote corresponds to the vector $\mathbf{v}_i \in \mathbb{R}^2$ and confidence score $P(\mathbf{v}_i) \in \mathbb{R}^{[0,1]}$:

$$P(\mathbf{v}_i) = \text{Sigmoid}(x_i) = \frac{e^{x_i}}{e^{x_i} + 1} \tag{3.8}$$

where x_i is the i-th logit predicted by the CPN. If the votes are independent and the deviations $\Delta\mathbf{v}_i$ of the votes $\mathbf{v}_i$ from the mean $\overline{\mathbf{v}} \in \mathbb{R}^2$ are normally distributed with

the same standard deviation σ_v, we can model the confidence scores $P(\mathbf{v}_i)$ using the non-normalized Gaussian distribution, which ensures that $P(\mathbf{v}_i)$ is in $\mathbb{R}^{[0,1]}$:

$$P(\mathbf{v}_i) = \frac{e^{x_i}}{e^{x_i}+1} = e^{-\frac{\|\Delta\mathbf{v}_i\|^2}{2\sigma_v^2}} \tag{3.9}$$

Using (3.9) we can derive:

$$\begin{aligned} \|\Delta\mathbf{v}_i\|^2 =& -2\sigma_v^2 \ln(P(\mathbf{v}_i)) = -2\sigma_v^2 \ln\left(\frac{e^{x_i}}{e^{x_i}+1}\right) \\ =& -2\sigma_v^2(x_i - \ln(e^{x_i}+1)) \end{aligned} \tag{3.10}$$

and compute the weighted arithmetic mean of the votes $\mathbf{v}_i$ from weighted least squares:

$$\overline{\mathbf{v}} = \frac{\sum_{i=1}^{N} \frac{\mathbf{v}_i}{\|\Delta\mathbf{v}_i\|^2}}{\sum_{i=1}^{N} \frac{1}{\|\Delta\mathbf{v}_i\|^2}} = \frac{\sum_{i=1}^{N} \frac{\mathbf{v}_i}{x_i - \ln(e^{x_i}+1)}}{\sum_{i=1}^{N} \frac{1}{x_i - \ln(e^{x_i}+1)}} \tag{3.11}$$

In addition, using error propagation by taking the weighted mean of the squared deviations:

$$\begin{aligned} \overline{\|\Delta\mathbf{v}\|^2} =& \sum_{i=1}^{N} \left(\frac{\partial\overline{\mathbf{v}}}{\partial\mathbf{v}_i}\|\Delta\mathbf{v}_i\|\right)^2 = \sum_{i=1}^{N} \left(\frac{\frac{1}{\|\Delta\mathbf{v}_i\|^2}}{\sum_{j=1}^{N}\frac{1}{\|\Delta\mathbf{v}_j\|^2}}\|\Delta\mathbf{v}_i\|\right)^2 \\ =& \left(\sum_{i=1}^{N}\frac{1}{\|\Delta\mathbf{v}_i\|^2}\right)^{-1} = \left(\sum_{i=1}^{N}\frac{1}{-2\sigma_v^2(x_i - \ln(e^{x_i}+1))}\right)^{-1}, \end{aligned} \tag{3.12}$$

we can determine the confidence score for the weighted arithmetic mean $\overline{\mathbf{v}}$ in (3.11):

$$P(\overline{\mathbf{v}}) = e^{-\frac{\overline{\|\Delta\mathbf{v}\|^2}}{2\sigma_v^2}} = e^{\frac{1}{\sum_{i=1}^{N}\frac{1}{x_i - \ln(e^{x_i}+1)}}} \tag{3.13}$$

Note that if the number of votes is $N = 1$, we obtain:

$$P(\overline{\mathbf{v}}) = e^{x_1 - \ln(e^{x_1}+1)} = \frac{e^{x_1}}{e^{x_1}+1} = \text{Sigmoid}(x_1), \tag{3.14}$$

which is the standard Sigmoid function. However, the formulation in (3.11) and (3.13) has several numeric issues. The overflow and underflow in the exponential and logarithmic functions can be prevented by using different functions for positive and negative values of x_i exploiting the two definitions of the Sigmoid function:

$$\text{Sigmoid}(x_i) = \begin{cases} \frac{1}{e^{-x_i}+1} & , x_i > 0 \\ \frac{e^{x_i}}{e^{x_i}+1} & , x_i \leq 0 \end{cases} \tag{3.15}$$

In addition, Laplace smoothing [188] is used to avoid division by zero. Thus, the

following formulation for Deep Multi-Scale Consensus Voting which is numerically more robust than (3.11) and (3.13) is used:

$$\overline{\mathbf{v}} = \frac{\sum_{i=1}^{N} \mathrm{f}(x_i)\mathbf{v}_i}{\sum_{i=1}^{N} \mathrm{f}(x_i)}, \tag{3.16a}$$

$$P(\overline{\mathbf{v}}) = e^{-\frac{1}{\sum_{i=1}^{N} \mathrm{f}(x_i)}}, \tag{3.16b}$$

$$\mathrm{f}(x_i) = \begin{cases} \frac{1+\epsilon}{\ln(e^{-x_i}+1)+\epsilon} & , x_i > 0 \\ \frac{1+\epsilon}{\ln(e^{x_i}+1)-x_i+\epsilon} & , x_i \leq 0 \end{cases} \tag{3.16c}$$

where ϵ is a small constant (e.g., 10^{-6}). The computation can be easily parallelized on GPUs. Therefore, the proposed consensus voting scheme only introduces a minor computational overhead compared to RetinaNet.

Fast Centroid Non-Maximum Suppression

For the proposed Deep Consensus Network a centroid-based Non-Maximum Suppression (NMS) approach analogously to bounding-box based detection methods is proposed. NMS is a greedy solution to the NP-hard weighted independent set problem ([189, 190]). However, still the computation time increases quadratically with the number of objects. NMS can be separated into smaller problems by exploiting the spatial structure of the input data. This reduces the algorithmic complexity from logarithmic to linear in the best case, while the algorithmic growth rate in the worst case does not increase. In the proposed approach, the predicted proposals are arranged in a grid of bins by consensus voting. Clusters of proposals that are spatially disconnected in the grid do not conflict. Therefore, resolving conflicts by NMS can be performed separately for each cluster. In the proposed approach, clusters are determined by connected component analysis after thresholding the confidence scores using a threshold T_s. The NMS is performed in parallel on each partition. The pseudocode for the proposed centroid-based NMS approach is outlined in Algorithm 1. In the pseudocode, the function "connectedcomponents" yields connected components in a grid of binary values, "dilation" performs morphological dilation using a specified window size, and "argmax" yields the index of the input with the highest value.

Figure 3.9 shows the computation time for performing the proposed centroid-based NMS approach compared to a vanilla NMS for an image of size 5000×5000 pixels averaged over 10 runs using a workstation with an Intel i7-8550U CPU and a NVIDIA Geforce MX150. It can be seen that the vanilla NMS has quadratic growth with increasing number of objects while the proposed NMS approximately yields linear

growth.

Input: $\mathbf{v} = \{\mathbf{v}_1, ..., \mathbf{v}_N\}$; $P(\mathbf{v}) = \{P(\mathbf{v_1}), ..., P(\mathbf{v}_N)\}$; T_s; T_d
$\mathbf{v}$: initial centroids
$P(\mathbf{v})$: confidence scores for centroids
T_s: threshold for confidence score
T_d: threshold for minimum distance between two centroids
Output: $\mathbf{v}^*$; $P(\mathbf{v}^*)$
$\mathbf{v}^* \leftarrow \{\}$;
$P(\mathbf{v}^*) \leftarrow \{\}$;
for c **in** connectedcomponents(dilation($P(\mathbf{v}) > T_s, T_d$)) **do**
    $\mathbf{v}_c \leftarrow \mathbf{v}[c]$;
    $\mathbf{P}_c \leftarrow P(\mathbf{v})[c]$;
    $\mathbf{v}_c^* \leftarrow \{\}$;
    while $\mathbf{v}_c \neq \{\}$ **do**
        $m \leftarrow$ argmax($\mathbf{P}_c$);
        $\mathbf{M} \leftarrow \mathbf{v}_c[m]$;
        add = True;
        for $\mathbf{v}_{ci}^*$ **in** $\mathbf{v}_c^*$ **do**
            if $\|\mathbf{v}_{ci}^* - \mathbf{M}\| \leq T_d$ **then**
                add = False;
                break;
            end
        end
        if add **then**
            $\mathbf{v}_c^* \leftarrow \mathbf{v}_c^* \cup \mathbf{M}$;
            $P(\mathbf{v}^*) \leftarrow P(\mathbf{v}^*) \cup P_c[m]$;
        end
        $\mathbf{v}_c \leftarrow \mathbf{v}_c \backslash \mathbf{M}$;
        $\mathbf{P}_c \leftarrow \mathbf{P}_c \backslash P_c[m]$;
    end
    $\mathbf{v}^* \leftarrow \mathbf{v}^* \cup \mathbf{v}_c^*$;
end

Algorithm 1: Centroid Non-Maximum Suppression (NMS) pseudocode.

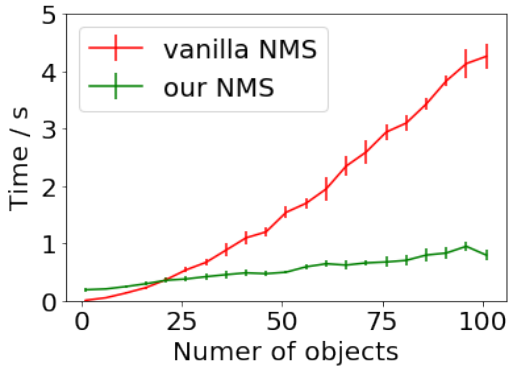


Figure 3.9: Computation time for the proposed centroid-based NMS vs. a vanilla NMS. The standard deviation over 10 runs is shown by error bars.

Normalized Mutual Information as Loss Function for Classification

In object detection, the ratio of positive and negative samples is typically very imbalanced. This is caused by many easy to classify background samples and rare, difficult to classify foreground (objects) samples. To cope with imbalance of the foreground and background class, previous methods perform scaling of the Cross-Entropy loss. For example, [25] introduce a Focal loss that emphasizes hard negatives (samples that are easy incorrectly classified):

$$\mathrm{FL}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{N} \left(1 - P_k(X_i)\right)^{\gamma} P_k(Y_i) \ln(P_k(X_i)). \tag{3.17}$$

$P_k(\mathbf{X}) \in \mathbb{R}^N$ are the predicted probabilities for the samples $\mathbf{X} = (X_1, ..., X_N)$ and $P_k(\mathbf{Y}) \in \mathbb{R}^N$ the ground truth probabilities for the labels $\mathbf{Y} = (Y_1, ..., Y_N)$ of the corresponding k-th class, N the number of samples, γ is the scaling (penalty) factor for false negative predictions, and K the number of classes. $P_k(X_i)$ and $P_k(Y_i)$ denote the i-th probability of the k-th class for $\mathbf{X}$ and $\mathbf{Y}$, respectively. In our application, $P_k(\mathbf{X})$ are the aggregated probabilities $P(\overline{\mathbf{v}})$ from consensus voting in (3.16b). The Focal loss reduces the expected number of the positive predictions $P(\mathbf{X}|\mathbf{Y} = 1)$ by down-weighting the loss of true positives. Thereby, false negatives are emphasized. The Focal loss was extended in [3] by improving the robustness of the training using momentum-based optimizers:

$$\mathrm{NFL}(\mathbf{X}, \mathbf{Y}) = -\frac{\sum_{k=1}^{K} \sum_{i=1}^{N} w_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}, k, i) \ln(P_k(X_i))}{\sum_{k=1}^{K} \sum_{i=1}^{N} w_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}, k, i)} \tag{3.18a}$$

$$w_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}, k, i) = P_k(Y_i) \left(1 - P_k(X_i)\right)^{\gamma}. \tag{3.18b}$$

However, the loss functions used in ([25, 3]) were derived without a probabilistic interpretation. In the following, existing loss functions are analyzed and a novel loss function for object detection within a Bayesian framework is derived. The loss function is based on Normalized Mutual Information and copes with class imbalance as well as emphasizes correlation.

First the Cross-Entropy loss function, which is often used in deep learning methods, is considered. This loss can be formulated using the Bernoulli scheme for the positive predictions:

$$P(\mathbf{X}|\mathbf{Y} = 1) \propto \prod_{k=1}^{K} \prod_{i=1}^{N} P_k(X_i)^{P_k(Y_i)} \tag{3.19}$$

for K classes and N samples. We assume that the samples in $\mathbf{X}$ are identically

distributed and independent. The negative log-likelihood of $P(\mathbf{X}|\mathbf{Y}=1)$ is equal to the Cross-Entropy:

$$\mathrm{CE}(\mathbf{X},\mathbf{Y}) = \mathrm{H}(\mathbf{Y}) + \mathrm{D}_{\mathrm{KL}}(\mathbf{X},\mathbf{Y}) = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N} P_k(Y_i)\ln(P_k(X_i)), \tag{3.20}$$

where $\mathrm{H}(\mathbf{Y})$ is the entropy of $\mathbf{Y}$, and $\mathrm{D}_{\mathrm{KL}}(\mathbf{X},\mathbf{Y})$ the Kullback-Leibler divergence. Thus, minimizing the Cross-Entropy, maximizes the likelihood using the Bernoulli scheme. However, this derivation assumes identically distributed samples within classes which is often not the case in practice.

Other loss functions can be formulated based on the confusion matrix of the two probability distributions of $\mathbf{X}$ and $\mathbf{Y}$. The *normalized confusion matrix* in the *binary case* is:

$$\begin{aligned}\frac{1}{N}\begin{pmatrix}\mathrm{TP} & \mathrm{FP}\\ \mathrm{FN} & \mathrm{TN}\end{pmatrix} &= \frac{1}{N}\begin{pmatrix}\sum_{i=1}^{N} P_2(X_i)P_2(Y_i) & \sum_{i=1}^{N} P_2(X_i)P_1(Y_i)\\ \sum_{i=1}^{N} P_1(X_i)P_2(Y_i) & \sum_{i=1}^{N} P_1(X_i)P_1(Y_i)\end{pmatrix}\\ &= \frac{1}{N}\begin{pmatrix}\sum_{i=1}^{N} P_2(X_i)P_2(Y_i) & \sum_{i=1}^{N} P_2(X_i)(1-P_2(Y_i))\\ \sum_{i=1}^{N}(1-P_2(X_i))P_2(Y_i) & \sum_{i=1}^{N}(1-P_2(X_i))(1-P_2(Y_i))\end{pmatrix},\end{aligned} \tag{3.21}$$

where $P_2(X_i)$ and $P_2(Y_i)$ are the predicted and ground truth foreground class probabilities, respectively, and $P_1(X_i)$ and $P_1(Y_i)$ the predicted and ground truth background class probabilities, respectively, which are used to estimate the true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). If we assume that the entries in the normalized confusion matrix follow a multinomial distribution, the common Dice loss can be derived as shown in the following.

The expectation of $P_k(\mathbf{Y}|\mathbf{X})$ of a particular class k can be formulated using a Bernoulli scheme with a beta prior and yields the Precision [191]:

$$\mathrm{Prec}_k(\mathbf{X},\mathbf{Y}) = \overline{P_k}(\mathbf{Y}|\mathbf{X}) = \sum_{i=1}^{N} P_k(Y_i)P_k(X_i|\mathbf{X}) \tag{3.22}$$

where $P_k(X_i|\mathbf{X})$ can be estimated empirically by $P_k(X_i|\mathbf{X}) = P_k(X_i)/\sum_{j=1}^{N} P_k(X_j)$. The beta prior is a conjugate prior probability distribution for the Bernoulli distribution, since it is in the same probability distribution family. We can also formulate the expectation of $P(\mathbf{X}|\mathbf{Y})$ which yields the Sensitivity:

$$\mathrm{Sens}_k(\mathbf{X},\mathbf{Y}) = \overline{P_k}(\mathbf{X}|\mathbf{Y}) = \sum_{i=1}^{N} P_k(X_i)P_k(Y_i|\mathbf{Y}) \tag{3.23}$$

where $P_k(Y_i|\mathbf{Y})$ can be estimated empirically by $P_k(Y_i|\mathbf{Y}) = P_k(Y_i)/\sum_{j=1}^{N} P_k(Y_j)$. Both (3.22) and (3.23) should be optimized to increase the joint occurrence of $\mathbf{X}$ and $\mathbf{Y}$. Taking the (negative) *arithmetic mean* of Precision and Sensitivity in (3.22)

and (3.23) results in a loss function with a symmetric distance measure regarding $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathcal{L}_{\text{arith}}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{2} \sum_{k=1}^{K} \left(\text{Prec}_k(\mathbf{X}, \mathbf{Y}) + \text{Sens}_k(\mathbf{X}, \mathbf{Y})\right) \tag{3.24}$$

Another possibility of combining Precision and Sensitivity is using the *minimum* of them:

$$\mathcal{L}_{\min}(\mathbf{X}, \mathbf{Y}) = -\sum_{k=1}^{K} \min\left\{\text{Prec}_k(\mathbf{X}, \mathbf{Y}), \text{Sens}_k(\mathbf{X}, \mathbf{Y})\right\} \tag{3.25}$$

Taking the minimum has the advantage compared to the arithmetic mean that the relatively rare worst case (maximum loss) is emphasized over the frequent average case. Alternatively, the *harmonic mean* can be used which yields values between the arithmetic mean and the minimum, if the values are in $\mathbb{R}_0^+$, which is the case for (3.22) and (3.23). Using the harmonic mean leads to the multi-class Dice loss [191, 72]:

$$\begin{aligned} \mathcal{L}_{\text{Dice}}(\mathbf{X}, \mathbf{Y}) &= -2 \sum_{k=1}^{K} \left(\frac{1}{\text{Prec}_k(\mathbf{X}, \mathbf{Y})} + \frac{1}{\text{Sens}_k(\mathbf{X}, \mathbf{Y})} \right)^{-1} \\ &= -2 \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} P_k(X_i) P_k(Y_i)}{\sum_{i=1}^{N} (P_k(X_i) + P_k(Y_i))} \end{aligned} \tag{3.26}$$

However, the estimators for Precision and Sensitivity of the Dice loss in (3.26) have been derived using Bayesian inference of a Bernoulli scheme with a beta prior [191]. Thus, the estimators are overoptimistic and focus on the true positives in the confusion matrix while the other entries are not considered [192]. Therefore, the Dice loss is not optimal to emphasize hard negatives as in the Focal loss [25].

Another possibility for a loss function is using the Matthews Correlation Coefficient (MCC), which considers all entries in the confusion matrix [193]. MCC is defined based on the geometric mean of the Markedness and Informedness regression coefficients of the problem and its dual [192]. Analogously to the multi-class Dice loss in (3.26), we can formulate the multi-class MCC loss for K classes to be defined as:

$$\mathcal{L}_{\text{MCC}}(\mathbf{X}, \mathbf{Y}) = \frac{-\sum_{k=1}^{K} \sum_{l=1}^{K} \sum_{m=1}^{K} \mathrm{g}_{kk}\, \mathrm{g}_{lm} - \mathrm{g}_{kl}\, \mathrm{g}_{mk}}{\sqrt{\sum_{k=1}^{K} \left(\sum_{l=1}^{K} \mathrm{g}_{lk} \right) \left(\sum_{\substack{m=1 \\ m \neq k}}^{K} \sum_{n=1}^{K} \mathrm{g}_{mn} \right)} \sqrt{\sum_{k=1}^{K} \left(\sum_{l=1}^{K} \mathrm{g}_{kl} \right) \left(\sum_{\substack{m=1 \\ m \neq k}}^{K} \sum_{n=1}^{K} \mathrm{g}_{nm} \right)}} \tag{3.27a}$$

$$g_{kl} = \sum_{i=1}^{N} P_k(X_i) P_l(Y_i) \tag{3.27b}$$

The *binary* MCC is equivalent to the discrete case of the Pearson Correlation Coefficient:

$$\rho_{\mathbf{X},\mathbf{Y}} = \frac{\mathrm{cov}(\mathbf{X}, \mathbf{Y})}{\sigma_{\mathbf{X}}\, \sigma_{\mathbf{Y}}} \tag{3.28}$$

which describes a normalized version of the linear dependency of the two variables $\mathbf{X}$ and $\mathbf{Y}$ using their variances $\sigma_{\mathbf{X}}$, $\sigma_{\mathbf{Y}}$ and covariance $\mathrm{cov}(\mathbf{X}, \mathbf{Y})$. Therefore, if one variable is known, then the other variable can be predicted. The square of the MCC is related to the χ^2 statistic, which is the likelihood-ratio test statistic for χ^2 distributions. Thus, the MCC does not include an additional beta prior like the Dice. Due to the underlying assumptions, the MCC assumes homoscedasticity, which means that the sum of squared deviations for each class has a Gaussian distribution, with parameters uniformly distributed over all classes. In practice, a Gaussian distribution cannot be assumed in the presence of hard negative samples. Therefore, the MCC is prone to outliers due to hard negative samples and label noise.

Based on the analysis of loss functions above and the gained insights, a novel loss function assuming a Bernoulli distribution instead of the Gaussian distribution as for the MCC is proposed. The Bernoulli distribution is better suited in the case of a small number of samples [194]. For a Bernoulli trial, the analog concept of covariance is the mutual information, which represents the information gain knowing both variables instead of just one:

$$\mathrm{MI}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{N} P(X_i, Y_i) \log \frac{P(X_i, Y_i)}{P(X_i) P(Y_i)} \tag{3.29}$$

MI assembles the popular Cauchy-Schwarz divergence [195] weighted by the joint probability of $\mathbf{X}$ and $\mathbf{Y}$. Compared to the Cauchy-Schwarz divergence, MI maximizes the joint probability of $\mathbf{X}$ and $\mathbf{Y}$ in addition to the angle between marginal and joint probabilities, which turned out to be beneficial in our multi-task learning problem. Moreover, a complete geometric interpretation of the Cauchy-Schwarz divergence and MI exists [195]. Since discrete events are considered, MI cannot be calculated pointwise. Instead, MI is determined based on the confusion matrix in (3.21) and assume independence between $\mathbf{X}$ and $\mathbf{Y}$. The summand for the i-th entry in the confusion matrix is computed by:

$$\mathrm{MI}_i(X_i, Y_i) = P(X_i, Y_i) \log \frac{P(X_i, Y_i)}{P(X_i) P(Y_i)} \tag{3.30}$$

A normalization of MI analogously to the Pearson Correlation Coefficient in (3.28) is

suggested by using the uncertainty coefficients of MI [196]:

$$\mathrm{C_{XY}}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} \frac{\mathrm{MI}_i(X_i,Y_i)}{\mathrm{CE}(Y_i,Y_i)},\ \mathrm{C_{YX}}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} \frac{\mathrm{MI}_i(X_i,Y_i)}{\mathrm{CE}(X_i,X_i)} \tag{3.31}$$

where CE is the Cross-Entropy defined in (3.20). The uncertainty coefficients in (3.31) can be combined in different ways yielding different variants of a normalized loss function. Using the *arithmetic mean* leads to:

$$\mathrm{NMI_{AM}}(\mathbf{X},\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{N} \mathrm{MI}_i(X_i,Y_i)\left(\frac{1}{\mathrm{CE}(X_i,X_i)} + \frac{1}{\mathrm{CE}(Y_i,Y_i)}\right)\ , \tag{3.32}$$

the *geometric mean* yields:

$$\mathrm{NMI_{GM}}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} \frac{\mathrm{MI}_i(X_i,Y_i)}{\sqrt{\mathrm{CE}(X_i,X_i)\,\mathrm{CE}(Y_i,Y_i)}}\ , \tag{3.33}$$

and the *harmonic mean* leads to:

$$\mathrm{NMI_{HM}}(\mathbf{X},\mathbf{Y}) = 2\sum_{i=1}^{N} \frac{\mathrm{MI}_i(X_i,Y_i)}{\mathrm{CE}(X_i,X_i) + \mathrm{CE}(Y_i,Y_i)} \tag{3.34}$$

The arithmetic mean in (3.32) is not well suited since, similar to the Dice and MCC loss, the aim is to increase the influence of the estimator with the lowest value onto the overall loss. Instead, we can utilize the geometric or harmonic mean. In the following, it is shown that the harmonic mean in (3.34) is better suited since the geometric mean in (3.33) generally increases the complexity of the loss surface and therefore the expected number of local minima.

Without loss of generality, we show this for two functions $f(x)$ and $g(x) \in \mathbb{R}_0^+$. Assuming that $f(x)$ and $g(x)$ are presented by Taylor series (polynomials) with degrees m and n of the original functions, the derivative (gradient) for the *geometric mean* can be calculated as:

$$\frac{\partial}{\partial x}\left(\sqrt{f(x)g(x)}\right) = \frac{g(x)f'(x) + f(x)g'(x)}{2\sqrt{f(x)g(x)}} \tag{3.35}$$

The polynomial complexity of (3.35) can be determined by analyzing and summing

up the degree (deg) of each term:

$$\begin{aligned}\deg\left(\frac{g(x)f'(x)+f(x)g'(x)}{2\sqrt{f(x)g(x)}}\right) &= \max\{(n+m-1),(m+n-1)\}-\left(\frac{n+m}{2}\right)\\ &= n+m-1-\frac{n+m}{2}\\ &= \frac{n+m}{2}-1 \end{aligned} \tag{3.36}$$

The gradient of the *harmonic mean* is given by

$$\frac{\partial}{\partial x}\left(\frac{2f(x)g(x)}{f(x)+g(x)}\right)=\frac{2(g(x)^2f'(x)+f(x)^2g'(x))}{(f(x)+g(x))^2} \tag{3.37}$$

and the polynomial degree is:

$$\deg\left(\frac{2(g(x)^2f'(x)+f(x)^2g'(x))}{(f(x)+g(x))^2}\right)= \max\{(2n+m-1),(2m+n-1)\}-(2\max\{n,m\}) \tag{3.38a}$$

and without loss of generality for $n \leq m$ we obtain:

$$\begin{aligned}\deg\left(\frac{2(g(x)^2f'(x)+f(x)^2g'(x))}{(f(x)+g(x))^2}\right) &= -2m-1+\begin{cases}2n+m & ,n\geq m\\ 2m+n & ,\text{otherwise}\end{cases}\\ &= 2m+n-2m-1\\ &= n-1\end{aligned} \tag{3.38b}$$

Since $n \leq m$ is assumed, the degree of the harmonic mean $n-1$ in (3.38b) is always smaller or equal to the degree of the geometric mean $(n+m)/2-1$ in (3.36). Thus, the harmonic mean increases the complexity of the loss surface generally less than the geometric mean. Therefore, the harmonic mean $\mathrm{NMI}_{\mathrm{HM}}$ is used in the proposed approach. Usually, the mutual information is calculated over all entries in the confusion matrix. This enables arbitrary correlation of variables, which is useful in clustering where data to label association is unknown. However, in a supervised setting as in our case the class affiliation is known. Therefore, we maximize $\mathrm{NMI}_{\mathrm{HM}}$ over the diagonal entries of the confusion matrix (true positives and true negatives) and minimize the off diagonal entries (false positives and false negatives) by weighting the mutual information. However, in the experiments it was found that minimizing the entries for the false positives reduced the performance. Therefore, only the entries for the true positives and the true negatives are maximized. In the binary case, where $\mathbf{X}=(X_1,...,X_N)$ and $\mathbf{Y}=(Y_1,...,Y_N)$ denote the foreground class and $\mathbf{X}^-=(1-X_1,...,1-X_N)$ and $\mathbf{Y}^-=(1-Y_1,...,1-Y_N)$ the background class, we

then have to minimize:

$$\mathcal{L}_{\mathrm{NMI}}(\mathbf{X}, \mathbf{Y}) = -\,\mathrm{NMI}_{\mathrm{HM}}(\mathbf{X}, \mathbf{Y}) - \mathrm{NMI}_{\mathrm{HM}}(\mathbf{X}^{-}, \mathbf{Y}^{-}) \tag{3.39}$$

Interestingly, due to the assumed multinomial model for the entries of the confusion matrix, the mutual information of the confusion matrix is related to the G-test (up to a factor of $2N$) [197]. Therefore, maximizing the mutual information can be interpreted as maximizing the G-test statistic. Moreover, mutual information can be interpreted as the KL-divergence of the theoretical distribution from the empirical distribution of $\mathbf{X}, \mathbf{Y}$ pairs [198].

Note that normalization of the mutual information can also be based on interpreting the mutual information as intersection of marginal entropies $\mathrm{H}(\mathbf{X})$ and $\mathrm{H}(\mathbf{Y})$. In the context of classical medical image registration, [199] proposed a normalization using the joint entropy $\mathrm{H}(\mathbf{X}, \mathbf{Y})$ which is the union of $\mathrm{H}(\mathbf{X})$ and $\mathrm{H}(\mathbf{Y})$:

$$\frac{\mathrm{H}(\mathbf{X}) + \mathrm{H}(\mathbf{Y})}{\mathrm{H}(\mathbf{X}, \mathbf{Y})} = 1 + \frac{\mathrm{MI}(\mathbf{X}, \mathbf{Y})}{\mathrm{H}(\mathbf{X}, \mathbf{Y})} \tag{3.40}$$

The normalized mutual information in (3.40) is the Intersection over Union (IoU), while the proposed NMI in (3.34) corresponds to the Dice of the marginal entropies. In general, IoU tends to penalize single estimates more than the Dice. In our application, the estimation is performed for each batch. The Dice tends to be closer to the average model performance, while the IoU is closer to the worst case model performance. Thus, NMI in (3.34) is generally more robust to changes among batches compared to (3.40).

The derived loss function $\mathcal{L}_{\mathrm{NMI}}$ in (3.39) emphasizes class balance and correlation of $\mathbf{X}$ and $\mathbf{Y}$. To achieve smooth predicted probabilities, the loss function is combined with the binary Cross-Entropy and balance it so that the gradients of $\mathcal{L}_{\mathrm{NMI}}$ and CE yield approximate equal contribution. The novel loss function for classification (cls) is then given by:

$$\mathcal{L}_{cls}(\mathbf{X}, \mathbf{Y}) = \frac{3}{4K}\mathcal{L}_{\mathrm{NMI}}(\mathbf{X}, \mathbf{Y}) + \frac{1}{4N}\,\mathrm{CE}(\mathbf{X}, \mathbf{Y}), \tag{3.41}$$

where K is the number of classes and N the number of samples. Figure 3.10 shows the gradient of CE, $\mathcal{L}_{\mathrm{Dice}}$, $\mathcal{L}_{\mathrm{NMI}}$, and $\mathcal{L}_{cls}$ for two samples X_i, Y_i in batches of samples with varying class balance. In contrast to the other losses, CE is agnostic to class imbalance. In addition, $\mathcal{L}_{\mathrm{Dice}}$ and $\mathcal{L}_{\mathrm{NMI}}$ are penalizing class imbalance more than incorrect classification. $\mathcal{L}_{cls}$ balances well between penalizing incorrect classification and class imbalance.

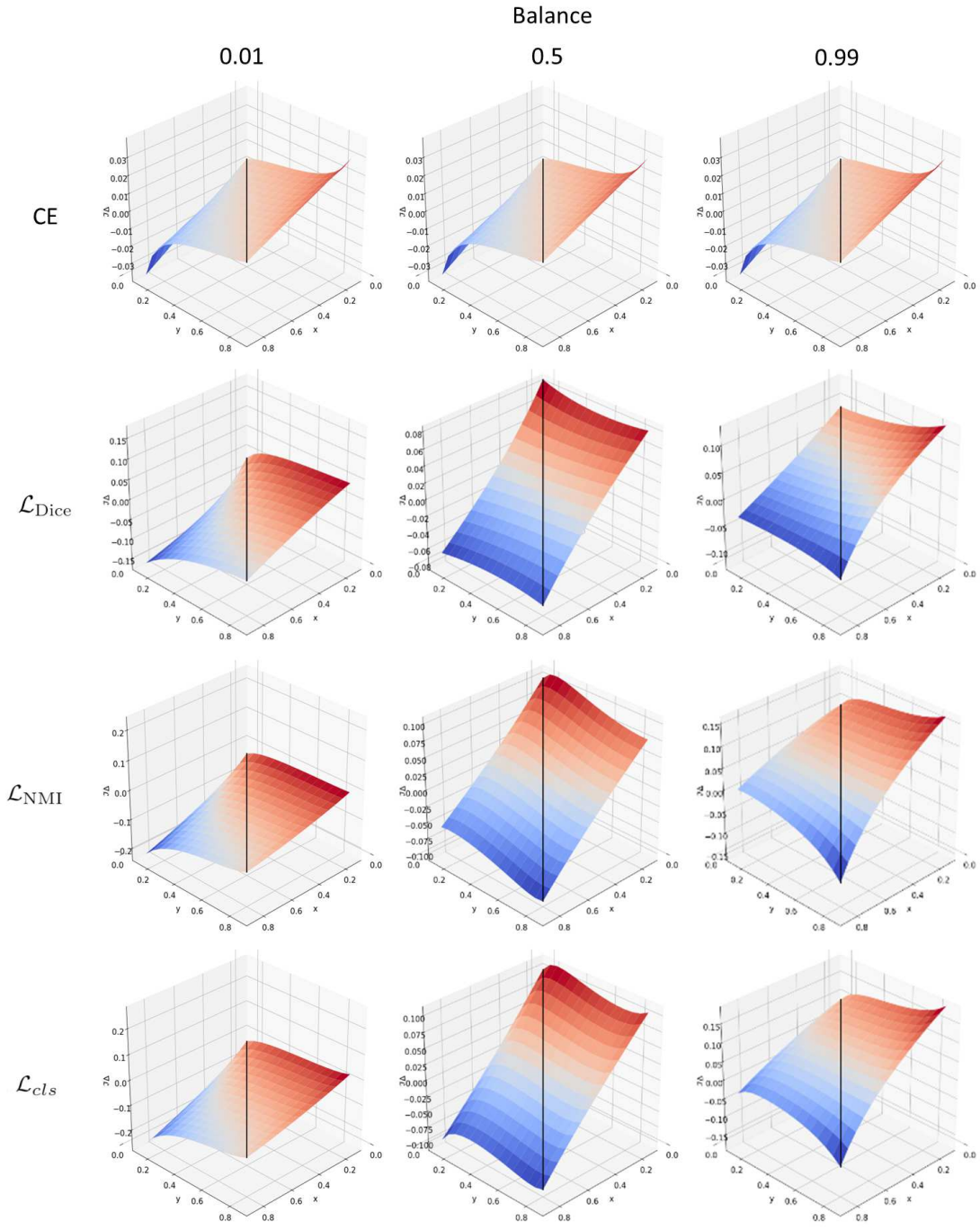


Figure 3.10: Gradient of CE, $\mathcal{L}_{\text{Dice}}$, $\mathcal{L}_{\text{NMI}}$, and $\mathcal{L}_{cls}$ calculated on samples X_i, Y_i within batches $\mathbf{X}, \mathbf{Y}$ with class balance of 0.01, 0.50, and 0.99. The curve where x is equal to y is marked in black.

Model Training

The proposed Consensus Voting Network is trained by jointly performing regression (reg) of object centroids and classification (cls) based on corresponding predictions of confidence scores. For regression, the Charbonnier loss function [200]

is adopted which serves as a smooth approximation to the Huber loss for robust regression [75]:

$$\mathcal{L}_{reg}(\mathbf{v}, \mathbf{v}^{\mathrm{GT}}) = \sum_{i=1}^{N} \mathrm{h}(\mathbf{v}_i^{\mathrm{GT}}, T_d) \frac{1}{N T_d} \left\| \sqrt{1 + \left(\frac{2\ (\overline{\mathbf{v}}_i - \mathbf{v}_i^{\mathrm{GT}})}{T_d} \right)^2} - 1 \right\|_1 \tag{3.42}$$

where $\overline{\mathbf{v}}$ denotes the aggregated predicted votes, $\mathbf{v}^{\mathrm{GT}}$ the ground truth votes, and $\mathrm{h}(\mathbf{v}_i^{\mathrm{GT}}, T_d)$ an indicator function which is set to one if the length of $\mathbf{v}_i^{\mathrm{GT}}$ is smaller than T_d (minimum distance between two detections). The ground truth votes are the relative offset vectors of each anchor to the nearest ground truth detection. The confidence scores are optimized using the NMI-based classification loss $\mathcal{L}_{cls}(P(\overline{\mathbf{v}}), P(\mathbf{v}^{\mathrm{GT}}))$ in (3.41), where $P(\overline{\mathbf{v}})$ are the aggregated predicted confidence scores and $P(\mathbf{v}^{\mathrm{GT}})$ the ground truth confidence scores. The i-th ground truth confidence score is calculated by:

$$P(\mathbf{v}_i^{\mathrm{GT}}) = z\ \exp\left(-\frac{\|\mathbf{v}_i - \mathbf{v}_i^{\mathrm{GT}}\|_2^2}{2(T_d/(4\sqrt{2}))^2} \right), \tag{3.43}$$

where z is an indicator variable which is set to one if an object is present in the ground truth. The final multi-task loss function $\mathcal{L}$ is a combination of the NMI-based loss in (3.41) and the regression loss in (3.42):

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{reg} \tag{3.44}$$

The multi-task loss is minimized using the AMSGrad optimizer [56] with decoupled weight decay $w = 10^{-4}$ [61] and a learning rate of $l_{\mathrm{init}} = 0.0001$ as well as $\beta_1 = 0.975$ and $\beta_2 = 0.999$. The training datasets are augmented by random cropping, brightness changes, contrast changes, flipping, and rotation.

4 Segmentation of Microscopy Images

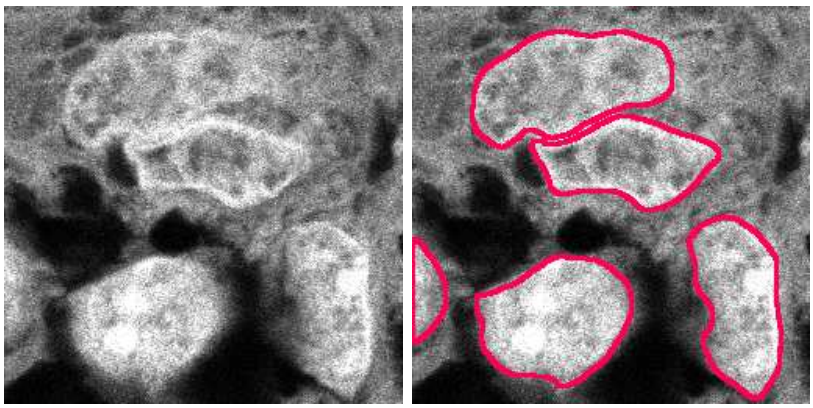
4.1 Overview and Task Description

Segmentation of prominent structures such as cells in microscopy images is a frequent and important task. In particular, features computed from cell nucleus and cytoplasm segmentations are used to determine phenotypes in quantitative microscopy.

There exist different types of methods for segmentation, but in recent years methods based on deep learning dominate the field of computer vision. Segmentation methods based on deep learning have been successfully used for cell segmentation in microscopy images (e.g., [27, 110, 152, 153]). In microscopy images, context is important to identify objects in complex data like histological images. Moreover, splitting of close objects and robust fusion of information from multiple scales is important to yield accurate object segmentations. In addition, usually, many clustered objects of the same class have to be segmented in an image and object-wise readout has to be calculated. Therefore, object instance identification is crucial. In this chapter, novel methods for tackling these domain-specific challenges are presented. In Section 4.2 an approach for increasing context information by increasing the receptive field of hourglass-shaped neural networks is presented and a method for automatic telomere quantification in glioblastoma and prostate tissue images is described. In Section 4.3, an architecture which combines a CNN with an RNN for robust fusion of information from multiple scales and a novel objective for multi-scale feature extraction and iterative refinement of features are proposed. Approaches for identification of clustered objects of the same class are investigated in Section 4.4. The methods were published in Wollmann et al. [15, 13, 3].

4.2 ASPP-Net for Cell Segmentation

Splitting of cells is one of the major challenges in microscopic cell segmentation. For example, the data from 3D tissue microscopy images of glioblastoma cells is very challenging due to strong intensity variation, cell clustering, poor edge information, missing object borders, strong shape variation, and low signal-to-noise ratio, which can be seen in Figure 4.1. The receptive field of a CNN has to be relatively large to collect enough clues in terms of context to determine if a pixel is an object

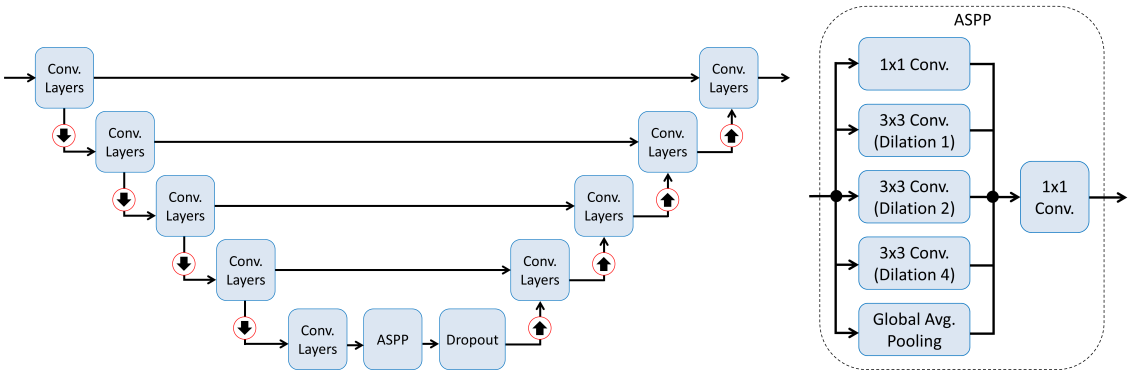


(a) Original image (b) Manual annotation

Figure 4.1: DAPI channel of original tissue image of glioblastoma cells and ground truth annotation.

border. Therefore, a novel deep neural network based on atrous spatial pyramid pooling (ASPP) [28] for cell segmentation is introduced in this thesis. The work has been published in [15, 13].

The proposed deep learning method ASPP-Net combines a U-Net [27] with batch normalization [63], residual connections [29], and ASPP [28]. ASPP has the advantage that large context information can be captured at multiple image scales. An ASPP was modified by using dilations of 1, 2, and 4 as well as global average pooling (pooling kernel equal to feature maps) to capture information from the whole image (Figure 4.2b). After the ASPP block, Gaussian dropout ($p = 0.5$) is employed. For the deep learning model, PReLU [59] was used as activation function. Using a U-Net in conjunction with a PReLU activation function, it was observed that the first layers mostly favour negative activations. However, PReLU increases the computation time. Therefore, PReLU was only used in the first layer to make use of negatively activated features while saving computation time. The network's architecture is outlined in Figure 4.2a. The network is trained using cross-validation



(a) Overview of ASPP-Net network architecture (b) Modified ASPP block

Figure 4.2: Deep neural network architecture of ASPP-Net

and early stopping with the Adam optimizer and a learning rate of $l_{init} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The dataset is always split into 50 % training, 25 % validation, and 25 % testing data. Datasets are augmented using random flipping, rotation, cropping (200×200 pixels), color shift, and elastic deformations.

The developed ASPP-Net was applied for telomere quantification in glioblastoma and prostate tissue images. Telomeres form the end of linear chromosomes in humans and prevent the DNA damage signaling machinery from recognizing chromosome ends as double-strand breaks. During each cell replication, the number of 5'-TTAGGG-3' sequence repeats of the telomeres decrease until it reached a critical limit, which results in apoptosis. Cancer cells circumvent this control mechanism by telomerase, which extends telomeres. Therefore, they can proliferate indefinitely. Mutations in the promoter of the TERT gene as well as structural rearrangements of TERT enhancers lead to telomerase [201, 202, 203]. On the other hand, alternative lengthening of telomeres (ALT) mechanisms based on DNA recombination and repair processes can also suppress telomerase, which can lead to heterogeneous telomere length distribution within individual cells and across tumor cell populations [204]. For example, PITX1 gene suppresses TERT activity by binding to the TERT promoter [205]. Thus, studying telomere length and PITX1 expression could lead to better understanding of TERT and yield a precursor for novel cancer therapies.

In this thesis, a workflow for automatic large scale quantification of telomere length and PITX1 expression per cell is proposed. Tissue slides acquired from glioblastoma and prostate tumors are considered which have been prepared and imaged similar to the protocol in [206]. The proposed workflow outlined in Figure 4.3 projects the 3D patches to 2D using maximum intensity projection (MIP). The patches are stitched to a tissue core and the tumor is manually annotated by a pathologist. The binary masks are sliced to match the patches and within the tumor regions segmentation of cell nuclei is performed using ASPP-Net presented in Section 4.2. Within the segmentation masks telomeres are detected using parametric intensity models [207] and the mean intensity of PITX1 in the cell nuclei and telomere-wise

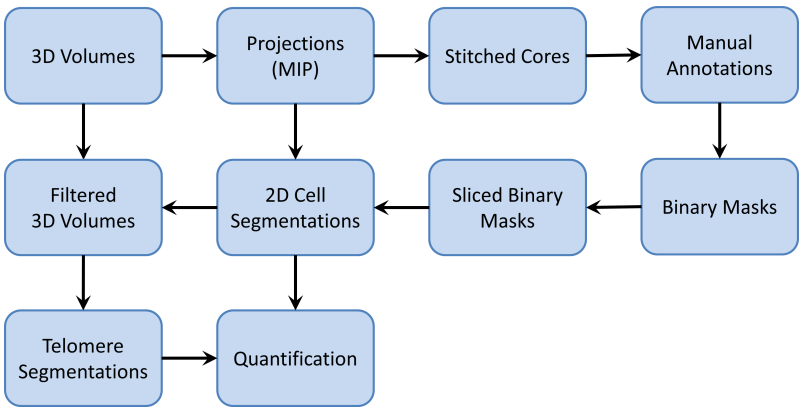


Figure 4.3: Workflow for telomere quantification in glioblastoma and prostate tissue images.

mean staining intensity is quantified. ASPP-Net is pre-trained on all training images from the Cell Tracking Challenge [126]. Furthermore, the fractal nature of microscopy images is exploited by training with progressive resizing from one quarter resolution to the original resolution of the dataset, which significantly reduces training time. Finally, the network is fine-tuned on the respective glioblastoma and prostate datasets containing 50 manually annotated images each.

4.3 GRUU-Net: Integrated Convolutional and Gated Recurrent Neural Network for Cell Segmentation

For cell segmentation, hourglass-shaped convolutional neural networks (CNNs) such as the U-Net [27] or Deconvolution Network [125] are typically used, which aggregate features at multiple image scales. In contrast, recurrent neural networks (RNNs) iteratively refine the segmentation result by exploiting the recurrent structure and mimic Conditional Random Fields (CRFs) or Level Sets [144, 145]. Often, RNN approaches are used in a subsequent step to refine segmentation results from an hourglass-shaped CNN [28]. Segmentation using multi-scale feature aggregation by CNNs and iterative refinement performed by RNNs have distinct strengths and weaknesses. It has been shown that CNNs are very effective in capturing hierarchical patterns and extracting abstract features [46]. However, a drawback of standard CNNs is that they handle each pixel as a separate classification task and do not explicitly include global priors like shape. In contrast, RNNs iteratively minimize global energies. Multiple weak predictions are combined and the final prediction is iteratively improved using global priors like shape. Therefore, RNNs are robust to local errors and require less parameters than CNNs. However, current RNN-based approaches for segmentation [144, 145] incorporate features only at a single scale. Combining iterative refinement with multi-scale feature aggregation and exploiting their strengths could be beneficial. Recently, a CNN for segmentation of street scenes in video images was proposed, which uses a full-resolution feature path combined with hierarchical feature aggregation [140]. However, iterative refinement of features is limited to summing up the extracted feature maps of each Full-Resolution Residual Unit (FRRU). Other approaches perform full-resolution feature extraction using dilated convolutions [88, 16]. However, with these approaches, undesirable "checkerboard" artifacts occur [130]. In addition, [88, 140, 16] do not use an RNN for iterative refinement. Generally, deep neural networks tend to outperform shallow networks [47], but due to non-linear activation functions and multiplications, they suffer from gradient vanishing. In recent years, deep neural network (DNN) architectures like ResNet [29] or DenseNet [30] have been proposed to improve the gradient flow. Residual Connections [151] and Densely Connected blocks [132] have been transferred from classification tasks to semantic segmentation.

Despite the effectiveness of deep learning methods dealing with large image datasets

of natural scenes like ImageNet or MS COCO, it has been shown that training is feasible with relatively small datasets. Common approaches for training on small datasets are transfer learning, adversarial training, and data augmentation. For microscopy images, it has been shown that transfer learning is not very effective, as the properties of the images are quite different from natural images [124]. Adversarial training improves the performance, but does not incorporate domain knowledge, which can help to reduce overfitting (e.g., [155]). In contrast, data augmentation (e.g., [27, 13, 15]) is a computational efficient and effective method to increase the training data set size, incorporate domain knowledge, and prevent overfitting. However, data augmentation for real datasets poses a number of challenges. Enlarging the dataset has to be performed with care to improve and not harm the training. In particular, the utilized sampling strategy for the data can bias the network to a certain class or feature. On the other hand, performing transformations like elastic deformation can lead to degenerated objects. In addition, technical challenges arise when data augmentation is performed with a huge amount of data. Heavy augmentation of datasets can quickly result in millions of images, which exceed terabytes of data volume, and even simple operations are then computationally demanding. By naively transferring the generated images to the GPU memory for further processing, the capabilities of the GPU are generally not fully exploited. Therefore, smart techniques for efficient data streaming are required.

In this section, a novel deep neural network is introduced, which combines both paradigms of multi-scale feature aggregation of CNNs and iterative refinement of RNNs (Wollmann et al. [3]). Compared to previous approaches, in the proposed method, a convolutional and a recurrent neural network are integrated to aggregate features from different image scales. By employing Densely Connected blocks in the CNN part and a gated recurrent unit (GRU) in the RNN part of the proposed network, the number of learnable parameters and feature tensors are kept to a minimum. Since the proposed network combines a GRU with a U-Net-like network, it is denoted as GRUU-Net. A novel focal loss function is proposed for momentum-based optimizers, which enforces the network to learn separating touching objects. Also, a framework for performing data augmentation for generating huge amounts of data is described. Pitfalls and solutions in data handling, dataset sampling, and data transformations are described. A quantitative comparison with state-of-the art methods using challenging real microscopy image data of DAPI stained cell nuclei in glioblastoma tissue is performed. Insights into the proposed novel loss function, the refinement process, and the proposed data augmentation scheme are provided. In addition, the proposed method is benchmarked using a wide spectrum of all 22 real 2D and 3D datasets of the Cell Tracking Challenge, and yields superior or competitive results for most of the datasets.

Architecture of GRUU-Net

GRUU-Net has a fully convolutional network architecture as sketched in Figure 4.4. The neural network unifies a recurrent processing stream with a pooling stream. Both streams are based on different paradigms. The recurrent stream iteratively refines features at full resolution, whereas the pooling stream extracts high-level features within a large receptive field. The two streams are capable of exchanging information on each resolution level. To maximize the gradient flow instead of using a Feed-Forward Network [92], a Residual Network [29] is used, which is also a recurrent network. Carefully designed recurrent units are capable of using a residual as shown in [96]. Therefore, the principle of residual connections is kept throughout the network to maximize gradient flow.

Recurrent Stream

The recurrent stream of GRUU-Net performs iterative refinement of initially extracted features at full resolution. A GRU [208] is used and unfolded over all scales in both bottom-up and top-down paths of the pooling stream. A GRU (Figure 2.5) computes a candidate state $\tilde{\mathbf{h}}_t \in \mathbb{R}^{m \times n \times p}$ from the previous state $\mathbf{h}_{t-1} \in \mathbb{R}^{m \times n \times p}$ and uses the update gate $\mathbf{z}_t \in \mathbb{R}^{m \times n \times p}$ to weight the previous state and the candidate state. Instead of a standard GRU, which operates in a fully-connected manner on a fixed image size, a convolutional version of a GRU [89] is used. Therefore, all fully-connected layers within the standard GRU are replaced by 3×3 convolutions. First, the reset gate $\mathbf{r}_t \in \mathbb{R}^{m \times n \times p}$ and update gate $\mathbf{z}_t$ are calculated using the input $\mathbf{x}_t \in \mathbb{R}^{m \times n \times p}$

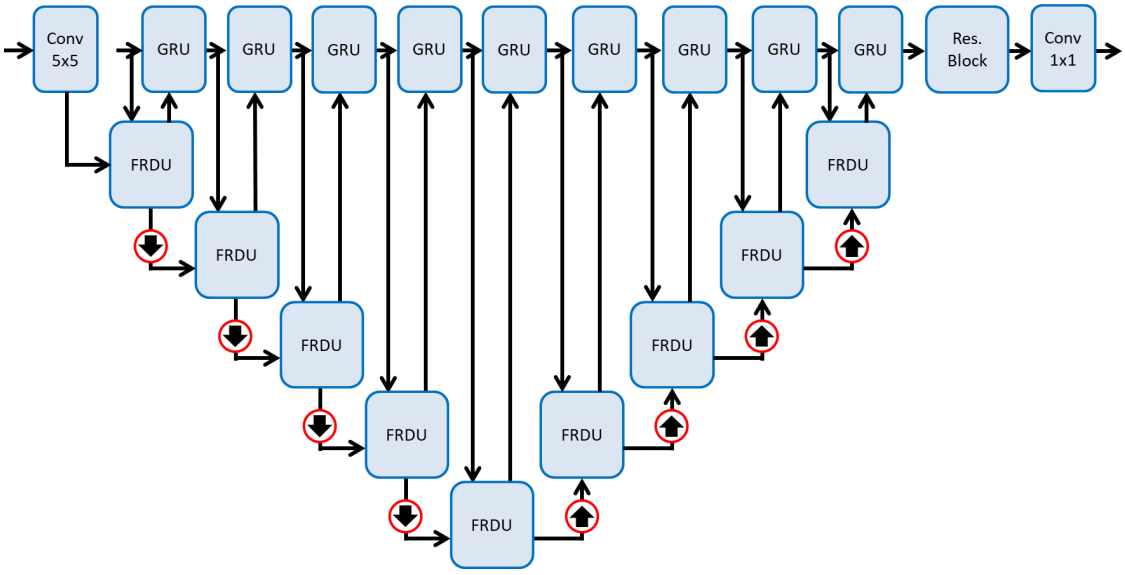


Figure 4.4: GRUU-Net architecture. Red circles with an arrow pointing upward/downward denote unpooling/pooling. At each scale Full-Resolution Dense Units (FRDUs) extract features, which are aggregated by a gated recurrent unit (GRU).

and the parameters $\mathbf{W}_r \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_r \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{b}_r \in \mathbb{R}$, $\mathbf{W}_z \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_z \in \mathbb{R}^{k\times k\times p\times g}$, and $\mathbf{b}_z \in \mathbb{R}$:

$$\mathbf{r}_t = \sigma_g(\mathbf{W}_r * \mathbf{x}_t + \mathbf{U}_r * \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{4.1}$$

$$\mathbf{z}_t = \sigma_g(\mathbf{W}_z * \mathbf{x}_t + \mathbf{U}_z * \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{4.2}$$

$$\sigma_g(x) = \frac{e^x}{e^x + 1} \tag{4.3}$$

where the operator "$*$" denotes convolution. Then, the candidate state $\tilde{\mathbf{h}}_t$ is calculated using the parameters $\mathbf{W}_h \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{U}_h \in \mathbb{R}^{k\times k\times p\times g}$, $\mathbf{b}_h \in \mathbb{R}$:

$$\tilde{\mathbf{h}}_t = \sigma_h(\mathbf{W}_h * \mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{4.4}$$

where the operator "$\odot$" denotes the Hadamard product. For the activation function σ_h, Leaky Rectified Linear Units (LReLU) [58] is used. Finally, the previous state $\mathbf{h}_{t-1}$ and the candidate state $\tilde{\mathbf{h}}_t$ are weighted to determine the new state $\mathbf{h}_t \in \mathbb{R}^{m\times n\times p}$.

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + \ (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \tag{4.5}$$

Pooling Stream

The pooling stream consists of pooling blocks, Full-Resolution Dense Units (FRDUs), and unpooling blocks. To increase the size of the receptive field and the number of feature maps of the network, a bottom-up path with max pooling blocks is included. To restore the original resolution and perform top-down inference, a top-down path is constructed. Within this path, bilinear interpolation is performed instead of transposed convolution as in the U-Net. In [106], it has been shown that both bottom-up and top-down paths for feature extraction are important for capturing the semantic information of an image. Both bottom-up and top-down paths consist of alternating pooling/unpooling and FRDU blocks.

FRDU blocks (Figure 4.5) combine information from the recurrent stream with the pooling stream and feed back the results to both streams. Therefore, the FRDU is capable of integrating convolutional and gated recurrent neural networks. Thus, high resolution information can be stored in the recurrent stream, and at the same time, high-level features can be extracted in the pooling stream at multiple resolutions. To combine the feature maps from both streams $\mathbf{h}_{t-1}$ and $\mathbf{o}_{t-1} \in \mathbb{R}^{m\times n\times p}$, max pooling (arrow down) is used to map $\mathbf{h}_{t-1}$ to the resolution of $\mathbf{o}_{t-1}$ and concatenate both feature maps. Afterwards, a batch normalized (BN) 1×1 convolution is performed to create an embedding. Using bilinear interpolation instead of max pooling decreased the stability of the training. Features $\mathbf{o}_t \in \mathbb{R}^{m\times n\times p}$ at the current

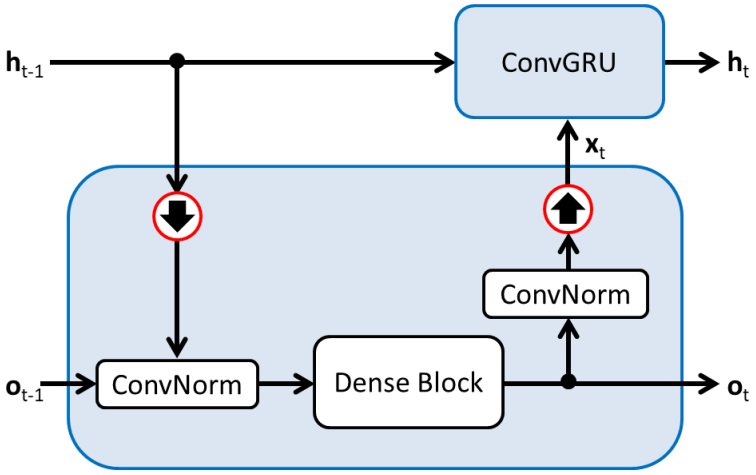


Figure 4.5: Full-Resolution Dense Unit (FRDU)

resolution are extracted by a Densely Connected block (Dense Block) [30] with k layers. By including additional skip connections, the number of parameters can be significantly reduced while increasing the depth of the network without harming gradient flow or performance. The input $\mathbf{x}_t$ of the GRU is extracted from $\mathbf{o}_t$ by performing a 1×1 convolution and nearest neighbor interpolation (arrow up) to the resolution of $\mathbf{h}$. Using bilinear interpolation yielded inferior results.

Details on the layer configuration of the GRUU-Net are provided in Table 4.1. In addition to the pooling and recurrent stream, the initial feature maps are extracted by performing a 5×5 convolution in the first layer. It has been shown that early layers benefit from negative activations of the filters [141, 12, 13]. To minimize the number of parameters while keeping the negative activations, Leaky Rectified Linear Units (LReLU) [58] (see (2.9)) is used for all non-linear layers with a leakage factor of 0.2. All filters are initialized using a scaled random normal distribution [59]. The stability of the training is increased by using reflection-padding instead of zero-padding. For computing the final prediction, a Residual Block and a 1×1 convolution for the output $\mathbf{x} \in \mathbb{R}^{m \times n \times g}$ of the recurrent stream is used followed by the softmax function to compute the pixel-wise foreground and background probability. The proposed network could be extended by using an additional class for object borders. To better focus on the improvements of the base network, these extensions were not explored, and no refinement with an additional CRF (e.g., [144]) was performed.

Focal Loss Function

The network is trained using an extension of the focal loss in [107], which was previously used for object detection in images of natural scenes using a stochastic gradient descent optimizer. The focal loss is an extension of the cross-entropy loss, which addresses very large class imbalance and performs implicit negative mining by enforcing a higher loss on uncertain predictions. In this specific application,

Table 4.1: GRUU-Net layer configuration. The superscripts denote the filter size for the convolutions and the number of layers k in the Dense blocks of the FRDU. The subscripts represent the number of output feature maps.

	$\text{conv}_{32}^{5\times5}$+BN+LReLU	
Pooling Stream	$\text{FRDU}_{32}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$
	max pooling	
	$\text{FRDU}_{64}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$
	max pooling	
	$\text{FRDU}_{128}^{3\times3,k=6}$	$\text{GRU}_{32}^{3\times3}$
	max pooling	
	$\text{FRDU}_{256}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$
	max pooling	
	$\text{FRDU}_{512}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$
	bilin. upsampling	
	$\text{FRDU}_{256}^{3\times3,k=12}$	$\text{GRU}_{32}^{3\times3}$
	bilin. upsampling	
	$\text{FRDU}_{128}^{3\times3,k=6}$	$\text{GRU}_{32}^{3\times3}$
	bilin. upsampling	
	$\text{FRDU}_{64}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$
	bilin. upsampling	
	$\text{FRDU}_{32}^{3\times3,k=3}$	$\text{GRU}_{32}^{3\times3}$
	-	Residual $\text{Block}_{32}^{3\times3}$
	-	$\text{conv}_{2}^{1\times1}$+BN
	softmax	

background pixels that are separating cells are rare compared to inner and outer pixels of cells, and can hardly be learned via a traditional cross-entropy loss. Using the focal loss relieves designing weighting functions as done in [27] and naturally generalizes too many difficult applications. The focal loss in [107] was extended by introducing a normalization and adapting it to semantic segmentation using a momentum-based optimizer. The focal loss in [107] is defined by:

$$\text{FL}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{bmng} \text{vec}(-\mathbf{w}_{\text{FL}}(\mathbf{X},\mathbf{Y}) \odot P(\mathbf{Y}) \odot \log(P(\mathbf{X})))_i \tag{4.6}$$

which is calculated pixel-wise probabilities over the vectorized (vec operator) predictions $P(\mathbf{X}) \in \mathbb{R}^{b\times m\times n\times g}$ and ground truth $P(\mathbf{Y}) \in \mathbb{R}^{b\times m\times n\times g}$, and summed up over all pixels $m \times n$, the two classes $g = 2$ (background, foreground), and the b samples

within a batch, weighted using the weights:

$$\mathbf{w}_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}) = P(\mathbf{Y}) \odot (\mathbf{1} - P(\mathbf{X}))^{\gamma} \tag{4.7}$$

where $\mathbf{1}$ denotes a tensor of ones and γ the focusing parameter, and the cross-entropy:

$$\mathrm{CE}(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{bmng} \mathrm{vec}(-P(\mathbf{Y}) \odot \log(P(\mathbf{X})))_i \tag{4.8}$$

Since scaling by $\mathbf{w}_{\mathrm{FL}}$ is equivalent to changing the learning rate, the focal loss leads to an unequal learning rate over training batches. This can be seen when inserting the focal loss FL into the equation of a standard gradient step to compute a network weight $W^t \in \mathbb{R}$ in one layer, using the learning rate l, the network prediction $P(\mathbf{X}^{t-1}) \in \mathbb{R}^{b \times m \times n \times g}$ at iteration $t-1$, and the corresponding ground truth $P(\mathbf{Y}^{t-1}) \in \mathbb{R}^{b \times m \times n \times g}$:

$$W^t \leftarrow W^{t-1} - l \, \nabla_{W^{t-1}} \left[\mathrm{FL}(P(\mathbf{X}^{t-1}), P(\mathbf{Y}^{t-1})) \right] \tag{4.9}$$

$$\begin{aligned} \mathrm{FL}(\mathbf{X}^{t-1}, \mathbf{Y}^{t-1}) &= \sum_{i=1}^{bmng} \mathrm{vec}(-\mathbf{w}_{\mathrm{FL}}(\mathbf{X}^{t-1}, \mathbf{Y}^{t-1}) \odot P(\mathbf{Y}^{t-1}) \odot \log(P(\mathbf{Y}^{t-1})))_i \\ &= \sum_{i=1}^{bmng} (-\operatorname{Diag}(\mathrm{vec}(\mathbf{w}_{\mathrm{FL}}(\mathbf{X}^{t-1}, \mathbf{Y}^{t-1}))) \\ &\qquad \mathrm{vec}(P(\mathbf{Y}^{t-1}) \odot \log(P(\mathbf{X}^{t-1}))))_i \end{aligned} \tag{4.10}$$

where the diagonal matrix $\operatorname{Diag}(\mathrm{vec}(\mathbf{w}_{\mathrm{FL}}(\mathbf{X}^{t-1}, \mathbf{Y}^{t-1})))$ performs an anisotropic scaling of the cross-entropy. Momentum-based optimizers like ADAM [55] or AMSGrad [56] use the loss to adjust the momentum and therefore the learning rate, which interferes with the scaling by the focal loss. Combining focal loss and momentum-based optimizers can therefore result in unstable training. To improve the stability during training, normalizing the weights $\mathbf{w}_{\mathrm{FL}}$ to one within a batch using the sum of all weights could be performed. Normalization of the focal loss for each image independently was less stable. The same effect can be observed for the Dice loss [72]. Incorporating an additional class weight did not improve the results in the experiments. Thus, the proposed *normalized focal loss* $\mathcal{L}(\mathbf{X}, \mathbf{Y})$ is defined by:

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{bmng} \mathrm{vec}(-\mathbf{w}_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}) \odot P(\mathbf{Y}) \odot \log(P(\mathbf{X})))_i}{\sum_{i=1}^{bmng} \mathrm{vec}(\mathbf{w}_{\mathrm{FL}}(\mathbf{X}, \mathbf{Y}))_i} \tag{4.11}$$

In all experiments, a $\gamma = 2$ as in [107] was used. By normalizing $\mathbf{w}_{\mathrm{FL}}$ to one, the trace of $\operatorname{Diag}(\mathrm{vec}(\mathbf{w}_{\mathrm{FL}}(\mathbf{X}^{t-1}, \mathbf{Y}^{t-1})))$ and thus the overall scaling remains the same in all iterations. It was found that the proposed normalized focal loss improved the

stability significantly.

Training

The datasets were augmented to increase the variability of the training data without changing the semantic information. Since some data augmentation steps are computationally demanding, a scheme for distributed data augmentation (Figure 4.6) was developed. Data augmentation is usually done on a single machine (e.g., [209, 210, 211]). When performing computationally demanding augmentation steps, the GPU cannot be fully utilized. In this thesis distributed data augmentation using multiple compute nodes, which has additional technical challenges (e.g., computation resource management, job coordination, data transfer), is being utilized instead. A single control node coordinates the data augmentation nodes, which generate augmented training data, and the training nodes, which perform the actual training. Each data augmentation node starts several threads that generate training data chunks in a fast readable binary format (TFRecord). Files are transferred to the training nodes via a shared file system and read by multiple CPU reader threads. These readers constantly transfer the data to the GPU memory to prevent the GPU from being

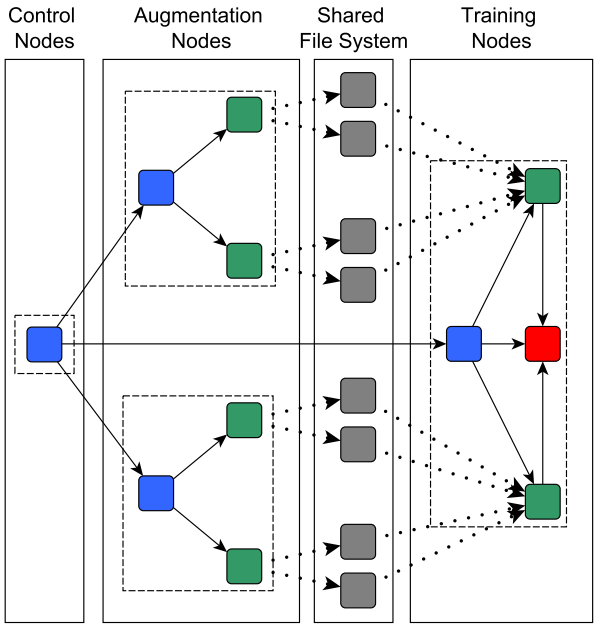


Figure 4.6: Scheme for distributed data augmentation and training. Blue boxes are CPU management processes and green boxes CPU compute threads. Grey boxes represent pre-processed files and dotted lines indicate file access. Red boxes represent GPU computations. Dashed rectangles denote compute nodes connected by threads creation (solid lines) outlining the hierarchy tree of thread forks.

idle. Separate augmentation nodes for generating training data and validation data are used. The nodes of the distributed system are connected by high throughput InfiniBand, data is stored on up-to-date SSDs, and the CPUs are fourth generation Intel Xeon CPUs. When using online multi-threaded data augmentation, a mean GPU utilization of about 60% was observed. With the proposed distributed data augmentation scheme, it was possible to increase the mean GPU utilization to 98%. Note that the performance of multi-threaded and multi-machine data augmentation strongly depends on the local hardware infrastructure. In this case, the utilized distributed system has a negligible IO overhead, which is beneficial for distributed data augmentation.

For training and validation, N_e epochs are randomly sampled from the original images and respective ground truth data. $N_e - 1$ epochs are augmented using the proposed distributed computing scheme. The last epoch is not augmented, so that the network is fine tuned to the dataset. Instead of using whole images, small crops with approximately the size of the largest object in the dataset are extracted. For the regions of interest (ROIs), the bounding box of the ground truth segmentations are used. During training, image crops from the ROIs are sampled to achieve a balance between foreground and background samples. Each crop is augmented by rotation, flipping, brightness, zoom, and elastic deformation. Augmenting by zoom and elastic deformation pose special challenges in the case of microscopy images, as altering the object structure in the ground truth can wrongly change the semantics of the training data (e.g., cell splitting). A grid-based method to perform elastic deformation is used. In this method, displacement vectors of the grid anchor points are sampled from a normal distribution. The deformed image is then generated using bicubic interpolation. To prevent merging of objects with the same label, an identity is assigned to each object in the ground truth, and data augmentation is performed. Afterwards, morphologic operations are used to ensure that previously separated objects are still separated by at least one pixel. A one-hot encoding (vector of zeros except one value) for each pixel of the crop is generated. Augmented crops exceeding the original image dimensions are filled up with reflection padding.

The network is trained using the AMSGrad optimizer in [56]. A batch size of two and an initial learning rate $l_{init} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is used. Each dataset is split into 50% for training, 25% for validation, and 25% for testing, and the network is trained using early stopping and cross-validation. The proposed model was implemented in Tensorflow [212], and an Intel i7-6700K workstation with a NVIDIA GeForce GTX 1070 Ti GPU is used.

4.4 Instance Segmentation for Cell Images

Semantic segmentation in natural images usually deals with multiple object classes but few objects with similar class. In contrast, microscopy images often contain many objects of the same class. Instance segmentation approaches solve identification

of objects by determining the object instance of a pixel in addition to the class. The segmentation models proposed in this chapter can be combined with instance segmentation approaches. However, instance segmentation approaches based on deep learning use different feature extractors and thus their performance is not directly comparable. Therefore, a comparison of four instance segmentation approaches using a consistent experimental setup is conducted. In this setup, each method is used with the same pre-processing and data augmentation pipeline and the methods are modified to use a FPN [106] for feature extraction.

A FPN [106] was used as a baseline and compared to Mask R-CNN [146], the Discriminative Loss [213], the Cosine Embedding Loss [214], and the Deep Watershed Transform [215]. Mask R-CNN uses a Faster R-CNN detection network for detecting objects which could not be easily changed to a FPN. It uses a decoder to segment the object in each detected bounding box. The Discriminative Loss and the Cosine Embedding Loss are used along with an FPN to learn a metric used in a clustering post-processing step for identifying object instances. The Discriminative Loss minimizes the Euclidean distance in a cluster of pixels denoting an object and maximizes the distance between clusters. The Cosine Embedding Loss uses the Cosine similarity instead of the Euclidean distance for comparing embeddings. Deep Watershed Transform learns the watershed transform energy of the Watershed algorithm. In contrast, to the two-stage approach in [215], a FPN is used to predict the binary segmentation mask and the watershed transform energy.

5 Hyperparameter Optimization

Automatic analysis of microscopy data typically requires complex pipelines comprising multiple methods to solve different image analysis tasks (e.g., image classification [216], cell segmentation [27], particle tracking [163], and image registration [217]). However, most methods suffer from determining application dependent hyperparameters to obtain the best performance. More generally, medical decision support systems evaluating the patient status in the clinic can highly depend on specific hyperparameters [218]. Also, the quality and performance of image-guided intervention [219] generally highly depends on hyperparameters. For complex analysis pipelines, manual optimization of hyperparameters is generally very time-consuming and difficult for a high-dimensional hyperparameter space. Thus, automated optimization is required. However, computation of the gradient of the loss function is often analytically or computationally infeasible, which prevents the use of first or higher order optimization methods. This limitation can be overcome by using zero-order optimization also known as black-box optimization [220], which does not require gradient information of the loss function. Black-box optimization uses only a limited number of evaluations (hyperparameter configurations) to determine a (local) optimum of the generally non-convex optimization problem.

In this chapter, a framework for black-box hyperparameter optimization for biomedical image analysis pipelines called HyperHyper is proposed. The work has been published in Wollmann, Ritter, et al. [11, 4]. The HyperHyper framework has several advantages compared to existing hyperparameter optimization frameworks such as Google Vizier [221], Sherpa [222], Auto-WEKA [223], Spearmint [224], and Hyperopt [225]. In Table 5.1 an overview of key features of most popular existing optimization frameworks and HyperHyper is provided. The table extends the comparison in [222] and also includes updated information about the frameworks. Existing frameworks lack certain features (e.g., modular optimizer, job wrapper, and integrated scheduler), which are essential to optimize complex image analysis pipelines using different computing paradigms and environments. The pipelines for biomedical image analysis typically include a large variety of hyperparameters, which increase the complexity of the hyperparameter space and make optimization challenging. To determine optimal solutions, the proposed HyperHyper framework employs more than 40 different optimization methods, while existing frameworks include significantly less methods (e.g., up to five methods as in Table 5.1). The high number of optimizers in HyperHyper was realized by separation of hyperparameter sampling and optimization strategy. Except Auto-Weka, all frameworks in Table 5.1 can operate in a distributed comput-

Table 5.1: Comparison of different hyperparameter optimization frameworks.

Feature	Google Vizier	Sherpa	Auto-WEKA	Spearmint	HyperOpt	HyperHyper
Number of optimization methods	3	5	1	5	5	>**40**
Modular optimizer	No	No	No	No	No	**Yes**
Job wrapper	No	No	No	No	No	**Yes**
Distributed	**Yes**	**Yes**	No	**Yes**	**Yes**	**Yes**
Integrated scheduler	No	No	No	No	**Yes**	**Yes**
Early stopping	**Yes**	**Yes**	No	No	No	**Yes**
Transfer learning	**Yes**	No	No	No	No	**Yes**
Visualization	**Yes**	**Yes**	No	No	No	**Yes**

ing environment. However, the frameworks (except HyperOpt) do not include an integrated scheduler. To optimize hyperparameters on different cluster computing infrastructures, an integrated scheduler, which is advantageous when deploying image analysis methods on heterogeneous computing infrastructures was implemented.

Overview of HyperHyper

The computing environments in the scientific community are very heterogeneous due to different computing paradigms (e.g., HPC, Cloud, Mainframe) and multiple programming languages. Moreover, the use cases for optimization of hyperparameters vary a lot. Incorporation of prior knowledge about the hyperparameters from domain knowledge or previous optimizations can significantly reduce the search space. Therefore, a hyperparameter optimization framework should be designed to be environment agnostic (e.g., programming language, compute infrastructure), extendable through modularity, and should allow incorporating prior knowledge. Moreover, flexible distribution of the computation should be supported since evaluation of hyperparameter configurations is often computational expensive. In addition, visualization of the optimization process and hyperparameter space is important to reveal insights about the optimization problem.

In this chapter, the black-box optimization framework HyperHyper for distributed computing is proposed. This framework subdivides hyperparameter optimization in a hyperparameter space definition, a general optimizer containing a hyperparameter candidate sampler and optimization strategy, and an evaluation loop (Figure 5.1).

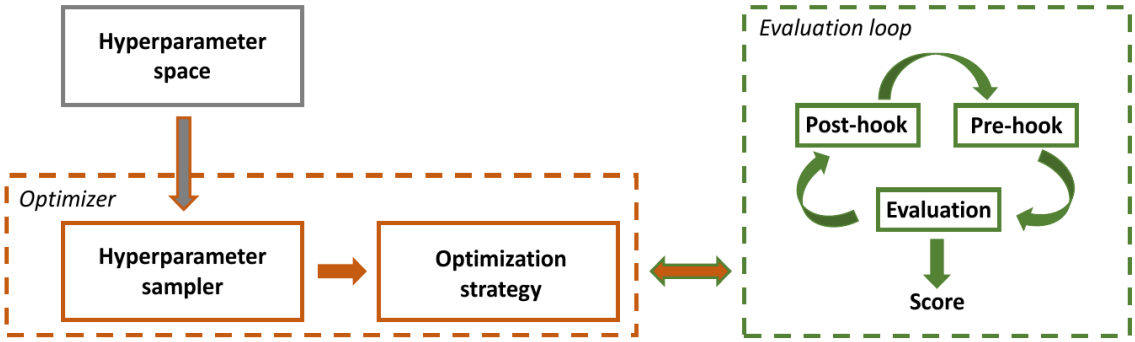


Figure 5.1: Schematic representation of HyperHyper software architecture with modular structure.

The candidate sampler and optimization strategy can be selected from a model zoo to design an optimizer for a specific application. In addition, the hyperparameter space definition incorporates prior distributions, bounds, and the sampling resolution. The candidate sampler and optimization strategy can exploit the structure of the hyperparameter space to improve convergence of the optimization. To find the global optimum, Grid Search can be used. Moreover, by design the execution of the evaluation loop can be performed highly distributed and is programming language agnostic. Modules for monitoring and visualization to analyse the optimization problem have been integrated. These visualizations including an infimum projection can reveal insights into, for example, the performance of the optimization process and the dependencies of the hyperparameters.

Optimizer

To perform optimization, constraints on the hyperparameter space have to be specified. This includes the bounds and hierarchy of each parameter, the sampling resolution, and additional prior distributions (e.g., discrete or continuous uniform, Gaussian, log Gaussian, exponential distributions). In the conducted experiments, pipelines that involve non-ordinal parameters were used. Therefore, optimizers were chosen which can handle variables without a natural order (Table 5.2). To create optimizers in HyperHyper as listed in Table 5.2, the sampling and optimization strategy can be selected from the model zoo.

The most naive optimization strategy is to perform Random Search (Random) by random sampling from the prior distributions. In Sequential Model-based Optimization (SMBO) like SMAC [226], a surrogate model is fitted to the best performing hyperparameters. SMAC with the original random forest (SMAC-RF), and with XGBoost [227] as surrogate model were investigated. It was decided to use XGBoost, since it is currently one of the most popular decision tree based models. Moreover, the Tree of Parzen Estimator (TPE), which performs a nonparametric density approximation of the best performing hyperparameter configurations [228] was in-

Table 5.2: Investigated optimizers and corresponding sampling and optimization strategies

Optimizer	Sampling strategy	Optimization strategy
Random Search	random	-
TPE	Parzen estimator	-
CMA-ES	multivariate normal	evolutionary
SMAC-RF	random	random forest
SMAC-XGBoost	random	XGBoost

vestigated. Finally, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is a generic population-based meta-heuristic based optimizer [229] was used. In CMA-ES feature sets are assumed as "genomes", which undergo evolutionary processes like selection, recombination, or mutation to increase the probability for sampling promising hyperparameter configurations.

HyperHyper evaluation loop

In the HyperHyper software architecture, the hyperparameter evaluation is decoupled from the optimization strategy. The hyperparameter evaluation is split into the pre-hook, the evaluation, and a post-hook (see Figure 5.1). The pre-hook is used for preparation of the experiment by performing a single experiment with a specific set of hyperparameters based on the hyperparameter sampling and optimization strategy. The evaluation step calculates the performance of the current hyperparameter configuration using the desired objective function. The post-hook performs clean up operations. Due to the generic formulation of the evaluation loop, any concrete implementations can be used in this plug and play system. Direct hooks for Python, job wrapper, and remote execution of workflows (e.g., Galaxy Imaging [230, 8]) were investigated. For scripts directly written in Python, entry points can be called directly by HyperHyper. A job wrapper, wraps pipelines that can be called via command line. This approach is the most generic, since it can handle arbitrary programming languages. Finally, remote execution of workflows in workflow engines like Galaxy is useful to leverage already optimized third party high performance computing (HPC) or cloud infrastructure. The execution of the loop can be distributed using a central database for coordination and workers for actual execution. Moreover, the distributed optimization process can be monitored by retrieving status information from the central database. The workers can be scheduled to available HPC or cloud infrastructure using, for example, Nextflow [231]. This approach has the benefit, that it can leverage and even combine a vast variety of schedulers or cloud systems, even at multiple sites.

6 Transfer Learning for Microscopy Image Data

6.1 Overview and Task Description

Acquisition of images at multiple imaging sites or with differing staining protocols result in significant variations in appearance. However, most deep learning methods are trained on a dataset from a subset of acquisition sites and specific staining protocols. Transfer learning can be leveraged to reuse data from different sites and protocols to reduce data requirements and improve performance.

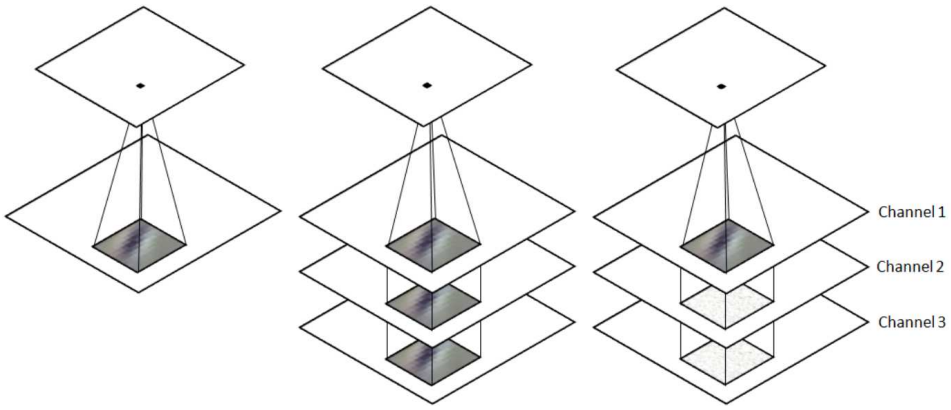
In this chapter, a novel method based on deep learning for network transfer between domains with a varying number of input color channels is presented. In addition, an unsupervised domain adaption method for end-to-end grading of whole-slide images with multiple data sources is presented. The work has been published in Wollmann et al. [13, 12].

6.2 Multi-Channel Deep Transfer Learning

It is common in image analysis of natural scenes to pre-train a deep neural network on a large dataset like ImageNet and fine-tune the network on the considered target dataset, when only a small dataset is available for training [232]. However, images of natural scenes are usually color images represented by three color channels, but microscopy images generally have a varying number of color channel, often more than three channels. For a convolutional neural network, in the first layer a filter is used for each color channel to extract corresponding feature maps. Hence, the number of channels is fixed in the network according to the considered data, and the pre-trained network cannot directly be transferred to data with a different number of channels. To cope with this, two transfer learning approaches were developed, which use only one color channel for training and perform fine-tuning on more color channels (Wollmann et al. [13]).

For transfer learning, two approaches are employed (Figure 6.1). In the first approach, an ASPP-Net trained on a dataset with one color channel is used (Figure 6.1a) for a dataset with multiple color channels by altering the first layer of the network and using the same trained convolutional filters for all channels (Figure 6.1b). This is motivated by the assumption that the trained filters are generic for different types

of images and can therefore be applied to other channels with different stainings. In the second approach, the trained convolutional filters are used for the corresponding channel in the new dataset and initialize the filters for the other channels by HE initialization [59] (Figure 6.1c). With this approach, the pre-trained filters are kept for one channel and all other filters are trained from scratch. The networks are trained using cross-validation and early stopping with the Adam optimizer and a learning rate of $l_{init} = 0.001$ as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The dataset is always split into 50 % training, 25 % validation, and 25 % testing data. Datasets are augmented using random flipping, rotation, cropping (200×200 pixels), color shift, and elastic deformations.



(a) Original filter trained on one channel (b) Same trained filter used for all channels (c) Individual trained filter for one channel

Figure 6.1: Different approaches for transfer learning.

6.3 Unsupervised Domain Adaption for End-to-End Grading of Whole-Slide Images

In this section, the grading of lymph node metastases in histopathology whole-slide images (WSIs) is considered. In order to grade the progression of cancers, the TNM system is used [233]. In the TNM classification system, the parameter T describes the size and tissue invasion status of the primary tumor. The parameter N reflects the degree of cancer spread to regional lymph nodes. Metastasis developments are graded by the parameter M. The proposed method focuses on determining the parameter N for grading cancer spread into regional lymph nodes. Currently, pathologists perform the pathological N-stage (pN-stage) grading manually, which is tedious, time-consuming, and error-prone. Automation of this process, or at least semi-automated assistance, could reduce manual work and errors. In recent years, a number of methods for analyzing WSIs have been introduced, and challenges comparing these methods have been carried out (e.g., [183, 234, 5]). Most methods

use a sliding window approach for dense classification of WSIs, which is, however, relatively slow and requires processing of many image regions [235, 236, 124]. Sparse selection of regions of interest can significantly speed up the classification [17]. This is important for performing the image analysis on a workstation to support the decision process of a pathologist. Unfortunately, sparse classification generally reduces the classification performance. However, this can be alleviated by model averaging [237]. More importantly, previous classification methods (e.g., CAMELYON17 challenge) require pixel level annotations for training (e.g., [124]). Generating such annotations is highly time-consuming and difficult. In addition, one has to cope with the large variation of WSIs from different data sources (e.g., medical centers). A promising approach to tackle this challenge is data normalization. In [124, 238] an unsupervised clustering approach was used for normalization of different data in color space. However, such approaches are heuristically designed based on a priori knowledge about the domains. An unsupervised neural network method for learning nonlinear transformations could reveal complex hidden properties to improve the classification result. In [239] a neural network with an adversarial loss and a classification network (DANN) were used to enforce domain invariant features for mitotic cell detection. There, the number of domains is equal to the number of classes, and labeled data from all considered domains is required. Separating domain adaptation from the classifier would enable reusing the classifier with new unlabelled data. Therefore, a new deep learning method for sparse classification of WSIs and automatic breast cancer grading is introduced. The method was published in Wollmann et al. [12].

Compared to previous methods based on pixel level annotations for training, the proposed method uses end-to-end learning and requires only slide level annotations. A Cycle-Consistent Generative Adversarial Network (CycleGAN) [240] is combined with a densely connected deep neural network (DenseNet) [30]. The latter type of network recently showed outstanding results for natural images. The CycleGAN enables unpaired domain adaptation to transfer the appearance of data from one source to another source (e.g., different medical centers) in an unsupervised manner. In the proposed method, domain adaptation by CycleGAN is separated from the classifier. Thus, domain adaption and classification can be trained independently, and labeled data from only one source is required. The trained transfer networks are used for domain adaptation and data augmentation. The proposed method was evaluated on the challenging CAMELYON17 dataset [241, 216]. It turned out that domain adaptation improves the classification result compared to state-of-the-art data augmentation.

In the proposed method, first a region of interest is automatically selected by color thresholding and then classified by a densely connected deep neural network. The classification results are used to determine a slide level class and are further aggregated to predict a patient level grade. The network is trained using domain adaptation by a CycleGAN. Figure 6.2 illustrates the overall workflow of the proposed

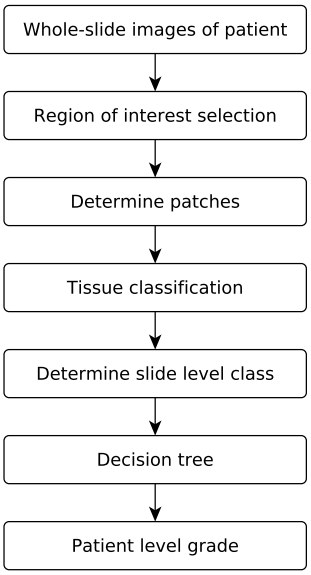


Figure 6.2: Overall workflow of the proposed method.

method.

Region of Interest Selection

Regions of interest (ROIs) within a WSI are determined by color thresholding, similarly to the method in [17]. Thresholding is performed on ratio images using the intensities of the green and red channels. Afterwards, a median filter with a disk-shaped structuring element with a window size of 50 pixels is applied. Based on the WSI with a downsampling factor of 64 as input, a ROI map is computed. For tissue classification, 20 image patches with a size of 512×512 pixels are extracted from a 64 times downsampled WSI using the ROI map. The downsampling factor was chosen so that metastases are visible. Since the WSIs have different spatial resolutions, they are normalized using bilinear interpolation, so that a pixel in all training images has the same spatial resolution of $0.24\mu m \times 0.24\mu m$ prior to downsampling.

Domain Adaptation

A CycleGAN is used for domain adaptation and augmentation of the data. WSIs from different data sources (e.g., different medical centers) typically have quite different appearances due to different imaging techniques (e.g., different stainings and different slide-scanner models). This significantly increases the difficulty of classification. In the application, paired examples to learn a direct transformation between data from different sources are not available, i.e. no data of the same sample imaged at different medical centers is available. However, CycleGANs can be trained with unpaired examples. A CycleGAN for each combination of data sources is trained for 100

epochs. The models are trained using the Adam optimizer with an initial learning rate $l_{init} = 0.001$, as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Tissue Classification

Tissue classification is performed on image patches determined by ROI selection. All patches are classified into a one-hot encoding (vector of zeros except one value) of four classes. The first class consists of WSIs containing isolated tumour cells (ITC). The second class includes WSIs with macro metastases, and the third class comprises WSIs with micro-metastases. The fourth class consists of WSIs that contain no cancerous cells (negative). Model averaging is conducted by applying online data augmentation (during training) using random rotations and flipping of the patches, and subsequent averaging of the classification results. Thus, slide level classes are determined by summing up the class activations of the last layer of the proposed network and calculating the maximum over all 20 extracted image patches. Besides using the maximum, an approach, which ranks the classes (by macro metastasis, micro metastasis, ITC, and negative) and selects the highest ranked class, was also investigated. However, the latter approach did not improve the classification result.

For classification, a method based on a DenseNet [30] was used. Compared to ReLUs, PReLUs have the advantage that they allow negative activations, which prevents discarding the information of negatively activated filters. However, the computation time is significantly higher. It was observed that lower layers particularly prefer negative activations, thus PReLU is used in the first layer and ReLU in all other layers. This enables improved feature extraction and only somewhat increases the computation time. In total, 32 feature maps are extracted in the first layer. In addition, the feature maps are increased in every downsampling block by a factor of two. Afterwards a dense block is used, since due to downsampling in the proposed network, objects of interest are small. A downsampling block is employed twice followed by dense blocks with feature map growth rates of 3, 6, and 4. Finally, global average pooling (pooling kernel equal to feature maps) is used and a dense layer is applied to determine the prediction.

Patient Level Grading

A patient level grade is determined based on the slide level classification (several WSIs correspond to one patient). The following decision rules provided in the CAMELYON17 challenge [241, 216] are used:

- **pN0:** No micro-metastases, macro-metastases, or ITCs found.
- **pN0(i+):** Only ITCs found.
- **pN1mi:** Micro-metastases found, but no macro-metastases.

- **pN1:** Metastases found in 1-3 lymph nodes, of which at least one is a macro-metastasis.
- **pN2:** Metastases found in 4-9 lymph nodes, of which at least one is a macro-metastasis.

Model Training

The cross-entropy loss function is used to train the deep neural network model. The network is trained on 50 % of the data (computation time of about two hours) and kept 25 % of the data for validation and 25 % for testing the model. In each epoch the class occurrences are balanced and augmentation of the image patches is performed.

The tissue classification network is trained using mini-batches containing 10 image patches each. The image patches of size 512×512 pixels are extracted from the downsampled WSIs using color thresholding, augmented using random rotation and flipping, and passed to the deep neural network. Data augmentation and transfer to the GPU node are optimized using multi-threaded data streaming jobs running on a CPU cluster.

7 Experimental Results

Extensive qualitative and quantitative evaluations of the methods presented in Chapters 3, 4, 5, and 6 were conducted. In this chapter, the experimental results are presented. The methods are benchmarked on a large variety of microscopy imaging modalities and datasets. The experimental results show that the proposed methods are competitive or outperform state-of-the-art methods. Some of the proposed methods achieved top ranks in international challenges in biomedical computer vision.

7.1 Detection in Microscopy Images

The detection methods proposed in Chapter 3 were benchmarked against state-of-the-art methods. In this section, comprehensive experiments are presented where the performance of the algorithms was quantified using the following evaluation measures:

F1: The F1 score is the harmonic mean of precision and recall and measures the similarity of two paired sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}|$ and $|\mathbf{Y}|$ are the cardinalities of the sets:

$$\begin{aligned} \mathrm{F1}(\mathbf{X},\mathbf{Y}) &= \left(\frac{(\mathrm{precision}(\mathbf{X},\mathbf{Y}))^{-1} + (\mathrm{recall}(\mathbf{X},\mathbf{Y}))^{-1}}{2} \right)^{-1} \\ &= \frac{2\,\mathrm{precision}(\mathbf{X},\mathbf{Y}) \cdot \mathrm{recall}(\mathbf{X},\mathbf{Y})}{\mathrm{precision}(\mathbf{X},\mathbf{Y}) + \mathrm{recall}(\mathbf{X},\mathbf{Y})} = \frac{2|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X}| + |\mathbf{Y}|} \end{aligned} \tag{7.1}$$

RMSE: The root-mean-squared error measures the spatial deviation of the paired sets of vectors $\mathbf{X}$ and $\mathbf{Y}$ using the square root of the second sample moment of their differences, where $\mathbb{E}$ denotes the expected value operator:

$$\mathrm{RMSE}(\mathbf{X},\mathbf{Y}) = \sqrt{\mathbb{E}[(\mathbf{X}-\mathbf{Y})^2]} \tag{7.2}$$

Matching pairs of ground truth and prediction detections is performed either with nearest neighbor or the Munkres algorithm [242] and a gating distance.

7.1.1 DetNet: Deep Neural Network for Particle Detection in Fluorescence Microscopy Images

The DetNet method presented in Section 3.2 has been benchmarked using data from the Particle Tracking Challenge including particles with different shapes (round and elongated). In addition, DetNet was evaluated on live cell fluorescence microscopy data of fluorescently labeled hepatitis C virus (HCV) proteins. The data is very challenging due to low and different SNR levels and bleaching as well as different particle sizes and shapes.

The performance of DetNet has been assessed using data from the Particle Tracking Challenge and a comparison with the Spot-Enhancing Filter (SEF) [161] and the H-Dome transform [159] was performed. SEF consists of applying a Laplacian-of-Gaussian filter (LoG) with standard deviation σ_{LoG}, followed by thresholding the filtered image to detect particles. H-Dome also uses an LoG filter followed by a H-Dome transform [243] and thresholding the transformed image.

The detection and localization performance of DetNet, SEF, and H-Dome has been evaluated for all 2D scenarios of the challenge comprising round shaped vesicles and receptors as well as elongated microtubules. SNR levels from SNR=1 to SNR=7 and different object densities ranging from low to high particle density have been used. In total, 3,600 images with size 512×512 pixels have been employed and the data in each category was evaluated with a random split of 50% for training, 25% for validation, and 25% for testing. For the detection performance, the mean F1 score $\in [0, 1]$ has been computed for each image sequence. For the localization performance, the mean root mean square error (RMSE) between the individually assigned detections and ground truth positions has been computed for each image sequence. The assignment between particle detections and ground truth was determined by the Munkres algorithm [242] with a maximal gating distance of five pixels.

The obtained performance values for all scenarios and all SNR levels are provided in Table 7.1. For almost all scenarios and SNR levels, DetNet outperforms SEF and H-Dome. For all scenarios with SNR=4 and 7, DetNet yields a mean F1 score higher than 0.95. For a lower SNR level of SNR=2, the mean F1 score is still always higher than 0.8. A crucial parameter of DetNet is the shift a of the sigmoid activation in the last layer (see (3) above). Table 7.2 shows the result for DetNet with and without optimized shift a as well as in comparison to SEF and H-Dome for vesicle data with SNR=1, where the best results are highlighted in bold. It can be seen that DetNet with optimized sigmoid shift yields a much better detection and localization performance than DetNet with a fixed non-optimized sigmoid shift ($a = 0.5$), and outperforms SEF and H-Dome.

Table 7.1: Performance for the Particle Tracking Challenge data (mean ± standard deviation).

Scenario	SNR	Method	F1	RMSE
Microtubule	1	SEF	0.293 ± 0.072	2.955 ± 0.140
		H-Dome	0.129 ± 0.070	3.851 ± 0.217
		DetNet	$\mathbf{0.481 \pm 0.107}$	$\mathbf{2.419 \pm 0.195}$
	2	SEF	0.447 ± 0.051	2.941 ± 0.106
		H-Dome	0.159 ± 0.074	4.181 ± 0.282
		DetNet	$\mathbf{0.819 \pm 0.035}$	$\mathbf{1.310 \pm 0.150}$
	4	SEF	0.518 ± 0.098	2.880 ± 0.100
		H-Dome	0.350 ± 0.062	4.221 ± 0.110
		DetNet	$\mathbf{0.964 \pm 0.020}$	$\mathbf{0.550 \pm 0.087}$
	7	SEF	0.524 ± 0.100	2.855 ± 0.095
		H-Dome	0.416 ± 0.208	4.265 ± 0.265
		DetNet	$\mathbf{0.977 \pm 0.017}$	$\mathbf{0.411 \pm 0.094}$
Receptor	1	SEF	0.170 ± 0.068	1.959 ± 0.246
		H-Dome	0.147 ± 0.083	3.624 ± 0.270
		DetNet	$\mathbf{0.255 \pm 0.124}$	$\mathbf{1.789 \pm 0.445}$
	2	SEF	0.429 ± 0.149	1.033 ± 0.235
		H-Dome	0.186 ± 0.088	3.741 ± 0.276
		DetNet	$\mathbf{0.802 \pm 0.076}$	$\mathbf{0.693 \pm 0.078}$
	4	SEF	0.673 ± 0.023	0.497 ± 0.123
		H-Dome	0.351 ± 0.077	3.512 ± 0.158
		DetNet	$\mathbf{0.978 \pm 0.017}$	$\mathbf{0.415 \pm 0.069}$
	7	SEF	0.682 ± 0.010	$\mathbf{0.413 \pm 0.146}$
		H-Dome	0.557 ± 0.058	3.777 ± 0.218
		DetNet	$\mathbf{0.974 \pm 0.019}$	0.440 ± 0.082
Vesicle	1	SEF	0.257 ± 0.078	1.904 ± 0.187
		H-Dome	0.108 ± 0.056	3.782 ± 0.283
		DetNet	$\mathbf{0.423 \pm 0.127}$	$\mathbf{1.857 \pm 0.193}$
	2	SEF	0.577 ± 0.031	1.200 ± 0.121
		H-Dome	0.145 ± 0.075	3.913 ± 0.322
		DetNet	$\mathbf{0.939 \pm 0.022}$	$\mathbf{0.766 \pm 0.080}$
	4	SEF	0.686 ± 0.015	0.610 ± 0.158
		H-Dome	0.354 ± 0.117	3.686 ± 0.107
		DetNet	$\mathbf{0.977 \pm 0.016}$	$\mathbf{0.459 \pm 0.073}$
	7	SEF	0.688 ± 0.011	0.527 ± 0.124
		H-Dome	0.605 ± 0.104	3.751 ± 0.277
		DetNet	$\mathbf{0.976 \pm 0.016}$	$\mathbf{0.427 \pm 0.090}$

Table 7.2: Impact of sigmoid shift a (vesicle data, SNR=1).

Method	F1	RMSE
SEF	0.257 ± 0.078	1.904 ± 0.187
H-Dome	0.108 ± 0.056	3.782 ± 0.283
DetNet ($a = 0.5$)	0.106 ± 0.088	2.402 ± 1.365
DetNet (a optimized)	$\mathbf{0.423 \pm 0.127}$	$\mathbf{1.857 \pm 0.193}$

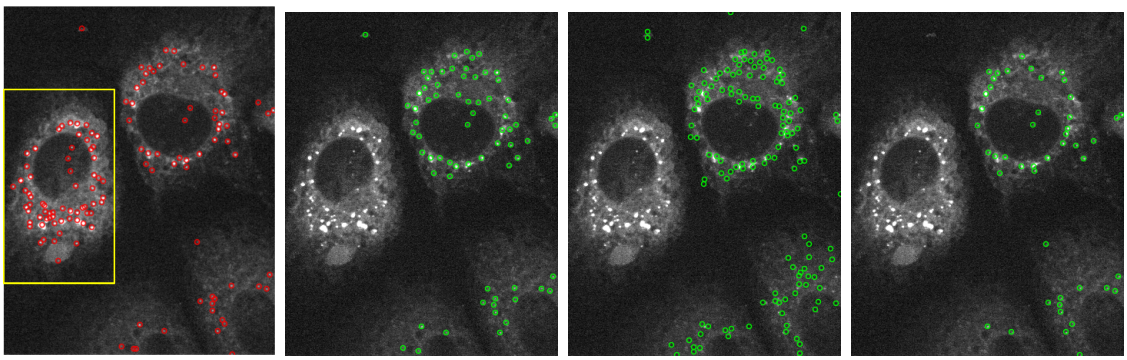
Evaluation on Live Cell Microscopy Data

DetNet was also evaluated using challenging live cell microscopy data displaying fluorescently labeled HCV proteins NS5A (Figure 7.1). The image data was acquired with an Ultra-View ERS spinning disk confocal microscope and has an image size of 355×447 pixels (Bartenschlager lab). To train DetNet, one cell of the image data with 66 ground truth annotations (yellow box in Figure 7.1a on the left) was used. For evaluation, 128 ground truth annotations of the other three cells have been employed.

The detection performance was evaluated by precision, sensitivity, and the F1 score. The localization performance was assessed using the mean RMSE between detected particles and ground truth. Detection results for DetNet, SEF, and H-Dome are shown in Figure 7.1. Quantitative performance values are provided in Table 7.3. It can be seen that DetNet yields significantly better results than SEF and H-Dome for all performance metrics.

7.1.2 Deep Residual Hough Voting for Mitotic Cell Detection in Histopathology Images

The Deep Residual Hough Voting method presented in Section 3.3 was quantitatively evaluated based on the AMIDA13 challenge dataset consisting of invasive breast carcinoma histology images [157]. In total, the dataset comprises 606 high power



(a) Ground truth (b) SEF (c) H-Dome (d) DetNet

Figure 7.1: Detection results for HCV live cell microscopy data. a) Ground truth particles indicated by red circles. The yellow box represents the training data. Detection results for b) SEF, c) H-Dome, and d) DetNet.

Table 7.3: Performance for HCV live cell microscopy data.

Method	Precision	Sensitivity	F1	RMSE
SEF	0.467	0.677	0.553	1.574 ± 0.935
H-Dome	0.140	0.387	0.205	3.953 ± 0.371
DetNet	**0.763**	**0.726**	**0.744**	$\mathbf{1.390 \pm 0.746}$

field RGB images of 23 patients. Each image has a size of 2000×2000 pixels with a spatial resolution of $0.25\,\mu m/pixel$. The task is the automatic detection of mitotic cells. To quantify the performance, the F1 score was used, and a comparison with previous methods was performed.

For training of the proposed method, the dataset was augmented by generating all permutations of flipped and rotated images. Since apoptotic and mitotic cells appear very similar, but negative examples (e.g., apoptotic cells) are not labeled in the dataset, the proposed network was trained one epoch on image patches of size 256×256 pixels containing mitotic cells only. The full dataset was processed using this classifier and a new training set was generated by extracting 256×256 pixels patches from the augmented dataset containing true-positive, false-positive, and false-negative patches. The final training was performed on 60% of the generated dataset, and model validation and selection on 20% of the generated dataset. Testing was done on the remaining 20% of the corresponding original dataset. All training steps used mini-batches of 80 samples. Maximum radius was set to 64 pixels and the determined threshold for votes was 30% of the pixels within the voting circle. The hyperparameter λ in (3.4) was set to the harmonic mean (0.5). To determine votes at image borders, the images were padded by the size of the radius and filled with a mirrored version of the image. The padded area was cropped again after voting. As an example, Figure 7.2 shows the voting result for a region of an original image. Intermediate results of the predicted radius r and the predicted angle φ are shown as well. It can be seen that the mitotic cell in the middle of the region is well extracted. Table 7.4 shows the performance of the proposed approach for all 606 images from the AMIDA13 dataset. The scores represent the performance of detecting mitotic cells. The results of the best three methods from the AMIDA13 challenge are provided for comparison. The corresponding scores are taken from the official ranking of the organizers [157]. The best results for Precision, Recall, and the F1 Score are highlighted in bold. It can be seen that the proposed method yields the best performance for Recall, and the F1 score is very similar to that of the best method from the AMIDA13 challenge [111, 157]. An advantage of the proposed

(a) Original image (b) Voting map v (c) Predicted radius values r (d) Predicted angle values φ

Figure 7.2: Example region of an original image with corresponding voting result and predictions of the radius and angle.

Table 7.4: Performance of the proposed approach and comparison with top-3 methods from AMIDA13.

Method	Precision	Recall	F1 score
IDSIA [111, 157]	**0.610**	0.612	**0.611**
DTU [244, 157]	0.427	0.555	0.483
SURREY [157]	0.357	0.332	0.344
Proposed	0.547	**0.686**	0.609

method is that it is 200 times faster than the reported computation time of previous methods in [111, 157].

Computations were performed on an Intel i7-6700K workstation with a NVIDIA Geforce GTX 970. The implementation was done in Tensorflow [212]. Final training took 2.5 days for 17300 update iterations. The computation time for an image of size 2000×2000 pixels is 2.5 seconds.

7.1.3 Grading of Whole-Slide Images based on Mitotic Cell Counts

The method presented in Section 3.4 for grading of WSIs was applied to histology images of invasive breast carcinoma from the MICCAI Tumor Proliferation Assessment Challenge 2016 (TUPAC16) [183]. The method was used to predict the proliferation score based on mitosis counting. Mitotic cell positions in images of 73 breast cancer cases were provided to train the mitotic cell detector. In addition, 500 WSIs with annotated proliferation scores were available. For testing, WSIs of 321 breast cancer cases were provided. To quantify the performance, the quadratic weighted Cohen's kappa value between the predicted and ground truth proliferation scores was used. The proposed method achieved a Cohen's kappa of 0.417 using cross-validation on the training dataset. Different automatic methods were compared in the TUPAC16 challenge [5]. LUNIT uses cell density estimation for ROI detection, a ResNet [29] with hard negative mining for mitotic cell detection, and a SVM to grading. CONTEXTVISION uses color thresholding for ROI detection [111] with hard negative mining for mitotic cell detection and heuristics for grading. HARKER uses multiple custom CNNs for ROI detection and mitotic cell detection in combination with hard negative mining and a SVM for grading. BELARUS uses [245] for ROI detection and directly performs grading by averaging the prediction of a linear classifier over 20 ROIs. RADBOUD directly performs grading by a custom CNN by averaging 500 predictions from random crops. FLORIDA uses color thresholding for ROI detection, AlexNet [1] for mitotic cell detection, and heuristics for grading. Results of the challenge for grading of whole-slide images for methods that only trained on the provided dataset are shown in Table 7.5. It can be seen that the proposed method was among the top-3 methods.

Table 7.5: Results for the methods in the TUPAC16 challenge that only used the provided training dataset.

Method	weighted Cohen's kappa	98% CI
LUNIT	**0.567**	[0.454, 0.671]
CONTEXTVISION	0.534	[0.422, 0.646]
Proposed	0.417	[0.293, 0.540]
HARKER	0.367	[0.242, 0.492]
BELARUS	0.321	[0.190, 0.452]
RADBOUD	0.290	[0.171, 0.409]
FLORIDA	0.177	[0.052, 0.302]

7.1.4 Deep Consensus Network for Particle and Cell Detection

The Deep Consensus Network presented in Section 3.5 was applied to different types of synthetic and real datasets and performed a quantitative comparison with state-of-the-art methods. Datasets with a variety of image conditions regarding object type, object density, scale, image noise, and appearance variability were considered. Moreover, a comparison of the proposed NMI-based loss with other loss functions was performed. Assignments between detections and ground truth are determined by either the Nearest Neighbor or the Munkres algorithm [242] using a gating distance.

Evaluation of the NMI-based loss function

The robustness of the proposed NMI-based loss function $\mathcal{L}_{\text{cls}}$ in (3.41) in comparison to Cross-Entropy (CE), Weighted Cross-Entropy (WCE), Normalized Focal Loss (NFL) [3], Dice loss ($\mathcal{L}_{\text{Dice}}$) in (3.26), and MCC loss ($\mathcal{L}_{\text{MCC}}$) in (3.27a) on synthetic data is analyzed. Samples for predictions and ground truth were generated for normalized confusion matrices (3.21) at grid positions. The sampling of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) was performed 1000 times while the predictions were augmented with noise from an exponential distribution, which models the time between events in a Poisson point process. Mean and standard deviation of the metrics TP, FP, TN, and FN for each trial were calculated to examine the effect of imbalance in the confusion matrix and the robustness of the loss functions. Figure 7.3 shows the results for pairs of varying entries in the normalized confusion matrix while the other entries were fixed to 0.25. All loss functions were normalized by the mean and standard deviation for reasons of comparability. It can be seen in the "FP vs. 1-FN" plot (top left) and the "TP vs. 1-TN" plot (bottom right) that CE and NFL are agnostic to class imbalance, since they are nearly constant (values close to zero). $\mathcal{L}_{\text{Dice}}$ favours a balance between FP and FN while WCE, $\mathcal{L}_{cls}$, and $\mathcal{L}_{\text{MCC}}$ favour an imbalance between FP and FN. In all other plots the loss functions show similar behavior. Table 7.6 provides the averaged standard deviation $\sigma_{\text{a/b}}$ for the ab-th entry in the normalized confusion matrix calculated for each loss function. A low standard deviation indicates that the

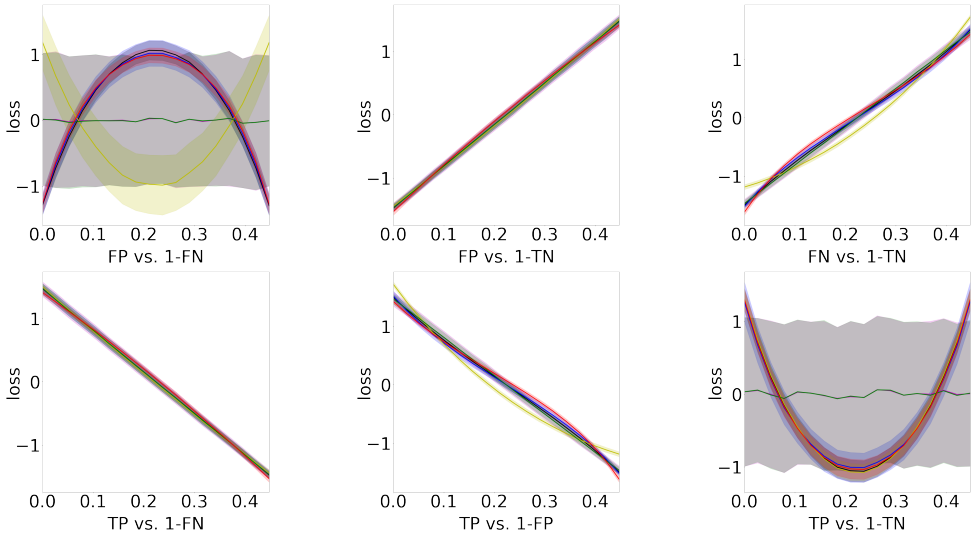


Figure 7.3: CE (green), WCE (blue), NFL (magenta), $\mathcal{L}_{\text{Dice}}$ (yellow), $\mathcal{L}_{\text{MCC}}$ (black), and $\mathcal{L}_{cls}$ (red) for prediction/ground truth sample pairs from the normalized confusion matrix. The predictions were augmented with exponential noise. This scheme was repeated 1000 times. The figure shows the mean as line and the standard deviation as colored area. For each subplot two entries of the normalized confusion matrix were changed and the other two were fixed (set to 0.25).

Table 7.6: Standard deviation of normalized performance metrics when changing two entries of the confusion matrix (cf. Figure 7.3).

Loss	$\sigma_{\text{FP/1-FN}}$	$\sigma_{\text{FP/1-TN}}$	$\sigma_{\text{FN/1-TN}}$	$\sigma_{\text{TP/1-FN}}$	$\sigma_{\text{TP/1-FP}}$	$\sigma_{\text{TP/1-TN}}$
CE	0.999	0.067	0.068	0.067	0.999	0.068
WCE	0.165	0.067	0.057	0.067	0.208	0.058
NFL	0.999	0.090	0.091	0.090	0.999	0.091
$\mathcal{L}_{\text{Dice}}$	0.404	**0.044**	0.040	**0.044**	**0.103**	0.040
$\mathcal{L}_{\text{MCC}}$	0.133	**0.044**	0.039	**0.044**	0.133	0.039
$\mathcal{L}_{cls}$	**0.121**	**0.044**	**0.037**	**0.044**	0.126	**0.037**

loss is more robust to noise in the normalized confusion matrix (noise can occur due to variability of label noise, sample difficulty, or influence of regularization across neural network training steps). It turns out that the proposed NMI-based loss $\mathcal{L}_{cls}$ yields the best overall result and is thus most robust.

Particle Tracking Challenge dataset

The performance of the proposed Deep Consensus Network was assessed for detection of biological particles using data from the Particle Tracking Challenge [163]. The high-resolution Deep Consensus Network was used and a comparison with the state-of-the-art methods SEF [161], H-Dome [159], and DetNet [10] was performed. SEF uses a Laplacian-of-Gaussian filter (LoG) with standard deviation σ_{LoG}, followed

by thresholding the filtered image. H-Dome also employs an LoG filter followed by a H-Dome transform [243] and thresholding the transformed image. DetNet is an application specific hourglass-shaped deep neural network which performs detection on large image patches without requiring a sliding window scheme. Also, DetNet uses hyperparameter optimization to improve the performance.

The detection and localization performance of the proposed method was evaluated for all 2D scenarios of the challenge (receptor, vesicle, microtubule) comprising round shaped vesicles and receptors as well as elongated microtubules. All SNR levels from SNR=1 to SNR=7 and all object densities from low to high particle density were considered. Example sections of the used image data are shown in Figure 7.4. In total, 3.600 images with size 512×512 pixels were considered and the data was evaluated in each scenario with a random split of 50% for training, 25% for validation,

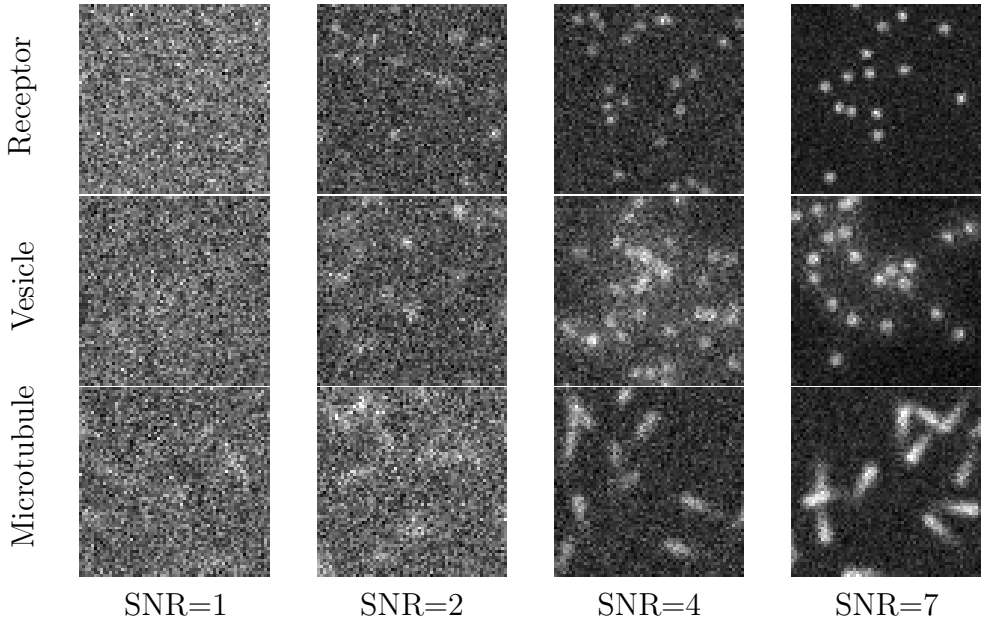
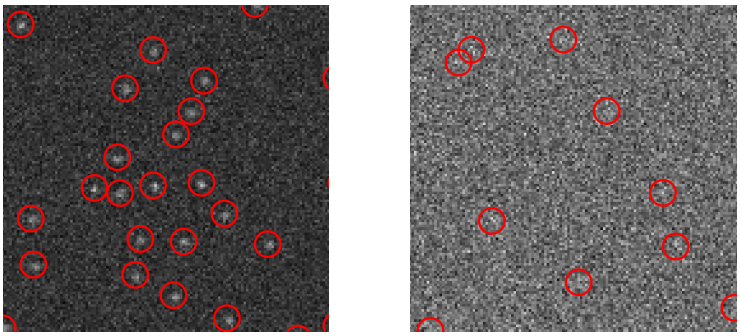


Figure 7.4: Example images showing image sections of all employed scenarios from the Particle Tracking Challenge dataset.



(a) Receptor, SNR=4 (b) Vesicle, SNR=1

Figure 7.5: Example detection results of the Deep Consensus Network for sections from the Particle Tracking Challenge dataset.

Table 7.7: Performance of different detection methods for the Particle Tracking Challenge receptor data.

SNR	Method	F1	RMSE
1	SEF	0.170 ± 0.068	1.959 ± 0.246
	H-Dome	0.147 ± 0.083	3.624 ± 0.270
	DetNet	0.255 ± 0.124	1.789 ± 0.445
	Deep Consensus Network	$\mathbf{0.296 \pm 0.105}$	$\mathbf{1.396 \pm 0.149}$
2	SEF	0.429 ± 0.149	1.033 ± 0.235
	H-Dome	0.186 ± 0.088	3.741 ± 0.276
	DetNet	0.802 ± 0.076	$\mathbf{0.693 \pm 0.078}$
	Deep Consensus Network	$\mathbf{0.822 \pm 0.063}$	0.788 ± 0.113
4	SEF	0.673 ± 0.023	0.497 ± 0.123
	H-Dome	0.351 ± 0.077	3.512 ± 0.158
	DetNet	0.978 ± 0.017	$\mathbf{0.415 \pm 0.069}$
	Deep Consensus Network	$\mathbf{0.993 \pm 0.006}$	0.425 ± 0.016
7	SEF	0.682 ± 0.010	0.413 ± 0.146
	H-Dome	0.557 ± 0.058	3.777 ± 0.218
	DetNet	0.974 ± 0.019	$\mathbf{0.440 \pm 0.082}$
	Deep Consensus Network	$\mathbf{0.993 \pm 0.006}$	0.492 ± 0.021

Table 7.8: Performance of different detection methods for the Particle Tracking Challenge vesicle data.

SNR	Method	F1	RMSE
1	SEF	0.257 ± 0.078	1.904 ± 0.187
	H-Dome	0.108 ± 0.056	3.782 ± 0.283
	DetNet	0.423 ± 0.127	$\mathbf{1.857 \pm 0.193}$
	Deep Consensus Network	$\mathbf{0.523 \pm 0.132}$	1.887 ± 0.121
2	SEF	0.577 ± 0.031	1.200 ± 0.121
	H-Dome	0.145 ± 0.075	3.913 ± 0.322
	DetNet	0.939 ± 0.022	$\mathbf{0.766 \pm 0.080}$
	Deep Consensus Network	$\mathbf{0.950 \pm 0.023}$	0.816 ± 0.098
4	SEF	0.686 ± 0.015	0.610 ± 0.158
	H-Dome	0.354 ± 0.117	3.686 ± 0.107
	DetNet	0.977 ± 0.016	$\mathbf{0.459 \pm 0.073}$
	Deep Consensus Network	$\mathbf{0.993 \pm 0.005}$	0.552 ± 0.043
7	SEF	0.688 ± 0.011	0.527 ± 0.124
	H-Dome	0.605 ± 0.104	3.751 ± 0.277
	DetNet	0.976 ± 0.016	0.427 ± 0.090
	Deep Consensus Network	$\mathbf{0.992 \pm 0.006}$	$\mathbf{0.353 \pm 0.016}$

and 25% for testing. For determining the detection performance, for each image sequence the mean F1 score $\in [0, 1]$ was computed using the Munkres algorithm to determine assignments with a gating distance of five pixels. For quantifying the localization performance, for each image sequence the mean root mean square error (RMSE) between the individually assigned detections and ground truth positions

Table 7.9: Performance of different detection methods for the Particle Tracking Challenge microtubule data.

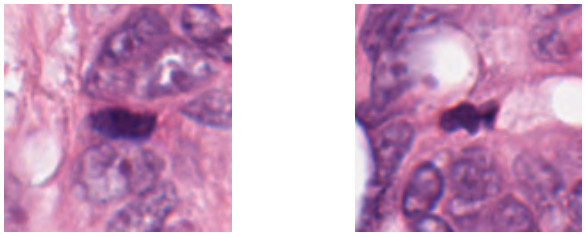
SNR	Method	F1	RMSE
1	SEF	0.293 ± 0.072	2.955 ± 0.140
	H-Dome	0.129 ± 0.070	3.851 ± 0.217
	DetNet	0.481 ± 0.107	2.419 ± 0.195
	Deep Consensus Network	$\mathbf{0.549 \pm 0.126}$	$\mathbf{2.185 \pm 0.137}$
2	SEF	0.447 ± 0.051	2.941 ± 0.106
	H-Dome	0.159 ± 0.074	4.181 ± 0.282
	DetNet	0.819 ± 0.035	$\mathbf{1.310 \pm 0.150}$
	Deep Consensus Network	$\mathbf{0.829 \pm 0.019}$	1.354 ± 0.148
4	SEF	0.518 ± 0.098	2.880 ± 0.100
	H-Dome	0.350 ± 0.062	4.221 ± 0.110
	DetNet	0.964 ± 0.020	$\mathbf{0.550 \pm 0.087}$
	Deep Consensus Network	$\mathbf{0.972 \pm 0.018}$	0.679 ± 0.085
7	SEF	0.524 ± 0.100	2.855 ± 0.095
	H-Dome	0.416 ± 0.208	4.265 ± 0.265
	DetNet	0.977 ± 0.017	$\mathbf{0.411 \pm 0.094}$
	Deep Consensus Network	$\mathbf{0.980 \pm 0.014}$	0.526 ± 0.037

was calculated. The results for the different scenarios are presented in Tables 7.7, 7.8, and 7.9. Example detection results of the Deep Consensus Network are shown in Figure 7.5.

It can be seen that the proposed Deep Consensus Network yields the best F1 score for all datasets and the best RMSE for 4 out of 12 datasets. Thus, the detection performance of the Deep Consensus Network is superior to the previous methods. Concerning the localization performance in terms of RMSE, DetNet is often the best, but the difference between Deep Consensus Network and DetNet is relatively small. Note that DetNet is an application specific network for particle detection while Deep Consensus Network is a general network and applicable to a wide spectrum of data. Figure 7.5 shows that the Deep Consensus Network can still detect objects if they are hardly visible for a human observer.

Histopathological TUPAC16 challenge dataset

The proposed Deep Consensus Network was also evaluated based on the TUPAC16 challenge dataset consisting of invasive breast carcinoma histopathological images [5]. This dataset is very different from the Particle Tracking Challenge dataset considered in Section 7.1.4 above. In total, the dataset comprises 606 high power field RGB images of 73 patients from three pathology centers. The dataset includes 23 cases from the AMIDA13 challenge [157] and 50 new cases. The images were acquired with a magnification of 40x and a spatial resolution of $0.25\,\mu m/pixel$, and consist of up to 5657×5657 pixels. A main task is the automatic detection of mitotic cells which is important for grading breast cancer tissue. Example image sections



(a) Mitotic cell (b) Non-mitotic cell

Figure 7.6: Example images showing hard to distinguish mitotic and non-mitotic cells from the TUPAC16 challenge dataset.

with hard to distinguish mitotic and non-mitotic cells are shown in Figure 7.6. To quantify the performance, the F1 score with Nearest Neighbor assignment and a gating distance of 30 pixels was used, as in TUPAC16 [157, 5]. A comparison with the fully automatic methods in the TUPAC16 challenge was conducted, namely the methods of Lunit Inc., Heidelberg University (Deep Hough Voting) [16], Pakistan Institute of Engineering and Applied Sciences, University of South Florida, University of Warwick, Shiraz University of Technology, Inha University, Instituto Politécnico Nacional, and Healthcare Technology Innovation Centre IIT Madras [5]. For a fair comparison, methods that use additional data for training or model ensembling to boost their performance (e.g., Contextvision and Radboud [5]) were excluded. The methods of Lunit Inc., Heidelberg University, University of South Florida, and University of Warwick, as well as the proposed Deep Consensus Network, are based on a deep neural network to detect mitosis along with hard negative mining. Note that the methods of Lunit Inc., University of South Florida, and University of Warwick perform pixel-wise classification using a sliding window scheme. In contrast, the proposed Deep Consensus Network does not require a sliding window scheme, and is thus more general. The proposed network is based on *centroid-based* object detection and allows processing of large image patches, which makes inference much faster. In the Deep Hough Voting method, multiple detection proposals are aggregated in a post-processing step using the generalized Hough transform. In contrast, Deep Consensus Network does not require a post-processing step, can be trained end-to-end, and directly predicts the final detections. In addition, the proposed method was compared with the deep neural network DetNet (see Section 3.2), which showed promising results for particle detection. DetNet is an application specific deep neural network for particle detection based on a Deconvolution Network (see also Section 7.1.4). The proposed Deep Consensus Network was applied without and with anchor regularization (see Section 3.5). The results are presented in Table 7.10. It can be seen that Deep Consensus Network is among the top-3 methods and yields similar results as the application specific network Deep Hough Voting. It also turns out that the anchor regularization of the Deep Consensus Network improves the result. An example detection result of the Deep Consensus Network is shown in Figure 7.7, for which all detections are correct. It can also be seen that the object density (mitotic

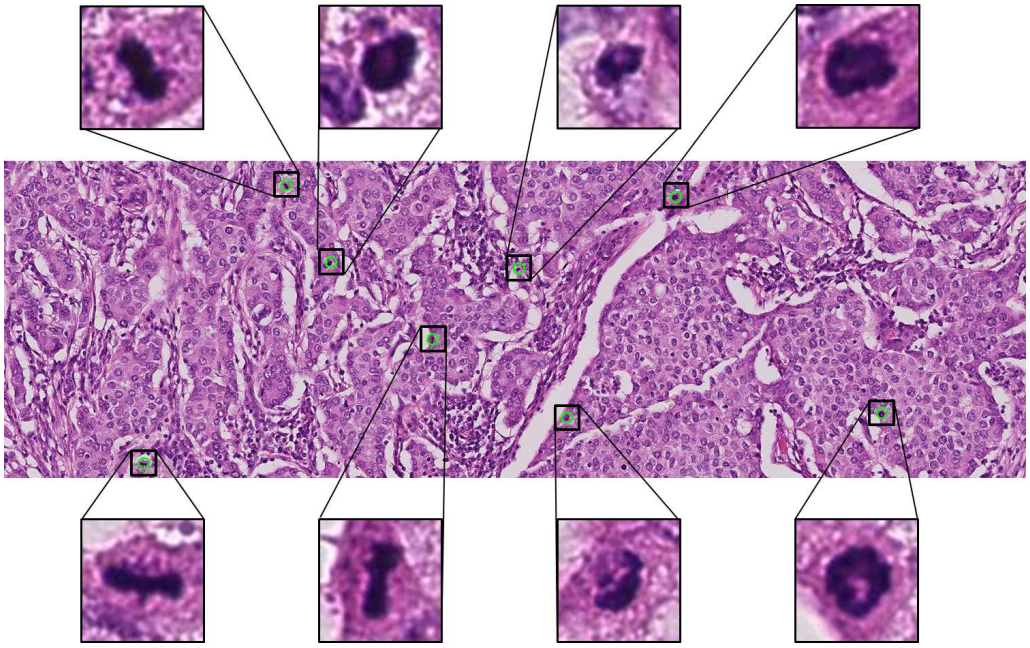


Figure 7.7: Example detection result of Deep Consensus Network for a section of 5657×3880 pixels from the TUPAC16 challenge dataset.

Table 7.10: Performance of the proposed approach and comparison with previously proposed methods and top performer from TUPAC16.

Method	F1 score
Lunit	**0.652**
Heidelberg (Deep Hough Voting)	**0.481**
Florida	0.440
Pakistan	0.424
Warwick	0.396
Shiraz	0.330
Inha	0.251
Instituto Politécnico Nacional	0.135
IIT Madras	0.017
DetNet	0.136
Deep Consensus Network (no anchor regularization)	0.421
Deep Consensus Network	**0.470**

cells) is very low, while for the Particle Tracking Challenge data (Section 7.1.4) the object density is high. Thus, the proposed method can cope with both low and high object density, and very different types of objects and image data.

7.2 Segmentation of Microscopy Images

The segmentation methods proposed in Chapter 4 were benchmarked against state-of-the-art methods. In this section, comprehensive experiments are presented where

the performance was quantified using the evaluation measures:

Dice: The Sørensen-Dice coefficient measures the similarity of two sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}|$ and $|\mathbf{Y}|$ are the cardinalities of the sets:

$$\text{Dice}(\mathbf{X}, \mathbf{Y}) = \frac{2|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X}| + |\mathbf{Y}|} \tag{7.3}$$

JI: The Jaccard index measures the similarity of two sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}|$ and $|\mathbf{Y}|$ are the cardinalities of the sets:

$$\text{Jaccard}(\mathbf{X}, \mathbf{Y}) = \frac{|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X} \cup \mathbf{Y}|} \tag{7.4}$$

SEG: The object-wise Jaccard similarity index measures the Jaccard similarity index of two matching objects [127]. An object in the two sets $\mathbf{X}$ and $\mathbf{Y}$ is matched if the overlap is more than 50%. Objects consisting of just one pixel are discarded.

HD: The Hausdorff distance measures the maximum occurring Euclidean distance d between two sets $\mathbf{X}$ and $\mathbf{Y}$:

$$\text{Hausdorff}(\mathbf{X}, \mathbf{Y}) = \max(\sup_{x \in \mathbf{X}} \inf_{y \in \mathbf{Y}} d(x, y), \sup_{y \in \mathbf{Y}} \inf_{x \in \mathbf{X}} d(x, y)) \tag{7.5}$$

Warping Error: The Warping Error [246] is the minimum mean square error between pixels of the segmentation and pixels of the topology-preserving warped ground truth. All performance measures are calculated for each image and averaged over the whole dataset.

RI: The Rand index measures the pixel-wise prediction accuracy of two sets $\mathbf{X}$ and $\mathbf{Y}$, where $|\mathbf{X}|$ and $|\mathbf{Y}|$ are the cardinalities, and $\mathbf{X}^c$ and $\mathbf{Y}^c$ the complements of the sets:

$$\text{RI} = \frac{|\mathbf{X} \cap \mathbf{Y}| + |\mathbf{X}^c \cap \mathbf{Y}^c|}{|\mathbf{X}| + |\mathbf{Y}| + |\mathbf{X}^c| + |\mathbf{Y}^c|} \tag{7.6}$$

7.2.1 ASPP-Net for Cell Segmentation

The performance of ASPP-Net described in Section 4.2 as well as different methods for segmenting nuclei from tissue microscopy images of glioblastoma cells was investigated.

In previous work, different comparisons of methods for cell segmentation in fluorescence microscopy images were performed. Dima et al. [247] compared different segmentation methods using fluorescence microscopy data from two cell lines. The study revealed that K-means clustering yielded the best results, however, the used

data does not seem to be very difficult and machine learning methods were not considered. Coelho et al. [248] evaluated segmentation algorithms using hand-labeled datasets including clustered nuclei. An approach based on merging multiple regions from watershed segmentation performed best, however, the focus of the study was on methods for high-throughput settings, and therefore, complex and time-intensive methods were not included. Cheng and Rajapakse [249] presented a method to segment and separate clustered nuclei using shape markers in a watershed-like algorithm. The method was applied to noisy neuronal cell images. However, the data they used is much less difficult compared to the data used in this study, particularly the image contrast at edges is much better. In Maška et al. [127], different methods for cell tracking were compared. The focus was on tracking, but the segmentation performance was also quantified. Different types of data were considered compared to this study (e.g., human breast carcinoma cells, mouse embryonic stem cells, human squamous lung carcinoma cells).

In this section, nine different methods comprising thresholding, deformable models, region growing, unsupervised learning, and supervised learning for segmentation of glioblastoma cells in tissue microscopy images were studied:

Global and local thresholding: Global thresholding based on maximum entropy was used and for local thresholding the intensity contrast was employed.

Fast marching and region competition: The image was first smoothed by a Gaussian filter and intensity maxima were used as seed points. For the deformable model in the fast marching method [250]. For the region competition method [251], local intensity maxima were used for initialization of a model based on a piece-wise constant energy function. To cope with changes of the topology, creation of handles and fission was utilized.

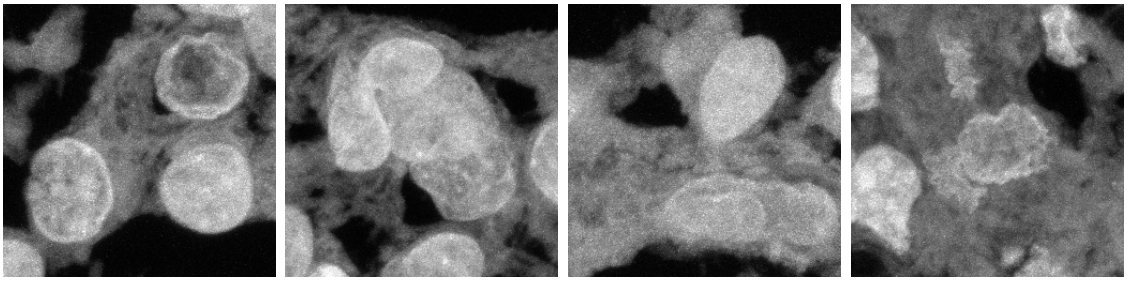
Unsupervised learning by K-means clustering: The image was smoothed by a Gaussian filter and the number of clusters for K-means clustering was set to three (foreground, background, unspecific signal). The foreground cluster was used as segmentation result.

Supervised learning using Weka and Ilastik: A random forest classifier containing 200 trees with unlimited depth from Weka [252] was used. Feature selection resulted in the following features: Gaussian blur, Hessian, Membrane projections, Mean, Maximum, Anisotropic diffusion, Gabor, Laplacian, Entropy, Variance, Minimum, Median, Bilateral, Kuwahara, Structure, and Neighbors. In addition, a random forest classifier from Ilastik [253] was used. All features provided by Ilastik were used. Both classifiers (Weka, Ilastik) were trained using 20 images.

Deep learning: Two deep neural networks based on U-Net [27] were used. The first network is a U-Net with additional batch normalization and residual blocks. The second network is the ASPP-Net described in Section 4.2 which combines a U-Net with atrous spatial pyramid pooling (ASPP) [28]. Data augmentation was performed using random flipping, rotation, scaling, noise addition, and edge-aware elastic deformation. The network was trained on 128×128 pixel patches. Patches were sampled to have approximately a balanced ratio of foreground and background pixels. The network was trained using cross-validation and early stopping with the Adam optimizer.

Multiple well-established pixel-based performance measures, namely the Dice coefficient (Dice), Jaccard index (JI), Rand index (RI), Hausdorff distance (HD), sensitivity (Sens.), specificity (Spec.), and accuracy (Acc.), were employed to compare the segmentation results with ground truth data. In addition, object-based performance measures were determined. The Jaccard index (JI) was used at object level and also determined the number of missing, erroneously added, and incorrectly merged objects.

All nine segmentation methods have been applied to 50 fluorescence microscopy tissue images of glioblastoma cells. The images were acquired, using a Leica TCS SP5 point scanning confocal microscope with a 63x objective lens (Erfle lab). The voxel size was 100 nm in the xy-plane and 250 nm in z-direction. 45 axial layers were acquired for each stack of the DAPI channel by exciting with a violet (405 nm) laser. As can be seen in Figure 7.8, this data is very challenging due to strong intensity variation, cell clustering, overlapping cells, poor edge information, missing object borders, strong shape variation, and low signal-to-noise ratio. For segmentation, MIP images of the stacks were used. Example images are shown in Figure 7.8. The dataset was split into 30 images for hyperparameter optimization and 20 images for performance evaluation. Example segmentation results of the investigated methods are shown in Figure 7.9. It can be seen that only ASPP-Net achieved a clear



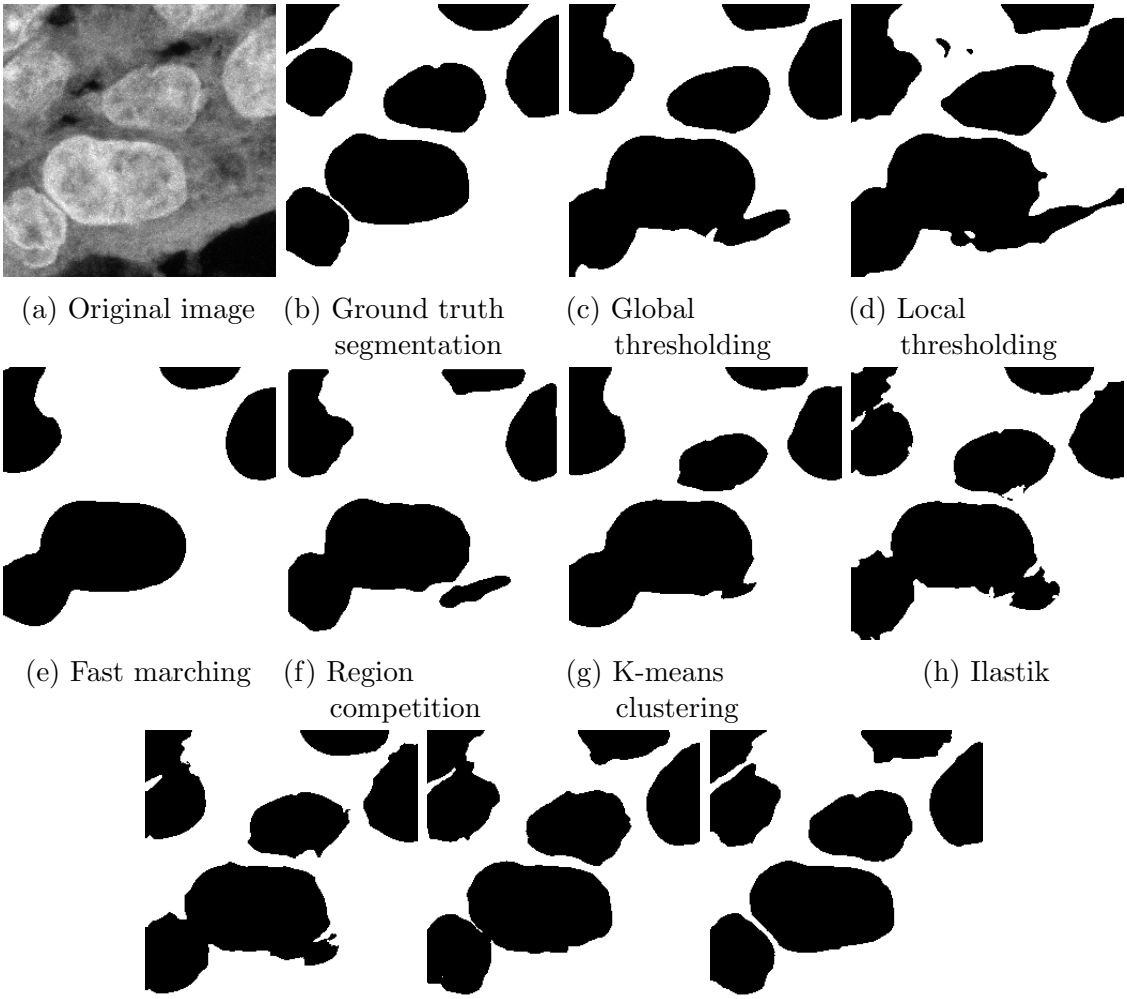
(a) Strong intensity variation (b) Overlapping cells (c) Poor edge information (d) Strong shape variation

Figure 7.8: Examples of tissue microscopy images of glioblastoma cells with different challenges for image analysis.

separation of close-by objects.

Quantitative results of all nine segmentation methods using pixel-based and object-based performance measures are provided in Table 7.11 and Table 7.12. For the pixel-based metrics, it can be seen that the two thresholding methods generally yielded the lowest values. For the machine learning methods, better results for specificity and sensitivity were obtained. Fast marching yielded the best result for HD. ASPP-Net achieved the best values for Dice (0.925), Jaccard index (0.866), Rand index (0.853), sensitivity (0.953), specificity (0.871), and accuracy (0.917). The best random forest classifier (Weka) was worse (Dice: 0.914, accuracy: 0.904). Deep learning with ASPP yielded slightly lower values (Dice: 0.911, accuracy: 0.901).

The results for the object-based metrics show that both deep learning models yielded an above-average result for the number of erroneously added objects of 15 (without ASPP) and 23 (with ASPP). The ASPP-Net showed a slightly lower



(a) Original image (b) Ground truth segmentation (c) Global thresholding (d) Local thresholding

(e) Fast marching (f) Region competition (g) K-means clustering (h) Ilastik

(i) Weka (j) Deep learning (k) Deep learning and ASPP

Figure 7.9: Segmentation results of different methods for an example of a tissue microscopy image of glioblastoma cells.

Table 7.11: Pixel-based performance metrics for different segmentation methods. The values are mean values over 20 images. The best results are highlighted in bold. * Two different parameter settings used.

Method	Dice	JI	RI	HD	Sens.	Spec.	Acc.
Global thresholding	0.884	0.807	0.794	1.665	0.928	0.792	0.864
Local thresholding	0.881	0.792	0.773	1.139	0.893	0.823	0.864
Fast marching*	0.905	0.832	0.814	**0.775**	0.933	0.836	0.891
Region competition	0.904	0.829	0.810	0.986	0.934	0.828	0.890
K-means clustering	0.910	0.839	0.821	0.848	0.927	0.846	0.896
Ilastik	0.911	0.845	0.828	0.794	0.941	0.841	0.897
Weka	0.914	0.848	0.833	0.814	0.939	0.853	0.904
Deep learning	**0.925**	**0.866**	**0.853**	1.102	**0.953**	**0.871**	**0.917**
ASPP-Net	0.911	0.843	0.829	1.299	0.946	0.847	0.901

Table 7.12: Object-based performance metrics for different segmentation methods. The values are mean values over 20 images. The best results are highlighted in bold. * Two different parameter settings used.

Method	JI	Missing	Added	Merged
Global thresholding	0.342	1	25	43
Local thresholding	0.321	2	14	50
Fast marching*	0.321	3	**5**	42
Region competition	0.331	3	9	36
K-means clustering	0.332	1	8	44
Ilastik	0.354	**0**	11	34
Weka	0.345	1	8	36
Deep learning	**0.377**	**0**	15	23
ASPP-Net	0.366	**0**	23	**14**

object-based Jaccard index (0.366) compared to the model without ASPP (0.377), which represents the best value. Region competition yielded 36 merged objects, which is better compared to K-means clustering (44) and fast marching (42), although three objects were not detected. Both deep learning models merged only few objects incorrectly. The best result among all methods was achieved by ASPP-Net (14), the second best result was achieved for the model without ASPP (23). Thus, ASPP-Net was most effective in splitting merged cells.

7.2.2 GRUU-Net: Integrated Convolutional and Gated Recurrent Neural Network for Cell Segmentation

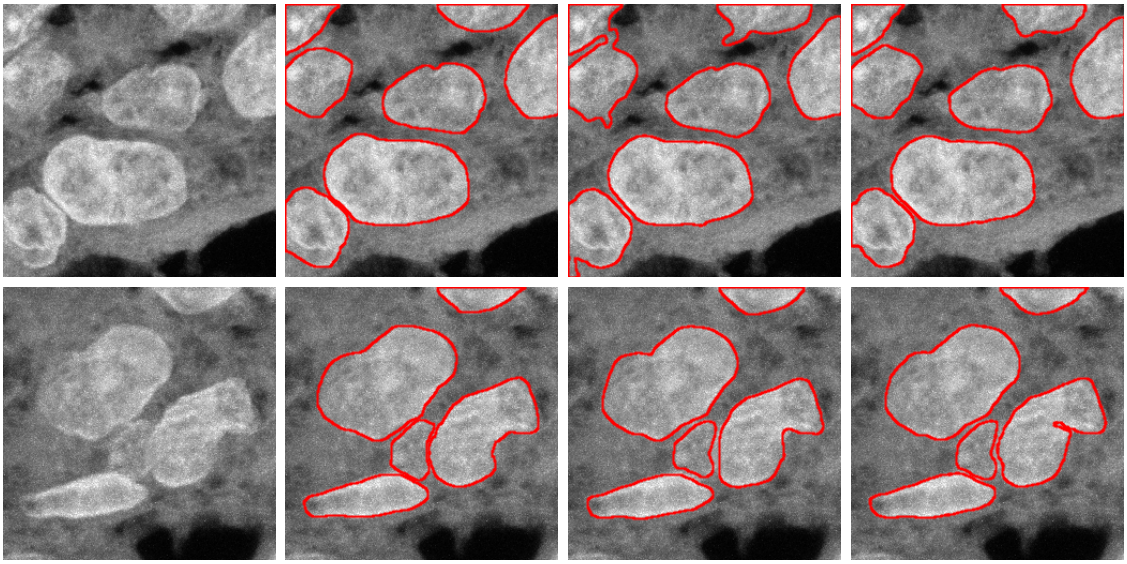
The GRUU-Net presented in Section 4.3 was applied to different types of datasets and a quantitative comparison with state-of-the-art methods was performed. To quantify the performance, Dice, SEG, and Hausdorff measures, calculated as one score integrated over all test images, were used.

Ablation study for data augmentation method

An ablation study was performed to investigate the effectiveness of the proposed data augmentation scheme. Therefore, different augmentation steps were disabled and the performance of the proposed method was evaluated. A challenging dataset was used consisting of 50 maximum intensity projection tissue images of glioblastoma cells [15]. The images have a size of 2048×2048 pixel and a resolution of $0.12\,\mu m \times 0.12\,\mu m$, and were acquired, using confocal spinning disc microscopy and show cell nuclei with fluorescently stained telomeres, centromers, PML proteins, and DNA (Erfle lab). The dataset is challenging due to high image noise, strongly heterogeneous intensity variation, cell clustering and overlaps, high shape variation, and poor contour information. Two experts manually determined the ground truth by drawing contours using ImageJ for more than 250 cell nuclei. The dataset was split into 25 training, 5 validation, and 20 test images. The proposed distributed computing scheme was used with different disabled augmentation steps to generate training datasets. These datasets were used for training of GRUU-Net. To demonstrate the generalization ability of the proposed data augmentation scheme for CNNs, a standard U-Net [27] was also used. Both networks were trained with early stopping by checking (every 100 iterations) whether a plateau is reached, and evaluated on the test images. Table 7.13 shows the experimental results. It can be observed that each augmentation step generally increases the performance. However, some augmentation steps such as zoom decrease the performance of some measures due to significantly increased variability of the dataset, and thus more difficult training. The GRUU-Net yields better results than the U-Net, but not for all ablated augmentation steps. The best result is obtained using all data augmentation steps (last column). Note that the maximum training iteration number increases with the number of augmentation steps. As expected, the number of iterations before a plateau of the

Table 7.13: Ablation study of the proposed data augmentation method for the glioblastoma dataset using the U-Net and the proposed GRUU-Net

Experiment		1	2	3	4	5	6
Cropping			✓	✓	✓	✓	✓
Flipping/Rotation				✓	✓	✓	✓
Zoom					✓	✓	✓
Brightness						✓	✓
Deformation							✓
U-Net	Training Iteration	2000	2500	3500	5000	10000	10000
	SEG	0.629	0.695	0.804	0.784	0.798	**0.807**
	Dice	0.892	0.889	0.907	0.912	0.926	**0.932**
	Hausdorff	36.844	34.190	22.106	27.019	22.277	**15.489**
GRUU-Net	Training Iteration	7500	9000	10000	12000	10000	10000
	SEG	0.647	0.695	0.723	0.751	0.811	**0.840**
	Dice	0.909	0.917	0.922	0.914	0.923	**0.933**
	Hausdorff	56.091	71.864	60.918	52.457	20.436	**14.179**



(a) Original images (b) Ground truth (c) U-Net (d) GRUU-Net

Figure 7.10: Segmentation results of GRUU-Net, U-Net, and corresponding ground truth annotations for two example images of tissue microscopy images of glioblastoma cells (top, bottom)

loss is reached (when using early stopping) increases with more data augmentation due to increased variability in the training dataset. With increasing variability in the training dataset, the generalization abilities increase and the network gets less prone to overfitting. Sample images and segmentation results of GRUU-Net compared to the U-Net using all augmentation steps are shown in Figure 7.10. It can be seen that the GRUU-Net yields superior results and separates the cell nuclei better.

Evaluation of normalized focal loss

The proposed GRUU-Net was investigated using the original focal loss [107] and the proposed normalized focal loss in Section 4.11 for the same glioblastoma dataset employed in Section 7.13 above. To demonstrate the generalization ability of the proposed normalized focal loss, a U-Net with the original focal loss and the proposed normalized focal loss was also applied. In addition, a comparison with other methods including supervised and unsupervised machine learning methods was performed. Below, these methods are outlined.

Local thresholding [254]: Gaussian filtering was performed with $\sigma = 4$ followed by Bernsen's thresholding method using a contrast threshold of 15.

Fast Marching [250]: The fast marching algorithm is based on level sets and uses a deformable model. An image was first smoothed by a Gaussian filter ($\sigma = 4$), and intensity maxima were used as seed points for the deformable model.

K-means clustering [255]: A Gaussian filter ($\sigma = 4$) was applied for smoothing, and after that, the intensity values were clustered into two clusters. The manually selected foreground cluster was used as segmentation result.

Ilastik [253]: Ilastik uses a random forest classifier for pixel-wise segmentation. All provided features were used and the image scales were defined by $\sigma = \{0.3, 0.7, 1.0, 1.6, 3.5, 5.0, 10.0\}$. The classifier was trained using 20 fully annotated images from the training set.

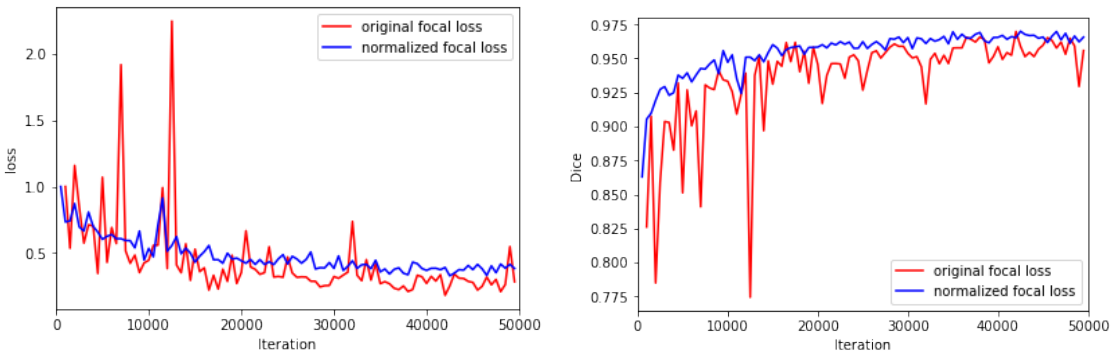
U-Net [27]: U-Net is a popular hourglass-shaped convolutional neural network for semantic segmentation. A multi-scale classifier is learned while preserving high resolution features through skip connections. Learning of difficult samples is enforced using a hand-crafted cross-entropy weight map computed by morphological operations. Training was performed using the same training data split and data augmentation as for GRUU-Net.

ASPP-Net [13]: ASPP-Net is an hourglass-shaped convolutional neural network for semantic segmentation. Compared to the U-Net, it incorporates an additional atrous spatial pyramid pooling (ASPP) block to achieve a larger receptive field than the U-Net.

From the results in Table 7.14 it can be seen that the proposed GRUU-Net yields the best performance. It also turns out that using the proposed normalized focal loss improves the performance for SEG of GRUU-Net (0.840), ASPP-Net (0.833), and of the U-Net (0.807), compared to using the Weighted CE loss by Ronneberger et al. (U-Net: 0.553, ASPP-Net: 0.798, GRUU-Net: 0.772) or the original Focal

Table 7.14: Comparison of methods for the glioblastoma dataset

Method	SEG	Dice	Hausdorff
Local thresholding	0.480	0.881	42.558
Fast Marching	0.491	0.905	36.678
K-means clustering	0.531	0.910	35.518
Ilastik	0.610	0.911	25.016
U-Net (Weighted CE loss)	0.770	0.925	18.024
U-Net (Non-Normalized FL)	0.553	0.865	61.278
U-Net (Normalized FL)	0.807	0.932	15.489
ASPP-Net (Weighted CE loss)	0.798	0.877	65.228
ASPP-Net (Non-Normalized FL)	0.708	0.844	69.299
ASPP-Net (Normalized FL)	0.833	0.911	23.351
GRUU-Net (Weighted CE loss)	0.772	0.930	18.020
GRUU-Net (Non-Normalized FL)	0.777	**0.933**	16.024
GRUU-Net	**0.840**	**0.933**	**14.179**



(a) (b)

Figure 7.11: (a) Original and normalized focal loss for the validation set during training. The values were normalized with respect to the maximum value. (b) Dice coefficient for the validation set during training for original and normalized focal loss.

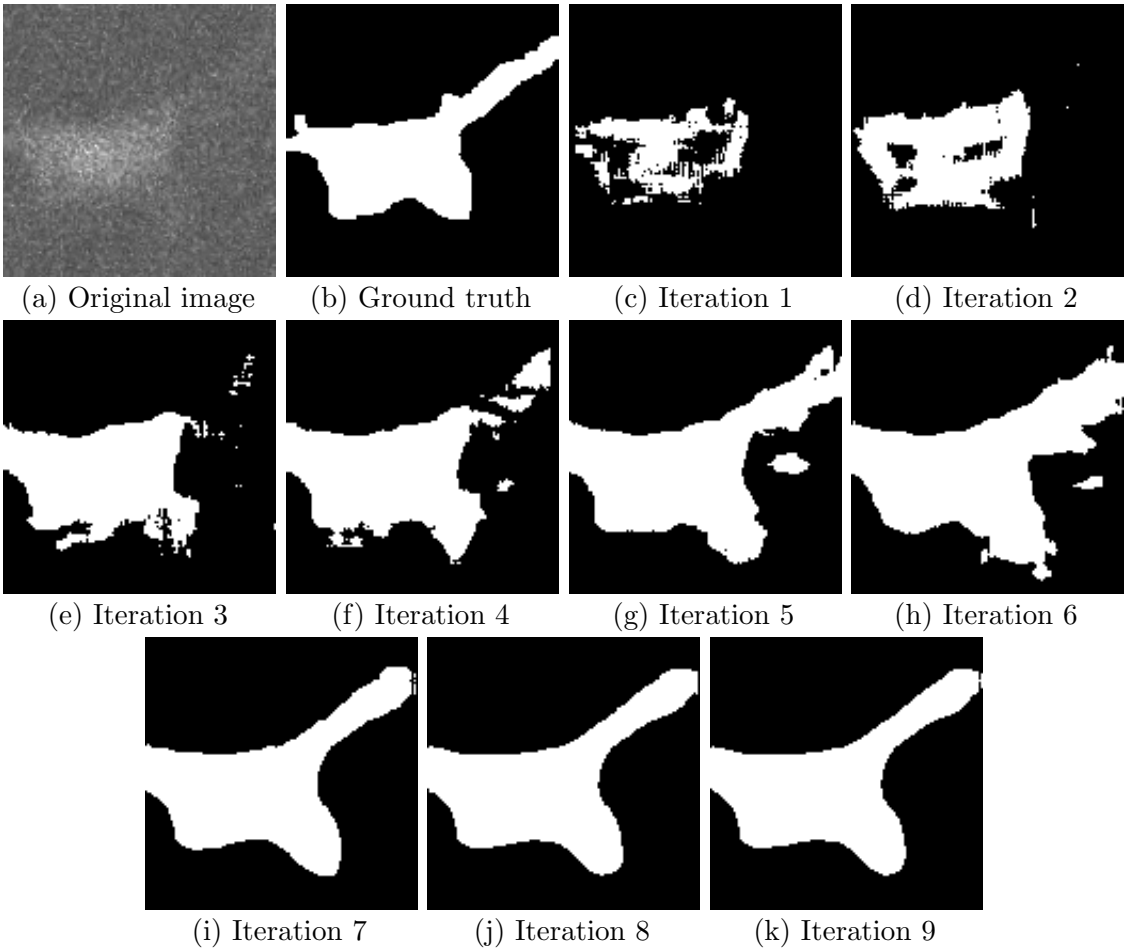
loss (U-Net: 0.770, ASPP-Net: 0.708, GRUU-Net: 0.777). Figure 7.11 shows the convergence curves of the original and normalized focal loss during training. It can be seen that the proposed normalized focal loss leads to more stable training than the original focal loss.

Visualization of iterative refinement of the GRUU-Net

To provide insight into the refinement process of GRUU-Net, segmentation results at different iterations were investigated. As example image, a fluorescence microscopy image of rat mesenchymal stem cells (Fluo-C2DL-MSC) from the Cell Tracking Challenge [127, 126] was used. The results at different iterations were obtained by applying the final residual block, convolution, and softmax function to the corresponding hidden state of the GRU (cf. Figure 4.4). The refined results as a function of the number of iterations are shown in Figure 7.12. It can be observed that the segmentation is improved in each iteration. It can also be seen that in the contracting path of the GRUU-Net (iterations 1 to 4), the segmented region is continuously enlarged. In the expanding path (iterations 5 to 9) the segmented object is smoothed.

Method comparison for Cell Tracking Challenge Data

The performance of GRUU-Net was also evaluated using the Cell Tracking Challenge training data [127, 126]. The challenge compared several cell segmentation and tracking methods (e.g., [256, 257, 258, 27]). GRUU-Net was applied to all available real 2D and 3D datasets, comprising 11 different categories of data which represent a very wide spectrum of cell microscopy data (see Figure 7.13). The datasets comprise different microscope modalities (fluorescence, differential interference contrast, phase-



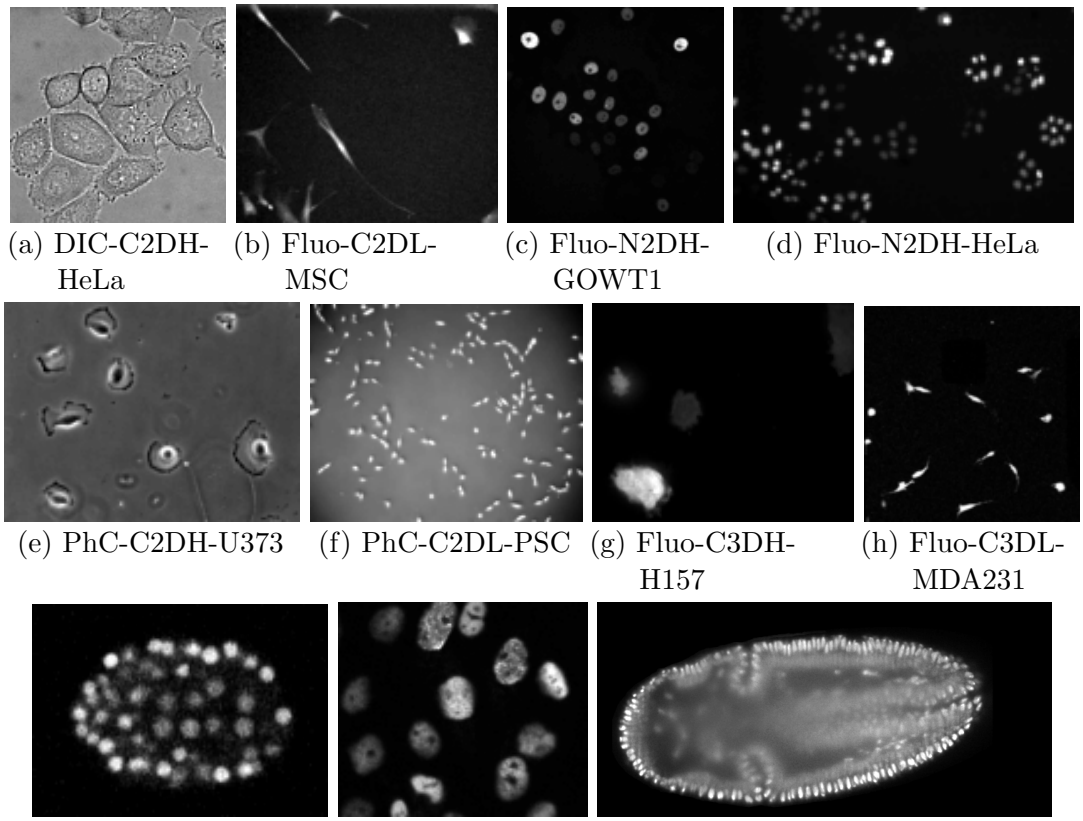
(a) Original image (b) Ground truth (c) Iteration 1 (d) Iteration 2

(e) Iteration 3 (f) Iteration 4 (g) Iteration 5 (h) Iteration 6

(i) Iteration 7 (j) Iteration 8 (k) Iteration 9

Figure 7.12: (a) Original fluorescence microscopy image of rat mesenchymal stem cells (Fluo-C2DL-MSC) from the Cell Tracking Challenge, (b) corresponding ground truth, and (c)-(k) segmentation results of GRUU-Net for different iterations.

contrast) and cells (rat mesenchymal stem cells, mouse stem cells, lung cancer cells, human breast carcinoma cells, HeLa cells, U373 cells, pancreatic stem cells, C. elegans embryo, CHO nuclei). In [126], only one method, namely UP-PT, was applied to all these 22 real data of the challenge. Each category of datasets consists of two videos. GRUU-Net was trained using the fully labeled frames of one video and tested on the other video. Thus, quite limited data was used for training. In [126], the measure SEG was employed to quantify the segmentation performance. To complement the results in [126], the mean Dice coefficient and the mean Hausdorff distance were computed if fully annotated images were available. Tables 7.15 and 7.16 respectively show the results of the proposed method for the 2D and 3D datasets. For the 2D datasets, results for different variants of the proposed network (Weighted Cross-Entropy loss, Non-Normalized Focal loss, and the proposed Normalized Focal loss) were also provided. The results were also compared with the local adaptive thresholding approach HD-Har [256] and the U-Net [27]. Note that in [126], for



(a) DIC-C2DH-HeLa (b) Fluo-C2DL-MSC (c) Fluo-N2DH-GOWT1 (d) Fluo-N2DH-HeLa

(e) PhC-C2DH-U373 (f) PhC-C2DL-PSC (g) Fluo-C3DH-H157 (h) Fluo-C3DL-MDA231

(i) Fluo-N3DH-CE (j) Fluo-N3DH-CHO (k) Fluo-N3DL-DRO

Figure 7.13: Sample images showing the variability of image data in the Cell Tracking Challenge datasets (partially contrast-enhanced for better visibility).

the U-Net both videos of a dataset category were used for training and testing. In this study, training was performed on one video and testing on the other video to guarantee a fair comparison. In addition, results of other previous methods, which are briefly outlined below, were included.

CPN [110]: A U-Net is used for cell segmentation and a Faster R-CNN [23] for cell detection. The result of the Faster R-CNN is used by ROI pooling to crop features from the U-Net to improve cell splitting.

HD-Har [256]: Local thresholding based on Otsu's method on a Gaussian filtered image is used after Gaussian filtering. A local threshold is computed if the intensity variance within an image patch is higher than a threshold, otherwise global Otsu thresholding is used.

CVXELL [259]: Ellipses are fitted to the regions of interest (ROIs) using a sequence of convex programs. The ROIs are determined using a blob detector and a modified Voronoi tessellation.

BLOB [154]: Either graph-cuts or thresholding are used for initial segmentation. Generalized Laplacian of Gaussian (gLOG) filter banks and non-maxima suppression are employed to split cell clusters.

GC-ME [260]: This method uses graph cuts with asymmetric boundary costs for cell segmentation.

UP-PT [257]: Non-maxima suppression is performed on the result of an LoG filter. The cell shape is determined using a local convergence filter.

Table 7.15: Results for the real 2D datasets of the Cell Tracking Challenge

Dataset	Video	Method	SEG	Dice	Hausdorff
DIC-C2DH-HeLa	1	UP-PT	0.345		
		U-Net	0.327	0.880	52.404
		GRUU-Net (Weighted CE loss)	0.258	0.885	103.858
		GRUU-Net (Non-Normalized FL)	0.290	0.907	88.803
		GRUU-Net	**0.648**	**0.886**	**36.673**
	2	UP-PT	0.125		
		U-Net	0.219	0.853	63.463
		GRUU-Net (Weighted CE loss)	0.333	0.901	88.479
		GRUU-Net (Non-Normalized FL)	0.420	0.899	84.652
		GRUU-Net	**0.490**	**0.870**	**46.856**
Fluo-C2DL-MSC	1	UP-PT	0.382		
		HD-Har	**0.450**	0.593	109.631
		U-Net	0.408	**0.711**	**78.912**
		GRUU-Net (Weighted CE loss)	0.209	0.361	338.677
		GRUU-Net (Non-Normalized FL)	0.222	0.451	381.431
		GRUU-Net	0.329	0.620	84.126
	2	UP-PT	0.264		
		HD-Har	**0.598**	0.745	**101.842**
		U-Net	0.502	0.672	189.401
		GRUU-Net (Weighted CE loss)	0.535	0.793	290.017
		GRUU-Net (Non-Normalized FL)	0.543	0.792	293.935
		GRUU-Net	0.550	**0.772**	137.963

Fluo-N2DH-GOWT1	1	UP-PT	0.703		
		HD-Har	0.545	0.883	**6.833**
		CPN	0.851		
		CVXELL	0.821	0.637	
		U-Net	0.814	0.864	23.219
		GRUU-Net (Weighted CE loss)	0.854	0.939	100.644
		GRUU-Net (Non-Normalized FL)	0.866	0.946	99.016
		GRUU-Net	**0.888**	**0.901**	43.788
	2	UP-PT	0.798		
		HD-Har	0.898	0.925	**8.080**
		CPN	0.873		
		CVXELL	0.913	0.894	
		U-Net	0.832	0.826	21.995
		GRUU-Net (Weighted CE loss)	0.843	0.929	176.479
		GRUU-Net (Non-Normalized FL)	0.840	0.926	60.839
		GRUU-Net	**0.929**	**0.956**	11.776
Fluo-N2DH-HeLa	1	UP-PT	0.627		
		HD-Har	0.744	**0.887**	9.943
		CPN	**0.831**		
		BLOB	0.795		
		U-Net	0.775	0.875	**6.674**
		GRUU-Net (Weighted CE loss)	0.706	0.838	91.530
		GRUU-Net (Non-Normalized FL)	0.788	0.888	1.500
		GRUU-Net	0.749	0.858	7.145
	2	UP-PT	0.709		
		HD-Har	0.814	0.897	**6.651**
		CPN	**0.845**		
		BLOB	0.839		
		U-Net	0.798	0.892	7.581
		GRUU-Net (Weighted CE loss)	0.813	0.899	7.193
		GRUU-Net (Non-Normalized FL)	0.788	0.901	7.009
		GRUU-Net	0.809	**0.911**	7.341

PhC-C2DH-U373		UP-PT	0.356		
		CPN	0.734		
		GC-ME	0.875		
	1	U-Net	0.812	0.869	59.156
		GRUU-Net (Weighted CE loss)	0.926	0.930	57.507
		GRUU-Net (Non-Normalized FL)	0.922	0.941	53.957
		GRUU-Net	**0.938**	**0.942**	**47.463**
		UP-PT	0.359		
		CPN	0.738		
		GC-ME	0.757		
	2	U-Net	0.739	0.791	71.665
		GRUU-Net (Weighted CE loss)	0.787	0.859	75.490
		GRUU-Net (Non-Normalized FL)	0.796	0.874	42.402
		GRUU-Net	**0.814**	**0.889**	**34.513**
PhC-C2DL-PSC		UP-PT	0.514		
		HD-Har	0.464	**0.720**	**7.374**
		CPN	0.661		
	1	U-Net	0.347	0.663	8.141
		GRUU-Net (Weighted CE loss)	0.256	0.497	105.252
		GRUU-Net (Non-Normalized FL)	0.264	0.524	96.013
		GRUU-Net	**0.684**	0.711	9.142
		UP-PT	0.477		
		HD-Har	0.465	0.415	12.479
		CPN	**0.648**		
	2	U-Net	0.272	0.635	**8.520**
		GRUU-Net (Weighted CE loss)	0.311	0.121	39.735
		GRUU-Net (Non-Normalized FL)	0.329	0.598	100.996
		GRUU-Net	0.422	**0.686**	9.310

Table 7.16: Results for the real 3D datasets of the Cell Tracking Challenge

Dataset	Video	Method	SEG	Dice	Hausdorff
Fluo-C3DH-H157	1	UP-PT	0.458		
		HD-Har	0.753	0.922	105.897
		U-Net	0.017	0.007	**21.664**
		GRUU-Net	**0.759**	**0.929**	29.216
	2	UP-PT	0.557		
		HD-Har	0.573	0.766	**36.825**
		U-Net	0.032	0.037	166.007
		GRUU-Net	**0.602**	**0.865**	55.383
Fluo-C3DL-MDA231	1	UP-PT	0.348		
		HD-Har	0.196	0.494	**59.969**
		U-Net	0.340	0.521	90.787
		GRUU-Net	**0.570**	**0.703**	75.506
	2	UP-PT	0.429		
		HD-Har	0.290	0.521	**5.663**
		U-Net	**0.516**	0.649	70.452
		GRUU-Net	0.503	**0.792**	12.657
Fluo-N3DH-CE	1	UP-PT	0.385		
		HD-Har	0.566	**0.772**	31.735
		U-Net	**0.627**	0.760	**17.844**
		GRUU-Net	0.598	0.716	19.485
	2	UP-PT	0.355		
		HD-Har	0.486	0.735	24.539
		U-Net	**0.636**	0.683	**20.256**
		GRUU-Net	**0.636**	**0.747**	34.010
Fluo-N3DH-CHO	1	UP-PT	0.625		
		HD-Har	**0.814**	**0.875**	**27.622**
		U-Net	0.579	0.661	29.887
		GRUU-Net	0.595	0.671	36.449
	2	UP-PT	0.682		
		HD-Har	**0.903**	**0.950**	**8.740**
		U-Net	0.746	0.815	19.961
		GRUU-Net	0.729	0.810	24.564
Fluo-N3DL-DRO	1	UP-PT	0.296		
		U-Net	0.423		
		GRUU-Net	**0.534**		
	2	UP-PT	0.205		
		U-Net	0.640		
		GRUU-Net	**0.709**		

From the results in Tables 7.15 and 7.16 it turns out that the proposed method

achieved the best performance for SEG for 13 out of 22 datasets, and was among the top two methods for 14 out of 22 datasets. For the Dice coefficient, the proposed method was best in 14 out of 20 datasets, and among the top two methods for 17 datasets. It was investigated whether GRUU-Net yields a statistically significant improvement for SEG and Dice compared to UP-PT and U-Net, which were applied to all datasets. A Shapiro-Wilk test revealed that the results for SEG and Dice do not follow a normal distribution. Therefore, a Wilcoxon signed-rank test was conducted with significance level of 5%. For the comparison of GRUU-Net with UP-PT, $p < 0.001$ was obtained for SEG and Dice. GRUU-Net and U-Net yielded $p < 0.003$ for SEG and $p < 0.004$ for Dice. Thus, the proposed method yields a statistically significant improvement over UP-PT and U-Net. Comparing the different variants of the proposed network in Table 7.15, it turns out that the results are consistent with the results of the ablation study in Table 7.14. For some datasets, a relatively high Hausdorff distance was observed, which is an indication for missed objects (the Hausdorff distance was computed for the whole image). For some datasets (e.g. DIC-C2DH-HeLa, Fluo-C3DH-H157), it can be observed that U-Net overfitted much faster than the proposed GRUU-Net, which is indicated by the maximum number of training iterations using early stopping (cf. Table 7.13). In addition, the cell appearance in the two videos for a dataset is quite different. Thus, the reason for the low performance is probably that the networks overfitted on the specific appearance of one video and did not generalize well to the other video. Partially, classical methods that do not use machine learning performed quite well. However, these methods were probably tuned based on all training and challenge data, which generally leads to overfitting. Since the proposed method achieved the best results for SEG in most datasets, it can cope better with the high variability in the 2D and 3D datasets compared to previous methods. GRUU-Net was part of the Cell Segmentation Benchmark of the Cell Tracking Challenge at ISBI 2019 and achieved top-3 rankings in three categories.

7.2.3 Instance Segmentation for Cell Images

A preliminary study of the instance segmentation methods described in Section 4.4 has been conducted using an Fully convolutional Networks (FPN) [106] backbone on the 2018 Data Science Bowl challenge dataset to identify promising instance segmentation approaches for microscopy cell segmentation (Tian, Wollmann et al., see [261]). The mean average precision (MAP) for Jaccard index thresholds, in a range from 0.5 to 0.95 with a step size of 0.05, was used for evaluation. The results are shown in Table 7.17. The Deep Watershed Transform performs best. It was observed that anchors for small objects and false detections in Mask R-CNN [146] harmed the performance. The performance of embedding-based approaches [213, 214] highly relied on their post processing. Approaches predicting a distance map, like the Deep Watershed Transform [215], turned out to be relatively robust.

Table 7.17: Mean average precision of instance segmentation methods using a FPN backbone on the 2018 Data Science Bowl challenge dataset.

Method	MAP
FCN	0.234
Mask R-CNN	0.084
Discriminative Loss	0.070
Cosine Embedding Loss	0.072
Deep Watershed Transform	**0.282**

Therefore, approaches based on predicting distance maps seem promising for instance segmentation in microscopy images.

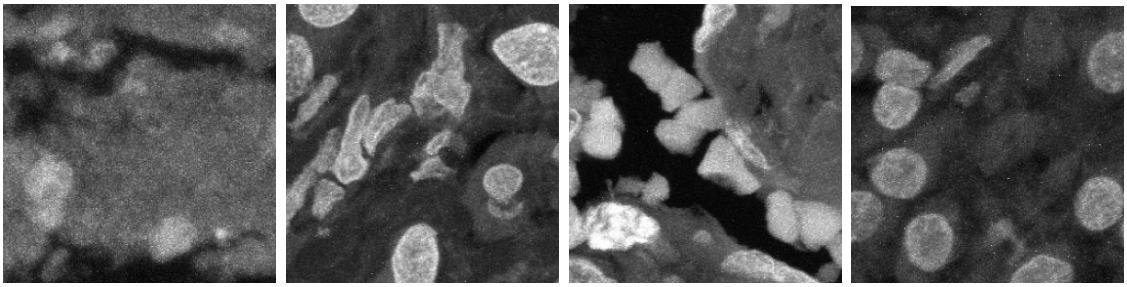
7.3 Hyperparameter Optimization

To showcase the HyperHyper optimization framework presented in Chapter 5, four different experiments were conducted. First, the segmentation of cell nuclei in prostate tissue slides using a clustering and a deep learning pipeline was considered. Second, the detection of hepatitis C virus (HCV) proteins in live cell fluorescence microscopy images was studied. In these two experiments, it is revealed that the separation of sampling and optimization strategy improves the optimal solution. In the third experiment, an extension of the pipeline for detection of HCV proteins by image pre-processing was considered. In this experiment, it is shown how an infimum projection of the loss surface can provide insights into the optimization problem. In the fourth experiment a larger pipeline for particle detection and tracking was studied. In the following, the used hyperparameters that need to be optimized are highlighted in italics.

Experiment 1: Segmentation of Cell Nuclei

In the field of histopathology, segmentation of cell nuclei in tissue microscopy images is a pivotal and essential task. 2D DAPI stained prostate tissue image slices using an Opera spinning disk confocal microscope at 60x magnification with a resolution of $107.7\,nm \times 107.7\,nm$ were acquired (Erfle lab). Cell nuclei segmentation is needed for telomere quantification on a single-cell basis. Due to image noise, variation of cell shape and image intensity, and low contrast, cell nuclei segmentation is challenging (Figure 7.14) [11]. For analyzing high-content histological screening data, hyperparameters of image analysis methods need to be optimized, which is often very difficult or not feasible to perform manually due to the high dimensionality of the hyperparameter space. HyperHyper was applied in conjunction with two different segmentation pipelines to analyze tissue images of the prostate and investigate the suitability of the proposed black-box hyperparameter optimization framework.

The first pipeline consists of K-means clustering after image smoothing by a



(a) (b) (c) (d)

Figure 7.14: Examples of prostate tissue images showing various challenges for image analysis. a) Strong background noise. b) Strong shape variation. c) Strong intensity variation. d) Low contrast.

Gaussian filter with σ_{Gauss}. The type of *cluster initialization* (random, k-means++) is optimized and the seed value is set to a fixed value to generate a deterministic segmentation pipeline. Subsequently, median filtering and morphological closing of small holes was applied. Then, a *geometric feature* (e.g., major axis length, eccentricity) determined by optimization was computed and for each cluster compared to the mean of all clusters to assign the cluster to foreground. The segmented objects are thresholded regarding *area* (upper and lower threshold) and *solidity*. In summary, for this pipeline, the hyperparameters σ_{Gauss}, *cluster initialization*, *geometric feature*, *area*, and *solidity* need to be optimized.

The second pipeline consists of a U-Net [27] with Adam optimizer [55] and early stopping. For the network, the same seed value was used for sampling the initial weights as for the K-means clustering pipeline for a fair comparison. Data augmentation was performed with image rotation, flipping, and elastic deformation. To discard small objects, the geometric feature "area" was thresholded with *area threshold*. In summary, for this pipeline, the hyperparameters *learning rate*, *batch size*, and *area threshold* need to be optimized.

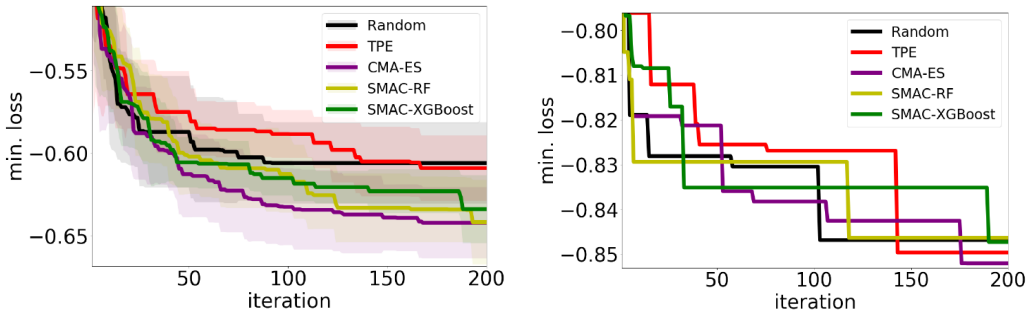
HyperHyper was applied for both pipelines using multiple optimizers, namely Random, TPE, CMA-ES, SMAC-RF, and SMAC-XGBoost, and performed a distributed optimization on 20 computer nodes. The K-means clustering pipeline was evaluated with 10 runs and 200 evaluations for each optimizer (in total: 10,000 evaluations), and the U-Net pipeline with 1 run and 200 evaluations for each optimizer (in total: 1,000 evaluations). To determine the global optimum, Grid Search was applied using 17,388 evaluations. The tissue images have different sizes and were divided into 256×256 pixel image patches before randomly splitting them into 75% for training and 25% for testing. For optimization purposes, 60 ground truth images annotated by an expert were used. The K-means clustering and U-Net segmentation performance was optimized with the soft Dice loss [72].

The results for both pipelines are reported in Table 7.18 as mean and standard deviation of Dice (balancing precision and sensitivity), and as mean and standard deviation of the difference (ΔDice) to the Dice value after the warm-up phase. The

Table 7.18: Results for the K-means clustering and U-Net pipeline with different optimizers. The table shows the improvement ΔDice (mean ± std.) after the warm-up phase and the absolute Dice value (mean ± std.). The best results are highlighted in bold.

Pipeline	Optimizer	ΔDice (Improvement)	Dice
K-means clustering	Random	0.030±**0.028**	0.606±0.025
	TPE	0.045±**0.028**	0.609±**0.020**
	CMA-ES	0.077±0.034	**0.642**±0.021
	SMAC-RF	**0.094**±0.043	**0.642**±0.026
	SMAC-XGBoost	0.064±0.038	0.634±0.021
	Grid Search (coarse)	-	0.614
	Grid Search	-	*0.654*
U-Net	Random	0.019	0.847
	TPE	0.038	0.850
	CMA-ES	0.033	**0.852**
	SMAC-RF	0.017	0.846
	SMAC-XGBoost	**0.039**	0.847
	Grid Search	-	*0.864*

optimizers perform a warm-up phase for each run, and explore the hyperparameter space by evaluating 20 random samples before applying the optimization strategy. The overall improvement by the optimization for each optimizer is reflected by the mean ΔDice. The Dice values as a function of the number of iterations for all optimizers are shown in Figure 7.15. For the K-means clustering pipeline, CMA-ES and SMAC-RF yield the best segmentation performance with a Dice value of 0.642. However, CMA-ES has a lower standard deviation with a Dice value of 0.021 compared to SMAC-RF. TPE yields the lowest standard deviation regarding Dice among all optimizers. The global optimum of 0.654 was determined by Grid Search. Instead of using (dense) Grid Search with a high number of evaluations (17,388 evaluations), a coarse grid with a similar number of evaluations as for the optimizers (198 evaluations) was used. For the K-means clustering pipeline a Dice value of 0.614 was obtained, which



(a) (b)

Figure 7.15: Convergence of different optimizers as a function of the number of iterations. a) K-means clustering pipeline. b) U-Net pipeline.

is lower compared to SMAC-XGBoost, SMAC-RF, and CMA-ES, but higher than Random and TPE. Regarding ΔDice, SMAC-RF obtains the largest improvement ΔDice = 0.094 compared to all other optimizers. For the U-Net pipeline, CMA-ES achieves the largest Dice value of 0.852, SMAC-XGBoost obtains the highest improvement ΔDice = 0.039. In Figure 7.16, the segmentation results for the best optimizer for each pipeline for Dice and the best optimizer for ΔDice are shown. Comparing the results with the ground truth, it can be seen that the K-means clustering pipeline has problems with splitting cell nuclei for both best optimizers for Dice and ΔDice. Overall, the best segmentation performance determined by Grid Search is achieved by the U-Net pipeline, which yields a significantly higher Dice value of 0.864 compared to the K-means clustering pipeline with 0.654. For K-means clustering, comparing SMAC-RF and SMAC-XGBoost in Table 7.18, SMAC-RF yields a better Dice value than SMAC-XGBoost. In contrast, for the U-Net pipeline, SMAC-XGBoost yields a better Dice value than SMAC-RF. Since SMAC-RF and SMAC-XGBoost use the same sampling strategy, but different optimization strategies, this demonstrates that the proposed separation of sampling and optimization strategy in HyperHyper yields better solutions.

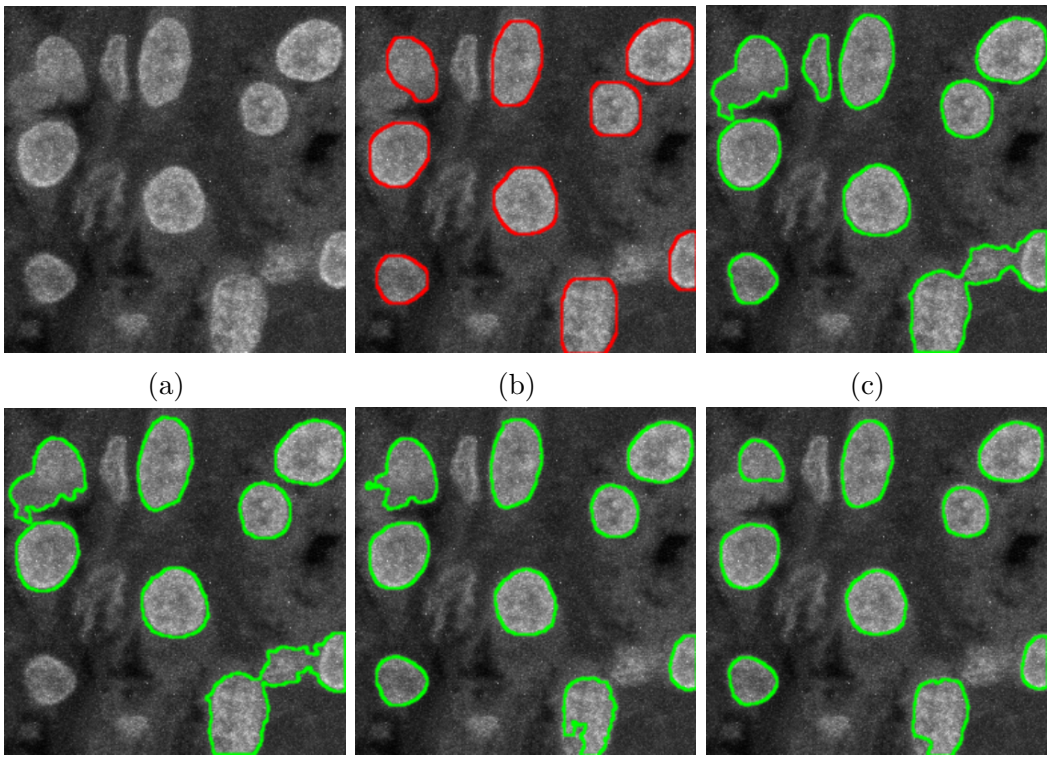


Figure 7.16: Comparison of segmentation results for both pipelines with different optimizers. a) Original image. b) Original image with ground truth annotations by an expert. c) K-means clustering pipeline with CMA-ES. d) K-means clustering pipeline with SMAC-RF. e) U-Net pipeline with CMA-ES. f) U-Net pipeline with SMAC-XGBoost.

Experiment 2: Detection of HCV Proteins

In the second experiment, HyperHyper was evaluated for live cell fluorescence microscopy data displaying fluorescently labeled HCV NS5A as small round particles. Detection of subcellular structures such as proteins is a prerequisite for tracking [163] to obtain quantitative information on cellular processes. The live cell data was acquired using an Ultra-View ERS spinning disk confocal microscope with an image size of 355×447 pixels (Bartenschlager lab). To detect HCV proteins, the spot-enhancing filter (SEF) [161] was used, which consists of applying a Laplacian-of-Gaussian filter (LoG) with standard deviation σ_{LoG}, followed by thresholding the filtered image. The threshold is based on the mean intensity of the filtered image plus a factor c times the standard deviation of the filtered image intensities [40, 262]. To detect HCV proteins, the hyperparameters σ_{LoG} and c have to be optimized.

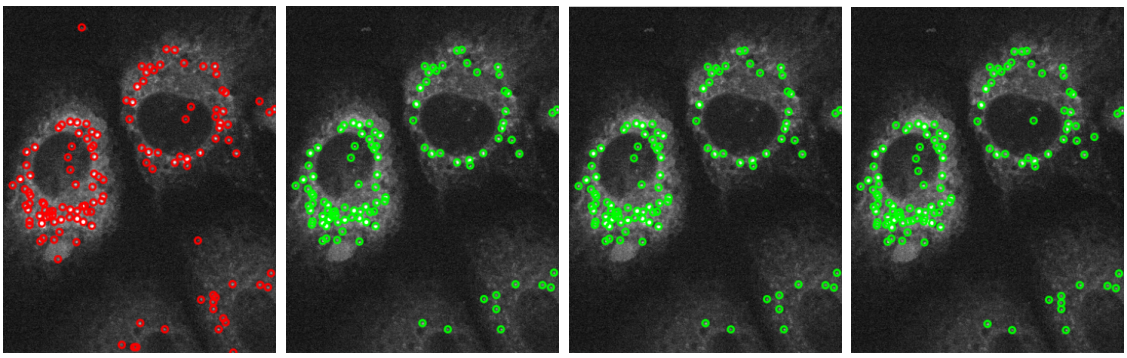
As in experiment 1, Random, TPE, CMA-ES, SMAC-RF, and SMAC-XGBoost for hyperparameter optimization were used. 10 runs per optimizer with 3,500 evaluations distributed on 20 compute nodes (in total: 175,000 evaluations) were performed. To determine the global optimum, dense Grid Search with 35,000 evaluations distributed on 20 compute nodes was used. The performance of SEF detection was optimized and evaluated using the F1 score (balancing precision and sensitivity) and 128 ground truth annotations. The assignment between the ground truth annotations and SEF detections was determined using the Munkres algorithm [242] and a gating distance of 5 pixels. Similar to experiment 1, the mean and standard deviation of ΔF1 showing the difference to the F1 score after the warm-up phase was computed.

The results for the different optimizers are shown in Table 7.19. The best performance is obtained by SMAC-RF and SMAC-XGBoost with an F1 score of 0.872, which are the only optimizers reaching the global optimum. The largest improvement ΔF1 is obtained by SMAC-RF with 0.050. In Figure 7.17 c), the result for SMAC-RF (green circles) is shown together with the ground truth (red circles) in Figure 7.17 a) and the global optimum (Grid Search) in Figure 7.17 b). The F1 score as a function of the number of iterations for all optimizers is depicted in Figure 7.18. To better assess the convergence of the optimizers, only the first 2,000 iterations

Table 7.19: Results for the HCV protein detection pipeline with different optimizers. The table shows the improvement ΔF1 (mean ± std.) after the warm-up phase and the absolute F1 score (mean ± std.). The best results are highlighted in bold.

Pipeline	Optimizer	ΔF1 (Improvement)	F1
SEF	Random	0.043±0.033	0.871±0.002
	TPE	0.041±0.023	0.867±0.000
	CMA-ES	0.022±**0.008**	0.871±0.001
	SMAC-RF	**0.050**±0.037	**0.872±0.000**
	SMAC-XGBoost	0.041±0.037	**0.872±0.000**
	Grid Search	-	*0.872*

are shown. The fastest convergence is obtained by SMAC-RF and SMAC-XGBoost, whereas TPE is the slowest. To obtain more insights into the optimization process and to visualize the dependency between the hyperparameters, the loss surface was computed with Grid Search. The loss surface is shown in Figure 7.19 a), and the global optimum is marked by a blue star. A clear dependence between c and σ_{LoG} is visible by the valley shape of the loss surface. In addition, a visualization of the trail of an optimizer on the loss surface is provided for spatial assessment of the convergence process. In Figure 7.19 b), the best trails of Random and SMAC-RF are depicted. A trail represents the connection of the best evaluations per optimization step. It can be seen that Random finds the optimum more directly compared to SMAC-RF. However, according to Figure 7.18, SMAC-RF has a much faster convergence than Random.



(a) (b) (c) (d)

Figure 7.17: Detection results for HCV live cell microscopy data with different hyperparameter optimizations. a) Ground truth annotated by an expert. b) Experiment 2 using Grid Search c) Experiment 2 using SMAC-RF. d) Experiment 3 using Grid Search.

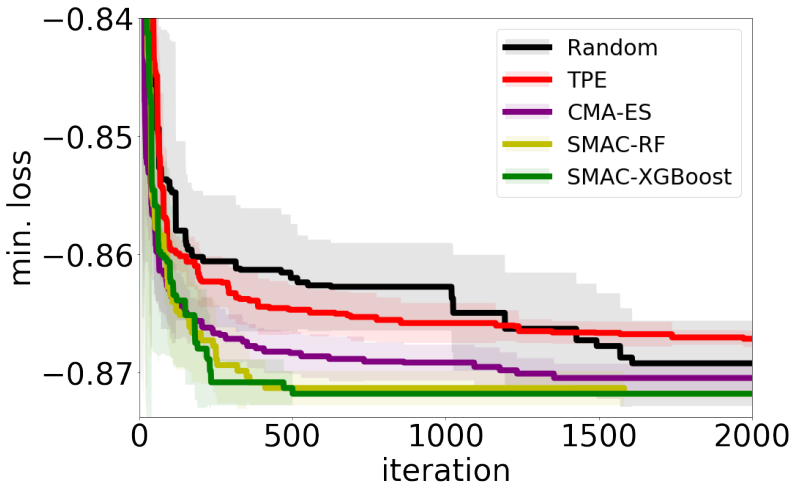
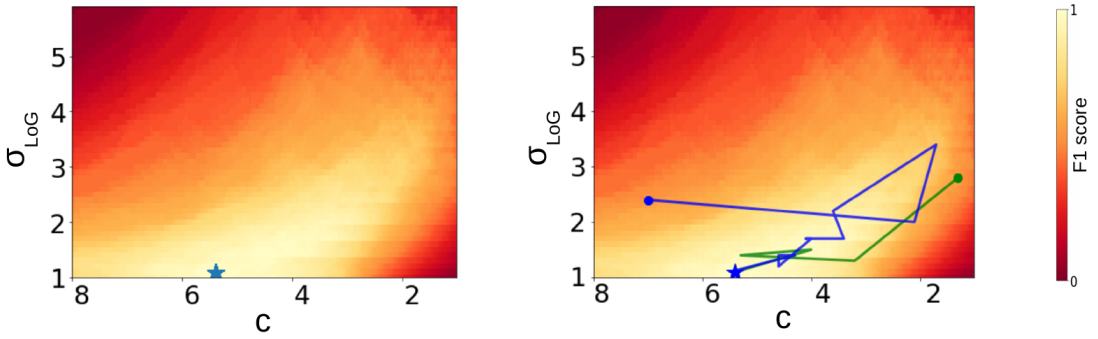


Figure 7.18: Convergence of different optimizers as a function of the number of iterations.



(a) (b)

Figure 7.19: Loss surface of experiment 2 for 2D hyperparameter space (c and σ_{LoG}). a) The hyperparameter space was sampled with Grid Search and the global optimum is marked with a blue star. b) Same as in a), but with optimization trails of Random (green) and SMAC-RF (blue). For both trails, the dot is the starting point and the star shows the found optimal solution. Both trails represent the best evaluations per optimization step over time.

Experiment 3: Image Pre-Processing for Detection of HCV Proteins

With this experiment, the importance of an infimum projection as visualization of the loss function to gain further insight on the dependency of the hyperparameters is shown. An additional image pre-processing step for the pipeline for HCV protein detection from experiment 2 above is studied. As pre-processing step, the image is smoothed with a Gaussian filter with standard deviation σ_{Gauss} and subtracted the filtered image from the original image to enhance the particles and suppress background noise. We now have a 3D hyperparameter space containing σ_{Gauss}, σ_{LoG}, and c. The global optimum was computed with Grid Search (total: 175,000 evaluations).

Figures 7.20 a) and b) display the original and filtered live cell data respectively. Table 7.20 shows the improvement using the pre-processing step. The F1 score using pre-processing is 1.6 % higher compared to the result from experiment 2 without pre-processing. The detection result using pre-processing is displayed in Figure 7.17 d). All HCV proteins within the upper two cells are detected, whereas without pre-processing, only a few of them were detected.

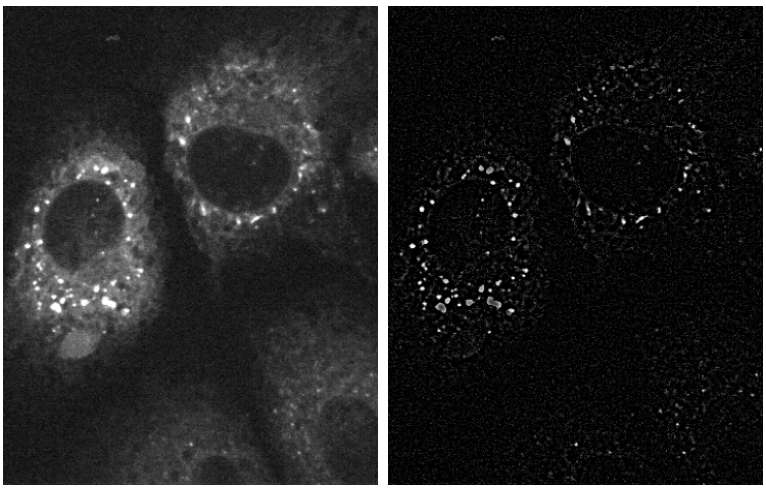
Table 7.20: Results for the two HCV protein detection pipelines (experiment 2 and 3) with Grid Search. The table shows the absolute F1 score. The best result is highlighted in bold.

Pipeline	F1
SEF	0.872
SEF + Pre-processing	**0.888**

Table 7.21: Results of PCA for the whole loss surface data. The table provides the eigenvectors and eigenvalues of the four principal components (PC) together with the ratio between the cumulative variance and the total variance in [%].

PCA	Variable	PC 1	PC 2	PC 3	PC 4
Eigenvectors	c	0.224	-0.948	$0.001 \cdot 10^{-13}$	0.224
	σ_{LoG}	-0.668	-0.316	-0.081	-0.668
	σ_{Gauss}	-0.054	-0.026	0.997	-0.054
	$loss$	-0.707	$-0.002 \cdot 10^{-12}$	$0.003 \cdot 10^{-14}$	0.707
Eigenvalues		1.692	1.000	1.000	0.308
Cumulative variance ratio		42.3 %	67.3 %	92.3 %	100.0 %

To obtain insights into the optimization process and to quantify the dependency of the hyperparameters, a principal component analysis (PCA) [263] of the loss function was conducted. The results are shown in Table 7.21. The values of the loss function were normalized (zero mean and variance of one), and the eigenvectors with corresponding eigenvalues were computed (principal components, PCs). It can be seen that in order to represent 90 % of the variance, the first three PCs need to be taken into account. For the first PC, the hyperparameter σ_{Gauss} has a more than ten times smaller influence than c and σ_{LoG}. In addition, the other two PCs have a minor influence on the loss. Therefore, the influence of σ_{Gauss} on the loss is relatively small. Table 7.20 shows that pre-processing by a Gaussian filter improves the detection pipeline performance. To further investigate the dependencies of the hyperparameters, it is proposed to generate infimum projection visualizations. The infimum projection of a countable finite n-dimensional loss $\mathrm{L} : Q_1 \times ... \times Q_n \rightarrow \mathbb{R}$ into a lower dimensional projection P onto the index set $I \subseteq [\#Q]$ of features



(a) (b)

Figure 7.20: HCV fluorescence microscopy data. a) Original image. b) Pre-processed image with optimal σ_{Gauss} obtained by Grid Search.

$Q = \{Q_1, ..., Q_n\}$ with n elements can be performed by:

$$\mathrm{P}(I; q_1, ..., q_n) = \min_{\substack{q_k \in Q_k \\ k \notin I}} \{\mathrm{L}(q_1, ..., q_n)\} \tag{7.7}$$

In Figure 7.21 a) - c), the infimum projections between the three hyperparameters are shown. Figure 7.21 a) can be compared with the loss surface for c and σ_{LoG} in experiment 2 in Figure 7.19 a), where both hyperparameters (c and σ_{LoG}) are plotted. Both loss surfaces show the same structure, and therefore a priori knowledge from experiment 2 could be transferred to the optimization problem in experiment 3. In addition, from the loss surfaces in Figure 7.21 b) and c), one can see that the former hyperparameters (c and σ_{LoG}) and the additional parameter σ_{Gauss} seem to be independent due to the homogeneous structures of the loss surfaces. Thus, the infimum projection yields additional information for the PCA analysis. Figure 7.21 c) and d) together with Table 7.20 indicate that the optimization problem can be restructured by optimizing separately σ_{Gauss} and c along with σ_{LoG}. This sequential optimization procedure reduces the 3D hyperparameter optimization to a 1D optimization along with a 2D optimization.

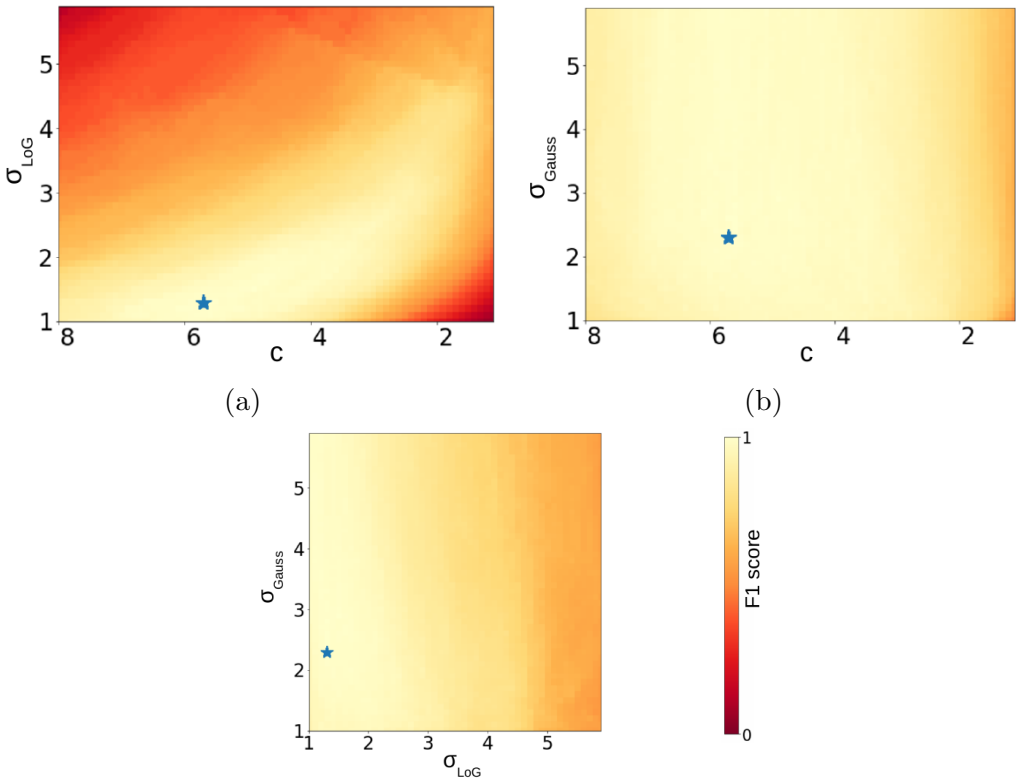


Figure 7.21: Infimum projections of the loss surface from experiment 3 for the 3D hyperparameter space (c, σ_{LoG}, and σ_{Gauss}) sampled with Grid Search. The global optimum is marked with a blue star.

Experiment 4: Particle Detection and Tracking

Preliminary studies using larger image analysis pipelines have been conducted. A combination of the DetNet particle detection network (Section 3.2) and the probabilistic particle tracking method PDAE (Probabilistic Data Association with Elliptical Sampling) [40] was used to track particles in microscopy image sequences (Ritter, Wollmann, et al. [9]). DetNet-PDAE combines the benefits of both methods and can cope with a small number of training samples. PDAE uses multiple measurements from DetNet and an elliptical sampler, and integrates the information using a Kalman filter via combined innovation. Model parameters of PDAE were optimized using CMA-ES in HyperHyper [4]. The DetNet-PDAE method has been benchmarked using data from the Particle Tracking Challenge.

It turned out that optimized DetNet-PDAE outperforms the non-optimized DetNet-PDAE in all performance measures of the Particle Tracking Challenge.

7.4 Transfer Learning for Microscopy Image Data

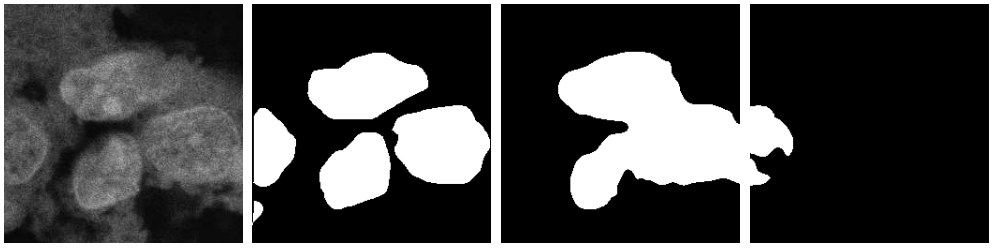
In this section, the transfer learning methods proposed in Chapter 6 are evaluated.

7.4.1 Multi-Channel Deep Transfer Learning

The transfer learning method presented in Section 6.2 considers the segmentation of nuclei from 3D tissue microscopy images of glioblastoma cells. This data is very challenging due to strong intensity variation, cell clustering, poor edge information, missing object borders, strong shape variation, and low signal-to-noise ratio. The dataset consists of five 3D images acquired, using a Leica TCS SP5 point scanning confocal microscope with a 63x objective lens and a voxel size of $100 \times 100 \times 250\,\mathrm{nm}$ (Erfle lab). Four color channels were imaged sequentially: PML antibody stain (Alexa 647), FISH CY3 telomere probe, FAM labeled CENP-B PNA probe, and DAPI nuclei stain. 45 axial sections were acquired for each 3D stack. Deep learning models with transfer learning were trained on a dataset with glioblastoma cells containing 50 images stained with DAPI nuclei stain before training on the considered four color channel dataset. However, the first dataset has only one channel and consists of maximum intensity projection (MIP) images. Therefore, standard transfer learning is not applicable and other approaches such as the two transfer learning strategies described in Section 6.2 are needed. Four performance measures were used for quantitative evaluation: SEG, IOU, Dice, and Warping Error (see Section 7.2 above). For a quantitative comparison, thresholding in combination with mean shift clustering was used as well. The 3D images were pre-processed using 3D Gaussian filtering ($\sigma = 2\,\mathrm{pixels}$). An empirically determined threshold of 160 was used. In addition, an approach based on Gaussian filtering, mean shift clustering, and 3D fast-marching level sets [264] was used. The segmentation results were post-processed

Table 7.22: Performance of different segmentation methods. Bold and underline highlights the best result, and bold indicates the second best result.

Method	SEG	JI	Dice	Warp Error [10^{-4}]
Clustering & Thresholding	0.5782	**<u>0.6520</u>**	**<u>0.7884</u>**	0.093
Fast-Marching Level Set	0.5065	0.5682	0.7194	0.149
U-Net	0.6666	0.5774	0.7168	**<u>0.011</u>**
Proposed NN	0.7814	0.6154	0.7581	0.040
Proposed NN (transf., same filter)	**0.7913**	0.5221	0.6709	0.045
Proposed NN (transf., indiv. filters)	**<u>0.7981</u>**	**0.6426**	**0.7775**	**0.030**



(a) Original image (b) Ground truth (c) Clustering & Thresholding (d) Fast-Marching Level Set

(e) U-Net (f) Proposed NN (g) Proposed NN (transfer, same filter) (h) Proposed NN (transfer, individual filters)

Figure 7.22: Example tissue microscopy image of glioblastoma cells, ground truth, and segmentation results of different methods.

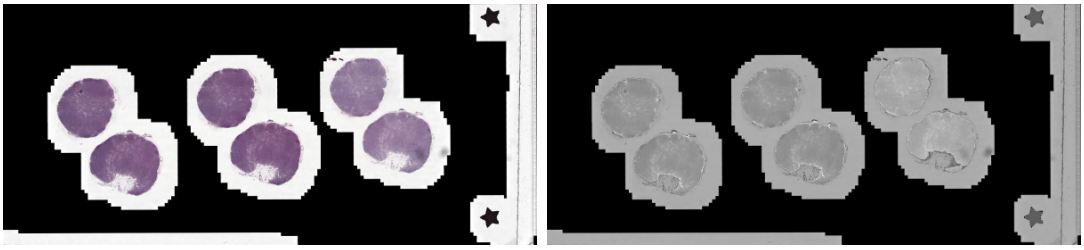
using hole filling. All segmentation methods were evaluated on the 3D images from the four channel dataset which were not used for training. The segmentation results for five 3D images each containing 65 sections (in total 325 2D images per channel) were compared. Ground truth segmentations for all images were determined by manual annotation. Table 7.22 shows the results for all methods for the different evaluation metrics.

It turns out that the proposed neural network combined with transferring individual filters performs best for SEG and second best for JI, Dice and Warping Error. Segmentation results for an example image are provided in Figure 7.22. It can be seen that the proposed network performs best. In addition, transferring individual filters improves cell separation. The high SEG and low Warping Error indicates that the proposed model is more suited to correctly merge and split objects.

7.4.2 Unsupervised Domain Adaption for End-to-End Grading of Whole-Slide Images

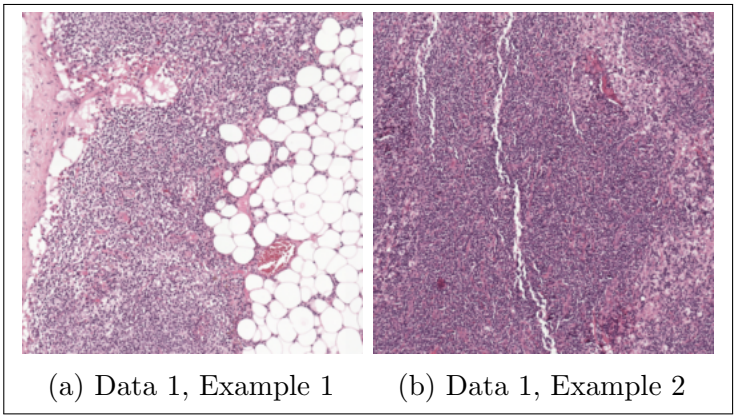
The method presented in Section 6.3 for end-to-end grading of whole-slide images was evaluated using the CAMELYON17 dataset [241]. The method was the only segmentation-free method that participated in the CAMELYON17 challenge [241]. The data consists of WSIs from 100 patients collected from five medical centres in the Netherlands. The WSIs show five lymph nodes per patient and are labelled with a patient level pN-stage and a slide level class. Figure 7.23 shows examples of automatically selected ROIs using the method. For evaluation of the proposed deep learning method, three different experiments were performed. In the first experiment (Standard), state-of-the-art data augmentation (without CycleGAN) is used. In the second experiment (Domain adaptation), CycleGAN was used for domain adaptation to transform all data from different medical centers to the data from one center (Data 1). In a third experiment, CycleGAN was used to create all combinations of transformations (10 GANs) between sources of data. However, the latter approach performed worse, since the network was not able to learn invariance from the very large variation in the dataset. In Figure 7.24 two examples from Data 1 (Figure 7.24a,b) and results of image transfer from Data 3 and 4 to Data 1 are shown. It can be seen that the transformations learned by the CycleGANs well capture the appearance of Data 1. For a quantitative evaluation, the weighted Cohen's kappa score was calculated for each experiment, which was also used in the CAMELYON17 challenge. The weighted Cohen's kappa score measures the inter-rater agreement for exclusive classes.

The results are shown in Table 7.23. It turns out that domain adaptation using CycleGAN yields a significant improvement compared to state-of-the-art data augmentation. The computation time of the method for the classification of a 512×512 image patch was performed on average in 0.04 seconds on an Intel i7-6700K workstation with a NVIDIA Geforce GTX 1070. The computation time for classification of a whole WSI is 0.78 seconds and in total 3.90 seconds for predicting a patient level grade. The forward pass in the proposed method is tenfold faster compared to the original DenseNet.

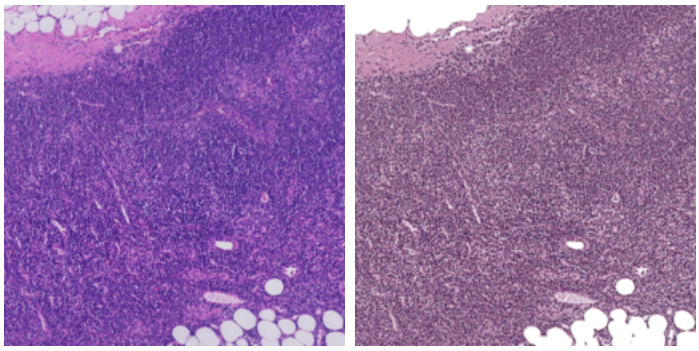


(a) Original image (b) Regions of interest

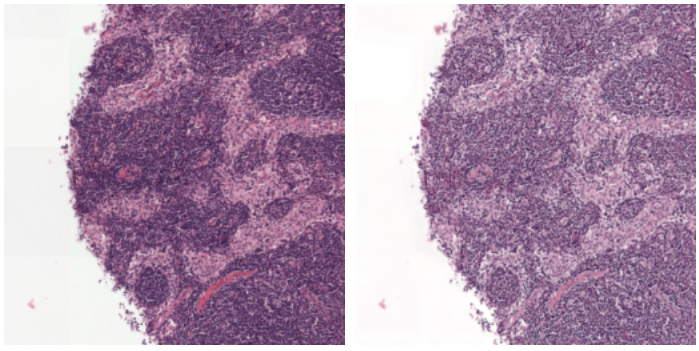
Figure 7.23: Examples of regions of interest automatically selected from a WSI.



(a) Data 1, Example 1 (b) Data 1, Example 2



(c) Data 3 (d) Data 3 transformed to Data 1



(e) Data 4 (f) Data 4 transformed to Data 1

Figure 7.24: Images (512×512 pixels) from different data sources mapped to Data 1.

Table 7.23: Results for state-of-the-art data augmentation and for domain adaptation.

Method	weighted Cohen's kappa
Standard data augmentation	0.114
Domain adaptation	**0.165**

8 Web-Based Microscopy Image Analysis

Since several years, high-content (HC) as well as high-throughput (HT) microscopy and screening techniques have lead to many advances in biology and medicine. This fast emerging technology is transforming cell biology into a big-data driven science [265]. New challenges for data storing, processing, analysis, and interpretation arise due to the huge amount of generated image data. Automated analysis of microscopy images is a main bottleneck [266, 267, 268] and, unfortunately, scaling up workflows is not straightforward. Moreover, there exists a vast variety of microscopy image analysis software with multiple versions for different platforms, which often impedes sharing workflows or reproducing analysis results [269]. Setting up the computation environment for deep learning algorithms is especially challenging. In biomedical research, a short time between the development of a novel microscopy image analysis method and its application has the potential to accelerate research. Therefore, translational efforts between image analysis researchers and biologists are conducted. Developing workflows for a biological or medical image analysis project involves constant exchange between the cooperation partners. The continuous improvement of workflows, algorithms, and simultaneous generation of new data yields challenges for FAIR [270] principles. The FAIR principles are findability, accessibility, interoperability, and reusability [270]. Missing metadata can cause a lack of findability. Impaired accessibility can arise from data or algorithms which can be exclusively used by only one of the partners. General lack of usability and software that is cumbersome to install prohibit reusability. Finally, non-standardized communication between software can prevent interoperability of tools. An infrastructure for sharing resources such as collaboration functionalities with co-workers or cooperation partners, data storage, or computer cluster execution needs to be established. Hence, there are several reoccurring challenges in large scale microscopy image analysis for biomedical research. To cope with these challenges, a web-based system which supports complex scientific microscopy image analysis workflows has been developed and is presented in this chapter. The developed web-based system is complemented with a deployment platform for biomedical image analysis software. The system was evaluated in usability studies and used in research projects with biomedical partners. The work was partially published in [8, 7, 6].

8.1 Workflow Systems for Microscopy Image Analysis

Several scientific workflow management systems (SWMS) [271] exist that facilitate the creation, provision, and maintenance of data analysis workflows (e.g., Galaxy, KNIME, Taverna). SWMS allow scaling of image analyses and enable collaborative data analyses and reproducible science. Moreover, they ease the use of image analysis pipelines and high-performance computing (HPC) environments or a cloud by non-expert users. To maximize the benefit for the scientific community, sharing of data, software, and results should be as easy as possible [272]. Therefore, tool-, data-, and workflow-sharing platforms have to be established. A microscopy image analysis workflow processes acquired images to retrieve quantitative information about a biological experiment. Quantitative information can be determined from the images as a whole or from individual objects of interest in these images (e.g., cells or subcellular structures). Such an analysis can span multiple levels of image resolution and multiple image dimensions depending on the biological question or imaging technique. As outlined in Section 1.1.1, a wide variety of different microscopy imaging modalities exists. In addition, a large range of readouts can be obtained (e.g., global intensity level, cellular and subcellular constellations, colocalization information, cell count, cell shape). Common and central tasks in microscopy image analysis are cell segmentation, counting, and feature extraction. The readout is combined with additional metadata (e.g., position within a plate or well, color channel, time point, or layout information), and specific statistics and visualizations are generated.

In an SWMS, established image analysis toolboxes should be integrated so that they can be used in a standardized way. For all most popular programming environments, there exists at least one microscopy image analysis toolbox [273]. For Matlab, the Matlab Image Processing Toolbox and DIPimage are widely used. Within Python, scikit-image [274] and Mahotas [275] are popular. ImageJ [276] is heavily used in the Java community. Projects like Fiji [266], CellProfiler [277] or Icy [278] use image analysis features of ImageJ. For C++, OpenCV [279], VTK [280], ITK [281], 3DSlicer [282], MITK [283], Ilastik [253], and many more toolboxes are prevalent. Some of the C++ toolboxes offer wrappers for Matlab, Python, and Java. These toolboxes share several basic image analysis features, but also have exclusive advantages. An ideal SWMS would integrate all features of the toolboxes in a meaningful way, achieving interoperability.

Microscopy image analysis toolboxes typically have components for input/output (IO), image processing, image analysis, and visualization. Usually, the toolboxes embed IO libraries to read and write image files. Image analysis toolboxes typically focus on a specific application area and therefore cannot read all common image file types. Moreover, they include custom data structures to optimize image processing and access (e.g., kd-trees, tensors). Image processing algorithms like filtering (e.g., mean filter, Gaussian filter, median filter) for noise reduction and low level feature extraction (e.g., Haralick features, SIFT, HOG, LBP) are used in microscopy image

analysis. The capabilities of the different microscopy image analysis toolboxes differ considerably. Object detection, object segmentation, object classification, object tracking, colocalization analysis, and image registration are common microscopy tasks. There exists a huge variety of different methods to solve these tasks. Moreover, a diversity of implementations and optimization schemes for different use cases are available. Many image analysis methods heavily rely on vector/matrix/tensor operations, which are highly parallelizable (e.g., convolution, rendering). These operations are often accelerated by a graphics processing unit (GPU).

Most SWMS like Nextflow [231], Toil [284], Snakemake [285], or Bpipe [286] do not offer a Graphical User Interface (GUI), being designed for power users and less suitable for biologists and physicians. SWMS like KNIME [287] or Taverna [288] focus on a desktop client which uses local resources. In contrast, the Galaxy platform [289, 290, 230] focuses on a web-based client running in a high-performance computing environment. However, Galaxy, KNIME, and Taverna provide web-based and local clients, and high-performance computing support. The systems have different mechanisms for workflow sharing, and a direct conversion of workflows between the systems is currently not possible. The project MyExperiment [291] enables the exchange of workflows for Galaxy, KNIME, and Taverna. In the following, similarities, differences, and problems of SWMS for large scale microscopy image analysis based on Galaxy and KNIME are described, as KNIME is already established for bioimage analysis and Galaxy is an emerging fully open-source web-based platform for biological data analysis.

KNIME is a platform for data analysis, reporting, and integration [292]. Individual tasks are visually represented as modules or nodes which can be orchestrated to workflows. KNIME consists of an open-source core and commercial extensions that include multi-user functionality, web portal access, or server execution. External tools can either be integrated through a command line call node or integrated as plugin via the Eclipse plugin architecture. For some programming languages (e.g., Java, Python, Perl, R), dedicated nodes can directly execute code snippets. In KNIME, data is represented as a tabular structure and used to pass data between nodes. Each node has a table viewer, but can be replaced by specialized viewers for specific data types. KNIME supports conditional execution and loop structures. Nodes for image handling, conversion, normalization, filtering, labeling, segmentation, or feature extraction along with visualization are available.

Galaxy is a software platform that enables biologists to perform web-based computational analysis [289, 290, 230]. It supports the whole workflow of data storing, data processing, data analysis, data visualization, and data publishing. Galaxy allows processing of jobs in the cloud or on computer cluster systems like Terascale Open-source Resource and QUEue Manager (TORQUE) or Slurm. Galaxy was developed for genome analysis, and for DNA sequence analysis in particular. The main Galaxy client is a web-based graphical user interface. In addition, the Bioblend library can be integrated into third-party clients to communicate with the Galaxy

API. Galaxy can be used through a public Galaxy server (https://usegalaxy.org), cloud installations, or locally. Administrators can install tools and workflows to Galaxy using the ToolShed (https://usegalaxy.org/toolshed). The ToolShed provides a web-interface to install not only tools and workflows, but also the required dependencies automatically. Dependencies can be automatically installed from the ToolShed itself, Conda repositories, Docker repositories, and PyPA pip. To provide long term availability of dependencies, the Galaxy community launched the Cargo Port package repository, which is archiving source code of registered ToolShed tools (https://depot.galaxyproject.org/software/). Over 7000 tools are available in the official ToolShed. Galaxy also supports the Common Workflow Language (CWL) [272] and the EDAM ontology [293] to be interoperable with workflows created in other SWMS or databases providing ontology annotated data. In addition to workflows, Galaxy offers the concept of interactive environments (GIE) [294], which allows for interactive exploratory data analysis. For example, Jupyter and RStudio are available for programming, Neo4J for graph analysis, and Ethercalc for modifying tabular data. In Galaxy, tools are executed via command line calls and XML-based tool descriptions are used to automatically generate dedicated user interfaces without the need of writing complex plugins. So called tool runners are used to schedule the computation jobs for local, cloud, or cluster execution. The Pulsar component can transfer required files, scripts, configuration files, and results over Secure Shell (SSH), if no direct communication to the computation infrastructure is possible. Data can be uploaded by the user to Galaxy (e.g., web-interface, API, FTP) or retrieved from a public or private database (e.g., EBI SRA, UC,SC Main, EuPathDB). In 2019, more than 100 publicly accessible servers were available (https://galaxyproject.org/galaxy-project/statistics/). The Galaxy community has a strong focus on training. The Galaxy Training Network [295] organizes training materials, workshops, and conferences like the annual GCC highlighting new Galaxy capabilities and research which is using the Galaxy platform.

With both Galaxy and KNIME, design and execution of data processing workflows is straightforward and easily comprehensible due to the graphical representation of the workflow structure. However, the SWMS differ in their scope and applicability (Table 8.1). KNIME focuses on tight coupling [296] with tools and hardware and a finer tool granularity. Tools are either directly integrated within the KNIME framework as plugins or by the user via command line calls. The performance of KNIME strongly depends on the computation environment of the user. In the Galaxy platform, tools are always executed via command line calls. Therefore, Galaxy focuses more on a loose coupling [296] with tools. Galaxy also has loose coupling with computing resources, since command line calls can also be executed easily on a remote system. In Galaxy, cutting edge tools and algorithms can be easily made available, resources can be assigned as required. The usage of Galaxy is straightforward and the user does not require an installation of a local client or knowledge on using cloud or cluster resources. KNIME can only resolve tool dependencies using the Eclipse Integrated

Table 8.1: Comparison of KNIME and Galaxy characteristics.

Characteristic	KNIME	Galaxy
Code license	GPL v3.0	AFL v3.0
Coupling with tools	tight	loose
Preferred tool granularity	fine	coarse
Stepwise workflow execution	possible	possible
Workflows executable in other SWMS	impossible	via CWL
Client type	desktop client	web-based client
Required computation resources	client-side	server-side
Cluster support	commercial extension	native

Development Environment (IDE)-based system provided by KNIME. In contrast, Galaxy can use several dependency resolvers like Conda or Docker. Thus, Galaxy can benefit from packaging efforts of multi-purpose repositories like Conda. As pointed out, both workflow systems can solve similar tasks in data analysis in their respective environment, but are complementary in implementation into scientific workflows. KNIME is focused on providing a platform for orchestration of tools at the side of the user, where the user maintains the KNIME installation, tools, workflows, and infrastructure. Galaxy is focused on providing a platform for remote data analysis where computer science experts provide tools, workflows, and infrastructure to less experienced users. Hence, Galaxy is more suitable to support the workflow for image analysis research projects presented in Section 1.1.2 by providing a platform for the collaboration of image analysis experts and biologists. However, Galaxy lacks the broad integration of image analysis tools in comparison to KNIME.

8.2 Deployment of Biomedical Data Analysis Software

Installation of cutting edge data analysis software can be cumbersome due to the focus on method development in research instead of software engineering. Therefore, a lightweight solution to help researchers to manage software dependencies and distribute software is required. The Conda package manager (https://conda.io) is a cross-platform package manager. It is programming ecosystem and operating system agnostic. Conda builds the software packages in a controlled environment and provides the resulting binary software artifacts. The software is build to work in an isolated environment and can therefore be installed in multiple versions and combinations for different projects. Conda is nowadays integrated into data analysis software for reproducible science like Galaxy [230], bcbio-nextgen (https://github.com/chapmanb/bcbio-nextgen), and Snakemake [285]. Bioconda (https://bioconda.github.io) is a Conda software channel dedicated for the life sciences [7]. The Bioconda project [7] provides over 6,000 software packages.

Moreover, each Bioconda package is also provided in containers through Docker and Singularity via the closely collaborating Biocontainers project [297]. The Bioconda project has become a backbone of bioinformatics infrastructure with over 6.3 million downloads until 2018 and the largest software repository for life sciences [7].

As a basis for Galaxy Image Analysis (see below), within this thesis image analysis software was integrated into Bioconda and the formation of a Bioconda community particularly for image analysis was supported [7]. More than 15 image analysis related packages with over 200,000 downloads until the end of 2019 (https://anaconda.org/bioconda) have been included into Bioconda.

8.3 Galaxy Image Analysis

Galaxy is widespread in the field of biological data analysis and easy to use by non-experts via the web-client. Therefore, in this thesis Galaxy was chosen as platform and extended for the use in the field of microscopy image analysis. However, Galaxy has been developed for genomic data, which has different requirements to an SWMS compared to microscopy image data (Table 8.2). For example, the number of image files is multiple orders higher in microscopy image analysis compared to genome analysis. This poses the challenge of handling large amounts of metadata. In addition, in microscopy imaging, metadata like the acquisition channel, plate position, or well is often encoded in the filename which has to be parsed within Galaxy. Imaging data also lack standardization in file type and metadata encoding. Reading out this information can be even more complex, as images are often stored in container-based data types (e.g., TIFF, VTK, PSD, HDF5). This is also a main difference to genomic data analysis, where less data types are used. Even wrapping existing microscopy image analysis tools into Galaxy can be more difficult than wrapping genomic data analysis tools. Typically, microscopy image analysis tools have dozens of dependencies with very specific libraries. Moreover, they are mainly controlled via a graphical user interface (GUI). In genomic data analysis, it is more common to use a command line interface (CLI). In addition, the field of genomic analysis already uses widespread deployment structures like Bioconductor [298], Biocontainers [297], or Bioconda [7]. A GUI is not only an obstacle for automation, but also often requires a processing node equipped with a graphics card, which is not always the case in a high-performance computing environment. Obviously, the resulting data from applying microscopy image analysis workflows require visualization tools enabled for the web. These visualization tools have to cope with the additional challenge that image files are generally too large to be transferred and rendered in the browser. Therefore, server-side tiling and rendering in combination with streaming to a browser-based thin client is necessary.

These requirements were tackled in this thesis by contributing to the Galaxy platform. Support for more than 10 microscopy image file types, support for datasets with thousands of images (e.g., used in phenotype screenings), extensions for image

Table 8.2: Comparison of genome data and image data.

Characteristic	Genome data	Image data
File size	large	medium
File count	small	large
File types	few with proprietary extensions	proprietary and open container-based types
Metadata	own file types (many derivations)	implicit encoded / proprietairy file types
Dataset standardization	medium	low
Visualization	1D/2D	2D/3D/ +channels/ +time
Galaxy tools available	many	very few
Tool interface	mostly via Command Line Interface	mostly via Graphical User Interface

data visualization, and over 50 image analysis tools were contributed. The work has been published in Wollmann et al. [8]. In addition, workflows for Galaxy for specific use cases were developed and published in [6]. Moreover, training material for Galaxy Image Analysis was provided using the Galaxy Training Network [295]. In the following, an example workflow for cell segmentation is presented. In this workflow, the metadata is extracted from the image file and written to a tabular file, the image is pre-processed, the cells are segmented, counted, and finally, the metadata and the cell counts are merged into a unified table. Instead of calculating global features for the whole image, object-wise features like mean intensity, area, or major axis length can be computed. In case the relation of objects in two image channels is important, a co-localization analysis can be performed. The presented workflows are easily executable, shareable, and extendable through the Galaxy workflow management capabilities. Galaxy interactive environments (GIEs) enable the user to interactively explore data within the Galaxy platform. For example, for editing a readout of a screen, which is stored in a tabular file, the Ethercalc (Figure 8.2a) GIE can be

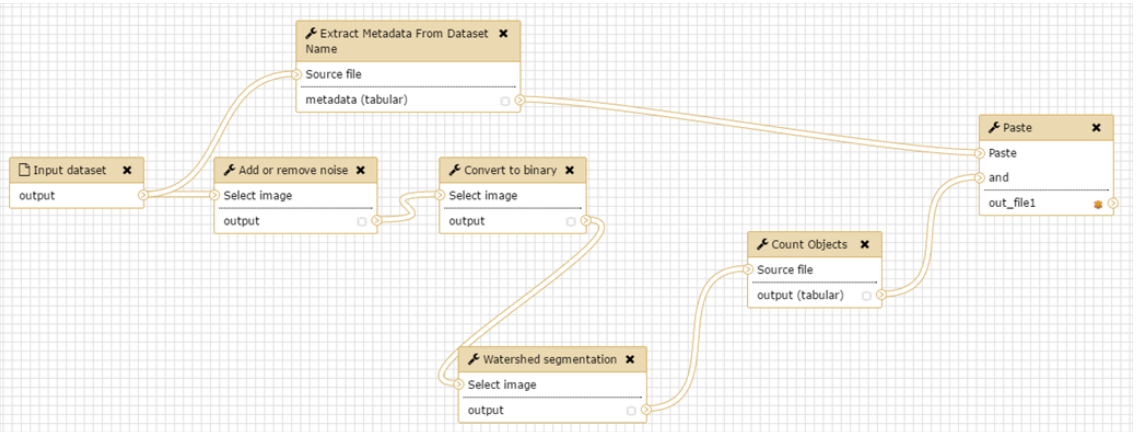
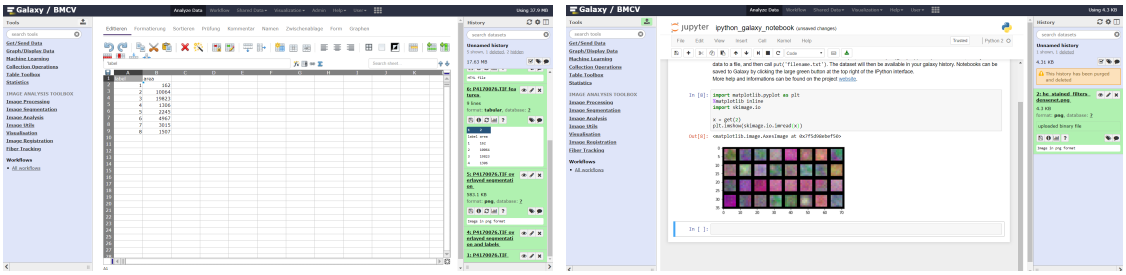


Figure 8.1: Galaxy Image Analysis workflow for cell segmentation and counting.

used. More advanced users can use the Jupyter (Figure 8.2b) GIE for custom data analysis in Python or R. However, visualization of images is not possible. Especially multi-dimensional data (e.g., 3D images, videos) are frequent in microscopy image analysis and cannot be directly visualized in the browser. Therefore, a GIE based on ParaView [299] was developed (Figure 8.3). The GIE performs server-side off-screen rendering using the Mesa 3D graphics library, which supports GPU- and CPU-based rendering. A JavaScript-based client communicates with the rendering engine using the WebSocket protocol. However, for large images like histological slides, this visualization technique is inefficient. Typically, large images are sliced and delivered using lazy loading to the browser (e.g., OpenStreetMap, Google Maps). Therefore, a GIE based on a Deep Zoom implementation of OpenSlide [300] was developed (Figure 8.4).

8.4 Usability Evaluation of Galaxy Image Analysis

To measure the applicability of Galaxy Image Analysis, several usability studies were conducted within this thesis. According to EN ISO 9241-11, *usability* is defined as the product of effectiveness, efficiency, and satisfaction of a system in a specific context of use [301]. *Effectiveness* describes the ability of the user to complete



(a) Ethercalc GIE (b) Jupyter GIE

Figure 8.2: GIEs for explorative data analysis.

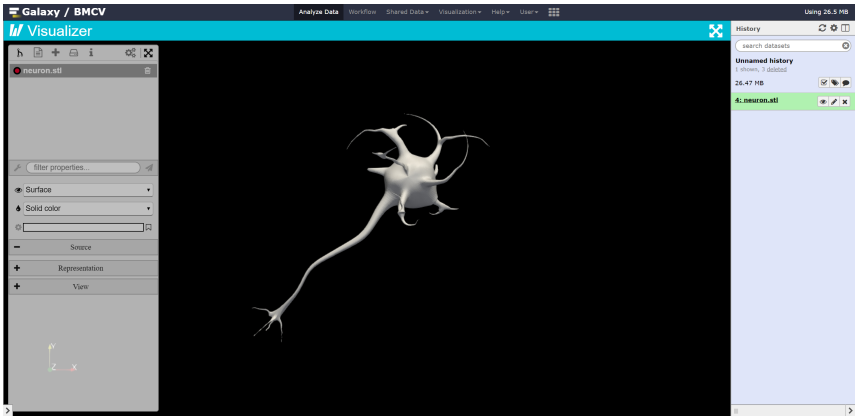


Figure 8.3: GIE for server-side image rendering and visualization based on ParaView.

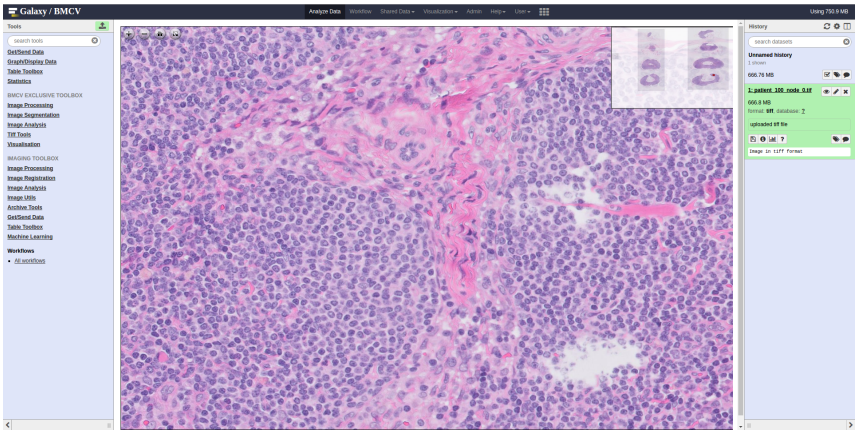


Figure 8.4: GIE for WSI visualization based on OpenSlides Deep Zoom implementation.

tasks and achieve specific goals [301]. The effort the user has to spend is described by the *efficiency* [301]. The *satisfaction* characterizes the user's perceived ease of use [301]. In addition, *learnability* describes the time span that is required so that a user can efficiently use the system [301]. These factors can be quantified by usability metrics [301]. The EN ISO 9241-11 proposed several methods such as expert reviews, interviews, observations, and questionnaires to measure these factors [302]. Questionnaires are a quick tool to measure the overall usability of a software. Brooke et al. introduced the System Usability Scale (SUS) [303]. The SUS questionnaire consists of 10 questions on the Likert scale from one to five. The Likert scale is a bipolar scale and is able to measure positive or negative responses to a given statement [304]. The score of every second answer is reversed and all answers in the questionnaire are then summed up in the SUS score [305].

In the conducted studies within this thesis, the SUS was adapted with suggestions from Finstad and Bangor et al. [306, 307], who suggest to change "cumbersome" to "awkward", to make it better understandable. Moreover, "system" was changed to "software". According to Lewis et al. [308], changing "system" to a different term has no measurable effect on the SUS score. The questionnaire was combined with a master data section consisting of age, gender (options: male, female, divers), profession, and software usage (options: never, occasionally, monthly, daily) to better understand the cohort and identify confounding bias factors. According to Nielsen [309], five test participants achieve an optimal cost-benefit ratio to identify major usability issues using a quantitative study. Rubin et al. [310] recommend using at least eight participants to have some buffer and strengthen statistical significance. The target user group of the software system in this thesis are biomedical researchers in a university environment. Therefore, students and researchers with biological background were recruited. To form a baseline, a SUS study was performed in 2016 for the systems ImageJ, KNIME, and Galaxy Image Analysis. In 2018, a controlled study on Galaxy Image Analysis was performed to identify usability issues. As part

Table 8.3: SUS questionnaire usability studies for ImageJ, KNIME, and Galaxy.

Platform	Year	M/F/D	Age	SUS	US	LS
ImageJ	2016	6/11/0	29.2 ± 4.8	64.55 ± 21.06	67.61 ± 21.80	52.27 ± 23.60
KNIME	2016	7/7/0	30.7 ± 5.6	63.21 ± 11.96	66.52 ± 10.93	50.00 ± 27.00
Galaxy	2016	6/11/0	28.1 ± 5.2	50.68 ± 12.55	54.55 ± 12.91	35.23 ± 30.01
Galaxy	2018	5/7/0	22.7 ± 1.1	77.14 ± 16.98	77.68 ± 17.06	75.00 ± 23.94
ImageJ	2019	6/7/0	30.3 ± 4.0	65.71 ± 19.02	69.20 ± 16.18	51.79 ± 32.62
KNIME	2019	5/8/0	29.6 ± 4.2	52.81 ± 23.62	55.86 ± 24.40	40.62 ± 25.66
Galaxy	2019	5/8/0	29.6 ± 4.2	63.12 ± 23.21	64.06 ± 27.60	59.38 ± 19.76

of the study protocol, a structured tutorial on the respective software systems were provided to the study cohort by an expert. Then, the participants were instructed to perform different workflows on their own. Finally, a questionnaire containing the SUS had to be filled and an interview was conducted to gather diagnostic information on usability problems. Finally, in 2019, a SUS study on ImageJ, KNIME, and Galaxy Image Analysis was conducted to measure any change in usability over the project period. The results of the questionnaire are shown in Table 8.3. In this study, the participants were only introduced into Galaxy and learning to use the system and the perceived usability are mixed in the SUS. Therefore, the usability (US) and learnability (LS) factors of the SUS identified by Lewis et al. [308] were calculated. All users were able to complete their given tasks. For ImageJ, the interface design and the complicated menu structure was criticized. Moreover, scaling up processing is only possible using scripting. Participants in the KNIME study did not like the limited amount of tools for image analysis. Moreover, they complained about software crashes. In the 2016 Galaxy study, users had problems with software performance, overloaded and non-intuitive menus, and the lack of tools. This changed tremendously in the 2018 Galaxy study. In this study, it was noticeable that specific tool options were not user-friendly enough for users to understand their functionality. Participants still had some issues with the general usage of Galaxy due to overloaded and non-intuitive menus. In the 2016 study, higher SUS, learnability, and usability were measured for ImageJ and KNIME than for Galaxy. In comparison, in 2018 and 2019, Galaxy had improved SUS, learnability, and usability. It is noticeable that the learnability increased more than the usability. This results probably from an improved tutorial structure compared to 2016, and not from the system itself. The cohorts for the studies were relatively similar in terms of age, gender, profession, and previous knowledge. Only in the 2018 Galaxy study, all participants were students and their mean age was lower than in the other studies. Age can have an influence on reported usability [311, 312] due to different levels of digital nativeness. Therefore, the Pearson correlation coefficients between age and SUS (-0.08), US (-0.04), and LS (-0.14) were calculated. Since the correlation is not large, the overall conclusion should remain, but the quantitative result is maybe biased to an overestimation of usability in the 2018 study. Moreover, the Pearson correlation coefficients between

Table 8.4: SUS questionnaire usability scores only for participants trained in the use of the respective software.

Platform	Year	M/F/D	Age	SUS	US	LS
ImageJ	2016	6/9/0	30.2 ± 4.8	66.17 ± 17.70	66.88 ± 19.65	63.33 ± 22.89
KNIME	2016	3/1/0	33.2 ± 10.5	65.00 ± 10.21	66.41 ± 9.33	59.38 ± 18.75
Galaxy	2016	2/0/0	28.5 ± 2.1	63.75 ± 5.30	59.38 ± 8.84	81.25 ± 8.84
Galaxy	2018	0/0/0	-	-	-	-
ImageJ	2019	5/6/0	28.3 ± 4.6	71.59 ± 12.96	75.28 ± 11.13	56.82 ± 27.02
KNIME	2019	1/3/0	27.2 ± 5.7	46.88 ± 22.49	49.22 ± 23.99	37.50 ± 22.82
Galaxy	2019	0/4/0	31.5 ± 4.2	79.38 ± 14.91	81.25 ± 18.58	71.88 ± 18.75

software usage on a scale from never (0) to daily (3) and SUS (0.22), US (0.20), and LS (0.18) were calculated. It can be concluded that software usage is a potential confounding bias factor for all scores. The introduction for the study was relatively brief and ImageJ, KNIME, and Galaxy have a different level in software feature complexity. Thus, only the scores for participants who are already trained in the use of the respective software are reported in Table 8.4. Due to considerations of previous experience in the respective systems, the variance of the scores was reduced. It can be see seen that in 2016, the usability for participants used to the software was similar for all systems. However, in 2019, SUS, usability, and learnability was highest for Galaxy. The results should be handled with care, since the size of the subgroups in Table 8.4 is small compared to Table 8.3. In general, it can be concluded that the applicability of Galaxy Image Analysis increased during the project.

8.5 Applications of Galaxy Image Analysis

In addition to the general workflows described in Section 8.3, use-case specific workflows have been developed within this thesis, which are presented in this section.

8.5.1 Quantification of Viral Spread in Cells

Several pathogenic viruses are capable of spreading within a host by multiple modes of transmission (e.g., cell-free, cell-to-cell). In various viral infections, cell-to-cell spread seems to play a dominant role, such as those caused by hepatitis C virus (HCV) or human immunodeficiency virus (HIV) [313]. However, the spread of viruses by each of these modes of transmission has not been quantified so far. Therefore as part of this thesis, quantification of viral spread is conducted. The data is used for mathematical modelling by the Graw lab. Counts of infected cells from 2D and 3D in vitro cultures are used to test the mathematical models [313]. Due to the large amount of acquired images and the high number of visible cells, automatic image analysis is required. Imaging was performed by scientists from the Uprichard lab at the Loyola University of Chicago, and mathematical modelling by scientists from

the Graw lab at Heidelberg University. An image analysis pipeline was developed and integrated into Galaxy Image Analysis (Section 8.3).

The workflow performs segmentation of the foci using a deep learning method (Figure 8.5). If hardware requirements for the deep learning method are not met, a segmentation method based on color thresholding can be used. Subsequently, objects which are close are merged in the label map. Finally, an interactive tool (Figure 8.6) can be used to estimate counts of cells based on mean area. The user can aggregate the counts of multiple foci, which is important to analyse spatially non-connected, but simultaneously infected cells. Segmentation based on deep learning uses the ASPP-Net (Section 6.2). Example results for color thresholding and ASPP-Net along with the corresponding ground truth are shown in Figure 8.7. By using Galaxy Image Analysis, a cutting edge image analysis pipeline and computational resources has been easily provided. Moreover, on the same platform, initial analysis could be performed by the wet lab that acquired the images, and downstream analysis by a mathematician.

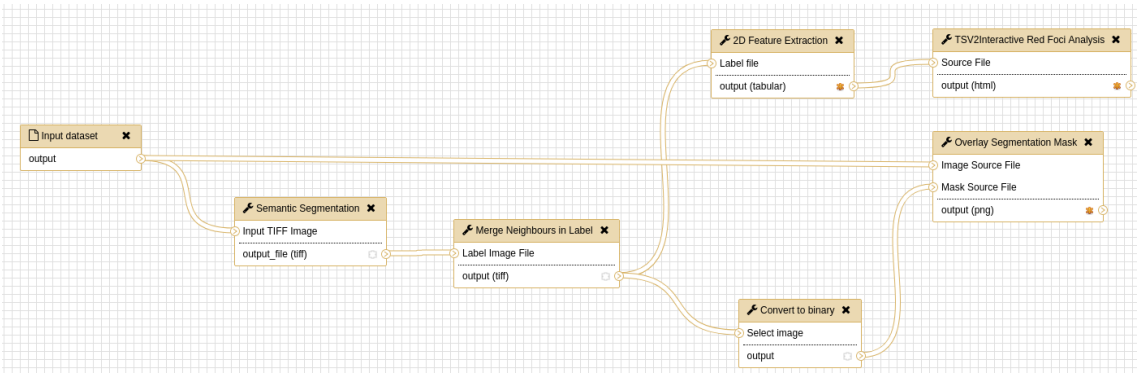


Figure 8.5: Galaxy Image Analysis workflow for foci analysis.

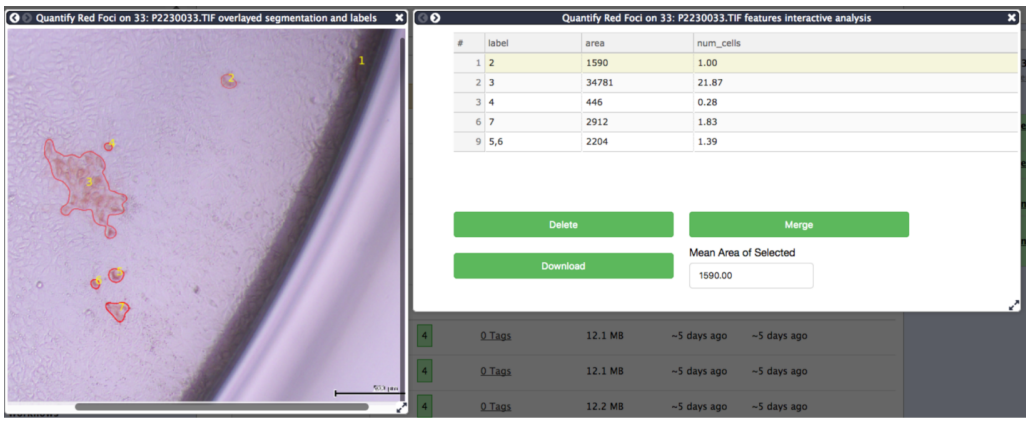
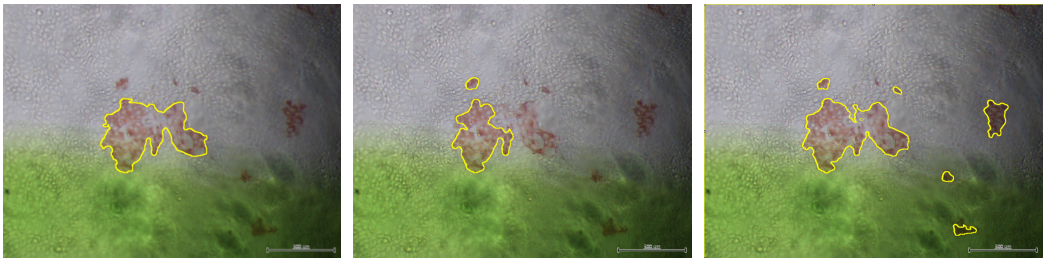


Figure 8.6: User interface for interactive merging foci segmentation masks within the foci analysis workflow.



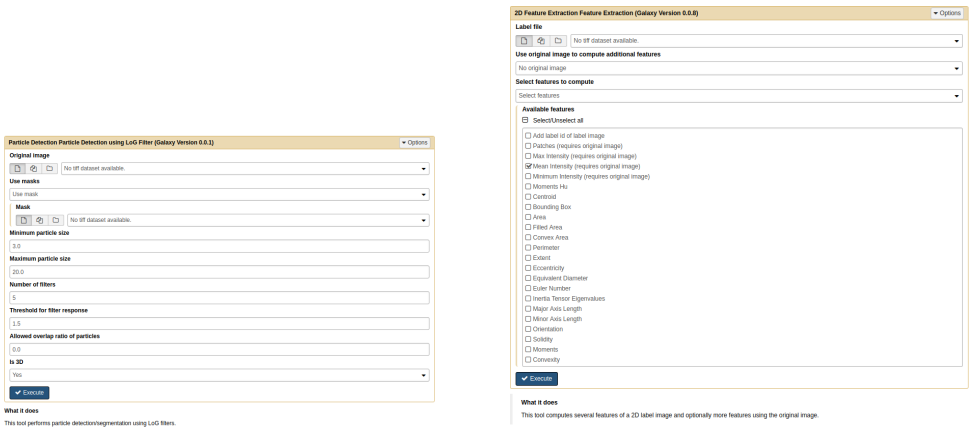
(a) Ground truth (b) Color thresholding (c) ASPP-Net

Figure 8.7: Example original and annotated image showing infected cells.

8.5.2 Quantification of Neurons in 3D Brain Tissue Images of Mice

Alzheimer's Disease is the most prevalent neurodegenerative disease in the elderly. It is mainly characterized by extracellular senile plaques, composed of Amyloid-β (Aβ), and intracellular neurofibrillary tangles (NFTs) that consist of hyperphosphorylated tau protein [314]. Neurotoxic fragments and neuroprotective and neurotrophic fragments are produced during processing amyloid precursor protein (APP) [315, 316]. Overexpression of the neuroprotective fragment can ameliorate the loss of inhibitory interneurons. Different antibodies are used by the U. Müller lab to analyze the inhibitory interneurons in the hippocampus of mutant mice.

Within this thesis, an image analysis pipeline was developed and integrated into Galaxy Image Analysis (Section 8.3). Tools for particle detection and feature extraction are leveraged to quantify stain intensity for detected interneurons in specified regions of interest (Figure 8.8). The performance of the detection method was benchmarked on a sample parvalbumin- and a SST-stained image (from U. Müller lab). The results are shown in Table 8.5.



(a) Particle detection tool (b) Feature extraction tool

Figure 8.8: Galaxy user interface for configuring neuron quantification tools.

Table 8.5: Quantitative evaluation of the neuron detection approach.

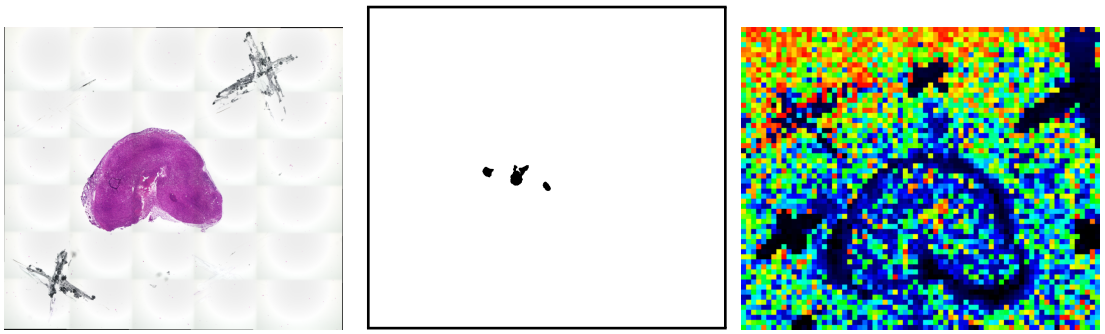
	Parvalbumin	SST
Precision	0.605	0.803
Sensitivity	0.629	0.917
F1	0.617	0.856
RMSE	8.144	7.155

8.5.3 Joint Analysis of MALDI and H&E Tissue Images

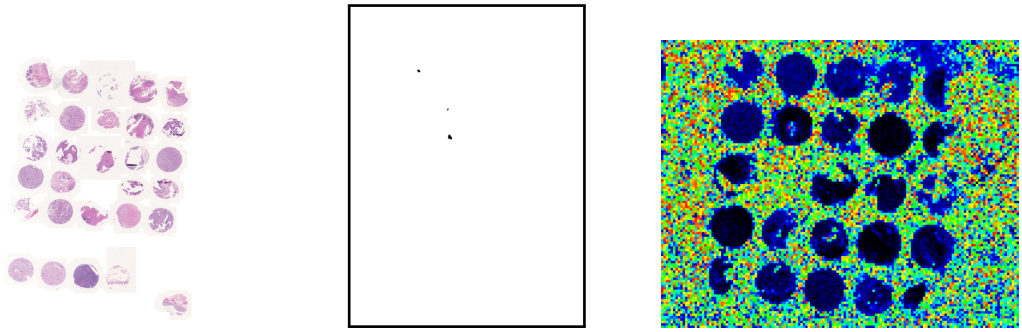
Analysis of mass spectrometry imaging (MSI) yields a broad range of biological and clinical data and can be used for spatial proteomics [317, 318, 319, 320]. The most common ionization sources are matrix-assisted laser desorption/ionization (MALDI), desorption electrospray ionization (DESI), and secondary ion mass spectrometry (SIMS). Time of flight (TOF) sensors and ion traps are used to quantify mass. The variety of MSI applications hinders harmonization and standardization of MSI protocols. Recently, efforts to develop standardized sample preparation protocols have been made [321, 322, 323, 324]. To overcome problems with accessibility of software and computing resources, standardization, and reproducibility, MALDI/H&E data analysis tools and workflows for Galaxy Image Analysis were established and published in [6]. A reoccurring task is the selection of specific ROIs in the MALDI image for further analysis. However, annotation on the MALDI image is not feasible due to lack of visual features. Therefore, pathologists annotate the ROIs on H&E-stained whole-slide images (WSIs) or tissue microarrays (TMAs) (Figure 8.9). Mapping the ROIs from the coordinate system of the H&E-stained image to the MALDI images requires registration. Therefore, in this thesis an algorithm for automatic registration has been developed. The full registration workflow is shown in Figure 8.10. The MALDI image is scaled and registered with the H&E-stained image. The estimated transformation matrix is used to transform the ROIs to the coordinate system of the MALDI image. Coordinates of the ROIs are extracted for further analysis and an overlay is generated for visual assessment of the registration performance. If the user notices by visual assessment of the registration result that the quality of the automatic registration is not sufficient, a backup approach for registration using manual annotated landmarks and random sample consensus (RANSAC) [325] can be used, which has also been implemented.

Due to the large variety of visual appearance of objects in MALDI and H&E images, the automatic registration approach segments the tissue in both modalities first, and registers the segmentations afterwards. Segmentation is performed by color thresholding along with post-processing using morphological operations. Registration is performed in two steps. In the first step, a coarse registration is performed by feature-based registration using the centroids of detected objects. Matching of centroids is performed by the Hungarian algorithm [242]. The Histogram of Oriented Neighbors is used to eliminate outliers. As part if this thesis, the Histogram of

Oriented Neighbors is developed to yield a translation, rotation, and scale invariant descriptor of a centroid in a grid of centroids. The descriptor is calculated by accumulating the angle between the object and its neighbors into a histogram. To achieve invariance, the values in the histogram are rotated, so that the most frequent occurring angle is in the first bin. The Euclidean distance between the two histograms



(a) H&E WSI (b) WSI ROI (c) MALDI WSI



(d) H&E TMA (e) TMA ROI (f) MALDI TMA

Figure 8.9: Example H&E-stained and MALDI WSI and TMA images along with ROIs annotated on the H&E-stained images.

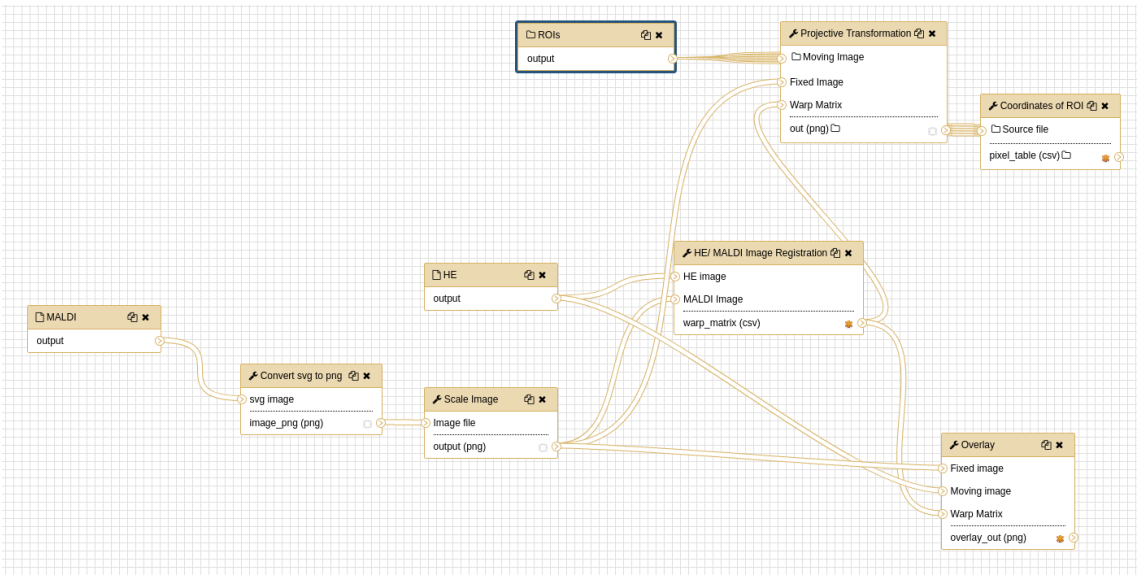


Figure 8.10: MALDI and HE image registration workflow.

Table 8.6: Quantitative evaluation of the automatic H&E and MALDI registration approach.

	H&E	TMA
RMSE / px	2.07 ± 0.36	6.20 ± 1.70
Minimum distance / px	1.03 ± 0.32	2.30 ± 0.60
Maximum distance / px	2.65 ± 0.33	10.0 ± 2.60

is thresholded to eliminate outliers. The transformation matrix is estimated by the iterative closest point (ICP) algorithm [326, 327]. To handle large rotations and escape local minima, registration is performed multiple times with different initial rotations. The best fit is used as result. In the second step, a fine registration is calculated with intensity-based registration. A normalized distance transform of the segmentation masks is calculated and registration is performed using phase correlation [328]. An example registration result is shown in Figure 8.11. The maximum acceptable error range for the clinicians in this application is three pixels. Therefore, a quantitative evaluation has been conducted on three WSI/MALDI and six TMA/MALDI image pairs (from Schilling lab). Ground truth landmark pairs have been selected by an expert and the deviation after registration has been measured. The results for root mean squared error (RMSE), minimum, and maximum landmark distance along with their standard deviation are shown in Table 8.6. It can be seen that for the dataset the registration performance for H&E-stained images is sufficient, but the performance of TMA registration should be improved in follow up work.

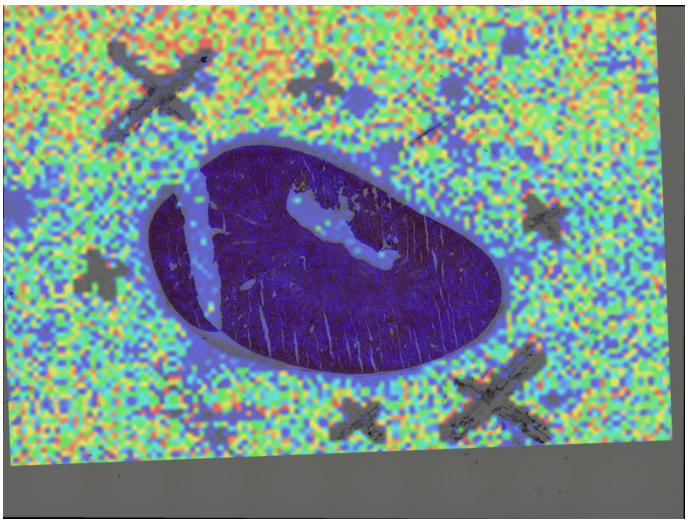


Figure 8.11: Example result for the registration of an H&E image with a MALDI image.

9 Summary and Outlook

In this thesis, different challenges for successfully using deep learning in high-content microscopy image analysis have been considered. Novel deep learning-based methods specialized for major tasks of microscopy image analysis such as detection and segmentation have been proposed. Dataset-specific challenges have been addressed, for example, the automatic optimization of the hyperparameters of biomedical image analysis pipelines, and the lack of training data using techniques like data augmentation and transfer learning. Furthermore, a platform for web-based deployment of these cutting edge computer vision methods in a research environment using large scale computing infrastructure has been presented. The proposed approaches were evaluated qualitatively and quantitatively in different experiments, biomedical image analysis challenges, and biomedical research projects. The presented approaches contribute to the biomedical community by supporting the workflow of research projects that include image analysis. Below, the proposed approaches are summarized, limitations are discussed, and possible directions for future work are described.

9.1 Summary

In this section, the contributions of this thesis are summarized.

Detection in Microscopy Images

- A novel deep learning method *DetNet* for particle detection in fluorescence microscopy images was proposed. Compared to existing deep neural networks, the number of parameters is significantly reduced. The proposed method can cope well with particles of different shapes.

- A new efficient method for mitotic cell detection in histopathology images was proposed which combines Deep Residual Networks with Hough voting. The *Deep Residual Hough Voting* network architecture was optimized for fast factor disentangling, resulting in relatively low computation time. Since the proposed approach requires only cell centroids for training, generation of training data is relatively easy. The centroids can be determined, for example, from bounding box annotations or specification of a point. Since the proposed method predicts the location of centroids, coping with overlapping objects is

facilitated. The proposed method is relatively general and does not require specific pre-processing or post-processing. Thus, the method is applicable to detect diverse objects in different types of image data.

- An automatic method for breast cancer scoring based on whole-slide images (WSIs) was presented. The proposed method utilizes a neural network for detecting mitotic cells, which are aggregated and used in a shallow decision tree to score the tumor. The algorithm can process WSIs in approximately five minutes on a standard workstation by using an attention mechanism and by classifying large image regions at once. The trade-off between statistical power and speed can be optimized by varying the number of regions in WSIs where mitotic cells are counted. The proposed mitotic cell detection method is trained on centroids, which are relatively easy to determine.

- The *Deep Consensus Network*, a new deep neural network for centroid-based object detection in microscopy images, was introduced. The proposed method employs a combination of an Feature Pyramid Network (FPN) with a differentiable voting space. The method relies on a consensus of object detection hypotheses and uses a novel Centroid Proposal Network (CPN) to predict hypotheses at multiple image scales. Advances from object detection methods using bounding boxes were exploited for centroid-based object detection. Therefore, the proposed method can utilize standard convolutional neural network architectures in combination with other techniques like Feature Pyramid Networks, anchors, and Non-Maximum Suppression (NMS). To increase the robustness of training, a novel anchor regularization scheme was introduced. The spatial structure of the voting space is exploited to improve centroid-based NMS, which significantly reduces the algorithmic complexity. Based on a thorough analysis of existing loss functions for neural networks and their relationships, a novel loss function for object detection based on Normalized Mutual Information (NMI) was derived within a Bayesian framework. This loss function copes with class imbalance and also emphasizes correlation. Compared to previous voting-based methods, the proposed network is trained end-to-end and from scratch without requiring pre-training.

Segmentation in Microscopy Images

- The *ASPP-Net* for cell segmentation was introduced. The method is based on an hourglass-shaped deep neural network with an atrous spatial pyramid pooling (ASPP) block to increase the receptive field.

- The *GRUU-Net* was presented. A new deep neural network which integrates convolutional neural networks and gated recurrent neural networks. The proposed method combines a convolutional Gated Recurrent Unit (GRU) with a dense hourglass-shaped U-Net architecture for iterative, multi-scale feature

aggregation and refinement. The proposed network has much less parameters (0.7 M) compared to a U-Net (1.9 M) and a Deconvolution Network (1.1 M). To increase the robustness of the training and improve segmentation, a novel normalized focal loss for momentum-based optimizers was introduced. The proposed focal loss did not only improve the segmentation result of the proposed network, but also the result of other deep neural networks such as the U-Net. The network was trained end-to-end from scratch using relatively few example images. Compared to previous deep learning approaches, all layers in the proposed model have access to features from all previous layers over a common memory at full resolution to improve the sharing of information and better gradient flow. A common feature representation over all scales, which introduces skip connections between all layers, is used to reduce overfitting when using only a limited number of training samples. A distributed scheme for data augmentation and optimized training of the proposed GRUU-Net was also presented.

Hyperparameter Optimization

- A new hyperparameter optimization framework named *HyperHyper* was proposed, which has several advantages compared to existing optimization frameworks. While existing frameworks include only a limited number of optimization methods, HyperHyper comprises more than 40 different optimizers, which was achieved by a modular architecture that separates the sampling and optimization strategy. Using two pipelines for segmentation of cell nuclei in tissue microscopy images, the impact of separating sampling and optimization was demonstrated. Furthermore, HyperHyper includes an integrated scheduler and job wrapper to deal with different cluster computing infrastructures and pipelines written in various programming languages. In addition, it was shown that an infimum projection of the loss function can provide insights into the structure of the optimization problem. This might also help in selecting an optimal sampling and optimization strategy for similar optimization problems.

Transfer Learning for Microscopy Image Data

- The proposed *ASPP-Net* for cell segmentation was combined with a novel approach for transfer learning to use trained networks from one-channel data to multi-channel data. The method improves performance when using only a limited number of training samples with multi-channel information.

- A novel, fast, and automatic deep learning method for patient level breast cancer grading using lymph node whole-slide images (WSIs) was presented. The proposed method requires only slide level annotations. To reduce the effect of the variability of data from different data sources (medical centers), a

generative model based on CycleGAN for domain adaptation was introduced which is trained without supervision and does not require paired data. The proposed method utilizes a region of interest selection and a densely connected deep neural network (DenseNet) to perform sparse classification. The method determines a patient level grade based on five WSIs in about four seconds on a standard workstation.

Web-Based Microscopy Image Analysis

- *Galaxy Image Analysis*, a web-based platform for image analysis using Galaxy was introduced. The platform provides tools, workflows, and visualizations for microscopy image analysis to users with no computer science background. Deployment of cutting edge data analysis software and compliance with FAIR (findable, accessible, interoperable, reusable) principles is eased significantly by establishing image analysis software into Bioconda, which emerged to the largest biomedical software repository. The Galaxy Image Analysis platform can be installed locally or centrally on a cloud or a high performance computing (HPC) system. Therefore, Galaxy has the potential to accelerate research by supporting the image analysis workflow in complex scientific projects.

9.2 Outlook

The following research questions could be addressed in future work.

- The detection approaches Deep Residual Hough Voting and Deep Consensus Network leverage consensus by multiple predictions to improve the final result. The proposed components like the Normalized Focal loss, the Normalized Mutual Information (NMI) loss, anchor regularization, or the Consensus Voting layer could also be used for other deep learning applications.

- GRUU-Net uses a full resolution branch to store cues for performing segmentation. This memory module could be exploited in other tasks and combined with other memory module-based methods like in [329]. Training data demands could be reduced by leveraging few shot learning or self-supervision techniques (e.g., [330, 331]).

- Currently, the hyperparameter optimization framework HyperHyper only supports transfer of hyperparameter distributions to a new hyperparameter study. Deeper exploration of transfer learning in hyperparameter optimization could be beneficial for reducing computed and required training data.

- The proposed transfer learning method supports transferring a network trained on less color channels to a network with more color channels. The opposite

direction of training a network on more color channels and transferring it to a dataset with less color channels could be explored in future work.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, pp. 1097–1105, 2012.

[2] **T. Wollmann** and K. Rohr, "Deep Consensus Network: Aggregating predictions to improve object detection in microscopy images," *under review*, 2020.

[3] **T. Wollmann**, M. Gunkel, I. Chung, H. Erfle, and K. Rohr, "GRUU-Net: Integrated convolutional and gated recurrent neural network for cell segmentation," *Med. Image Anal.*, vol. 56, pp. 68–79, 2019.

[4] C. Ritter, **T. Wollmann**, P. Bernhard, M. Gunkel, D. M. Braun, J.-Y. Lee, J. Meiners, R. Simon, G. Sauter, H. Erfle, K. Rippe, R. Bartenschlager, and K. Rohr, "Hyperparameter optimization for image analysis: Application to prostate tissue images and live cell data of virus-infected cells," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 11, pp. 1847–1857, 2019.

[5] M. Veta, Y. J. Heng, N. Stathonikos, B. E. Bejnordi, F. Beca, **T. Wollmann**, K. Rohr, M. A. Shah, D. Wang, M. Rousson, *et al.*, "Predicting breast tumor proliferation from whole-slide images: The TUPAC16 challenge," *Med. Image Anal.*, vol. 54, pp. 111–121, 2019.

[6] M. C. Föll, L. Moritz, **T. Wollmann**, M. N. Stillger, N. Vockert, M. Werner, P. Bronsert, K. Rohr, B. A. Grüning, and O. Schilling, "Accessible and reproducible mass spectrometry imaging data analysis in Galaxy," *GigaScience*, vol. 8, no. 12, p. giz143, 2019.

[7] B. Grüning, R. Dale, A. Sjödin, B. A. Chapman, J. Rowe, C. H. Tomkins-Tinch, R. Valieris, A. Caprez, B. Batut, M. Haudgaard, T. Cokelaer, K. A. Beauchamp, B. S. Pedersen, Y. Hoogstrate, D. Ryan, A. Bretaudeau, G. L. Corguillé, C. Brueffer, D. Yusuf, S. Luna-Valero, R. Kirchner, K. Brinda, M. Raden, **T. Wollmann**, J. Köster, *et al.*, "Bioconda: A sustainable and comprehensive software distribution for the life sciences," *Nat. Methods*, vol. 15, pp. 475—-476, 2018.

[8] **T. Wollmann**, H. Erfle, R. Eils, K. Rohr, and M. Gunkel, "Workflows for microscopy image analysis and cellular phenotyping," *J. Biotechnol*, vol. 261, pp. 70–75, 2017.

[9] C. Ritter, **T. Wollmann**, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep learning particle detection for probabilistic tracking in fluorescence microscopy," in *Proc. ISBI*, IEEE, 2020.

[10] **T. Wollmann**, C. Ritter, J.-N. Dohrke, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "DetNet: Deep neural network for particle detection in microscopy images," in *Proc. ISBI*, pp. 517–520, IEEE, 2019.

[11] **T. Wollmann**, P. Bernhard, M. Gunkel, D. M. Braun, J. Meiners, R. Simon, G. Sauter, H. Erfle, K. Rippe, and K. Rohr, "Black-box hyperparameter optimization for nuclei segmentation in prostate tissue images," in *Proc. BVM*, pp. 345–350, Springer, 2019.

[12] **T. Wollmann**, C. S. Eijkman, and K. Rohr, "Adversarial domain adaptation to improve automatic breast cancer grading in lymph nodes," in *Proc. ISBI*, pp. 582–585, IEEE, 2018.

[13] **T. Wollmann**, J. Ivanova, and K. Rohr, "Multi-channel deep transfer learning for nuclei segmentation in glioblastoma cell tissue images," in *Proc. BVM*, pp. 316–321, Springer, 2018.

[14] R. Spilger, **T. Wollmann**, Y. Qiang, A. Imle, J.-Y. Lee, B. Müller, O. T. Fackler, R. Bartenschlager, and K. Rohr, "Deep Particle Tracker: Automatic tracking of particles in fluorescence microscopy images Using deep learning," in *Proc. MICCAI Workshop DLMIA*, pp. 128–136, Springer, 2018.

[15] D. Baltissen, **T. Wollmann**, M. Gunkel, I. Chung, H. Erfle, K. Rippe, and K. Rohr, "Comparison of segmentation methods for tissue microscopy images of glioblastoma cells," in *Proc. ISBI*, pp. 396–399, IEEE, 2018.

[16] **T. Wollmann** and K. Rohr, "Deep residual Hough voting for cell detection in histopathology images," in *Proc. ISBI*, pp. 341–344, IEEE, 2017.

[17] **T. Wollmann** and K. Rohr, "Automatic grading of breast cancer whole-slide histopathology images," in *Proc. BVM*, pp. 249–253, Springer, 2017.

[18] Y. Li and L. Chen, "Big biological data: Challenges and opportunities," *Genom. Proteom. Bioinf.*, vol. 12, no. 5, p. 187, 2014.

[19] M. Mulisch and U. Welsch, *Romeis-Mikroskopische Technik*. Springer-Verlag, 2015.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[21] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[22] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. NeurIPS*, pp. 91–99, 2015.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, pp. 779–788, IEEE, 2016.

[25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. ICCV*, pp. 2980–2988, IEEE, 2017.

[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, pp. 3431–3440, IEEE, 2015.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, pp. 234–241, Springer, 2015.

[28] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Med. Imaging*, vol. 40, no. 4, pp. 834–848, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, pp. 770–778, 2016.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proc. CVPR*, vol. 1, p. 3, IEEE, 2017.

[31] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, "Deep Learning in Microscopy Image Analysis: A Survey," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 4550–4568, 2017.

[32] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, pp. 740–755, Springer, 2014.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis*, vol. 115, no. 3, pp. 211–252, 2015.

[35] M. Aichler and A. Walch, "MALDI imaging mass spectrometry: Current frontiers and perspectives in pathology research and practice," *Lab. Invest.*, vol. 95, no. 4, p. 422, 2015.

[36] D. S. Cornett, M. L. Reyzer, P. Chaurand, and R. M. Caprioli, "MALDI imaging mass spectrometry: molecular snapshots of biochemical systems," *Nat. Methods*, vol. 4, no. 10, p. 828, 2007.

[37] A. I. Awad and M. Hassaballah, *Image feature detectors and descriptors*. Springer, 2016.

[38] H. Trevor, T. Robert, and J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2 ed., 2016.

[39] J. C. Russ and J. C. Russ, *Introduction to image processing and analysis*. CRC press, 2017.

[40] W. J. Godinez and K. Rohr, "Tracking multiple particles in fluorescence time-lapse microscopy images via probabilistic data association," *IEEE Trans. Med. Imag.*, vol. 34, no. 2, pp. 415–432, 2015.

[41] T. A. Nketia, H. Sailem, G. Rohde, R. Machiraju, and J. Rittscher, "Analysis of live cell images: Methods, tools and opportunities," *Methods*, vol. 115, pp. 65–79, 2017.

[42] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, p. 386, 1958.

[43] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[44] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control. Signal.*, vol. 2, no. 4, pp. 303–314, 1989.

[45] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *arXiv:1611.03530*, 2016.

[46] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?," *J. Stat. Phys.*, vol. 168, no. 6, pp. 1223–1247, 2017.

[47] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review," *Int. J. Autom. Comput.*, vol. 14, no. 5, pp. 503–519, 2017.

[48] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *arXiv:1609.04836*, 2016.

[49] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. NeurIPS*, pp. 6389–6399, 2018.

[50] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, "Sharp minima can generalize for deep nets," in *Proc. ICML*, pp. 1019–1028, ACM, 2017.

[51] M. Song, A. Montanari, and P. Nguyen, "A mean field view of the landscape of two-layers neural networks," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, pp. 7665–7671, 2018.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[53] L. Bottou, "Stochastic learning," in *Advanced lectures on machine learning*, pp. 146–168, Springer, 2004.

[54] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," 2012.

[55] D. P. Kingma and L. Ba, "ADAM: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[56] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. ICLR*, 2018.

[57] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. AISTATS*, pp. 315–323, 2011.

[58] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, p. 3, 2013.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, pp. 1026–1034, IEEE, 2015.

[60] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," Master's thesis, Technical University of Munich, 1991.

[61] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," in *arXiv:1711.05101*, 2017.

[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[63] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," in *arxiv:1502.03167*, 2015.

[64] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," in *arXiv:1809.00846*, 2018.

[65] V. L. D. U. A. Vedaldi, "Instance Normalization: The missing ingredient for fast stylization," in *arXiv:1607.08022*, 2016.

[66] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "Brain tumor segmentation and radiomics survival prediction: Contribution to the BRATS 2017 challenge," in *Proc. MICCAI Workshop Brainlesion*, pp. 287–297, Springer, 2017.

[67] M. Ghafoorian, N. Karssemeijer, T. Heskes, I. van Uder, F.-E. de Leeuw, E. Marchiori, B. van Ginneken, and B. Platel, "Non-uniform patch sampling with deep convolutional neural networks for white matter hyperintensity segmentation," in *Proc. ISBI*, pp. 1414–1417, IEEE, 2016.

[68] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. ICML*, pp. 41–48, ACM, 2009.

[69] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, "Test-time augmentation with uncertainty estimation for deep learning-based medical image segmentation," in *OpenReview:Byxv9aioz*, 2018.

[70] K. Zeeshan, "The impact of regularization on convolutional neural networks," Master's thesis, University of Jyväskylä, 2018.

[71] L. N. Smith and N. Topin, "Super-Convergence: Very fast training of residual networks using large learning rates," in *arXiv:1708.07120*, 2017.

[72] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 3DV*, pp. 565–571, IEEE, 2016.

[73] F. Isensee, J. Petersen, A. Klein, D. Zimmerer, P. F. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert, *et al.*, "nnU-Net: Self-adapting framework for U-Net-based medical image segmentation," in *arXiv:1809.10486*, 2018.

[74] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Med. Imaging*, vol. 35, no. 8, pp. 1798–1828, 2013.

[75] P. J. Huber *et al.*, "Robust estimation of a location parameter," *Ann. Math. Stat.*, vol. 35, no. 1, pp. 73–101, 1964.

[76] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NeurIPS*, pp. 2672–2680, 2014.

[77] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *arXiv:1611.01236*, 2016.

[78] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *arXiv:1312.6199*, 2013.

[79] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," in *arXiv:1610.01945*, 2016.

[80] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *arXiv:1804.09170*, 2018.

[81] D. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowl. Inf. Syst.*, vol. 36, no. 3, pp. 537–556, 2013.

[82] S. J. Pan, Q. Yang, *et al.*, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[83] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *arXiv:1409.7495*, 2014.

[84] H. Bilen and A. Vedaldi, "Universal representations: The missing link between faces, text, planktons, and cat breeds," in *arXiv:1701.07275*, 2017.

[85] Y. Tamaazousti, H. Le Borgne, and C. Hudelot, "Mucale-net: Multi categorical-level networks to generate more discriminating features," in *Proc. CVPR*, pp. 6711–6720, IEEE, 2017.

[86] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Proc. NeurIPS*, pp. 700–708, 2017.

[87] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," in *arXiv:1603.07285*, 2016.

[88] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *arXiv:1511.07122*, 2015.

[89] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," in *arXiv:1511.06432*, 2015.

[90] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," in *arXiv:1704.06857*, 2017.

[91] A. King, S. M. Bhandarkar, and B. M. Hopkinson, "A comparison of deep learning methods for semantic segmentation of coral reef survey images," in *Proc. CVPR*, pp. 1394–1402, 2018.

[92] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *arXiv:1409.1556*, 2014.

[93] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, pp. 1–9, IEEE, 2015.

[94] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, pp. 2818–2826, IEEE, 2016.

[95] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, p. 12, 2017.

[96] Q. Liao and T. Poggio, "Bridging the gaps between residual learning, recurrent neural networks and visual cortex," in *arXiv:1604.03640*, 2016.

[97] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 79, no. 8, pp. 2554–2558, 1982.

[98] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural. Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[99] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in *Proc. ICANN*, vol. 2, pp. 850–855, IET, 1999.

[100] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, pp. 580–587, IEEE, 2014.

[101] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis*, vol. 104, no. 2, pp. 154–171, 2013.

[102] R. Girshick, "Fast R-CNN," in *Proc. CVPR*, pp. 1440–1448, IEEE, 2015.

[103] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. CVPR*, pp. 6517–6525, IEEE, 2017.

[104] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *arXiv:1804.02767*, 2018.

[105] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. ECCV*, pp. 21–37, IEEE, 2016.

[106] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, vol. 1, p. 4, IEEE, 2017.

[107] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Med. Imaging*, 2018.

[108] S. Rao, "MITOS-RCNN: A novel approach to mitotic figure detection in breast cancer histopathology images using region based convolutional neural networks," in *arXiv:1807.01788*, 2018.

[109] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä, "Cell proposal network for microscopy image analysis," in *Proc. ICIP*, pp. 3199–3203, IEEE, 2016.

[110] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä, "Cell tracking via proposal generation and selection," in *arXiv:1705.03386*, 2017.

[111] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Proc. MICCAI*, pp. 411–418, Springer, 2013.

[112] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, "Beyond classification: Structured regression for robust cell detection using convolutional neural network," in *Proc. MICCAI*, pp. 358–365, Springer, 2015.

[113] K. Sirinukunwattana, S. E. A. Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[114] P. R. Gudla, K. Nakayama, G. Pegoraro, and T. Misteli, "SpotLearn: Convolutional neural network for detection of fluorescence in situ hybridization (FISH) signals in high-throughput imaging approaches," in *Proc. CSH Symposia on Quant. Biol.*, pp. 57–70, CSH Laboratory Press, 2017.

[115] M. A. Mabaso, D. J. Withey, and B. Twala, "Spot detection in microscopy images using convolutional neural network with sliding-window approach," *Bioimaging*, 2018.

[116] J. M. Newby, A. M. Schaefer, P. T. Lee, M. G. Forest, and S. K. Lai, "Convolutional neural networks automate detection for tracking of submicron-scale particles in 2D and 3D," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 115, no. 36, pp. 9026–9031, 2018.

[117] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Comput. Meth. Biomech. Biomed. Eng.*, vol. 6, no. 3, pp. 283–292, 2018.

[118] K. Nishida and K. Hotta, "Robust cell particle detection to dense regions and subjective training samples based on prediction of particle center using convolutional neural network," *PloS One*, vol. 13, no. 10, p. e0203646, 2018.

[119] E. Lu, W. Xie, and A. Zisserman, "Class-agnostic counting," in *Proc. ACCV*, pp. 669–684, 2018.

[120] V. Ranjan, H. Le, and M. Hoai, "Iterative crowd counting," in *Proc. ECCV*, pp. 270–285, Springer, 2018.

[121] Y. Tian, Y. Lei, J. Zhang, and J. Z. Wang, "PaDNet: Pan-Density Crowd Counting," *IEEE Trans. Image Process.*, vol. 29, pp. 2714–2727, 2020.

[122] H. Idrees, M. Tayyab, K. Athrey, D. Zhang, S. Al-Maadeed, N. Rajpoot, and M. Shah, "Composition loss for counting, density map estimation and localization in dense crowds," in *Proc. ECCV*, pp. 532–546, Springer, 2018.

[123] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proc. NeurIPS*, pp. 2843–2851, 2012.

[124] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, *et al.*, "Detecting cancer metastases on gigapixel pathology images," in *arXiv:1703.02442*, 2017.

[125] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. ICCV*, pp. 1520–1528, IEEE, 2015.

[126] V. Ulman, M. Maška, K. E. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, *et al.*, "An objective comparison of cell-tracking algorithms," *Nat. Methods*, vol. 14, no. 12, p. 1141, 2017.

[127] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. Balak, *et al.*, "A benchmark for comparison of cell tracking algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.

[128] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *Proc. ECCV*, pp. 75–91, Springer, 2016.

[129] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, *et al.*, "Attention U-Net: Learning where to look for the pancreas," in *arXiv:1804.03999*, 2018.

[130] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, p. e3, 2016.

[131] G. Lin, A. Milan, C. Shen, and I. D. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. CVPR*, vol. 1, p. 5, IEEE, 2017.

[132] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proc. CVPR*, pp. 1175–1183, IEEE, 2017.

[133] R. LaLonde and U. Bagci, "Capsules for object segmentation," in *arXiv:1804.04241*, 2018.

[134] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. NeurIPS*, pp. 3856–3866, 2017.

[135] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[136] F. Milletari, S.-A. Ahmadi, C. Kroll, A. Plate, V. Rozanski, J. Maiostre, J. Levin, O. Dietrich, B. Ertl-Wagner, K. Bötzel, *et al.*, "Hough-CNN: Deep learning for segmentation of deep brain regions in MRI and ultrasound," *Comput. Vis. Image Underst.*, vol. 164, pp. 92–102, 2017.

[137] F. Milletari, *Hough Voting Strategies for Segmentation, Detection and Tracking*. PhD thesis, Technical University of Munich, 2018.

[138] R. Li, K. Li, Y.-C. Kuo, M. Shu, X. Qi, X. Shen, and J. Jia, "Referring image segmentation via recurrent refinement networks," in *Proc. CVPR*, pp. 5745–5753, 2018.

[139] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE T. Pattern Anal.*, vol. 39, no. 12, pp. 2481–2495, 2017.

[140] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proc. CVPR*, pp. 4151–4160, IEEE, 2017.

[141] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," in *arXiv:1606.02147*, 2016.

[142] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. CVPR*, pp. 2881–2890, IEEE, 2017.

[143] R. Hu, M. Rohrbach, and T. Darrell, "Segmentation from natural language expressions," in *Proc. ECCV*, pp. 108–124, Springer, 2016.

[144] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proc. ICCV*, pp. 1529–1537, 2015.

[145] T. H. N. Le, K. G. Quach, K. Luu, C. N. Duong, and M. Savvides, "Reformulating level sets as deep recurrent neural network approach to semantic segmentation," *IEEE Trans. Image Process.*, vol. 27, no. 5, pp. 2393–2407, 2018.

[146] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, pp. 2980–2988, IEEE, 2017.

[147] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *arXiv:1412.7062*, 2014.

[148] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," in *arXiv:1706.05587*, 2017.

[149] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. ECCV*, pp. 801–818, Springer, 2018.

[150] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, pp. 1251–1258, IEEE, 2017.

[151] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *Proc. MICCAI Workshop LABELS*, pp. 179–187, Springer, 2016.

[152] S. K. Sadanandan, P. Ranefall, S. Le Guyader, and C. Wählby, "Automated training of deep convolutional neural networks for cell segmentation," *Scientific reports*, vol. 7, no. 1, p. 7860, 2017.

[153] J. Yi, P. Wu, D. J. Hoeppner, and D. Metaxas, "Pixel-wise neural cell instance segmentation," in *Proc. ISBI*, pp. 373–377, IEEE, 2018.

[154] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä, "Joint cell segmentation and tracking using cell proposals," in *Proc. ISBI*, pp. 920–924, IEEE, 2016.

[155] A. Arbelle and T. R. Raviv, "Microscopy cell segmentation via adversarial neural networks," in *Proc. ISBI*, pp. 645–648, IEEE, 2018.

[156] J. C. Caicedo, J. Roth, A. Goodman, T. Becker, K. W. Karhohs, M. Broisin, C. Molnar, C. McQuin, S. Singh, F. J. Theis, *et al.*, "Evaluation of deep learning strategies for nucleus segmentation in fluorescence images," *Cytometry Part A*, vol. 95, no. 9, pp. 952–965, 2019.

[157] M. Veta, P. J. Van Diest, S. M. Willems, H. Wang, A. Madabhushi, A. Cruz-Roa, F. Gonzalez, A. B. Larsen, J. S. Vestergaard, A. B. Dahl, *et al.*, "Assessment of algorithms for mitosis detection in breast cancer histopathology images," *Med. Image Anal.*, vol. 20, no. 1, pp. 237–248, 2015.

[158] P. Ruusuvuori, T. Äijö, S. Chowdhury, C. Garmendia-Torres, J. Selinummi, M. Birbaumer, A. M. Dudley, L. Pelkmans, and O. Yli-Harja, "Evaluation of methods for detection of fluorescence labeled subcellular objects in microscope images," *BMC Bioinf.*, vol. 11, no. 248, pp. 1–17, 2010.

[159] I. Smal, M. Loog, W. Niessen, and E. Meijering, "Quantitative comparison of spot detection methods in fluorescence microscopy," *IEEE Trans. Med. Imag.*, vol. 29, no. 2, pp. 282–301, 2010.

[160] K. Štěpka, P. Matula, P. Matula, S. Wörz, K. Rohr, and M. Kozubek, "Performance and sensitivity evaluation of 3D spot detection methods in confocal microscopy," *Cytometry Part A*, vol. 87, no. 8, pp. 759–772, 2015.

[161] D. Sage, F. R. Neumann, F. Hediger, S. M. Gasser, and M. Unser, "Automatic tracking of individual fluorescence particles: Application to the study of chromosome dynamics," *IEEE Trans. Image Process.*, vol. 14, no. 9, pp. 1372–1383, 2005.

[162] A. Basset, J. Boulanger, J. Salamero, P. Bouthemy, and C. Kervrann, "Adaptive spot detection with optimal scale selection in fluorescence microscopy images," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 4512–4527, 2015.

[163] N. Chenouard, I. Smal, F. De Chaumont, M. Maška, I. F. Sbalzarini, Y. Gong, J. Cardinale, C. Carthel, S. Coraluppi, M. Winter, A. R. Cohen, W. J. Godinez, K. Rohr, Y. Kalaidzidis, L. Liang, J. Duncan, H. Shen, Y. Xu, K. E. Magnusson, J. Jaldén, H. M. Blau, P. Paul-Gilloteaux, P. Roudot, C. Kervrann, F. Waharte, J.-Y. Tinevez, S. L. Shorte, J. Willemse, K. Celler, G. P. van Wezel, H.-W. Dan, Y.-S. Tsai, C. Ortiz de Solarzano, J.-C. Olivo-Marin, and E. Meijering, "Objective comparison of particle tracking methods," *Nat. Methods*, vol. 11, no. 3, pp. 281–289, 2014.

[164] D. Sage, F. R. Neumann, F. Hediger, S. M. Gasser, and M. Unser, "Automatic tracking of individual fluorescence particles: application to the study of chromosome dynamics," *IEEE Trans. Image Process.*, vol. 14, no. 9, pp. 1372–1383, 2005.

[165] N. Chenouard, I. Bloch, and J. C. Olivo-Marin, "Multiple hypothesis tracking for cluttered biological image sequences," *IEEE Trans. Med. Imaging*, vol. 35, no. 11, pp. 2736–3750, 2013.

[166] N. Chenouard *et al.*, "Objective comparison of particle tracking methods," *Nat. Methods*, vol. 11, no. 3, pp. 281–289, 2014.

[167] I. Sbalzarini and P. Koumoutsakos, "Feature point tracking and trajectory analysis for video imaging in cell biology," *J. Struct. Biol.*, vol. 151, no. 2, pp. 182–195, 2005.

[168] L. Liang, H. Shen, P. D. Camilli, and J. S. Duncan, "A novel multiple hypothesis based particle tracking method for clathrin mediated endocytosis analysis using fluorescence microscopy," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1844–1857, 2014.

[169] P. Roudot, L. Ding, K. Jaqaman, C. Kervrann, and G. Danuser, "Piecewise-stationary motion modeling and iterative smoothing to track heterogeneous particle motions in dense environments," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5395–5410, 2017.

[170] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proc. AAAI*, pp. 4225–4232, 2017.

[171] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. ICCV*, pp. 300–311, 2017.

[172] T. He, H. Mao, J. Guo, and Z. Yi, "Cell tracking using deep neural networks with multi-task learning," *Image Vision Comput.*, vol. 60, pp. 142–153, 2017.

[173] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting with fully convolutional regression networks," in *Proc. MICCAI Workshop DLMIA*, pp. 283–292, 2015.

[174] H. Chen, X. Wang, and P. A. Heng, "Automated mitosis detection with deep regression networks," in *Proc. ISBI*, pp. 1204–1207, IEEE, 2016.

[175] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. ECCV*, pp. 630–645, Springer, 2016.

[176] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *arXiv:1605.07146*, 2016.

[177] D. P. Nikolaev, S. M. Karpenko, I. P. Nikolaev, and P. P. Nikolayev, "Hough transform: underestimated tool in the computer vision field," in *Proc. ECMS*, vol. 238, p. 246, 2008.

[178] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, pp. 249–256, 2010.

[179] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. NeurIPS*, pp. 2366–2374, 2014.

[180] A. McGuire, J. A. Brown, C. Malone, R. McLaughlin, and M. J. Kerin, "Effects of age on the detection and management of breast cancer," *Cancers*, vol. 7, no. 2, pp. 908–929, 2015.

[181] "MITOS and ATYPIA 14." `https://grand-challenge.org/site/mitos-atypia-14/`, 2014. Accessed: 15.09.2016.

[182] E. Aptoula, N. Courty, and S. Lefèvre, "Mitosis detection in breast cancer histological images with mathematical morphology," in *Proc. SIU*, pp. 1–4, IEEE, 2013.

[183] "Tumor Proliferation Assessment Challenge 2016." `http://tupac.tue-image.nl/`, 2016. Accessed: 10.04.2020.

[184] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[185] P. F. Jaeger, S. A. Kohl, S. Bickelhaupt, F. Isensee, T. A. Kuder, H.-P. Schlemmer, and K. H. Maier-Hein, "Retina U-Net: Embarrassingly simple exploitation of segmentation supervision for medical object detection," in *arXiv:1811.08661*, 2018.

[186] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, pp. 7132–7141, IEEE, 2018.

[187] Y. Wu and K. He, "Group normalization," in *Proc. ECCV*, pp. 3–19, Springer, 2018.

[188] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[189] J. M. Robson, "Algorithms for maximum independent sets," *J. Algorithms*, vol. 7, no. 3, pp. 425–440, 1986.

[190] S. Sakai, M. Togasaki, and K. Yamazaki, "A note on greedy algorithms for the maximum weighted independent set problem," *Discrete Appl. Math.*, vol. 126, no. 2-3, pp. 313–322, 2003.

[191] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation," in *Proc. ECIR*, pp. 345–359, Springer, 2005.

[192] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Res.*, vol. 2, pp. 37–63, 2011.

[193] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochim. Biophys. Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[194] P. Harremoës and G. Tusnády, "Information divergence is more chi squared distributed than the chi squared statistics," in *arXiv:1202.1125*, 2012.

[195] J. C. Principe, D. Xu, J. Fisher, and S. Haykin, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[196] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[197] R. Sokal and F. Rohlf, *Biometry: The Principles and Practice of Statistics in Biological Research*. W. H. Freeman, 1995.

[198] J. Hoey, "The two-way likelihood ratio (G) test and comparison to two-way chi squared test," in *arXiv:1206.4881*, 2012.

[199] C. Studholme, D. L. Hill, and D. J. Hawkes, "An overlap invariant entropy measure of 3d medical image alignment," *Pattern recognition*, vol. 32, no. 1, pp. 71–86, 1999.

[200] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Deterministic edge-preserving regularization in computed imaging," *IEEE Trans. Image Process.*, vol. 6, no. 2, pp. 298–311, 1997.

[201] B. Heidenreich, P. S. Rachakonda, K. Hemminki, and R. Kumar, "TERT promoter mutations in cancer development," *Curr. Opin. Genet. Dev.*, vol. 24, pp. 30–37, 2014.

[202] M. Peifer, F. Hertwig, F. Roels, D. Dreidax, M. Gartlgruber, R. Menon, A. Krämer, J. L. Roncaioli, F. Sand, J. M. Heuckmann, *et al.*, "Telomerase activation by genomic rearrangements in high-risk neuroblastoma," *Nature*, vol. 526, no. 7575, p. 700, 2015.

[203] L. J. Valentijn, J. Koster, D. A. Zwijnenburg, N. E. Hasselt, P. van Sluis, R. Volckmann, M. M. van Noesel, R. E. George, G. A. Tytgat, J. J. Molenaar, *et al.*, "TERT rearrangements are frequent in neuroblastoma and identify aggressive tumors," *Nat. Genet.*, vol. 47, no. 12, p. 1411, 2015.

[204] T. M. Bryan, A. Englezou, J. Gupta, S. Bacchetti, and R. R. Reddel, "Telomere elongation in immortal human cells without detectable telomerase activity," *EMBO J.*, vol. 14, no. 17, pp. 4240–4248, 1995.

[205] D.-L. Qi, T. Ohhira, C. Fujisaki, T. Inoue, T. Ohta, M. Osaki, E. Ohshiro, T. Seko, S. Aoki, M. Oshimura, *et al.*, "Identification of pitx1 as a TERT suppressor gene located on human chromosome 5," *Mol. Cell. Biol.*, vol. 31, no. 8, pp. 1624–1636, 2011.

[206] M. Gunkel, I. Chung, S. Wörz, K. I. Deeg, R. Simon, G. Sauter, D. T. Jones, A. Korshunov, K. Rohr, H. Erfle, *et al.*, "Quantification of telomere features in tumor tissue sections by an automated 3D imaging-based workflow," *Methods*, vol. 114, pp. 60–73, 2017.

[207] S. Wörz, P. Sander, M. Pfannmöller, R. J. Rieker, S. Joos, G. Mechtersheimer, P. Boukamp, P. Lichter, and K. Rohr, "3D geometry-based quantification of colocalizations in multichannel 3D microscopy images of human soft tissue tumors," *IEEE Trans. Med. Imag.*, vol. 29, no. 8, pp. 1474–1484, 2010.

[208] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *arXiv:1409.1259*, 2014.

[209] Google Brain Team, "tf.data: Build TensorFlow input pipelines — TensorFlow Core." `https://www.tensorflow.org/guide/datasets`, 2019. Accessed: 10.04.2020.

[210] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Writing Custom Datasets, DataLoaders and Transforms." `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html`, 2019. Accessed: 10.04.2020.

[211] Microsoft Research, "CNTK 201: Part B - Image Understanding." `https://cntk.ai/pythondocs/CNTK_201B_CIFAR-10_ImageHandsOn.html`, 2019. Accessed: 10.04.2020.

[212] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, pp. 265–283, 2016.

[213] B. De Brabandere, D. Neven, and L. Van Gool, "Semantic instance segmentation with a discriminative loss function," in *arXiv:1708.02551*, 2017.

[214] C. Payer, D. Štern, T. Neff, H. Bischof, and M. Urschler, "Instance segmentation and tracking with cosine embeddings and recurrent hourglass networks," in *Proc. MICCAI*, pp. 3–11, Springer, 2018.

[215] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proc. CVPR*, pp. 5221–5229, IEEE, 2017.

[216] P. Bandi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. E. Bejnordi, B. Lee, K. Paeng, A. Zhong, *et al.*, "From detection of individual metastases to classification of lymph node status at the patient level: The CAMELYON17 challenge," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 550–560, 2019.

[217] M. Tektonidis and K. Rohr, "Diffeomorphic multi-frame non-rigid registration of cell nuclei in 2D and 3D live cell images," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1405–1417, 2017.

[218] M. A. Cypko, M. Stoehr, M. Kozniewski, M. J. Druzdzel, A. Dietz, L. Berliner, and H. U. Lemke, "Validation workflow for a clinical Bayesian network model in multidisciplinary decision making in head and neck oncology treatment," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 12, no. 11, pp. 1959–1970, 2017.

[219] K. Cleary and T. M. Peters, "Image-Guided Interventions: Technology Review and Clinical Applications," *Annu. Rev. Biomed. Eng.*, vol. 12, pp. 119–142, 2010.

[220] Y. Wang, S. Du, S. Balakrishnan, and A. Singh, "Stochastic zeroth-order optimization in high dimensions," in *arXiv:1710.10551*, 2017.

[221] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A service for black-box optimization," in *Proc. SIGKDD*, pp. 1487–1495, ACM, 2017.

[222] L. Hertel, J. Collado, P. Sadowski, and P. Baldi, "Sherpa: Hyperparameter optimization for machine learning models," in *Proc. NeurIPS Workshop MLOSS*, 2018.

[223] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. SIGKDD*, pp. 847–855, ACM, 2013.

[224] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. NeurIPS*, pp. 2951–2959, 2012.

[225] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn," in *Proc. ICML Workshop AutoML*, pp. 2825–2830, 2014.

[226] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proc. LION*, pp. 507–523, Springer, 2011.

[227] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. SIGKDD*, pp. 785–794, ACM, 2016.

[228] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat*, vol. 33, no. 3, pp. 1065–1076, 1962.

[229] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[230] E. Afgan, D. Baker, B. Batut, M. Van Den Beek, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, B. A. Grüning, *et al.*, "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update," *Nucleic Acids Res.*, vol. 46, no. 1, pp. 537–544, 2018.

[231] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nat. Biotechnol.*, vol. 35, no. 4, p. 316, 2017.

[232] M. Huh, P. Agrawal, and A. A. Efros, "What makes ImageNet good for transfer learning?," in *arXiv:1608.08614*, 2016.

[233] S. B. Edge and C. C. Compton, "The American Joint Committee on Cancer: the 7th edition of the AJCC cancer staging manual and the future of TNM," *Ann. Surg. Oncol.*, vol. 17, no. 6, pp. 1471–1474, 2010.

[234] "CAMELYON16." `https://camelyon16.grand-challenge.org`, 2016. Accessed: 10.04.2020.

[235] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep learning for identifying metastatic breast cancer," in *arXiv:1606.05718*, 2016.

[236] R. Chen, Y. Jing, and H. Jackson, "Identifying metastases in sentinel lymph nodes with deep convolutional neural networks," in *arXiv:1608.01658*, 2016.

[237] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. CVPR*, pp. 3642–3649, IEEE, 2012.

[238] F. Ciompi, O. Geessink, B. E. Bejnordi, G. S. de Souza, A. Baidoshvili, G. Litjens, B. van Ginneken, I. Nagtegaal, and J. van der Laak, "The importance of stain normalization in colorectal tissue classification with convolutional networks," in *Proc. ISBI*, pp. 160–163, IEEE, 2017.

[239] M. W. Lafarge, J. P. Pluim, K. A. Eppenhof, P. Moeskops, and M. Veta, "Domain-adversarial neural networks to address the appearance variability of histopathology images," in *Proc. MICCAI Workshop DLMIA*, pp. 83–91, Springer, 2017.

[240] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. ICCV*, pp. 2223–2232, IEEE, 2017.

[241] "CAMELYON17." `https://camelyon17.grand-challenge.org`, 2017. Accessed: 10.04.2020.

[242] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist.*, vol. 2, pp. 7–21, 2005.

[243] L. Vincent, "Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms," *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 176–201, 1993.

[244] A. B. L. Larsen, J. S. Vestergaard, and R. Larsen, "HEp-2 cell classification using shape index histograms with donut-shaped spatial pooling," *IEEE Trans. Med. Imag.*, vol. 33, no. 7, pp. 1573–1580, 2014.

[245] V. A. Kovalev, F. Kruggel, H.-J. Gertz, and D. Y. von Cramon, "Three-dimensional texture analysis of mri brain datasets," *IEEE Trans. Med. Imag.*, vol. 20, no. 5, pp. 424–433, 2001.

[246] V. Jain, B. Bollmann, M. Richardson, D. R. Berger, M. N. Helmstaedter, K. L. Briggman, W. Denk, J. B. Bowden, J. M. Mendenhall, W. C. Abraham, *et al.*, "Boundary learning by optimization with topological constraints," in *Proc. CVPR*, pp. 2488–2495, IEEE, 2010.

[247] A. A. Dima, J. T. Elliott, J. J. Filliben, M. Halter, A. Peskin, J. Bernal, M. Kociolek, M. C. Brady, H. C. Tang, and A. L. Plant, "Comparison of segmentation algorithms for fluorescence microscopy images of cells," *Cytometry Part A*, vol. 79, no. 7, pp. 545–559, 2011.

[248] L. P. Coelho, A. Shariff, and R. F. Murphy, "Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms," in *Proc. ISBI*, pp. 518–521, IEEE, 2009.

[249] J. Cheng and J. C. Rajapakse, "Segmentation of clustered nuclei with shape markers and marking function," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 3, pp. 741–748, 2009.

[250] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 93, no. 4, pp. 1591–1595, 1996.

[251] J. Cardinale, G. Paul, and I. F. Sbalzarini, "Discrete region competition for unknown numbers of connected regions," *IEEE Trans. Image Process.*, vol. 21, no. 8, pp. 3531–3545, 2012.

[252] I. Arganda-Carreras, V. Kaynig, C. Rueden, K. W. Eliceiri, J. Schindelin, A. Cardona, and H. S. Seung, "Trainable Weka Segmentation: A machine learning tool for microscopy pixel classification," *Bioinformatics*, pp. 2424–2426, 2017.

[253] C. Sommer, C. Straehle, U. Köthe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *Proc. ISBI*, pp. 230–233, IEEE, 2011.

[254] J. Bernsen, "Dynamic thresholding of grey-level images," in *Proc. ICPR*, pp. 1251–1255, 1986.

[255] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. SIAM*, pp. 1027–1035, ACM, 2007.

[256] N. Harder, F. Mora-Bermúdez, W. J. Godinez, A. Wünsche, R. Eils, J. Ellenberg, and K. Rohr, "Automatic analysis of dividing cells in live cell movies to detect mitotic delays and correlate phenotypes in time," *Genome Research*, vol. 19, no. 11, pp. 2113–2124, 2009.

[257] T. Esteves, P. Quelhas, A. M. Mendonça, and A. Campilho, "Gradient convergence filters and a phase congruency approach for in vivo cell nuclei detection," *Mach. Vision Appl.*, vol. 23, no. 4, pp. 623–638, 2012.

[258] K. E. Magnusson and J. Jaldén, "A batch algorithm using iterative application of the viterbi algorithm to track cells and construct cell lineages," in *Proc. ISBI*, pp. 382–385, IEEE, 2012.

[259] L. Kostrykin, C. Schnörr, and K. Rohr, "Segmentation of cell nuclei using intensity-based model fitting and sequential convex programming," in *Proc. ISBI*, pp. 654–657, IEEE, 2018.

[260] R. Bensch and O. Ronneberger, "Cell segmentation and tracking in phase contrast images using graph cut with asymmetric boundary costs," in *Proc. ISBI*, pp. 1220–1223, IEEE, 2015.

[261] H. Tian, "Analysis of deep learning based instance segmentation methods for nuclei segmentation," 2019. Heidelberg University internship report.

[262] C. Ritter, A. Imle, J. Y. Lee, B. Müller, O. T. Fackler, R. Bartenschlager, and K. Rohr, "Two-filter probabilistic data association for tracking of virus particles in fluorescence microscopy images," in *Proc. ISBI*, pp. 957–960, IEEE, 2018.

[263] H. B. Mitchell, *Data fusion: Concepts and ideas*. Springer, 2012.

[264] J. A. Sethian, *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge University Press, 1999.

[265] M. M. Usaj, E. B. Styles, A. J. Verster, H. Friesen, C. Boone, and B. J. Andrews, "High-content screening for quantitative cell biology," *Trends Cell Biol.*, vol. 26, no. 8, pp. 598–611, 2016.

[266] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, *et al.*, "Fiji: An open-source platform for biological-image analysis," *Nat. Methods*, vol. 9, no. 7, pp. 676–682, 2012.

[267] J. R. Swedlow and K. W. Eliceiri, "Open source bioimage informatics for cell biology," *Trends Cell Biol.*, vol. 19, no. 11, pp. 656–660, 2009.

[268] H. Peng, "Bioimage informatics: A new area of engineering biology," *Bioinformatics*, vol. 24, no. 17, pp. 1827–1836, 2008.

[269] S. N. Goodman, D. Fanelli, and J. P. Ioannidis, "What does research reproducibility mean?," *Sci. Transl. Med.*, vol. 8, no. 341, p. 12, 2016.

[270] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Sci. Data*, vol. 3, 2016.

[271] K. G. Achilleos, C. C. Kannas, C. A. Nicolaou, C. S. Pattichis, and V. J. Promponas, "Open source workflow systems in life sciences informatics," in *Proc. BIBE*, pp. 552–558, IEEE, 2012.

[272] J. Ison, K. Rapacki, H. Ménager, M. Kalaš, E. Rydza, P. Chmura, C. Anthon, N. Beard, K. Berka, D. Bolser, *et al.*, "Tools and data services registry: A community effort to document bioinformatics resources," *Nucleic Acids Res.*, vol. 44, no. D1, p. D38, 2016.

[273] K. W. Eliceiri, M. R. Berthold, I. G. Goldberg, L. Ibáñez, B. S. Manjunath, M. E. Martone, R. F. Murphy, H. Peng, A. L. Plant, B. Roysam, *et al.*, "Biological imaging software tools," *Nat. Methods*, vol. 9, no. 7, pp. 697–710, 2012.

[274] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. 453, 2014.

[275] L. P. Coelho, "Mahotas: Open source software for scriptable computer vision," in *arXiv:1211.4907*, 2012.

[276] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "NIH Image to ImageJ: 25 years of image analysis," *Nat. Methods*, vol. 9, no. 7, p. 671, 2012.

[277] A. E. Carpenter, T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, *et al.*, "CellProfiler: Image analysis software for identifying and quantifying cell phenotypes," *Genome Biol.*, vol. 7, no. 10, p. 100, 2006.

[278] F. de Chaumont, S. Dallongeville, N. Chenouard, H. Herve, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Montagner, *et al.*, "Icy: An open community platform for bioimage informatics," *Nat. Methods*, vol. 26, pp. 690–6, 2013.

[279] G. Bradski *et al.*, "The OpenCV library," *Dr. Dobb's J.*, vol. 25, no. 11, pp. 120–126, 2000.

[280] W. J. Schroeder and K. M. Martin, *The Visualization Toolkit-3.0*. Elsevier, 1996.

[281] H. J. Johnson, M. M. McCormick, and L. Ibanez, "The ITK software guide," 2015.

[282] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J.-C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, *et al.*, "3D slicer as an image computing platform for the quantitative imaging network," *J. Magn. Reson. Imaging*, vol. 30, no. 9, pp. 1323–1341, 2012.

[283] I. Wolf, M. Vetter, I. Wegner, M. Nolden, T. Böttger, M. Hastenteufel, M. Schöbinger, T. Kunert, and H.-P. Meinzer, "The medical imaging interaction toolkit (MITK)–a toolkit facilitating the creation of interactive software by extending VTK and ITK," in *Proc. SPIE*, vol. 5367, p. 17, 2004.

[284] J. Vivian, A. A. Rao, F. A. Nothaft, C. Ketchum, J. Armstrong, A. Novak, J. Pfeil, J. Narkizian, A. D. Deran, A. Musselman-Brown, *et al.*, "Toil enables reproducible, open source, big biomedical data analyses," *Nat. Biotechnol.*, vol. 35, no. 4, p. 314, 2017.

[285] J. Köster and S. Rahmann, "Snakemake - a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, pp. 2520–2522, 2012.

[286] S. P. Sadedin, B. Pope, and A. Oshlack, "Bpipe: a tool for running and managing bioinformatics pipelines," *Bioinformatics*, vol. 28, no. 11, pp. 1525–1526, 2012.

[287] M. Berthold, N. Cebron, F. Dill, T. Gabriel, T. Kotter, T. Meinl, P. Ohl, K. Thiel, and B. K. Wiswedel, "The Konstanz information miner: Version 2.0 and beyond," *SIGKDD Explorations*, vol. 11, pp. 26–31, 2009.

[288] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, *et al.*, "The Taverna workflow suite: Designing and executing workflows of Web Services on the desktop, web or in the cloud," *Nucleic Acids Res.*, vol. 41, no. 1, pp. 557–561, 2013.

[289] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, *et al.*, "Galaxy: A platform for interactive large-scale genome analysis," *Genome research*, vol. 15, no. 10, pp. 1451–1455, 2005.

[290] E. Afgan, D. Baker, M. Van den Beek, D. Blankenberg, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, C. Eberhard, *et al.*, "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update," *Nucleic Acids Res.*, vol. 44, no. 1, p. 3, 2016.

[291] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, *et al.*, "myExperiment: A repository and social network for the sharing of bioinformatics workflows," *Nucleic Acids Res.*, vol. 38, no. suppl 2, pp. 677–682, 2010.

[292] C. Dietz and M. R. Berthold, "KNIME for open-source bioimage analysis: A tutorial," in *Focus on Bio-Image Informatics*, pp. 179–197, Springer, 2016.

[293] J. Ison, M. Kalaš, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer, and P. Rice, "EDAM: An ontology of bioinformatics operations, types of data and identifiers, topics and formats," *Bioinformatics*, vol. 29, no. 10, pp. 1325–1332, 2013.

[294] B. Grüning, E. Rasche, B. Rebolledo-Jaramillo, C. Eberhart, T. Houwaart, J. Chilton, N. Coroar, R. Backofen, J. Taylor, and A. Nekrutenko, "Enhancing pre-defined workflows with ad hoc analytics using Galaxy, Docker and Jupyter," in *bioRxiv:075457*, 2016.

[295] B. Batut, S. Hiltemann, A. Bagnacani, D. Baker, V. Bhardwaj, C. Blank, A. Bretaudeau, L. Brillet-Guéguen, M. Čech, J. Chilton, *et al.*, "Community-driven data analysis training for biology," *Cell Syst.*, vol. 6, no. 6, pp. 752–758, 2018.

[296] D. Kaye, *Loosely coupled: The missing pieces of Web services*. RDS Strategies LLC, 2003.

[297] F. da Veiga Leprevost, B. A. Grüning, S. Alves Aflitos, H. L. Röst, J. Uszkoreit, H. Barsnes, M. Vaudel, P. Moreno, L. Gatto, J. Weber, *et al.*, "BioContainers: An open-source and community-driven framework for software standardization," *Bioinformatics*, vol. 33, no. 16, pp. 2580–2582, 2017.

[298] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, *et al.*, "Bioconductor: Open software development for computational biology and bioinformatics," *Genome Biol.*, vol. 5, no. 10, p. R80, 2004.

[299] J. Ahrens, B. Geveci, and C. Law, "Paraview: An end-user tool for large data visualization," in *The visualization handbook*, Elsevier, 2005.

[300] A. Goode, B. Gilbert, J. Harkes, D. Jukic, and M. Satyanarayanan, "OpenSlide: A vendor-neutral software foundation for digital pathology," *J. Pathol. Inform.*, vol. 4, pp. 4–27, 2013.

[301] IOS, "ISO 9241: Ergonomics of human-system interaction," 1996.

[302] IOS, "ISO/TR 16982: Ergonomics of human-system interaction - usability methods supporting human-centred design," 2002.

[303] J. Brooke *et al.*, "SUS – A quick and dirty usability scale," in *Usability evaluation in industry*, vol. 189, pp. 4–7, CRC Press, 1996.

[304] G. Albaum, "The Likert scale revisited," *J. Mark. Res.*, vol. 39, no. 2, pp. 1–21, 1997.

[305] J. Sauro and J. R. Lewis, *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[306] K. Finstad, "The system usability scale and non-native english speakers," *J. Usability Stud.*, vol. 1, no. 4, pp. 185–188, 2006.

[307] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Int. J. Hum. Comput. Interact.*, vol. 24, no. 6, pp. 574–594, 2008.

[308] J. R. Lewis and J. Sauro, "The factor structure of the system usability scale," in *Proc. HCD*, pp. 94–103, Springer, 2009.

[309] J. Nielsen, "Usability Metrics." `https://www.nngroup.com/articles/usability-metrics/`, 2006. Accessed: 10.04.2020.

[310] J. Rubin and D. Chisnell, *Handbook of usability testing: How to plan, design and conduct effective tests*. John Wiley & Sons, 2008.

[311] S. Wirtz, E.-M. Jakobs, and M. Ziefle, "Age-specific usability issues of software interfaces," in *Proc. IEA*, vol. 17, 2009.

[312] A. Sonderegger, S. Schmutz, and J. Sauer, "The influence of age in usability testing," *Appl. Ergon.*, vol. 52, pp. 291–300, 2016.

[313] P. Kumberger, *Quantifying the impact of cell-to-cell transmission on viral spread*. PhD thesis, Heidelberg University, 2018.

[314] D. J. Selkoe and J. Hardy, "The amyloid hypothesis of Alzheimer's disease at 25 years," *EMBO Mol. Med.*, vol. 8, no. 6, pp. 595–608, 2016.

[315] R. Fol, J. Braudeau, S. Ludewig, T. Abel, S. W. Weyer, J.-P. Roederer, F. Brod, M. Audrain, A.-P. Bemelmans, C. J. Buchholz, *et al.*, "Viral gene transfer of APPsα rescues synaptic failure in an alzheimer's disease mouse model," *Acta Neuropathol.*, vol. 131, no. 2, pp. 247–266, 2016.

[316] N. Richter, N. Beckers, O. A. Onur, M. Dietlein, M. Tittgemeyer, L. Kracht, B. Neumaier, G. R. Fink, and J. Kukolja, "Effect of cholinergic treatment depends on cholinergic integrity in early alzheimer's disease," *Brain Res.*, vol. 141, no. 3, pp. 903–915, 2018.

[317] D. R. Bhandari, Q. Wang, W. Friedt, B. Spengler, S. Gottwald, and A. Römpp, "High resolution mass spectrometry imaging of plant tissues: Towards a plant metabolite atlas," *Analyst*, vol. 140, no. 22, pp. 7696–7709, 2015.

[318] W. D. Hoffmann and G. P. Jackson, "Forensic mass spectrometry," *Annu. Rev. Anal. Chem.*, vol. 8, pp. 419–440, 2015.

[319] P.-M. Vaysse, R. M. Heeren, T. Porta, and B. Balluff, "Mass spectrometry imaging for clinical research–latest developments, applications, and current limitations," *Analyst*, vol. 142, no. 15, pp. 2690–2712, 2017.

[320] O. Karlsson and J. Hanrieder, "Imaging mass spectrometry in drug development and toxicology," *Arch. Toxicol.*, vol. 91, no. 6, pp. 2283–2294, 2017.

[321] A. Römpp, J.-P. Both, A. Brunelle, R. M. Heeren, O. Laprévote, B. Prideaux, A. Seyer, B. Spengler, M. Stoeckli, and D. F. Smith, "Mass spectrometry imaging of biological tissue: an approach for multicenter studies," *Anal. Bioanal. Chem.*, vol. 407, no. 8, pp. 2329–2335, 2015.

[322] A. Buck, B. Heijs, B. Beine, J. Schepers, A. Cassese, R. M. Heeren, L. A. McDonnell, C. Henkel, A. Walch, and B. Balluff, "Round robin study of formalin-fixed paraffin-embedded tissues in mass spectrometry imaging," *Anal. Bioanal. Chem.*, vol. 410, no. 23, pp. 5969–5980, 2018.

[323] A. M. Porcari, J. Zhang, K. Y. Garza, R. M. Rodrigues-Peres, J. Q. Lin, J. H. Young, R. Tibshirani, C. Nagi, G. R. Paiva, S. A. Carter, *et al.*, "Multicenter study using desorption-electrospray-ionization-mass-spectrometry imaging for breast-cancer diagnosis," *Anal. Chem.*, vol. 90, no. 19, pp. 11324–11332, 2018.

[324] A. Ly, R. Longuespée, R. Casadonte, P. Wandernoth, K. Schwamborn, C. Bollwein, C. Marsching, K. Kriegsmann, C. Hopf, W. Weichert, *et al.*, "Site-to-site reproducibility and spatial resolution in MALDI–MSI of peptides from formalin-fixed paraffin-embedded samples," *Proteom. Clin. App.*, vol. 13, no. 1, p. 1800029, 2019.

[325] M. A. Fischler and R. C. Bolles, "RANdom SAmple Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[326] B. P. Makay, "A method for registration of 3d shape," *IEEE T. Pattern Anal.*, vol. 14, pp. 239–256, 1992.

[327] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[328] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE Trans. Image Process.*, vol. 5, no. 8, pp. 1266–1271, 1996.

[329] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proc. ICML*, pp. 1842–1850, 2016.

[330] N. Dong and E. Xing, "Few-shot semantic segmentation with prototype learning," in *Proc. BMVC*, vol. 1, p. 6, 2018.

[331] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proc. ICCV*, pp. 8059–8068, IEEE, 2019.