DISSERTATION

submitted to the

Combined Faculty of Mathematics,

Engineering and Natural Sciences

of

Heidelberg University, Germany

for the degree of

Doctor of Engineering

Put forward by

Roman Spilger

born in Heidelberg, Germany

Deep Learning Methods for Detection and Tracking of Particles in Fluorescence Microscopy Images

Advisor: PD Dr. Karl Rohr

Oral examination: .....

Dedicated to the memory of my beloved father Ernst-Friedrich.

Abstract

Studying the dynamics of sub-cellular structures such as receptors, filaments, and vesicles is a prerequisite for investigating cellular processes at the molecular level. In addition, it is important to characterize the dynamic behavior of virus structures to gain a better understanding of infection mechanisms and to develop novel drugs. To investigate the dynamics of fluorescently labeled sub-cellular and viral structures, time-lapse fluorescence microscopy is the most often used imaging technique. Due to the limited spatial resolution of microscopes caused by diffraction, these very small structures appear as bright, blurred spots, denoted as particles, in microscopy images. To draw statistically meaningful biological conclusions, a large number of such particles need to be analyzed. However, since manual analysis of fluorescent particles is very time consuming, fully automated computer-based methods are indispensable.

We introduce novel deep learning methods for detection and tracking of multiple particles in fluorescence microscopy images. We propose a particle detection method based on a convolutional neural network which performs image-to-image mapping by density map regression and uses the adaptive wing loss. For particle tracking, we present a recurrent neural network that exploits past and future information in both forward and backward direction. Assignment probabilities across multiple detections as well as the probabilities for missing detections are computed jointly. To resolve tracking ambiguities using future information, several track hypotheses are propagated to later time points. In addition, we developed a novel probabilistic deep learning method for particle tracking, which is based on a recurrent neural network mimicking classical Bayesian filtering. The method includes both aleatoric and epistemic uncertainty, and provides valuable information about the reliability of the computed trajectories. Short and long-term temporal dependencies of individual object dynamics are exploited for state prediction, and assigned detections are used to update the predicted states. Moreover, we developed a convolutional Long Short-Term Memory neural network for combined particle tracking and colocalization analysis in two-channel microscopy image sequences. The network determines colocalization probabilities, and colocalization information is exploited to improve tracking. Short and long-term temporal dependencies of object motion as well as image intensities are taken into account to compute assignment probabilities jointly across multiple detections. We also introduce a deep learning method for probabilistic particle detection and tracking. For particle detection, temporal information is integrated to regress a density map and determine sub-pixel particle positions. For tracking, a fully Bayesian neural network is presented that mimics classical Bayesian filtering and takes into account both aleatoric and epistemic uncertainty. Uncertainty information of individual particle detections is considered. Network training for the developed deep learning-based particle tracking methods relies only on synthetic

data, avoiding the need of time-consuming manual annotation. We performed an extensive evaluation of our methods based on image data of the Particle Tracking Challenge as well as on fluorescence microscopy images displaying virus proteins of HCV and HIV, chromatin structures, and cell-surface receptors. It turned out that the methods outperform previous methods.

Zusammenfassung

Die Untersuchung der Dynamik subzellulärer Strukturen wie beispielsweise Rezeptoren, Filamente und Vesikel ist eine Voraussetzung für die Erforschung zellulärer Prozesse auf molekularer Ebene. Darüber hinaus ist es wichtig, das dynamische Verhalten von Viruspartikeln zu charakterisieren, um ein besseres Verständnis der Infektionsmechanismen zu erlangen und neue Medikamente zu entwickeln. Um die Dynamik von fluoreszenzmarkierten subzellulären Strukturen und Viruspartikeln zu untersuchen, ist die Zeitraffer-Fluoreszenzmikroskopie die am häufigsten verwendete Bildgebungstechnik. Aufgrund der begrenzten räumlichen Auflösung von Mikroskopen, die durch Beugung verursacht wird, erscheinen diese sehr kleinen Strukturen in Mikroskopiebildern als helle, unscharfe Flecken, die als Partikel bezeichnet werden. Um statistisch aussagekräftige biologische Schlussfolgerungen zu ziehen, muss eine sehr große Anzahl solcher Partikel analysiert werden. Da die manuelle Analyse von fluoreszierenden Partikeln jedoch sehr zeitaufwändig ist, sind vollautomatische computergestützte Methoden unerlässlich.

Wir haben neuartige Deep Learning Methoden zur Detektion und Verfolgung (Tracking) mehrerer Partikel in Fluoreszenzmikroskopie-Bildern entwickelt. Wir stellen eine Partikeldetektionsmethode vor, die auf einem Convolutional Neural Network basiert, das eine Bild-zu-Bild-Abbildung durch Dichtekartenregression vornimmt und den Adaptive Wing Loss verwendet. Außerdem stellen wir ein Recurrent Neural Network für die Partikelverfolgung vor, das vergangene und zukünftige Informationen sowohl in Vorwärts- als auch in Rückwärtsrichtung ausnutzt. Die Zuordnungswahrscheinlichkeiten für mehrere Detektionen sowie die Wahrscheinlichkeiten für fehlende Detektionen werden gleichzeitig berechnet. Um Mehrdeutigkeiten bei der Verfolgung mit Hilfe von Zukunftsinformationen aufzulösen, werden mehrere Verfolgungshypothesen zu späteren Zeitpunkten propagiert. Darüber hinaus haben wir eine neuartige probabilistische Deep Learning Methode für die Partikelverfolgung entwickelt, die auf einem Recurrent Neural Network basiert, das die klassische Bayes'sche Filterung nachahmt. Die Methode berücksichtigt sowohl aleatorische als auch epistemische Unsicherheiten und liefert wertvolle Informationen über die Zuverlässigkeit der berechneten Trajektorien. Kurz- und langfristige zeitliche Abhängigkeiten der Dynamik einzelner Objekte werden für die Zustandsvorhersage ausgenutzt, und zugewiesene Detektionen werden zur Aktualisierung der vorhergesagten Zustände verwendet. Darüber hinaus haben wir ein Convolutional Neural Network mit Lang- und Kurzzeitgedächtnis für die kombinierte Partikelverfolgung und Kolokalisationsanalyse in zweikanaligen Mikroskopie-Bildsequenzen entwickelt. Das Netzwerk ermittelt Kolokalisationswahrscheinlichkeiten, und die Kolokalisationsinformationen werden zur Verbesserung der Verfolgung genutzt. Kurz- und langfristige zeitliche Abhängigkeiten der Objektbewegung sowie der Bildintensitäten werden berücksichtigt, um Zuordnungswahrscheinlichkeiten gleichzeitig für mehrere De-

tektionen zu berechnen. Des Weiteren stellen wir eine Deep Learning Methode für die probabilistische Partikelerkennung und -verfolgung vor. Für die Partikeldetektion werden zeitliche Informationen integriert, um eine Dichtekarte zu regressieren und Sub-Pixel-Partikelpositionen zu bestimmen. Für die Verfolgung wird ein vollständig Bayes'sches neuronales Netzwerk vorgestellt, das die klassische Bayes'sche Filterung nachahmt und sowohl aleatorische als auch epistemische Unsicherheiten berücksichtigt. Die Unsicherheitsinformationen der einzelnen Partikeldetektionen werden ebenfalls einbezogen. Das Netzwerktraining für die entwickelten Deep Learning basierten Partikelverfolgungsmethoden beruht nur auf synthetischen Daten, so dass eine zeitaufwändige manuelle Annotation nicht erforderlich ist. Wir haben unsere Methoden anhand von Bilddaten der Particle Tracking Challenge sowie anhand Fluoreszenzmikroskopie-Bilder, die Virusproteine von HCV und HIV, Chromatinstrukturen und Zelloberflächenrezeptoren zeigen, umfassend evaluiert. Es stellte sich heraus, dass die Methoden bessere Ergebnisse als frühere Methoden erzielen.

Acknowledgements

I am very grateful to numerous people without whose support this doctoral thesis at the Faculty of Engineering at Heidelberg University would not have been possible. My thanks also go to the university for providing a pleasant and productive scientific working environment.

First of all, I would like to express my deep gratitude to my supervisor *PD Dr. Karl Rohr* for providing me with very valuable guidance, advice, and support. I am very grateful for his patience and for giving me the freedom to work on the topics that interested me the most. In addition, I would like to thank *Prof. Dr. Ralf Bartenschlager*, *Prof. Dr. Ulrike Müller*, *Dr. Ilka Bischofs*, and *Prof. Dr. M. Cristina Cardoso* for the successful cooperation.

I am very grateful to my colleagues from the Biomedical Computer Vision (BMCV) group for providing support and sharing ideas. It was a great pleasure to work with you in a team. My special thanks go to *Christian Ritter*, *Carola Krug*, and *Leonid Kostrykin* for the great collaboration and excellent team spirit. We had numerous very interesting and fruitful discussions, which I always enjoyed very much. I would also like to thank *Kerem Celikay* for the cooperation and valuable tips on registration. In addition, I am very grateful to *Sabrina Wetzel* for her dedication and indispensable support in all kinds of administrative tasks. I would also like to thank the many people at BioQuant who make a pleasant and productive working environment possible. My special thanks go to *Nina Beil* and *Jürgen Beneke* for the scientific discussions and humorous conversations about life during the lunch break.

I also gratefully acknowledge support from the DFG (German Research Foundation) within the SFB 1129 (projects Z4, P11) and the SPP 2202 as well as the BMBF (Federal Ministry of Education and Research) within de.NBI and CancerTelSys.

I would like to thank my brother-in-law *Jan* and my closest friends *Martin* and *Frederik* for their great support as well as wonderful company. You have always believed in me and are a very important part of my life. I also want to thank my niece and godchild *Luisa* for cheering me up with her adorable smile since January 2019. I am already full of anticipation for the birth of your brother. I am very grateful to my girlfriend *Madelaine* for her patience, her love, and for being part of my life. I am looking forward to starting new chapters together with you. My deepest thanks go to my parents, *Gudrun* and *Ernst-Friedrich*, and my sister *Dana-Maria*. Your unconditional love and unlimited support throughout my life let me overcome all self-doubts and setbacks. Dad, I will never forget you!

Heidelberg, January 2023 *Roman Spilger*

Contents

List of figures

List of tables

Notation conventions

Mathematical symbols frequently used in this thesis.

$\mathbb{R}$	Space of real numbers
$\mathbb{R}_+$	Space of non-negative real numbers
f	Activation function
$\mathbf{W}$	Weight tensor
$\mathbf{b}$	Bias vector
N	Number of neurons / samples / objects
i	Index for neuron / sample / object
M	Number of neurons / detections
j	Index for neuron / detection
$\mathbf{x}$	Input vector / state vector
$\mathbf{y}$	Output vector / detection vector
$\mathcal{L}$	Loss function
η	Learning rate
$\odot$	Hadamard product
t	Time point
$\mathbf{h}$	Hidden state
$\tilde{\mathbf{h}}$	Candidate hidden state
$\mathbf{c}$	Cell state
$\mathbf{f}$	Forget gate
$\mathbf{i}$	Input gate
$\mathbf{o}$	Output gate
$\mathbf{r}$	Reset gate
$\mathbf{r}$	Update gate
$*$	Convolution operation
$\mathbf{a}$	Assignment probabilities

Publications

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

Peer-reviewed journals

A. M. Poos, C. Schroeder, N. Jaishankar, D. Röll, M. Oswald, J. Meiners, D. M. Braun, C. Knotz, L. Frank, M. Gunkel, **R. Spilger**, T. Wollmann, A. Polonski, G. Makrypidi-Fraune, C. Fraune, M. Graefen, I. Chung, A. Stenzel, H. Erfle, K. Rohr, A. Baniahmad, G. Sauter, K. Rippe, R. Simon, and R. König, "PITX1 is a regulator of TERT expression in prostate cancer with prognostic power," *Cancers*, vol. 14, no. 5, 2022

C. S. Bold, D. Baltissen, S. Ludewig, M. K. Back, J. Just, L. Kilian, S. Erdinger, M. Banicevic, L. Rehra, F. Almouhanna, M. Nigri, D. P. Wolfer, **R. Spilger**, K. Rohr, O. Kann, C. J. Buchholz, J. von Engelhardt, M. Korte, and U. C. Müller, "APPsα rescues Tau-induced synaptic pathology," *J. Neurosci.*, vol. 42, no. 29, pp. 5782–5802, 2022

R. Spilger, J.-Y. Lee, V. O. Chagin, L. Schermelleh, M. C. Cardoso, R. Bartenschlager, and K. Rohr, "Deep probabilistic tracking of particles in fluorescence microscopy images," *Med. Image Anal.*, vol. 72, 102128, 2021

R. Spilger, A. Imle, J.-Y. Lee, B. Müller, O. T. Fackler, R. Bartenschlager, and K. Rohr, "A recurrent neural network for particle tracking in microscopy images using future information, track hypotheses, and multiple detections," *IEEE Trans. Image Process.*, vol. 29, pp. 3681–3694, 2020

Peer-reviewed conference proceedings

R. Spilger, J.-Y. Lee, M. T. Pham, R. Bartenschlager, and K. Rohr, "Deep learning method for probabilistic particle detection and tracking in fluorescence microscopy images," *submitted for publication*, 2022

R. Spilger, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep neural network for combined particle tracking and colocalization analysis in two-channel microscopy images," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, Kolkata, India, 2022

R. Spilger, V. O. Chagin, C. S. Bold, L. Schermelleh, U. C. Müller, M. C. Cardoso, and K. Rohr, "Deep neural network for 3D particle detection in 3D fluorescence microscopy images via density map regression," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, Kolkata, India, 2022

C. Ritter, **R. Spilger**, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep learning for particle detection and tracking in fluorescence microscopy images," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, Nice, France, 2021, pp. 873–876

R. Spilger, J. Hellgoth, J.-Y. Lee, S. Hänselmann, D.-P. Herten, R. Bartenschlager, and K. Rohr, "Tracking of particles in fluorescence microscopy images using a spatial distance model for Brownian motion," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, Iowa City, Iowa, USA, 2020, pp. 969–972

R. Spilger, T. Schwackenhofer, C. Kaspar, I. Bischofs, and K. Rohr, "Deep segmentation of bacteria at different stages of the life cycle," in *Proc. Workshop Bildverarbeitung für die Medizin (BVM)*, Berlin, Germany, Informatik aktuell, Springer-Verlag Berlin Heidelberg, 2020, pp. 8–13

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. International Workshop on Deep Learning in Medical Image Analysis (DLMIA)*, vol. 11045, Granada, Spain, 2018, pp. 128–136

Chapter 1

Introduction

1.1 Motivation

The cell is considered the smallest structural and functional unit of all living organisms. Cellular systems are affected by internal as well as external signals and are characterized by a complex interplay of numerous cellular and molecular processes. Thus, cells can adapt to different conditions and perform a variety of tasks. Quantifying the dynamic behavior of sub-cellular structures (e.g., receptors [1], vesicles [2], cytoskeletal filaments [3], chromatin structures [4]) with high spatial and temporal resolution provides important insights into cellular processes at the molecular level, which is a prerequisite for understanding higher-level biological processes such as pathogenesis, neuronal communication, apoptosis, and embryogenesis. In addition, it is important to investigate the dynamic behavior of virus particles to achieve a complete understanding of infection mechanisms (e.g., [5]).

Time-lapse fluorescence microscopy is a powerful imaging technique for studying the dynamics of sub-cellular structures and virus particles specifically labeled with fluorescent dyes. These very small fluorescent structures appear in microscopic images as bright, blurry spots, called particles, due to the diffraction-limited resolution of optical systems. A detailed quantitative characterization of the underlying dynamic processes can be obtained by reconstructing the observed particle trajectories, from which biophysical parameters such as the diffusion coefficient, the velocity, and the acceleration can be determined. In addition, analyzing protein dynamics in multi-channel time-lapse fluorescence microscopy images allows estimating binding affinities between different sub-cellular structures. To draw statistically significant conclusions in biological studies, a large number of such particles must be examined. However, since manual detection and tracking of fluorescent particles is tedious, fully automated computer-based methods are indispensable.

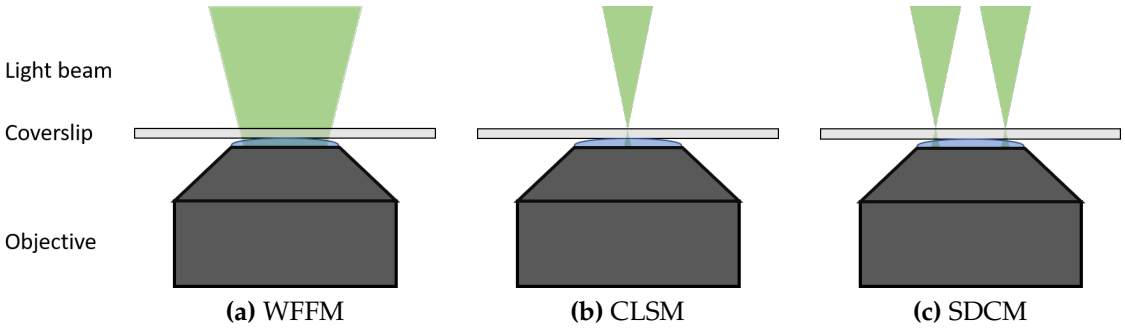
In recent years, deep learning methods have been introduced yielding state-of-the-art performance in numerous computer vision and medical image analysis tasks [6, 7, 8], including image classification [9, 10, 11], object detection [12, 13, 14],

object segmentation [15, 16, 17], and object tracking [18, 19, 20]. Deep learning has been used to analyze biological objects in fluorescence microscopy images (e.g., [21, 22, 23, 24]), however, most of the existing deep learning-based detection and tracking methods have been developed for objects (pedestrians, cars) in images of natural scenes. Compared to images of natural scenes, detection and tracking of fluorescent particles has additional task-specific challenges (e.g., small objects, lack of appearance characteristics, high object density, complex motion behavior, low SNR). In addition, large training data sets consisting of real images, such as VisDrone-DET [25] or MOTChallenge [26] for natural images, are not available and difficult to generate for fluorescent particle detection and tracking tasks.

This thesis presents novel deep learning methods for particle detection and tracking in fluorescence microscopy images, while considering various task-specific challenges. Experiments based on image data of various virus structures (e.g., virus proteins, virus particles) and sub-cellular structures (e.g., cell surface receptors, chromatin structures) acquired using different fluorescence microscopy techniques show that the developed methods yield better results compared to previous methods. In the following, we describe the acquisition process of fluorescence microscopy images and elaborate on the tasks and challenges involved in detecting and tracking fluorescent particles.

1.2 Fluorescence Microscopy

The basic idea of *fluorescence microscopy* is to irradiate a biological specimen with light of a specific wavelength (or wavelengths) using a strong light source (e.g., Xenon lamp, laser) and then separate the emitted light of a higher wavelength (Stokes' shift) from the excitation light [27]. In the detection path of the microscope, only the emitted light is transmitted by optical filters and then measured by a camera sensor (e.g., charge-coupled device, complementary metal-oxide-semiconductor [28]) so that the fluorescent structures are displayed with high contrast on a dark background. This allows to determine information about the presence, position, distribution, and number of these fluorescent structures in the specimen. However, most sub-cellular structures and virus particles have no intrinsic fluorescence property, so labeling of the objects with fluorescent probes (e.g., fluorescent proteins, dyes, quantum dots) is required for their observation, whereas not labeled objects are expected to emit no (or little) light and are therefore (almost) invisible. There exist numerous labeling strategies to specifically bind the fluorescent probes to the structures of interest (e.g., protein tags, artificial amino acids) [29, 30]. In *multi-channel fluorescence microscopy*, different emission wavelengths are measured (almost) simultaneously, allowing imaging of different fluorescent probes binding to different types of structures in the specimen. In *time-lapse fluorescence microscopy*, images of the fluorescent labeled structures are acquired in a temporally sequential manner, which enables the observation of the structures over time.



(a) WFFM **(b)** CLSM **(c)** SDCM

Figure 1.1 Illustration of different illumination patterns in fluorescence microscopy.

To obtain high magnification and high resolution images, various fluorescence microscopy techniques have been developed, such as widefield, confocal laser scanning, and confocal spinning disk microscopy. The main differences of the illumination patterns are shown in Fig. 1.1. The underlying technical principles are similar, but with different adaptations, which also indicates their characteristics and limitations. In *widefield fluorescence microscopes* (WFFM), image acquisition is fast and the whole field-of-view is simultaneously illuminated by the light source, exciting all fluorescent probes and collecting emitted light from the entire depth [31]. A major drawback of this technique is the limited contrast and resolution (especially for thick specimens) due to blur caused by out-of-focus light emitted from the specimen. In contrast, *confocal microscopes* utilize the pinhole principle to prevent out-of-focus light from reaching the detector [31, 32]. *Confocal laser scanning microscopes* (CLSM) use a laser as light source, that scans specimens sequentially point-by-point. A photomultiplier tube is typically used for detecting the emitted light of individual points. However, since each position of the specimen is imaged individually, image acquisition rates are relatively low and mainly determined by the scanning speed [31]. In *spinning disk confocal microscopy* (SDCM), the problem of scanning speed is solved by the multiplex principle, which allows imaging of fast dynamic processes and living cells. Numerous excitation light beams are projected onto the specimen in parallel, and the emitted light is then detected at the different excitation coordinates. A Nipkow disk with multiple pinholes is used, which rotates mechanically at high speed. A major limitation of primary SDCM designs with only one Nipkow disk is poor illumination efficiency (about 4%) [33]. In double-disk SDCM, a Nipkow disk with thousands of pinholes that produce confocality is combined with a parallel rotating second Nipkow disk that has a matching array of microlenses. Thus, each pinhole is associated with a microlens at the same disk coordinate. When illuminated, each microlens produces a focused spot of light, of which about 40% can pass into the associated pinhole [28]. Typically charge-coupled device (CCD) cameras are used, which capture the light from all pinholes simultaneously [31]. CCD cameras have significantly

higher quantum efficiencies than photomultiplier tubes, and therefore, the available emission photons are collected more efficiently [34].

Due to diffraction phenomena in optical systems, a fluorescent point source (e.g., fluorescent molecule) does not appear as an infinitely sharp point when imaged with a conventional microscope, but as an blurred focal spot with a finite size [35]. The imaging process can be described by a convolution operation between the intensity density map of the specimen and the *point spread function* (PSF), which characterizes the response of the optical system of the microscope to a fluorescent point source and defines its intensity profile in the blurred image [35, 36, 37]. Consequently, two identical fluorescent point sources separated by a distance less than the full width at half maximum (FWHM) of the PSF appear as an unresolvable large blurred spot ('particle') [35, 38]. Thus, conventional fluorescence microscopes have limited spatial resolution of about 200 nm in the focal plane and 500 nm along the optical axis and cannot spatially resolve smaller structures [39, 40]. The PSF of optical systems can be detemined analytically based on wave optics or empirically based on point objects. Both approaches have limitations [41]. In various fluorescence microscopy techniques (e.g., WFFM, CLSM, SDCM), the Gaussian function is a sufficient approximation of the PSF [42]. Here, sub-resolution structures appear in the blurred images as Gaussian-like fluorescent particles whose appearance is parameterized by the peak intensity and standard deviation of the underlying Gaussian function.

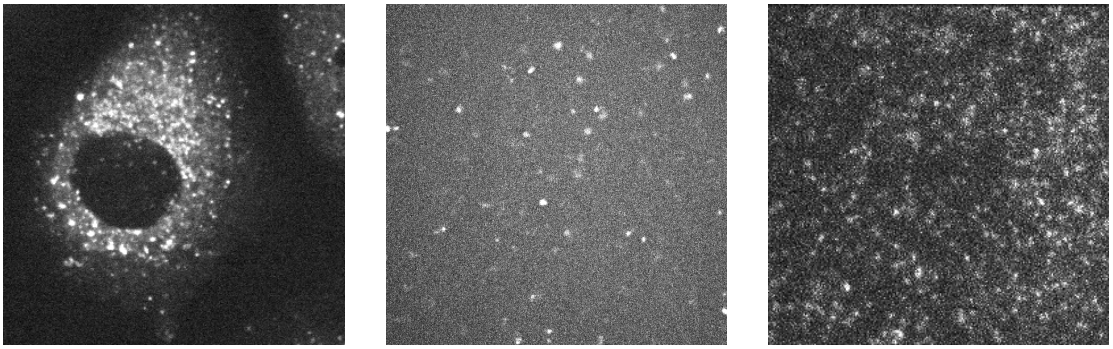
In recent years, several *super-resolution microscopy* techniques have been developed that overcome the diffraction-limited spatial resolution in conventional fluorescence microscopes. These techniques can be divided into two main categories, but all exploit a chemical or physical property of a fluorescent probe to keep adjacent molecules in different states (i.e., on and off), allowing them to be imaged spatially resolved from each other [43]. Techniques of the first category utilize patterned illumination, whereby not all fluorescent probes are excited simultaneously in a diffraction-limited spatial region. This category includes *stimulated emission depletion microscope* (STED) [39] and *structured illumination microscopy* (SIM) [44]. Techniques of the second category are based on photoswitching or other processes to stochastically activate and deactivate individual fluorescent probes in a diffraction-limited spatial region [31]. This category includes *photoactivation localization microscopy* (PALM) [45] and *stochastic optical reconstruction microscopy* (STORM) [46]. For example, STED has achieved a lateral resolution of about 20 nm in a biological specimen [35]. However, super-resolution microscopes are significantly more expensive, more complex to operate, and not yet as widely used as conventional fluorescence microscopes.

Besides *limited spatial resolution*, there are other challenges in fluorescence microscopy of biological specimens. Since sub-cellular structures are usually very small, only a few fluorescent probes can be attached to them. This results in a relatively *weak fluorescence signal*. In addition, cells exhibit an intrinsic natural

fluorescence, called *auto-fluorescence*, due to some cellular components (e.g., mitochondria) and metabolites [47, 48, 49]. The weak fluorescence signal of labeled sub-cellular structures combined with cellular auto-fluorescence results in *low image contrast*. In addition, exposure to high-intensity excitation light can degrade fluorescent probes, causing loss of fluorescence properties [50]. This process, known as *photobleaching*, occurs particularly during time-lapse fluorescence microscopy, where a temporal image sequence is acquired. For each image acquisition, the fluorescent structures are exposed to excitation light, resulting in a decrease of the fluorescence signal of the labeled structures over time. Photobleaching consequently affects the image acquisition rate (number of images per time interval) to be selected for an experimental setup. A high image acquisition rate allows events to be imaged with high *temporal resolution*, but the sub-cellular structures can only be observed over a relatively short period of time. A low image acquisition rate allows the sub-cellular structures to be observed over a long period of time, but the temporal resolution may be too low for relevant short events. A possible compromise between these two image acquisition strategies is to reduce the intensity of the excitation light, reducing the degradation of fluorescent probes. However, this results in weak fluorescence signals. *Phototoxicity* due to frequent exposure to high-intensity excitation light leads to cellular damage that can affect the entire cell physiology and often occurs during live-cell fluorescence microscopy. Fluorescent probes and cellular organic molecules (e.g., flavins, andporphyrins) exposed to excitation light can react with oxygen and become degraded, producing free radicals that severely damage cellular structures and components (e.g., deoxyribonucleic acid, unsaturated fatty acids, enzyme cofactors) [51, 52]. To minimize phototoxicity, low-intensity excitation light and image acquisition rate should be preferred [53]. Since few fluorescent probes attach to very small sub-cellular structures and low-intensity excitation light is used to minimize photobleaching as well as phototoxicity, the fluorescence signal in time-lapse fluorescence microscopy of living cells is usually weak. The weak fluorescence signal in combination with cellular auto-fluorescence and other *noise sources* in the microscopy system (e.g., leakage current, extraneous light, photon noise) generally results in images with low contrast and *signal-to-noise ratio* (SNR) [54].

1.3 Tasks and Challenges of Particle Detection and Tracking

To quantitatively characterize the dynamic behavior of fluorescent labeled sub-cellular or virus structures from temporal fluorescence microscopy image sequences, *particle trajectories* must be reconstructed, from which biophysical parameters such as the diffusion coefficient, the velocity, and the acceleration can be obtained. The reconstruction of particle trajectories over a temporal image sequence can be decomposed into two main tasks: (i) detect the particles in each image and (ii) link individual particle positions over time using the obtained particle detections, i.e.



(a) HCV proteins **(b)** HIV-1 particles **(c)** Cell-surface receptors

Figure 1.2 Example image sections from three different real temporal fluorescence microscopy image sequences. The images vary greatly and the SNR values are rather low.

track individual particles. Both tasks are generally demanding in computer vision. In fluorescence microscopy, the typically *low image quality* due to image acquisition limitations must be additionally addressed. In the following, we will describe the tasks and challenges for particle detection and tracking in time-lapse fluorescence microscopy images.

The aim of *detection* is to determine image regions that belong to particles, i.e. distinguish fluorescent particles from the background. The main limiting factor of the detection performance is the SNR level of the image data [55, 56]. In most cases, the SNR level is low due to few fluorescent probes attaching to the very small structures, auto-fluorescence of the cells, noise sources in the microscopy system, and low intensity excitation light used to minimize photobleaching as well as phototoxicity. Moreover, despite recent technological advances, the resolution of microscopes is too low compared to the size of sub-cellular and virus structures. Consequently, it is often difficult even for biology experts to distinguish particles from spurious fluorescent background structures or image noise [55]. The limiting factors in particle detection typically result in an erroneous set of noisy detections: Some artifacts are falsely detected as particles (missing detections) and some particles are not detected (missing detections). To visualize the challenges of particle detection, Fig. 1.2 shows example fluorescence microscopy image sections of various sub-cellular and virus structures.

The aim of *tracking* is to link individual particle positions over time using the obtained particle detections. *Correspondence finding* addresses the task of determining whether or not detections in different images belong to the same object [57]. Since particles usually cannot be distinguished by their *appearance*, correspondence finding is generally based only on assumptions about the underlying motion model [58]. However, particles perform various motion types (e.g. Brownian motion, directed motion), sometimes confined to a very small volume, and undergo abrupt changes

in direction as well as speed. Due to the *complex motion behavior*, correspondence finding becomes very challenging and ambiguous, especially for high particle densities, which often exist in biological samples. False detections and missing detections make correspondence finding even more difficult. In addition, since moving particles can enter and leave the focal plane or imaged region, particle appearance and disappearance has to be addressed. To incorporate uncertainty in position estimation, *spatio-temporal filtering* is usually employed. This is often formulated within a *Bayesian framework*, where a posterior distribution on the position of a particle is estimated recursively over time, based on a sequence of assigned noisy detections and assumptions about the motion type [58].

1.4 Contributions

In this thesis, novel deep learning methods for particle detection and tracking in fluorescence microscopy image sequences are presented that address several task-specific challenges. More specifically, the main contributions are:

- **Recurrent Neural Networks for Particle Tracking:** A novel particle tracking method is presented based on a recurrent neural network that exploits past information about individual object dynamics for state prediction and correspondence finding. The network computes assignment probabilities jointly across multiple detections and determines probabilities for missing detections. Training is performed using only simulated data, avoiding the need for tedious manual annotation of training data. As an extension, a new deep learning method is proposed based on a recurrent neural network that exploits past and future information in both forward and backward direction. To handle track initiation and termination, the network calculates existence probabilities. Information at later time points can be used to resolve ambiguities by propagating track hypotheses to future time points. Handcrafted similarity measures and motion features are not required. In contrast to classical tracking methods, manual tuning of tracking parameters and prior assumptions about probability distributions are not required. Compared to previous deep learning methods for fluorescent particle tracking, future information and multiple tracking hypotheses are exploited for correspondence finding and track initiation as well as termination. The work has been published in [59, 60].

- **Deep Probabilistic Tracking of Particles in Fluorescence Microscopy Images:** A novel probabilistic deep learning method for fluorescent particle tracking is introduced, which is based on a recurrent neural network that mimics classical Bayesian filtering. Compared to previous deep learning methods for particle tracking, both aleatoric and epistemic uncertainty is taken into account. Thus, information about the reliability of the computed

trajectories is determined. Manual tuning of tracking parameters is not necessary and prior knowledge about the noise statistics is not required. Short and long-term temporal dependencies of individual object dynamics are exploited for state prediction, and assigned detections are used to update the predicted states. For correspondence finding, a neural network is used which computes assignment probabilities jointly across multiple detections as well as determines the probabilities of missing detections. Training requires only simulated data and therefore tedious manual annotation of ground truth is not needed. The work has been published in [61].

- **Deep Neural Network for Combined Particle Tracking and Colocalization Analysis:** A new deep learning method for combined particle tracking and colocalization analysis in two-channel microscopy image sequences is presented. Short and long-term temporal dependencies of object motion as well as image intensities are taken into account to compute assignment probabilities jointly across multiple detections and determine colocalization probabilities. Compared to previous deep learning methods for particle tracking in fluorescence microscopy images, two channels are exploited, colocalization analysis is performed, and image intensities are used for correspondence finding. The work has been published in [62].

- **Deep Learning for Particle Detection and Tracking:** A novel deep learning method for 3D particle detection in 3D fluorescence microscopy images is introduced. Instead of pixel-wise binary classification or direct coordinate regression, image-to-image mapping is performed based on regressing a density map. Detections close to particles are rewarded during network training, and highly nonlinear direct prediction of point coordinates is avoided. To focus on particles in comparison to background image points, we suggest using the adaptive wing loss. A weighted loss map is employed to cope with the very strong imbalance between particle and background image points for 3D images. Compared to previous deep learning methods for fluorescent particle detection, density map regression is performed, the adaptive wing loss is used, and full 3D information is exploited. In addition, the particle detection method is extended for integrating temporal information and determining sub-pixel positions, and combined with a fully Bayesian neural network for tracking. In contrast to previous work, temporal information is taken into account for fluorescent particle detection and a fully Bayesian neural network is used for tracking that considers uncertainty information of individual detections. The work has been published in [63] and submitted for publication [64].

- **Quantitative Performance Evaluation:** Quantitative performance studies of the proposed particle detection and tracking methods have been performed

based on data from the Particle Tracking Challenge (PTC, [65]) as well as real fluorescence microscopy images displaying various sub-cellular and viral structures. It turned out that the proposed methods yield improved results compared to previous methods.

1.5 Outline of the Thesis

This thesis is organized as follows. In Chapter 2, fundamentals of deep learning are described, and previous work on object detection and tracking is discussed. Chapter 3 presents the metrics and state-of-the-art methods used for quantitative performance evaluation and comparison. Chapter 4 introduces new deep learning methods for particle tracking in fluorescence microscopy images that exploit temporal information by recurrent neural networks. In Chapter 5, we introduce a probabilistic deep learning method for particle tracking in fluorescence microscopy images. A recurrent neural network mimics classical Bayesian filtering, and a feed-forward neural network is used for correspondence finding. Chapter 6 presents a convolutional Long Short-Term Memory based neural network for combined particle tracking and colocalization analysis in two-channel microscopy image sequences. In Chapter 7, we propose a method for particle detection based on a convolutional neural network that performs density map regression. In addition, a deep learning method for probabilistic detection and tracking of particles in fluorescence microscopy images is introduced. Chapter 8 summarizes the thesis, points out limitations, and provides suggestions for future research.

Chapter 2

Foundations and Previous Work

In this chapter, we describe the fundamental concepts that are essential for this thesis. We present foundations of deep learning and discuss recent developments in the field of object detection and tracking. In addition, an overview of state-of-the-art methods for detection and tracking of particles in fluorescence microscopy images is provided.

2.1 Foundations of Deep Learning

A key feature of intelligent life is the ability to acquire knowledge from experience. Deep learning is a subfield of machine learning in which artificial neural networks (ANNs) learn from sample data. Since Deep Learning is a flexible data-driven approach with universal approximation capabilities, it is used for various tasks.

2.1.1 Artificial Neural Networks

Artificial neural networks (ANNs) mimic the structure and function of biological neural networks in animals. Similar to neurons in the brain, ANNs consist of *artificial neurons* which are connected to each other and arranged in various layers. An ANN layer with N neurons has the weight parameters $\mathbf{W} \in \mathbb{R}^{M \times N}$ and

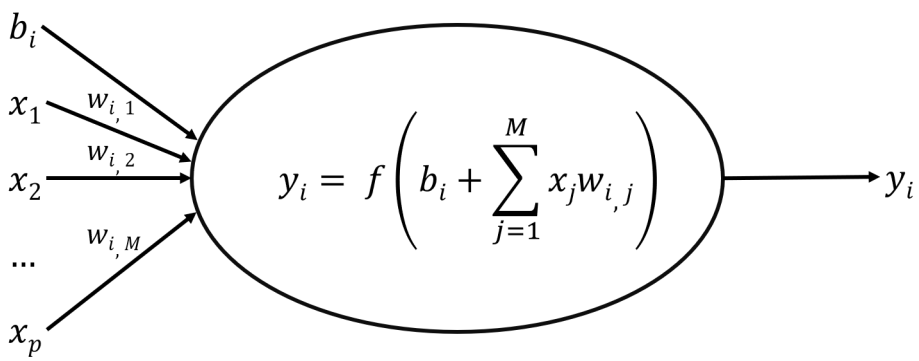


Figure 2.1 Depiction of an artificial neuron.

bias parameters $\mathbf{b} \in \mathbb{R}^N$, where $M \in \mathbb{R}$ represents the number of neurons in the preceding layer. Fig. 2.1 illustrates the i-th neuron of the layer with a bias parameter $b_i \in \mathbb{R}$ and $M \in \mathbb{R}$ connections, each of which receives an input value $x_j \in \mathbb{R}$ from a single neuron j in the preceding layer. The input values $\mathbf{x} \in \mathbb{R}^{M\times 1}$ are multiplied by the weight parameters $\mathbf{w}_i \in \mathbb{R}^{M\times 1}$ and then summed together with the bias parameter b_i. Then, an *activation function* f is applied to the weighted sum s_i to obtain the final output $y_i \in \mathbb{R}$ of the i-th neuron [66, 67]:

$$y_i = f(s_i) = f(\mathbf{w}_i^{\mathrm{T}}\mathbf{x} + b_i) = f(b_i + \sum_{j=1}^{M} x_j w_{i,j}) \tag{2.1}$$

The output y_i can serve as input for neurons in the subsequent layer or be a part of the ANN output. The activation function f is mainly used to introduce non-linear variations, allowing complex properties to be captured in the data. An overview of frequently used activation functions is provided in Sec. 2.1.3. Note that the bias parameter b is not always used in ANNs but sometimes omitted. In network training, the weight and bias parameters, also known as *learnable parameters*, are adjusted to a specific task by applying *optimization algorithms* given a training set of input-output pairs. Applying trained neural networks to data is called *inference*. The neural network architecture describes the pattern according to which the individual artificial neurons are connected and must be adapted to the complexity as well as nature of the task. Typically, ANNs are organized into three different parts:

(1) **Input layer:** The first layer of an ANN, which accepts the input data (e.g. images, measurements) and passes it directly to the first hidden layer.

(2) **Hidden layer:** An intermediate layer between the input and output layers. ANNs can have one or more hidden layers that perform numerous tasks such as data transformation and feature extraction. An ANN with more than two hidden layers is commonly referred to as a deep neural network (DNN) [68].

(3) **Output layer:** The last layer of an ANN computing and representing the network output for given input data.

The simplest ANN architecture is the *feed-forward neural network* (FFNN), in which information can only flow in the forward direction. This means that the output of a layer feeds forward to the neurons of the subsequent layer, and there are no cycles that would allow direct feedback. Typically, FFNNs consist of *fully-connected* (FC) layers in which each neuron is connected to every neuron in the adjacent layers. In Fig. 2.2, the architecture of an FFNN is shown. An early single-layer FFNN (no hidden layer) that uses the Heaviside step function as activation function and acts as linear binary classifier is called Perceptron [69]. A multi-layer Perceptron (MLP) has at least one hidden layer. In general, the universal approximation theorem

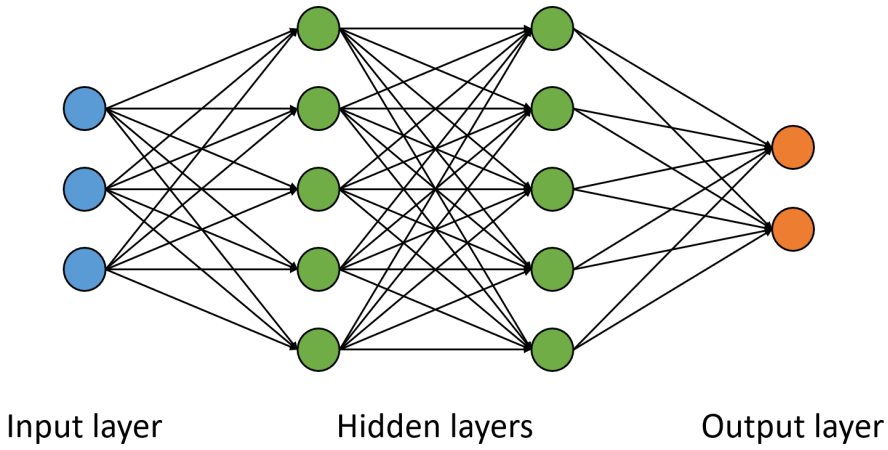


Figure 2.2 Feed-forward neural network with two hidden layers.

proves that an FFNN with one hidden layer and a finite number of neurons can approximate any continuous function in a compact subset $\Omega \in \mathbb{R}^n$ with arbitrary precision [70, 71]. However, the number of neurons can become exponentially large, making the training of such a network very difficult. Thus, neural networks that can be learned more effectively in practice are needed. The universal approximation theorem has also been proved for FFNNs with several hidden layers [72, 73], which are much more efficient than shallow FFNNs in terms of computation and number of neurons.

2.1.2 Neural Network Training

In ANN training, an optimization algorithm is utilized to minimize the loss by adjusting all learnable parameters. Due to the non-linearity and depth (i.e., number of hidden layers) of an ANN, the loss surface is not convex and contains local minima, which makes the training procedure difficult. However, it is sufficient to find a local minimum with a reasonably low loss, which avoids the need to determine the global minimum. *Gradient descent algorithms* combined with *back-propagation* are by far the most common technique to train neural networks. In the following, the individual steps of the training procedure to determine the weight parameters are described:

(1) **Initialization:** Appropriate initialization of learnable parameters is essential for efficient neural network training. The bias values are usually set to be zero. The weights are initialized with random values so that they maintain a certain distribution over several layers, and break the symmetry, allowing each neuron to learn a different feature. In addition, to avoid vanishing or exploding gradients, the activations should be zero-mean and have uniform variance across every layer. The most common methods for weight initialization use

random values in the weight matrix $\mathbf{W}$ sampled from a scaled uniform distribution or a Gaussian distribution with zero-mean. To ensure uniform variance of activations in adjacent layers, the initialization method introduced by Glorot and Bengio [74] uses a distribution with zero-mean and a variance computed as follows:

$$\mathrm{Var}(\mathbf{W}) = \frac{2}{N_{\text{in}} + M_{\text{out}}} \tag{2.2}$$

where N_{in} and M_{out} are the number of neurons in the preceding and current layer, respectively. However, this method was developed for sigmoid-based activation functions, and for functions such as ReLU and PReLU, half of the results are truncated. To solve this problem, He et al. [75] proposed a modified initialization method using a Gaussian distribution with zero-mean and a variance computed as follows:

$$\mathrm{Var}(\mathbf{W}) = \sqrt{\frac{2}{N_{\text{in}}}} \tag{2.3}$$

(2) **Forward pass:** During the forward pass, a batch of input samples is fed into the network and passed sequentially through all layers until the last layer computes the output for each sample. Since the weights are initialized randomly, the performance of the network is usually poor at the beginning of the training.

(3) **Loss computation:** The loss $\mathcal{L}$ is computed to measure the difference between the network output and the ground truth (desired output) for a batch of training samples. Loss functions are discussed in more detail in Sec. 2.1.4.

(4) **Back-propagation (backward pass):** The back-propagation algorithm uses the chain rule of derivatives to compute the *gradient* of the loss with respect to each learnable parameter of the neural network. For a weight parameter $w_{ij} \in \mathbb{R}$ of the weight matrix $\mathbf{W}$, the gradient $\nabla_{w_{ij}}\mathcal{L}$ is computed as follows:

$$\nabla_{w_{ij}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial w_{ij}} \tag{2.4}$$

where y is the final output of the neuron computed according Eq. (2.1).

(5) **Gradient descent:** The gradient descent algorithm performs an update for each learnable parameter of the neural network based on the computed gradients. The update $w_{ij,\text{new}}$ of weight parameter w_{ij} is given as:

$$w_{ij,\text{new}} = w_{ij} - \eta \nabla_{w_{ij}}\mathcal{L} \tag{2.5}$$

where $\eta \in \mathbb{R}^+$ denotes the *learning rate*, which controls how much learnable parameters are adjusted at each update step. The learning rate η must be chosen properly, since a too small value leads to an unnecessarily long training procedure trapped in a sub-optimal local minimum, while a too large value can make the training unstable and even cause divergence. In practice, a learning rate schedule is often used that defines exactly when the learning rate is reduced (or increased) (e.g., [76, 77]).

(6) **Iterations:** Steps (2) - (5) are repeated with all training batches until the loss converges or a maximum number of epochs is reached. The training progress is monitored by the *validation loss* using an unbiased data set.

For any bias parameter b, back-propagation and gradient descent work in the same way as described in steps (1) - (6) for the weight parameter w_{ij}. A major challenge in training very deep neural networks with gradient descent and back-propagation is the *vanishing and exploding gradient problem* [78, 79]. It occurs mainly because the gradient flows backwards through all layers using the chain rule of derivatives, i.e., the gradients of the neurons are obtained by multiplying the gradients in later layers (closer to the output layer). Vanishing gradients result from an accumulation of small gradients in later layers and lead to insignificant parameter updates, while exploding gradients result from an accumulation of large gradients and lead to very large parameter updates. Both effects lead to unstable training or poorly trained neural networks. In addition to the back-propagation algorithm, improper parameter initialization and inappropriate choice of activation function also contribute to the vanishing and exploding gradient problem. Methods to cope with this challenge are discussed later in this chapter.

The most commonly used gradient descent algorithms are typically refinements or extensions of the *stochastic gradient descent* (SGD) algorithm [80]. Compared to vanilla gradient descent, SGD performs a parameter update based on each training sample instead of the entire training data set. Since frequent updates with high variance lead to oscillations of the loss, in practice parameters are typically updated based on several training samples (mini-batch) instead of a single one. SGD oscillates strongly in areas of the loss surface with long and narrow ravines, which are often located near local minima [81]. Moreover, SGD often gets trapped in sub-optimal local minima and converges relatively slowly. SGD with *momentum* addresses these problems by accumulating an exponentially decaying moving average of previous gradients [66, 82]:

$$w_{ij,\text{new}} = w_{ij} - v_{ij,\text{new}} \tag{2.6a}$$

$$v_{ij,\text{new}} = \gamma v_{ij} + \eta \nabla_{w_{ij}} \mathcal{L} \tag{2.6b}$$

where v_{ij} represents the velocity vector and $\gamma \in [0, 1]$ is the momentum term that regulates the influence of the previous gradients. This accelerates SGD in the relevant direction, resulting in faster convergence and a reduction in oscillations. Since the gradient magnitude of the learnable parameters varies greatly and changes during network training, a pre-defined global learning rate is often not appropriate. Thus, optimizers have been introduced which adjust the learning rate for each update step and parameter individually (e.g. [83, 84, 85, 86]). This can greatly improve the speed and robustness of the learning process. *Root mean squared propagation* (RMSProp) [84] is a widely used and very effective extension of SGD that addresses the problem of very strongly decreasing learning rates of *adaptive gradient* (AdaGrad) [83]. The key idea of RMSprop is to modify the learning rate for each parameter based on an exponentially decaying average of squared gradients $(\nabla_{w_{ij}} \mathcal{L})^2 \in \mathbb{R}$:

$$w_{ij,\text{new}} = w_{ij} - \eta \frac{\nabla_{w_{ij}} \mathcal{L}}{\sqrt{v_{ij,\text{new}}} + \epsilon} \tag{2.7a}$$

$$v_{ij,\text{new}} = \beta v_{ij} + (1 - \beta)(\nabla_{w_{ij}} \mathcal{L})^2 \tag{2.7b}$$

where $\epsilon \in \mathbb{R}^+$ is a very small value added to prevent division by zero and $\beta \in \mathbb{R}^+$ defines the influence of the momentum (typically $\beta = 0.9$ is used). *Adaptive moment estimation* (Adam) is another very popular optimizer that determines adaptive learning rates for each network parameter [85]. Besides using the exponentially decaying average of squared previous gradients v_{ij} as in RMSProp (second momentum), the exponentially decaying average of previous gradients m_{ij} is also included as in SGD with momentum (first momentum). Since m_{ij} and v_{ij} are biased toward zero (especially during the first update steps), Adam additionally includes bias-corrected estimates of the first $\hat{m}_{ij}$ and second $\hat{v}_{ij}$ moments:

$$w_{ij,\text{new}} = w_{ij} - \eta \frac{\hat{m}_{ij,\text{new}}}{\sqrt{\hat{v}_{ij,\text{new}}} + \epsilon} \tag{2.8a}$$

$$\hat{m}_{ij,\text{new}} = \frac{m_{ij,\text{new}}}{1 - \beta_1} \tag{2.8b}$$

$$\hat{v}_{ij,\text{new}} = \frac{v_{ij,\text{new}}}{1 - \beta_2} \tag{2.8c}$$

$$m_{ij,\text{new}} = \beta_1 m_{ij} + (1 - \beta_1) \nabla_{w_{ij}} \mathcal{L} \tag{2.8d}$$

$$v_{ij,\text{new}} = \beta_2 v_{ij} + (1 - \beta_2)(\nabla_{w_{ij}} \mathcal{L})^2 \tag{2.8e}$$

where $\beta_1 \in [0, 1]$ and $\beta_2 \in [0, 1]$ define the influence of the moments m_{ij} and v_{ij}, respectively. Typically $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are used. *AMSGrad* is an extension of Adam that avoids large abrupt changes in learning rate to improve the convergence and generalization properties of the optimizer [86]. Instead of the exponentially decaying average, the maximum of the previous squared gradients is used to update the parameters.

$$w_{ij,\text{new}} = w_{ij} - \eta \frac{\hat{m}_{ij,\text{new}}}{\sqrt{\hat{v}_{ij,\text{new}}} + \epsilon} \tag{2.9a}$$

$$\hat{m}_{ij,\text{new}} = \frac{m_{ij,\text{new}}}{1 - \beta_1} \tag{2.9b}$$

$$\hat{v}_{ij,\text{new}} = \frac{\max(v_{ij,\text{new}}, \hat{v}_{ij})}{1 - \beta_2} \tag{2.9c}$$

$$m_{ij,\text{new}} = \beta_1 m_{ij} + (1 - \beta_1) \nabla_{w_{ij}} \mathcal{L} \tag{2.9d}$$

$$v_{ij,\text{new}} = \beta_2 v_{ij} + (1 - \beta_2) (\nabla_{w_{ij}} \mathcal{L})^2 \tag{2.9e}$$

2.1.3 Activation Functions

In a neural network, the *activation function* f transforms the sum s of the bias value and weighted input values into the final output of the neuron. The activation function is usually a *non-linear function* that allows learning complex structures in the data, but it can also be the *identity function* (linear function). Activation functions are typically differentiable, which is required for training neural networks using back-propagation. The effect of their derivatives on the gradient is an important issue. In the literature, several non-linear functions have been introduced (e.g., [75, 87, 88, 89]). The *sigmoid function* (σ) squashes s to the range of $[0, 1]$:

$$\sigma(s) = \frac{1}{1 + e^{-s}} \tag{2.10}$$

Since the output of the sigmoid function is not zero-centered, all gradients of the parameters will either all be positive or all be negative, which can lead to zig-zag dynamics in the gradient updates. This can be overcome by using the zero-centered *hyperbolic tangent* (tanh) function, which squashes s to the range of $[-1, 1]$:

$$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} \tag{2.11}$$

Compared to sigmoid, tanh leads to easier and faster learning. However, both sigmoid and tanh saturate relatively quickly due to their boundedness, resulting

in gradients close to zero and vanishing gradients in deep neural networks. The non-bounded and non-zero-centered *Rectified Linear Unit* (ReLU) is one of the most commonly used activation functions in neural networks [87, 90], where all negative values of s are set to zero.

$$\text{ReLU}(s) = \max(0, s) \tag{2.12}$$

ReLU is computationally cheap and has been shown to significantly accelerate the convergence speed when training neural networks with SGD compared to sigmoid as well as tanh [91]. In addition, since ReLU is not saturating for positive values of s, deep neural networks do not suffer from the vanishing gradient problem [87]. However, since ReLU outputs zero and has gradients of zero for all negative values of s, ReLU can saturate at zero and become inactive (dead) forever. To reduce this so-called *dying ReLU problem*, several extended variants have been introduced. The *Leaky Rectified Linear Unit* (LReLU) [88] allows small negative outputs when s is less than zero by introducing a predefined *leakage parameter* $a = 0.01$:

$$\text{LReLU}(s) = \begin{cases} s, & \text{if } s > 0 \\ -as, & \text{otherwise} \end{cases} \tag{2.13}$$

Compared to ReLU, LReLU is more balanced and may therefore learn faster. However, LReLU does not retain the sparse characteristic of ReLU, where only subsets of neurons are active due to mapping negative values to zero. The *Parametric Rectified Linear Unit* (PReLU) [75] is a generalization of LReLU where the leakage parameter is not predefined but learned along with the other network parameters during training. The *softmax function* is commonly used as an activation function in the output layer of neural networks acting as *multi-class classifiers* [66]. It normalizes the output values of the k neurons in the layer to a probability distribution consisting of k probabilities that always sum to one. The final output $\text{softmax}(s_i) \in [0, 1]$ of the i-th neuron in the layer is computed as follows:

$$\text{softmax}(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{k} e^{s_j}} \tag{2.14}$$

2.1.4 Loss Functions

A *loss function* (also called cost function) quantifies the difference between the output computed by the algorithm and the ground truth (desired output). The loss $\mathcal{L}$ usually represents the average of the individual sample losses of a mini-batch or an entire data set:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i \tag{2.15}$$

where N denotes the number of samples and $\mathcal{L}_i$ is the loss of sample i. For neural networks, several loss functions exist for both classification and regression tasks.

In *classification tasks*, each sample is assigned to exactly one of two (binary) or multiple possible classes. Thus, one ground truth label $\tilde{y}_i$ exists for each sample i. The most common loss function used for classification tasks is the *cross-entropy* $\mathbb{H}$, which originates from information technology and generally measures the difference between two probability distributions p and q [92]. For discrete probability distributions, the cross-entropy $\mathbb{H}$ is given as:

$$\mathbb{H}(p, q) = -\sum_{j} p(j) \log(q(j)) \tag{2.16}$$

where $p(j)$ and $q(j)$ denote the probabilities for event j in p and q, respectively. For a sample i, the cross-entropy loss is computed as follows:

$$\mathcal{L}_i^{\text{CE}} = -\sum_{j=1}^{C} \tilde{y}_{i,j} \log(y_{i,j}) \tag{2.17}$$

where C is the number of possible classes and $\tilde{\mathbf{y}}_i \in \{0, 1\}^C$ is a one-hot encoding vector representing the ground truth labels, i.e., a vector whose elements are all zeros except at the index of the true class. The vector $\mathbf{y}_i \in [0, 1]^C$ represents the normalized class probabilities computed by the neural network for sample i, i.e., $\sum_{j=1}^{C} y_{i,j} = 1$, where $y_{i,j}$ represents the probability for class j. In *multi-class classification* tasks, the cross-entropy is often referred to as the *categorical cross-entropy* and the softmax activation function is used in the output layer of the neural network. Since $\tilde{\mathbf{y}}_i$ is one-hot encoding vector, Eq. (2.17) can be simplified:

$$\mathcal{L}_i^{\text{CE}} = -\log(y_{i,j}) \tag{2.18}$$

where j is the index of the true class. For *binary classification* tasks, Eq. (2.17) can be written as:

$$\mathcal{L}_i^{\text{BCE}} = -(\tilde{y}_i \log(y_i) + (1 - \tilde{y}_i) \log(1 - y_i)) \tag{2.19}$$

where $\tilde{y}_i \in \{0, 1\}$ defines the ground truth class and $y_i \in [0, 1]$ is the probability for class 1 computed by the neural network. The binary cross-entropy is often combined with the sigmoid activation function in the output layer. To cope with strong class imbalance in neural network training (e.g., object detection), the *focal loss* has been introduced by Lin et al. [93]. Here, the binary cross-entropy is dynamically scaled, with the scaling decreasing with increasing confidence in the correct class. The focal loss $\mathcal{L}_i^{\text{FL}}$ for a sample i is given as:

$$\mathcal{L}_i^{\text{FL}} = -(1 - y_i)^{\gamma} \log(y_i) \tag{2.20}$$

where $\gamma \geq 0$ is a *focusing parameter* that down-weights easy samples and emphasizes hard samples. Classification tasks can also be considered from the perspective

of set theory. In this case, loss functions based on the Dice coefficient (e.g., [94]), Jaccard similarity coefficient, and cosine similarity can be used.

In *regression tasks*, a continuous real-value quantity is predicted instead of a class label, and the loss function is typically based on a ℓ_p norm $\|.\|_p$ [66]. The ℓ_1 norm $\|.\|_1$ is used in the $\mathcal{L}^{\ell_1}$ loss, which is also known as *mean absolute error* (MAE). For an individual sample i, the $\mathcal{L}_i^{\ell_1}$ loss between the predicted vector $\mathbf{y}_i$ and the ground truth vector $\tilde{\mathbf{y}}_i$ is computed as:

$$\mathcal{L}_i^{\ell_1} = \|\mathbf{y}_i - \tilde{\mathbf{y}}_i\|_1 \tag{2.21}$$

The l_1 norm is robust to outliers, but does not always have a unique solution, making its optimization difficult. The ℓ_2 norm $\|.\|_2$, also called *Euclidean norm*, has a unique solution and is typically preferred over the ℓ_1 norm. To avoid the square root in norm computation and stabilize the training, the squared ℓ_2 norm is used in the $\mathcal{L}^{\ell_2}$ loss, which is also known as *mean square error* (MSE). For a sample i, the $\mathcal{L}_i^{\ell_2}$ loss is given as follows:

$$\mathcal{L}_i^{\ell_2} = \|\mathbf{y}_i - \tilde{\mathbf{y}}_i\|_2^2 \tag{2.22}$$

However, since squared terms strongly increase the error, the $\mathcal{L}^{\ell_2}$ loss is very sensitive to outliers. The *Huber loss* combines the advantages of both loss functions [95]. It behaves like the $\mathcal{L}^{\ell_1}$ loss for errors larger than the threshold $\delta \in \mathbb{R}^+$, and like the $\mathcal{L}^{\ell_2}$ loss otherwise. Thus, it is more robust than the $\mathcal{L}_i^{\ell_2}$ loss. For a sample i, the loss between the predicted vector $\mathbf{y}_i \in \mathbb{R}^K$ and the ground truth vector $\tilde{\mathbf{y}}_i \in \mathbb{R}^K$ is computed by summing the per-element Huber loss:

$$\mathcal{L}_i^{\mathrm{H}} = \sum_{j=1}^{D} \begin{cases} \frac{1}{2}(y_{i,j} - \tilde{y}_{i,j})^2, & if \left|y_{i,j} - \tilde{y}_{i,j}\right| \leq \delta \\ \delta(\left|y_{i,j} - \tilde{y}_{i,j}\right| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{2.23}$$

2.1.5 Neural Network Regularization

Training a deep neural network to generalize well to new data is a challenging task. A network with a capacity that exceeds the training data set is prone to *overfitting*. In this case, even the smallest random fluctuations and noise in the training data are learned and reproduced, resulting in poor generalization and performance on new data. *Regularization methods* have been introduced to reduce the network complexity during training, preventing overfitting. The commonly used ℓ_2 regularization prevents overfitting by forcing the neural network to have small weights (but not equal to zero). In addition, the network is encouraged to use all inputs consistently. The loss function is extended with a regularization term defined by the ℓ_2 norm (Euclidean norm) of the weights:

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \psi \sum_{j=1}^{N} w_j^2 \tag{2.24}$$

where $\psi \in \mathbb{R}$ is the *regularization rate* and N represents the number of weights in the neural network [67]. The frequently used ℓ_1 regularization extends the loss function with a regularization term defined by the ℓ_1 norm of the weights:

$$\mathcal{L}_{\text{reg}} = \mathcal{L} + \psi \sum_{j=1}^{N} |w_j| \tag{2.25}$$

It ensures that the network weights are very small or even zero for irrelevant features. This leads to sparse weight matrices, so that only meaningful features are considered [67]. In general, ℓ_2 regularization performs better than ℓ_1 regularization and is easier to adapt. ℓ_1 and ℓ_2 regularization can also be combined, which is known as *elastic net regularization*. The *maximum norm method* provides regularization and avoids large weights by constraining the norm of weights to an absolute upper bound defined by a constant [96]. To enforce this constraint, the projected gradient descent is used [67]. An effective and simple method to address the problem of overfitting is *Dropout* [97]. During training, individual neurons with all their connections are temporarily removed (dropped) from the network with a probability p. Thus, dropout can be considered as an ensemble technique where multiple sub-networks are trained jointly. Dropout prevents strong dependencies between neurons (co-adaptation), which may lead to overfitting. Moreover, it forces the neural network to favor redundant representations. In general, dropout increases training time but improves network performance. *Batch normalization* (BN) is a commonly used method that normalizes the output of layers based on the mean $\mu_b \in \mathbb{R}$ and variance $\sigma_b^2 \in \mathbb{R}^+$ of the current mini-batch [98]. This reduces the internal covariate shift, leading to stabilization and acceleration of the neural network training. Since a training sample is presented several times in different random mini-batches, μ_b and σ_b^2 can be considered as noise for the training sample providing regularization [99]. For the output y_i of neuron i, BN is computed during network training as follows:

$$\text{BN}(y_i) = \gamma_i \hat{y}_i + \beta_i \tag{2.26}$$

where the learnable parameters γ_i and β_i represent the scale and the shift, respectively. The normalized output $\hat{y}_i$ ensuring zero mean and unit variance is defined as:

$$\hat{y}_i = \frac{y_i - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} \tag{2.27}$$

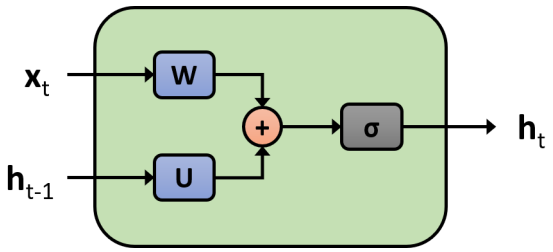


Figure 2.3 Vanilla RNN architecture.

where $\epsilon \in \mathbb{R}^+$ is a very small value added to prevent division by zero. The mini-batch statistics μ_b and σ_b^2 are given as:

$$\mu_b = \frac{1}{N} \sum_{j=1}^{N} y_j \tag{2.28a}$$

$$\sigma_b^2 = \frac{1}{N} \sum_{j=1}^{N} (y_j - \mu_b)^2 \tag{2.28b}$$

where N is the number neurons in the same layer as neuron i. For inference, a fixed mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}^+$ are used, which are computed based on the whole training data or approximated by running statistics during training.

2.1.6 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a special class of neural network architectures designed to solve *sequence learning tasks*. Unlike feed-forward layers, recurrent layers also have cyclic (recurrent) connections that allow information encoded in the hidden state to flow backwards and persist within the network. Thus, sequential dependencies can be be exploited [67].

The *vanilla RNN* shown in Fig. 2.3 is the most basic RNN layer and has been used in a variety of applications. The *hidden state* $\mathbf{h}_t \in \mathbb{R}^N$ of a vanilla RNN layer with N neurons at time t is computed as follows:

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

where f denotes a sigmoid (σ) or tanh activation function, $\mathbf{h}_{t-1} \in \mathbb{R}^N$ is the previous hidden state, and $\mathbf{x}_t \in \mathbb{R}^M$ is the output of the preceding layer with M neurons. $\mathbf{W} \in \mathbb{R}^{M \times N}$ and $\mathbf{U} \in \mathbb{R}^{N \times N}$ are the weight parameters, and $\mathbf{b} \in \mathbb{R}^N$ denotes the bias parameters. Note that the hidden state $\mathbf{h}_t$ also represents the output of the vanilla RNN layer at time point t. Since the network architecture includes cyclic (recurrent) connections, an RNN cannot be trained directly using the back-propagation algorithm described in Sec. 2.1.2. However, the RNN can

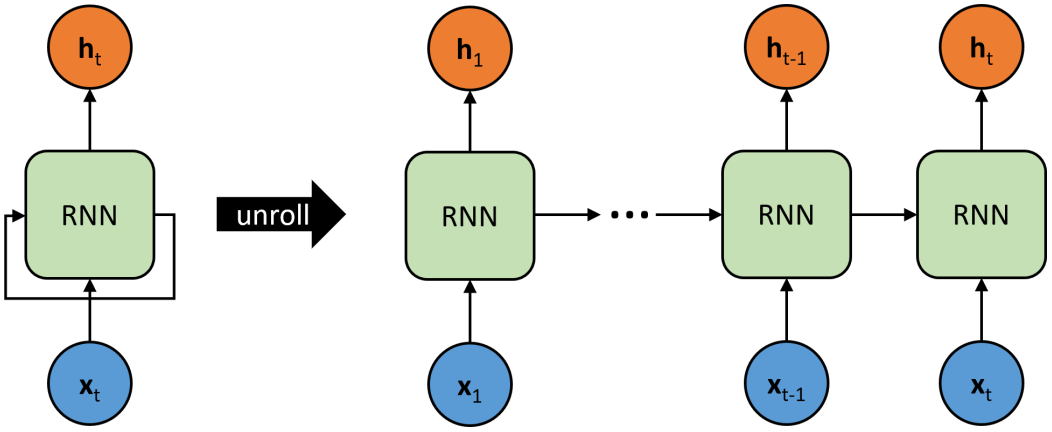


Figure 2.4 Schematic illustration of unrolling an RNN over time.

be transformed into an FFNN by unrolling it over time (see Fig. 2.4), where the network is replicated for each time point. Connections with shared recurrent weights link consecutive time replicas, resulting in a very deep FFNN that can be trained by gradient descent algorithms combined with back-propagation. This training strategy for RNNs is called *back-propagation through time* (BPTT) [100]. However, since BPTT also back-propagates gradients through time using the chain rule of derivatives, vanilla RNNs are prone to vanishing and exploding gradients (see Sec. 2.1.2) and thus not suitable for learning *long-term dependencies* [78, 79].

The *Long Short-Term Memory* (LSTM) was developed to avoid the vanishing and exploding gradient problem in BPTT [101, 102]. A major component of the LSTM is the *cell state* $\mathbf{c}$, which acts as a *long-term memory*. Internal mechanisms called *gates* interact with each other to regulate the flow of information and update the hidden state $\mathbf{h}$ and the cell state $\mathbf{c}$. In the following, we consider one forward pass through an LSTM layer with N units that receive the input vector $\mathbf{x}_t \in \mathbb{R}^M$ at time point t (see Fig. 2.5). Matrices $\mathbf{W} \in \mathbb{R}^{M \times N}$ and $\mathbf{U} \in \mathbb{R}^{N \times N}$ represent weight parameters, and vectors $\mathbf{b} \in \mathbb{R}^N$ denote bias parameters. First, the *forget gate* $\mathbf{f}_t \in \mathbb{R}^N$ [102] determines which information is removed from the previous cell state $\mathbf{c}_{t-1} \in \mathbb{R}^N$:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f^{\mathrm{T}} \mathbf{x}_t + \mathbf{U}_f^{\mathrm{T}} \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.30}$$

The next step is to define what new information enters the cell state. The *input gate* $\mathbf{i}_t \in \mathbb{R}^N$ decides which information is updated and a tanh layer extracts candidate values $\tilde{\mathbf{c}}_t \in \mathbb{R}^N$ that could be added to the cell state:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i^{\mathrm{T}} \mathbf{x}_t + \mathbf{U}_i^{\mathrm{T}} \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.31}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_{\tilde{c}}^{\mathrm{T}} \mathbf{x}_t + \mathbf{U}_{\tilde{c}}^{\mathrm{T}} \mathbf{h}_{t-1} + \mathbf{b}_{\tilde{c}}) \tag{2.32}$$

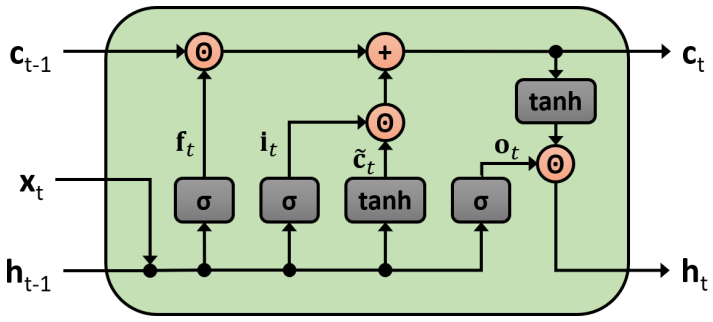


Figure 2.5 LSTM architecture.

Following, the new cell state $\mathbf{c}_t \in \mathbb{R}^N$ is computed as:

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

where $\odot$ denotes the Hadamard product. Finally, the new hidden state $\mathbf{h}_t \in \mathbb{R}^N$ is determined, which is also the output of the LSTM. The *output gate* $\mathbf{o}_t \in \mathbb{R}^N$ decides which information of the new cell state $\mathbf{c}_t$ is used. To ensure that only the desired information is output, $\mathbf{o}_t$ is multiplied by the cell state that previously passed through a tanh activation:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o^{\mathrm{T}}\mathbf{x}_t + \mathbf{U}_o^{\mathrm{T}}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.34}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.35}$$

The *Gated Recurrent Unit* (GRU) was introduced to reduce the number of learnable parameters and the computational burden of the LSTM, while maintaining the ability to learn both short- and long-term dependencies in sequential data [103]. To get rid of a gate and its corresponding learnable parameters, the GRU combines the forget and input gates into a single *update gate* $\mathbf{z}$. A second gate of the GRU is called *reset gate* $\mathbf{r}$. Basically, these two gates decide which information is kept and which information is removed. In addition, the GRU has just a hidden state and no cell state. The architecture of the GRU is sketched in Fig. 2.6. In the following, we consider one forward pass through an GRU layer with N units that receive the input vector $\mathbf{x}_t \in \mathbb{R}^M$ at time point t. $\mathbf{W} \in \mathbb{R}^{M \times N}$ and $\mathbf{U} \in \mathbb{R}^{N \times N}$ are weight matrices, and $\mathbf{b} \in \mathbb{R}^N$ represents bias parameters. First, the reset gate $\mathbf{r}_t \in \mathbb{R}^N$ and the update gate $\mathbf{z}_t \in \mathbb{R}^N$ are computed as:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r^{\mathrm{T}}\mathbf{x}_t + \mathbf{U}_r^{\mathrm{T}}\mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2.36}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z^{\mathrm{T}}\mathbf{x}_t + \mathbf{U}_z^{\mathrm{T}}\mathbf{h}_{t-1} + \mathbf{b}_z) \tag{2.37}$$

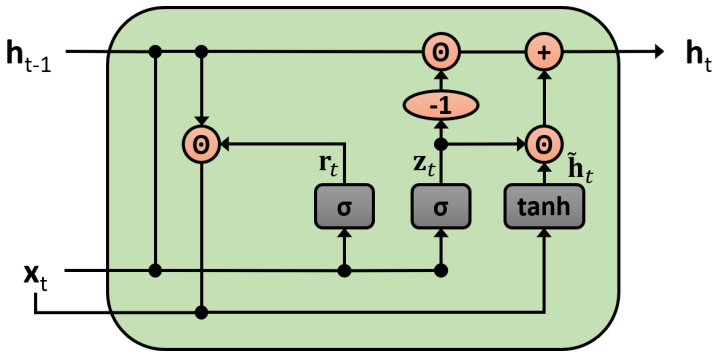


Figure 2.6 GRU architecture.

Following, the candidate hidden state $\tilde{\mathbf{h}}_t \in \mathbb{R}^N$ is computed using the reset gate $\mathbf{r}_t$:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{\tilde{\mathbf{h}}}^{\mathrm{T}}\mathbf{x}_t + \mathbf{U}_{\tilde{\mathbf{h}}}^{\mathrm{T}}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_{\tilde{\mathbf{h}}}) \tag{2.38}$$

Finally, the new hidden state $\mathbf{h}_t \in \mathbb{R}^N$ is determined by weighting the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^N$ and the candidate hidden state $\tilde{\mathbf{h}}_t$ using the update gate $\mathbf{z}_t$:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{2.39}$$

2.1.7 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special class of neural networks that use convolutional layers to extract important information from multidimensional input (e.g., 2D image, 3D volume). In convolutional layers, a *linear convolution operation* is applied to the input using *filters* (also called kernels), followed by a non-linear activation function. The filters have a defined size $k_h \times k_w$ (e.g., height and width), which is significantly smaller than the input. However, the number of filter channels C is equal to the number of input channels. Each filter slides step-wise across the entire input and computes for each position the dot product between the *learnable filter weights* and the corresponding region of the input. This results in an output called *feature map*. Applying N filters to the input yields N feature maps, each representing different features [104]. Following, the feature maps are passed through a non-linear activation function f (e.g., sigmoid, tanh, ReLU). The output $\mathbf{y}$ of a convolutional layer receiving input $\mathbf{x}$ is defined as:

$$\mathbf{y} = f(\mathbf{W} * \mathbf{x} + \mathbf{b}) \tag{2.40}$$

where $*$ denotes the convolution operation, $\mathbf{W} \in \mathbb{R}^{k_h \times k_w \times C \times N}$ represents the weight parameters of the N filters, and $\mathbf{b} \in \mathbb{R}^N$ are the bias parameters. Since the weights of

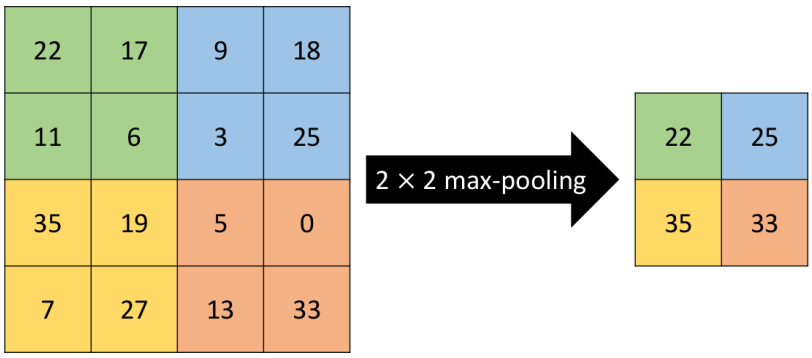


Figure 2.7 Example of a 2×2 max-pooling operation with a stride of 2.

the filters are fixed for the entire input (referred to as weight sharing), convolutional layers require less learnable parameters compared to FC layers, which increases efficiency. Stacking convolutional layers to form deep CNNs allows learning of low-level features in early layers (e.g., edges) as well as high-level or abstract features in later layers (e.g., objects). In CNN training, the filter weights and the bias parameters are optimized for a specific task and data set.

To achieve local translation invariance and reduce the number of convolutional layers, the feature maps are down-sampled, i.e., the spatial resolution is reduced. In general, filters applied to a down-sampled feature map have a higher receptive field than filters applied to the original feature map. *Down-sampling* can be achieved by using a convolution layer with increased stride, which defines the step size the filter slides across the input. However, a more efficient and robust approach is to employ *pooling layers*, which apply a pooling operation to each feature map independently. The most common pooling operations are *max-pooling* and *average-pooling*, which provide the maximum or average value for each patch of the feature map. Fig. 2.7 shows an example of a 2×2 max-pooling operation with a stride of 2. In contrast to pooling layers, *dilated convolutions* (also called atrous convolutions) introduce gaps between the weights in the convolutional filters to increase the receptive field without additional learnable parameters [105]. To increase the spatial resolution of feature maps (e.g., for networks that perform image-to-image mapping), simple and effective *up-sampling layers* that have no learnable parameters can be used. Common up-sampling layers perform *interpolation* (e.g., nearest neighbor, bilinear, bicubic). Note that interpolation can also be used for down-sampling. *Transposed convolution* uses learnable weights instead of a predefined interpolation method to increase the spatial resolution of the input [106].

The basic CNN architecture includes stacks of convolutional and pooling layers, followed by one or more fully-concatenated (FC) layers. The first two layer types are used for feature extraction, while FC layers are employed to compute a final output vector or value based on the extracted features [107]. However, numerous improved CNN architectures have been introduced for different tasks and applica-

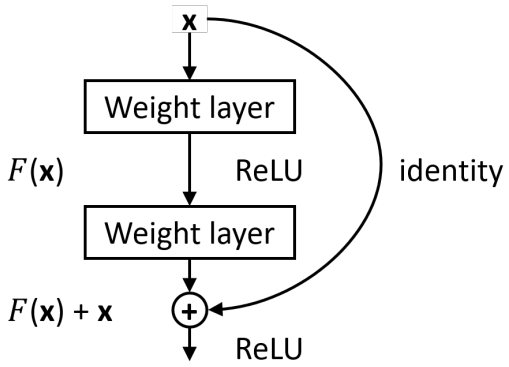


Figure 2.8 Residual block.

tions. *AlexNet* has five convolutional layers and three FC layers [91]. It employs max-pooling layers, ReLU activations, and dropout. In addition, *local response normalization* was introduced. The *VGGNet* architecture is based on AlexNet, but convolutional layers with a large receptive field are replaced by three stacked convolutional layers with a small receptive field (kernel size of 3, stride of 1) [108]. This significantly reduces the number of learnable parameters. Several variants of the VGGNet have been proposed, differing only in the total number of layers in the network. In 2014, *GoogleLetNet* was introduced, which uses the *inception module* to analyze information at different scales and aggregate it locally [109]. The inception module processes the input feature maps in parallel by three convolutional layers with different filter sizes (1, 3, and 5) and a max-pooling layer. The respective outputs are then concatenated. To reduce the number of learnable parameters and computations, 1×1 convolutional layers which perform channel-wise pooling are applied before the convolutional layers with large filter sizes (3 and 5) and after the max-pooling layer. The Inception module was later improved (e.g., by using spatially separable convolution and batch normalization). In 2015, the *Residual Network* (ResNet) architecture has been introduced yielding superior results [110]. The key component of ResNet is the *residual block*, which improves the training of very deep neural networks. A residual block is a stack of layers with connections that skip one or more layers (see Fig. 2.8). These connections are called skip connections and provide an alternative shortcut for the gradient. This avoids the vanishing and exploding gradient problem. The *Densely Connected Convolutional Network* (DenseNet) architecture is a logical extension of ResNet in which each layer is directly connected to every other layer within a dense block [111]. *MobileNets* are lightweight CNN architectures that employ depth-wise separable convolutional layers to significantly reduce the number of learnable parameters compared to a CNN with conventional convolutional layers and the same depth [112].

2.1.8 Bayesian Deep Learning

Bayesian deep learning aims to enable neural networks to accurately quantify their *predictive uncertainty*, which can result from both inherent noise in the data (aleatoric uncertainty) as well as the uncertainty of the network parameters (epistemic uncertainty). *Aleatoric uncertainty* can be modeled by placing a distribution over the output of the neural network and cannot be reduced by providing more training data. *Epistemic uncertainty* results from a lack of knowledge due to limited or inappropriate training data and can be captured by *Bayesian Neural Networks* (BNNs). Conventional neural network training leads to deterministic point estimates of the learnable parameters, which can be interpreted as *maximum likelihood estimation*. In contrast, BNNs specify a *prior distribution* $P(\mathbf{W})$ over the learnable parameters $\mathbf{W}$ (e.g., Gaussian distribution), which is updated to the *posterior distribution* $P(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ by using *Bayesian inference* [113]:

$$P(\mathbf{W}|\mathbf{X}, \mathbf{Y}) = \frac{P(\mathbf{Y}, \mathbf{X}|\mathbf{W})P(\mathbf{W})}{\int P(\mathbf{Y}, \mathbf{X}|\mathbf{W})P(\mathbf{W})d\mathbf{W}} \tag{2.41}$$

where $P(\mathbf{Y}, \mathbf{X}|\mathbf{W})$ denotes the likelihood function that defines the variation of the training labels $\mathbf{Y}$ given the neural network inputs $\mathbf{X}$. The posterior distribution $P(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ captures all plausible learnable parameters given the training data. However, Eq. (2.41) is typically not tractable for neural networks, so different approximations have been introduced (e.g., [114, 115, 116, 117, 118]). In *variational inference*, the posterior distribution is approximated with a simple surrogate distribution $Q_\theta(\mathbf{W})$ parameterized by θ. For example, $Q_\theta(\mathbf{W})$ can be assumed to be Gaussian, with θ representing mean and variance. In neural network training, the parameters θ are learned instead of directly optimizing the original network parameters $\mathbf{W}$ by maximum likelihood or maximum a posteriori estimation [113]. This involves maximizing the *Evidence Lower Bound* (ELBO), which equals minimizing the *Kullback-Leibler* (KL) divergence between $Q_\theta(\mathbf{W})$ and the true posterior distribution $P(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ (e.g., [114, 115, 119]). Thus, the intractable problem of averaging over all learnable parameters (referred to as marginalization) is replaced by an optimization task [113]. To train such BNNs with a standard gradient descent algorithm, the gradient of the ELBO with respect to θ can be estimated using the *reparameterization trick* introduced by Kingma and Welling [119]. *Dropout variational inference* is a common and simple approach to approximate Bayesian inference in large neural networks [117, 118]. Basically, this approach is equivalent to performing variational inference [113]. Here, dropout (see Sec. 2.1.5) is applied not only during training but also at test time to sample from the approximate posterior by performing *stochastic forward passes* (Monte Carlo dropout). Dropout can be interpreted as an approximation to the posterior with a distribution that is a mixture of two Gaussians with small variances. The mean of one of the Gaussians is fixed at zero. Bayesian learning based on *stochastic gradient Langevin dynamics*

is a simple approach compared to variational inference [120]. During iterative learning from small mini-batches, noise is added to each estimate of the gradient, resulting in iterates that converge to samples from the true posterior of the learnable parameters. *Ensemble learning* is a simple and powerful scheme providing predictive uncertainty of neural networks. Predictions for the same input obtained from different networks are combined to determine a final prediction, with the variance interpreted as epistemic uncertainty. To obtain diverse network ensembles, various approaches have been introduced such as stochastic weight averaging (SWA), snapshot ensembling, and random parameter initialization. *SWA* combines weights obtained at different stages of the network training and is based on the assumptions that the SGD trajectory provides useful information about the shape (local geometry) of the true posterior distribution [121]. In *snapshot ensembling*, the neural network converges to and escapes from several local minima using a cyclic (cosine) learning rate schedule. At each local minimum, a snapshot of the neural network is taken and used as ensemble member. In other ensemble approaches, the neural network is trained several times with *different parameter initializations* [122, 123].

2.2 Object Detection

Object detection is a computer vision task that involves identifying and localizing semantic instances of specific objects (e.g., pedestrians, animals, vehicles) in images. The detection result represents the exact position of an object by the *axis-aligned bounding box* or a *single point coordinate* (e.g., centroid). In this section, we describe deep learning methods for object detection in general computer vision (Sec. 2.2.1), and then provide an overview of previous classical and deep learning methods for particle detection in fluorescence microscopy images (Sec. 2.2.2). For more comprehensive reviews, we refer to [55, 56, 124, 125].

2.2.1 Deep Learning Methods in Computer Vision

In recent years, numerous deep learning methods for object detection in computer vision have been introduced providing state-of-the art performance. In general, these methods can be classified into two-stage and single-stage detectors.

Two-stage detectors consider object detection as a classification task, where *region proposals* are generated and then classified as object or background. Such detectors are usually more flexible and accurate than single-stage detectors. *Region-based Convolutional Neural Network* (R-CNN) is one of the first methods that successfully use CNNs for bounding box-based object detection [126]. A selective search algorithm is applied to generate and extract class-independent region proposals from the image that might contain objects (e.g., candidate bounding boxes) [127]. These region proposals, cropped from the image and warped to the same size, are

used as input to a deep CNN (e.g., AlexNet [91]) for feature extraction. Following, the extracted features are fed into a set of class-specific linear support vector machines (SVMs) to distinguish the region proposals into object classes. Finally, a simple class-specific linear regressor is employed to refine the bounding boxes. Since CNN-based feature extraction is performed independently for each region proposal in every image, the training and application of R-CNN is slow and storage intensive. Moreover, since R-CNN is a multi-stage pipeline involving separate components (region proposal, feature extractor, classifier, bounding box regressor), it cannot be trained end-to-end. *Fast R-CNN* has been proposed to overcome some of these limitations [128]. A CNN enables *end-to-end detector training* by simultaneously learning classification and bounding box regression. Moreover, training and application is significantly faster compared to R-CNN by sharing the convolution computations for all region proposals. In Fast R-CNN, instead of individual region proposals, the entire image is fed into the CNN. To generate a fixed-size feature vector for region proposals of arbitrary size, the feature map of the last convolutional layer is fed into a region of interest (ROI) pooling layer. The extracted feature vector can then be passed through fully-connected (FC) layers. Finally, a softmax layer determines the object class of the region proposal and a FC layers computes the offset values of the bounding box. The classification and regression task are learned by minimizing a multi-task loss function which is composed of the cross-entropy and the Huber loss [95]. *Faster R-CNN* integrates the region proposal algorithm into the CNN to improve speed and detection performance [129]. A single unified network architecture consists of the so-called *Region Proposal Network* (RPN) and the Fast R-CNN with shared convolutional layers. The RPN is a *fully-convolutional network* (FCN) that acts as an *attention mechanism* for the Fast R-CNN. It generates multiple region proposals with different scales as well as aspect ratios (anchors) and computes the corresponding binary objectness scores. To reduce redundancy, non-maximum suppression (NMS) is applied to the region proposals based on their objectness scores [129]. *Mask R-CNN* extends Faster R-CNN with a third branch for predicting a pixel-wise binary mask in parallel to the two existing branches for computing the object class and the offset values of the bounding box [130]. The third branch is a FCN on top of the CNN feature map. To avoid misalignment due to quantization operations, a ROI alignment layer is used instead of the ROI pooling layer.

Single-stage detectors omit the region proposal stage and perform detection directly over a small sampling of possible locations. In general, one-stage detectors are simpler and faster compared to two-stage detectors. *You Only Look Once* (YOLO) considers object detection as a regression task and employs a single CNN that takes the entire image as input and is trained end-to-end [12]. YOLO divides the input image into a grid of cells with equal size. For each grid cell, class probabilities as well as bounding boxes with the corresponding confidence values are computed. Thus, the grid cell in which the center of an object is located is responsible for

its detection. Detections with a low confidence value are removed directly. Since some objects may be detected multiple times (e.g., large objects), NMS is applied to remove overlapping bounding boxes of the same class. YOLO has been improved in several follow-up works (e.g., [131, 132, 133]). The *Single-Shot MultiBox Detector* (SSD) [134] builds on VGG-16 [108] using a set of *auxiliary convolutional layers* instead of FC layers. Small objects are detected in the early layers of the network, and the deeper layers are responsible for box offsets as well as aspect ratios [125]. The bounding box regression technique of *SSD* is based on *MultiBox* introduced by Szegedy et al. [135]. Confidence scores, box offsets, and class probabilities are computed for a fixed set of bounding boxes. The final detections are obtained by applying NMS. During neural network training, hard negative mining and data augmentation are utilized. In *CenterNet*, each object is represented by one center point and a pair of corners (triplet of keypoints) [136]. A backbone FCN uses cascade corner pooling and center pooling to compute two corner heatmaps as well as one center point heatmap. Similar to *CornerNet* [137], candidate bounding boxes are determined by using corner pairs and similar embeddings. Then, the final bounding boxes are determined based on the center points. *EfficientDet* [138] is a family of scalable object detectors that utilizes EfficientNet [139] as the backbone network, *Bi-directional Feature Pyramid Network* (BiFPN) as the feature extractor, as well as a shared classification and bounding box prediction network. BiFPN enables simple and fast cross-scale connections of input features. A new compound scaling method is employed to simultaneously increase the resolution and all dimensions of the backbone, BiFPN, and classification/bounding box networks.

Commonly used backbone networks in object detectors include AlexNet [91], VGGNet [108], GoogleLeNet [109], ResNet [110], and EfficientNet [139]. *Swin Transformer* [140] builds on the work of Dosovitskiy et al. [141] and adapts transformers from natural language processing (NLP) to computer vision. Transformers are a special type of neural network architecture based on the *self-attention mechanism* that weights each part of the input data according to its importance. Swin Transformer generates hierarchical feature maps by starting with small image patches that are gradually merged in deeper layers. A shifted window scheme restricts self-attention to non-overlapping local windows, while also allowing cross-window connection. Thus, Swin Transformer is able to consider image information at various scales, with the computational complexity being linear to the input image size. In contrast to previous transformer-based architectures, these attributes enable Swin Transformer to serve as a general-purpose backbone network for computer vision tasks such as object detection. For detecting objects at vastly different scales, *Feature Pyramid Network* (FPN) [14] can be used as a generic feature extractor in detectors (e.g., Faster R-CNN). FPN comprises a top-down architecture with lateral connections to generate high-level semantic feature maps at various scales [14]. It comprises a bottom-up as well as top-down pathway. The bottom-up pathway computes a feature hierarchy at different scales, while the top-down pathway

up-samples spatially coarser (but semantically stronger) feature maps from higher levels to high-resolution features. To enhance these high-resolution features based on features from the bottom-up pathway, lateral connections are used.

2.2.2 Particle Detection in Fluorescence Microscopy Images

In previous work on *particle detection* in fluorescence microscopy images, classical methods have been introduced which often comprise three subsequent steps: noise reduction, signal enhancement of fluorescent particles, and signal thresholding. *Noise reduction* is performed to increase the signal-to-noise ratio (SNR) and improve the quality of the input image $\mathcal{I}$. A common method is *linear filtering*, where at each pixel the intensity is replaced by a linear combination of the intensities of its neighborhood. In most cases, a Gaussian filter kernel $G_\sigma^{2\mathrm{D}}$ with standard deviation σ is used, which is defined in 2D as follows:

$$G_\sigma^{2\mathrm{D}}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{2.42}$$

For a noisy 2D input image $\mathcal{I}$, the filtered image $\mathcal{J}$ is obtained by the following convolution:

$$\mathcal{J}(x, y) = (G_\sigma^{2\mathrm{D}} * \mathcal{I})(x, y) \tag{2.43a}$$

$$\mathcal{J}(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} G_\sigma^{2\mathrm{D}}(i, j)\mathcal{I}(x - i, y - j) \tag{2.43b}$$

where the kernel size is defined as $N = 2n + 1$. Since the intensity profile of fluorescent particles can usually be approximated by a Gaussian function, this filter corresponds to a *matched filter* that maximizes the SNR of images with additive white Gaussian noise [42, 55, 142]. The threshold step can already be applied to the filtered image $\mathcal{J}$ to obtain particle detections. In practice, however, this typically leads to poor results due to inhomogeneous background and strongly varying particle intensities. Further methods for noise reduction include non-linear filters (e.g., median filter), wavelet transform-based denoising (e.g., [143, 144]), variance-stabilizing schemes (e.g., [145]), and patch-based techniques (e.g., [55, 146]). *Signal enhancement* is used to enhance the intensity of regions belonging to particles while suppressing the signal of background structures. Some of the methods also include noise reduction, which eliminates the need to perform this step separately. The optimal filter (whitened matched filter) for reducing noise and enhancing Gaussian-like particles in fluorescence microscopy images can be well approximated by the *Laplacian of Gaussian* (LoG), also referred to as the Mexican hat filter [147]. For a

2D image, the $\mathrm{LoG}_{\sigma}^{\mathrm{2D}}$ filter kernel is defined as follows:

$$\mathrm{LoG}_{\sigma}^{\mathrm{2D}}(x,y) = \frac{x^2+y^2-2\sigma}{\sigma^4} G_{\sigma}^{\mathrm{2D}}(x,y) \tag{2.44}$$

where the standard deviation parameter σ must be adapted according to the size of the particles. Due to the associative property of convolution, applying the LoG filter is equivalent to using a Gaussian filter followed by a Laplacian filter. In the case of fluorescent particles, the LoG filter is also known as the *spot-enhancing filter* (SEF). For strongly varying particle sizes, SEF with different σ values can be applied. In [148], a method for automatically selecting the optimal standard deviation parameter σ is presented. Additional signal enhancing methods are based on wavelet decomposition (e.g., [143, 145]), image features (e.g., [149]), dynamic thresholding (e.g., top-hat filter [150]), and h-dome transformation [55]. The output map C of the signal enhancing step is converted into a binary mask based on thresholding or more complex methods such as Bayesian segmentation. Thresholding approaches assume that pixels in C with a signal greater than the threshold represent particles and all other pixels belong to the background. Since the result strongly depends on the selected threshold, it must be tuned to the data. Thus, techniques for adaptive thresholding and automatic determination of the optimal threshold have been introduced (e.g., [148, 151]). To assign a unique label to each set of adjacent foreground pixels (connected regions) representing individual particles, a *connected-components labeling* algorithm is applied to the binary mask. The label mask is used to determine the particle properties such as position and size. To localize particles with sub-pixel resolution, non-fitting and numerical fitting methods have been introduced [38]. *Non-fitting methods* are usually fast, effective, and widely applicable (e.g., centroid computation [152]). *Numerical fitting methods* are more specific and more commonly used. Here, a model function that depends on the appearance of the object is fitted to the image intensities using maximum likelihood estimation or least-square minimization. For fluorescent particles, the intensity profile corresponds to the PSF of the microscope, which can be well approximated by a Gaussian function. A major drawback of classical methods for particle detection in fluorescence microscopy images is that a predefined appearance model is required, which does not necessarily hold.

Most existing deep learning-based detection methods have been developed for objects (e.g., pedestrians, cars) in images of natural scenes (see Sec. 2.2.1). Deep learning also shows promising results for object detection in microscopy images (e.g., cells [15, 24]). However, fluorescent particle detection involves task-specific challenges (e.g., low SNR, clustering, missing appearance characteristics, small object size) that need to be addressed. In [153, 154, 155, 156], a CNN performs image-to-image mapping based on *pixel-wise binary classification*. The result is a binary mask in which fluorescent particles are represented by one or a few pixels. In [153, 154, 155], a sliding window scheme is employed. Dmitrieva et al. [157] use

a lightweight CNN to select candidate detections generated by a multi-scale SEF. Zakrzewski et al. [158] use a neural network architecture that is based on *RetinaNet* [93] to *directly regress* offsets of bounding boxes. In [159], feature extraction is performed by U-Net [15] and the input image is divided into grid cells by a second encoder. Similar to YOLO [12], the network computes the confidence and position of a particle for each grid cell.

2.3 Object Tracking

Object tracking is a computer vision task that aims to link individual object positions over time using the obtained detections. *Correspondence finding* addresses the problem whether two detections from different images belong to the same object or not. In addition, *track initiation and termination* must be considered to deal with a variable number of objects. The final tracking result represents the individual object trajectories. In this section, we describe deep learning methods for object tracking in general computer vision (Sec. 2.3.1), and then provide an overview of previous classical and deep learning methods for particle tracking in fluorescence microscopy images (Sec. 2.3.2). For more comprehensive reviews, we refer to [160, 161, 162].

2.3.1 Deep Learning Methods in Computer Vision

In recent years, numerous deep learning-based methods for object tracking in computer vision (e.g., cars, pedestrians) have been introduced yielding state-of-the art performance. Here, neural networks are used to perform one or more tasks (e.g., feature extraction, assignment score computation, motion prediction).

CNNs are commonly used to learn rich representations and extract meaningful visual features from image data, which are then incorporated into a classical tracking algorithm (e.g., multiple hypothesis tracking [163]). The *Deep Learning Tracker* (DLT) is based on a stacked *denoising autoencoder* (DAE) that learns robust generic image features [164]. The final network output acts as a confidence in a classical particle filtering framework. *Structured Outcome DLT* (SO-DLT) improves DLT by replacing the stacked DAE with a CNN that computes a pixel-wise probability map to distinguish between objects and background [165]. This allows a structured loss and yields computational scalability. Ma et al. [166] use rich feature hierarchies extracted from a CNN for *visual object tracking*. To exploit both important semantic features encoded in the last layer and more precise spatial information captured by earlier layers, linear correlation filters are applied to the feature maps of three different convolutional layers. To localize the object, a coarse-to-fine search is performed on the resulting correlation response maps. In [167], similar to Ma et al. [166], a switching mechanism is developed to consider two complementary convolutional layers from different levels. Thus, both semantic

and discriminative features are considered jointly. To reduce redundancy and improve performance by removing irrelevant and noisy feature maps, a feature map selection method is introduced. In [168], visual features of samples are extracted from a CNN and classified by an online trained SVM. To create a saliency map for each positive sample, object-relevant features of the sample are identified based on the SVM and subsequently back-propagated through the CNN. Bayesian filtering is employed to perform tracking by exploiting the combined saliency maps of the positive samples. Chi et al. [169] propose a dual network to fine-tune the features extracted by VGGNet for a specific object. In [170], the appearance embedding is integrated into a single-shot detector. To compute the similarity between objects and detections based on the appearance embeddings, cosine similarity is used.

Deep learning is commonly employed to compute *assignment scores*, typically by learning a similarity measure directly from the data. This eliminates the need of heuristic hand-crafted features. To establish one-to-one correspondences based on the computed assignment scores, a classical combinatorial optimization algorithm (e.g., Hungarian algorithm [171]) can be used. The *Siamese INstance Search Tracker* (SINT) builds on a two-stream Siamese deep neural network that learns a similarity measure to determine the candidate patch that best matches the initial object patch [172]. Candidate patches are generated based on the radius sampling strategy [173]. *Quadruplet Convolutional Neural Networks* (Quad-CNN) combine appearance embedding with motion-aware position embedding to learn a similarity measure that is used for correspondence finding via minimax label propagation [174]. In addition, bounding box regression is performed by Quad-CNN to fine-tune the initial object detections. Sadeghian et al. [19] propose a hierarchical RNN that exploits temporal dependencies in appearance, motion, and interaction of an object to compute the assignment score for a candidate detection. In [175], a Siamese LSTM exploits temporal and spatial features to compute a assignment score between two trajectories for correspondence finding. Ma et al. [176] use a bidirectional GRU to split tracklets of different objects into sub-tracklets. Then, all sub-tracklets belonging to the same object are re-connected based on the assignment scores computed by a Siamese GRU. A bilinear LSTM is employed to learn a measure for motion and appearance gating in a classical multiple hypothesis tracking (MHT) framework [177]. The CNN-based *Deep Affinity Network* (DAN) determines assignment scores between objects in different images using appearance features at different levels of abstraction [178].

Deep learning is also used to perform *regression tasks* for object tracking (e.g., motion prediction). The social LSTM introduced by Alahi et al. [179] predicts the future trajectories of all objects jointly based on their past dynamic behaviors. A *social pooling layer* is proposed to include inter-object dependencies. In [180], the *Behavior-CNN* learns to predict the dynamics of all objects based on a behavior encoding scheme that provides a general representation of the past object dynamics. Milan et al. [18] present an RNN-based neural network that mimics classical Bayesian

filtering. The network learns to predict the next object position and to update it based on an assigned detection. In addition, track initiation and termination are identified by computing existence probabilities. For correspondence finding, an LSTM-based RNN is used that receives as input the Euclidean distance between the predicted position and the detections. *Recurrent YOLO* (ROLO) combines YOLO with LSTM for single object tracking [181]. ROLO exploits previous positions and high-level visual features of an object to directly regress bounding box coordinates or heatmaps. *Tracktor++* predicts the next object position by using bounding box regression of the detector [182]. In Faster R-CNN, this is achieved by applying the ROI pooling operation to the features of the current image but with the previous bounding box. However, it is assumed that the bounding boxes of an object overlap in successive images. To generate appearance feature vectors for object re-identification, a Siamese neural network is utilized.

2.3.2 Particle Tracking in Fluorescence Microscopy Images

In previous work on particle tracking in fluorescence microscopy images, both classical methods as well as deep learning-based methods have been introduced. Classical methods can be subdivided into deterministic and probabilistic approaches [58].

Deterministic methods perform tracking by establishing correspondences between particle detections in images from different time points. Since fluorescent particles usually cannot be distinguished by their appearance, only the motion behavior of the objects is considered. Assumptions about the underlying *motion model* are translated into a cost function that defines the degree of correspondence between two detections obtained at different time points. For Brownian particles, the *nearest neighbor* (NN) model is commonly used, which is based only on position information [162]. The *assignment cost* $c_t^{i,j}$ between detection $\mathbf{y}_{t-1}^i$ at the previous time point $t-1$ and detection $\mathbf{y}_t^j$ at the current time point t is computed as follows:

$$c_t^{i,j} = \|\mathbf{y}_t^j - \mathbf{y}_{t-1}^i\| \tag{2.45}$$

The lower the cost $c_t^{i,j}$, the higher the degree of correspondence between $\mathbf{y}_{t-1}^i$ and $\mathbf{y}_t^j$. A simple strategy based on the NN model is the *nearest neighbor search* (NNS) (e.g., [183, 184]), where a correspondence is established between the detections $\mathbf{y}_{t-1}^i$ and $\mathbf{y}_t^j$ by determining the minimum cost $c_t^{i,j}$. To consider only likely assignments, *gating* can be performed where a threshold defines the maximum cost (or distance) between two detections at different time points. However, since NNS considers the correspondences of all detections at time point $t-1$ independently, *conflicting correspondences* may not be resolved properly. To address this issue, establishing one-to-one correspondences between two sets of detections from two consecutive

time points $t-1$ and t can be formulated as a *combinatorial optimization problem*:

$$\min \sum_{i=0}^{N_{t-1}} \sum_{j=0}^{N_t} c_t^{i,j} a_t^{i,j} \tag{2.46a}$$

$$\sum_{j=0}^{N_t} a_t^{i,j} = 1, \quad 1 \leq i \leq N_{t-1} \tag{2.46b}$$

$$\sum_{i=0}^{N_{t-1}} a_t^{i,j} = 1, \quad 1 \leq j \leq N_t \tag{2.46c}$$

where N_{t-1} and N_t are the number of particle detections at time point $t-1$ and t, respectively. The binary assignment variable $a_t^{i,j}$ is equal to 1 if $\mathbf{y}_{t-1}^i$ is assigned to $\mathbf{y}_t^j$, or 0 if not. Assignments of $\mathbf{y}_{t-1}^i$ and $\mathbf{y}_t^j$ to dummy detections representing missing detections are denoted by $a_t^{i,0}$ and $a_t^{0,j}$. The assignment costs $c_t^{i,0}$ and $c_t^{0,j}$ are usually equal to the gating threshold. *Combinatorial optimization approaches* such as the Hungarian algorithm [171], Munkres algorithm [185]. and Jonker–Volgenant shortest augmenting path algorithm [186] are used to solve Eq. (2.46). Since correspondences of all detections from the same time point are considered simultaneously and the NN model is used as cost function, this technique is called *global nearest neighbor* (GNN) method (e.g., [187, 188, 189]). For particles that perform directed motion, the *nearly constant velocity* (NCV) model is commonly used as cost function (e.g., [144]). In NCV, it is assumed that the velocity vector (speed and direction) of a particle changes only slightly between successive time points [162]:

$$c_t^{i,j} = \|(\mathbf{y}_t^j - \mathbf{y}_{t-1}^i) - (\mathbf{y}_{t-1}^i - \mathbf{y}_{t-2}^{i'})\| \tag{2.47}$$

where $\mathbf{y}_{t-2}^{i'}$ denotes the detection at time point $t-2$ assigned to $\mathbf{y}_{t-1}^i$. The more constant the velocity of a particle, the smaller becomes $c_t^{i,j}$. To improve the tracking performance under challenging conditions (e.g., low SNR, high object density), the cost function can also be minimized over more than two time points (e.g., [147, 190]). In this case, *Lagrangian relaxation techniques* are typically applied to solve the NP-hard combinatorial optimization problem [162]. Deterministic methods are computationally efficient but do not consider uncertainties, which often reduces the performance under challenging conditions (e.g., low SNR, high object density).

Probabilistic methods are formulated within a *Bayesian framework* and incorporate uncertainties by defining a posterior distribution on the variables (e.g., position, velocity) that describe the particle state based on a set of detections and assumptions about the dynamic behavior. The spatial-temporal posterior can be resolved recursively over time by using a sequential *Bayesian filter* such as the Kalman filter [191] or particle filter. Bayesian filtering is described in more detail in Sec. 5.2.2. The *Kalman filter* is computationally efficient and was developed for linear systems

with additive Gaussian noise. The posterior corresponds to a Gaussian distribution whose mean and covariance are updated recursively. Methods based on the Kalman filter perform particle detection and state estimation independently (e.g., [151, 192, 193, 194, 195]). For each particle, one filter is utilized to which detections are assigned by using a correspondence finding approach. The *particle filter* (e.g., [58, 149, 196, 197]) performs sequential importance sampling with additional resampling to approximate the posterior distribution by a set of weighted random samples ('particles') [198]. Thus, in contrast to the Kalman filter, it is a non-parametric approach that can cope with non-linear and non-Gaussian systems. In addition, particle detection and state estimation are combined by directly considering multiple image positions, which leads to more robust results. To ensure a good approximation of the posterior distribution, a large number of samples is required, resulting in high computational costs. Thus, for tracking multiple objects, the Kalman filter is usually preferred over the particle filter. Probabilistic methods that include uncertainties for the task of correspondence finding have also been introduced. *Joint probabilistic data association* (JPDA) [199] is a well-known method that estimates correspondences over two consecutive time points. Normalized probabilities are computed jointly for all possible assignments. Then, Bayesian filters are used to update the particle states based on all detections weighted according to the joint probabilities. However, JPDA assumes that the number of particles is known in advance and does not change over time. *Multiple hypothesis tracking* (MHT) [200] is well suited to address numerous challenges such as particle appearance and disappearance, detection errors, and conflicting correspondences. MHT maintains multiple competing track hypotheses so that later time points can be exploited to resolve ambiguities. Finally, the most likely combination of non-competing tracks is selected. However, computational and memory costs increase rapidly with the number of particles and time points. Thus, approximations and adaptations of JPDA (e.g., [151, 201, 202]) and MHT (e.g., [193, 203, 204, 205]) have been developed for particle tracking in fluorescence microscopy images. In general, major drawbacks of classical methods for fluorescent particle tracking are the required manual tuning of (numerous) parameters and selection of a suitable motion model. Moreover, assumptions about the probability distributions are needed, which do not necessarily hold.

Most existing deep learning-based tracking methods have been developed for objects in images of natural scenes (e.g., pedestrians, cars, see Sec. 2.3.1). Deep learning also shows promising results for object tracking in microscopy images (e.g., cells [21, 206, 207, 208]). However, fluorescent particles differ greatly from natural objects and cells (e.g., size, shape, dynamic behavior). In addition, the appearance of (almost) indistinguishable Gaussian-like particles does not provide reliable information for correspondence finding, and low SNR values lead to numerous detection errors. Thus, specialized deep learning methods are required for particle tracking in fluorescence microscopy images. In [209], a bi-

directional convolutional LSTM exploits a temporal sequence of image patches to approximate the posterior position density of a particle. For correspondence finding, a classical nearest neighbor search is performed instead of deep Learning. Yao et al. [210] presented an LSTM-based RNN that exploits past hand-crafted motion features of a particle to compute the assignment score for a candidate detection. In addition, the network predicts the particle position at the next time point to cope with a missing detection. The network is trained in a multi-task fashion by minimizing the weighted sum of the cross-entropy loss (assignment score) and the Huber loss [95] (predicted position). In the follow-up work [22], hand-crafted and learned features are combined to compute the assignment score for a candidate sequence of detections at successive time points. Smal et al. [211] use a denoising autoencoder and score matching within a classical MHT-based approach to learn the underlying motion model directly from the data and compute the assignment score for a candidate detection. To cope with missing detections, the denoising autoencoder estimates the next particle position based on the learned motion model. In [22, 210, 211], the assignment scores of a particle are computed independently for each candidate detection or sequence.

Chapter 3

Performance Metrics and State-of-the-Art Methods

In this chapter, we describe the metrics for quantitative performance evaluation of the developed particle detection and tracking methods. In addition, state-of-the-art methods used for performance comparison are presented.

3.1 Performance Metrics

To quantitatively evaluate the *tracking performance*, we used the five metrics α, β, JSC, JSC_θ, and $RMSE$ introduced by Chenouard et al. [65]. First, pairing is performed by establishing one-to-one correspondences between a set $\mathbf{X}$ of $N_{\text{estimated}}$ estimated tracks and a set $\mathbf{Y}$ of N_{true} true tracks (ground truth). To deal with missing estimated tracks, $\mathbf{X}$ is extended with a set $\varnothing$ of N_{true} dummy tracks resulting in $\mathbf{Z}$. A dummy track consists only of dummy detections, which are also used for missing points in an estimated track. The degree of correspondence between a true track $Y_j \in \mathbf{Y}$ and an estimated or dummy track $Z_i \in \mathbf{Z}$ is represented by the pairing distance $d(Z_i, Y_j)$, which is equal to the gated Euclidean distance over all time points t:

$$d(Z_i, Y_j) = \sum_{t=1}^{T} \|\mathbf{x}_t^{X_i} - \mathbf{x}_t^{Y_j}\|_{2,\epsilon} \tag{3.1}$$

where T is the total number of time points and $\mathbf{x}_t$ represents the position of a track at time point t. Two tracks are non-matching at time t if their Euclidean distance is larger than the gating parameter $\epsilon \in \mathbb{R}_+$, which is selected according to the Rayleigh criterion and also acts as a fixed penalty for unpaired track points. The total pairing distance $d(\mathbf{X}, \mathbf{Y})$ between $\mathbf{X}$ and $\mathbf{Y}$ is computed as:

$$d(\mathbf{X}, \mathbf{Y}) = \sum_{j=1}^{N_{\text{true}}} d(Z_i, Y_j) \tag{3.2}$$

where Z_i represents an estimated or dummy track paired with the true track Y_j. To determine the globally optimal pairing, $d(\mathbf{X}, \mathbf{Y})$ is minimized using the Munkres algorithm [185]. In the absence of a matching estimated track, each true track is paired with a dummy track, resulting in a maximum total pairing distance of $d(\varnothing, \mathbf{Y})$. All five tracking metrics are computed based on the optimal pairing with a total pairing distance $d^*(\mathbf{X}, \mathbf{Y})$. The overall degree of matching between $\mathbf{X}$ and $\mathbf{Y}$ without considering the set of unpaired estimated tracks $\overline{\mathbf{X}}$ is represented by $\alpha \in [0, 1]$:

$$\alpha = 1 - \frac{d^*(\mathbf{X}, \mathbf{Y})}{d(\varnothing, \mathbf{Y})} \tag{3.3}$$

The metric $\beta \in [0, \alpha]$ additionally penalizes $\overline{\mathbf{X}}$ using the term $d(\varnothing, \overline{\mathbf{X}})$:

$$\beta = \frac{d(\varnothing, \mathbf{Y}) - d^*(\mathbf{X}, \mathbf{Y})}{d(\varnothing, \mathbf{Y}) + d(\varnothing, \overline{\mathbf{X}})} \tag{3.4}$$

$\alpha = \beta$ when there are no estimated tracks in $\overline{\mathbf{X}}$. The rate of correct track points in $\mathbf{X}$ is determined by the Jaccard similarity coefficient $JSC \in [0, 1]$:

$$JSC = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{3.5}$$

where TP (true positives) and FP (false positives) are the total number of paired and unpaired track points in $\mathbf{X}$, respectively. FN (false negatives) is the total number of dummy detections paired with a true track. The Jaccard similarity coefficient for entire tracks $JSC_\theta \in [0, 1]$ is computed as:

$$JSC_\theta = \frac{\mathrm{TP}_\theta}{\mathrm{TP}_\theta + \mathrm{FP}_\theta + \mathrm{FN}_\theta} \tag{3.6}$$

where TP_θ and FP_θ are the total number of paired and unpaired tracks in $\mathbf{X}$, respectively. FN_θ is the total number of dummy tracks paired with a true track. The root mean square error *RMSE* represents the overall localization accuracy of correctly estimated track points (TP in Eq. (3.5)). For all metrics except *RMSE*, better tracking performance is indicated by a higher value.

To quantitatively evaluate the *detection performance*, we used the metrics F1 score and *RMSE*. For each image, the globally optimal pairing between the set $\mathbf{X}$ of estimated detections and the set $\mathbf{Y}$ of true particle positions (ground truth) is determined by minimizing the total pairing distance, which is equal to the gated Euclidean distance over all pairs. The gate parameter is selected according to the Rayleigh criterion and combinatorial optimization is performed by using the Jonker-Volgenant shortest augmenting path algorithm [186]. Both detection metrics are computed based on the optimal pairing. The F1 score $\in [0, 1]$ represents the

harmonic mean of precision and recall:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.7a}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.7b}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.7c}$$

where TP and FP are the total number of paired and unpaired detections in $\mathbf{X}$, respectively. FN is the total number of unpaired true positions in $\mathbf{Y}$. A higher F1 score indicates a better detection performance. The overall localization accuracy of the correct particle detections is determined by the *RMSE*.

3.2 State-of-the-Art Methods

We evaluated our proposed *tracking approaches* using data from the Particle Tracking Challenge (PTC, [65]) and compared the performance with the overall top-three methods of the challenge (Methods 5, 1, and 2).

- **Method 5 [151]:** The spot-enhancing filter (SEF, [147]) is used for particle detection and the Kalman filter [191] is employed for spatio-temporal filtering. Correspondences are established based on probabilistic data association.
- **Method 1 [187]:** For particle detection, local maxima selection and iterative intensity-weighted centroid computation is performed. Combinatorial optimization based on greedy hill climbing with nearest neighbor initialization and topological constraints is used for correspondence finding.
- **Method 2 [204]:** Particle detection involves a convolution step with a disk- or spherical-shaped filter, followed by adaptive local-maxima selection. Correspondence finding is based on multiple multiple hypothesis tracking (MHT).

We also assessed our tracking methods based on real fluorescence microscopy images displaying different virus structures and sub-cellular structures. We compared the performance with three state-of-the-art methods for fluorescent particle tracking that are widely used and publicly available.

- **ParticleTracker (PT, [187]):** PT uses iterative intensity-weighted centroid for particle detection and combinatorial optimization via greedy hill climbing for correspondence finding. This method is implemented in the ImageJ plugin MosaicSuite and corresponds to Method 1 of the PTC.
- **Kalman filter based approach (KF, [189]):** KF employs SEF for particle detection and the Kalman filter for motion prediction. Correspondence

finding is considered as a linear assignment problem, which is very similar to u-track [190] and solved using the Munkres algorithm [185]. KF is implemented in the ImageJ plugin TrackMate.

- **Multiple hypothesis tracking (MHT, [193]):** A wavelet-based method is applied for particle detection [143]. MHT, which uses Kalman filters and multiple motion models, is implemented in Icy [212].

We evaluated our proposed methods for *particle detection* using image data from the PTC as well as real fluorescence microscopy images, and conducted a quantitative performance comparison with three state-of-the-art methods.

- **Spot-enhancing filter (SEF, [147]):** The Laplacian of Gaussian (LoG) filter is applied to enhance the signal of regions belonging to particles while reducing noise and removing background structures. Thresholding followed by connected component labeling is used to identify individual particles in the filtered image. For a more detailed description of SEF, see Sec. 2.2.2.

- **DetNet [156]:** An hourglass-shaped deep neural network performs image-to-image mapping based on pixel-wise binary classification. Individual particles are detected in the binary mask based on a connected component labeling algorithm. In the 3D version of DetNet (DetNet3D), all 2D convolutions have been replaced by 3D convolutions.

Chapter 4

Recurrent Neural Networks for Particle Tracking

In this chapter, we introduce deep learning methods for particle tracking in fluorescence microscopy images that exploit temporal information by using recurrent neural networks. We first describe a method that considers past information, and then present an extension that additionally takes into account future information.

4.1 Recurrent Neural Network for Particle Tracking Using Past Information

In this section, we present a particle tracking method based on a recurrent neural network that exploits past information about object dynamics for state prediction and correspondence finding. The work has been published in [59].

4.1.1 Introduction

Previous work on tracking biological particles can be subdivided into deterministic and probabilistic methods. Deterministic approaches follow a two step-paradigm comprising particle detection and correspondence finding (e.g., [147, 187]). Probabilistic approaches are formulated within a Bayesian framework and take into account uncertainties to improve the robustness. The solution is determined using Kalman filters or particle filters (e.g., [65, 151, 193, 194, 205]). A disadvantage of traditional tracking methods is that a handcrafted similarity measure is used to determine the degree of correspondence between detections in successive images. In addition, a suitable dynamic model needs to be selected, and often tedious manual tuning of (numerous) parameters is required. Often, these approaches have difficulties in cluttered environments with clustering objects. Deep learning methods have the potential to improve the performance. This has been demonstrated for different tasks such as segmentation and classification in the fields of

computer vision and medical image analysis (e.g., [7]), however, much less work exists on tracking.

In the field of computer vision, Milan et al. [18] proposed a recurrent neural network (RNN) for tracking pedestrians in video images of natural scenes. However, tracking pedestrians is quite different from tracking biological particles since the motion and shape are very different, and appearance is not a reliable cue. Also, in [18] a handcrafted similarity measure is used for correspondence finding. In addition, two separate networks need to be trained for state prediction and data association. Sadeghian et al. [19] introduced an appearance-based RNN for tracking pedestrians in video images. However, there the similarity measure for correspondence finding is determined independently for each detection, and information on missing detections is not provided by the network. Also, a fixed input sequence length is used (last 6 time points). For training, manually labeled data was used. Yao et al. [210] used a similar approach as in [19] to track microtubules in synthetic data. However, the similarity measure for correspondence finding is not jointly computed across multiple detections, and a fixed input sequence length is used (as in [19]). In addition, objects are not automatically detected but ground truth positions are used, and real microscopy data was not considered. He et al. [206] introduced an approach based on convolutional neural networks (CNNs) for tracking of cells. However, this approach does not use an RNN, and tracking of particles was not considered.

In this contribution, we introduce a new approach for particle tracking in time-lapse fluorescence microscopy images based on an RNN. Both short- and long-term temporal dependencies of individual object dynamics are exploited for state prediction and correspondence finding using a Long Short-Term Memory (LSTM) [101]. The network automatically learns to determine assignment probabilities for correspondence finding, without requiring a handcrafted similarity measure. In contrast to [19, 210], our network computes assignment probabilities jointly across multiple detections, and also determines the probabilities of missing detections. In addition, the input sequence length is not limited but can be arbitrary long. Thus, we exploit more information and intrinsically cope with missing detections. Moreover our approach does not require manually labeled data (in contrast to [18, 19, 210]). Both state prediction and data association are trained within one network. Compared to traditional tracking methods, the dynamic model is automatically selected, and tuning of tracking parameters is not required. We performed a quantitative evaluation using data from the Particle Tracking Challenge as well as using real live cell microscopy data of human immunodeficiency virus type 1 (HIV-1) particles and hepatitis C virus (HCV) proteins. It turned out that our approach yields better tracking results than previous methods.

4.1.2 Method

Our approach, denoted as Deep Particle Tracker (DPT), relies on a tracking-by-detection paradigm. For spot detection, we use the spot-enhancing filter (SEF) [147] yielding a set of detections. For correspondence finding, we introduce an LSTM-based recurrent neural network that determines assignment probabilities between tracked objects and particle detections. To establish one-to-one correspondences using the computed assignment probabilities of all objects and the probabilities of missing detections, the Hungarian algorithm is employed.

Network Architecture

In our DPT approach, for each object we use one neural network with the same network architecture. We employ both LSTM and fully-connected (FC) layers each consisting of K units (we used $K = 250$). We apply Gaussian dropout after each layer. Below, we describe the network architecture in more detail.

Let the vector $\mathbf{x}_t^i \in \mathbb{R}^D$ denote the state of an object i at time point t. In our work, we used $\mathbf{x}_t^i = (x_t^i, y_t^i, s_t^i, \alpha_t^i)$, i.e. $D = 4$. (x_t^i, y_t^i) is the object position. The speed and direction of the object motion is denoted by s_t^i and α_t^i (computed using the positions at two successive time points). The detections (positions as well as speed and direction for an assignment to object i) are represented by the vector $\mathbf{y}_t^i \in \mathbb{R}^{M \cdot D}$, where M is the overall number of detections. Note that M is often very high (in cluttered environments) and varies strongly between different images of a sequence. On the other hand, the neural network requires a fixed input vector size. To address this, in our approach we exploit the M-nearest detections (we used $M = 5$). For each time point $t - 1$, the network computes two output vectors for the next time point t: $\hat{\mathbf{x}}_t^i \in \mathbb{R}^D$ is the predicted object state, and $\mathbf{a}_t^i \in [0, 1]^{M+1}$ represents the assignment probabilities between object i and the M-nearest detections as well as probabilities for missing detections.

We use an LSTM to predict the state of an object i for the next time point t. The LSTM is composed of layers interacting which each other to determine the new hidden state $\mathbf{h}_t^i \in \mathbb{R}^K$ of dimension K which also represents the output. The main component of an LSTM is the cell state $\mathbf{c}_t^i \in \mathbb{R}^K$ which serves as long-term memory [101]. At each time point t, different types of gates determine which information is added to or removed from the previous cell state $\mathbf{c}_{t-1}^i$. Note that all gates compute their output based on the previous hidden state $\mathbf{h}_{t-1}^i$ and the current input. In our case, the input is the object state $\mathbf{x}_{t-1}^i$ mapped to the vector $\mathbf{z}_t^i \in \mathbb{R}^K$ by using a fully-connected (FC) layer and a hyperbolic tangent activation function. At time point t, the LSTM for an object i is updated as follows:

$$\mathbf{i}_t^i = \sigma(\mathbf{W}_{zi}\mathbf{z}_t^i + \mathbf{W}_{hi}\mathbf{h}_{t-1}^i + \mathbf{b}_i) \tag{4.1}$$

$$\mathbf{f}_t^i = \sigma(\mathbf{W}_{zf}\mathbf{z}_t^i + \mathbf{W}_{hf}\mathbf{h}_{t-1}^i + \mathbf{b}_f) \tag{4.2}$$

$$\mathbf{o}_t^i = \sigma(\mathbf{W}_{zo}\mathbf{z}_t^i + \mathbf{W}_{ho}\mathbf{h}_{t-1}^i + \mathbf{b}_o) \tag{4.3}$$

$$\mathbf{g}_t^i = \tanh(\mathbf{W}_{zg}\mathbf{z}_t^i + \mathbf{W}_{hg}\mathbf{h}_{t-1}^i + \mathbf{b}_g) \tag{4.4}$$

$$\mathbf{c}_t^i = \mathbf{f}_t^i \odot \mathbf{c}_{t-1}^i + \mathbf{i}_t^i \odot \mathbf{g}_t^i \tag{4.5}$$

$$\mathbf{h}_t^i = \mathbf{o}_t^i \odot \tanh(\mathbf{c}_t^i) \tag{4.6}$$

where $\mathbf{i}_t^i$ is the input gate, $\mathbf{f}_t^i$ is the forget gate, $\mathbf{o}_t^i$ is the output gate, and $\mathbf{g}_t^i$ is the input modulation gate. Weight matrices $\mathbf{W} \in \mathbb{R}^{K\times K}$ and bias vectors $\mathbf{b} \in \mathbb{R}^K$ represent the parameters of a gate. σ is the logistic sigmoid activation function, and $\odot$ denotes element-wise (Hadamard) multiplication. We use the new hidden state $\mathbf{h}_t^i$ of the LSTM to compute the predicted object state $\hat{\mathbf{x}}_t^i$ by employing a FC layer and a hyperbolic tangent activation function. Since $\mathbf{h}_t^i$ is a function of all object states $\mathbf{x}_{1:t-1}^i$ from time point 1 to time point $t-1$, the network can exploit both short and long-term temporal dependencies for state prediction.

The vector $\mathbf{y}_t^i$ of the detections is passed to a FC layer with a hyperbolic tangent activation function for mapping it to a K-dimensional vector, which is then concatenated with the hidden state $\mathbf{h}_t^i$ of the LSTM. The resulting vector of dimension $2K$ is passed to another FC layer which maps it to a vector of dimension K. This vector is fed into a fully connected linear output layer with softmax normalization so that the final network output vector $\mathbf{a}_t^i$ can be interpreted as $M+1$ assignment probabilities, i.e. $\forall i : \sum_{j=1}^{M+1} a_t^{ij} = 1$, where a_t^{ij} denote the assignment probabilities between object i and detection j ($j = 1, \ldots, M$), and $a_t^{i(M+1)}$ are the probabilities of missing detections. The computed assignment probabilities and the probabilities for missing detections (dummy detections in the probability matrix) are used as input for the Hungarian algorithm. Note that a handcrafted similarity measure for the predicted state and the detections (e.g., Euclidean distance) is not required to compute the assignment probabilities.

The LSTM-based neural network is trained by minimizing the loss $\mathcal{L}$ over all trajectories defined by:

$$\mathcal{L} = \sum_{i=1}^{N} \mathcal{L}^i, \qquad \mathcal{L}^i = \sum_{t=1}^{T^i} \left(\frac{1}{D} \|\hat{\mathbf{x}}_t^i - \tilde{\mathbf{x}}_t^i\|^2 - \sum_{j=1}^{M+1} \tilde{a}_t^{ij} \log(a_t^{ij}) \right) \tag{4.7}$$

where N is the overall number of trajectories, $\mathcal{L}^i$ denotes the loss for the trajectory of object i, $\hat{\mathbf{x}}_t^i$ is the predicted state and $\tilde{\mathbf{x}}_t^i$ the true state at time point t. The deviation between the states is quantified by the mean squared error (MSE). The cross-entropy is used to measure the deviation between the computed assignment probabilities a_t^{ij} and the ground truth $\tilde{a}_t^{ij}$. T^i defines the total number of time points for a trajectory.

Training

Since deep learning architectures involve a large number of parameters, vast amounts of training data are generally required. However, ground truth for microscopy image sequences of biological particles is hardly available and manual annotation is very tedious. Therefore, in our approach we do not use manually labeled data but rely on synthetic data for training. We generated a large number of simulated trajectories of particles, which perform Brownian motion or directed motion. The diffusion coefficients and velocities of individual particles were sampled from a uniform distribution and the initial positions were chosen randomly. In addition, we randomly removed particle positions which enables the network to learn coping with missing detections.

For training our network, we used the RMSprop optimizer [84] with an initial learning rate of 3×10^{-5}, which was decreased by 5% when the validation loss stopped improving. To avoid overfitting, we employed early stopping and set the Gaussian dropout rate to $p = 0.2$. We used a dataset with 85,000 synthetically generated trajectories with variable track length. The dataset was split into 82% for training and 18% for validation. We used a mini-batch size of 10 trajectories.

4.1.3 Experimental Results

Particle Tracking Challenge Data

We evaluated our DPT approach based on data of the Particle Tracking Challenge [65] and compared the performance with the overall top-three approaches (Methods 5, 1, and 2) described in Sec. 3.2. In addition, we compared the performance of DPT with a recent approach employing a piecewise-stationary motion model smoother (PMMS) [194]. This approach uses SEF for particle localization and linear programming for linking (extension of u-track [190]).

To study the performance in cluttered environments, we used data of the vesicle scenario for signal-to-noise ratios of SNR = 4 and SNR = 7 as well as medium and high particle densities (medium: 500 particles/frame, high: 1000 particles/frame). The data is challenging due to conflicting correspondences (in total 15,682 trajectories). The image sequences consist of 100 images (512×512 pixels) with random appearance and disappearance of particles. To quantitatively assess the performance of the tracking methods, we computed the metrics α, β, JSC, JSC_θ, and $RMSE$ [65] described in Sec. 3.1.

The quantitative results are presented in Table 4.1 (bold values indicate the best performance). It can be seen that DPT performs best for all metrics and cases. Note that for PMMS the results in [194] are given only up to two decimal places and $RMSE$ is not provided. Note that for our DPT approach, we did not use the Particle Tracking Challenge data for training, but used our own generated synthetic data as described in Sec. 4.1.2 above.

Table 4.1 Tracking performance of different approaches for data of the vesicle scenario from the Particle Tracking Challenge. Bold indicates best performance.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	4	Method 5	0.658	0.588	0.641	0.776	0.754
		Method 1	0.687	0.609	0.652	0.767	0.607
		Method 2	0.582	0.514	0.59	0.757	0.97
		PMMS	0.67	0.60	0.64	0.77	-
		DPT	**0.695**	**0.624**	**0.658**	**0.790**	**0.545**
	7	Method 5	0.677	0.605	0.646	0.783	0.667
		Method 1	0.7	0.619	0.65	0.758	0.544
		Method 2	0.611	0.547	0.606	0.775	0.828
		PMMS	0.68	0.61	0.64	0.78	-
		DPT	**0.711**	**0.631**	**0.651**	**0.790**	**0.525**
High	4	Method 5	0.488	0.408	0.466	0.671	1.004
		Method 1	0.531	0.442	0.487	0.641	0.801
		Method 2	0.43	0.356	0.429	0.649	1.208
		PMMS	0.51	0.44	0.48	0.67	-
		DPT	**0.547**	**0.462**	**0.505**	**0.680**	**0.746**
	7	Method 5	0.533	0.453	0.503	0.698	0.931
		Method 1	0.582	0.494	0.526	0.683	0.683
		Method 2	0.466	0.395	0.458	0.665	1.027
		PMMS	0.55	0.48	0.51	0.69	-
		DPT	**0.590**	**0.507**	**0.535**	**0.702**	**0.677**

Real Fluorescence Microscopy Images of Virus Structures

We also evaluated the performance of the DPT approach using real fluorescence microscopy image sequences displaying human immunodeficiency virus type 1 (HIV-1) particles and hepatitis C virus (HCV) proteins. The fluorescence labeled HIV-1 particles were imaged by a confocal spinning disk microscope and an EM-CCD camera. For our evaluation we used two image sequences (each 50 time points, 1000×1000 pixels, 16-bit) denoted by Seq. A and Seq. B. We also used one image sequence showing the HCV nonstructural protein 5A (30 time points, 1000×1000 pixels, 16-bit) denoted by Seq. C (an example section with 115×115 pixels is shown in Fig. 4.1). The images were acquired by a confocal spinning disk microscope and a CMOS camera. This dataset is challenging due to relatively low SNRs and clutter (high particle density, often crossing of trajectories). Ground truth trajectories for regions with clutter and large motion were determined by manual annotation. Seq. A, Seq. B, and Seq. C comprise 117, 125, and 55 ground truth trajectories, respectively (with up to 30 time points).

We compared the performance of DPT with the ParticleTracker (PT) [187], a Kalman filter based approach (KF) [189], and multiple-hypothesis tracking (MHT) using multiple motion models [193] (see Sec. 3.2). For PT, KF, and MHT we performed a grid search to determine optimal parameter settings. Note that for DPT, adaption of tracking parameters was not necessary (except the two detection parameters for SEF), i.e. we directly applied our tracking approach to the real data while training was performed only on synthetic data (see Sec. 4.1.2 above). Table 4.2

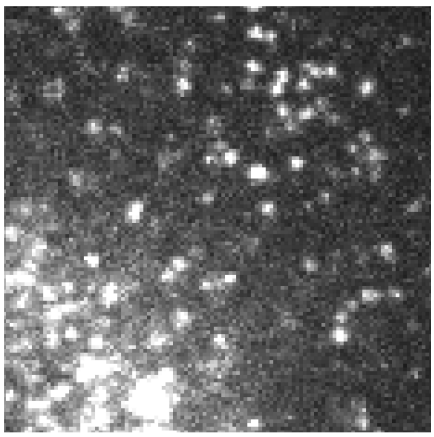


Figure 4.1 Section of image sequence Seq. C (HCV). The image contrast was enhanced.

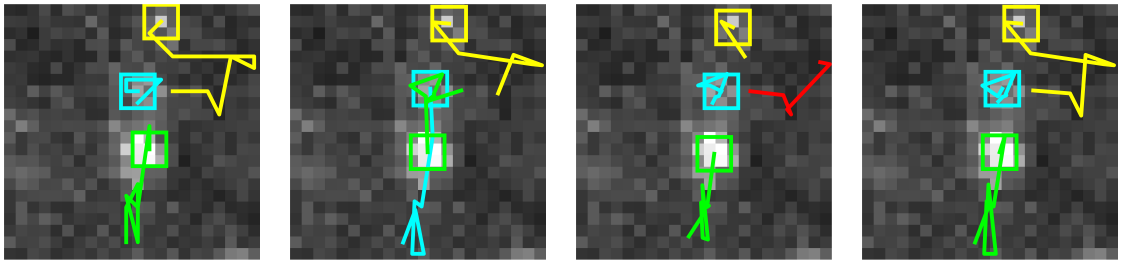
Table 4.2 Tracking performance of different approaches for real fluorescence microscopy images. Bold indicates best performance.

Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. A (HIV-1)	PT	0.312	0.255	0.348	0.442	2.701
	KF	0.388	0.317	0.421	0.456	2.775
	MHT	0.367	0.304	0.454	0.440	3.393
	DPT	**0.413**	**0.360**	**0.462**	**0.497**	**2.673**
Seq. B (HIV-1)	PT	0.328	0.261	0.338	0.399	2.559
	KF	0.352	0.312	0.396	0.373	**2.121**
	MHT	0.366	0.303	0.429	0.416	2.991
	DPT	**0.435**	**0.331**	**0.444**	**0.527**	2.717
Seq. C (HCV)	PT	0.590	0.496	0.629	0.557	1.064
	KF	0.559	0.481	0.564	0.550	1.088
	MHT	0.540	0.480	0.588	0.611	1.237
	DPT	**0.647**	**0.571**	**0.669**	**0.625**	**1.024**

shows the tracking performance for all three image sequences. It turns out that DPT outperforms the other methods for all metrics and sequences (except $RMSE$ for Seq. B). Sample results for Seq. C are shown in Fig. 4.2. It can be seen that DPT yields the best result and maintains the correct identity for all three particles. KF causes an identity switch (between the blue and green trajectory). MHT yields a broken trajectory (yellow).

4.1.4 Conclusion

We presented a novel approach for tracking particles in time-lapse microscopy images using an LSTM-based recurrent neural network which computes assignment probabilities jointly across multiple detections and also determines probabilities for missing detections. Manually labeled data is not required. In addition, a handcrafted similarity measure is not needed. We evaluated our approach based



(a) Ground truth **(b)** KF **(c)** MHT **(d)** DPT

Figure 4.2 Ground truth and results of different tracking approaches for image sequence Seq. C (HCV). The image contrast was enhanced for better visualization.

on synthetic and real image sequences. It turned out that our approach yields better results than previous methods.

4.2 Recurrent Neural Network for Particle Tracking Using Past and Future Information

In this section, we introduce a new particle tracking approach based on a recurrent neural network that exploits past and future information about object dynamics for correspondence finding. The work has been published in [60].

4.2.1 Introduction

In previous work on tracking of particles in fluorescence microscopy images, traditional deterministic and probabilistic methods have been introduced. Deterministic approaches are based on a two-step paradigm comprising particle detection and correspondence finding (e.g., [187, 190, 213, 214, 215, 216]). While being efficient, deterministic approaches do not incorporate spatial-temporal uncertainties, which reduces the performance in demanding tracking scenarios (e.g., low SNR, high particle density). Probabilistic approaches formulated within a Bayesian framework take into account spatial-temporal uncertainties by defining a posterior distribution on the variables describing the object state. The spatial-temporal posterior can be resolved via a Kalman filter (e.g., [144, 151, 192, 193, 194]) or a particle filter (e.g., [58, 149, 196, 197]). Probabilistic approaches for correspondence finding have also been proposed. These include joint probabilistic data association (JPDA) (e.g., [151, 202]) and multiple hypothesis tracking (MHT) (e.g., [193, 203, 204, 205]). Disadvantages of traditional tracking methods are that generally tedious manual tuning of (numerous) parameters is required and that the expected dynamic model must be selected. In addition, assumptions about the probability distributions need to be made (e.g, probability of object appearance, disappearance, and occlusion),

which do not necessarily hold. Often, traditional approaches have difficulties in cluttered environments and low SNR image data.

Deep learning methods have the potential to improve the performance. This has been demonstrated for different computer vision and medical image analysis tasks, including image classification, object detection, and object segmentation (e.g., [6, 7, 8]). In recent years, deep learning approaches have been introduced for tracking objects in video images of natural scenes (e.g., pedestrians, cars). Typically, convolutional neural networks (CNNs) are used for learning appearance models (e.g., [217, 218]). Other approaches include temporal information by recurrent neural networks (RNNs) (e.g., [18, 19, 177, 219]), which typically exploit appearance information and determine the similarity measure independently for each detection.

Compared to video images of natural scenes, much less work exists on deep learning methods for tracking biological objects in microscopy images. In [21, 206], CNN- and recurrent CNN-based approaches for cell tracking were introduced which exploit appearance features and do not use future information. However, cell tracking and tracking objects in natural scenes is quite different from tracking biological particles since the motion and shape are very different, and appearance can hardly be exploited. In [153, 154, 155, 156], CNNs were used for particle detection in fluorescence microscopy images. However, deep learning was not employed for correspondence finding. [209] describes an RNN to approximate the filter-backward-sample-forward algorithm for Bayesian filtering to track clathrin-coated pits. For correspondence finding, a traditional local nearest neighbor algorithm was used, but deep learning was not employed. In [210], an RNN-based approach was employed to track microtubules in synthetic data. For correspondence finding, a similarity measure based on handcrafted motion features is computed independently for each detection. In [211], a denoising autoencoder was used to approximate the motion model within a traditional MHT-based approach to track microtubules in real data. The assignment scores are computed independently for each possible track extension, and track initiation and termination are not handled by the network. In our previous work [59] (described in Sec. 4.1), we introduced a deep learning approach for tracking virus particles in real microscopy images. A Long Short-Term Memory (LSTM)-based RNN determines assignment probabilities jointly across multiple detections using handcrafted motion features. None of the previous methods for particle tracking in fluorescence microscopy images uses a deep neural network that exploits future information and multiple track hypotheses for correspondence finding as well as determines assignment probabilities jointly across multiple detections.

In this contribution, we introduce a new deep learning method for particle tracking in temporal fluorescence microscopy images. Our method is based on a deep RNN architecture that exploits past and future information in both forward and backward direction. Assignment probabilities are determined jointly across multiple detections, and the probability of missing detections is also computed. To

handle track initiation and termination, the network computes existence probabilities. In addition, multiple track hypotheses are exploited so that decisions about correspondences can be delayed until ambiguities are resolved. This is similar to the classical multiple hypothesis tracking algorithm [200], however, here we suggest a deep learning formulation. An advantage of our formulation is that it corresponds to a two-dimensional assignment problem including past and future information, which can be solved efficiently since it has only polynomial complexity. In contrast, the traditional MHT corresponds to a multi-dimensional assignment problem, which is NP-hard and the solution is typically approximated by relaxation methods (e.g., [203]). Our novel deep neural network includes fully-connected (FC) layers, temporal convolution layers, and stacked bidirectional LSTMs [220] that utilize short and long-term dependencies in both directions. The length of the temporal sequence of learned features in our network is not limited, but can be arbitrary long. Manually labeled data is not required for network training. The network is trained using only simulated data and then applied to all data sets in the experiments. A handcrafted similarity measure is not necessary for correspondence finding, and assignment and existence probabilities are computed within one network. Handcrafted motion features (e.g., velocity) are not required. Instead, our approach directly employs the position information. Compared to traditional tracking methods, the dynamic model does not need to be selected, and tuning of tracking parameters as well as prior assumptions about probability distributions (e.g., probability of object appearance, disappearance, and occlusion) are not necessary.

We introduce the first deep learning method for particle tracking in fluorescence microscopy that exploits future information and multiple track hypotheses for correspondence finding as well as determines assignment probabilities jointly across multiple detections. We quantitatively evaluated the performance of our approach using data from the Particle Tracking Challenge as well as using real live cell microscopy image sequences displaying fluorescently labeled human immunodeficiency type 1 (HIV-1) particles and hepatitis C virus (HCV) proteins. It turned out that our approach outperforms previous methods.

4.2.2 Method

In this section, we describe our deep learning approach for particle tracking in microscopy images, denoted as Deep Particle Hypotheses Tracker (DPHT), including the construction of the track tree, the network architecture, the loss function, and the training of the network.

Overview of the Tracking Approach

A schematic overview of the proposed DPHT approach is shown in Fig. 4.3. DPHT relies on a tracking-by-detection framework. For spot detection, we use the

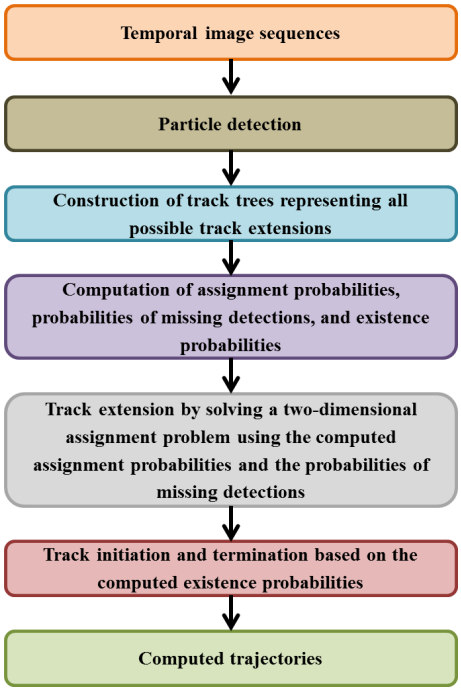


Figure 4.3 Schematic overview of the proposed DPHT approach.

spot-enhancing filter (SEF) [147] combined with Gaussian fitting yielding a set of detections. For correspondence finding at a certain time point, we introduce a deep RNN that determines assignment probabilities between tracked objects and particle detections by exploiting track hypotheses propagated into the future by track trees. The network architecture consists of FC layers, 1D max-pooling layers, temporal convolution layers, and stacked bidirectional LSTMs (BLSTMs) [220]. Information at past and future time points is used in forward and backward direction to improve correspondence finding at the current time point. The computed assignment probabilities of all objects and the probabilities of missing detections are used for a two-dimensional assignment problem, which can be solved efficiently by employing the Jonker-Volgenant shortest augmenting path algorithm [186] to establish one-to-one correspondences. To handle track initiation and termination, our DPHT network computes existence probabilities.

Track Tree Construction

In the following, we describe the construction of the track tree. The set of M_t particle detections at time point t is denoted by $\mathbf{Y}_t = \{\mathbf{y}_t^j\}_{j=1}^{M_t}$, where each detection $\mathbf{y}_t^j \in \mathbb{R}^D$ is represented by the image coordinates (x_t^j, y_t^j) for $D = 2$. The set of N_t existing tracks at time point t is denoted by $\mathbf{\Theta}_t = \{\boldsymbol{\theta}_t^i\}_{i=1}^{N_t}$, where each track $\boldsymbol{\theta}_t^i$ is a temporal sequence of assigned detections corresponding to the same object i. For an object, the track up to time point t is defined as $\boldsymbol{\theta}_t^i = \{\mathbf{y}_{t_{init}^i}^{i,j}, \ldots, \mathbf{y}_t^{i,j}\}$, where $t_{init}^i \leq t$ is the

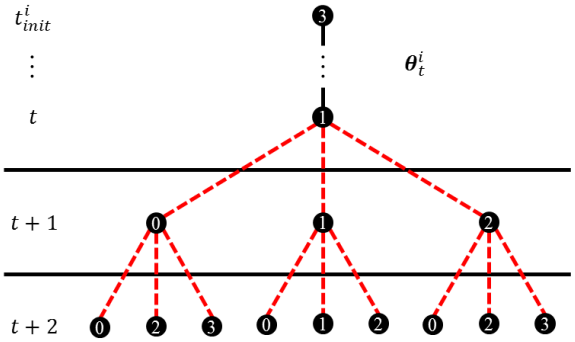


Figure 4.4 Example tree representing potential extensions of track $\boldsymbol{\theta}^i_t$ for two subsequent time points ($d = 2$). Numbers in the circles denote the index j of the detections at the corresponding time point. Each node represents a detection, which is either a (real) detection ($j > 0$) or a dummy detection ($j = 0$). Dashed red lines indicate hypothetical assignments and black lines denote confirmed assignments. For simplification, we used $M = 2$ (real) detections in this example.

time point of track initiation and $\mathbf{y}^{i,j}_t$ is the detection assigned at time point t. Note that the dummy detection $\mathbf{y}^{i,0}_t$ represents a missing detection at time point t (false negative, e.g., due to occlusion).

Let us assume that a set of confirmed tracks $\boldsymbol{\Theta}_t$ are available at time point t. For time point $t + 1$, the track $\boldsymbol{\theta}^i_t$ corresponding to object i can be assigned either with a (real) detection $\mathbf{y}^j_{t+1} \in \mathbf{Y}_{t+1}$ or with a dummy detection $\mathbf{y}^0_{t+1}$ representing a missing detection. All possible assignments of $\boldsymbol{\theta}^i_t$ at time point $t + 1$ form a set of track hypotheses $\boldsymbol{\Gamma}^i_{t+1} = \{\boldsymbol{\gamma}^{i,j}_{t+1}\}^{M_{t+1}}_{j=0}$, where $\boldsymbol{\gamma}^{i,j}_{t+1} = \{\boldsymbol{\theta}^i_t, \mathbf{y}^j_{t+1}\}$ denotes the track hypothesis formed by the hypothetical assignment of $\boldsymbol{\theta}^i_t$ with detection $\mathbf{y}^j_{t+1}$, which is either a real detection ($j > 0$) or a dummy detection ($j = 0$). We recursively apply this assignment process for every set $\{\boldsymbol{\Gamma}^i_{t+1}, \ldots, \boldsymbol{\Gamma}^i_{t+d}\}$, yielding the set of all possible track hypotheses $\boldsymbol{\Gamma}^i_{t+1:t+d}$ from time point $t + 1$ to $t + d$. Note that the number of detections M is often very high (in cluttered environments) and varies strongly between different images of a sequence. On the other hand, a neural network requires a fixed input vector size. To address this, in our approach each track hypothesis is assigned with the M-nearest detections (we used $M = 4$) at the next time point to form $M + 1$ new track hypotheses. Thus, the set of all track hypotheses for all detections j is given as $\boldsymbol{\Gamma}^i_{t+1:t+d} = \{\boldsymbol{\gamma}^i_{t+1:t+d,k}\}^{(M+1)^d}_{k=1}$ and the overall number of hypotheses over d time points is $(M + 1)^d$. As illustrated in Fig. 4.4, the potential extensions of existing tracks can be represented by trees of hypothetical assignments with detections from time point $t + 1$ to $t + d$.

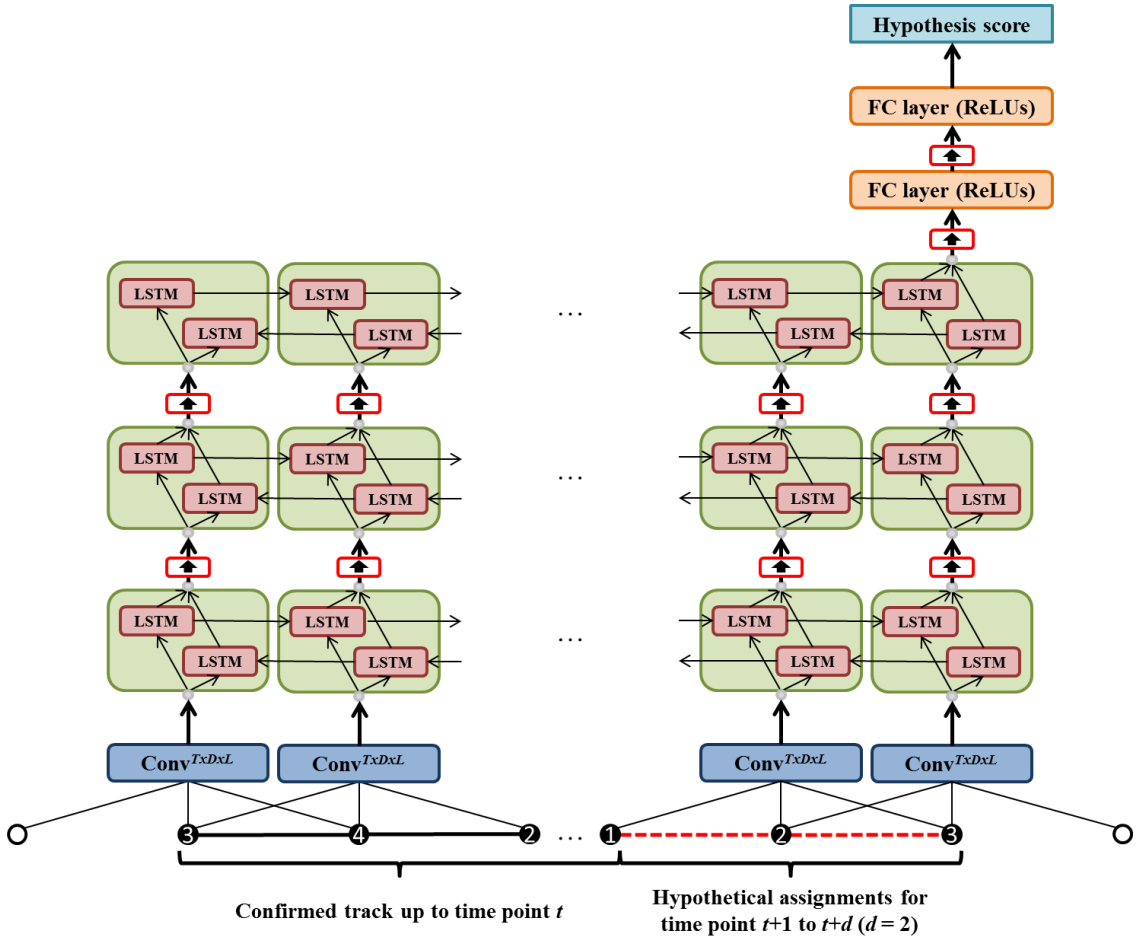


Figure 4.5 Architecture of the proposed subnetwork to compute the score of individual track hypotheses. Red boxes with bold arrows denote Gaussian dropout during network training. Dashed red lines indicate hypothetical assignments and black lines indicate confirmed assignments. Unfilled black circles denote constant padding before temporal convolution.

Network Architecture

In our DPHT approach, for each object i we use one neural network with the same network architecture. For hypothesis scoring, the network includes $(M+1)^d$ identical subnetworks, one for each track hypothesis $\gamma^i_{t+1:t+d,k} \in \mathbf{\Gamma}^i_{t+1:t+d}$. The network input is a tensor of size $(M+1)^d \times \Delta t \times D$ representing all track hypotheses in the track tree. $\Delta t = t + d + 1 - t^i_{init}$ denotes the temporal length of the track hypotheses. First, the input tensor is sliced into $(M+1)^d$ matrices of dimension $\Delta t \times D$, each of which is used as input for a different subnetwork. In the following, we describe the network and subnetwork architecture in more detail.

The subnetwork architecture is sketched in Fig. 4.5 and the layer configuration is provided in Table 4.3. To avoid defining handcrafted motion features (e.g., velocity), we directly use the position information (image coordinates of particles)

Table 4.3 Subnetwork layer configuration.

Layer type	Output size
Input	$\Delta t \times D$
Padding	$(\Delta t + 2) \times D$
Temporal convolution	$\Delta t \times L$
BLSTM	$\Delta t \times 2K$
BLSTM	$\Delta t \times 2K$
BLSTM	$2K$
FC layer	K
FC layer	1

and perform a temporal convolution with L filters (we used $L = 12$) employing a convolution window size of T time points ($Conv^{T \times D \times L}$, we used $T = 3$) on the track hypothesis padded with a constant (we used "-1" for constant padding). This results in a temporal sequence of learned dynamic features represented by a tensor of size $\Delta t \times L$. Note that when using only position information without temporal convolution the training of the network was not successful. To exploit both short and long-term dependencies in forward and backward direction in the sequence of learned dynamic features, we use n_l stacked bidirectional Long Short-Term Memory (BLSTM) [220] layers. Bidirectional RNNs were previously used for object tracking (e.g., [176, 209, 221]), but have not yet been introduced for determining assignment probabilities jointly across multiple detections or exploiting multiple track hypotheses. Each BLSTM layer consists of a forward Long Short-Term Memory (LSTM) [101] sublayer and a backward LSTM sublayer. Both LSTM sublayers consist of K blocks (we used $K = 64$), known as memory blocks. Each memory block contains in our case one recurrently connected memory cell whose activation is called cell state and three multiplicative gates. The gates, called input, output, and forget gate, provide functions analogous to write, read, and reset operations that control the information to and from the cell state. More precisely, the input to the cell state is multiplied by the activation of the input gate, the output is multiplied by the activation of the output gate, and the previous cell state is multiplied by the activation of the forget gate. Thus, the network can only interact with the memory cell via the gates. The forward LSTM sublayer processes the temporal input sequence (i.e., the learned dynamic features or the output of the previous BLSTM layer) from time point t^i_{init} of track initiation for an object i to $t + d$, while the backward sublayer processes the temporal input sequence from time point $t + d$ to t^i_{init}. The output of the forward and backward sublayer are combined by a concatenation operation to form the BLSTM layer output. Note that the length of the input sequence in our network is not limited as in [19, 209, 210, 219], but can be arbitrary long. The $2K$-dimensional output vector of the last BLSTM layer is passed to a fully-connected (FC) layer with K rectified

Table 4.4 Network layer configuration. For simplification, slicing and concatenation operations are not shown.

Layer type		Output size	
Input		$(M+1)^d \times \Delta t \times D$	
Subnetworks		$(M+1)^d$	
1D max-pooling		$M+1$	
FC layer	FC layer	K	K
FC layer	FC layer	K	K
Softmax	Softmax	$M+1$	2

linear units (ReLUs) mapping it to a K-dimensional vector, that is fed into a single ReLU to compute the score for the hypothesis $\gamma^i_{t+1:t+d,k}$.

The overall architecture of our DPHT network is sketched in Fig. 4.6, and the layer configuration is provided in Table 4.4. The computed scores of all hypotheses in the track tree are concatenated to form an $(M+1)^d$-dimensional vector. To select the highest score of all track hypotheses containing the same detection at time point $t+1$, we perform 1D max-pooling over a window size of $(M+1)^{d-1}$. The resulting vector of dimension $M+1$ is fed into two separate output paths, each comprising two consecutive FC layers with K ReLUs followed by a fully connected linear output layer with softmax normalization. The network output vector $\boldsymbol{a}^i_{t+1} \in [0,1]^{M+1}$ computed by the first output path can be interpreted as $M+1$ assignment probabilities for time point $t+1$, i.e. $\sum_{j=0}^{M} \boldsymbol{a}^{i,j}_{t+1} = 1$, where $\boldsymbol{a}^{i,j}_{t+1}$ denote the assignment probabilities between object i and detection j ($j = 1, \ldots, M$), and $\boldsymbol{a}^{i,0}_{t+1}$ are the probabilities of missing detections. To establish one-to-one correspondences at time point $t+1$, the computed assignment probabilities and the probabilities for missing detections (dummy detections) are used as input for the Jonker-Volgenant shortest augmenting path algorithm [186]. Note that our approach corresponds to a two-dimensional assignment problem, which can be solved efficiently since it has only polynomial complexity. This is an advantage compared to the traditional MHT which corresponds to a multidimensional assignment problem, which is NP-hard and where the solution is typically approximated by relaxation methods (e.g., [203]). The network output vector $\boldsymbol{\varepsilon}^i_{t+1} \in [0,1]^2$ computed by the second output path can be interpreted as normalized existence probabilities. The first element $\varepsilon^{i,alive}_{t+1}$ indicates the probability that object i exists at time point $t+1$, whereas the second element $\varepsilon^{i,dead}_{t+1}$ represents the probability that object i does no longer exist. Note that since we used a 1D max-pooling layer the number of learnable parameters of the network is independent of d.

Track Tree Maintenance

After having established one-to-one correspondences between confirmed tracks $\boldsymbol{\Theta}_t$ and detections $\mathbf{Y}_{t+1}$ for time point $t+1$, we prune the trees to preserve their

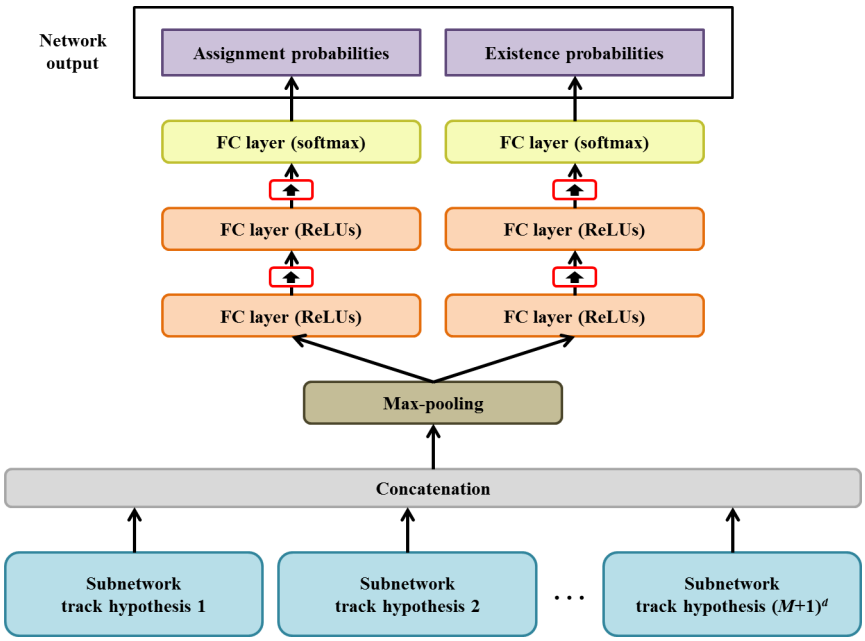


Figure 4.6 Network architecture of the proposed DPHT approach. Red boxes with bold arrows denote Gaussian dropout during network training.

compatibility with the updated tracks $\boldsymbol{\Theta}_{t+1}$. The principle is sketched in Fig. 4.7 for the track tree in Fig. 4.4. Track initiation and termination is handled using the computed existence probabilities. For each detection $\mathbf{y}_{t+1}^j \in \mathbf{Y}_{t+1}$ that is not assigned to a track $\boldsymbol{\theta}_{t+1}^i \in \boldsymbol{\Theta}_{t+1}$, a new tree with is constructed with $\mathbf{y}_{t+1}^j$ as root node. If $\varepsilon_{t+1}^{alive}$ is larger than ε_{t+1}^{dead} for the tree with root node $\mathbf{y}_{t+1}^j$, then $\mathbf{y}_{t+1}^j$ is considered to belong to a newly appearing object and a new track is initiated. Otherwise $\mathbf{y}_{t+1}^j$ is marked as false positive detection and the newly constructed tree is discarded. The track of an object i is terminated when $\varepsilon_{t+1}^{i,dead}$ is larger than $\varepsilon_{t+1}^{i,alive}$.

Loss Function

Our network is trained by multi-task learning using the loss function

$$\mathcal{L} = \mathcal{L}_a(\boldsymbol{a}, \tilde{\boldsymbol{a}}) + \lambda \mathcal{L}_\varepsilon(\boldsymbol{\varepsilon}, \tilde{\boldsymbol{\varepsilon}}) \tag{4.8}$$

where the first term is the cross-entropy loss used to measure the deviation between the computed assignment probabilities $\boldsymbol{a}$ and the ground truth $\tilde{\boldsymbol{a}}$:

$$\mathcal{L}_a(\boldsymbol{a}, \tilde{\boldsymbol{a}}) = -\sum_{j=0}^{M} \tilde{\boldsymbol{a}}^j \log(\boldsymbol{a}^j) \tag{4.9}$$

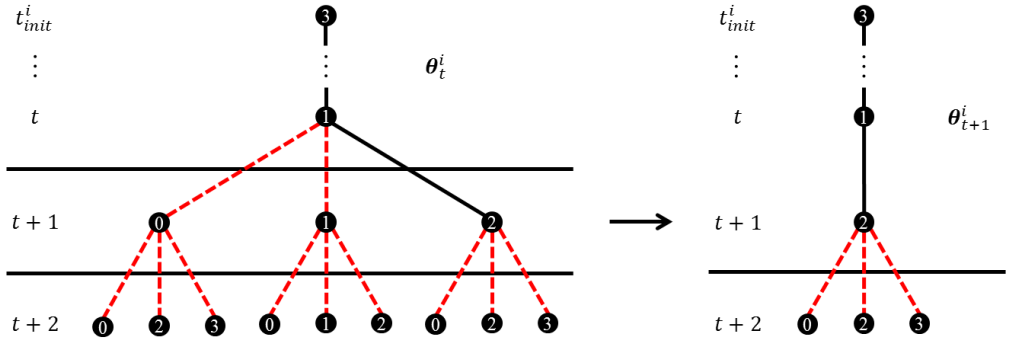


Figure 4.7 Principle of the tree maintenance strategy illustrated for a track $\boldsymbol{\theta}^i_t$ for two subsequent time points ($d = 2$). Numbers in the circles represent the index j of the detections at the corresponding time point. Dashed red lines denote hypothetical assignments and black lines denote confirmed assignments. All branches on the left that are in conflict with the newly established assignment between object i and detection $\mathbf{y}^2_{t+1}$ are removed yielding the pruned track tree on the right. For simplification, we used $M = 2$ (real) detections in this example.

The second term is the cross-entropy loss for the predicted existence probabilities ε^{alive} and ε^{dead} using the ground truth $\tilde{\varepsilon}^{alive}$ and $\tilde{\varepsilon}^{dead}$:

$$\mathcal{L}_\varepsilon(\boldsymbol{\varepsilon}, \tilde{\boldsymbol{\varepsilon}}) = -(\tilde{\varepsilon}^{alive} \log(\varepsilon^{alive}) + \tilde{\varepsilon}^{dead} \log(\varepsilon^{dead})) \tag{4.10}$$

In all our experiments we used $\lambda = 1$ in Eq. (4.8).

Network Training

Deep learning architectures generally require a vast amount of training data to achieve convergence without overfitting while still generalizing well to new data. However, ground truth for microscopy image sequences of fluorescent particles is hardly available and manual annotation is very time-consuming. Therefore, in our approach we do not use manually labeled data but rely on simulated data for training. We simulated trajectories and generated different image sequences. For the dynamics of particles we used four different motion models, namely Brownian (random-walk) motion, directed motion at constant velocity, accelerated motion, and random switching between Brownian motion and directed motion. The parameters for the motion of individual particles and the particle density were sampled from uniform distributions. The initial particle positions were chosen randomly. The particles appear and disappear randomly, and we also simulated movement out of the focus and the field of view. The SNRs of the image sequences were also sampled from a uniform distribution. We used SNR $= (I_0 - I_{bg})/\sqrt{I_o}$ as in [187], where I_0 denotes the maximum particle intensity and I_b is the mean background intensity. To enable our network to learn assignment and existence probability computation

under consideration of detector errors, we generated training samples based on real detections. Given the synthetically generated image sequences, particles were automatically detected using the spot-enhancing filter [147]. Then, the detections were mapped to ground truth trajectories by nearest neighbor search (using a validation gate of 5 pixels). The resulting trajectories consisting of real detections were used for network training. Thus, prior knowledge about detection errors is not required. In contrast, in our previous work [59], prior knowledge about the probabilities of missing detections was required. There, simulated particle trajectories were used for network training, and missing detections were included by randomly removing a predefined percentage of particle positions. In addition, false positive detections were not taken into account.

We trained our network using the Adam optimizer [85] with an initial learning rate $\eta_{init} = 0.001$, as well as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. To avoid overfitting, we employed early stopping and Gaussian dropout with a rate of $p = 0.25$. For training and validation we used trajectories and detections from 10 synthetic image sequences each consisting of 40 images (512×512 pixels, 16-bit). The data set was split into 85% for training and 15% for validation. We used a mini-batch size of 32 samples.

4.2.3 Experimental Results

We evaluated the performance of our DPHT approach based on data of the Particle Tracking Challenge [65] as well as real live cell fluorescence microscopy data displaying human immunodeficiency virus type 1 (HIV-1) particles and hepatitis C virus (HCV) proteins. We also performed a comparison with previous methods as well as a unidirectional DPHT approach.

Particle Challenge Data

We applied the DPHT approach to image sequences of the Particle Tracking Challenge [65] and performed a comparison with the overall top-three approaches (Methods 5, 1, and 2) described in Sec. 3.2. We also performed a comparison with a recently introduced approach employing piecewise-stationary motion modeling and iterative smoothing (PMMS) [194]. PMMS establishes correspondences by linear programming [190] and localizes particles by SEF. In addition, we compared the performance of our (bidirectional) DPHT approach with a unidirectional DPHT variant (DPHT-uni), which processes the information only in forward direction by using n_l forward LSTM layers ($K = 100$) instead of n_l BLSTM layers ($K = 64$). We also performed a comparison with our previous deep particle tracker (DPT) approach [59] (Sec. 4.1), which employs an LSTM-based RNN for correspondence finding, but does not exploit future information and multiple track hypotheses. Note that for DPHT, DPHT-uni, and DPT we used the same set of detections. Note

also that for these approaches we did not use the Particle Tracking Challenge data for training, but our own generated simulated data as described in Sec. 4.2.2 above.

To study the performance for different motion models under demanding conditions, we used image data from the microtubule and receptor scenario of the Particle Tracking Challenge with medium (~500 particles/frame) and high (~1000 particles/frame) particle densities, and SNRs of 2 and 4. In total, the data comprises 37,613 trajectories. The image data is challenging due to image noise, particle clustering, and particle appearance and disappearance leading to conflicting correspondences and detection errors. Each temporal image sequence consists of 100 images (512×512 pixels, 8-bit). The microtubule data (pixel size of 50 nm) includes elongated particles (anisotropic Gaussian appearance) which perform a directed motion with constant velocity. The receptor data (pixel size of 67 nm) shows round particles (isotropic Gaussian appearance) that switch between random walk and randomly oriented directed motion. Appearance and disappearance of individual particles is defined by a random process. The Rayleigh distance (minimum distance between diffraction-limited particles to allow visual separation) of the image data is 5 pixels. The tracking performance was quantified using the five metrics α, β, JSC, JSC_θ, and $RMSE$ [65] described in Sec. 3.1.

We investigated how the number of BLSTM layers n_l in our DPHT approach affects the tracking performance and computation time. We studied different values $n_l = 1, 2, 3, 4$, and 5. In Fig. 4.8, the tracking performance and computation time of DPHT are shown as a function of n_l for the receptor scenario with medium and high particle density for SNR = 4. It can be seen that for $n_l = 3$, DPHT yields the best overall performance. The computation time increases only somewhat with n_l. Thus, we use $n_l = 3$ in all our experiments (for DPHT and DPHT-uni).

In addition, we investigated how the hyperparameter d (number of future time points for track tree construction) in our DPHT approach affects the tracking performance and computation time. We tested different values $d = 2, 3, 4$, and 5. In Fig. 4.9, the tracking performance and computation time of DPHT are shown as a function of d for the receptor data with medium and high particle density for SNR = 2 and SNR = 4. It can be seen that increasing d up to 4 time points generally improves the tracking performance of DPHT, while for $d = 5$ the performance often decreases. The reason is probably that exploiting too many future time points in cluttered environments causes merged tracks due to appearance and disappearance events in close proximity. The computation time increases with d. Since $d = 4$ is a good compromise between tracking performance and computation time, we use this value in all our experiments (for DPHT and DPHT-uni). Interestingly, for SNR = 2 the performance for high density is better than for medium density (for several metrics), and this is also the case for other methods (cf. Table 4.5). The reason is probably that for high density the particles are better visible due to higher contrast compared to medium density. Computations were performed on

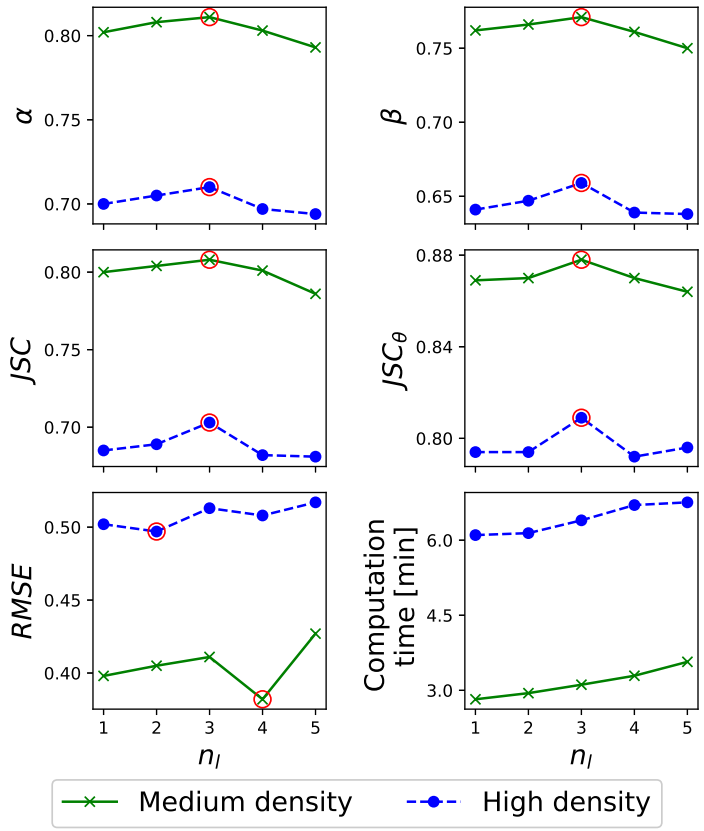


Figure 4.8 Tracking performance and computation time as a function of the number of BLSTM layers n_l for our DPHT approach. We used image data from the receptor scenario with medium and high particle density and SNR $=$ 4. Red circles indicate the best performance.

a workstation with Intel Core i7-8700K CPU (six cores at 3,7 GHz), 32 GB RAM, NVIDIA GeForce GTX1070, and 64-bit Linux operating system.

We also studied the robustness of DPHT with respect to detection errors. We generated data sets with specific percentages of false positive (FP) and false negative (FN) detections based on the ground truth data. Neighboring detections with an Euclidean distance smaller than the Rayleigh distance in the data (5 pixels) were replaced by a single detection to obtain data sets with 0% FN. By randomly removing or adding detections we generated data sets with defined percentages of FN (0%, 5%, 10%, 15%, 20%, 25%, 30%, 40%, 50%, 60%, and 70%) and FP (0%, 25%, 50%, and 70%). In Fig. 4.10, the tracking performance of DPHT is shown as a function of the FN percentage and for different FP percentages for the receptor scenario with medium density and SNR = 4. It can be seen that the tracking performance decreases with increasing FN percentage. With increasing FP percentage, the tracking performance also decreases but less compared to the FN percentage.

The quantitative results for the receptor scenario and microtubule scenario are shown in Table 4.5 and Table 4.6, respectively. Bold and underline represents the best performance, and bold indicates the second best performance. It can be seen that DPHT generally outperforms the other methods. In seven out of eight cases, DPHT performs best in terms of α, β, and JSC. For the receptor scenario with medium density and SNR = 2, DPHT performs second best. In terms of JSC_θ, DPHT performs best in six cases and second best in the two other cases. For three cases DPHT outperforms the other approaches in terms of $RMSE$. DPHT-uni often performs second best. Note that for PMMS the results in[194] are not provided for SNR = 2 and $RMSE$, and all values are given only up to two decimal digits. Example results for image sections (215×215 pixels) of the receptor and microtubule scenario with medium density and SNR = 4 for DPHT are provided in Fig. 4.11. It can be seen that the tracking result well agrees with the ground truth.

In addition, we performed an evaluation of our DPHT approach for all SNR levels (SNR = 1, 2, 4, 7) and all particle densities (low, medium, high) for all 2D scenarios of the Particle Tracking Challenge (receptor, microtubule, vesicle). In total, the data set comprises 36 temporal image sequences and 113,246 trajectories. The results for the overall top-three approaches (Methods 5, 1, and 2), PMMS, DPT, DPHT-uni, and DPHT are provided as Supplementary Material (Table S-I, Table S-II, and Table S-III). The results generally confirm our findings. For α, β, JSC, JSC_θ, and $RMSE$, the DPHT approach performs best in 22, 26, 21, 23, and 6 out of 36 cases, respectively. DPHT-uni often performs second best. The overall mean values of the tracking performance are given in Table 4.7. It can be seen that DPHT outperforms the other approaches in terms of α, β, JSC, and JSC_θ. DPHT-uni performs second best. For RMSE the results of DPHT, DPHT-uni, and DPT are similar. Note that PMMS is not included in the table since in [194] results are not provided for SNR = 1 and SNR = 2, and thus the overall performance cannot be determined. To investigate whether the improvement of DPHT compared to the

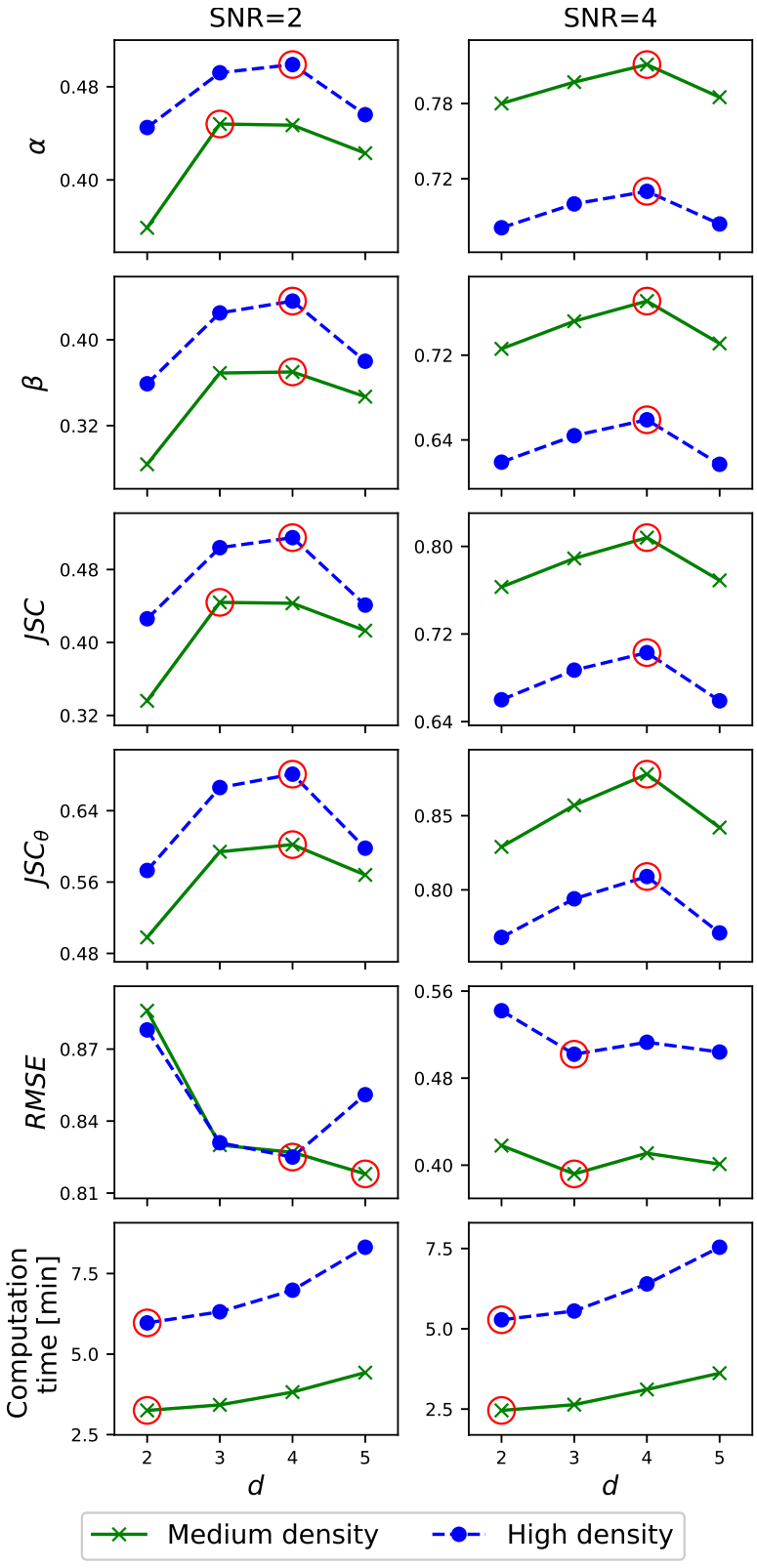


Figure 4.9 Tracking performance and computation time as a function of the number of future time points d for our DPHT approach. We used image data from the receptor scenario with SNR = 2 (left column) and SNR = 4 (right column). Red circles indicate the best performance.

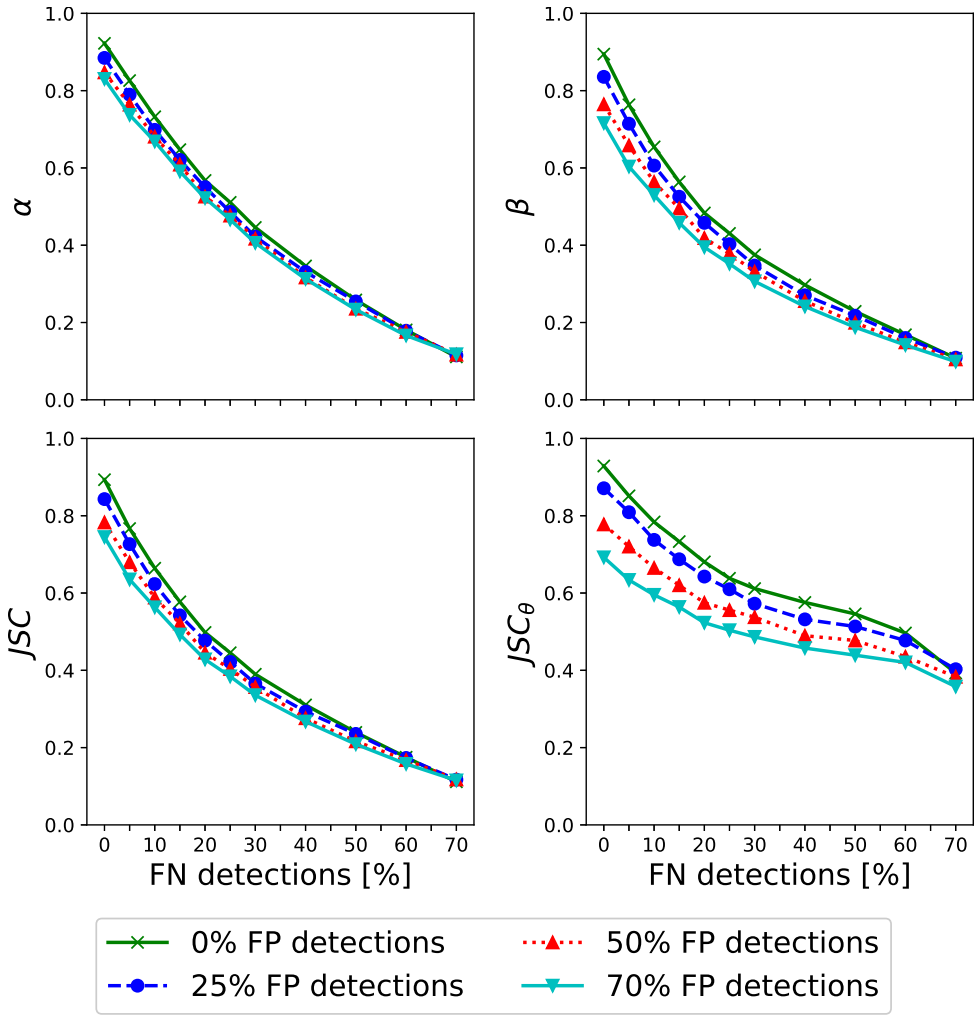


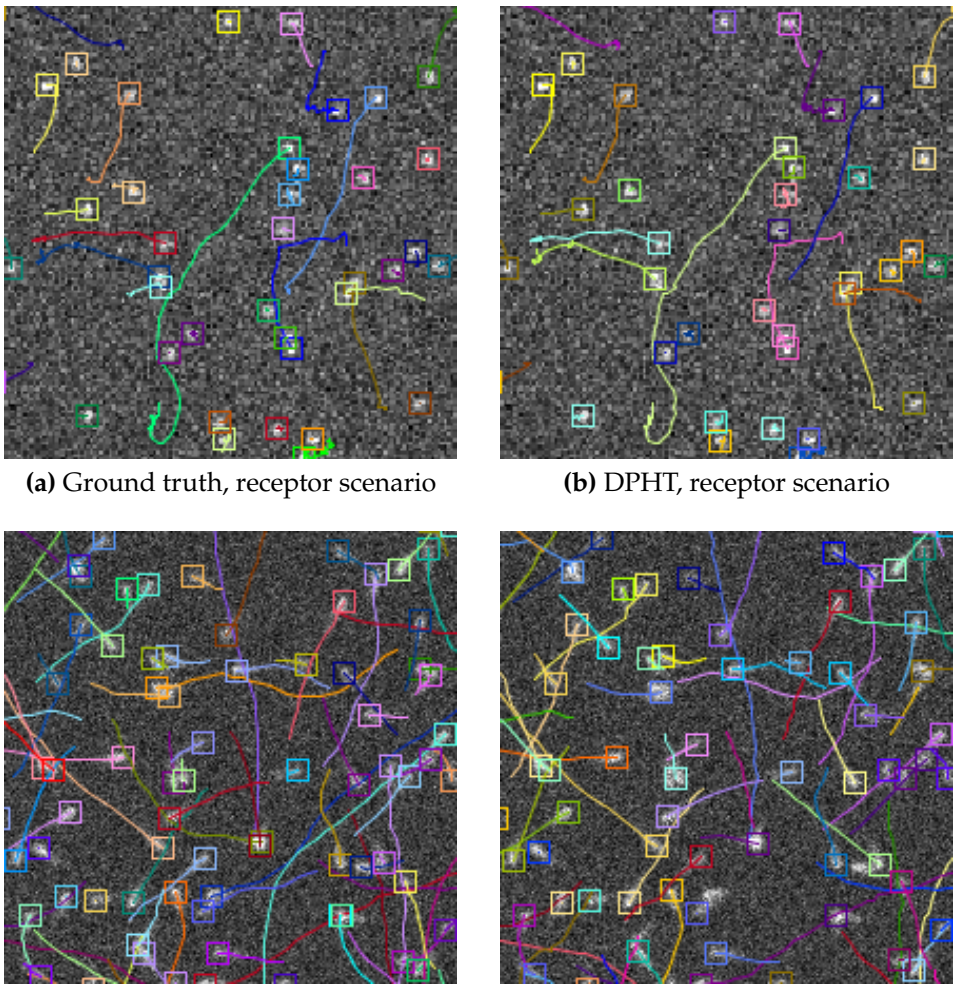
Figure 4.10 Tracking performance of the DPHT approach as a function of the FN percentage and for different FP percentages for the receptor scenario with medium density and SNR = 4.

Table 4.5 Tracking performance of different approaches for data of the receptor scenario from the Particle Tracking Challenge. Bold and underline represents the best performance, and bold indicates the second best performance.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	2	Method 5	0.414	0.347	0.428	0.586	1.207
		Method 1	0.21	0.172	0.197	0.41	0.884
		Method 2	0.439	**0.375**	**0.474**	**0.618**	1.253
		PMMS	-	-	-	-	-
		DPT	0.395	0.289	0.351	0.416	**0.840**
		DPHT-uni	**0.448**	0.368	0.441	0.59	0.851
		DPHT	**0.447**	**0.370**	**0.443**	**0.602**	**0.827**
	4	Method 5	0.714	0.662	0.712	0.828	0.586
		Method 1	0.758	0.704	0.740	0.840	0.477
		Method 2	0.647	0.560	0.632	0.722	0.751
		PMMS	0.78	0.74	0.78	0.86	-
		DPT	0.738	0.677	0.713	0.816	0.45
		DPHT-uni	**0.805**	**0.764**	**0.802**	**0.877**	**0.401**
		DPHT	**0.811**	**0.771**	**0.808**	**0.878**	**0.411**
High	2	Method 5	0.404	0.345	0.423	**0.613**	1.179
		Method 1	0.396	0.306	0.352	0.48	0.881
		Method 2	**0.448**	**0.354**	**0.451**	0.588	1.129
		PMMS	-	-	-	-	-
		DPT	0.432	0.345	0.416	0.566	0.894
		DPHT-uni	0.438	0.352	0.421	0.578	**0.878**
		DPHT	**0.499**	**0.436**	**0.515**	**0.681**	**0.825**
	4	Method 5	0.545	0.474	0.529	0.712	0.797
		Method 1	0.622	0.55	0.59	0.731	0.573
		Method 2	0.584	0.504	0.578	0.701	0.857
		PMMS	0.68	0.62	0.67	0.78	-
		DPT	0.617	0.546	0.587	0.736	0.572
		DPHT-uni	**0.707**	**0.655**	**0.699**	**0.801**	**0.522**
		DPHT	**0.710**	**0.659**	**0.703**	**0.809**	**0.513**

Table 4.6 Tracking performance of different approaches for data of the microtubule scenario from the Particle Tracking Challenge. Bold and underline represents the best performance, and bold indicates the second best performance.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	2	Method 5	0.283	0.248	0.351	0.562	1.721
		Method 1	0.262	0.197	0.324	0.496	2.16
		Method 2	0.371	0.158	0.266	0.306	2.282
		PMMS	-	-	-	-	-
		DPT	0.348	0.263	0.349	0.491	**1.506**
		DPHT-uni	**0.410**	**0.352**	**0.456**	**0.616**	**1.526**
		DPHT	**0.443**	**0.393**	**0.505**	**0.674**	1.586
	4	Method 5	0.46	0.402	0.523	0.696	1.403
		Method 1	0.353	0.264	0.373	0.55	1.682
		Method 2	0.465	0.225	0.341	0.363	2.036
		PMMS	0.44	0.39	0.58	0.7	-
		DPT	0.488	0.373	0.449	0.556	**1.066**
		DPHT-uni	**0.625**	**0.561**	**0.663**	**0.772**	**1.073**
		DPHT	**0.655**	**0.618**	**0.723**	**0.839**	1.118
High	2	Method 5	0.183	0.163	0.24	0.48	1.899
		Method 1	0.19	0.153	0.25	0.512	2.178
		Method 2	0.296	0.126	0.223	0.301	2.354
		PMMS	-	-	-	-	-
		DPT	0.307	0.209	0.294	0.4	**1.663**
		DPHT-uni	**0.374**	**0.306**	**0.415**	**0.576**	**1.667**
		DPHT	**0.402**	**0.320**	**0.438**	**0.551**	1.71
	4	Method 5	0.314	0.264	0.371	0.602	1.633
		Method 1	0.272	0.21	0.299	0.544	1.715
		Method 2	0.396	0.194	0.306	0.361	2.102
		PMMS	0.35	0.3	0.46	0.63	-
		DPT	0.414	0.313	0.389	0.524	**1.209**
		DPHT-uni	**0.509**	**0.451**	**0.546**	**0.709**	**1.198**
		DPHT	**0.548**	**0.501**	**0.605**	**0.758**	1.255



(a) Ground truth, receptor scenario

(b) DPHT, receptor scenario

(c) Ground truth, microtubule scenario

(d) DPHT, microtubule scenario

Figure 4.11 Ground truth and tracking results for image sections (215×215 pixels) of the receptor scenario (top) and microtubule scenario (bottom) with medium density and SNR = 4. The image contrast was enhanced for better visualization.

Table 4.7 Overall mean values of the tracking performance of different approaches for all 2D image data from the Particle Tracking Challenge (all SNR levels and all particle densities for receptor, microtubule, and vesicle). Bold and underline represents the best performance, and bold indicates the second best performance.

Method	α	β	JSC	JSC_θ	$RMSE$
Method 5	0.454	0.408	0.480	0.615	1.319
Method 1	0.391	0.342	0.400	0.542	1.227
Method 2	0.441	0.352	0.444	0.549	1.597
DPT	0.464	0.400	0.449	0.584	**1.034**
DPHT-uni	**0.503**	**0.455**	**0.514**	**0.651**	**1.068**
DPHT	**0.515**	**0.473**	**0.533**	**0.673**	1.069

other methods (Method 5, Method 1, Method 2, DPT, DPHT-uni) is statistically significant for α, β, JSC, and JSC_θ, we performed a Wilcoxon signed-rank test (non-parametric test) with significance level of 5% for all 2D Particle Tracking Challenge data (all SNR levels and all particle densities for receptor, microtubule, and vesicle; Table 4.7). We obtained $p < 0.002$ for JSC and $p < 0.0005$ for α, β, and JSC_θ. Thus, DPHT yields a statistically significant improvement compared to the other methods.

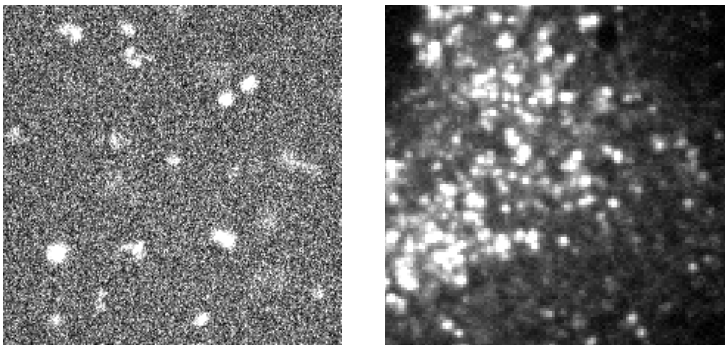
To study the reproducibility of DPHT, we have performed multiple training runs (15 runs) and applied the approach to data of the receptor scenario with medium density and SNR = 4. For the mean and standard deviation for the five performance metrics we obtained: $\alpha = 0.811 \pm 0.004$, $\beta = 0.771 \pm 0.006$, $JSC = 0.809 \pm 0.007$, $JSC_\theta = 0.878 \pm 0.005$, $RMSE = 0.412 \pm 0.002$. Compared to the original values in Table 4.5 ($\alpha = 0.811$, $\beta = 0.771$, $JSC = 0.808$, $JSC_\theta = 0.878$, $RMSE = 0.411$) the results are very similar and the standard deviation is very small indicating a good reproducibility.

Real Live Cell Fluorescence Microscopy Images

We also evaluated the DPHT approach based on real fluorescence microscopy image sequences displaying human immunodeficiency virus type 1 (HIV-1) particles and hepatitis C virus (HCV) proteins. We used five image sequences with fluorescently labeled HIV-1 particles acquired with a confocal spinning disk microscope and an EM-CCD camera. Each image sequence consists of 50 images (16-bit) with 1000×1000 pixels (Seq. 1 to Seq. 5). We also used three image sequences displaying the nonstructural protein 5A (NS5A) of HCV (30 time points, 1000×1000 pixels, 16-bit) denoted by Seq. 6 to Seq. 8. The images were acquired by a confocal spinning disk microscope and a CMOS camera. Example sections of Seq. 2 with 230×230 pixels and Seq. 6 with 115×115 pixels are shown in Fig. 4.12. The image sequences of this data set differ in the level of image noise, the motion behavior of the particles, and the particle density. For the real images, ground truth for performance evaluation for regions with clutter and large motion was obtained by

Table 4.8 Real Live Cell Fluorescence Microscopy Image Sequences.

Sequence	Object type	No. of particles
Seq. 1	HIV-1 particles	86
Seq. 2	HIV-1 particles	117
Seq. 3	HIV-1 particles	70
Seq. 4	HIV-1 particles	113
Seq. 5	HIV-1 particles	87
Seq. 6	NS5A (HCV)	75
Seq. 7	NS5A (HCV)	55
Seq. 8	NS5A (HCV)	90



(a) Seq. 2 (HIV-1) **(b)** Seq. 6 (HCV)

Figure 4.12 Sections of image sequence Seq. 2 (HIV-1) and Seq. 6 (HCV). The image contrast was enhanced for better visualization.

manual annotation using the ImageJ plugin MTrackJ [222]. An overview of the used image sequences is given in Table 4.8.

We compared the performance of DPHT with the ParticleTracker (PT) [187], the Kalman filter based approach (KF) implemented in the ImageJ plugin TrackMate [189], the multiple-hypothesis tracking (MHT) approach using Kalman filters and multiple motion models [193] implemented in the Icy software [212], and our previous DPT approach [59] (Sec. 4.1). For PT, KF, and MHT, we tested different parameter settings and applied the settings which yielded the best results. PT uses iterative intensity-weighted centroid calculation for particle localization and establishes correspondences by greedy hill-climbing optimization with topological constraints. We used a radius of 5 pixels, a cutoff of 0.001, a percentile of 0.3, a displacement of 10 pixels, and a linking range of 2 for the HIV-1 data. For the HCV data we set the radius to 3 pixels, the percentile to 1, and the displacement to 5. KF employs SEF for particle localization and particle linking is based on a linear assignment method used in u-track [190]. For the HIV-1 data, we used a diameter of 10 pixels and a threshold of 5, while for the HCV data we set the diameter to 5 pixels and the threshold to 60. We used an initial search radius of 15 pixels and a search radius of 20 pixels for the HIV-1 data, and for the HCV data we set both parameters to 7.5 pixels. For the other parameters we used the default values.

MHT localizes particles by a wavelet-based detection scheme. We used a scale of 4 with a coefficient threshold of 65 for the HIV-1 data, and a scale of 2 with a coefficient threshold of 55 for the HCV data. For the HIV-1 data, we set the detection probability for each particle to 0.95, the expected number of new tracks per frame to 10 for HIV-1 and 30 for HCV, the expected number of objects in the first frame to 100, the expected track length to 8, the average length of displacement to 12.5, and the threshold for the existence probability to 0.01. For the other parameters we used the automatically estimated values. For DPT, we set the hyperparameters as described in [59]. Note that for DPHT, DPHT-uni, and DPT we used the same set of detections, and tracking parameters were not adapted (except the two detection parameters for SEF), i.e. we directly applied our approaches to the real data while training was performed using only simulated data (see Sec. 4.2.2 above). The computation time for one image sequence of the HIV-1 and HCV data was about 0.5 min and 1 min, respectively, using a workstation as specified in Sec. 4.2.3 above.

The quantitative results for all approaches for the HIV-1 data and the HCV data are shown in Table 4.9 and Table 4.10, respectively. It can be seen that for most real image sequences, DPHT outperforms the other methods. For β and JSC, DPHT performs best for all eight images sequences. DPHT outperforms the other approaches in terms of α for seven image sequences, and in terms of $RMSE$ for three image sequences. For JSC_θ, DPHT performs best for four image sequences and second best for three image sequences. DPHT-uni often performs second best. Sample tracking results for image sequence Seq. 4 (HIV-1) are shown in Fig. 4.13. The sequence includes five real particles in close proximity. It can be seen that DPHT yields the best result and maintains the correct identity for all particles. PT yields a broken trajectory (blue) and a missing trajectory (green) due to missing detections. KF causes an identity switch between the blue and green trajectory. Also, the green trajectory was terminated too early and a false positive track was initiated due to a spurious particle detection (red trajectory). MHT and DPT yield a broken trajectory (green) due to temporal occlusion causing that two particles were falsely detected as one particle. In addition, the red trajectories consist of two track segments belonging to different particles (green trajectory and an already terminated trajectory). Sample tracking results for image sequence Seq. 7 (HCV) are shown in Fig. 4.14. The sequence includes two particles in close proximity. It can be seen that DPHT maintains the correct identity of the particles, while the other approaches yield a broken trajectory (yellow) due to a temporal merging event.

4.2.4 Conclusion

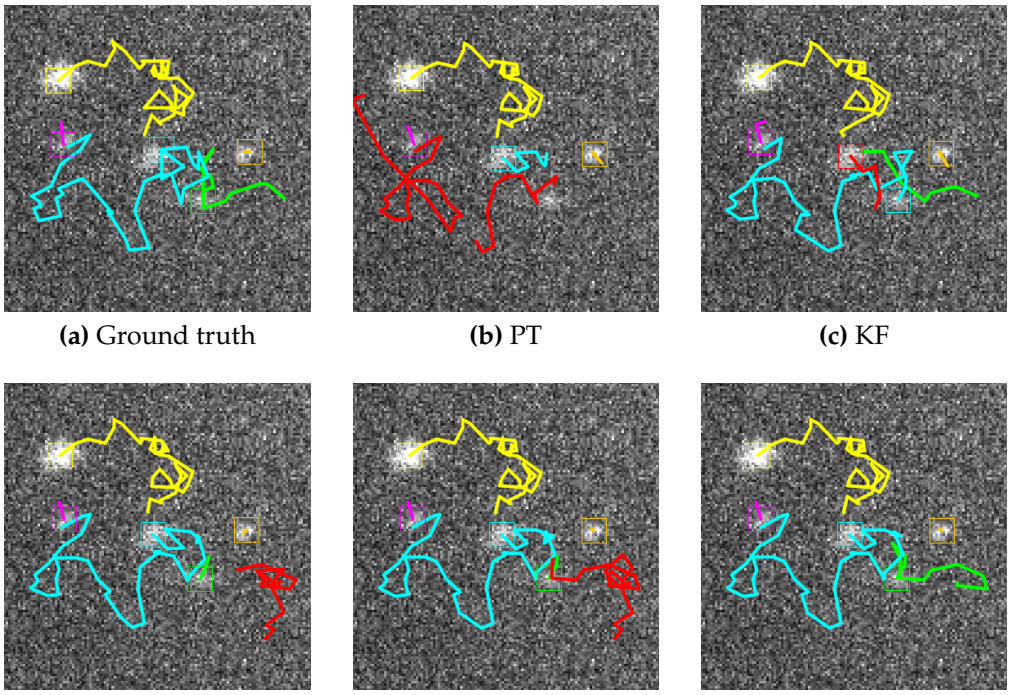
We introduced a new deep learning approach for tracking multiple particles in temporal fluorescence microscopy images. Our deep particle hypotheses tracker (DPHT) is based on an RNN architecture that exploits past and future information

Table 4.9 Tracking performance of different approaches for real fluorescence microscopy images displaying HIV-1 particles. Bold and underline represents the best performance, and bold indicates the second best performance. Mean values are also shown.

Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. 1	PT	0.349	0.141	0.245	0.173	2.989
	KF	0.398	0.314	0.445	0.518	**2.737**
	MHT	0.456	0.392	**0.618**	0.525	3.155
	DPT	0.483	0.403	0.525	**0.564**	**2.747**
	DPHT-uni	**0.508**	**0.452**	**0.618**	0.553	2.833
	DPHT	**0.535**	**0.478**	**0.633**	**0.573**	2.815
Seq. 2	PT	0.262	0.145	0.216	0.254	3.410
	KF	0.347	0.228	0.328	0.359	**2.818**
	MHT	0.367	**0.304**	**0.454**	**0.440**	3.393
	DPT	**0.389**	0.287	0.409	0.390	3.158
	DPHT-uni	0.371	0.287	0.401	0.396	**2.943**
	DPHT	**0.404**	**0.332**	**0.460**	**0.457**	3.089
Seq. 3	PT	0.317	0.066	0.119	0.074	2.680
	KF	0.381	0.209	0.325	0.227	2.586
	MHT	0.397	0.257	0.364	0.295	2.714
	DPT	0.406	0.255	0.392	0.327	2.671
	DPHT-uni	**0.498**	**0.429**	**0.550**	**0.402**	**2.517**
	DPHT	**0.507**	**0.446**	**0.564**	**0.425**	**2.401**
Seq. 4	PT	0.248	0.069	0.121	0.114	3.139
	KF	0.388	0.270	0.394	0.370	2.964
	MHT	0.409	**0.357**	**0.536**	**0.464**	3.242
	DPT	**0.424**	0.326	0.481	0.428	**2.901**
	DPHT-uni	0.422	0.352	0.509	0.414	2.909
	DPHT	**0.442**	**0.375**	**0.541**	**0.434**	**2.789**
Seq. 5	PT	0.346	0.157	0.235	0.219	2.797
	KF	0.397	0.282	0.379	0.372	2.757
	MHT	0.382	0.350	**0.518**	0.457	3.239
	DPT	**0.435**	0.360	0.477	**0.523**	**2.498**
	DPHT-uni	0.430	**0.392**	0.508	**0.500**	**2.624**
	DPHT	**0.439**	**0.411**	**0.545**	0.495	2.632
Mean values	PT	0.304	0.116	0.187	0.167	3.003
	KF	0.382	0.261	0.374	0.369	2.772
	MHT	0.402	0.332	0.498	0.436	3.149
	DPT	0.428	0.326	0.457	0.446	2.795
	DPHT-uni	**0.446**	**0.382**	**0.517**	**0.453**	**2.765**
	DPHT	**0.465**	**0.409**	**0.549**	**0.477**	**2.745**

Table 4.10 Tracking performance of different approaches for real fluorescence microscopy images displaying HCV proteins. Bold and underline represents the best performance, and bold indicates the second best performance. Mean values are also shown.

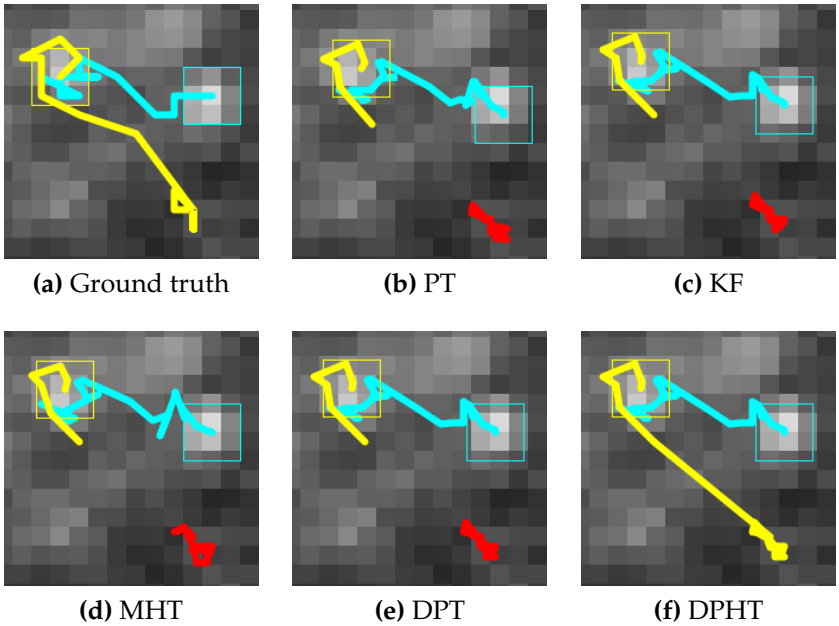
Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. 6	PT	0.545	0.506	0.635	0.644	1.253
	KF	0.528	0.505	0.586	0.659	**0.979**
	MHT	0.560	0.527	0.648	0.693	1.210
	DPT	**0.606**	0.559	0.649	0.644	**0.904**
	DPHT-uni	0.599	**0.567**	**0.664**	**0.713**	1.072
	DPHT	**0.610**	**0.575**	**0.676**	**0.697**	1.016
Seq. 7	PT	0.590	0.496	0.629	0.557	1.064
	KF	0.559	0.481	0.564	0.550	1.088
	MHT	0.540	0.480	0.588	**0.611**	1.237
	DPT	**0.633**	**0.540**	**0.632**	0.605	**1.008**
	DPHT-uni	0.603	0.531	0.616	0.568	**1.025**
	DPHT	**0.619**	**0.556**	**0.652**	**0.622**	1.117
Seq. 8	PT	0.324	0.306	0.382	0.414	1.255
	KF	0.525	0.432	0.499	0.529	1.137
	MHT	0.436	0.422	0.507	**0.588**	1.299
	DPT	**0.614**	**0.499**	0.586	0.527	**1.068**
	DPHT-uni	0.609	0.493	**0.592**	0.519	1.082
	DPHT	**0.626**	**0.510**	**0.614**	**0.538**	**1.052**
Mean values	PT	0.487	0.436	0.549	0.538	1.191
	KF	0.537	0.473	0.550	0.579	1.068
	MHT	0.512	0.476	0.581	**0.631**	1.249
	DPT	**0.617**	**0.533**	0.623	0.592	**0.993**
	DPHT-uni	0.604	0.530	**0.624**	0.600	**1.060**
	DPHT	**0.618**	**0.547**	**0.647**	**0.619**	1.062



(a) Ground truth **(b)** PT **(c)** KF

(d) MHT **(e)** DPT **(f)** DPHT

Figure 4.13 Ground truth and results of different tracking approaches for image sequence Seq. 4 (HIV-1). The image contrast was enhanced for better visualization.



(a) Ground truth **(b)** PT **(c)** KF

(d) MHT **(e)** DPT **(f)** DPHT

Figure 4.14 Ground truth and results of different tracking approaches for image sequence Seq. 7 (HCV). The image contrast was enhanced for better visualization.

in both forward and backward direction. Also, our network determines assignment probabilities jointly across multiple detections, and computes the probabilities of missing detections. In addition, DPHT propagates track hypotheses into the future so that information at later time points can be used to resolve ambiguities. To handle track initiation and termination, existence probabilities of individual objects are calculated. Manually labeled data and a handcrafted similarity measure are not needed. Tuning of tracking parameters and selection of a dynamic model are not required. Also, prior assumptions about probability distributions are not necessary.

We evaluated the performance of our approach using image data of the Particle Tracking Challenge as well as real fluorescence microscopy image sequences of virus structures. Our experimental results show that exploiting information from future time points by propagating hypotheses improves the tracking performance. In addition, processing the temporal information in forward and backward direction improves the result compared to processing the information only in forward direction. A quantitative comparison with previous methods showed that our approach yields superior results.

Chapter 5

Deep Probabilistic Tracking of Particles

In this chapter, we present a novel probabilistic deep learning method for particle tracking in fluorescence microscopy images. The work has been published in [61].

5.1 Introduction

In previous work on tracking particles in fluorescence microscopy images, classical deterministic and probabilistic approaches have been introduced. Deterministic approaches comprise two steps: Particle detection and correspondence finding (e.g., [187, 213, 214, 215, 216]). While being computationally efficient, these approaches do not take into account spatial and temporal uncertainties which often leads to difficulties under challenging conditions (e.g., low SNR, high object density). In comparison, probabilistic approaches follow a Bayesian paradigm and define a posterior distribution on the variables describing the object state. The posterior can be resolved via a sequential Bayesian filter such as the Kalman filter (e.g., [144, 151, 192, 193, 194, 223]) or the particle filter (e.g., [58, 149, 196, 197]). However, probabilistic approaches typically require selecting a suitable dynamic model and use prior assumptions about the noise statistics (e.g., image and motion noise), which do not necessarily hold. Moreover, classical tracking methods often involve numerous parameters that are difficult to adjust, particularly for non-experts, and do not always have a biophysical interpretation.

Deep learning methods provide state-of-the-art performance in various computer vision tasks including image classification, object detection, and segmentation (e.g., [6, 7, 8]). Approaches for tracking objects in video images of natural scenes (e.g., pedestrians, cars) use deep learning for different purposes [161]. Convolutional neural networks (CNNs) are employed for extracting appearance features (e.g., [224]), for generating a discriminative appearance model (e.g., [225]), for object detection (e.g., [130, 226]), for computing assignment scores (e.g., [227]), and for motion prediction (e.g., [228, 229]). Recurrent neural networks (RNNs) are often used to compute assignment scores between tracklets and detections (or

tracklets and other tracklets) (e.g., [18]), which typically exploit appearance features (e.g., [19, 230]) which can hardly be exploited to track indistinguishable particles. In Farrell et al. [231], CNNs and an RNN are combined for correspondence finding between detector hits in simulated high-energy physics data without using prediction and update steps as in Bayesian filtering.

Recently, deep learning methods for tracking biological objects in microscopy images have been introduced showing promising results. CNNs and RNNs have been used to exploit appearance features for cell tracking (e.g., [21, 206, 207, 208]). Since cells (and natural objects) are very different from fluorescent particles both regarding shape and dynamics, these approaches cannot be directly applied for particle tracking. In addition, appearance features are not a reliable cue for correspondence finding for (almost) indistinguishable particles. Few works employed CNNs for particle detection in fluorescence microscopy images (e.g., [153, 155, 156, 157]). In Zhong et al. [232], a CNN was used to determine the position of individual polystyrene particles along the z-direction of epifluorescence microscopy images. Sun and Paninski [209] proposed an RNN which approximates the posterior transition probability densities to track clathrin-coated pits. However, for correspondence finding a classical nearest neighbor strategy is used. RNNs using (past) temporal information for correspondence finding of fluorescent particles were introduced in Yao et al. [210] and in our work in Spilger et al. [59] (see Sec. 4.1). Smal et al. [211] employ a denoising autoencoder and score matching to learn a motion model from data within a classical multiple hypotheses tracking (MHT) framework. We introduced a bidirectional RNN exploiting past and future information as well as multiple track hypotheses for correspondence finding (see Sec. 4.2). Yao et al. [22] described an RNN that uses handcrafted and learned features. However, none of these deep learning methods takes into account uncertainty, neither in the network model (epistemic uncertainty) nor the inherent noise in the image data (aleatoric uncertainty).

Deep neural networks considering *uncertainty* have been introduced for natural and medical images for different tasks such as segmentation (e.g., street traffic scenes, CT images), disease detection (e.g., fundus images), super-resolution (e.g., diffusion MR images), and image translation (e.g., CT images to MR images) (e.g., [233, 234, 235, 236]). Since neural networks generally consist of a large number of parameters as well as non-linear activations, computing the (multi-modal) posterior distribution of a network output is intractable. Thus, approximation methods have been introduced, which are mainly based on Bayesian inference or Monte-Carlo sampling. *Bayesian neural networks* represent the parameters (weights) by probability distributions instead of using single values (e.g., [115, 119, 237, 238]). Consequently, the network outputs can also be represented by probability distributions and calculated analytically employing graphical models [239] or non-linear belief networks [240]. Alternatively, *Monte-Carlo sampling* can be employed. Often, Monte-Carlo samples are obtained using ensembles of neural networks.

These ensembles can be generated by differently trained neural networks (e.g., [122, 241]) or employing dropout during training and testing (Monte-Carlo dropout, [118]). However, Monte-Carlo sampling approaches include epistemic uncertainty (model uncertainty) but not aleatoric uncertainty (data uncertainty). In Kendall and Gal [113], Monte-Carlo dropout was employed to capture epistemic uncertainty and a standard deviation variable is added to each output to include aleatoric uncertainty. None of the above described methods considering uncertainty was employed for object tracking in microscopy images.

In this contribution, we present a novel deep neural network architecture for tracking particles in fluorescence microscopy images which exploits both aleatoric and epistemic uncertainty. Inspired by classical Bayesian filtering, the network learns to predict the next state and to correct the predicted state based on an assigned detection. Gated Recurrent Units (GRUs) [103] are used to exploit both short- and long-term temporal dependencies of individual object dynamics. Epistemic uncertainty is incorporated by variational Bayesian learning using an approximation of the lower bound for efficient learning [115, 119], which does not require computationally expensive iterative inference schemes such as Markov chain Monte Carlo. Bayesian layers with reparameterization are employed, where parameters are represented by Gaussian distributions. During network training via variational inference the parameters of these probability distributions are learned instead of directly learning the network weights. To capture aleatoric uncertainty due to the particle detector and noise of object motion, the network learns estimating the mean and standard deviation of Gaussian distributions from which the predicted and updated state can be determined. We also introduce a neural network that determines assignment probabilities for correspondence finding based on the Euclidean distance between the predicted states and particle detections obtained by the spot-enhancing filter [147] and Gaussian fitting. Assignment probabilities are computed jointly across multiple detections, and probabilities of missing detections are also determined. Network training is based on synthetic data only and manually annotated data is not needed. We propose a novel scheme to generate synthetic training images using automatically extracted information from the images in an application. This enables simulating a large number and spectrum of training images that represent well the images in an application. In contrast, our previous scheme in Spilger et al. [60] (described in Sec. 4.2) did not use automatically extracted information from the real data to generate training images. The uncertainty information determined by our approach for the computed trajectories is important to assess their reliability and, for example, to exclude unreliable tracks (or track points) to increase the accuracy of subsequent motion analysis (e.g., mean-squared displacement analysis) as we show in our experiments. In addition, we demonstrate that the uncertainty can be exploited to assess the suitability of the training data and to select the generated training data set with the best-suited motion model so that the training data better represents the real data in an application.

Our approach is the first probabilistic deep learning method for particle tracking in microscopy images and takes into account both aleatoric and epistemic uncertainty. We verified that both types of uncertainty are captured by the network. Besides taking into account and exploiting uncertainty information, we propose a novel neural network architecture which differs from our previous work in Spilger et al. [60] (Sec. 4.2), where prediction and update steps as in Bayesian filtering were not used nor Bayesian layers and GRU layers. In addition, we here use a different loss function, namely a balanced focal loss [93] and different activation functions PReLUs, [75]). We have conducted a quantitative performance evaluation based on synthetic and real 2D as well as 3D fluorescence microscopy images. We used data from the Particle Tracking Challenge as well as real live cell microscopy image sequences displaying the hepatitis C virus (HCV) protein NS5A, the HCV associated protein ApoE, and chromatin structures (labeled during DNA replication). It turned out that our approach yields state-of-the-art or improved results compared to previous methods.

5.2 Method

In this section, we present our novel probabilistic deep learning approach for tracking multiple particles in live cell fluorescence microscopy images, denoted as Deep Probabilistic Particle Tracker (DPPT). First, we give an overview of our approach. Then, we describe the classical Bayesian filtering framework used in previous particle tracking approaches. After that, we introduce our deep learning architecture mimicking classical Bayesian filtering and taking into account both aleatoric and epistemic uncertainty. We also present a neural network architecture to compute assignment probabilities for correspondence finding. Finally, we provide details on the network training.

5.2.1 Overview of the Proposed Tracking Approach

Fig. 5.1 provides a schematic overview of the proposed DPPT approach. For particle detection, we employ the spot-enhancing filter (SEF) [147] and Gaussian fitting yielding a set of detections represented by image positions. For state prediction and state update, a recurrent neural network (RNN) with Gated Recurrent Units (GRUs) [103] is introduced mimicking classical Bayesian filtering. The network takes into account both aleatoric and epistemic uncertainty. Epistemic uncertainty is captured by learning Gaussian distributions for the parameters of the network. To take into account aleatoric uncertainty, the network estimates the mean and standard deviation of Gaussian distributions from which the predicted and updated states are computed. For correspondence finding, we introduce a neural network that determines assignment probabilities as well as probabilities of missing detections between the predicted states and particle detecions. The Jonker-Volgenant shortest

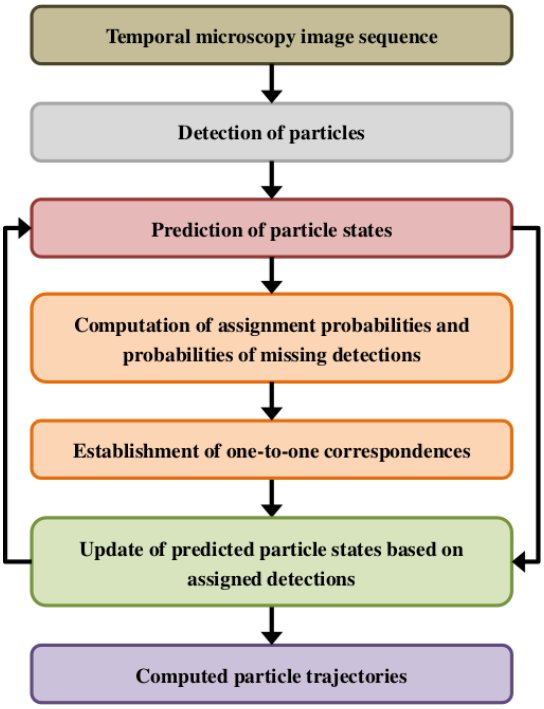


Figure 5.1 Overview of the proposed Deep Probabilistic Particle Tracker (DPPT).

augmenting path algorithm [186] is employed to establish one-to-one correspondences based on the computed assignment probabilities of all objects and the probabilities of missing detections.

5.2.2 Bayesian Filtering

We represent a biological particle i in a temporal microscopy image sequence at time point t by the state vector $\mathbf{x}_t^i \in \mathbb{R}^D$, which is reflected by the noisy measurement (detection) $\mathbf{y}_t^j \in \mathbb{R}^D$. In our settings, both the particle state $\mathbf{x}_t^i$ and the detection $\mathbf{y}_t^j$ are described by the image position at time point t, and therefore D equals the number of image dimensions in an experimental setup.

The goal of Bayesian filtering is to estimate $\mathbf{x}_t^i$ recursively over time based on a sequence of noisy detections $\mathbf{y}_{1:t}^j$. For each time point t, this estimation process comprises two consecutive steps: Prediction and update. Based on the posterior distribution $p(\mathbf{x}_{t-1}^i|\mathbf{y}_{1:t-1}^j)$ at time point $t-1$, the prediction step evaluates the particle dynamics using a dynamical model $p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)$ to determine the prior distribution $p(\mathbf{x}_t^i|\mathbf{y}_{1:t-1}^j)$ at time point t:

$$p(\mathbf{x}_t^i|\mathbf{y}_{1:t-1}^j) = \int p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_{t-1}^i|\mathbf{y}_{1:t-1}^j)\mathrm{d}\mathbf{x}_{t-1}^i \tag{5.1}$$

In the update step, Bayes' rule is applied to compute the posterior distribution $p(\mathbf{x}_t^i|\mathbf{y}_{1:t}^j)$ at time point t from the prior distribution $p(\mathbf{x}_t^i|\mathbf{y}_{1:t-1}^j)$ by incorporating the detection $\mathbf{y}_t^j$ via a measurement model $p(\mathbf{y}_t^j|\mathbf{x}_t^i)$:

$$p(\mathbf{x}_t^i|\mathbf{y}_{1:t}^j) \propto p(\mathbf{y}_t^j|\mathbf{x}_t^i)p(\mathbf{x}_t^i|\mathbf{y}_{1:t-1}^j) \tag{5.2}$$

The state $\mathbf{x}_t^i$ can be determined from the posterior distribution $p(\mathbf{x}_t^i|\mathbf{y}_{1:t}^j)$. The two most common approaches for solving Eq. (5.1) and Eq. (5.2) are the Kalman filter and the particle filter. In contrast, we propose a deep learning approach to mimic classical Bayesian filtering.

5.2.3 Bayesian Neural Network for Prediction and Update

The proposed probabilistic neural network mimics classical Bayesian filtering. The network architecture can be subdivided into a state prediction and update block (see Fig. 5.2). The prediction block employs a GRU-based RNN [103] exploiting both short- and long-term temporal dependencies in the dynamics of an individual particle to estimate its next state. The update block corrects the predicted state based on the assigned detection. Bayesian layers [115, 119] are used within the network to take into account *epistemic uncertainty*. In addition, the standard deviation for the predicted and updated state is computed to provide information about the *aleatoric uncertainty*. In contrast, previous work on object tracking did not consider uncertainty (e.g., [18, 19, 22, 59, 60, 210]). For each time point $t-1$, the network computes two output vectors for particle i for the next time point t: The predicted state $\hat{\mathbf{x}}_t^i \in \mathbb{R}^D$ and the updated state $\mathbf{x}_t^i \in \mathbb{R}^D$.

The prediction block of our network comprises L GRU layers (we used $L = 2$) each consisting of K units. The structure of GRU allows capturing both short- and long-term temporal dependencies. To solve the vanishing and exploding gradient problems that occur in standard RNNs, GRU uses a hidden state and two gates regulating the information to be kept or discarded at each time point. The gating is designed similarly to that in the Long Short-Term Memory (LSTM) [101], however, GRU is less complex than LSTM (e.g., it has only two gates instead of three in the LSTM) and is faster to compute. The reset gate of GRU determines which part of the previous hidden state is combined with the current input to compute a candidate state. The update gate of GRU determines which portion of the previous hidden state is preserved and which portion of the candidate state (derived from the reset gate) is added to the final hidden state. In more detail, the hidden state of layer l at time point t is represented by $\mathbf{h}_t^{i,l} \in \mathbb{R}^K$ (for $l = 1, \ldots, L$), while $\mathbf{h}_t^{i,0} \in \mathbb{R}^D$ denotes the network input vector. The output of the last GRU layer is denoted by $\mathbf{h}_t^{i,L}$. To predict the state of object i for the next time point t, the state vector $\mathbf{x}_{t-1}^i$ of the object at time point $t-1$ is used as input vector, i.e. $\mathbf{h}_t^{i,0} = \mathbf{x}_{t-1}^i$. For a particular GRU layer l and time point t, the update gate $\mathbf{z}_t^{i,l}$ and reset gate $\mathbf{r}_t^{i,l}$ are computed

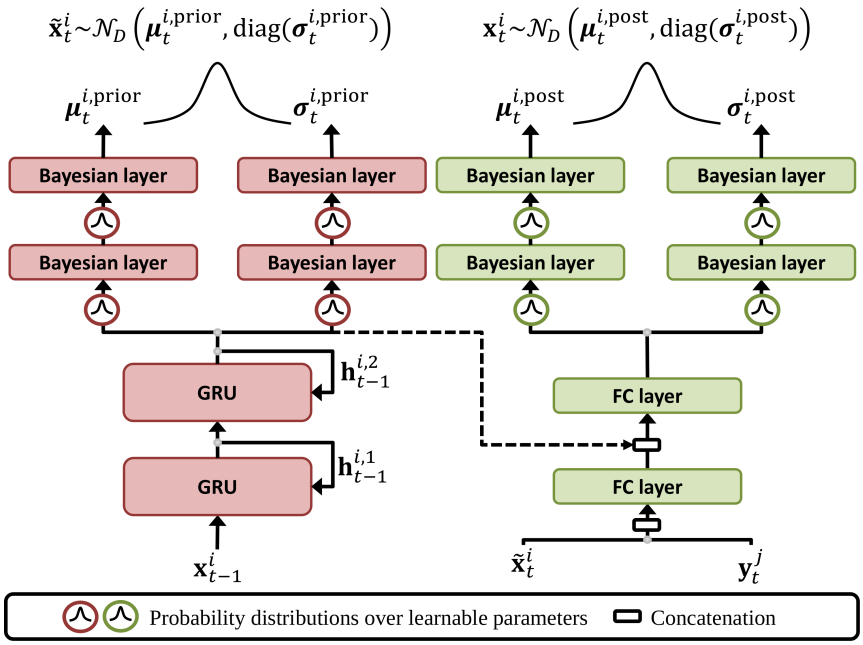


Figure 5.2 Architecture of the proposed probabilistic neural network. Red indicates the network block for state prediction, and green the network block for state update. The dashed line indicates how the hidden state of the last GRU layer is exploited by the update block.

based on the previous hidden state $\mathbf{h}_{t-1}^{i,l}$ at time point $t-1$ and the hidden state $\mathbf{h}_t^{i,l-1}$ of the previous GRU layer:

$$\mathbf{z}_t^{i,l} = \sigma(\mathbf{W}_z^l \mathbf{h}_t^{i,l-1} + \mathbf{U}_z^l \mathbf{h}_{t-1}^{i,l} + \mathbf{b}_z^l) \tag{5.3}$$

$$\mathbf{r}_t^{i,l} = \sigma(\mathbf{W}_r^l \mathbf{h}_t^{i,l-1} + \mathbf{U}_r^l \mathbf{h}_{t-1}^{i,l} + \mathbf{b}_r^l) \tag{5.4}$$

where $\mathbf{W}_z^l$, $\mathbf{U}_z^l$, $\mathbf{b}_z^l$, $\mathbf{W}_r^l$, $\mathbf{U}_r^l$, and $\mathbf{b}_r^l$ represent the learnable parameters of the two gates. σ is the logistic sigmoid activation. Then, the new candidate state $\tilde{\mathbf{h}}_t^{i,l}$ is computed as follows:

$$\tilde{\mathbf{h}}_t^{i,l} = \tanh(\mathbf{W}_h^l \mathbf{h}_t^{i,l-1} + \mathbf{U}_h^l (\mathbf{r}_t^{i,l} \odot \mathbf{h}_{t-1}^{i,l}) + \mathbf{b}_h^l) \tag{5.5}$$

where $\mathbf{W}_h^l$, $\mathbf{U}_h^l$, and $\mathbf{b}_h^l$ are the learnable parameters. $\odot$ denotes the element-wise (Hadamard) multiplication and tanh is the hyperbolic tangent activation function. The previous hidden state $\mathbf{h}_{t-1}^{i,l}$ and the candidate state $\tilde{\mathbf{h}}_t^{i,l}$ are weighted by the update gate $\mathbf{z}_t^{i,l}$ to determine the new hidden state $\mathbf{h}_t^{i,l}$:

$$\mathbf{h}_t^{i,l} = \mathbf{z}_t^{i,l} \odot \mathbf{h}_{t-1}^{i,l} + (1 - \mathbf{z}_t^{i,l}) \odot \tilde{\mathbf{h}}_t^{i,l} \tag{5.6}$$

Finally, the hidden state $\mathbf{h}_t^{i,L}$ of the last GRU layer L is fed into two separate heads, each comprising two consecutive Bayesian layers with K and D Parametric Rectified

Linear Units (PReLUs, [75]), respectively. We used PReLU since this activation function includes a learnable parameter (slope parameter for negative input values) to overcome shortcomings of the dying ReLU problem (inactive ReLU which outputs zero for any input value) and the inconsistent predictions of LeakyReLU for negative input values. The output vectors of the two heads represent the mean $\boldsymbol{\mu}_t^{i,\text{prior}} \in \mathbb{R}^D$ and standard deviation $\boldsymbol{\sigma}_t^{i,\text{prior}} \in \mathbb{R}^D$ of the prior probability distribution modeled as Gaussian distribution $\mathcal{N}_D\left(\boldsymbol{\mu}_t^{i,\text{prior}}, \text{diag}(\boldsymbol{\sigma}_t^{i,\text{prior}})\right)$. From this prior distribution the predicted state $\hat{\mathbf{x}}_t^i$ is obtained:

$$\hat{\mathbf{x}}_t^i \sim \mathcal{N}_D\left(\boldsymbol{\mu}_t^{i,\text{prior}}, \text{diag}(\boldsymbol{\sigma}_t^{i,\text{prior}})\right) \tag{5.7}$$

Therefore, the predicted state depends only on the current state $\mathbf{x}_{t-1}^i$ of the object and the hidden states $\mathbf{h}_{t-1}^{i,1:L}$ of the GRU layers.

Given the assigned detection $\mathbf{y}_t^j \in \mathbb{R}^D$ for the next time point t, the state is updated. First, the vectors $\mathbf{y}_t^j \in \mathbb{R}^D$ and $\hat{\mathbf{x}}_t^i$ are concatenated, and passed to a FC layer with PReLUs mapping it to a vector of dimension K. Then this vector is concatenated with the hidden state $\mathbf{h}_t^{i,L}$ of the last GRU layer resulting in a vector of dimension $2K$ passed to another FC layer with K PReLUs. Finally, this K-dimensional vector is fed into two separate heads, each comprising two consecutive Bayesian layers with K and D PReLUs, respectively. The output vectors of the two heads represent the mean $\boldsymbol{\mu}_t^{i,\text{post}} \in \mathbb{R}^D$ and standard deviation $\boldsymbol{\sigma}_t^{i,\text{post}} \in \mathbb{R}^D$ of the posterior probability distribution modeled as Gaussian distribution $\mathcal{N}_D\left(\boldsymbol{\mu}_t^{i,\text{post}}, \text{diag}(\boldsymbol{\sigma}_t^{i,\text{post}})\right)$. From this posterior distribution the updated state $\mathbf{x}_t^i$ is obtained:

$$\mathbf{x}_t^i \sim \mathcal{N}_D\left(\boldsymbol{\mu}_t^{i,\text{post}}, \text{diag}(\boldsymbol{\sigma}_t^{i,\text{post}})\right) \tag{5.8}$$

The computed standard deviations $\boldsymbol{\sigma}_t^{i,\text{prior}}$ and $\boldsymbol{\sigma}_t^{i,\text{post}}$ reflect the noise in the data (e.g., detector noise and motion noise) and are denoted as *aleatoric uncertainty*.

To take into account not only aleatoric uncertainty but also *epistemic uncertainty* (model uncertainty), we employ Bayesian layers in the two heads of the prediction and update block (four network heads in total), where learnable parameters are represented by probability distributions instead of single values. Since exact Bayesian inference is intractable, we employ a variational approximation. A differentiable estimator of the lower bound that can be optimized straightforwardly using a standard stochastic gradient approach is obtained by a reparameterization of the variational lower bound, also called evidence lower bound (ELBO) [115, 119]. This reparameterization strategy enables efficient learning of the neural network parameters, without requiring computationally expensive iterative inference schemes such as Markov chain Monte Carlo (e.g., [242, 243]). In more detail, a variational posterior distribution $Q(\mathbf{W}; \theta)$ parameterized by θ is used over the learnable pa-

rameters $\mathbf{W}$. The parameters θ of the variational posterior distribution $Q(\mathbf{W};\theta)$ representing the uncertainty of $\mathbf{W}$ are learned using variational inference. This is done by maximizing the evidence lower bound objective:

$$\mathrm{ELBO}(\theta) = -\int \mathrm{d}\mathbf{W} Q(\mathbf{W};\theta) \log P(\mathbf{Y}|\mathbf{X},\mathbf{W}) + \int \mathrm{d}\mathbf{W} Q(\mathbf{W};\theta) \log \frac{Q(\mathbf{W};\theta)}{P(\mathbf{W})} \quad (5.9)$$

where $P(\mathbf{W})$ is the prior distribution over the weights that represents the uncertainty in the weights before network training. The Bayesian layer performs a regularization and imposes the constraint that the posterior distribution $Q(\mathbf{W};\theta)$ is close to the prior distribution $P(\mathbf{W})$. For $Q(\mathbf{W};\theta)$ and $P(\mathbf{W})$, we use a multivariate normal distribution. $P(\mathbf{Y}|\mathbf{X},\mathbf{W})$ is the likelihood function that specifies the variation in the labels $\mathbf{Y}$ given the network inputs $\mathbf{X}$ and the learnable parameters $\mathbf{W}$. While the second term (Kullback-Leibler divergence of $Q(\mathbf{W};\theta)$ regarding $P(\mathbf{W})$) is used for regularization and determined analytically, the first term is used to compute labels from the inputs and is approximated by drawing a single random set of weights from $Q(\mathbf{W};\theta)$. The sampling for computing the first term yields an ensemble of different network outputs. Since always a new set of weights is sampled according to $Q(\mathbf{W};\theta)$, we obtain a different prior distribution $\mathcal{N}_D\left(\boldsymbol{\mu}_t^{\mathrm{i,prior}}, \mathrm{diag}(\boldsymbol{\sigma}_t^{\mathrm{i,prior}})\right)$ and posterior distribution $\mathcal{N}_D\left(\boldsymbol{\mu}_t^{\mathrm{i,post}}, \mathrm{diag}(\boldsymbol{\sigma}_t^{\mathrm{i,post}})\right)$ for each time the four network heads are applied. Thus, the diversity in the estimated prior and posterior distributions reflects the uncertainty in the weights (model uncertainty). In our sampling strategy, for each time point t and particle i, the four network heads are applied N times yielding N prior and N posterior distributions. We used $N = 20$, which is a good compromise between computation time and tracking performance. From each of these prior and posterior distributions a predicted state and updated state is obtained, respectively. The final predicted state $\hat{\mathbf{x}}_t^i$ and updated state $\mathbf{x}_t^i$ are determined by averaging over all predicted and updated states, respectively.

The parameters of our probabilistic network are learned by minimizing the loss function:

$$\mathcal{L}_{\mathbf{x},t}^i = -\left(\underbrace{\log P\left(\tilde{\mathbf{x}}_t^i | \boldsymbol{\mu}_t^{\mathrm{i,prior}}, \boldsymbol{\sigma}_t^{\mathrm{i,prior}}\right)}_{\text{state prediction}} + \underbrace{\lambda \log P\left(\tilde{\mathbf{x}}_t^i | \boldsymbol{\mu}_t^{\mathrm{i,post}}, \boldsymbol{\sigma}_t^{\mathrm{i,post}}\right)}_{\text{state update}} \right) \quad (5.10)$$

where $\tilde{\mathbf{x}}_t^i$ is the true state of particle i at time point t. The loss function consists of two terms: The negative log-likelihood loss for state prediction and state update. The loss function represents the loss for one training sample corresponding to a single time point t of an individual particle i. We used $\lambda = 1$ in all our experiments. For training, the prediction and update block are employed as a unified network. For performing tracking, the two blocks are used sequentially.

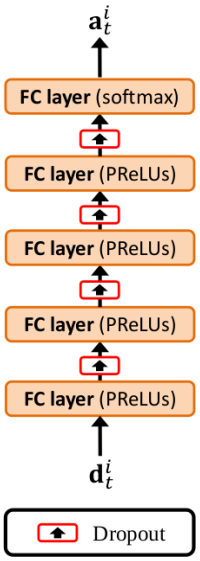


Figure 5.3 Architecture of the proposed neural network for correspondence finding.

5.2.4 Correspondence Finding

For correspondence finding, we propose a deep neural network which computes assignment probabilities and probabilities of missing detections. The network consists of four consecutive FC layers, each with R PReLUs (we used $R = 512$), followed by a fully connected linear output layer with softmax normalization. For each FC layer except the output layer, we employed dropout with a rate of 0.4 during training to avoid overfitting [97]. The network architecture is sketched in Fig. 5.3.

For each particle i and time point $t-1$, the network takes as input the vector $\mathbf{d}_t^i \in \mathbb{R}^M$, whose components are defined by $d_t^{i,j} = \|\tilde{\mathbf{x}}_t^i - \mathbf{y}_t^j\|_2$ as the Euclidean distance between the predicted state of particle i and detection j at time point t. M is the number of detections within a gate of radius r_g (we used $r_g = 20$ pixel) around the predicted position of particle i at time point t. Since the number of detections within the gate varies and our network requires a fixed input size, we consider at most the M-nearest detections (we used $M = 4$) within the gate. If there are less than M detections within the gate, we pad the vector $d_t^{i,j}$ with a placeholder (we used '-1'). The final output vector $\mathbf{a}_t^i \in [0,1]^{M+1}$ of the network contains the normalized assignment probabilities for time point t, i.e. $\sum_{j=0}^{M} a_t^{i,j} = 1$, where $a_t^{i,0}$ denotes the probability of a missing detection and $a_t^{i,j}$ represents the assignment probability between particle i and detection j (for $j = 1, \ldots, M$). The computed probabilities adapt to the local neighborhood, for example, in regions with high object density the probability of a missing detection is lower. We employ the Jonker-Volgenant shortest augmenting path algorithm [186] using the computed assignment probabilities and probabilities of missing detections as input to establish

one-to-one correspondences between all tracked particles and the set of detections obtained at time point t. Since the network computes probabilities, a threshold to determine missing detections is not needed.

To measure the deviation between the computed assignment probabilities $\mathbf{a}_t^i$ and ground truth $\tilde{\mathbf{a}}_t^i$, we use a multi-class variant of the α-balanced focal loss described in Lin et al. [93]. The loss $\mathcal{L}_{\mathbf{a},t}^i$ for one training sample (one time point t of an individual particle i) is given as:

$$\mathcal{L}_{\mathbf{a},t}^i = -\sum_{j=0}^{M} \alpha_j (1 - a_t^{i,j})^\gamma \cdot \tilde{a}_t^{i,j} \log(a_t^{i,j}) \tag{5.11}$$

where γ is a focusing parameter down-weighting easy samples and emphasizing hard samples. In all our experiments we used $\gamma = 2$. $\alpha_j \in \mathbb{R}^D$ represent weighting factors that address class imbalance in the training data and are calculated based on the class distribution in the training data.

5.2.5 Network Training

Training neural networks typically requires a vast amount of data to achieve convergence without overfitting. Since ground truth of particles in real fluorescence microscopy images is hardly available and accurate manual annotation is very tedious, our DPPT network is trained using simulated data only. In our data simulator, particle trajectories were simulated based on different motion models and the image data was generated using a Poisson noise model and automatically extracted information from the real images. For the particle dynamics we used four different motion types, namely directed motion, Brownian motion, random switching between directed motion and Brownian motion, and accelerated motion. Motion model parameters (e.g., diffusion coefficient, velocity, acceleration) of individual particles were drawn from uniform distributions. For the diffusion coefficient, we used a uniform distribution in the interval $[1, 6]$ both for the Particle Tracking Challenge data and the live cell fluorescence microscopy images. For the velocity we used an interval of $[1, 6]$ (directed motion) and $[1, 4]$ (accelerated motion and switching motion), and for the acceleration we employed $[0.2, 0.8]$. We used relatively large intervals to increase network generalization. Initial particle positions and particle appearance as well as disappearance are governed by random processes. Movement out of the field of view and out of focus was also simulated. The image noise is reflected by the SNR = $(I_{max} - I_{bg})/\sqrt{I_{max}}$ [187], where I_{max} is the maximum intensity of the particle and I_{bg} denotes the background intensity. Different to our previous work [59, 60] (Secs. 4.1 and 4.2), we use automatically extracted information from the real images to generate training images that well represent the real data. In our scheme, we detect particles in the real images by the spot-enhancing filter [147], and automatically determine the SNR, the particle size,

and the particle density based on the detected positions and local neighborhood information. We use Gaussian fitting at the detected positions to determine I_{max} and I_{bg} to compute the SNR as well as to determine the particle size σ. For I_{max}, I_{bg}, and σ, we computed the mean and standard deviation over all detected particles. The SNR of the training data was set according to the determined SNR in the real data. The appearance parameters of individual particles (I_{max}, σ) in the training data were sampled from Gaussian distributions with mean values and standard deviations determined in the real data. The particle density in the training data was set according to the determined average number of particle detections per frame in the real images. Instead, in our previous work, these parameters were drawn from uniform distributions with fixed manually defined intervals. Image size, stack size, and bit depth were set according to the meta-information of the real data. Main differences to the Particle Tracking Challenge simulator [65, 212] are that our simulator includes out-of-focus movement and accelerated motion as well as uses automatically extracted information from the real images to generate training images that well represent the real data. In addition, we exploit the computed uncertainty of our network to select the generated training data set with the best-suited motion model so that the training data well represents the real images. This was done by training our network with different motion models and then using the tracking results with the lowest epistemic uncertainty (see Sec. 5.3.1 below). Thus, we take advantage of the fact that the epistemic uncertainty of the network is lower when the particle motion in the training data agrees well with the motion in the real data. This strategy for automated motion model selection is novel and has not been used in previous work. Moreover, to generate training data we use automatic detections in the synthetically generated images to enable the network to learn the detector errors. Particle detection was performed using the spot-enhancing filter (SEF) [147] and Gaussian fitting. Then, the resulting detections were mapped to the ground truth trajectories using a nearest neighbor search with a validation gate of 5 pixel. The trajectories and the detections mapped on them were used for network training. Thus, our approach does not require prior knowledge about detection errors but learns this information from the image data. This training strategy allows generating synthetic training data that well represents the real data.

For network training, we employed the AMSGrad optimizer [86] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We used a mini-batch size of 64 and an initial learning rate of $\eta_{init} = 0.02$ for the Bayesian neural network and $\eta_{init} = 0.001$ for the network computing assignment probabilities. To avoid overfitting, early stopping was performed after convergence was reached. For training and validation we used about 102.400 training samples from synthetic image sequences. A training sample includes one time point for an individual particle. We split the data set into 80% for training and 20% for validation. The image dimensions are normalized to the range $[0, 1]$. Our model is implemented in Python 3.7 using Tensorflow 2.1.0 [244]

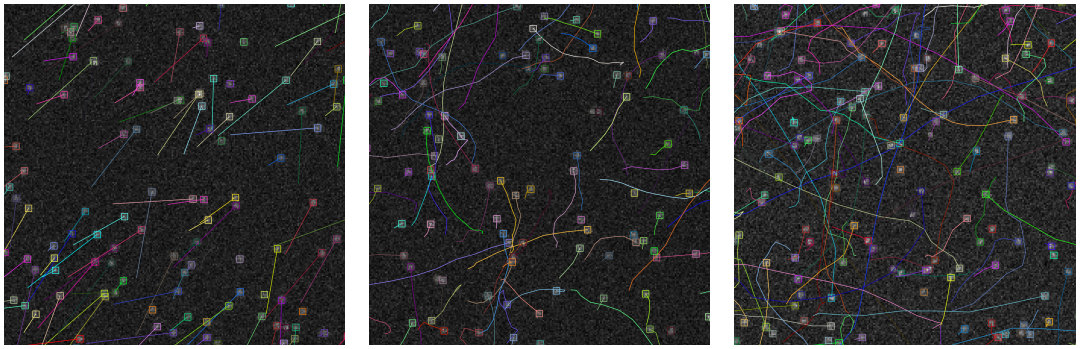
and TensorFlow Probability 0.9.0 [245]. We used a laptop with Intel(R) Core(TM) i7-7700HQ CPU, NVIDIA GeForce GTX 1050 Ti GPU, and a Linux operating system.

5.3 Experimental Results

In this section, we present experimental results of our Deep Probabilistic Particle Tracker (DPPT). First, we verify that DPPT captures both aleatoric and epistemic uncertainty. A performance evaluation is carried out using 2D as well 3D data of the Particle Tracking Challenge. In addition, we study the impact of using the uncertainty information for subsequent motion analysis. We also evaluate DPPT based on 2D and 3D real live cell fluorescence microscopy images. The five performance metrics α, β, JSC, JSC_θ, and $RMSE$ [65] are described in Sec. 3.1.

5.3.1 Evaluation and Analysis of Aleatoric and Epistemic Uncertainty

First, we evaluated uncertainty estimation by our network and verified whether both aleatoric and epistemic uncertainty are captured. To this end, we trained and tested DPPT based on generated 2D synthetic image sequences with varying conditions. The synthetic images (512×512, 8-bit) simulate real fluorescence microscopy images displaying multiple particles. The particles have an isotropic Gaussian intensity structure with Gaussian distributed standard deviation (mean $\sigma_{x,y} = 1.8$ pixel) and the images are distorted by additive Poisson noise (we used SNR = 4). For directed motion, the particle position at the next time point was determined by the current position plus the velocity vector (given by the velocity from the previous to the current time point) and an additional random change (Gaussian distribution) of both the position and the velocity vector. The changes in the position and the velocity vector were determined from a multivariate normal distribution with covariance matrix $Q = q((\sigma_{11}^2, \sigma_{12}^2), (\sigma_{12}^2, \sigma_{22}^2))$ for each image dimension as in Chenouard et al. [65]. σ_{11}^2 denotes the variance of the position noise, σ_{12}^2 the covariance of position and velocity noise, σ_{22}^2 the variance of the velocity noise, and q the motion noise influence factor. For all data we used $\sigma_{11}^2 = (1/3)$ frame^3, $\sigma_{12}^2 = (1/2)$ frame^2, and $\sigma_{22}^2 = 1$ frame. The motion noise of directed motion can be defined by the standard deviation $\sigma_m = \sqrt{q(\sigma_{11}^2 + 2\sigma_{12}^2 + \sigma_{22}^2)}$, which represents all elements of the covariance matrix Q and follows from the general formula for the variance of the sum of two random variables describing directed motion ($Var(aX+bY) = a^2Var(X)+2abCov(X,Y)+b^2Var(Y)$). For Brownian motion (random walk), the next particle position was determined by sampling from a Gaussian distribution centered at the current position and with standard deviation σ_m. For each studied condition we generated three synthetic image sequences, two of which were used for network training (training data) and one for testing (test data). Each temporal image sequence comprises 100 time points.

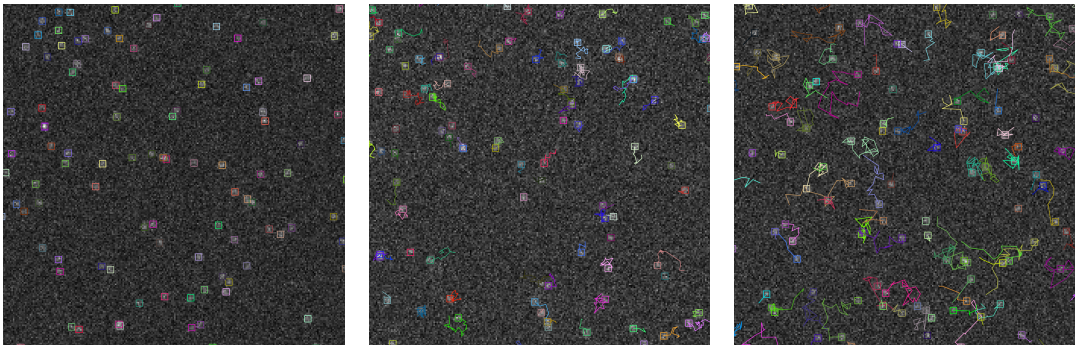


(a) $\sigma_m = 0$ pixel **(b)** $\sigma_m = 2$ pixel **(c)** $\sigma_m = 4$ pixel

Figure 5.4 Example image sections (225×225 pixels) from generated synthetic image sequences displaying particles performing directed motion with different levels of motion noise σ_m (time point $t = 12$). The image contrast was enhanced for better visibility.

Evaluation of Aleatoric and Epistemic Uncertainty

To evaluate uncertainty estimation by DPPT, we established ground truth image data which is used as benchmark. We first considered uncertainty estimation of the *state prediction* which captures the *motion noise*. We generated synthetic image sequences with seven different levels of motion noise σ_m ($\sigma_m = 0$, 0.5, 1, 2, 3, 4, and 5 pixel) for both Brownian motion and directed motion, and evaluated how well the motion noise is estimated by our network. In this experiment we did not consider detection noise (but in the following experiment). Example image sections with ground truth particle trajectories are shown in Fig. 5.4 for directed motion and in Fig. 5.5 for Brownian motion. It can be seen that the random fluctuation in the particle position (Brownian motion) and the deviation from a straight path (directed motion) increase with the strength of the motion noise σ_m. We applied DPPT to the generated image sequences and evaluated uncertainty estimation. We considered aleatoric uncertainty (σ_{alea}) and epistemic uncertainty (σ_{epi}). In Fig. 5.6 (a) and (b), ground truth and mean values of computed aleatoric and epistemic uncertainty of state prediction (over all trajectories) by DPPT are shown as a function of σ_m. It can be seen that the epistemic uncertainty is much smaller than the aleatoric uncertainty, which is expected for a well-trained network. For the image sequences of both motion models the computed aleatoric uncertainty agrees already relatively well with the ground truth. The *RMSE* between the aleatoric uncertainty and the ground truth is 0.143 pixel for Brownian motion and 0.134 pixel for directed motion. The result is further improved when considering the computed *combined* uncertainty comprising aleatoric and epistemic uncertainty (defined as the square root of the sum of the variances σ^2_{alea} and σ^2_{epi}), which yields a lower *RMSE* of 0.133 pixel for Brownian motion and 0.116 pixel for directed motion. This shows that both types of uncertainty (aleatoric and epistemic uncertainty) should

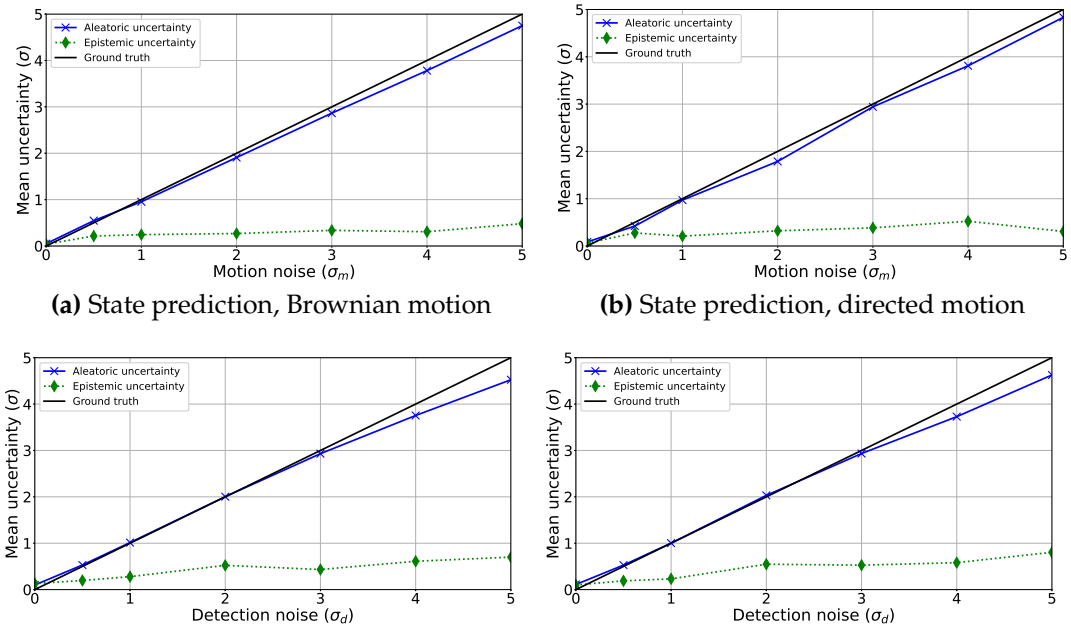


(a) $\sigma_m = 0$ pixel **(b)** $\sigma_m = 2$ pixel **(c)** $\sigma_m = 4$ pixel

Figure 5.5 Example image sections (225×225 pixels) from generated synthetic image sequences displaying particles performing Brownian motion with different levels of motion noise σ_m (time point $t = 20$). The image contrast was enhanced for better visibility.

be taken into account for motion noise estimation. The experiment demonstrates that the motion noise is well captured by our network.

In addition, we evaluated uncertainty estimation of the *state update* by DPPT. We generated synthetic image sequences using two motion models (Brownian motion, directed motion), simulated detections with different levels of additive white Gaussian noise ($\sigma_d = 0$, 0.5, 1, 2, 3, 4, and 5 pixel), and evaluated how well the *detection noise* is estimated by our network. For the motion noise we used $\sigma_m = 10$ pixel. Since the detection noise is smaller than the motion noise (i.e. the detection noise has a higher reliability) and since we considered only track points with an assigned detection, the state update mainly takes into account the detection information and the computed uncertainty represents the detection noise. In Fig. 5.6 (c) and (d), ground truth and computed mean aleatoric and epistemic uncertainty of state update (over all trajectories) by our network are shown as a function of the detection noise level. Also in this experiment the epistemic uncertainty is much smaller than the aleatoric uncertainty as expected for a well-trained network. For both motion models the computed aleatoric uncertainty agrees already relatively well with the ground truth. The *RMSE* between the aleatoric uncertainty and the ground truth is 0.208 pixel for Brownian motion and 0.181 pixel for directed motion. The result is further improved when considering the computed combined uncertainty yielding a lower *RMSE* of 0.191 pixel for Brownian motion and 0.161 pixel for directed motion. This shows that both types of uncertainty (aleatoric and epistemic uncertainty) should be taken into account for detection noise estimation. The experiment demonstrates that the detection noise is well captured by our network.



(a) State prediction, Brownian motion

(b) State prediction, directed motion

(c) State update, Brownian motion

(d) State update, directed motion

Figure 5.6 Computed mean aleatoric and epistemic uncertainty of state prediction as a function of the motion noise level for (a) Brownian motion and (b) directed motion, and of state update as a function of the detection noise level for (c) Brownian motion and (d) directed motion. Black lines indicate the ground truth.

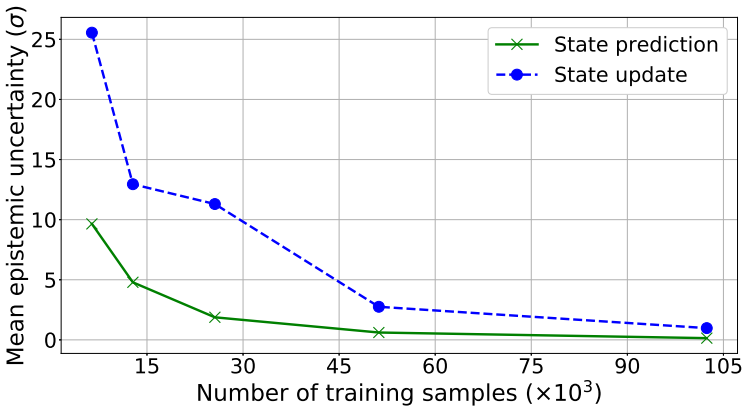


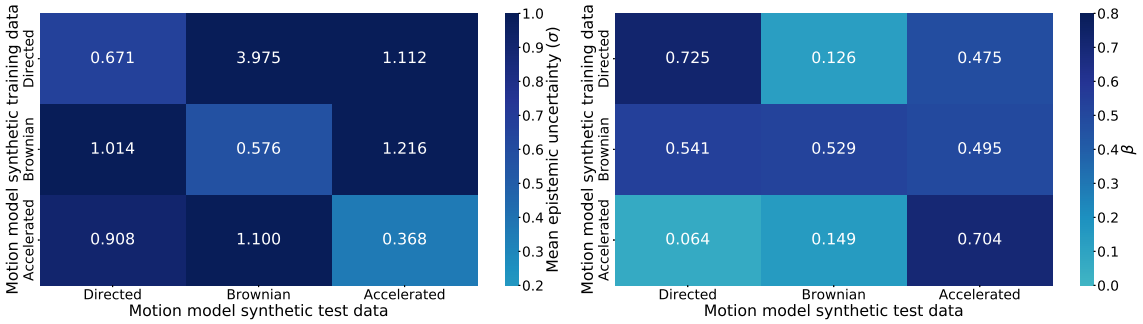
Figure 5.7 Mean epistemic uncertainty of state prediction and state update of DPPT as a function of the number of training samples.

Further analysis of epistemic uncertainty and exploitation to assess the suitability of the training data

To further analyze the *epistemic* uncertainty, we trained DPPT with different numbers of training samples (ranging from 6.4×10^3 to 102.4×10^3) using synthetic image data with directed motion and SNR = 4. One training sample represents one time point of a particle. The mean epistemic uncertainty over all particles and image dimensions as a function of the number of training samples is displayed in Fig. 5.7. As expected, the mean epistemic uncertainty decreases with the number of training samples, which demonstrates that the epistemic uncertainty is captured by our network. We also generated image data with different particle dynamics (SNR = 4). We used three different motion models, namely directed motion, Brownian motion, and accelerated motion. We trained DPPT on the training data of one of the three motion models and then applied it to the test data of all three motion models. We did this for all three models resulting in nine training and test data combinations. Fig. 5.8 (a) shows the mean epistemic uncertainty as heatmap for the nine combinations. It can be seen that the mean epistemic uncertainty is lowest when the same motion model is used for the training and test data, and otherwise the mean epistemic uncertainty is higher. This verifies that the epistemic uncertainty is captured by our network, and that the epistemic uncertainty can be exploited to assess the suitability of the training data for a considered test data. In addition, Fig. 5.8 (b) shows the tracking performance in terms of β as heatmap for the different motion models. It can be seen that selecting the training data set with the correct motion model improves the tracking performance.

5.3.2 Particle Tracking Challenge Data

We evaluated our DPPT approach based on both 2D as well as 3D image data from the Particle Tracking Challenge [65] and compared the tracking performance



(a) Epistemic uncertainty **(b)** Tracking performance

Figure 5.8 (a) Mean epistemic uncertainty and (b) tracking performance of DPPT for different motion models.

with the overall top-three methods (Methods 5, 1, and 2) described in Sec. 3.2. We also performed a comparison with our previous deep learning approaches Deep Particle Tracker (DPT) [59] (Sec. 4.1) and Deep Particle Hypotheses Tracker (DPHT) [60] (Sec. 4.2). DPT employs an LSTM-based RNN for state prediction and correspondence finding. This approach was developed for 2D data and was not applied to the 3D data. DPHT is based on a bidirectional LSTM-based RNN that exploits multiple track hypotheses for correspondence finding. Note that DPT and DPHT use deep learning for correspondence finding, while DPPT uses deep learning also for state prediction and update (as in classical Bayesian filtering) as well as for taking into account aleatoric and epistemic uncertainty. For DPPT, DPT, and DPHT, we used the same set of detections. The networks were trained based on own generated synthetic data, and we did not use the data of the Particle Tracking Challenge for network training.

We assessed the performance for different object dynamics using temporal image sequences from the 2D vesicle and 3D virus scenario of the Particle Tracking Challenge with medium (~500 particles/frame) and high (~1000 particles/frame) particle densities, and all SNR levels (SNR = 1, 2, 4, and 7). The data set (comprising 65.462 trajectories) is challenging due to complex motion in dense environments as well as image noise causing clutter and numerous detection errors. The 2D images of the vesicle scenario (512×512 pixels, 8-bit) display round particles performing Brownian (random walk) motion. The 3D image data of the virus scenario ($512 \times 512 \times 10$ voxels, 8-bit) shows spherical particles switching between Brownian and directed motion. For both scenarios, each image sequence consists of 100 images. Random processes define appearance and disappearance of individual particles compensating each other on average. The computation time for one image sequence of the vesicle scenario with high object density and SNR = 2 was about 203 seconds, using a laptop as specified above (end of Sec. 5.2.5).

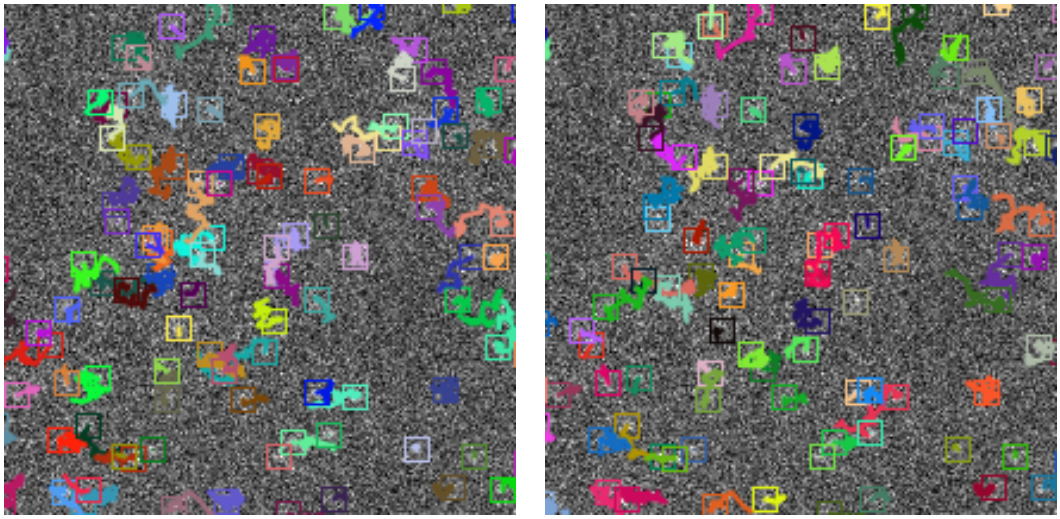
The quantitative results are presented in Tables 5.1 and 5.2. The best performance

Table 5.1 Quantitative tracking performance of different approaches for data of the vesicle scenario from the Particle Tracking Challenge. The best performance values are highlighted bold and underlined, and the second best performance values are bold.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	1	Method 5	0.162	**0.142**	**0.225**	**0.458**	2.172
		Method 1	0.027	0.026	0.034	0.300	**1.533**
		Method 2	**0.198**	0.111	0.192	0.335	2.386
		DPT	0.122	0.071	0.106	0.198	**1.701**
		DPHT	0.128	0.100	0.155	0.380	1.858
		DPPT	**0.172**	**0.139**	**0.223**	**0.407**	2.164
	2	Method 5	0.448	0.391	0.489	0.664	1.325
		Method 1	0.398	0.298	0.340	0.411	**0.840**
		Method 2	0.517	0.417	0.510	0.629	1.254
		DPT	0.450	0.356	0.413	0.577	**0.795**
		DPHT	**0.520**	**0.448**	**0.526**	**0.680**	0.874
		DPPT	**0.562**	**0.485**	**0.564**	**0.700**	1.014
	4	Method 5	0.658	0.588	0.641	0.776	0.754
		Method 1	0.687	0.609	0.652	0.767	0.607
		Method 2	0.582	0.514	0.590	0.757	0.970
		DPT	**0.695**	0.624	0.658	0.790	**0.545**
		DPHT	**0.697**	**0.638**	**0.671**	**0.804**	**0.567**
		DPPT	0.686	**0.630**	**0.669**	**0.813**	0.641
	7	Method 5	0.677	0.605	0.646	0.783	0.667
		Method 1	0.700	0.619	0.650	0.758	0.544
		Method 2	0.611	0.547	0.606	0.775	0.828
		DPT	**0.711**	0.631	0.651	**0.790**	**0.525**
		DPHT	0.705	**0.641**	**0.661**	**0.820**	**0.539**
		DPPT	**0.708**	**0.645**	**0.673**	**0.820**	0.638
High	1	Method 5	0.136	**0.120**	**0.198**	**0.460**	2.296
		Method 1	0.091	0.064	0.089	0.231	**1.859**
		Method 2	**0.163**	0.080	0.147	0.324	2.531
		DPT	0.059	0.056	0.076	0.443	**1.654**
		DPHT	0.121	0.104	0.158	**0.444**	1.984
		DPPT	**0.158**	**0.123**	**0.189**	0.391	2.055
	2	Method 5	0.353	0.295	**0.382**	**0.607**	1.484
		Method 1	0.294	0.217	0.256	0.379	1.088
		Method 2	0.356	0.249	0.331	0.515	1.582
		DPT	0.372	0.293	0.353	0.536	**1.025**
		DPHT	**0.383**	**0.311**	0.376	0.580	**1.044**
		DPPT	**0.421**	**0.341**	**0.416**	**0.601**	1.232
	4	Method 5	0.488	0.408	0.466	0.671	1.004
		Method 1	0.531	0.442	0.487	0.641	0.801
		Method 2	0.430	0.356	0.429	0.649	1.208
		DPT	**0.547**	**0.462**	**0.505**	0.680	**0.746**
		DPHT	0.531	0.458	0.500	**0.685**	**0.751**
		DPPT	**0.543**	**0.464**	**0.510**	**0.690**	0.867
	7	Method 5	0.533	0.453	0.503	0.698	0.931
		Method 1	**0.582**	0.494	0.526	0.683	**0.683**
		Method 2	0.466	0.395	0.458	0.665	1.027
		DPT	**0.590**	**0.507**	**0.535**	0.702	**0.677**
		DPHT	0.573	**0.506**	0.534	**0.717**	0.703
		DPPT	0.577	0.503	**0.541**	**0.722**	0.827

Table 5.2 Quantitative tracking performance of different approaches for data of the virus scenario from the Particle Tracking Challenge. The best performance values are highlighted bold and underlined, and the second best performance values are bold.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	1	Method 5	0.086	0.082	0.117	**0.341**	1.788
		Method 1	0.088	0.066	0.086	0.221	**1.525**
		Method 2	0.057	0.018	0.034	0.155	2.454
		DPHT	**0.114**	**0.101**	**0.143**	**0.412**	**1.551**
		DPPT	**0.150**	**0.121**	**0.174**	0.314	1.718
	2	Method 5	**0.646**	**0.590**	**0.716**	**0.802**	1.085
		Method 1	0.581	0.517	0.595	0.641	**0.776**
		Method 2	**0.655**	**0.581**	**0.714**	0.742	1.062
		DPHT	0.631	0.567	0.670	**0.777**	0.891
		DPPT	0.642	0.546	0.633	0.702	**0.855**
	4	Method 5	**0.776**	**0.748**	**0.854**	**0.891**	0.754
		Method 1	0.748	0.712	0.818	0.869	0.875
		Method 2	**0.769**	**0.725**	**0.826**	**0.880**	0.724
		DPHT	0.739	0.686	0.752	0.829	**0.576**
		DPPT	0.767	0.695	0.757	0.814	**0.534**
	7	Method 5	0.760	0.726	0.825	**0.876**	0.734
		Method 1	0.772	0.737	**0.839**	**0.891**	0.806
		Method 2	**0.786**	**0.738**	0.827	0.872	0.651
		DPHT	0.751	0.691	0.741	0.802	**0.448**
		DPPT	**0.827**	**0.775**	**0.833**	0.869	**0.449**
High	1	Method 5	0.122	0.115	0.164	**0.438**	1.768
		Method 1	0.140	0.099	0.130	0.253	**1.560**
		Method 2	0.105	0.046	0.081	0.261	2.305
		DPHT	**0.189**	**0.154**	**0.221**	**0.459**	**1.536**
		DPPT	**0.223**	**0.173**	**0.245**	0.390	1.651
	2	Method 5	**0.553**	**0.495**	**0.606**	**0.729**	1.101
		Method 1	0.528	0.460	0.539	0.610	**0.870**
		Method 2	**0.576**	**0.486**	**0.611**	**0.682**	1.140
		DPHT	0.542	0.472	0.555	0.680	0.880
		DPPT	0.547	0.469	0.548	0.645	**0.877**
	4	Method 5	0.670	**0.619**	**0.712**	**0.805**	0.796
		Method 1	0.642	0.582	0.677	0.776	0.936
		Method 2	**0.699**	**0.646**	**0.742**	**0.833**	0.792
		DPHT	**0.672**	0.607	0.667	0.766	**0.587**
		DPPT	0.670	0.582	0.641	0.727	**0.606**
	7	Method 5	0.665	0.617	**0.718**	**0.806**	0.831
		Method 1	0.665	0.612	0.702	0.794	0.881
		Method 2	**0.725**	**0.673**	**0.757**	**0.844**	0.706
		DPHT	0.706	0.645	0.694	0.781	**0.506**
		DPPT	**0.738**	**0.660**	0.710	0.782	**0.504**

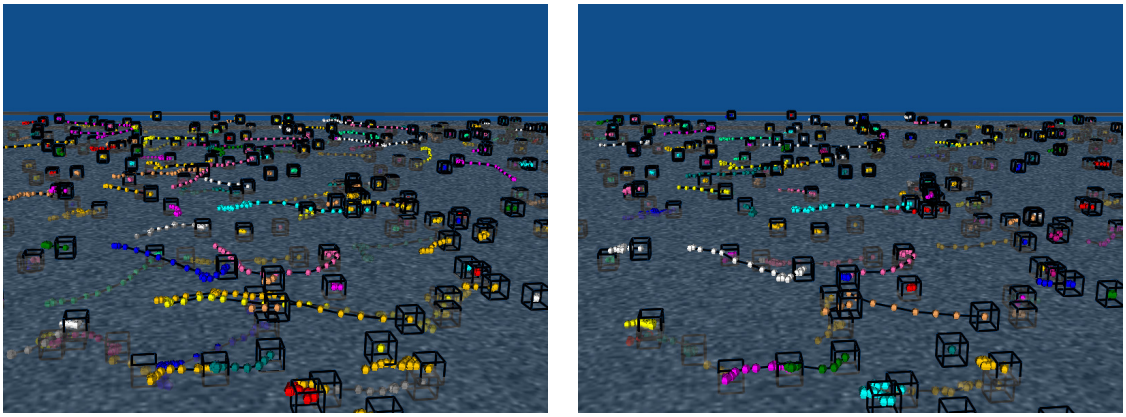


(a) Ground truth, vesicle scenario **(b)** DPPT, vesicle scenario

Figure 5.9 Ground truth and tracking results of DPPT for an image section (215×215 pixels) of the vesicle scenario with medium density and SNR = 2. The image contrast was enhanced for better visibility.

values are highlighted in bold and underlined, and bold indicates the second best performances. It can be seen that DPPT yields best or second best tracking results for almost all cases and SNR levels of the vesicle scenario. For the virus scenario, DPPT performs best or second best for the lowest (SNR = 1) and highest SNR level (SNR = 7). Overall for both scenarios, DPPT outperforms for β the other methods in eight out of 16 cases, and is second best in three cases. DPPT performs best in terms of α, JSC, JSC_θ, and $RMSE$ in six, seven, five, and two cases, respectively, and yields the second best result for α, JSC, and $RMSE$ in four cases, and for JSC_θ in two cases. For the vesicle scenario, DPPT is somewhat better for SNR = 2 compared to SNR = 1, while for the virus scenario it is vice versa. However, the performance values of our method compared to the best method in these cases are partially relatively similar. On the other hand, the results for the vesicle and virus data are not directly comparable since the data characteristics are very different (wide-field vs. confocal microscopy, Brownian vs. switching Brownian/directed motion, 2D vs. 3D data). It seems that there is no systematic dependency of the results of DPPT on the SNR level compared to previous methods. Sample trajectories obtained by DPPT are shown in Fig. 5.9 for an image section (215×215 pixels) of the vesicle scenario with medium density and SNR = 2. Fig. 5.10 shows sample tracking results of DPPT for the virus scenario with medium density and SNR = 2. In both cases the computed trajectories match well the ground truth despite the relatively low SNR level.

To demonstrate that DPPT copes well with image data that has somewhat different characteristics than the training data, we have applied our approach to



(a) Ground truth, virus scenario **(b)** DPPT, virus scenario

Figure 5.10 Ground truth and tracking results of DPPT for the virus scenario with medium density and SNR = 2. A z-slice ($z = 5$) of the original 3D data is shown. The current positions of the individual particles are indicated by cubes, intermediate positions are represented by small spheres along the trajectories. The image contrast was enhanced for better visibility.

data of the vesicle scenario with SNR = 2 and medium object density while it was trained with SNR = 4 and high object density. The obtained performance values ($\alpha = 0.547$, $\beta = 0.474$, $JSC = 0.557$, $JSC_\theta = 0.683$, $RMSE = 1.083$) differ only slightly from the results in Table 5.1 ($\alpha = 0.562$, $\beta = 0.485$, $JSC = 0.564$, $JSC_\theta = 0.700$, $RMSE = 1.014$). Thus, DPPT can cope well with somewhat different characteristics of the image data.

Impact of Object Dynamics

In addition, we studied how changes of the object dynamics affect the tracking performance of DPPT. We simulated image data of the vesicle scenario using the particle tracking benchmark generator of ICY [65, 212]. We used medium object density, SNR = 2, and different diffusion coefficients ($D_{\text{diff}} = 0, 0.5, 2, 4, 6, 8$ pixel^2/frame) yielding six image sequences with 100 time points each. We trained DPPT only once with training data from our generator, where D_{diff} of individual particles was drawn from a uniform distribution in the interval $[1, 6]$ (as in Sec. 5.2.5), and applied it to all six image sequences. In Fig. 5.11 it can be seen that our network is relatively robust and the performance for different metrics is relatively good (e.g., compare with the results of other methods in Table 5.1 for which there is $D_{\text{diff}} = 1.992$ pixel^2/frame). Also outside the training interval $[1, 6]$ reasonable results are obtained. For $RMSE$ the performance is relatively constant. For the other metrics ($\alpha, \beta, JSC, JSC_\theta$) the performance decreases with increasing D_{diff}. This is expected since the ambiguity and clutter increase with increasing object dynamics (motion strength).

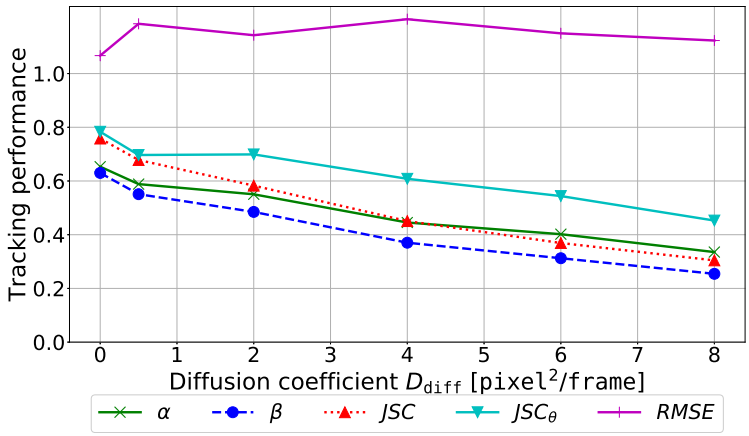


Figure 5.11 Tracking performance of DPPT as a function of the diffusion coefficient D_{diff}.

Table 5.3 Ablation study of our DPPT approach for the vesicle scenario with medium object density and SNR = 2. The best performance values are highlighted bold and underlined.

Experiment	1	2	3	4	5	6	7
Prediction block	✓			✓	✓		✓
Update block		✓		✓		✓	✓
Corresp. finding block			✓		✓	✓	✓
α	0.529	0.517	0.541	0.531	0.561	0.545	**0.562**
β	0.455	0.446	0.465	0.457	0.482	0.471	**0.485**
JSC	0.536	0.521	0.538	0.538	0.562	0.544	**0.564**
JSC_θ	0.686	0.696	0.704	0.685	0.702	**0.709**	0.700
$RMSE$	1.027	0.856	0.858	1.025	1.014	**0.845**	1.014

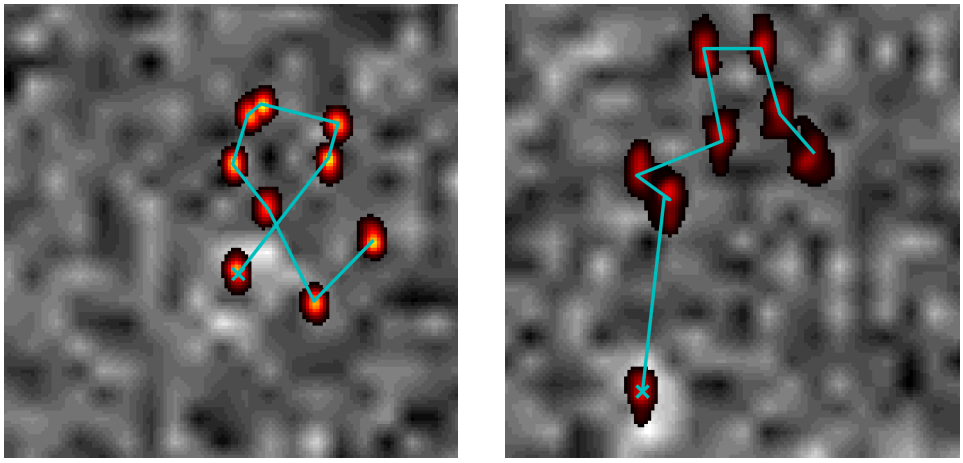
Ablation Study

We also conducted an ablation study to investigate the impact of the different components of DPPT on the tracking performance. Therefore, we disabled different components (prediction block, update block, correspondence finding block) and evaluated the performance for the vesicle scenario with medium object density and SNR = 2. In the case of disabling the prediction block, no state prediction was performed and the update step was determined based only on the assigned detections. With the update block disabled, no update step was performed. In this case, the particle states were determined based only on the assigned detections or the predictions (no detection was assigned). When the correspondence finding block was disabled, assignment probabilities as well as probabilities of missing detections were not computed. Table 5.3 shows the results. It can be seen that each component of DPPT generally improves the tracking result, and the combination yields a further improvement. The main drivers of performance are the prediction and the correspondence finding blocks. In terms of α, β, and JSC, the best results are obtained when all components of the approach are enabled. For JSC_θ similar values

are obtained for different constellations. When the prediction block is enabled, *RMSE* increases. This is expected, since with a prediction block enabled and using Brownian motion, track points are included which are based only on the prediction in the absence of an assigned detection, and because the *RMSE* of particle detection is generally much smaller than that of the prediction. We also compared our network with and without uncertainty (aleatoric and epistemic uncertainty). Using uncertainty yielded a slight improvement of the tracking performance ($\alpha = 0.562$, $\beta = 0.485$, $JSC = 0.564$, $JSC_\theta = 0.700$, $RMSE = 1.014$) compared to not using uncertainty ($\alpha = 0.560$, $\beta = 0.482$, $JSC = 0.562$, $JSC_\theta = 0.705$, $RMSE = 1.016$). A main advantage of the computed uncertainty is that the suitability of the training data can be assessed to select the data set with the best-suited motion model. If uncertainty is not exploited and a wrong motion model is selected then the tracking performance decreases (Sec. 5.3.1). For example, if training data with directed motion is used instead of Brownian motion for test data with this type of motion then the tracking performance decreases from $\beta = 0.529$ to $\beta = 0.126$ (see Fig. 5.8 b). Moreover, the computed uncertainty can be used to exclude low quality tracks to improve the accuracy of subsequent motion analysis (Sec. 5.3.2, Sec. 5.3.3). In addition, we compared GRU with LSTM in our network. GRU yielded a slightly better tracking performance ($\alpha = 0.562$, $\beta = 0.485$, $JSC = 0.564$, $JSC_\theta = 0.700$, $RMSE = 1.014$) compared to LSTM ($\alpha = 0.560$, $\beta = 0.481$, $JSC = 0.561$, $JSC_\theta = 0.700$, $RMSE = 1.015$) and the computation time was about 10% lower (due to the lower complexity of GRU compared to LSTM, see Sec. 5.2.3).

Exploiting Uncertainty Information for Motion Analysis

In addition, we studied the impact of exploiting the computed uncertainty information of DPPT for subsequent *motion analysis*. We considered the vesicle scenario with medium and high object density, and SNR = 1, and determined the diffusion coefficient D_{diff} of the particles. D_{diff} was computed by a mean-squared displacement analysis [212]. From the computed trajectories we excluded uncertain track points for which the epistemic uncertainty is high (we used a threshold of 1.0 pixel for the standard deviation). The ground truth for the data with high object density is $D_{\text{diff}} = 1.979\,\text{pixel}^2/\text{frame}$, and for medium density we have $D_{\text{diff}} = 2.011\,\text{pixel}^2/\text{frame}$ [65]. When considering all track points, the computed diffusion coefficients are $D_{\text{diff}} = 1.571\,\text{pixel}^2/\text{frame}$ with a relative error of 20.6% for the high density data, and $D_{\text{diff}} = 2.176\,\text{pixel}^2/\text{frame}$ with a relative error of 8.2% for the medium density data. Instead, when excluding uncertain track points, we obtain $D_{\text{diff}} = 1.874\,\text{pixel}^2/\text{frame}$ and $D_{\text{diff}} = 1.999\,\text{pixel}^2/\text{frame}$ with much lower relative errors of 5.3% and 0.6%, respectively. This demonstrates that the computed uncertainty information of our network can be exploited to improve the accuracy of subsequent motion analysis. Fig. 5.12 shows example tracking results of DPPT and computed uncertainty (aleatoric and epistemic uncertainty,



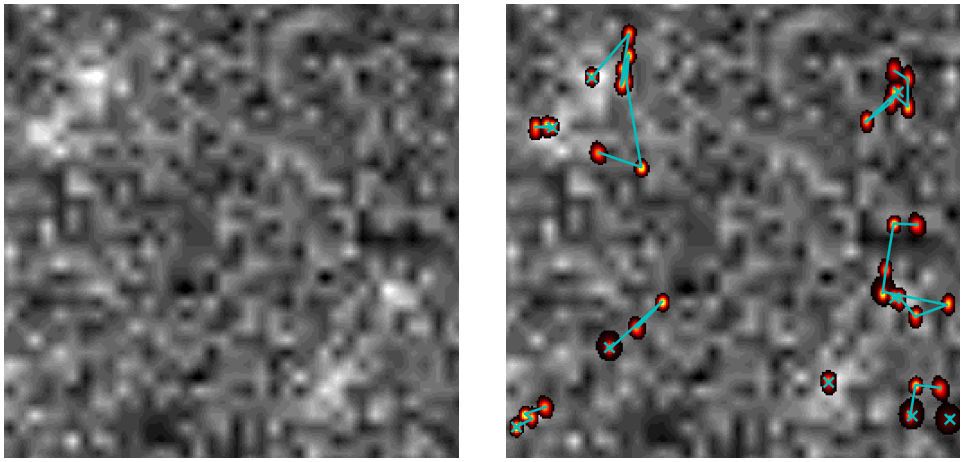
(a) Low uncertainty **(b)** High uncertainty

Figure 5.12 Tracking results of DPPT and computed uncertainty (aleatoric and epistemic uncertainty, probability density of state update, yellow: high probability values, red: low probability values) for two different trajectories (image sections of 19×19 pixels) of the vesicle scenario with medium density and SNR = 2. The current position is indicated by a cross. The original images were upscaled using interpolation by a factor 7 and the image contrast was enhanced for better visibility.

probability density of state update) for two different trajectories (image sections of 19×19 pixels). One trajectory has a low uncertainty (high probability density values) while the other trajectory has a high uncertainty (low probability density values). Fig. 5.13 shows tracking results of DPPT and computed uncertainty (aleatoric and epistemic uncertainty, probability density of state update) for a larger image section (36×36 pixels). It can be seen that the uncertainty varies between individual trajectories and track points.

Choice of Hyperparameters

We also studied the dependency of the results of DPPT on the hyperparameters. We used image data of the vesicle scenario from the Particle Tracking Challenge with medium object density and SNR = 2. Fig. 5.14 shows diagrams for the most relevant hyperparameters and for the performance metric β, which is the most comprehensive metric covering all error types (detection, localization, linking). The best parameter settings are marked by red circles, which are the values we used when applying DPPT (both for the Particle Tracking Challenge images and for the real live cell microscopy images). It can be seen that the performance of DPPT is relatively robust within certain ranges of the hyperparameters. The performance decreases strongly when the network gets too complex (e.g., $L > 3$, $R > 70$). Since for some hyperparameters the performance is similar for reduced values, we also tested our network with reduced complexity (less layers and units,



(a) Original image **(b)** Tracking results and uncertainty

Figure 5.13 Tracking results of DPPT and computed uncertainty (aleatoric and epistemic uncertainty, probability density of state update, yellow: high probability values, red: low probability values) for an image section (36×36 pixels) of the vesicle scenario with medium density and SNR = 2. The current position is indicated by a cross. The original images were upscaled using interpolation by a factor 7 and the image contrast was enhanced for better visibility.

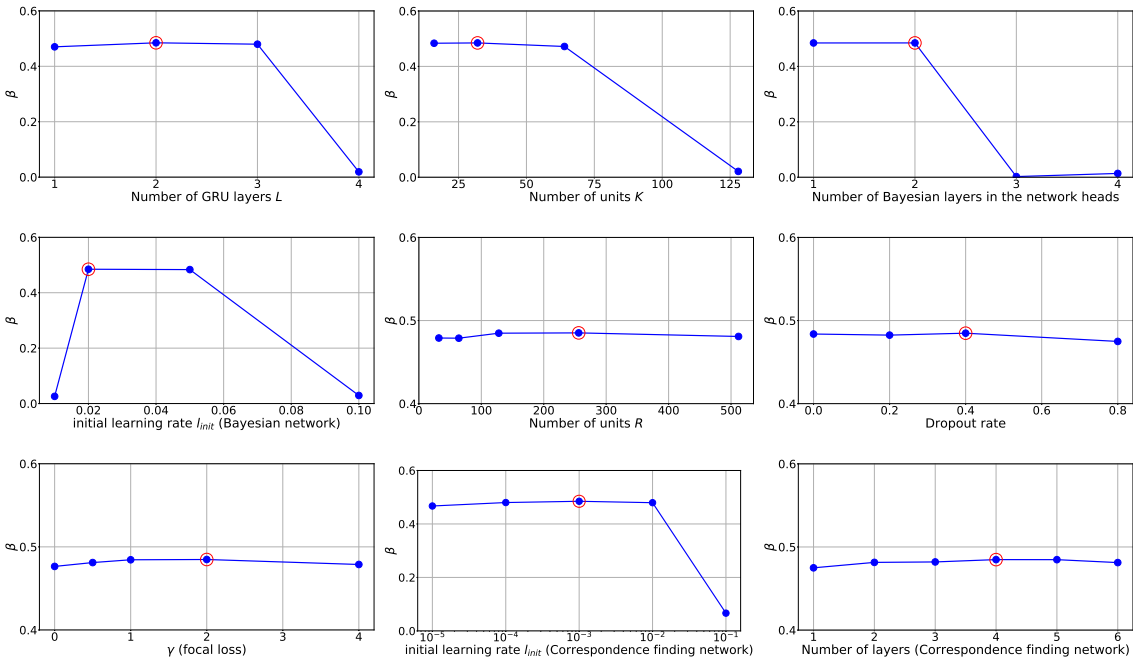


Figure 5.14 Tracking performance of DPPT as a function of hyperparameters. Red circles indicate the best performance.

Table 5.4 Overview of the real fluorescence microscopy sequences.

Sequence	Image dimension	Object type	No. of trajectories
Seq. 1	2D	HCV NS5A protein	75
Seq. 2	2D	HCV NS5A protein	55
Seq. 3	2D	HCV associated ApoE protein	90
Seq. 4	3D	Chromatin structures	66
Seq. 5	3D	Chromatin structures	71
Seq. 6	3D	Chromatin structures	35
Seq. 7	3D	Chromatin structures	60

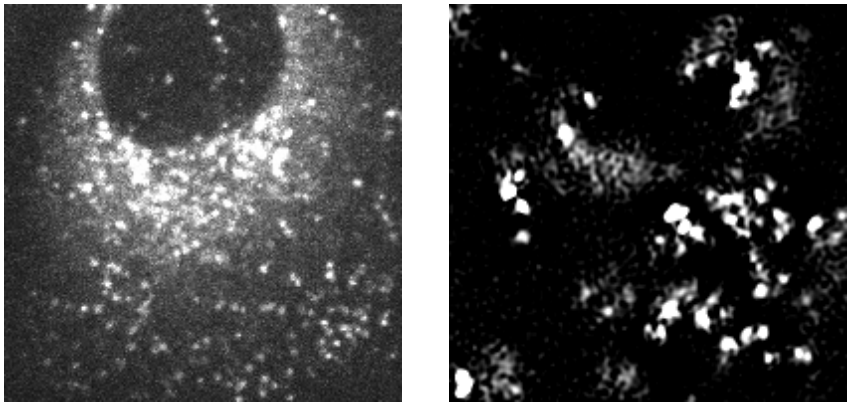
e.g., $L = 1$, $K = 10$, Bayesian layers=1, $R = 10$). However, then the performance was reduced, and more importantly, the epistemic uncertainty was too small and not well represented so that motion model selection for the training data did not work well.

5.3.3 Real Live Cell Fluorescence Microscopy Images

Tracking Performance

We also performed a quantitative evaluation of DPPT based on real live cell fluorescence microscopy image data of the hepatitis C virus (HCV) nonstructural protein 5A (NS5A), the HCV associated cellular Apolipoprotein E (ApoE), and chromatin structures (labeled during DNA replication). We considered three 2D image sequences (30 time points, 512×512 pixels, pixel size 0.22×0.22 μm^2, 16-bit) of proteins in HCV proteins expressing Huh7/LunetCD81H cells denoted by Seq. 1 to Seq. 3. Seq. 1 and Seq. 2 display the HCV protein NS5A and Seq. 3 shows the ApoE protein. The images were acquired using a confocal spinning disk microscope and an EMCCD camera [246]. This data set is challenging due to clustering of particles, clutter, out of focus movement, and relatively low SNR. We also used four 3D image sequences (11 time points, $512 \times 512 \times 5$ voxels, voxel size $0.0410 \times 0.0410 \times 0.125$ μm^3, 16-bit) of chromatin structures (labeled by nucleotide incorporation during DNA replication) in HeLa Kyoto cells denoted by Seq. 4 to Seq. 7. The data was acquired by super-resolution 3D structured illumination microscopy (3D-SIM) using a sCMOS camera [247]. Main challenges of this data set are clustering of objects and decreasing SNR over time due to photobleaching. Between 35 and 90 ground truth trajectories for difficult regions were manually annotated in each of the seven image sequences using the ImageJ plugin MTrackJ [222]. Table 5.4 gives an overview of the seven image sequences. Example image sections (200×200 pixels) for Seq. 2 and for a z-slice ($z = 3$) of Seq. 5 are shown in Fig. 5.15.

We compared the performance of DPPT with the ParticleTracker (PT) [187], the Kalman filter based approach (KF) implemented in the ImageJ plugin TrackMate [189], and the multiple hypothesis tracking (MHT) approach [193] implemented in Icy [212]. The methods are described in Sec. 3.2. For PT, KF, and MHT, we tested



(a) Seq. 2 (2D, HCV NS5A)

(b) Seq. 5 (3D, chromatin structures, z-slice $z = 3$)

Figure 5.15 Example image sections (200×200 pixels) of real live cell fluorescence microscopy data. The image contrast was enhanced for better visibility.

several parameter settings and used the settings yielding the best tracking results. We also compared DPPT with our previous deep learning approaches DPT (for 2D images, Sec. 4.1) and DPHT (for 2D and 3D images, Sec. 4.2). Note that DPPT, DPT, and DPHT were trained based on synthetic data only and then applied to real data. This means that tedious manual annotation of the real data was not required for network training.

Tracking results for all approaches for the 2D and 3D image data are presented in Tables 5.5 and 5.6, respectively. It can be seen that DPPT performs best for most image sequences. For β, DPPT outperforms the other approaches in five out of seven cases, and is second best in the remaining two cases. In terms of α and JSC, DPPT yields the best result in five out of seven cases, and the second best result in one case. In terms of JSC_θ, DPPT performs best in three cases and second best in four cases. For *RMSE*, DPPT yields the second best result in three cases. Fig. 5.16 shows ground truth and tracking results for a trajectory of image sequence Seq. 1 (HCV) with complex motion (19×19 pixels section). It can be seen that the previous approaches (PT, KF, MHT, DPHT) yield broken trajectories, whereas DPPT yields a trajectory without gaps which agrees well with the ground truth.

Exploiting Uncertainty Information for Motion Analysis

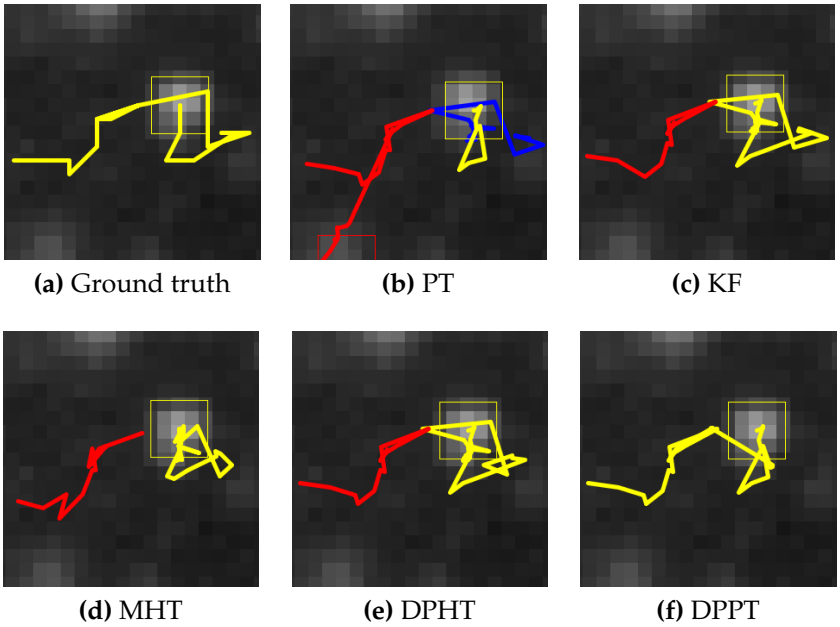
In addition, we studied the impact of exploiting the computed uncertainty of DPPT for subsequent *motion analysis* in real live cell microscopy images. For Seq. 1 and Seq. 2, we excluded uncertain track points for which the epistemic uncertainty is high (we used a threshold of 0.5 pixel for the standard deviation) and determined the diffusion coefficient D_{diff} of the particles (as in Sec. 5.3.2 for the Particle Tracking Challenge data). The ground truth for Seq. 1 and Seq. 2 is $D_{\text{diff}} = 0.199$ pixel^2/frame and $D_{\text{diff}} = 0.769$ pixel^2/frame, respectively. When considering all track points,

Table 5.5 Performance values of different tracking approaches for 2D real fluorescence microscopy image sequences Seq. 1 to 3 displaying HCV NS5A and HCV associated ApoE proteins. The best performance values are highlighted bold and underlined, and the second best performance values are bold. The mean values for all approaches are also shown.

Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. 1 (HCV NS5A)	PT	0.545	0.506	0.635	0.644	1.253
	KF	0.528	0.505	0.586	0.659	0.979
	MHT	0.560	0.527	0.648	0.693	1.210
	DPT	0.606	0.559	0.649	0.644	**0.904**
	DPHT	**0.610**	**0.575**	**0.676**	**0.697**	1.016
	DPPT	**0.648**	**0.611**	**0.713**	**0.711**	**0.975**
Seq. 2 (HCV NS5A)	PT	0.590	0.496	0.629	0.557	**1.064**
	KF	0.559	0.481	0.564	0.550	1.088
	MHT	0.540	0.480	0.588	0.611	1.237
	DPT	**0.633**	0.540	0.632	0.605	**1.008**
	DPHT	0.619	**0.556**	**0.652**	**0.622**	1.117
	DPPT	**0.639**	**0.567**	**0.658**	**0.630**	1.069
Seq. 3 (HCV associated ApoE)	PT	0.324	0.306	0.382	0.414	1.255
	KF	0.525	0.432	0.499	0.529	1.137
	MHT	0.436	0.422	0.507	**0.588**	1.299
	DPT	0.614	0.499	0.586	0.527	**1.068**
	DPHT	**0.626**	**0.510**	**0.614**	0.538	**1.052**
	DPPT	**0.640**	**0.516**	**0.619**	**0.543**	1.192
Mean values	PT	0.487	0.436	0.549	0.538	1.191
	KF	0.537	0.473	0.550	0.579	1.068
	MHT	0.512	0.476	0.581	**0.631**	1.249
	DPT	0.617	0.533	0.623	0.592	**0.993**
	DPHT	**0.618**	**0.547**	**0.647**	0.619	**1.062**
	DPPT	**0.642**	**0.565**	**0.663**	**0.628**	1.079

Table 5.6 Performance values of different tracking approaches for 3D real fluorescence microscopy image sequences Seq. 4 to 7 displaying chromatin structures. The best performance values are highlighted bold and underlined, and the second best performance values are bold. The mean values for all approaches are also shown.

Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. 4	PT	0.305	0.285	0.415	**0.500**	**1.823**
	KF	0.336	0.282	0.408	0.489	2.048
	MHT	**0.398**	0.292	0.420	0.462	2.014
	DPHT	**0.353**	**0.337**	**0.502**	**0.551**	1.992
	DPPT	0.331	**0.315**	**0.444**	**0.500**	**1.944**
Seq. 5	PT	0.334	0.255	**0.493**	0.523	2.454
	KF	**0.381**	0.290	**0.466**	0.528	2.112
	MHT	0.337	0.286	0.443	0.529	2.189
	DPHT	0.365	**0.302**	0.436	**0.565**	**1.902**
	DPPT	**0.384**	**0.326**	0.463	**0.593**	**1.956**
Seq. 6	PT	0.443	0.415	**0.657**	0.683	1.962
	KF	**0.540**	0.446	0.587	0.673	**1.582**
	MHT	0.407	0.390	0.572	0.590	2.044
	DPHT	0.493	**0.485**	0.641	**0.806**	**1.675**
	DPPT	**0.590**	**0.510**	**0.701**	**0.714**	1.696
Seq. 7	PT	0.439	0.400	0.587	0.649	1.930
	KF	0.415	0.330	0.534	0.553	2.272
	MHT	**0.453**	**0.417**	**0.604**	0.647	**1.926**
	DPHT	0.424	0.403	0.574	**0.708**	**1.904**
	DPPT	**0.447**	**0.415**	**0.624**	**0.701**	2.058
Mean values	PT	0.380	0.339	**0.538**	0.589	2.042
	KF	**0.418**	0.337	0.499	0.561	2.004
	MHT	0.399	0.346	0.510	0.557	2.043
	DPHT	0.409	**0.382**	**0.538**	**0.657**	**1.869**
	DPPT	**0.438**	**0.391**	**0.558**	**0.627**	**1.913**



(a) Ground truth **(b)** PT **(c)** KF

(d) MHT **(e)** DPHT **(f)** DPPT

Figure 5.16 Ground truth and tracking results of different approaches for a 19 × 19 pixels section of image sequence Seq. 1 (HCV NS5A). The image contrast was enhanced for better visibility.

the computed diffusion coefficients are $D_{\text{diff}} = 0.189\ \text{pixel}^2/\text{frame}$ with a relative error of 5.0% for Seq. 1, and $D_{\text{diff}} = 0.440\ \text{pixel}^2/\text{frame}$ with a relative error of 42.8% for Seq. 2. Instead, when excluding uncertain track points, we obtain $D_{\text{diff}} = 0.190\ \text{pixel}^2/\text{frame}$ for Seq. 1, and $D_{\text{diff}} = 0.816\ \text{pixel}^2/\text{frame}$ for Seq. 2 with lower relative errors of 4.5% and 6.1%, respectively. This confirms the results for the Particle Tracking Challenge data (Sec. 5.3.2), and demonstrates for real live cell microscopy images that the computed uncertainty information of our network can be exploited to improve the accuracy of subsequent motion analysis.

5.4 Conclusion

We have introduced a novel probabilistic deep learning approach for tracking multiple particles in fluorescence microscopy image sequences. The proposed Deep Probabilistic Particle Tracker (DPPT) is based on a recurrent neural network which mimics classical Bayesian filtering. Compared to previous methods for particle tracking, our approach takes into account uncertainty, both aleatoric (intrinsic noise in the data) and epistemic uncertainty (uncertainty in network weights). The network exploits short and long-term temporal dependencies in the object dynamics to predict the state at the next time point, and uses assigned detections to update the predicted state. For correspondence finding, we have introduced a neural network that computes assignment probabilities jointly across multiple

detections as well as determines probabilities for missing detections. Network training requires only simulated data and therefore tedious manual annotation of ground truth is not necessary. We proposed a novel scheme to generate synthetic training data using automatically extracted information from the real images. This enables simulating a large amount of training data that represent well the images in an application.

We verified that both types of uncertainty (aleatoric and epistemic uncertainty) are captured by the proposed Bayesian neural network and carried out an evaluation of uncertainty estimation. An advantage of our network is that information about the reliability of the extracted trajectories is determined. We demonstrated that the computed uncertainty can be exploited to increase the accuracy of subsequent motion analysis by excluding unreliable track points. In addition, we demonstrated that the uncertainty can be exploited to assess the suitability of the training data and to select the training data set with the best-suited motion model so that the training data better represents the real data in an application. The uncertainty could also be used to calibrate a microscopy experiment by adjusting acquisition parameters. We conducted a quantitative evaluation of the tracking performance using 2D and 3D image data of the Particle Tracking Challenge as well as 2D and 3D real live cell fluorescence microscopy image sequences. A comparison with previous methods showed that DPPT yields state-of-the-art or improved results.

Chapter 6

Neural Network for Combined Particle Tracking and Colocalization Analysis

In this chapter, we introduce a novel deep learning method for combined particle tracking and colocalization analysis. The work has been published in [62].

6.1 Introduction

Quantifying protein dynamics and protein-protein interactions from multi-channel fluorescence microscopy images is important to study biological processes. Thus, automatic approaches for particle tracking and colocalization analysis are needed.

In previous work on fluorescent particle tracking, *classical* deterministic and probabilistic methods were introduced. Probabilistic approaches have the advantage that they take into account spatial and temporal uncertainties by Bayesian filtering (e.g., [151, 193, 194]). Recently, *deep learning* methods for particle tracking have been introduced (e.g., [22, 60]). These methods use recurrent neural networks (RNNs) to represent information about object motion for correspondence finding. However, colocalization information is not taken into account for tracking, and a colocalization analysis is not performed.

In previous work on colocalization analysis in multi-channel microscopy data, *classical* pixel-based, object-based, and track-based methods have been proposed. Pixel-based approaches use intensity correlation (e.g., [248]), however, they are relatively sensitive to noise. Object-based approaches incorporate spatial information to improve the robustness to image noise (e.g., [249]). These methods do not include temporal information, which can lead to false positives. Track-based approaches exploit spatial as well as temporal information, and generally yield better results (e.g., [250, 251]). A disadvantage of these classical colocalization methods is that several parameters need to be tuned manually.

In this contribution, we introduce a novel *deep learning* approach for combined particle tracking and colocalization analysis in two-channel fluorescence microscopy

images. Short- and long-term temporal dependencies of object motion as well as image intensities from both channels of two-channel microscopy data are exploited by a convolutional Long Short-Term Memory (ConvLSTM) network [252]. Assignment probabilities are computed jointly across multiple detections and also probabilities of missing detections are determined without requiring handcrafted features. In addition, colocalization probabilities are computed and colocalization information is used to improve tracking. To our best knowledge we are the first to introduce such a deep learning approach. Previous deep learning methods for particle tracking use only one image channel for tracking, do not perform colocalization analysis, and do not use image intensities for correspondence finding (e.g., [22, 60]). The proposed network is trained based on synthetic data and thus tedious manual labeling is not required. We performed a quantitative evaluation using synthetic two-channel image sequences as well as real two-channel fluorescence microscopy data of hepatitis C virus (HCV) proteins. It turned out that our approach outperforms previous methods.

6.2 Method

The proposed approach, denoted as Deep Particle Tracker and Colocalization Analyzer (DPTCA), is based on a convolutional LSTM neural network that determines assignment and colocalization probabilities for combined particle tracking and colocalization analysis. The Jonker-Volgenant shortest augmenting path algorithm is employed to establish one-to-one correspondences using the computed assignment probabilities as well as the probabilities of missing detections. Disappearing objects are identified based on the number of subsequent frames with missing detections, and appearing objects are identified based on unassigned detections.

6.2.1 Network Architecture

In our DPTCA approach we use a neural network consisting of ConvLSTM layers [252], convolutional layers, and fully-connected (FC) layers. The network architecture is sketched in Fig. 6.1 and described below.

Let $\mathbf{x}_{t-1}^{i} \in \mathbb{R}^{P \times P \times (C+1)}$ represent a local patch with $P \times P$ pixels (we used $P = 30$) of an image (C channels) at time point $t-1$, and a binary mask representing the position of object i. The $\text{ConvLSTM}_{32}^{3\times3}$ layer (32 filters, 3×3 kernel size) takes as input sequence the patches $\mathbf{x}_{t-T}^{i}, \ldots, \mathbf{x}_{t-1}^{i}$ at T previous time points. A patch at time point t and M binary masks indicating the positions of the detections within the patch are represented by $\mathbf{y}_{t}^{i} \in \mathbb{R}^{P \times P \times (C+M)}$, where M denotes the number of detections (we used $M = 4$). If there are more than M detections, only the M nearest detections are considered. $\mathbf{y}_{t}^{i}$ is passed through a convolutional layer ($\text{Conv}_{32}^{3\times3}$, 32 filters, 3×3 kernel size) and then concatenated with the $\text{ConvLSTM}_{32}^{3\times3}$ output. The resulting tensor is fed into a stack of four convolutional layers with 32,

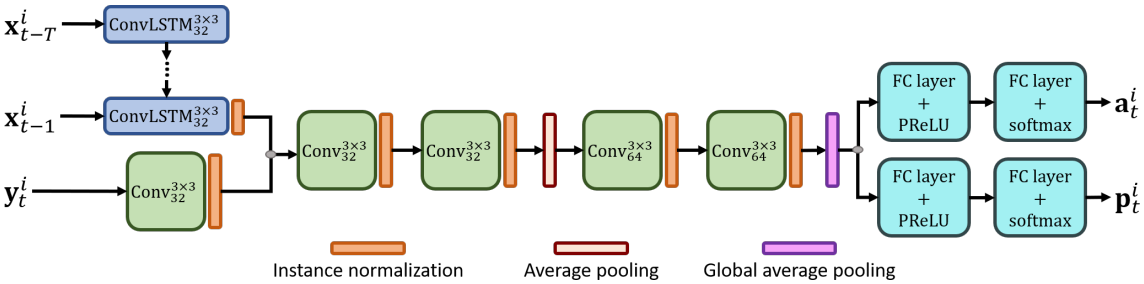


Figure 6.1 Network architecture of the proposed DPTCA.

32, 64, and 64 filters, respectively. We employ instance normalization after each convolutional layer to prevent instance-specific mean and covariance shift and improve network training. We use Parametric Rectified Linear Units (PReLU) to cope with the dying ReLU problem and inconsistent predictions of LeakyReLU for negative input values. Spatial pooling is carried out by average pooling (3×3, stride of 2) and global average pooling. The resulting vector of dimension 64 is fed into two separate output paths, each comprising one FC layer with 16 PReLUs followed by a fully-connected output layer with softmax normalization. The output vector $\mathbf{a}^i_t \in [0,1]^{M+1}$ of the first output path represents assignment probabilities as well as the probability of a missing detection for time point t, i.e. $\sum_{j=0}^{M} a_t^{i,j} = 1$, where $a_t^{i,j}$ is the assignment probability between object i and detection j ($j = 1, \ldots, M$), and $a_t^{i,0}$ the probability of a missing detection. The output vector $\mathbf{p}^i_t = (p_t^{i,1}, p_t^{i,2})$ of the second output path represents normalized colocalization probabilities. $p_t^{i,1}$ denotes the probability that object i is colocalized, whereas $p_t^{i,2}$ represents the probability that object i is not colocalized. Note that in the first pass, $\mathbf{y}^i_t$ contains only detections from the same channel in which object i is located. If $p_t^{i,1} > p_t^{i,2}$, then in a second pass, $\mathbf{y}^i_t$ contains detections from the other channel to determine with which object in the other channel, object i is colocalized. If a colocalized object i is not assigned to a detection from one channel, the detection from the other channel is exploited for tracking.

Our neural network is trained by minimizing the loss function $\mathcal{L}$. For one time point t of an object (particle) i, the loss is defined by:

$$\mathcal{L}^i_t = \mathcal{L}^i_{\mathbf{a},t} + \lambda \mathcal{L}^i_{\mathbf{p},t} \tag{6.1}$$

where the first term is a multi-class variant of the focal loss [93] measuring the deviation between the computed assignment probabilities $\mathbf{a}^i_t$ and ground truth $\mathbf{a}^i_t$:

$$\mathcal{L}^i_{\mathbf{a},t} = -\sum_{j=0}^{M} (1 - a_t^{i,j})^{\gamma} \cdot \tilde{a}_t^{i,j} \log(a_t^{i,j}) \tag{6.2}$$

with focusing parameter γ. The second term is the multi-class focal loss for the computed colocalization probabilities $\mathbf{p}_t^i$ using the ground truth $\mathbf{p}_t^i$:

$$\mathcal{L}_{\mathbf{p},t}^i = -\sum_{l=1}^{2}(1-p_t^{i,l})^\gamma \cdot \tilde{p}_t^{i,l}\log(p_t^{i,l}) \tag{6.3}$$

We used $\lambda = 1$ and $\gamma = 2$ in our experiments.

6.2.2 Network Training

Since manual annotation of biological particles is tedious, we use synthetic data for network training. Particle trajectories were simulated using multiple motion models (e.g., Brownian, directed), and two-channel microscopy images were generated using a Poisson noise model. Particles in the two channels randomly colocalize if they are less than 5 pixels apart (which is in the order of the Rayleigh distance). Appearance and disappearance of particles is defined by random processes. For training, we employed the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate was set to 0.001 and a mini-batch size of 12 was used. To avoid overfitting, we applied early stopping. The generated data set comprises about 100,000 particle trajectories. The data is split into 75% for training and 25% for validation.

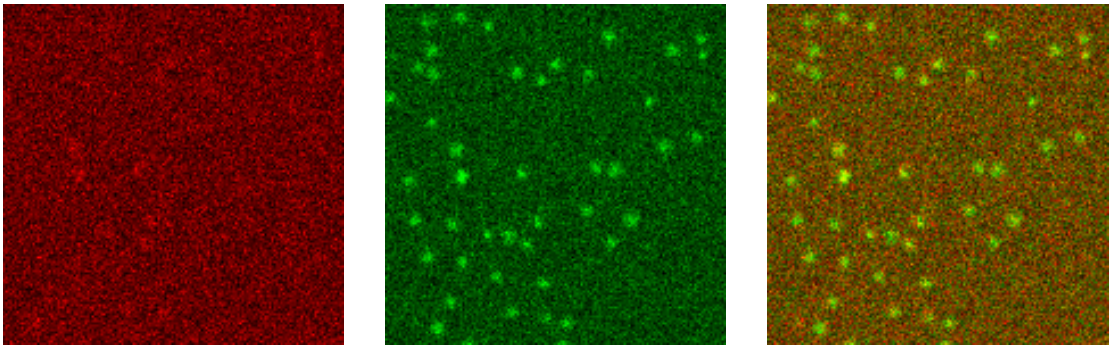
6.3 Experimental Results

6.3.1 Experimental Setup

We used the five metrics α, β, JSC, JSC_θ, and $RMSE$ [65] described in Sec. 3.1 to evaluate the tracking performance. We compared the performance of the proposed method (DPTCA) with state-of-the-art particle tracking methods: Particle Tracker (PT) [187], Kalman filter (KF) [189], multiple hypothesis tracking (MHT) [193], and Deep Particle Hypotheses Tracker (DPHT) [60]. PT, KF, and MHT are described in Sec. 3.2 and DPHT is introduced in Sec. 4.2. For DPTCA and DPHT, we used SEF and Gaussian fitting for particle detection.

6.3.2 Synthetic Image Sequences

We first evaluated DPTCA using three two-channel synthetic image sequences. The images simulate real two-channel fluorescence microscopy images and include about 500 particles per channel in each frame (512×512 pixels). The particles have an isotropic Gaussian intensity profile and perform Brownian motion. Channel 2 has a high SNR level (SNR = 5), while channel 1 has a low SNR level (SNR = 1, 2, 3). The sequences are denoted by Seq. 1 to Seq. 3. Colocalization of particles in the



(a) Low SNR channel 1 **(b)** High SNR channel 2 **(c)** Overlay

Figure 6.2 Section of synthetic image sequence Seq. 1.

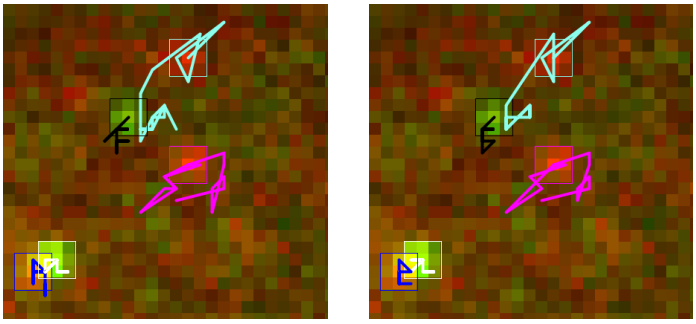
Table 6.1 Performance for the low SNR channel of the synthetic image sequences.

Sequence	Method	α	β	JSC	JSC_θ	$RMSE$
Seq. 1 (channel 1, SNR = 1)	PT	0.057	0.039	0.060	0.143	1.664
	KF	0.079	0.055	0.078	0.181	1.590
	MHT	0.106	0.104	0.157	**0.302**	2.182
	DPHT	0.146	0.108	0.155	**0.302**	1.562
	DPTCA	**0.300**	**0.208**	**0.273**	0.286	**1.528**
Seq. 2 (channel 1, SNR = 2)	PT	0.281	0.156	0.220	0.136	1.250
	KF	0.287	0.199	0.244	0.342	1.050
	MHT	0.397	0.312	0.375	0.458	1.342
	DPHT	0.473	0.397	0.458	0.516	**0.987**
	DPTCA	**0.555**	**0.434**	**0.495**	**0.557**	1.092
Seq. 3 (channel 1, SNR = 3)	PT	0.451	0.359	0.427	0.481	1.047
	KF	0.526	0.427	0.466	0.563	0.809
	MHT	0.453	0.361	0.411	0.478	1.175
	DPHT	0.569	0.476	0.513	**0.593**	**0.759**
	DPTCA	**0.620**	**0.489**	**0.535**	0.586	0.830

two channels is defined for a distance smaller than 5 pixels. The data includes random appearance and disappearance of particles. An example image section (150×150 pixels) for Seq. 1 is shown in Fig. 6.2. The data set is challenging due to conflicting correspondences, complex motion, and low SNR. The quantitative tracking results are provided in Table 6.1. Bold highlights the best performance. We show results for the low SNR channel 1 since this channel is most challenging. It can be seen that DPTCA outperforms the other methods. In terms of α, β, and JSC, DPTCA yields the best result for all three sequences. For JSC_θ, DPTCA performs best for one sequence, and second best for the other two sequences. In terms of $RMSE$, DPTCA outperforms the other methods for one sequence.

Table 6.2 Performance for real microscopy image data (HCV).

Channel	Method	α	β	JSC	JSC_θ	$RMSE$
1 (NS5A)	PT	0.470	0.441	0.537	0.646	1.154
	KF	0.585	0.428	0.509	0.411	1.206
	MHT	0.536	0.521	0.613	0.723	1.230
	DPHT	0.597	0.563	**0.630**	0.720	**1.086**
	DPTCA	**0.605**	**0.571**	**0.630**	**0.753**	1.139
2 (E2)	PT	0.615	0.609	0.723	0.718	0.840
	KF	0.725	0.619	0.731	0.552	0.811
	MHT	0.814	0.689	0.777	0.660	0.729
	DPHT	0.811	0.738	0.784	**0.825**	**0.596**
	DPTCA	**0.860**	**0.763**	**0.812**	**0.825**	0.807



(a) Ground truth **(b)** DPTCA

Figure 6.3 Ground truth and tracking results of DPTCA for an image section of the HCV data (overlay of both channels).

6.3.3 Real Two-Channel Fluorescence Microscopy Images

We also evaluated DPTCA based on real two-channel fluorescence microscopy images displaying the nonstructural protein 5A (NS5A, channel 1) and the structural protein E2 (channel 2) of hepatitis C virus (HCV). Colocalization of the two proteins is essential for virus particle assembly. We considered a temporal image sequence acquired with a confocal microscope (30 time points, 512×512 pixels, pixel size 0.22×0.22 μm^2). The data is challenging due to relatively low SNR, high object density, and clustering particles. To quantify the performance, 126 ground truth trajectories (91 for channel 1, 35 for channel 2) were manually annotated for a difficult image region. The result is shown in Table 6.2. Bold highlights the best performance. It can be seen that DPTCA yields better results than previous methods. For α, β, JSC, and JSC_θ, DPTCA performs best for both channels. Example tracking results of DPTCA for an image section of the HCV data are displayed in Fig. 6.3. It can be seen that the computed trajectories agree well with the ground truth.

6.4 Conclusion

We presented a novel deep learning approach for combined particle tracking and colocalization analysis in two-channel fluorescence microscopy images. The approach is based on a ConvLSTM network and exploits colocalization information to improve tracking. Manual annotation is not required for network training. We evaluated the performance of DPTCA using synthetic image data and real two-channel fluorescence microscopy images. It turned out that our approach outperforms previous methods.

Chapter 7

Deep Learning for Particle Detection and Tracking

In this chapter, we propose a method for particle detection based on a convolutional neural network that performs density map regression. In addition, a deep learning method for probabilistic detection and tracking of particles in fluorescence microscopy images is introduced.

7.1 Convolutional Neural Network for 3D Particle Detection

In this section, we present a convolutional neural network for 3D particle detection in 3D fluorescence microscopy images via density map regression. The work has been published in [63].

7.1.1 Introduction

In previous work on fluorescent particle detection, classical methods were introduced (e.g., [56]) such as a wavelet-based detector [143], the spot-enhancing filter (SEF, [147]), a HDome transform-based detector [201], and adaptive thresholding with autoselected scale [148]. However, these methods are generally based on a predefined, relatively simple appearance model (e.g., Gaussian function), which does not necessarily hold. Recently, convolutional neural networks (CNNs) for particle detection have been introduced which can cope with more general appearance structures and show promising results (e.g., [153, 155, 156, 158]). In [153, 155, 156], image-to-image mapping is performed based on pixel-wise binary classification, where each particle is represented by one or a few pixels in the binary ground truth mask. However, detections close to a particle but outside the particle region in the binary ground truth mask are not rewarded during network training. This decreases the stability of network training and makes training more difficult. In addition, [153] use a network with a relatively large number of parameters, and [155] employ a sliding window scheme which increases the computational cost. [158] use a CNN to directly regress offsets of bounding boxes, which, however,

involves a highly nonlinear mapping from input images to point coordinates. None of the previous deep learning methods employs density map regression for particle detection, which is often used for key point detection in videos of natural scenes (e.g., faces, persons) yielding state-of-the-art performance (e.g., [253, 254]). In [255], density map regression is employed for cell counting in 2D images. A mean square error loss is used, which is not sensitive to small errors and not adaptive. Moreover, all deep learning methods above perform 2D detection using 2D CNNs, thus valuable volumetric information of 3D microscopy images is not exploited.

In this contribution, we introduce a novel deep learning approach for 3D particle detection in 3D fluorescence microscopy images. Instead of pixel-wise binary classification [153, 155, 156] or direct coordinate regression [158], we perform image-to-image mapping based on regressing a density map. The density map encodes the probability that a particle is located at a certain position. Thus, highly nonlinear direct prediction of point coordinates is avoided, and detections close to particles are rewarded in the network training. In addition, compared to [153, 155, 156, 158], we exploit uncertainties in the manually annotated ground truth positions of particles for network training. To focus on particles in comparison to background image points, we suggest using the adaptive wing loss which was previously used for face recognition [253]. To cope with the very strong imbalance between particle and background image points for 3D images, we use a weighted loss map, which assigns high weights to particles and difficult background image points close to particles. Compared to [253], where separate density maps are computed for each key point, in our approach all particles of an image are represented by only one density map. Different to [155], a sliding window scheme is not required, and all particles within an image are detected at once by sharing full-image convolutional features. Our method is the first that performs image-to-image mapping via density map regression for particle detection in microscopy images. In contrast to [153, 155, 156, 158, 253, 254, 255], the full 3D image information is exploited. The proposed 3D particle detection approach has been evaluated using 3D data of the Particle Tracking Challenge (PTC, [65]) as well as real 3D fluorescence microscopy images of chromatin structures and interneurons. It turned out that our approach outperforms previous methods.

7.1.2 Method

Our proposed deep learning approach for 3D particle detection, denoted as Density Map DetNet 3D (DM-DetNet3D), performs image-to-image mapping via density map regression. An overview of the network architecture is given in Fig. 7.1. The network is based on the slim hourglass architecture of DetNet [156], which was used for 2D images. The network is composed of a contracting and expanding path, can handle objects at different scales, and does not require a sliding window scheme. To detect all particles within an image at once, full-image convolution features are

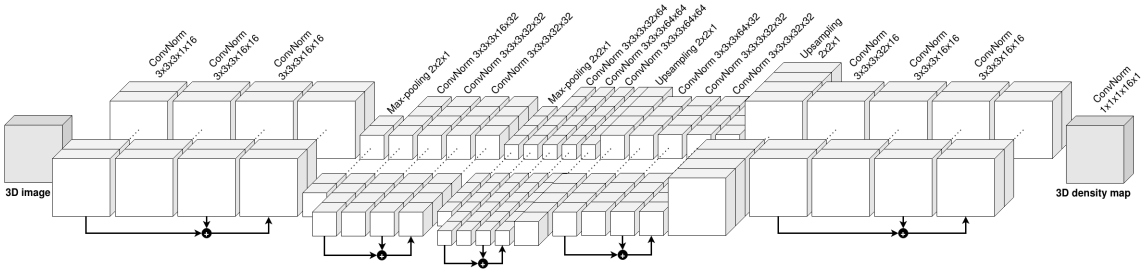


Figure 7.1 Architecture of the proposed DM-DetNet3D network.

shared. Since detailed boundary information is not needed to detect sub-resolution particles and to reduce the number of parameters, long range skipping connections are not employed. To cope with the vanishing gradient problem, residual blocks [**?**] are used. Since batch normalization requires a representative data set to compute meaningful statistics, which is difficult to achieve when using only a few training samples, instance normalization is utilized. Compared to [156], we exploit the full 3D information by 3D convolution operations instead of 2D convolution operations in the residual blocks.

In contrast to pixel-wise binary classification [153, 155, 156] and direct coordinate regression [158], DM-DetNet3D performs image-to-image mapping based on regressing a density map which encodes the probability that a particle is located at a certain position. Thus, particle detections close to the correct position are taken into account in the network training, and highly nonlinear direct prediction of point coordinates is avoided. In our method, ground truth points in the training data are treated as Gaussian distributions centered around the annotated positions rather than using discrete image points as in [156]. This reflects that manual annotation of noisy, sub-resolution particles is generally uncertain, particularly for 3D images. The level of uncertainty is represented by the standard deviations $\sigma_{x,y}$ and σ_z of the Gaussian distribution. To exploit the full range of the sigmoid function in the output layer of the network, we normalize the values of the ground truth density maps to the range $[0, 1]$. During inference, the network predicts a density map from which particle positions are obtained by determining local maxima. Fig. 7.2 illustrates the type of ground truth for particle detection by density map regression compared to pixel-wise binary classification.

For accurate particle detection and localization via density map regression, the prediction accuracy of the network for image points representing particles is very important, since even small errors have a large effect. In comparison, for background image points the prediction accuracy is less important, since small errors usually have a small effect. Thus, the network should focus on particle image points during training (increased sensitivity). This can be achieved by using the adaptive wing loss (AWing, [253]), which adapts to different values in the ground truth mask to increase the sensitivity to errors for particles compared to

(a) Original image **(b)** Pixel-wise binary classification **(c)** Density map regression

Figure 7.2 Original image and different types of ground truth. For visualization, all z-slices are max-pooled into one slice.

background image points. This is an advantage over the standard mean square error (MSE) loss, which is insensitive to small errors and not adaptive causing blurred and dilated density maps. AWing is an extension of the wing loss for density map regression and is differentiable around zero. For a 3D image with width w, height h, and depth d, the AWing loss for the position x_i in the predicted density map $\boldsymbol{X} \in [0,1]^{h \times w \times d}$ is defined by:

$$\text{AWing}(x_i, y_i) = \begin{cases} \omega \ln(1 + |\frac{y_i - x_i}{\epsilon}|^{\alpha - y_i}) & \text{if } |y_i - x_i| < \theta \\ A|y_i - x_i| - C & \text{otherwise} \end{cases} \tag{7.1}$$

where y_i is the position in the ground truth density map $\boldsymbol{Y} \in [0,1]^{h \times w \times d}$. The parameters $A = \omega(1/(1 + (\theta/\epsilon)^{(\alpha - y_i)}))(\alpha - y_i)((\theta/\epsilon)^{(\alpha - y_i - 1)})(1/\epsilon)$ and $C = (\theta A - \omega \ln(1 + (\theta/\epsilon)^{(\alpha - y_i)}))$ ensure that the loss is continuous and smooth at $|y_i - x_i| = \theta$. $\theta \in [0,1]$ is a threshold for switching between the linear and nonlinear part, $\alpha - y_i$ is used to adapt the curvature of the loss function to y_i (α has to be slightly larger than two). $\epsilon \in \mathbb{R}_{>0}$ limits the curvature of the nonlinear part and should not be set to a very small value since this would cause unstable training and exploding gradients for very small errors. In our experiments, we used $\alpha = 2.1$, $\omega = 14$, $\epsilon = 1$, and $\theta = 0.5$. Computing $\text{AWing}(x_i, y_i)$ for all positions in the predicted density map defines the *AWing loss map* $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$. In contrast to [253], we employ AWing to predict Gaussian distributions for all particles in an image using a *single* density map, whereas there for each key point a separate density map was used. Also, there a different application (face recognition) and 2D images were considered. We also tested our network using an MSE loss, which did not yield good results.

To address the very strong imbalance between particle and background image points for 3D images, we also use a *weight map* $\boldsymbol{W}$, which assigns high weights to particles and difficult background image points close to particles compared to

other background image points:

$$\boldsymbol{W} = \begin{cases} \lambda + 1 & \text{for } \boldsymbol{Y}^d \geq T \\ 1 & \text{otherwise} \end{cases} \tag{7.2}$$

where $\boldsymbol{Y}^d \in [0,1]^{h \times w \times d}$ is obtained by a 3×3 dilation of $\boldsymbol{Y}$ [253], λ defines the strength of the weighting (we used $\lambda = 10$), and $T \in [0,1]$ is a threshold. In $\boldsymbol{W}$, $\lambda + 1$ is assigned to particle image points and background image points close to particles, and 1 to all other image points. The *weighted loss map* $\mathcal{L}_{\mathrm{w}}(\boldsymbol{X}, \boldsymbol{Y})$ between $\boldsymbol{X}$ and $\boldsymbol{Y}$ combines the weight map $\boldsymbol{W}$ and the AWing loss map $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$ and is defined by:

$$\mathcal{L}_{\mathrm{w}}(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) \otimes \boldsymbol{W} \tag{7.3}$$

where $\otimes$ denotes the Hadamard product.

During network training, the mean value of $\mathcal{L}_{\mathrm{w}}(\boldsymbol{X}, \boldsymbol{Y})$ is minimized using the AMSGrad optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate was set to 0.001 and a mini-batch size of 4 was used. Data augmentation involved random cropping as well as horizontal and vertical flipping. To avoid overfitting, we applied early stopping after convergence is reached. The data set was randomly split into 50% for training, 25% for validation, and 25% for testing.

7.1.3 Experimental Results

Particle Tracking Challenge Data

We evaluated DM-DetNet3D using 3D images of the PTC [65] and performed a comparison with the 3D versions of SEF [147] (SEF3D) and DetNet [156] (DetNet3D). SEF3D is based on the Laplacian-of-Gaussian, and DetNet3D performs image-to-image mapping by voxel-wise binary classification (see Sec. 3.2).

We used all 3D images of the virus scenario from the PTC comprising different object densities (low, medium, high) and SNR levels (SNR = 1, 2, 4, 7). In total, we considered 1200 images ($512 \times 512 \times 10$ voxels) and studied the results for each SNR level (300 images per SNR level). To measure the detection performance, we computed the F1 score $\in [0,1]$ using a gate of 5 voxels. The localization accuracy of correct particle detections is measured by the root mean square error (RMSE). For each SNR level, we averaged the F1 score and the RMSE over the respective images. The results for all SNR levels as well as the average values over the SNR levels and corresponding standard deviations are provided in Table 7.1. For the F1 score, DM-DetNet3D outperforms the other methods for all SNR levels, especially for low SNR levels. For RMSE, DM-DetNet3D yields the best result in three out of four cases.

Table 7.1 Results for 3D images of the Particle Tracking Challenge (mean ± standard deviation).

SNR	Method	F1	*RMSE*
1	SEF3D	0.376 ± 0.141	2.491 ± 0.157
	DetNet3D	0.544 ± 0.118	**1.363** ± 0.132
	DM-DetNet3D	**0.623** ± 0.116	1.396 ± 0.119
2	SEF3D	0.891 ± 0.059	0.967 ± 0.046
	DetNet3D	0.896 ± 0.059	0.686 ± 0.034
	DM-DetNet3D	**0.973** ± 0.013	**0.612** ± 0.040
4	SEF3D	0.993 ± 0.005	0.655 ± 0.019
	DetNet3D	0.970 ± 0.012	0.522 ± 0.027
	DM-DetNet3D	**0.994** ± 0.005	**0.496** ± 0.016
7	SEF3D	0.994 ± 0.005	0.574 ± 0.011
	DetNet3D	0.982 ± 0.007	0.508 ± 0.012
	DM-DetNet3D	**0.995** ± 0.004	**0.505** ± 0.015

Table 7.2 Results for real 3D images of chromatin structures.

Method	F1	RMSE
SEF3D	0.774 ± 0.078	1.934 ± 0.243
DetNet3D	0.711 ± 0.068	1.817 ± 0.193
DM-DetNet3D	**0.820** ± 0.035	**1.773** ± 0.214

Real Fluorescence Microscopy Data

We also evaluated DM-DetNet3D based on real 3D live cell fluorescence microscopy images of chromatin structures acquired with super-resolution 3D structured illumination microscopy [247]. The data set comprises 60 3D images from five different temporal image sequences ($512 \times 512 \times 5$ voxels). Ground truth was determined manually for difficult image regions (2637 particle positions in total). Main challenges of the data set are clustering particles and varying SNR levels due to photobleaching over time. The performance values in Table 7.2 show that DM-DetNet3D yields the best result. Example results in Fig. 7.3 demonstrate that the result of DM-DetNet3D agrees well with the ground truth.

In addition, we used challenging real 3D confocal fluorescence microscopy images of calretinin-immunostained interneurons in hippocampus sections of mice. For this data set we used one image with 116 ground truth positions for training and validation, and one image with 196 ground truth positions for testing. DM-DetNet3D yields an F1 score of 0.747 and outperforms SEF3D and DetNet3D with F1 scores of 0.686 and 0.543, respectively. For RMSE, DM-DetNet3D (2.943) achieves similar results as SEF3D (2.871) and DetNet3D (2.835).

7.1.4 Conclusion

We presented a new deep learning approach for 3D particle detection in 3D fluorescence microscopy images that performs image-to-image mapping based on regressing a density map. During network training, detections close to particles

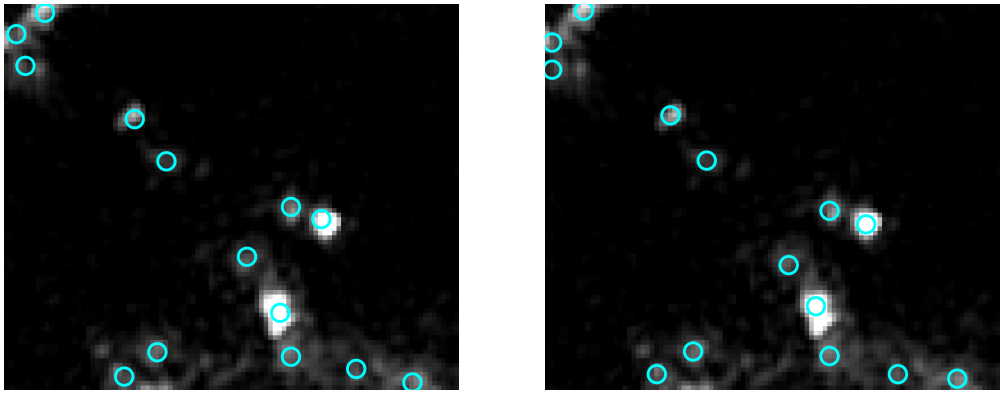


Figure 7.3 Ground truth (left) and results of DM-DetNet3D (right) for a real 3D image of chromatin structures (section, maximum intensity projection).

are rewarded and uncertainties in the manually annotated ground truth positions are exploited. To focus on particles in comparison to background image points, we suggest using the adaptive wing loss. We also employ a weighted loss map to cope with the very strong imbalance between particle and background image points for 3D images. Our experiments for 3D images of the PTC and real 3D microscopy images show that our approach outperforms previous methods.

7.2 Deep Probabilistic Particle Detection and Tracking

In this section, we present a deep learning method for probabilistic particle detection and tracking in fluorescence microscopy images. The work has been submitted for publication [64].

7.2.1 Introduction

Previous work on particle detection in fluorescence microscopy images has introduced classical methods (e.g., [56, 143, 147, 148]) typically based on predefined and simplified appearance models (e.g., Gaussian function). In recent years, deep learning methods for particle detection have been presented that show promising results (e.g., [63, 153, 155, 156, 158, 256]). In [153, 155, 156, 256], particle detection is considered as a pixel-wise binary classification task. However, detections that are close to, but not located within the particle regions of the binary ground truth mask, are not rewarded during network training. This reduces the stability of the training process. In [158], a CNN was used to directly regress the offsets of bounding boxes. However, direct mapping of input images to point coordinates is highly nonlinear. For particle detection in 3D images, we performed image-to-image mapping via density map regression [63] (described in Sec. 7.1). However, particle positions are not localized with sub-pixel resolution. In addition, none of the previous

particle detection methods exploits temporal information which can improve the performance.

In previous work on fluorescent particle tracking, classical deterministic and probabilistic methods were introduced. Compared to deterministic methods (e.g., [187]), probabilistic methods consider uncertainties by Bayesian filtering (e.g., [151, 157, 193, 194]). Recently, deep learning methods for particle tracking were proposed (e.g., [22, 60, 257]). Typically, a recurrent neural network (RNN) is used to exploit object motion for correspondence finding. In [61] (described in Chapter 5), we introduced a probabilistic deep learning method that takes into account aleatoric and epistemic uncertainty. However, a classical detection method was used and uncertainty of individual particle detections was not considered. Also, the RNN emulating Bayesian filtering is not fully probabilistic but includes non-Bayesian layers.

In this contribution, we present a novel deep learning method for probabilistic particle detection and tracking in fluorescence microscopy images. For particle detection, a slim hourglass CNN is proposed that integrates temporal information for regressing a density map, which represents the probability that a particle is located at a certain position. Thus, detections close to particles are rewarded during network training compared to pixel-wise binary classification [153, 155, 156, 256], and highly nonlinear regression of positions from image data [158] is avoided. In contrast to all previous particle detection methods (e.g., [56, 63, 143, 147, 148, 153, 155, 156, 158, 256]), temporal image information is exploited to improve the detection performance. Different to [63, 153, 155, 156, 158], sub-pixel particle positions are determined. For particle tracking, we introduce a Bayesian neural network that emulates Bayesian filtering. Short- and long-term temporal dependencies of individual object dynamics are represented by gated recurrent units (GRUs, [103]). Compared to [61] (see Chapter 5), the network is fully Bayesian, deep learning is used for particle detection, and uncertainty information of individual particle detections is considered. For correspondence finding, we use a neural network that computes assignment probabilities jointly across multiple detections as well as probabilities of missing detections. Manually annotated data is not needed for training the Bayesian neural network and the network for correspondence finding. We evaluated the proposed deep learning method based on data of the Particle Tracking Challenge (PTC, [65]). It turned out that our method outperforms previous methods. We also successfully applied the method to fluorescence microscopy images displaying hepatitis C virus (HCV) proteins.

7.2.2 Method

The proposed method combines a CNN for particle detection (Deep Particle Detector, DPD) with a fully Bayesian RNN for tracking (Deep Bayesian Tracker, DBT),

and is denoted DPD-DBT. To detect particles in fluorescence microscopy images, the suggested DPD employs a slim CNN that integrates temporal information and performs density map regression. The neural network is based on an hourglass-shaped architecture with residual blocks [63, 156] (see Sec. 7.1) that processes image objects at different scales and avoids the need for a sliding window scheme. To reward detections close to particles during network training compared to pixel-wise binary classification (e.g, [153, 155, 156, 256]), and to avoid highly nonlinear regression of positions (e.g., [158]), DPD regresses a density map representing the probability that a particle is located at a certain image position. For network training, ground truth density maps are generated by using Gaussian distributions centered at annotated particle positions. The values of the ground truth density map are normalized to the range of the sigmoid activation function $[0, 1]$ used in the final output layer. Compared to [63, 153, 155, 156, 158], particles are localized with sub-pixel resolution from the computed density map by Gaussian fitting. The value in the density map at the determined position represents its uncertainty. In contrast to previous particle detection methods [56, 63, 143, 147, 148, 153, 155, 156, 158, 256], we integrate temporal information to improve the performance under challenging conditions. For an image $\mathcal{I}_t \in \mathbb{R}^{w \times h}$ of width w and height h at time point t, DPD takes as input the temporal image sequence $\mathcal{I}_{t-T}, \ldots, \mathcal{I}_{t+T}$ concatenated along the channel dimension, where T denotes the number of previous and subsequent time points. To focus on particles instead of background during network training, we use the adaptive wing loss (AWing, [253] , described in Sec. 7.1.2). This loss adapts to the values in the ground truth density map and penalizes errors for particles more than for the background. Compared to previous methods for 3D particle detection [63] (see Sec. 7.1) and face recognition [253], an additional loss weight map is not required. We used the AMSGrad optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with an initial learning rate of 0.001 and a mini-batch size of 4 to minimize the mean value over the AWing loss of all image points. For data augmentation, we applied random cropping as well as horizontal and vertical flipping. For training, we used the training sequences of the PTC.

For tracking, the proposed Deep Bayesian Tracker (DBT) emulates classical Bayesian filtering. The network consists of a state prediction and update block (see Fig. 7.4). To estimate the next state of an individual object based on both short- and long-term temporal dependencies of its dynamics, the prediction block employs gated recurrent units (GRUs, [103]). The update block uses the assigned detection obtained by DPD to correct the predicted state. Compared to [61] (see Chapter 5), we use a deep learning method for particle detection and include uncertainty of individual particle detections. To capture epistemic uncertainty (model uncertainty) arising from lack of knowledge due to limited or insufficient training data, we employ Bayesian layers with reparametrization [119], where all learnable parameters are represented by Gaussian distributions. During network training, the mean and variance of the Gaussian distributions are learned instead

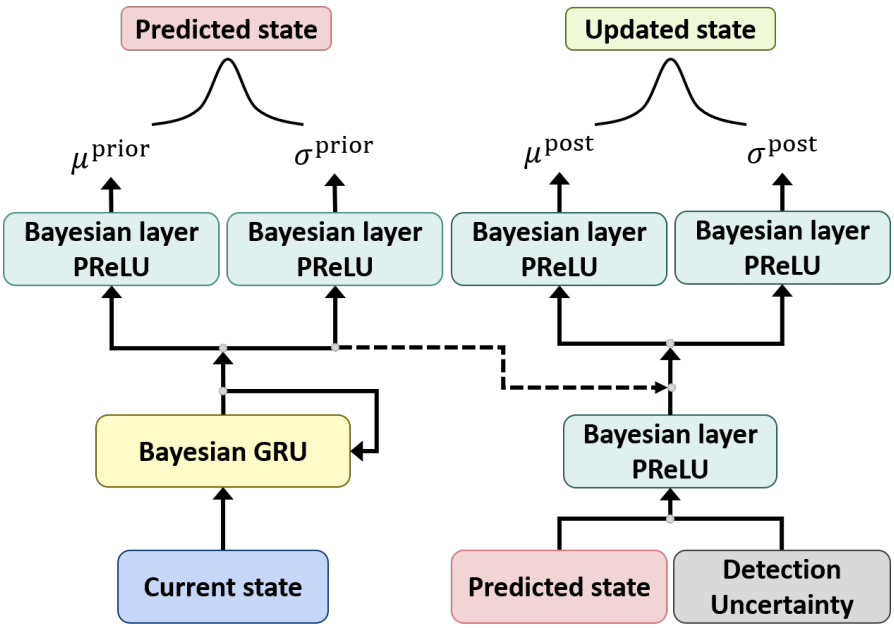


Figure 7.4 Architecture of the fully Bayesian network.

of directly determining point estimates of the learnable parameters. The network consists of a Bayesian GRU layer with 16 units, a Bayesian layer with 16 Parametric Rectified Linear Units (PReLU), and a Bayesian layer with 2 PReLUs in each of the 4 network heads. Thus, the entire network is Bayesian in contrast to [61], where only the network heads are Bayesian. This implies that all network layers are directly included in the computation of the epistemic uncertainty. Aleatoric uncertainty due to inherent noise in the data (e.g., object motion) is considered by learning to estimate the mean values (μ^{prior}, μ^{post}) and standard deviations (σ^{prior}, σ^{post}) of two Gaussian distributions from which the predicted state and the updated state are obtained, respectively. To improve subsequent motion analysis, the estimated uncertainties provide important information about the reliability of the computed trajectories (e.g., exclusion of unreliable tracks or track points), which has been demonstrated in [61]. For our fully Bayesian network, the loss for one training sample is computed as:

$$\mathcal{L} = \underbrace{-\log P\left(\tilde{\mathbf{x}}|\mu^{\text{prior}}, \sigma^{\text{prior}}\right)}_{\text{predicted state}} \underbrace{-\lambda \log P\left(\tilde{\mathbf{x}}|\mu^{\text{post}}, \sigma^{\text{post}}\right)}_{\text{updated state}} \tag{7.4}$$

where P denotes the probability and $\tilde{\mathbf{x}}$ is the true next state. Note that the prediction and update blocks are trained simultaneously as a unified network, but are applied sequentially during tracking. Compared to [61], we employed RMSprop ($\beta = 0.9$) instead of AMSGrad as optimizer, resulting in better convergence. We used an initial learning rate of 0.002 and $\lambda = 2$. For correspondence finding, a neural

network is employed comprising four consecutive fully connected (FC) layers, followed by a FC linear output layer with softmax normalization [61]. To avoid overfitting, dropout with a rate of 0.4 is applied during training. The network computes assignment probabilities jointly across multiple detections as well as probabilities of missing detections based on the Euclidean distance between the predicted states and particle detections. Training the fully Bayesian network and the network for correspondence finding requires only synthetic data and was performed as described in Sec. 5.2.5. To establish one-to-one correspondences using the computed probabilities, the Jonker-Volgenant shortest augmenting path algorithm is employed.

7.2.3 Experimental Results

For performance evaluation, we used image data from the Particle Tracking Challenge (PTC, [65]) and benchmarked our DPD-DBT against the overall top-three methods (Method 5, 1, 2) described in Sec. 3.2. We also compared the performance with previous deep learning methods for particle tracking: Deep Particle Hypotheses Tracker (DPHT, [60], Sec. 4.2) and Deep Probabilistic Particle Tracker (DPPT, [61], Chapter 5). Both methods employ SEF and Gaussian fitting for particle detection. In addition, we considered variants of DPD-DBT with SEF and Gaussian fitting (SEF-DBT) and DM-DetNet [63] (DM-DetNet-DBT) for detection.

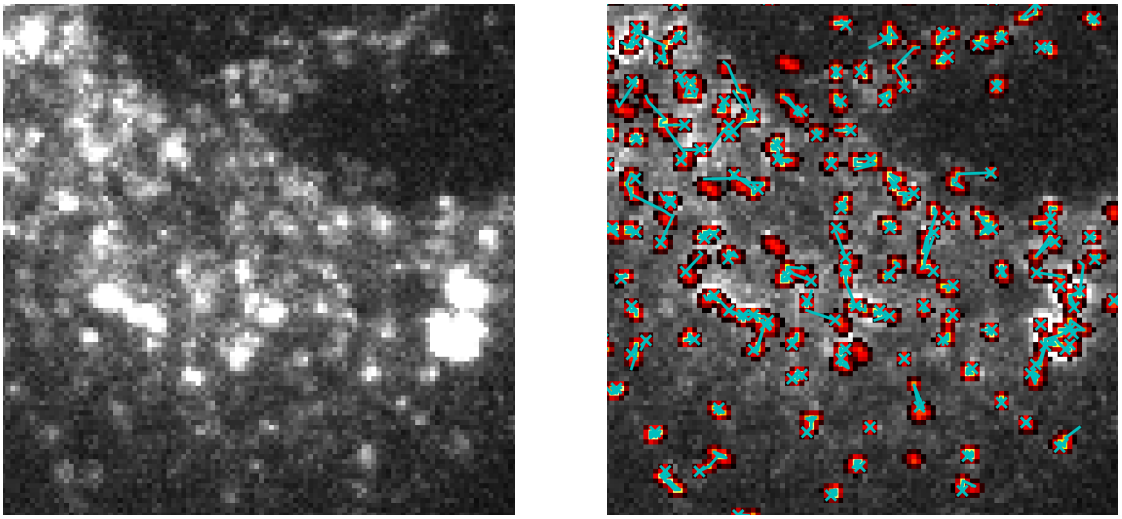
To examine the performance under challenging conditions due to cluttered environments and strong image noise, we used data of the vesicle scenario from the PTC with medium and high particle density (~500 and ~1000 particles/image) and low SNR values (SNR = 1 and SNR = 2). Each image sequence consists of 100 images (512×512 pixels) with randomly appearing and disappearing particles that perform Brownian motion. We used the tracking performance metrics α, β, *JSC*, JSC_θ, and *RMSE* described in Sec. 3.1. The higher the values, the better the performance (except for *RMSE*).

The quantitative results are presented in Table 7.3, with the best performance highlighted in bold. It can be seen that DPD-DBT outperforms the other methods in all cases for α, β, *JSC*, and JSC_θ. Using DPD for particle detection substantially improves the performance compared to SEF and DM-DetNet. In addition, DBT leads to improved or similar results compared to previous methods.

We also applied DPD-DBT to fluorescence microscopy images of hepatitis C virus (HCV) nonstructural protein 5A. We considered one image sequence (30 frames, 512×512 pixels) acquired using a confocal microscope. The data is challenging due to cluttered environments and low SNR. For a demanding image region, tracking results of DPD-DBT and computed aleatoric and epistemic uncertainty are shown in Fig. 7.5 (probability density of state update, yellow: high probability values, red: low probability values).

Table 7.3 Tracking performance for data of the vesicle scenario from the Particle Tracking Challenge. Bold indicates best performance.

Density	SNR	Method	α	β	JSC	JSC_θ	$RMSE$
Medium	1	Method 5	0.162	0.142	0.225	0.458	2.172
		Method 1	0.027	0.026	0.034	0.300	**1.533**
		Method 2	0.198	0.111	0.192	0.335	2.386
		DPHT	0.128	0.100	0.155	0.380	1.858
		DPPT	0.172	0.139	0.223	0.407	2.164
		SEF-DBT	0.180	0.139	0.225	0.406	2.208
		DM-DetNet-DBT	0.177	0.143	0.236	0.461	2.274
		DPD-DBT	**0.295**	**0.241**	**0.374**	**0.535**	1.997
	2	Method 5	0.448	0.391	0.489	0.664	1.325
		Method 1	0.398	0.298	0.340	0.411	**0.840**
		Method 2	0.517	0.417	0.510	0.629	1.254
		DPHT	0.520	0.448	0.526	0.680	0.874
		DPPT	0.562	0.485	0.564	0.700	1.014
		SEF-DBT	0.558	0.488	0.573	0.722	1.044
		DM-DetNet-DBT	0.603	0.536	0.624	0.774	0.966
		DPD-DBT	**0.616**	**0.550**	**0.625**	**0.776**	0.880
High	1	Method 5	0.136	0.120	0.198	0.460	2.296
		Method 1	0.091	0.064	0.089	0.231	**1.859**
		Method 2	0.163	0.080	0.147	0.324	2.531
		DPHT	0.121	0.104	0.158	0.444	1.984
		DPPT	0.158	0.123	0.189	0.391	2.055
		SEF-DBT	0.141	0.116	0.183	0.434	2.183
		DM-DetNet-DBT	0.185	0.148	0.233	**0.462**	2.182
		DPD-DBT	**0.206**	**0.157**	**0.274**	**0.462**	2.377
	2	Method 5	0.353	0.295	0.382	0.607	1.484
		Method 1	0.294	0.217	0.256	0.379	1.088
		Method 2	0.356	0.249	0.331	0.515	1.582
		DPHT	0.383	0.311	0.376	0.580	**1.044**
		DPPT	0.421	0.341	0.416	0.601	1.232
		SEF-DBT	0.421	0.352	0.426	0.626	1.246
		DM-DetNet-DBT	0.432	0.356	0.430	0.610	1.139
		DPD-DBT	**0.477**	**0.398**	**0.471**	**0.663**	1.139



(a) Original image (HCV) **(b)** DPD-DBT

Figure 7.5 Tracking results and computed uncertainty.

7.2.4 Conclusion

We introduced a deep learning method for probabilistic particle detection and tracking in fluorescence microscopy images. For detection, we proposed a CNN that integrates temporal information for regressing a density map from which sub-pixel positions are determined. For tracking, we suggested a fully Bayesian neural network that emulates Bayesian filtering and exploits uncertainty of individual detections. Experiments based on data of the PTC show that our method outperforms previous methods. We also successfully applied the method to microscopy images of hepatitis C virus proteins.

Chapter 8

Summary and Outlook

In this thesis, novel deep learning methods for detection and tracking of multiple particles in fluorescence microscopy images have been presented that address various task-specific challenges (e.g., low SNR, lack of appearance characteristics, complex motion behavior, high object density). Our experiments show that the developed methods are applicable to various virus structures (e.g., virus proteins, virus particles) and sub-cellular structures (e.g., cell surface receptors, chromatin structures) imaged by different fluorescence microscopy techniques. Below, the main contributions of the thesis are summarized, limitations are discussed, and possible future work is described.

8.1 Summary

In this section, we summarize the main contributions of the thesis.

- **Recurrent Neural Networks for Particle Tracking:** Novel deep learning methods for particle tracking in fluorescence microscopy images were developed that exploit temporal information by using recurrent neural networks. First, a method was proposed that takes into account past information about object dynamics for state prediction and correspondence finding. As an extension, a method was presented that considers past and future information for correspondence finding and track initiation as well as termination. To resolve ambiguities by using information at later time points, several track hypotheses are propagated into the future. A main advantage is that assignment probabilities are computed jointly across multiple detections, and probabilities for missing detections are also determined. In addition, handcrafted similarity measures as well as assumptions about probability distributions are not required, and network training is based solely on synthetic data.

- **Deep Probabilistic Particle Tracker:** We introduced the first probabilistic deep learning method for particle tracking in fluorescence microscopy image sequences. The method is based on a recurrent neural network that

mimics classical Bayesian filtering by learning to predict the next state and to update the predicted state based on an assigned detection. To capture aleatoric uncertainty, the network determines Gaussian distributions from which the predicted and updated states are obtained. Bayesian layers with reparametrization are employed to incorporate epistemic uncertainty. Experimental results demonstrate that the uncertainty information can be exploited to increase the accuracy of subsequent motion analysis by excluding unreliable tracks or track points. It was also shown that the uncertainty provides important information about the suitability of the training data and can be used to select the generated training data set with the best-suited motion model. To avoid tedious manual annotation of training data, a novel scheme was presented to generate arbitrary amounts of synthetic images based on automatically extracted information from the images of an application.

- **Deep Particle Tracker and Colocalization Analyzer:** The first deep learning method for combined particle tracking and colocalization analysis in two-channel fluorescence microscopy images was presented. A convolutional Long Short-Term Memory network is used to exploit short and long-term temporal dependencies of object motion as well as image intensities. In addition, colocalization probabilities are calculated, and colocalization information is used to improve tracking.

- **Deep Learning for Particle Detection and Tracking:** A novel deep learning method for 3D particle detection in 3D fluorescence microscopy images was introduced, which performs density map regression. Detections close to particles are rewarded during network training, and highly nonlinear direct prediction of point coordinates is avoided. To focus on particles compared to background image points, the adaptive wing loss is used. A weight map is applied to cope with the very strong imbalance between particle and background image points in 3D images. As an extension, temporal information is integrated and sub-pixel positions are determined. For tracking, the proposed particle detection method is combined with a fully Bayesian neural network that considers uncertainty information of individual particle detections.

- **Quantitative Performance Evaluation:** Extensive performance evaluations of the proposed particle detection and tracking methods have been conducted based on data from the Particle Tracking Challenge as well as fluorescence microscopy images of various viral and sub-cellular structures. It turned out that the proposed methods outperform previous methods.

8.2 Outlook

In this section, we describe possible future work.

- Particle detection and tracking could be combined in a unified neural network architecture. In addition, establishing one-to-one correspondences within a neural network would eliminate the necessity of using a classical combinatorial optimization algorithm. These are two challenging but important steps toward an end-to-end learning approach for fluorescent particle tracking.
- Since all methods presented in this thesis are based on supervised learning, self-supervised or unsupervised methods for particle detection and tracking could be developed in future work.
- Our method for combined particle tracking and colocalization analysis was developed for two-channel image data. Thus, this method could be extended for multi-channel data with more than two channels in future work. In addition, a deep learning method could be used for particle detection instead of a classical detection method.
- Deep learning methods for motion and behavior analysis of fluorescent particles could be investigated and combined with the methods proposed in this thesis.
- In future work, the fluorescent particle detection and tracking methods presented in this thesis could be adapted and employed for microscopy images of other applications.

Bibliography

[1] F. Wäldchen, J. Schlegel, R. Götz, M. Luciano, M. Schnermann, S. Doose, and M. Sauer, "Whole-cell imaging of plasma membrane receptors by 3D lattice light-sheet dSTORM," *Nat. Commun.*, vol. 11, no. 887, 2020.

[2] T.-C. Ku, Y.-N. Huang, C.-C. Huang, D.-M. Yang, L.-S. Kao, T.-Y. Chiu, C.-F. Hsieh, P.-Y. Wu, Y.-S. Tsai, and C.-C. Lin, "An automated tracking system to measure the dynamic properties of vesicles in living cells," *Microsc. Res. Tech.*, vol. 70, no. 2, pp. 119–134, 2007.

[3] A. Akhmanova and M. O. Steinmetz, "Tracking the ends: a dynamic protein network controls the fate of microtubule tips," *Nat. Rev. Mol. Cell Biol.*, vol. 9, no. 4, 309–322, 2008.

[4] B. Sinha, D. Bhattacharya, D. K. Sinha, S. Talwar, S. Maharana, S. Gupta, and G. Shivashankar, "Dynamic organization of chromatin assembly and transcription factories in living cells," *Methods Cell Biol.*, vol. 98, pp. 57–78, 2010.

[5] B. Brandenburg and X. Zhuan, "Virus trafficking – learning from single-virus tracking," *Nat. Rev. Microbiol.*, vol. 5, 197–208, 2007.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[7] H. Greenspan, B. van Ginneken, and R. M. Summers, "Deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1153–1159, 2016.

[8] 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.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[10] J.-B. Cordonnier, A. Mahendran, A. Dosovitskiy, D. Weissenborn, J. Uszkoreit, and T. Unterthiner, "Differentiable patch selection for image recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 2351–2360.

[11] Y. Rao, W. Zhao, Z. Zhu, J. Lu, and J. Zhou, "Global filter networks for image classification," in *Proc. Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017.

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[16] 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. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[17] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.

[18] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Proc. Conference on Artificial Intelligence (AAAI)*, 2017, pp. 4225–4232.

[19] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 300–311.

[20] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixe, and X. Alameda-Pineda, "How to train your deep multi-object tracker," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] C. Payer, D. Štern, M. Feiner, H. Bischof, and M. Urschler, "Segmenting and tracking cell instances with cosine embeddings and recurrent hourglass networks," *Med. Image Anal.*, vol. 57, pp. 106–119, 2019.

[22] Y. Yao, I. Smal, I. Grigoriev, A. Akhmanova, and E. Meijering, "Deep-learning method for data association in particle tracking," *Bioinformatics*, vol. 36, no. 19, pp. 4935–4941, 2020.

[23] A. A. Sekh, I. S. Opstad, A. B. Birgisdottir, T. Myrmel, B. S. Ahluwalia, K. Agarwal, and D. K. Prasad, "Learning nanoscale motion patterns of vesicles in living cells," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[24] T. Wollmann and K. Rohr, "Deep consensus network: Aggregating predictions to improve object detection in microscopy images," *Med. Image Anal.*, vol. 70, 102019, 2021.

[25] Y. Cao *et al.*, "VisDrone-DET2021: The vision meets drone object detection challenge results," in *Proc. IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 2847–2854.

[26] P. Dendorfer, A. Ošep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "MOTChallenge: A benchmark for single-camera multiple target tracking," *Int. J. Comput. Vis.*, vol. 129, no. 4, 845–881, 2021.

[27] J. W. Lichtman and J.-A. Conchello, "Fluorescence microscopy," *Nat. Methods*, vol. 2, no. 12, 910–919, 2005.

[28] J. Oreopoulos, R. Berman, and M. Browne, "Spinning-disk confocal microscopy: present technology and future trends," in *Quantitative Imaging in Cell Biology*, ser. Methods in Cell Biology, J. C. Waters and T. Wittman, Eds. Academic Press, 2014, vol. 123, pp. 153–175.

[29] A. Esposito, S. Schlachter, G. S. K. Schierle, A. D. Elder, A. Diaspro, F. S. Wouters, C. F. Kaminski, and A. I. Iliev, *Quantitative Fluorescence Microscopy Techniques*, 2010, pp. 117–142.

[30] B. L. Haas, J. S. Matson, V. J. DiRita, and J. S. Biteen, "Imaging live cells at the nanometer-scale with single-molecule microscopy: Obstacles and achievements in experiment optimization for microbiology," *Molecules*, vol. 19, no. 8, pp. 12 116–12 149, 2014.

[31] D. J. Stephens and V. J. Allan, "Light microscopy techniques for live cell imaging," *Science*, vol. 300, no. 5616, pp. 82–86, 2003.

[32] M. Petráň, M. Hadravský, M. D. Egger, and R. Galambos, "Tandem-scanning reflected-light microscope*," *J. Opt. Soc. Am.*, vol. 58, no. 5, pp. 661–664, 1968.

[33] G. Q. Xiao, T. R. Corle, and G. S. Kino, "Real-time confocal scanning optical microscope," *Appl. Phys. Lett.*, vol. 53, no. 8, pp. 716–718, 1988.

[34] D. Toomre and J. B. Pawley, "Disk-scanning confocal microscopy," in *Handbook Of Biological Confocal Microscopy*, J. B. Pawley, Ed. Springer US, 2006, pp. 221–238.

[35] B. Huang, H. Babcock, and X. Zhuang, "Breaking the diffraction barrier: Super-resolution imaging of cells," *Cell*, vol. 143, no. 7, pp. 1047–1058, 2010.

[36] D. A. Agard, Y. Hiraoka, P. Shaw, and J. W. Sedat, "Fluorescence microscopy in three dimensions," in *Fluorescence Microscopy of Living Cells in Culture Part B. Quantitative Fluorescence Microscopy—Imaging and Spectroscopy*, ser. Methods in Cell Biology, D. L. Taylor and Y.-L. Wang, Eds. Academic Press, 1989, vol. 30, pp. 353–377.

[37] W. Yang and R. Yuste, "In vivo imaging of neural activity," *Nat. Methods*, vol. 14, no. 4, pp. 349–359, 2017.

[38] C. Manzo and M. F. Garcia-Parajo, "A review of progress in single particle tracking: from methods to biophysical insights," *Rep. Prog. Phys.*, vol. 78, no. 12, 124601, 2015.

[39] S. W. Hell and J. Wichmann, "Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy," *Opt. Lett.*, vol. 19, no. 11, pp. 780–782, 1994.

[40] B. Huang, M. Bates, and X. Zhuang, "Super-resolution fluorescence microscopy," *Annu. Rev. Biochem.*, vol. 78, no. 1, pp. 993–1016, 2009.

[41] S. S. G. Buddha, R. Kalita, and B. R. Boruah, "Estimation of point spread function of an imaging system using a programmable target," in *Proc. Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing*, vol. 10499, 2018, pp. 122 – 127.

[42] B. Zhang, J. Zerubia, and J.-C. Olivo-Marin, "Gaussian approximations of fluorescence microscope point-spread function models," *Appl. Opt.*, vol. 46, no. 10, pp. 1819–1829, 2007.

[43] S. W. Hell, "Far-field optical nanoscopy," *Science*, vol. 316, no. 5828, pp. 1153–1158, 2007.

[44] M. G. L. Gustafsson, "Surpassing the lateral resolution limit by a factor of two using structured illumination microscopy," *J. Microsc.*, vol. 198, no. 2, pp. 82–87, 2000.

[45] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess, "Imaging intracellular fluorescent proteins at nanometer resolution," *Science*, vol. 313, no. 5793, pp. 1642–1645, 2006.

[46] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)," *Nat. Methods*, vol. 3, no. 10, pp. 793–796, 2006.

[47] H. Andersson, T. Baechi, M. Hoechl, and C. Richter, "Autofluorescence of living cells," *J. Microsc.*, vol. 191, no. 1, pp. 1–7, 1998.

[48] N. Billinton and A. W. Knight, "Seeing the wood through the trees: A review of techniques for distinguishing green fluorescent protein from endogenous autofluorescence," *Anal. Biochem.*, vol. 291, no. 2, pp. 175–197, 2001.

[49] J. Surre, C. Saint-Ruf, V. Collin, S. Orenga, M. Ramjeet, and I. Matic, "Strong increase in the autofluorescence of cells signals struggle for survival," *Sci. Rep.*, vol. 8, no. 1, 12088, 2018.

[50] L. Song, E. Hennink, I. Young, and H. Tanke, "Photobleaching kinetics of fluorescein in quantitative fluorescence microscopy," *Biophys. J.*, vol. 68, no. 6, pp. 2588–2600, 1995.

[51] H. Schneckenburger, P. Weber, M. Wagner, S. Schickinger, V. Richter, T. Bruns, W. Strauss, and R. Wittig, "Light exposure and cell viability in fluorescence microscopy," *J. Microsc.*, vol. 245, no. 3, pp. 311–318, 2012.

[52] J. Icha, M. Weber, J. C. Waters, and C. Norden, "Phototoxicity in live fluorescence microscopy, and how to avoid it," *BioEssays*, vol. 39, no. 8, 1700003, 2017.

[53] T. Schroeder, "Long-term single-cell imaging of mammalian stem cells," *N. Methods*, vol. 8, no. 4, pp. 30–35, 2011.

[54] E. Meijering, I. Smal, O. Dzyubachyk, and J. Olivo-Marin, "Time-lapse imaging," in *Microscope Image Processing*, Q. Wu, F. A. Merchant, and K. R. Castleman, Eds. Academic Press, 2008, pp. 401–440.

[55] I. Smal, M. Loog, W. Niessen, and E. Meijering, "Quantitative comparison of spot detection methods in fluorescence microscopy," *IEEE Trans. Med. Imaging*, vol. 29, no. 2, pp. 282–301, 2010.

[56] 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," *Cytom. A*, vol. 87, no. 8, pp. 759–772, 2015.

[57] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *Int. J. Comput. Vis.*, vol. 10, no. 1, pp. 53–66, 1993.

[58] W. Godinez, M. Lampe, S. Wörz, B. Müller, R. Eils, and K. Rohr, "Deterministic and probabilistic approaches for tracking virus particles in time-lapse fluorescence microscopy image sequences," *Med. Image Anal.*, vol. 13, no. 2, pp. 325–342, 2009.

[59] 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. International Workshop on Deep Learning in Medical Image Analysis (DLMIA)*, vol. 11045, 2018, pp. 128–136.

[60] R. Spilger, A. Imle, J.-Y. Lee, B. Müller, O. T. Fackler, R. Bartenschlager, and K. Rohr, "A recurrent neural network for particle tracking in microscopy images using future information, track hypotheses, and multiple detections," *IEEE Trans. Image Process.*, vol. 29, pp. 3681–3694, 2020.

[61] R. Spilger, J.-Y. Lee, V. O. Chagin, L. Schermelleh, M. C. Cardoso, R. Bartenschlager, and K. Rohr, "Deep probabilistic tracking of particles in fluorescence microscopy images," *Med. Image Anal.*, vol. 72, 102128, 2021.

[62] R. Spilger, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep neural network for combined particle tracking and colocalization analysis in two-channel microscopy images," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2022.

[63] R. Spilger, V. O. Chagin, C. S. Bold, L. Schermelleh, U. C. Müller, M. C. Cardoso, and K. Rohr, "Deep neural network for 3D particle detection in 3D fluorescence microscopy images via density map regression," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2022.

[64] R. Spilger, J.-Y. Lee, M. T. Pham, R. Bartenschlager, and K. Rohr, "Deep learning method for probabilistic particle detection and tracking in fluorescence microscopy images," *submitted for publication*, 2022.

[65] N. Chenouard, I. Smal, F. de Chaumont, M. Maska, 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. G. Magnusson, J. Jalde;n, 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 Solorzano, J.-C. Olivo-Marin, and E. Meijering, "Objective comparison of particle tracking methods," *Nat. Methods*, vol. 11, no. 3, pp. 281–289, 2014.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[67] N. Buduma and N. Locascio, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. O'Reilly Media, Inc., 2017.

[68] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[69] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

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

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

[72] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861–867, 1993.

[73] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, 6232–6240.

[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, 2010, pp. 249–256.

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[76] C. Darken, J. Chang, and J. Moody, "Learning rate schedules for faster stochastic gradient search," in *Proc. IEEE Workshop on Neural Networks for Signal Processing II*, 1992, pp. 3–12.

[77] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2017.

[78] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen," Master thesis, Technical University of Munich, Germany, 1991.

[79] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, 1994.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[81] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. Annual Conference of the Cognitive Science Society (CogSci)*, 1986.

[82] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[83] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, 2011.

[84] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[85] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[86] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[87] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15, 2011, pp. 315–323.

[88] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[89] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[90] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. International Conference on Machine Learning (ICML)*, 2010, 807–814.

[91] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[92] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2013.

[93] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, 2020.

[94] F. Milletari, N. Navab, and S. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.

[95] P. J. Huber, "Robust Estimation of a Location Parameter," *Ann. Math. Stat.*, vol. 35, no. 1, pp. 73 – 101, 1964.

[96] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2004, 1329–1336.

[97] 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. 56, pp. 1929–1958, 2014.

[98] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.

[99] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[100] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Netw.*, vol. 1, no. 4, pp. 339–356, 1988.

[101] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[102] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

[103] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv:1406.1078*, 2014.

[104] S. Dash, B. R. Acharya, M. Mittal, A. Abraham, and A. Kelemen, *Deep Learning Techniques for Biomedical and Health Informatics*. Academic Press, 2020.

[105] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[106] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arxiv:1603.07285*, 2016.

[107] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611 – 629, 2018.

[108] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[109] 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. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[110] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[111] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[112] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[113] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, 5580–5590.

[114] A. Graves, "Practical variational inference for neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 24, 2011.

[115] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 1613–1622.

[116] J. Hernández-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner, "Black-box alpha divergence minimization," in *Proc. International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 1511–1520.

[117] Y. Gal and Z. Ghahramani, "Bayesian convolutional neural networks with Bernoulli approximate variational inference," *arXiv:1506.02158*, 2015.

[118] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. International Conference on Machine Learning (ICML)*, 2016, 1050–1059.

[119] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[120] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient Langevin dynamics," in *Proc. International Conference on Machine Learning (ICML)*, 2011, 681–688.

[121] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018, pp. 876–885.

[122] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, 6405–6416.

[123] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv:1912.02757*, 2019.

[124] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, 261–318, 2020.

[125] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digit. Signal Process.*, vol. 126, 103514, 2022.

[126] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[127] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, 154–171, 2013.

[128] R. Girshick, "Fast R-CNN," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[129] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[130] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[131] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[132] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv:1804.02767*, 2018.

[133] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv:2004.10934*, 2020.

[134] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. European Conference on Computer Vision (ECCV)*, vol. 9905, 2016, pp. 21–37.

[135] C. Szegedy, S. E. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *arXiv:1412.1441*, 2014.

[136] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6568–6577.

[137] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. European Conference on Computer Vision (ECCV)*, vol. 11218, 2018, pp. 765–781.

[138] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 778–10 787.

[139] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. International Conference on Machine Learning (ICML)*, vol. 97, 2019, pp. 6105–6114.

[140] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10 002.

[141] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[142] G. Turin, "An introduction to matched filters," *IRE Trans. Inform. Theory*, vol. 6, no. 3, pp. 311–329, 1960.

[143] J.-C. Olivo-Marin, "Extraction of spots in biological images using multiscale products," *Pattern Recognit.*, vol. 35, no. 9, pp. 1989–1996, 2002.

[144] A. Genovesio, T. Liedl, V. Emiliani, W. J. Parak, M. Coppey-Moisan, and J. . Olivo-Marin, "Multiple particle tracking in 3-D+t microscopy: Method and application to the tracking of endocytosed quantum dots," *IEEE Trans. Image Process.*, vol. 15, no. 5, pp. 1062–1070, 2006.

[145] B. Zhang, M. J. Fadili, J.-L. Starck, and J.-C. Olivo-Marin, "Multiscale variance-stabilizing transform for mixed-Poisson-Gaussian processes and its applications in bioimaging," in *IEEE International Conference on Image Processing (ICIP)*, vol. 6, 2007, pp. VI – 233–VI – 236.

[146] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch-based image sequence restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1096–1102, 2007.

[147] 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.

[148] 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.

[149] I. Smal, K. Draegestein, N. Galjart, W. Niessen, and E. Meijering, "Particle filtering for multiple object tracking in dynamic fluorescence microscopy images: Application to microtubule growth analysis," *IEEE Trans. Med. Imag.*, vol. 27, no. 6, pp. 789–804, 2008.

[150] D. S. Bright and E. B. Steel, "Two-dimensional top hat filter for extracting spots and spheres from digital images," *J. Microsc.*, vol. 146, no. 2, pp. 191–200, 1987.

[151] 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.

[152] A. J. Berglund, M. D. McMahon, J. J. McClelland, and J. A. Liddle, "Fast, bias-free algorithm for tracking single particles with variable size and shape," *Opt. Express*, vol. 16, no. 18, pp. 14 064–14 075, 2008.

[153] 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," *Cold Spring Harb. Symp. Quant. Biol.*, vol. 82, pp. 57–70, 2017.

[154] M. Mabaso, D. Withey, and B. Twala, "Spot detection in microscopy images using convolutional neural network with sliding-window approach," in *Proc. International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC)*, vol. 2, 2018, pp. 67–74.

[155] 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.

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

[157] M. Dmitrieva, H. L. Zenner, J. Richens, D. S. Johnston, and J. Rittscher, "Protein tracking by CNN-based candidate pruning and two-step linking with Bayesian network," in *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2019)*, 2019, pp. 1–6.

[158] F. Zakrzewski, W. de Back, M. Weigert, T. Wenke, S. Zeugner, R. Mantey, C. Sperling, K. Friedrich, I. Roeder, D. Aust, G. Baretton, and P. Hönscheid, "Automated detection of the HER2 gene amplification status in fluorescence in situ hybridization images for the diagnostics of cancer tissues," *Sci. Rep.*, vol. 9, no. 1, 8231, 2019.

[159] B. T. Eichenberger, Y. Zhan, M. Rempfler, L. Giorgetti, and J. A. Chao, "deepBlink: threshold-independent detection and localization of diffraction-limited spots," *Nucleic Acids Res.*, vol. 49, no. 13, pp. 7292–7297, 2021.

[160] S. Krebs, B. Duraisamy, and F. Flohr, "A survey on leveraging deep neural networks for object tracking," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 411–418.

[161] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61 – 88, 2020.

[162] I. Smal and E. Meijering, "Quantitative comparison of multiframe data association techniques for particle tracking in time-lapse fluorescence microscopy," *Med. Image Anal.*, vol. 24, no. 1, pp. 163–189, 2015.

[163] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong, "Enhancing detection model for multiple hypothesis tracking," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 2143–2152.

[164] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26, 2013.

[165] N. Wang, S. Li, A. Gupta, and D. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *arXiv:1501.04587*, 2015.

[166] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015, pp. 3074–3082.

[167] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015, pp. 3119–3127.

[168] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *Proc. International Conference on Machine Learning (ICML)*, 2015, 597–606.

[169] Z. Chi, H. Li, H. Lu, and M.-H. Yang, "Dual deep network for visual tracking," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 2005–2015, 2017.

[170] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Proc. European Conference on Computer Vision (ECCV)*, vol. 12356, 2020, pp. 107–122.

[171] H. W. Kuhn, "The Hungarian method for the assignment problem," *Nav. Res. Logist. Q.*, vol. 2, no. 1–2, pp. 83–97, 1955.

[172] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1420–1429.

[173] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2011, pp. 263–270.

[174] J. Son, M. Baek, M. Cho, and B. Han, "Multi-object tracking with Quadruplet Convolutional Neural Networks," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3786–3795.

[175] X. Wan, J. Wang, and S. Zhou, "An online and flexible multi-object tracking framework using long short-term memory," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1311–13 118.

[176] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie, "Trajectory factory: Tracklet cleaving and re-connection by deep Siamese Bi-GRU for multiple object tracking," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[177] C. Kim, F. Li, and J. M. Rehg, "Multi-object tracking with neural gating using bilinear LSTM," in *Proc. European Conference on Computer Vision (ECCV)*, vol. 11212, 2018, pp. 208–224.

[178] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 1, pp. 104–119, 2021.

[179] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[180] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *Proc. European Conference on Computer Vision (ECCV)*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016, pp. 263–279.

[181] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[182] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 941–951.

[183] G. Seisenberger, M. U. Ried, T. Endreß, H. Büning, M. Hallek, and C. Bräuchle, "Real-time single-molecule imaging of the infection pathway of an Adeno-associated virus," *Science*, vol. 294, no. 5548, pp. 1929–1932, 2001.

[184] P. Kukura, H. Ewers, C. M¨uller, A. Renn, A. Helenius, and V. Sandoghdar, "High-speed nanoscopic tracking of the position and orientation of a single virus," *Nat. Methods*, vol. 6, no. 12, 923—927, 2009.

[185] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[186] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[187] 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.

[188] S. Jaensch, M. Decker, A. A. Hyman, and E. W. Myers, "Automated tracking and analysis of centrosomes in early Caenorhabditis elegans embryos," *Bioinformatics*, vol. 26, no. 12, pp. i13–i20, 06 2010.

[189] J.-Y. Tinevez, N. Perry, J. Schindelin, G. M. Hoopes, G. D. Reynolds, E. Laplantine, S. Y. Bednarek, S. L. Shorte, and K. W. Eliceiri, "TrackMate: An open and extensible platform for single-particle tracking," *Methods*, vol. 115, pp. 80 – 90, 2017.

[190] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser, "Robust single-particle tracking in live-cell time-lapse sequences," *Nat. Methods*, vol. 5, no. 8, pp. 695–702, 2008.

[191] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic. Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[192] L. Yang, Z. Qiu, A. H. Greenaway, and W. Lu, "A new framework for particle detection in low-snr fluorescence live-cell images and its application for improved particle tracking," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 7, pp. 2040–2050, 2012.

[193] N. Chenouard, I. Bloch, and J. C. Olivo-Marin, "Multiple hypothesis tracking for cluttered biological image sequences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2736–3750, 2013.

[194] 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.

[195] C. Ritter, T. Wollmann, J.-Y. Lee, A. Imle, B. Müller, O. Fackler, R. Bartenschlager, and K. Rohr, "Data fusion and smoothing for probabilistic tracking of viral structures in fluorescence microscopy images," *Med. Image Anal.*, vol. 73, 102168, 2021.

[196] J. Cardinale, A. Rauch, Y. Barral, G. Szekely, and I. F. Sbalzarini, "Bayesian image analysis with on-line confidence estimates and its application to microtubule tracking," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI 2009)*, Boston, MA, USA, Jun. 2009, pp. 1091–1094.

[197] L. Yuan, Y. F. Zheng, J. Zhu, L. Wang, and A. Brown, "Object tracking with particle filtering in fluorescence microscopy images: Application to the

motion of neurofilaments in axons," *IEEE Trans. Med. Imag.*, vol. 31, no. 1, pp. 117–130, Jan. 2012.

[198] S. Särkkä, *Bayesian Filtering and Smoothing*, ser. Institute of Mathematical Statistics textbooks. Cambridge University Press, 2013, vol. 3.

[199] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *Proc. IEEE Conference on Decision and Control (CDC)*, 1980, pp. 807–812.

[200] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automat. Contr.*, vol. 24, no. 6, pp. 843–854, 1979.

[201] I. Smal, W. Niessen, and E. Meijering, "A new detection scheme for multiple object tracking in fluorescence microscopy by joint probabilistic data association filtering," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2008, pp. 264–267.

[202] S. H. Rezatofighi, S. Gould, R. Hartley, K. Mele, and W. E. Hughes, "Application of the IMM-JPDA filter to multiple target tracking in total internal reflection fluorescence microscopy images," in *Proc. Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2012, pp. 357–364.

[203] L. Feng, Y. Xu, Y. Yang, and X. Zheng, "Multiple dense particle tracking in fluorescence microscopy images based on multidimensional assignment," *J. Struct. Biol.*, vol. 173, no. 2, pp. 219 – 228, 2011.

[204] S. Coraluppi and C. Carthel, "Multi-stage multiple-hypothesis tracking," *J. Adv. Inf. Fusion*, vol. 6, no. 1, pp. 57–67, 2011.

[205] 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.

[206] 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.

[207] J. Hayashida and R. Bise, "Cell tracking with deep learning for cell detection and motion estimation in low-frame-rate," in *Proc. Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI))*, 2019, pp. 397–405.

[208] S. Nishimoto, Y. Tokuoka, T. G. Yamada, N. F. Hiroi, and A. Funahashi, "Predicting the future direction of cell movement with convolutional neural networks," *PLOS ONE*, vol. 14, no. 9, pp. 1–14, 2019.

[209] R. Sun and L. Paninski, "Scalable approximate Bayesian inference for particle tracking data," in *Proc. International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 4800–4809.

[210] Y. Yao, I. Smal, and E. Meijering, "Deep neural networks for data association in particle tracking," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2018, pp. 458–461.

[211] I. Smal, Y. Yao, N. Galjart, and E. Meijering, "Facilitating data association in particle tracking using autoencoding and score matching," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2019, pp. 1523–1526.

[212] F. de Chaumont, S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Le Montagner, T. Lagache, A. Dufour, and J.-C. Olivo-Marin, "Icy: an open bioimage informatics platform for extended reproducible research," *Nat. Methods*, vol. 9, no. 7, pp. 690 – 696, 2012.

[213] C. Collinet, M. Stöter, C. R. Bradshaw, N. Samusik, J. C. Rink, D. Kenski, B. Habermann, F. Buchholz, R. Henschel, M. S. Mueller, W. E. Nagel, E. Fava, Y. Kalaidzidis, and M. Zerial, "Systems survey of endocytosis by multiparametric image analysis," *Nature*, vol. 464, no. 7286, pp. 243–249, 2010.

[214] F. Ruhnow, D. Zwicker, and S. Diez, "Tracking Single Particles and Elongated Filaments with Nanometer Precision," *Biophys. J.*, vol. 100, no. 11, pp. 2820–2828, 2011.

[215] K. T. Applegate, S. Besson, A. Matov, M. H. Bagonis, K. Jaqaman, and G. Danuser, "plusTipTracker: Quantitative image analysis software for the measurement of microtubule dynamics," *J. Struct. Biol.*, vol. 176, no. 2, pp. 168 – 184, 2011.

[216] L. Paavolainen, P. Kankaanpää, P. Ruusuvuori, G. McNerney, M. Karjalainen, and V. Marjomäki, "Application independent greedy particle tracking method for 3D fluorescence microscopy image series," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2012, pp. 672–675.

[217] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6931–6939.

[218] J. Gao, T. Zhang, X. Yang, and C. Xu, "Deep relative tracking," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1845–1858, 2017.

[219] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang, "Learning deep neural networks for vehicle re-ID with visual-spatio-temporal path proposals," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 1918–1927.

[220] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Net.*, vol. 18, no. 5, pp. 602 – 610, 2005.

[221] W. Zhang, X. Yu, and X. He, "Learning bidirectional temporal cues for video-based person re-identification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2768–2776, Oct. 2018.

[222] E. Meijering, O. Dzyubachyk, and I. Smal, "Methods for cell and particle tracking," in *Imaging and Spectroscopic Analysis of Living Cells*, ser. Methods in Enzymology. Academic Press, 2012, vol. 504, pp. 183 – 200.

[223] 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. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2018, pp. 957–960.

[224] M. Ullah and F. Alaya Cheikh, "Deep feature based end-to-end transportation network for multi-target tracking," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 3738–3742.

[225] Y. Chen, X. Yang, B. Zhong, S. Pan, D. Chen, and H. Zhang, "CNNTracker: Online discriminative object tracking via deep convolutional neural network," *Appl. Soft Comput.*, vol. 38, pp. 1088 – 1098, 2016.

[226] H. Ma, I. Smal, J. Daemen, and T. van Walsum, "Dynamic coronary roadmapping via catheter tip tracking in X-ray fluoroscopy with deep learning based Bayesian filtering," *Med. Image Anal.*, vol. 61, 101634, 2020.

[227] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai, "Online multi-object tracking with convolutional neural networks," in *IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 645–649.

[228] L. Wang, L. Xu, M. Y. Kim, L. Rigazico, and M. Yang, "Online multiple object tracking via flow and convolutional features," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3630–3634.

[229] A. Hernandez, J. Gall, and F. Moreno-Noguer, "Human motion prediction via spatio-temporal inpainting," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3823–3832.

[230] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proc. European Conference on Computer Vision (ECCV)*, 2018, pp. 379–396.

[231] S. Farrell, D. Anderson, P. Calafiura, G. Cerati, L. Gray, J. Kowalkowski, M. Mudigonda, Prabhat, P. Spentzouris, M. Spiropoulou, A. Tsaris, J.-R. Vlimant, and S. Zheng, "The HEP.TrkX Project: deep neural networks for HL-LHC online and offline tracking," *EPJ Web Conf.*, vol. 150, 00003, 2017.

[232] Y. Zhong, C. Li, H. Zhou, and G. Wang, "Developing noise-resistant three-dimensional single particle tracking using deep neural networks," *Anal. Chem.*, vol. 90, no. 18, pp. 10 748–10 757, 2018.

[233] C. Leibig, V. Allken, M. S. Ayhan, P. Berens, and S. Wahl, "Leveraging uncertainty information from deep neural networks for disease detection," *Sci. Rep.*, vol. 7, 17816, 2017.

[234] P. Esser and E. Sutter, "A variational U-Net for conditional appearance and shape generation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8857–8866.

[235] S. Kohl, B. Romera-Paredes, C. Meyer, J. De Fauw, J. R. Ledsam, K. Maier-Hein, S. M. A. Eslami, D. Jimenez Rezende, and O. Ronneberger, "A probabilistic U-Net for segmentation of ambiguous images," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6965–6975.

[236] R. Tanno, D. Worrall, E. Kaden, A. Ghosh, F. Grussu, A. Bizzi, S. N. Sotiropoulos, A. Criminisi, and D. C. Alexander, "Uncertainty quantification in deep learning for safer neuroimage enhancement," *arXiv:1907.13418*, 2019.

[237] J. M. Hernández-Lobato and R. P. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. International Conference on Machine Learning (ICML)*, 2015, 1861–1869.

[238] H. Wang, X. SHI, and D.-Y. Yeung, "Natural-parameter networks: A class of probabilistic neural networks," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 118–126.

[239] Q. Su, X. Liao, C. Chen, and L. Carin, "Nonlinear statistical learning with truncated Gaussian graphical models," in *Proc. International Conference on Machine Learning (ICML)*, 2016, pp. 1948–1957.

[240] B. J. Frey and G. E. Hinton, "Variational learning in nonlinear Gaussian belief networks," *Neural Comput.*, vol. 11, no. 1, pp. 193–213, 1999.

[241] K. Lee, Z. Wang, B. Vlahov, H. Brar, and E. A. Theodorou, "Ensemble Bayesian decision making with redundant deep perceptual control policies," in *Proc. IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 831–837.

[242] S. S. Gu, Z. Ghahramani, and R. E. Turner, "Neural adaptive sequential Monte Carlo," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[243] W. Gong, Y. Li, and J. M. Hernández-Lobato, "Meta-learning for stochastic gradient MCMC," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[244] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 2016, 265–283.

[245] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. D. Hoffman, and R. A. Saurous, "TensorFlow distributions," *arXiv:1711.10604*, 2017.

[246] J.-Y. Lee, M. Cortese, U. Haselmann, K. Tabata, I. Romero-Brey, C. Funaya, N. L. Schieber, Y. Qiang, M. Bartenschlager, S. Kallis, C. Ritter, K. Rohr, Y. Schwab, A. Ruggieri, and R. Bartenschlager, "Spatiotemporal coupling of the Hepatitis C virus replication cycle by creating a lipid droplet-proximal membranous replication compartment," *Cell Rep.*, vol. 27, pp. 3602–3617.e5, 2019.

[247] V. O. Chagin, C. S. Casas-Delucchi, M. Reinhart, L. Schermelleh, Y. Markaki, A. Maiser, J. J. Bolius, A. Bensimon, M. Fillies, P. Domaing, Y. M. Rozanov, H. Leonhardt, and M. C. Cardoso, "4D Visualization of replication foci in mammalian cells corresponding to individual replicons," *Nat. Commun.*, vol. 7, 11231, 2016.

[248] E. M. M. Manders, F. J. Verbeek, and J. A. Aten, "Measurement of co-localization of objects in dual-colour confocal images," *J. Microsc.*, vol. 169, no. 3, pp. 375–382, 1993.

[249] S. Wang, E. T. Arena, J. T. Becker, W. M. Bement, N. M. Sherer, K. W. Eliceiri, and M. Yuan, "Spatially adaptive colocalization analysis in dual-color fluorescence microscopy," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4471–4485, 2019.

[250] S. T. Low-Nam, K. A. Lidke, P. J. Cutler, R. C. Roovers, P. M. P. van Bergen en Henegouwen, B. S. Wilson, and D. S. Lidke, "ErbB1 dimerization is promoted by domain co-confinement and stabilized by ligand-binding," *Nat. Struct. Mol. Biol.*, vol. 18, pp. 1244–1249, 2011.

[251] Y. Qiang, J. Y. Lee, R. Bartenschlager, and K. Rohr, "Colocalization analysis and particle tracking in multi-channel fluorescence microscopy images," in

Proc. IEEE International Symposium on Biomedical Imaging (ISBI), 2017, pp. 646–649.

[252] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[253] X. Wang, L. Bo, and L. Fuxin, "Adaptive wing loss for robust face alignment via heatmap regression," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6970–6980.

[254] Z. Luo, Z. Wang, Y. Huang, L. Wang, T. Tan, and E. Zhou, "Rethinking the heatmap regression for bottom-up human pose estimation," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 259–13 268.

[255] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Comput. Methods Biomech. Biomed. Eng.: Imaging Vis.*, vol. 6, no. 3, pp. 283–292, 2018.

[256] B. T. Eichenberger, Y. Zhan, M. Rempfler, L. Giorgetti, and J. Chao, "deepBlink: threshold-independent detection and localization of diffraction-limited spots," *Nucleic Acids Res.*, vol. 49, no. 13, pp. 7292–7297, 2021.

[257] C. Ritter, R. Spilger, J.-Y. Lee, R. Bartenschlager, and K. Rohr, "Deep learning for particle detection and tracking in fluorescence microscopy images," in *Proc. IEEE International Symposium on Biomedical Imaging (ISBI)*, 2021, pp. 873–876.