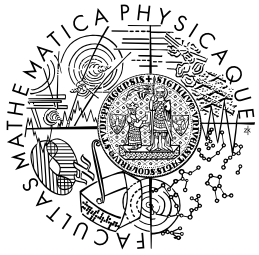


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

**BACHELOR THESIS**

Vítězslav Lužný

# Analysis and reconstruction of dynamical speckle patterns in interferometric scattering microscopy

Institute of Physics of Charles University

Supervisor of the bachelor thesis: Mgr. Marek Piliarik, Ph.D.
Study programme: Mathematics
Study branch: Mathematical modelling

Prague 2023

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date .....  .....

Author's signature

I would like to thank my supervisor Mgr. Marek Piliarik, Ph.D. as well as the whole Nano-Optics group at the Institute of Photonics and Electronics, Czech Academy of Sciences for their supervision, help, and equipment necessary for this work.

Title: Analysis and reconstruction of dynamical speckle patterns in interferometric scattering microscopy

Author: Vítězslav Lužný

Institute: Institute of Physics of Charles University

Supervisor: Mgr. Marek Piliarik, Ph.D., Department of Chemical Physics and Optics

Abstract: Optical microscopy has a diffraction limit that prevents imaging of objects smaller than hundreds of nanometers, making it challenging to observe certain biological samples. Interferometric scattering microscopy (iSCAT) has shown promise in overcoming this limit by detecting and enhancing the scattering light from nanoparticles. However, speckle patterns produced by a large number of nanoparticles close to the diffraction limit make analysis difficult. In this thesis, we demonstrate the potential of using Deep Neural Networks (DNNs) with fast data simulation to analyze these difficult samples. With one of our models achieved 81.47% accuracy in classifying simulated image sequences of fluctuating speckle patterns with respect to particle count. And another one of our models localizes particle positions closely resembling the ground truth data.

Keywords: single molecule, fluctuation, diffusion, iSCAT

# Contents

# Introduction

Prior to the invention of the microscope, the functioning of the human body and other living organisms was largely unknown and often attributed to myths. The development of the first microscopes in the 17th century revolutionized our understanding of biological processes and led to significant advancements in the fields of biology and medicine. Despite these advancements, there remain certain biological processes that are not yet fully understood due to the limitations of optical microscopy, specifically its resolution, known as the diffraction limit.

There are several alternatives to optical microscopy, each with its own set of constraints that limit their applicability for dynamic biological samples. For instance, electron microscopy can achieve a resolution as high as 50 picometers (Erni et al. [2009]) but requires that samples be measured in a vacuum. Cryogenic electron microscopy allows for the study of static biological specimens but does not permit observation of dynamic processes.

Another approach involves scanning a surface with a mechanical probe and detecting its movement. Atomic force microscopy can achieve a resolution below one nanometer (Giessibl [1995]) but is limited by its temporal resolution and can only scan surfaces.

Optical microscopes remain the most suitable tool for studying dynamic biological samples, necessitating the development of techniques to overcome the diffraction limit within them.

Fluorescence microscopy can achieve localization precision well below the diffraction limit (Thompson et al. [2002]) by filtering emitted light from fluorophores and fitting their point spread function (PSF) to the image. However, this technique is limited by photobleaching, which reduces its spatial and temporal resolution.

Interferometric scattering microscopy (iSCAT) is conceptually similar to fluorescence microscopy, however, instead of fluorescence it detects Rayleigh scattered light from nano-particles and enhances it through interference with a reference light field Amos and Amos [1991]. Rayleigh scattering however has the same wavelength as the incident light making it difficult to filter the background.

The last two techniques usually rely on a sufficient distance between localized particles and their contrast. High particle densities produce complex speckle patterns, that are very difficult to impossible to analyze using current techniques.

In this thesis, we will be trying to deal with these speckle patterns in the context of iSCAT microscopy.

First, we review the fundamentals of iSCAT microscopy and explore the potential application of Deep Learning techniques for iSCAT data analysis.

Then, in the practical part, we simulate iSCAT image sequences of speckle patterns from diffusing nano-particles and use them to train DNNs for their analysis and particle localization.

# 1. Review part

## 1.1 Microscopy

### 1.1.1 Diffraction limit in Optical microscopy

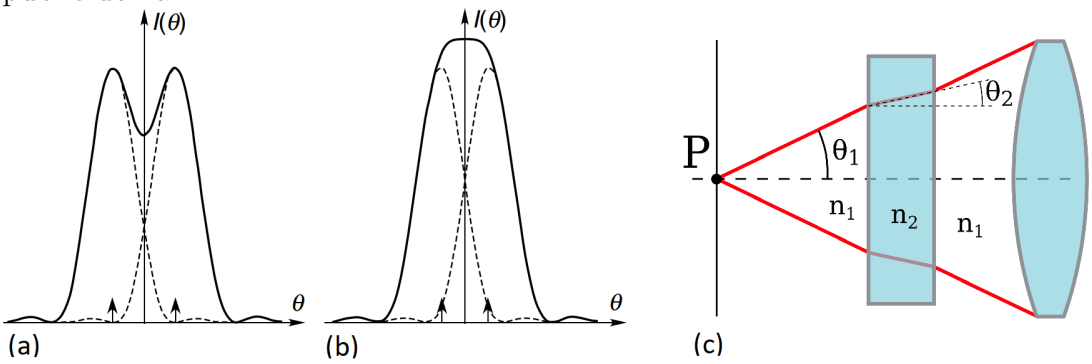
In 1873 Ernst Abbe described the modern theory of image formation in the microscope in order to explore ways for improving the optical performance of the microscope.

There he derived a resolution limit for microscopes, called the diffraction limit ( Lipson et al. [1981]).

$$s = \frac{\lambda}{2n \cdot \sin\theta} = \frac{\lambda}{2 \cdot NA} \tag{1.1}$$

where $s$ is the diffraction limit, $n$ is the refractive index of the medium, $\theta$ is the half-angle of the range over which the system can accept the light, $n \cdot \sin\theta$ is called the numerical aperture ($NA$) and $\lambda$ is the wavelength of light, see Figure 1.1.

Figure 1.1: (a) resolvable light sources, (b) unresolvable light sources being closer to each other than the Rayleigh criterion (a modification of the diffraction limit), taken from Lipson et al. [1981], (c) $NA = n_1 \cdot \sin\theta_1 = n_2 \cdot \sin\theta_2$, taken from the public domain




### 1.1.2 Fluorescence microscopy

#### Fluorescence

The first recorded observation of fluorescence was in 1845 by Sir Frederik William Herschel, who noticed that a quinine solution emits a blue light under certain conditions (Renz [2013]).

Fluorescence is the excitation cycle, in which electrons in a substance move to a higher energy level due to radiation absorption and then return back while causing light emission of a longer wavelength and smaller energy than the incident radiation.

Substances capable of this effect are called fluorophores.

After a certain number of excitation cycles the fluorophore molecules get chemically altered resulting in the loss of fluorescence ability. This effect is called photobleaching (Demchenko [2020]).

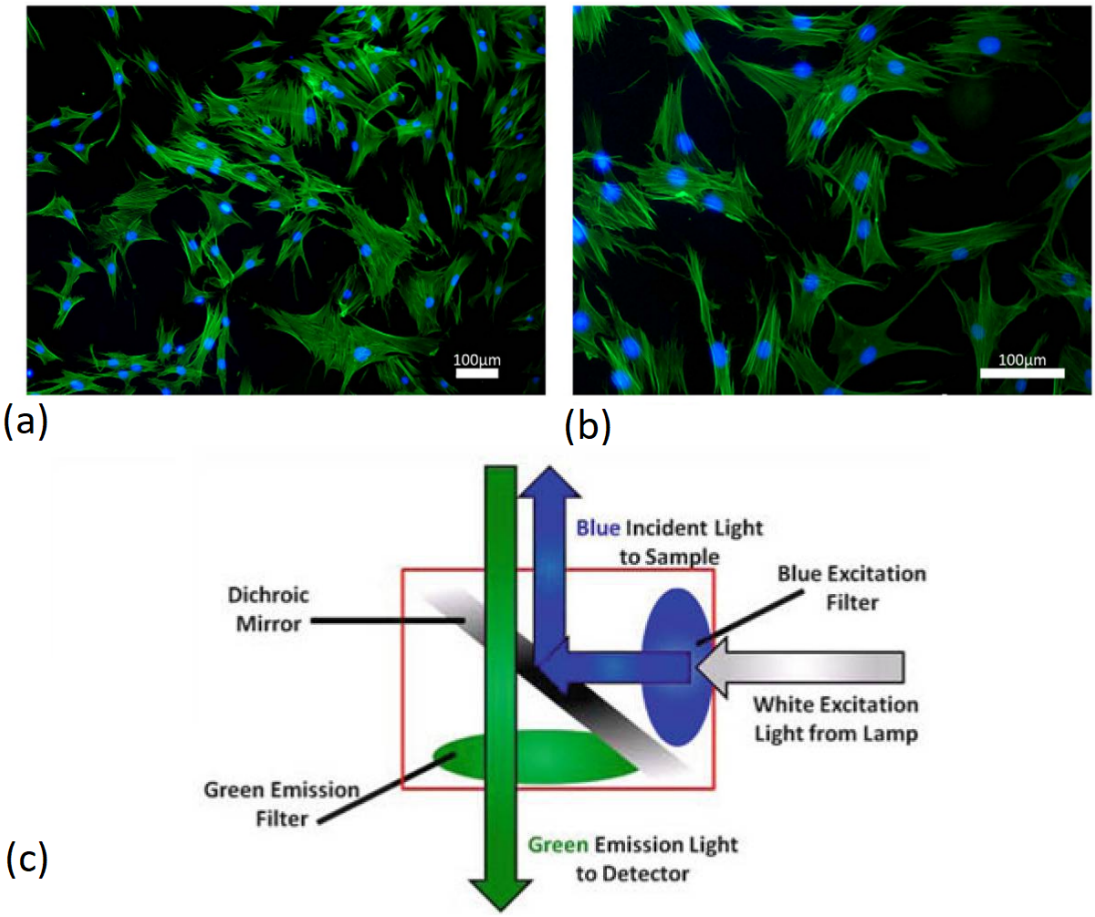
### History and properties

Fluorescence microscopy, where we detect emitted light from fluorophores, is very popular in biological applications, see Figure 1.2a,b.

The first fluorescence microscope was developed in 1911 by Oskar Heimstaedt. But there was a problem that the emitted fluorescence was much weaker than the incident and reflected light (Rusk [2009]).

Therefore in 1929 Philip Ellinger and August Hert further improved it with a configuration known as epi-illumination. There the light beam passes through the specimen in the opposite direction than the emitted light is detected, Figure 1.2c. The fact that the emitted light has a different wavelength than the incident light can be used to suppress it using an emission filter that allows only a certain wavelength to pass, resulting in an almost perfect background suppression.

Figure 1.2: (a) and (b) images of breast cancer cells in a fluorescence microscope, taken from Kazantseva et al. [2016] (c) Epifluorescence, taken from Webb and Brown [2012]




Further advancements in fluorescence microscopy reached single-molecule sen-

sitivity in 1980' sparking a new era in optical microscopy. With such sensitivity, the fluorescent signal breaks down to individual contributions of single molecules which was made possible to be imaged one at a time (Moerner and Kador [1989]).

When the distance between fluorophores is bigger than the diffraction limit, then each fluorophore gets displayed with intensity distribution approximated by a Gaussian function. This fact can be used to localize those fluorophores with potentially unlimited precision by fitting the function over the displayed image (Thompson et al. [2002]):

$$\left\langle (\Delta x)^2 \right\rangle = \frac{s^2}{N} + \frac{a^2}{12N} + \frac{4\sqrt{\pi} s^3 b^2}{aN^2} \tag{1.2}$$

where $s$ is the diffraction limit (1.1), $a$ is the pixel size, $N$ is the number of detected photons and $b$ is the background noise.

As we can see the main constraint to the localization precision is the number of detected photons. But there are only so many photons a fluorophore can emit before suffering from the effect of photobleaching.

#### Summary and limitations

To conclude, while fluorescence microscopy has a perfect background suppression thanks to the difference in wavelength between the incident and emitted light, the photobleaching effect ultimately constrains the localization precision of the fluorophores.

To reliably localize fluorophores, they must have sufficient distance from each other. This fact significantly limits potential use cases.

### 1.1.3 Interferometric Scattering microscopy (iSCAT)

#### Rayleigh scattering

Elastic scattering of light appears when a photon interacts with a particle in such a way that its direction changes while preserving the total kinetic energy of the system.

A special case of this effect called Rayleigh scattering occurs when the particles are smaller than the wavelength of light. A model of the Rayleigh scattering considers particles as point-induced dipoles of polarizability $\alpha$. The incident oscillating electromagnetic field results in dipole radiation called scattered light having the intensity (Seinfeld and Pandis [2006]):

$$I = I_0 \frac{1 + \cos\theta^2}{2R^2} \left(\frac{2\pi}{\lambda}\right)^4 \left(\frac{n^2 - 1}{n^2 + 2}\right)^2 \left(\frac{d}{2}\right)^6$$

where $I$ is the intensity of scattered light, $I_0$ is the intensity of incident light with wavelength $\lambda$, $\theta$ is the scattering angle, $n$ is the refractive index, $R$ is the distance from the particle and $d$ is the diameter of the particle.

#### History and properties

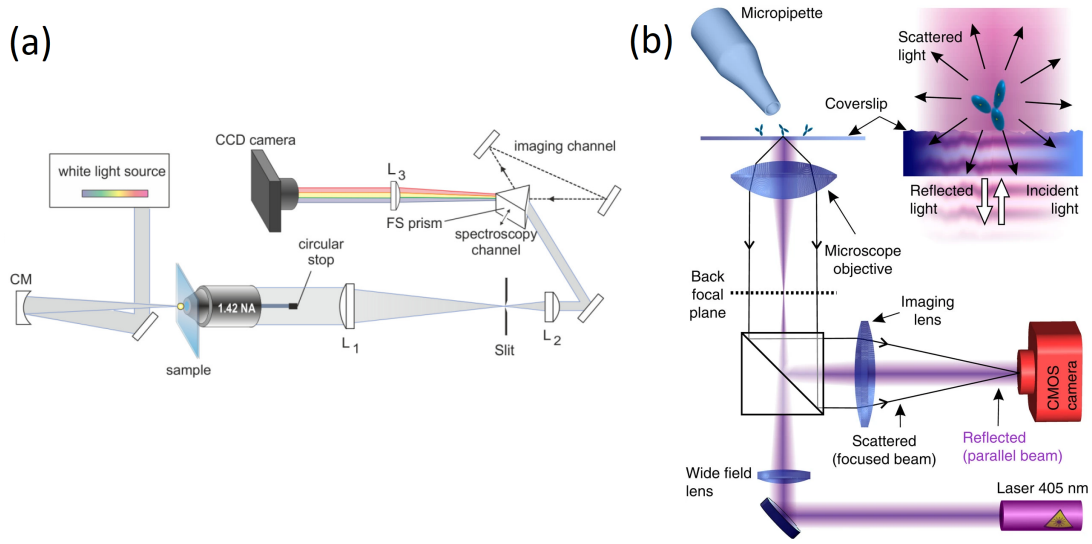
As scattering is the most fundamental interaction of light with matter, the history of microscopy is the history of scattering intensity detection. Similarly, as in

fluorescence, it is possible to block the illuminating light and detect only the scattered light in a configuration coined a dark field. In this case, there are no filters to separate the scattered light from the incident illumination based on its wavelength, because there is no wavelength shift. However, it is possible to separate the incident beam and the scattered beam geometrically, see Figure 1.3a.

There was great progress in improving the dark-field contrast over the last decades, and recent dark-field imaging configurations offer contrast separation comparable to the fluorescent microscope (Weigel et al. [2014]).

This approach was further improved by enhancing the scattered light by interfering it with a reference beam (Amos and Amos [1991]), this became known as *iSCAT microscopy*, see Figure 1.3b.

Figure 1.3: (a) Scheme of a Dark-field microscope, taken from Weigel et al. [2014] and (b) Scheme of an iSCAT microscope, taken from Piliarik and Sandoghdar [2014]




In iSCAT, the incident light is partially reflected from the glass surface and then scattered from the sample. The reflected light, often called reference light, interferes in the detector with the scattered light. The detected light intensity can then be described as (Lindfors et al. [2004]):

$$I_{det} = |E_r|^2 + |E_s|^2 + 2E_r E_s \cos\phi$$

$$E_r >> E_s$$

where $I_{det}$ is the detected light intensity, $E_r$ is the electric intensity of the reference light, $E_s$ is the electric intensity of the scattered light and $\phi$ is the difference in phase between them. The reference light has usually a much higher intensity than the scattered light.

Since the reference light wave can be approximated as a constant plane light wave, it can be easily removed from the image by normalizing it. The dominant part of the result is then the interference part $2E_r E_s \cos\phi$, where $\phi$ periodically oscillates with the distance from the scattering particles resulting in a pattern of light and dark circles around the particles.

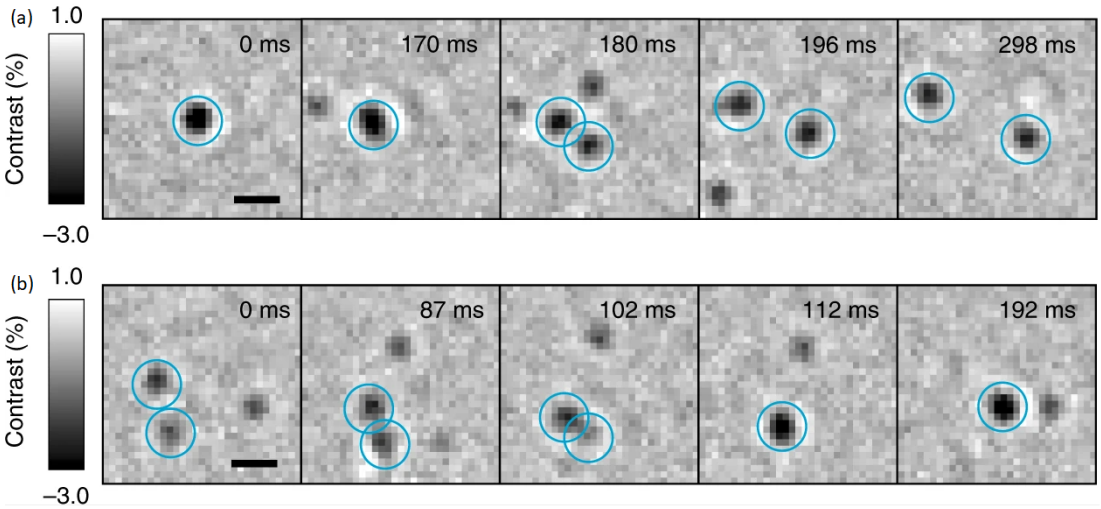
Individual particles with sufficient distance from each other can then be localized the same way as in fluorescence microscopy with precision being limited by the same formula (1.2). The main constraint still being the number of detected photons $N$. However, in fluorescence microscopy, this was limited by a photobleaching effect, here we can theoretically increase this variable as much as we want by using a stronger laser for the incident light wave. The only limitation in practice is the possibility of heat from the laser destroying the sample.

### Localization of diffusing particles

One of many possible use cases for iSCAT microscopy is label-free single-molecule tracking of membrane-associated proteins diffusing on supported lipid bilayers, as demonstrated by several papers (Heermann et al. [2021], Foley et al. [2021]).

This technique makes it possible to study biomolecular mechanisms of membrane-associated biological systems, see Figure 1.4.

Figure 1.4: iSCAT image sequences of protein complex dissociation (a) and association (b) events, taken from Foley et al. [2021]




However, current localization techniques relying on PSF fitting require a sufficient distance between particles that are being tracked. Making it impossible to study samples with particle densities $> 1\,\mu\text{m}^{-2}$ (Foley et al. [2021]).

### Summary and limitations

In iSCAT microscopy, background suppression is more difficult than in fluorescence microscopy, but the localization precision can be potentially much higher since we are not limited by photobleaching. iSCAT microscopy is also able to detect non-fluorophore particles, making its potential applications much more diverse.

Current techniques for particle localization still require sufficient distance between them. This fact significantly limits potential use cases.

## 1.2 Brownian motion

In 1827 Robert Brown discovered that particles of pollen immersed in water are being randomly displaced in time. This phenomenon is caused by the force of atomic bombardment with randomly changing direction (Brown [1828]).

This process can be approximated as a continuous random walk (Knight [1962]), where the displacement of a particle in each dimension of movement after time $t$ is normally distributed (Einstein and Fürth [2011]):

$$\Delta x \sim \mathcal{N}(0, \sigma^2)$$
$$\sigma^2 = 2Dt$$

where $\Delta x$ is the particle's displacement in one dimension, $D$ is its diffusion coefficient and $t$ is the elapsed time.

## 1.3 Machine Learning

### 1.3.1 Supervised Learning

In supervised machine learning, we are trying to approximate a certain function from a limited set of its input-output pairs called the *training dataset* (Russell and Norvig [2016]).

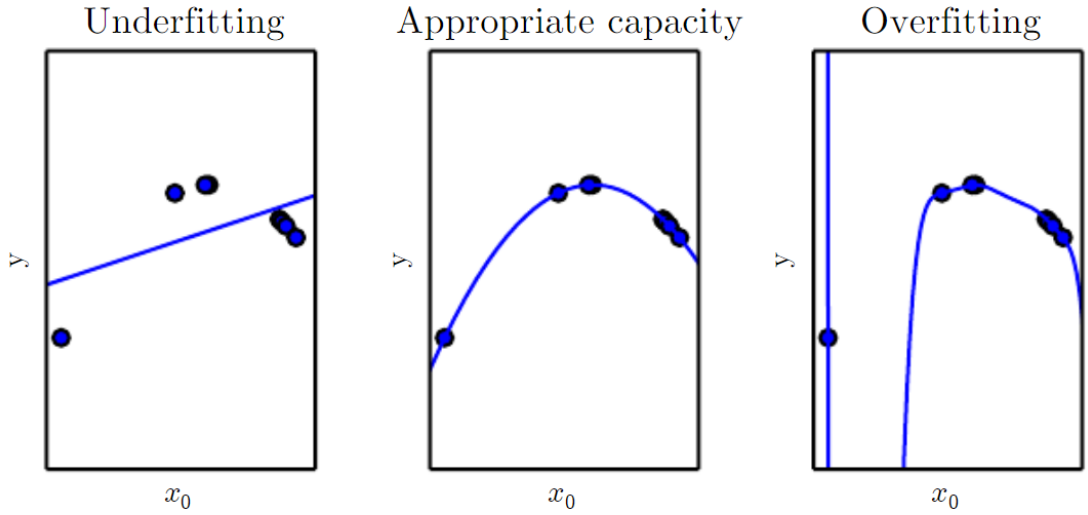
We usually define a so-called loss function, which we use to evaluate our model's predictions with respect to our dataset. Then we optimize the parameters of the model to minimize the loss.

However, if we minimize the loss too well, then we might face a problem called overfitting, where our model performs well on the training dataset, but does not generalize well on unseen data (see Figure 1.5).

The opposite problem of not performing well even on the training data is called underfitting.

Techniques used for preventing overfitting are called regularizations.

Figure 1.5: Illustration of underfitting and overfitting, taken from Goodfellow et al. [2016]



### 1.3.2 Minibatch Stochastic Gradient Descent (SGD)

Usually, our loss function will take the form:

$$L\left(f\left(\mathbb{X};\mathbf{w}\right),\mathbb{Y}\right)=\frac{1}{N}\sum_{i=0}^{N}L\left(f\left(x_i;\mathbf{w}\right),y_i\right)$$

where $L$ is the loss function, $\mathbb{X}=\{x_i\}_{i=0}^{N}, \mathbb{Y}=\{y_i\}_{i=0}^{N}$, $(x_i,y_i)$ are the input-output pairs from the training dataset, $\mathbf{w}$ are parameters (weights) of the model that we want to optimize, and $f$ is the model being trained.

In order to optimize model parameters $\mathbf{w}$ we can use the Minibatch Stochastic Gradient Descent algorithm, where we approximate the gradient of the loss with respect to the weights as an average of its gradients across a randomly chosen batch of samples (Goodfellow et al. [2016]):

$$\nabla_{\mathbf{w}}L(f(\mathbb{X};\mathbf{w}),\mathbb{Y})\approx\frac{1}{|\mathbb{B}|}\sum_{(x,y)\in\mathbb{B}}\nabla_{\mathbf{w}}L(f(x;\mathbf{w}),y)$$

where $\mathbb{B}\subset\{(x_i,y_i)\}_{i=0}^{N}$ is a random subset of the training dataset, $\mathbb{B}$ is called a *batch*.

**Algorithm 1** Minibatch Stochastic Gradient Descent

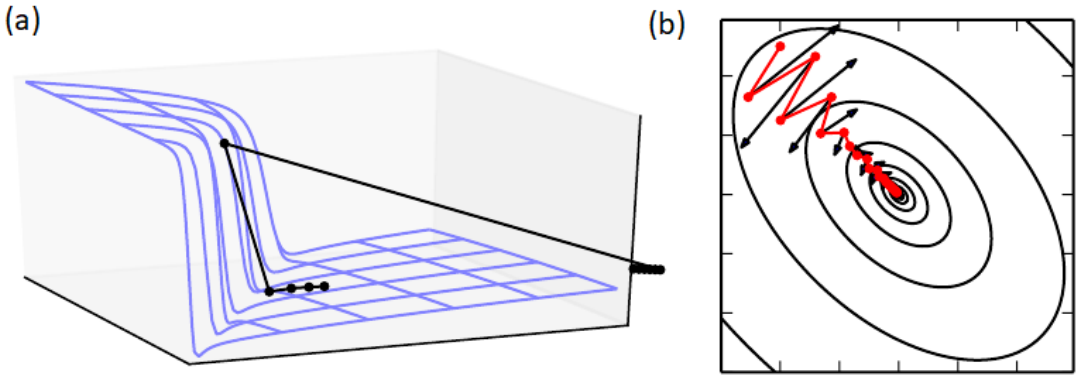
**Require:** $\{x_i,y_i\}_{i=0}^{N}$ ▷ Training dataset
**Require:** $M\in\mathbb{N}$ ▷ Number of algorithm iterations
**Require:** $B\in\mathbb{N}$ ▷ Batch size
**Require:** $\forall\alpha\in\{\alpha_i\}_{i=0}^{M}:\alpha\geq 0$ ▷ Sequence of learning rates
  $\mathbf{w}\leftarrow Rand$ ▷ Random initialization of weights
  **for** iter = 0, 1, ..., M **do**
    Randomly shuffle $\{x_i,y_i\}_{i=0}^{N}$
    **for** b = 0, 1, ..., $\frac{N}{B}-1$ **do**
      $\mathbf{w}\leftarrow\mathbf{w}-\alpha_{iter}\cdot\frac{1}{B}\sum_{i=b\cdot s}^{(b+1)\cdot s}\nabla_{\mathbf{w}}L(f(x_i;\mathbf{w}),y_i)$ ▷ Gradient descend
    **end for**
  **end for**

### 1.3.3 Adaptive Momentum Estimation (Adam)

Empirically, SGD does not perform well in some cases of loss gradients. These cases include, for example, steep gradients and saddle regions (see Figure 1.6a).

For this reason, modifications of SGD were developed, usually using some form of scaling the gradients and using so-called *momentum* (= Some weighted average of gradients from previous iterations) in addition to them (see Figure 1.6b).

Figure 1.6: (a) Example of a loss, where SGD does not perform well, (b) Example of SGD with momentum, black arrows showing the gradients and red lines the optimization, taken from Goodfellow et al. [2016]




These modifications were perfected into a so-called *Adam* algorithm (Kingma and Ba [2017]). This algorithm empirically performs well in most cases and is currently one of the most widely used optimization algorithms in Deep Learning.

**Algorithm 2** Adaptive Momentum Estimation (Adam)

**Require:** $\{x_i, y_i\}_{i=0}^{N}$ ▷ Training dataset
**Require:** $M \in \mathbb{N}$ ▷ Number of algorithm iterations
**Require:** $B \in \mathbb{N}$ ▷ Batch size
**Require:** $\forall \alpha \in \{\alpha_i\}_{i=0}^{M} : \alpha \geq 0$ ▷ Sequence of learning rates
**Require:** $\beta_1, \beta_2 \in [0, 1)$ ▷ Exponential decay rates for the momentum
**Require:** $\epsilon > 0$ ▷ A small constant for numerical stability
  $\mathbf{w} \leftarrow Rand$ ▷ Random initialization of weights
  $\mathbf{m} \leftarrow 0$ ▷ 1st moment vector
  $\mathbf{v} \leftarrow 0$ ▷ 2nd moment vector
  $t \leftarrow 0$ ▷ Timestep
  **for** iter = 0, 1, . . . , M - 1 **do**
    Randomly shuffle $\{x_i, y_i\}_{i=0}^{N}$
    **for** b = 0, 1, . . . , $\frac{N}{B} - 1$ **do**
      $t \leftarrow t + 1$
      $\mathbf{g} \leftarrow \frac{1}{B} \sum_{i=b \cdot s}^{(b+1) \cdot s} \nabla_{\mathbf{w}} L(f(x_i; \mathbf{w}), y_i)$ ▷ Gradient approximation
      $\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{m} + (1 - \beta_1) \cdot \mathbf{g}$ ▷ Update 1st moment
      $\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{v} + (1 - \beta_2) \cdot \mathbf{g}^2$ ▷ Update 2nd moment
      $\hat{\mathbf{m}} \leftarrow \frac{\mathbf{m}}{1 - \beta_1^t}$ ▷ Unbiased estimate of 1st moment
      $\hat{\mathbf{v}} \leftarrow \frac{\mathbf{v}}{1 - \beta_2^t}$ ▷ Unbiased estimate of 2nd moment
      $\mathbf{w} \leftarrow \mathbf{w} - \alpha_{iter} \cdot \frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{v}}} + \epsilon}$ ▷ Update weights
    **end for**
  **end for**

### 1.3.4 Multi-class Logistic Regression

The inputs to Multi-class Logistic regression are called *features* and we try to classify their combinations into $K$ classes by optimizing a matrix multiplication

(that can be interpreted as $K$ weighted averages of input values) and vector addition. Finally, we apply *Softmax* function on the result, getting probability prediction for each class. (Functions applied to layer outputs are called *Activation functions*) (Bishop [2016]).

$$f(\mathbf{x_i}; \mathbf{W}, \mathbf{b}) = \text{softmax}\left(\mathbf{x_i}^T\mathbf{W} + \mathbf{b}\right)$$

where $\mathbf{x_i} \in \mathbb{R}^m$ is the feature vector of the ith element from the training set, $\mathbf{W}$ and $\mathbf{b}$ are parameters of the model, $\mathbf{W} \in \mathbb{R}^{m \times k}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^k$ is the bias vector.

$$\left(\text{softmax}\left(\mathbf{z}\right)\right)_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=0}^{K} e^{\mathbf{z}_j}}$$

the softmax function makes it possible to interpret the output of the model as a probability.

Since cross-entropy is usually used for comparing probability distributions, a popular loss for classification tasks is the so-called *Categorical Cross-entropy*, in case of each element being assigned to a single class we call it the *Sparse Categorical Cross-entropy*:

$$\text{CCE}\left(f\left(\mathbb{X}; \mathbf{w}\right), \mathbb{Y}\right) = -\frac{1}{N}\sum_{i=0}^{N}\sum_{k=0}^{K} \delta_{y_i k} \cdot \log f(\mathbf{x_i}; \mathbf{w})_k = -\frac{1}{N}\sum_{i=0}^{N} \log f(\mathbf{x_i}; \mathbf{w})_{y_i}$$

where CCE is the Categorical cross-entropy, $\mathbb{X} = \{\mathbf{x_i}\}_{i=0}^{N}, \mathbb{Y} = \{y_i\}_{i=0}^{N}$, $(\mathbf{x_i}, y_i)$ are the input-output pairs from the training dataset, $y_i$ being the index of the class (between 0 and $K$), $\delta_{y_i k}$ represents the true probability of whether $\mathbf{x_i}$ class is assigned to the class $k$ (1.0 if $k = y_i$ and 0.0 otherwise).

The layer, where we apply a matrix multiplication to the input, is called a *Dense layer*.

In order to improve the performance of Logistic regression, a technique called *Feature engineering* is employed, where we transform the features to make them easier to process for the model.

Feature engineering techniques include *one-hot encoding* of integer values or *scaling* of real values for a standardized mean and variance (usually 0 and 1).

### 1.3.5 Multi-Layer Perceptron

The most simple type of model in the Deep Learning family is the *Multi-layer Perceptron* (MLP). This model is inspired by *Logistic regression* and builds upon it by adding more dense layers between the input and output (these layers are called *hidden*).

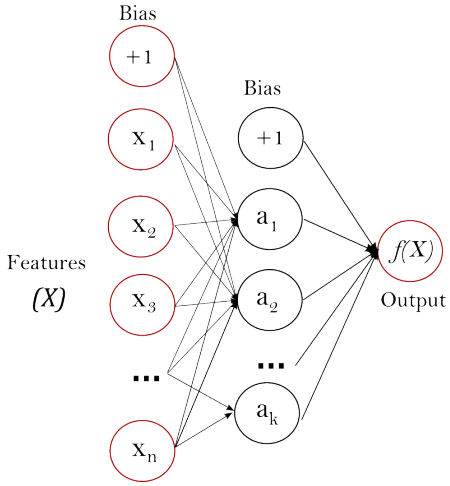
MLP with a single hidden layer would then look like as (Bishop [2016]):

$$f(\mathbf{x_i}; \mathbf{W^h}, \mathbf{b^h}, \mathbf{W^o}, \mathbf{b^o}) = a_o\left(\left(a_h\left(\mathbf{x_i}^T\mathbf{W^h} + \mathbf{b^h}\right)\right)^T\mathbf{W^o} + \mathbf{b^o}\right)$$

where $a_o$ is the activation function of the output layer and $a_h$ is the nonlinear activation function of the hidden layer.

The MLP can be visualized as a neural network, see Figure 1.7.

Figure 1.7: MLP with a single hidden layer visualization, taken from Pedregosa et al. [2011]




We can interpret the hidden layers as models performing the feature engineering for us. Their outputs are therefore sometimes called *Feature maps*.

*The universal approximation theorem* states that MLP with just a single hidden layer with a nonlinear activation function can approximate any continuous function to an arbitrary degree of accuracy (Russell and Norvig [2016]).

As we can see, the MLP should be able to approximate any continuous function. However, empirically it does not perform very well on dimensional data such as images and videos.

### 1.3.6 Convolutional Neural Networks

These types of models contain so-called *Convolutional layers*.

The inputs and outputs of these layers have (n+1) dimensions, with one dimension being called *channels*, *filters*, or *depth*.

There are input channels times output channels fixed size n-dimensional kernels, which get convolved with a specific channel of the input, with the result being added to a specific channel of the output.

A two-dimensional convolution layer without stride would then look like (Goodfellow et al. [2016]):

$$(K \star V)_{i,j,o} = \sum_{m,n,c} V_{i+m,j+n,c} K_{m,n,c,o}$$

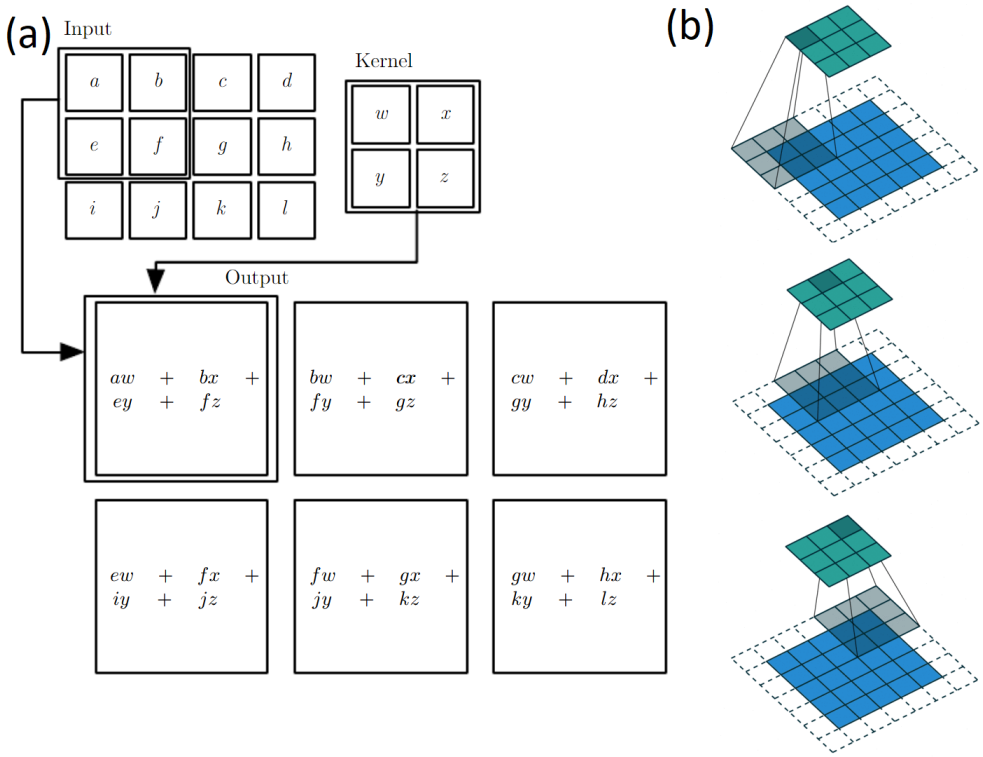
where $K \in \mathbf{R}^{M,N,C,O}$ is the kernel, $V \in \mathbf{R}^{I,J,C}$ is the input to the layer with $I, J$ dimensions and $C$ channels, $(K \star V) \in \mathbf{R}^{I,J,O}$ is the output from the layer (See Figure 1.8a).

In case we will use stride, then the processed tensor dimensions will be downscaled (See Figure 1.8n).

The better performance of these layers in processing dimensional data, when compared to *dense* layers, is usually attributed to the built-in shift-invariance in processing.

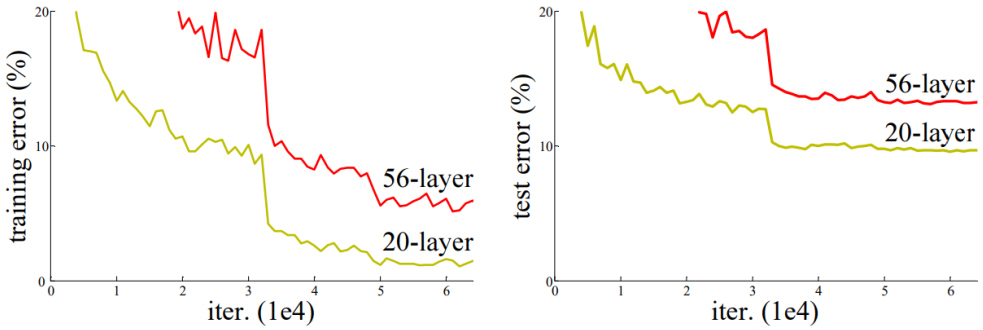
However, the more convolutional layers we stack, the less smooth its gradients will become (See Figure 1.10c). So while we increase the capacity of the model,

Figure 1.8: (a) Convolutional layer computation, taken from Goodfellow et al. [2016], (b) Convolution with stride and padding, taken from vdumoulin [2019]




we make it more difficult to optimize by increasing the chances of getting stuck in local minimums (He et al. [2015]) (See Figure 1.9).

Figure 1.9: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" convolutional networks. The deeper network has higher training error, and thus test error. Taken from He et al. [2015]

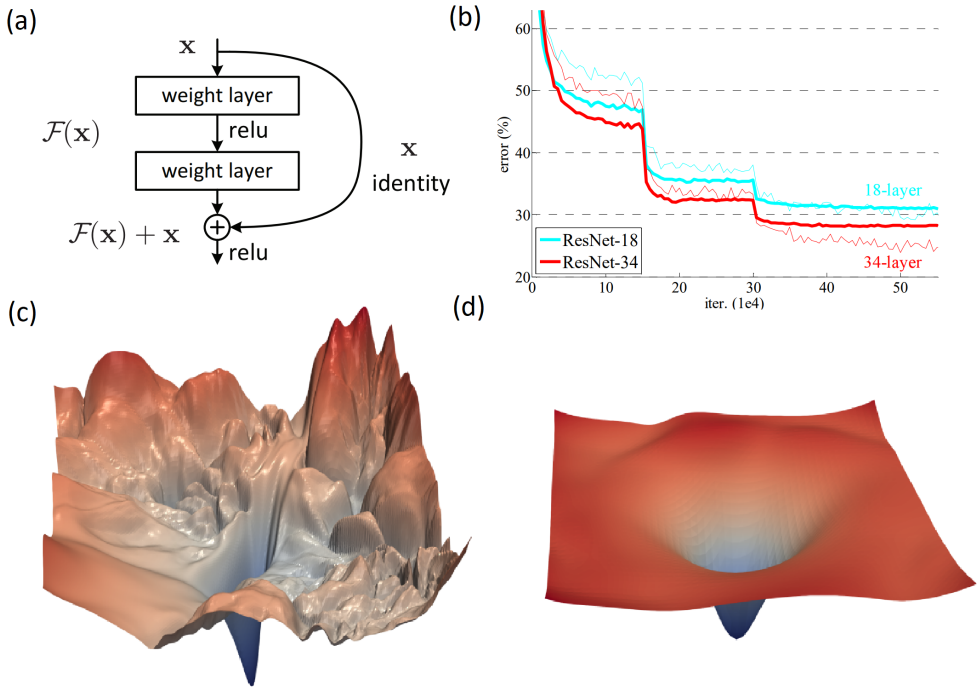



### 1.3.7 Residual Networks

The aforementioned problem was solved with the invention of so-called *residual blocks*. There we process the input to the block with just a few convolutional layers and activation functions, usually two or three, then we add the original input to the processed one, this step being called *the residual* or *skip connection*, finally one more activation function is usually performed at the end of the block (See Figure 1.10a).

When stacking these blocks we usually don't face the same problems as with raw convolutional layers (See Figure 1.10b,d).

Figure 1.10: **a,b** (a) Residual block. (b) Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of networks of 18 and 34 layers. Taken from He et al. [2015]. **c,d** The loss surfaces of ResNet-56 (c) without and (d) with residual connections. Taken from Li et al. [2018].

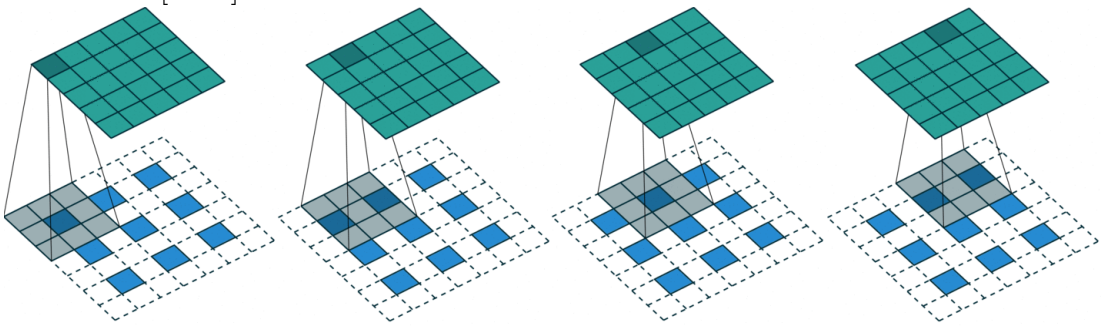



## 1.3.8 U-Net Architecture

### Transposed Convolutional Layers

Sometimes, however, we would instead like to upscale output dimensions. To achieve this, an analog to a convolutional layer called *transposed convolutional layer* was developed. When we use a stride here, then we upscale output dimensions (See Figure 1.11).

Figure 1.11: Transposed convolutional layer with stride and padding, taken from vdumoulin [2019]

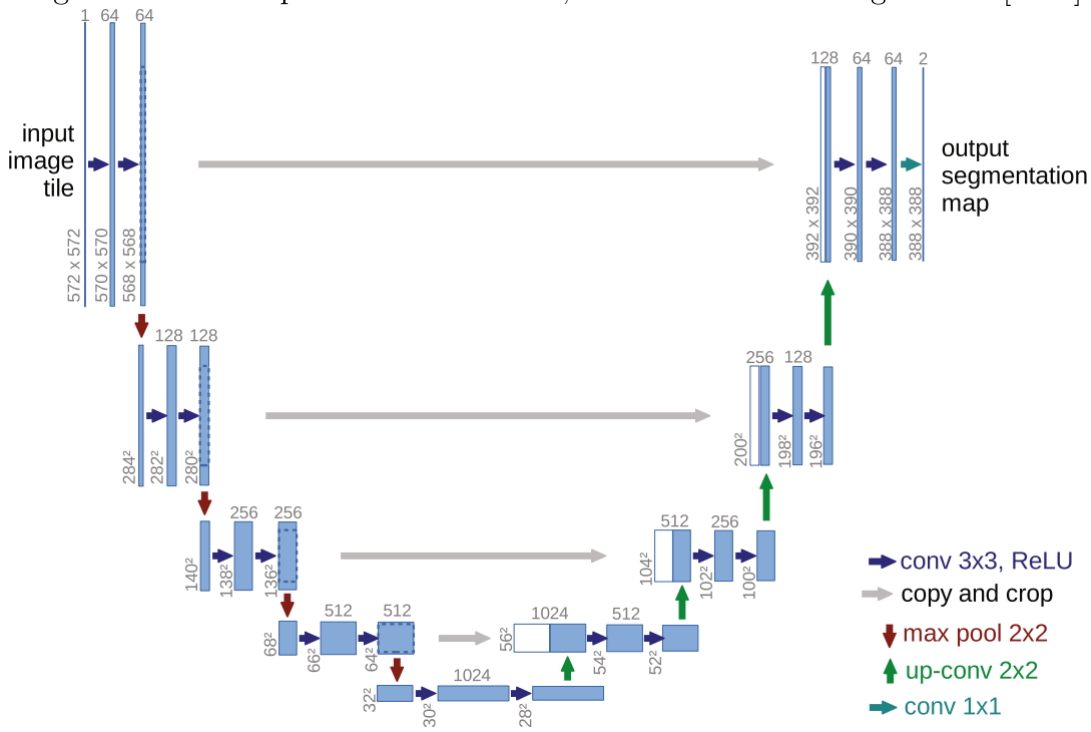


### Architecture Overview

U-Net model tries to solve segmentation tasks, where we want to find objects in images. This problem can be formulated as a classification of each pixel into background/foreground or in multi-object detection into background/object class.

Its architecture uses a classification convolutional neural network as a backbone. First, we select a few pre-trained feature maps from the backbone, then we take the smallest one and process it further using convolutional layers, upscale it using a transposed convolutional layer, and concatenate it with the next feature map, then we repeat the same procedure with the result until we get a segmentation map with the same dimensions as the input image (Ronneberger et al. [2015]) (See Figure 1.12).

Figure 1.12: Example of a U-net model, taken from Ronneberger et al. [2015]




## 1.4 Deep Learning and iSCAT

While localization of particles with high contrast and distance from each other can be done easily and reliably, resolving details smaller than the diffraction limit or with low signal-to-noise ratios remains a challenge.

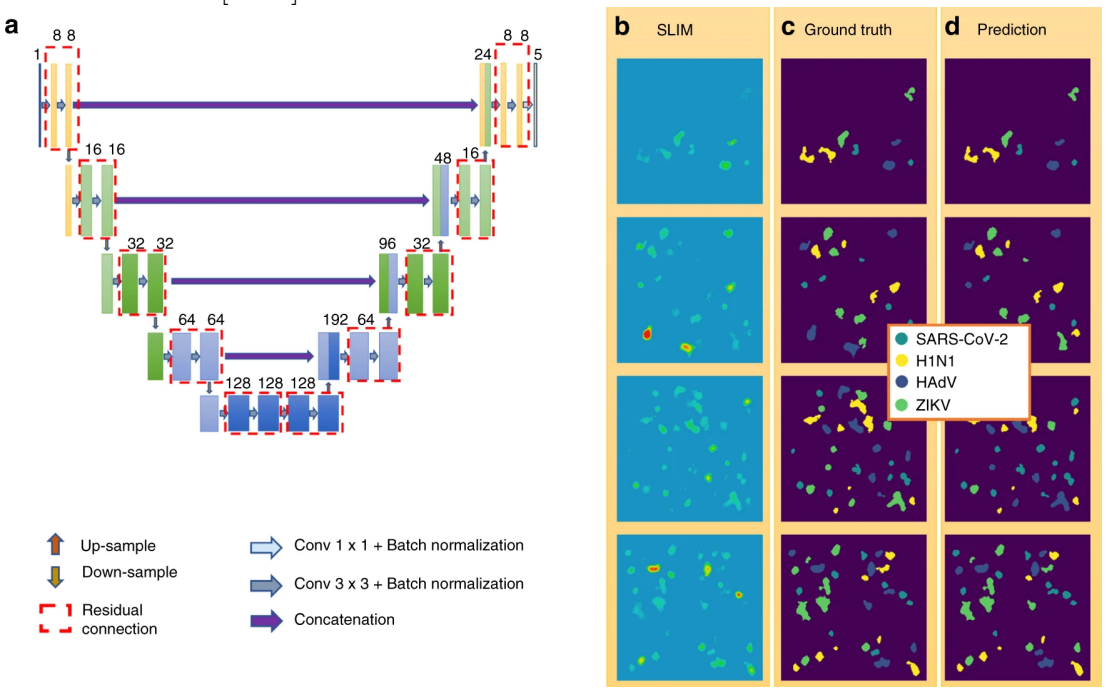
### Label-free virus classification

Details that distinguish viruses are well below the diffraction limit, making it difficult to classify them based on iSCAT images.

In 2021 (Goswami et al. [2021]) a U-Net architecture (See Figure 1.13a) was trained to segment SARS-CoV-2, H1N1 (influenza-A virus), HAdV (adenovirus),

and ZIKV (Zika virus) in the iSCAT image. the neural network was able to identify SARS-CoV-2 vs. the other viruses with 96% accuracy (See Figure 1.13b).

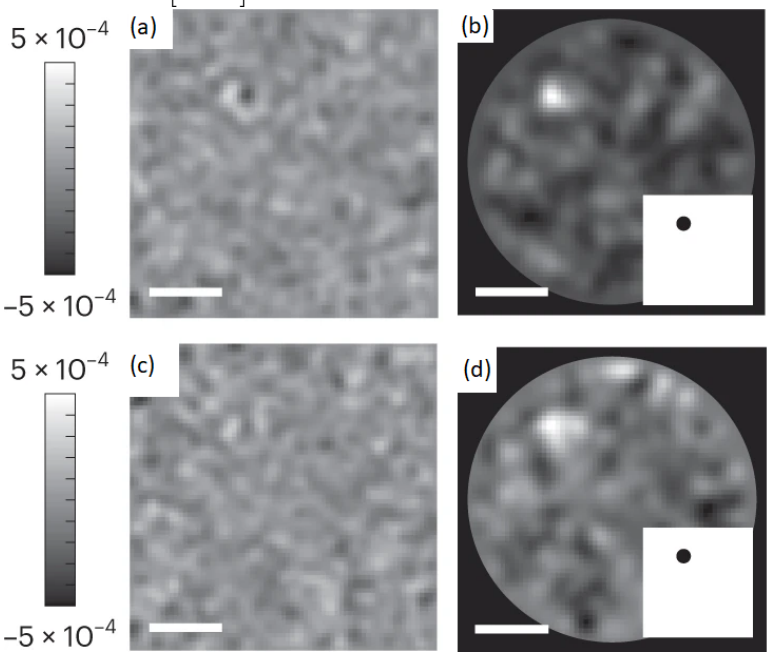
Figure 1.13: (a) A modified version of U-Net. (b) Synthesized images of mixed virus particles. (c) Ground truth label. (d) Model inference. Taken from Goswami et al. [2021]




### Pushing the sensitivity limit in label-free detection of single proteins

It was shown that an unsupervised machine learning isolation forest algorithm for anomaly detection can push the mass sensitivity limit by a factor of 4 (Dahmardeh et al. [2023]) (See Figure 1.14).

Figure 1.14: **a,c** Outcome of differential rolling average (DRA) of 750 (a) and 250 (c) frames. The color bars show the iSCAT contrast, **b,d** Probability maps based on the DNN approach for the DRA window sizes of 750 (b) and 250 (d) frames. Insets show the corresponding binary masks. Scale bars 1.5 $\mu$m. Taken from Dahmardeh et al. [2023]



# 2. Practical part

## 2.1 Data simulation

### 2.1.1 Setup

#### iSCAT Setup

In the practical part, we will model the image signal of the iSCAT setup with the following parameters. The setup uses a laser with a wavelength of 662 nm, the objective has NA equal to 1.4, and one pixel of the image detected in the camera will correspond to 23 nm.

We can see that the diffraction limit for this setup is equal to 236.4 nm or approximately 10.27 px.

#### Hardware Setup

Everything will be executed on either one of the following two machines.

The first machine contains a 12-Core 2.9GHz Intel(R) Core(TM) i9-7920X CPU, 128GB of 2400MHz DDR4 RAM, and NVIDIA GeForce GTX 1080 Ti GPU with 11GB of VRAM.

The second machine contains a 32-Core 3.69 GHz AMD Ryzen Threadripper 3970X CPU, 256GB of 2400MHz DDR4 RAM, and NVIDIA GeForce RTX 3080 GPU with 10GB of VRAM.

#### Software Setup

The OS used is Ubuntu 22.04 with Python 3.9.16, Tensorflow 2.11 with GPU support, cudatoolkit 11.2, cudnn 8.1, cython 0.29, and numpy 1.24.2.

### 2.1.2 Point Spread Function (PSF)

The image of a point source of light is called a *Point Spread Function* (PSF).

As we described in the review part 1.1.3, the image of a particle will consist of periodically changing bright and dark circles around its position.

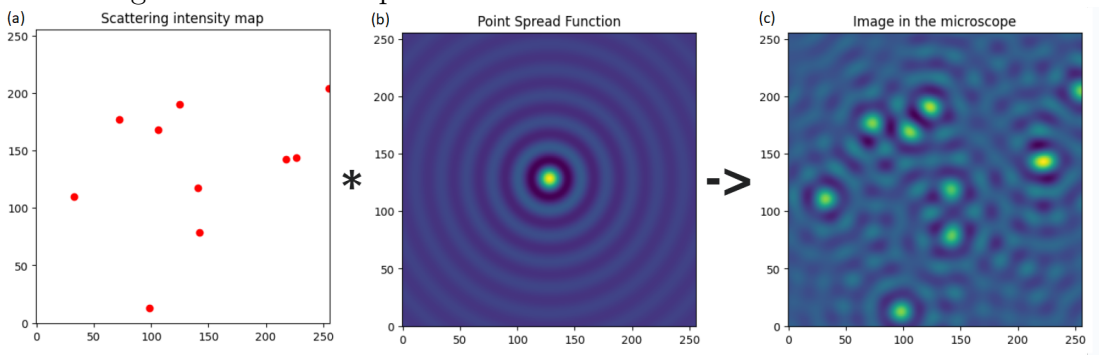
While FDTD simulations and vectorial diffraction models could be used for rigorous PSF simulation (Mahmoodabadi et al. [2020]). PSF usually looks like an Airy disc function. But for simplicity, we have decided to use approximation with similar properties:

$$\frac{\sin r}{r}$$

where $r$ is the distance from the particle's position.

In the next sections, we handle the PSF as a NumPy array and the results should therefore be easily reproducible with more rigorous approximations as well.

Figure 2.1: (a) Particle positions. (b) Our PSF approximation ($\frac{\sin r}{r}$). (c) Simulated image in the microscope




### 2.1.3 Sparse Subpixel Convolution

The image in the microscope is equal to the convolution between the sample's scattering intensity map and the microscope's point spread function.

#### Problem

However, if we would do the discrete convolution in the camera's resolution, the result would not appear very realistic. For example, in the case of nano-particles moving smaller distances than a single pixel, the simulated image would stay the same, although the sample changes.

We have therefore decided to apply convolution in a resolution higher than the camera's resolution by multiplying it with a constant, which we call *subpixel resolution*. But then we face a problem with the time complexity of such approach. If we would naively upscale the scattering intensity map of the sample, then use built-in convolution functions, like those in NumPy, and downscale the result back to the camera's resolution, the time complexity would look approximately like this:

$$O(r_m^2 \cdot r_o^2 \cdot s^4)$$

where $r_m^2$ is the resolution of the scattering intensity map of the sample, $r_o^2$ is the resolution of the camera and $s^4$ is the subpixel resolution.

#### Proposed solution

To get around these limitations, we proposed the following algorithm. We save our simulated PSF in slices with the same pixel size as the camera but with each slice shifted by a subpixel distance. The count of those slices will be $s^2$. Then for each particle in the sample, we pick the correct subpixel-shifted PSF, multiply it by the particle's scattering intensity, and finally, add it to the output image shifted by the particle's pixel position. While achieving the same result as the previous approach, here we get the time complexity equal to:

$$O(r_o^2 \cdot p)$$

where $p$ is the number of particles.

As we can see, now the time complexity is not dependent on the resolution of the simulated sample or subpixel resolution and in the computation, we use particle positions directly without generating their scattering intensity map first. We should therefore expect a performance increase by several orders of magnitude compared with the previous approach.

#### Our implementation

We implemented our proposed algorithm for fast subpixel convolution into a library.

For optimal performance, we have used the programming language C++ and accelerated the computations using OpenMP for multi-threaded as well as SIMD (Single Instruction/Multiple Data) optimizations.

We wrapped the C++ code using Cython to create a Python module, which we named **Sparse Subpixel Convolution**, that can be easily imported into Python scripts.

If we calculate particle positions in parallel as well, then each one of our machines' CPU is capable, using our library, to generate 524,288 frames with 64x64 image resolution and 170 diffusing particles in the sample in less than a minute.

## 2.2 Data Analysis using Deep Neural Networks

### 2.2.1 The Dataset

We simulated image sequences of diffusing particles viewed by the iSCAT setup.

We chose the parameters of our dataset generation in such a way so that the image sequences look physically plausible and contain speckle patterns complex enough that their analysis using techniques such as PSF fitting would not be possible most of the time.

#### The dataset generation

Since modern libraries specializing in machine learning use mainly GPU (or other highly parallel computation units) for model training, the CPU is left available during this computation.

Therefore, we can use our efficient implementation of the data simulation and generate samples on the fly in parallel with the training, without any performance penalty.

This way we will be able to have a different randomly generated dataset of iSCAT samples in each epoch and the model will use each sample only once during the whole training, eliminating the risk of overfitting.

Since overfitting will not be a problem, we don't need to consider regularization while designing our models at all, and we can just increase their capacity until they reasonably fit in our VRAM.

### The dataset parameters

The dataset generated for each epoch contains 16,384 image sequences with frame counts of 32, and camera FOV resolution 64x64.

We chose the subpixel resolution of convolutions as 64x64, for each possible subpixel shift we generated PSF with a resolution of 512x512. The simulated samples then have a size of 448x448 camera-sized pixels, so that the contribution of particles on sample edges is accounted for in the generated PSF.

The time difference between subsequent frames is 0.1 s. The particle count in the sample is random with a discrete uniform distribution $\mathcal{U}\{30, 310\}$, each particle's diffusion coefficient is $\sim \mathcal{U}\{1500, 3500\}\text{nm}^2\text{s}^{-1} \approx \mathcal{U}\{2.8, 6.6\}\text{px}^2\text{s}^{-1}$, and its scattering intensity $\sim \mathcal{U}\{0.6, 1.4\}$.

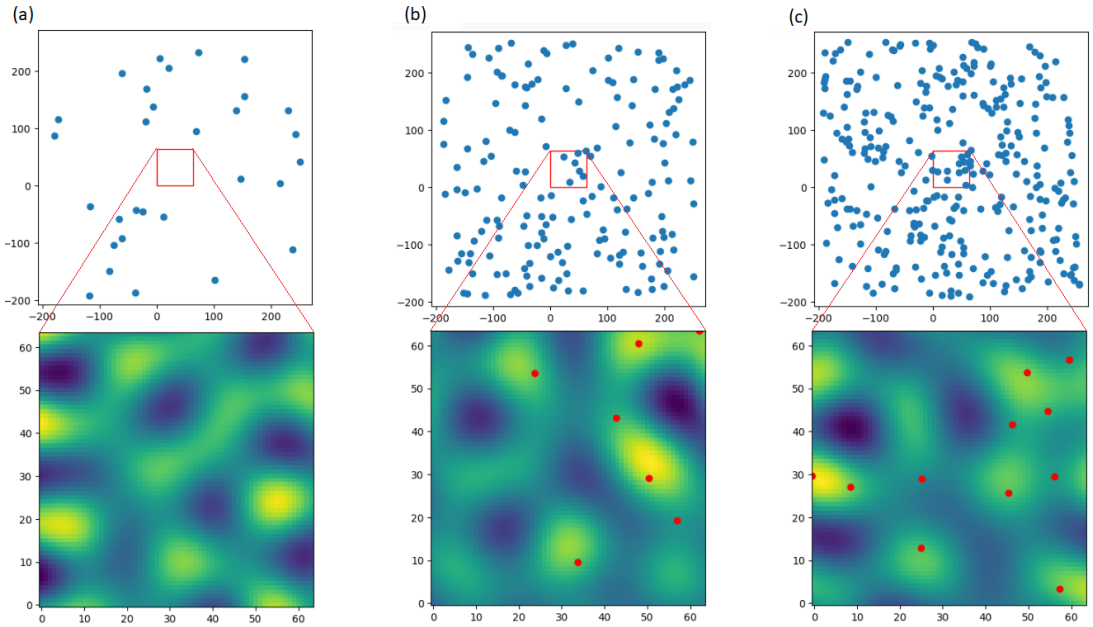
Examples of samples in our dataset are shown in Figure 2.2

In the next sections, we design and train models that count or localize particles in the image sequences.

In the classification task, we will be predicting the count of particles that are in the FOV in the 15th frame of the sequences.

In the localization task, we will be segmenting the 15th frame of the sequences into pixels with and without particles.

Figure 2.2: Figure shows three pairs of images corresponding to a single frame of some samples from our dataset. The top image shows the simulated sample, where blue dots correspond to particle positions and the red square shows the camera's FOV. The bottom image is the simulated image in the camera, corresponding to the sample, with red dots highlighting the particle positions. The samples have particle counts equal to (a) 30 (lower bound in our dataset), (b) 170 (average), and (c) 309 (upper bound).



### 2.2.2 Classification

We will be classifying each image sequence into 8 classes. Classes 0, 1, 2, ..., 6 correspond to particle counts of 0, 1, 2, ..., 6 in the camera's FOV (in the 15th frame of the sequence), and class 7 corresponds to a particle count of 7 or higher.

All of our models will have a training dataset of 16,384 image sequences updated during each epoch, and a validation dataset of 16,384 sequences generated only once in the beginning.

As our loss function, we always selected the Sparse Categorical Crossentropy.

We will be evaluating each model by its accuracy, mean deviation, and confusion matrix.

Accuracy is the ratio of correctly predicted classes to the number of elements in the dataset. Mean deviation is calculated by subtracting predicted class indices from true class indices and averaging the absolute value of the result. The confusion matrix shows us how often the model predicts different classes for each true class.

#### MLP

The first model we explored was the most simple one in the deep learning family - The Multi-Layer Perceptron.

Our MLP contains only a single hidden layer. The input layer consists of 131,072 neurons (flattened input image sequences) with *swish* activation function, the hidden layer contains 2000 neurons with *swish* activation, and the output layer of 8 neurons with *softmax* activation.

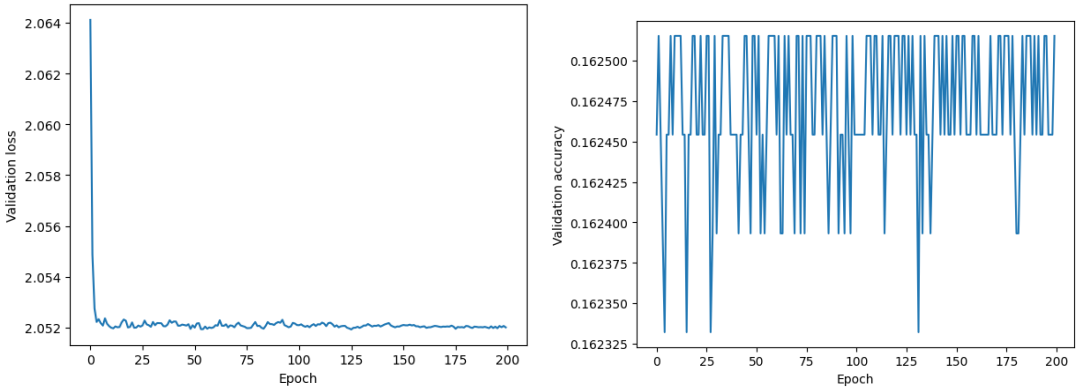
The model has 262,162,008 parameters in total. Tensorflow summary of the model:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 131072)            0

 dense (Dense)               (None, 2000)              262146000

 dense_1 (Dense)             (None, 8)                 16008

=================================================================
Total params: 262,162,008
Trainable params: 262,162,008
Non-trainable params: 0
_________________________________________________________________
```

**MLP - Training** We trained the MLP in 200 epochs, with batch size 12. The total number of image sequences used during the training is 3,276,800. We used cosine learning rate decay, starting at value $5 \cdot 10^{-4}$ and ending at 0 and Adam optimizer with otherwise default TensorFlow parameters.

The progress of the model during the training is shown in Figure 2.3.

Figure 2.3: The progress of our MLP through training epochs with the validation loss (left) and the validation accuracy (right).



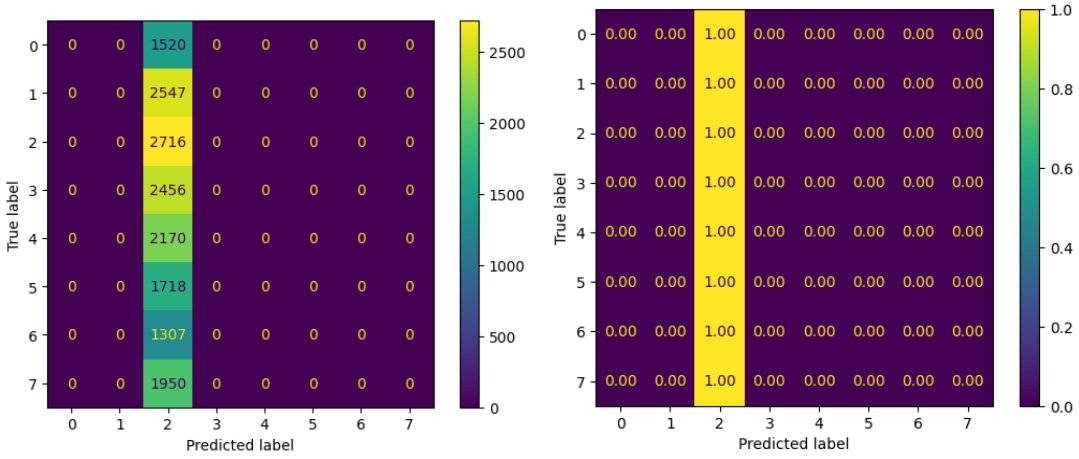

**MLP - Evaluation** The accuracy achieved by our MLP is 16.23% on the validation dataset with a mean deviation of 1.9970.

We would expect a random generator to have an accuracy of 12.5% and a mean deviation of 2.625.

We see that the MLP is just slightly better than a random generator at predicting the particle counts.

Upon closer inspection, we noticed that the MLP converged to predicting a particle count of 2 all the time, which was indeed the most frequent class in our dataset, see confusion matrices in Figure 2.4.

Figure 2.4: The confusion matrix of our MLP with absolute values (left) and ratios with respect to the sample count of each class (right).




As we can see, the MLP is not going to help us with this task.

### Two-dimensional ResNet

A much better approach is modifying the ResNet architecture, which was designed to process images and dimensional data.

However, we are not classifying isolated images, instead, we take whole image sequences into account. We decided, for the two-dimensional ResNet, that we will handle the temporal dimension as the channel dimension (instead of colors).

The model consists of the input layer, three stages of residual networks, and the output layer.

Our residual blocks consist of two 2D convolutions with kernel size 3x3, two batch normalizations, and two *swish* activations.

We decided that in the first step of the ResNet, we will keep the channel size equal to 32, instead of the usual 16 channel size, to prevent loss of information.

In each stage of the model, we process the tensor using 20 residual blocks.

In the next two stages, we halve the image dimensions (using a stride) and double the channel dimensions. So we process 64x32x32 Tensor in the second stage and 128x16x16 Tensor in the third stage.

Then we use a global average pooling layer, where we average 2D Tensor slices across the channel dimensions, resulting in 128 neurons.

Finally, we follow with the dense layer with the output of 8 neurons with *softmax* activation function.

The model has 3,808,456 parameters in total. Cropped Tensorflow summary of the model:

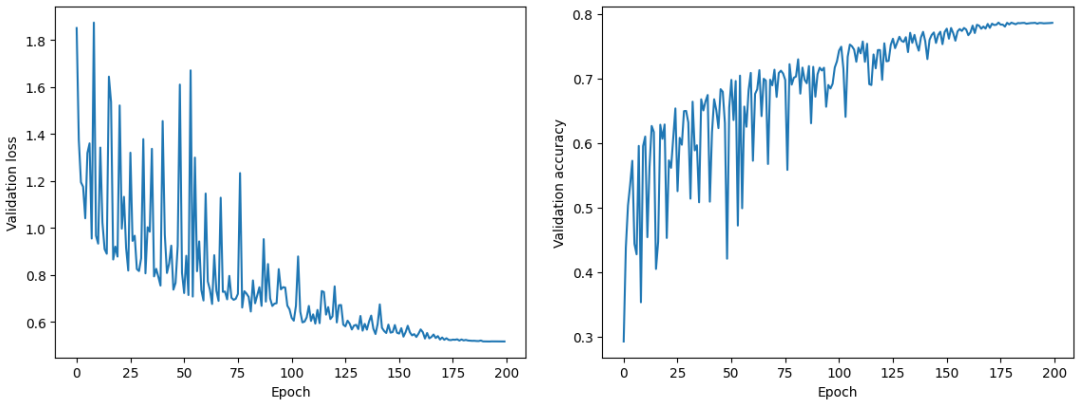
```
Model: "res_net"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1                      (None, 32, 64, 64)        0
 conv2d                       (None, 32, 64, 64)        9216
 batch_normalization          (None, 32, 64, 64)        256
 activation                   (None, 32, 64, 64)        0

 conv2d_1                     (None, 32, 64, 64)        9216
 batch_normalization_1        (None, 32, 64, 64)        256
 activation_1                 (None, 32, 64, 64)        0
 conv2d_2                     (None, 32, 64, 64)        9216
 batch_normalization_2        (None, 32, 64, 64)        256
 add                          (None, 32, 64, 64)        0
 activation_2                 (None, 32, 64, 64)        0

 ...
 repeat the residual block above 19 times
 ...

 conv2d_21                    (None, 64, 32, 32)        18432
 batch_normalization_21       (None, 64, 32, 32)        128
 activation_21                (None, 64, 32, 32)        0
 conv2d_23                    (None, 64, 32, 32)        2048

 ...
 repeat the residual block 20 times
 ...

 conv2d_42                    (None, 128, 16, 16)       73728
```

```
batch_normalization_42          (None, 128, 16, 16)  64
activation_41                   (None, 128, 16, 16)  0
conv2d_44                       (None, 128, 16, 16)  8192

...
repeat the residual block 20 times
...

global_average_pooling2d        (None, 128)          0
dense                           (None, 8)            1032

=================================================================
Total params: 3,808,456
Trainable params: 3,803,752
Non-trainable params: 4,704
_________________________________________________________________
```

**Two-dimensional ResNet - Training** We trained the 2D ResNet in 200 epochs, with batch size 32. The total number of image sequences used during the training is 3,276,800. We used cosine learning rate decay, starting at value $5 \cdot 10^{-5}$ and ending at 0 and Adam optimizer with otherwise default TensorFlow parameters.

The progress of the model during the training is shown in Figure 2.5.

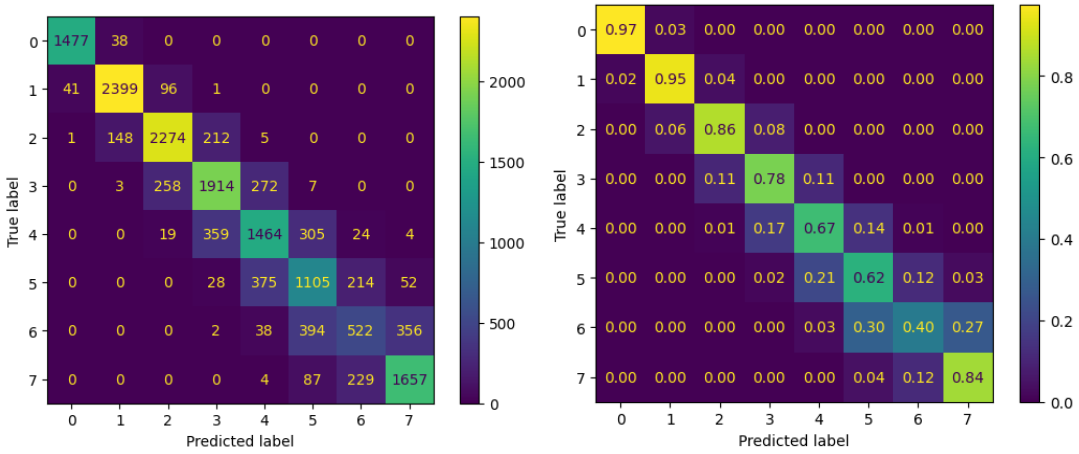
Figure 2.5: The progress of our 2D ResNet through training epochs with the validation loss (left) and the validation accuracy (right).




**Two-dimensional ResNet - Evaluation** The accuracy achieved by our 2D ResNet is 78.2% with a mean deviation of 0.2354.

The confusion matrix of the model on validation dataset is shown in Figure 2.6.

Figure 2.6: The confusion matrix of our ResNet with absolute values (left) and ratios with respect to the sample count of each class (right).




We see that our 2D ResNet model performs very well.

### Three-dimensional ResNet

The previous model contained 2-dimensional convolution layers, where we interpreted the temporal dimension of our data as a channel dimension. However, since we have 3-dimensional data, with 2 spatial and 1 temporal resolution, it might make more sense to use 3-dimensional convolution layers instead.

Now, we will take the architecture of our previous model and replace all 2-dimensional convolutions with 3-dimensional. We will interpret the input as a 4-dimensional tensor with a channel size equal to 1.

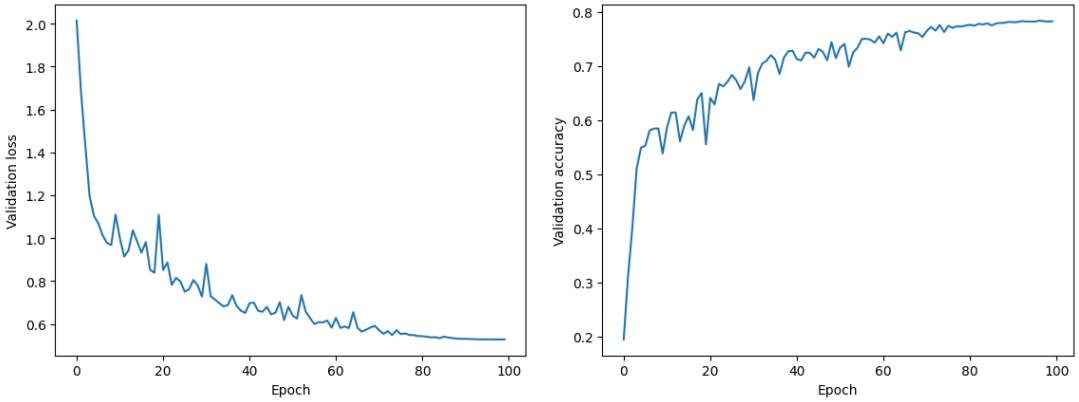
At the beginning of the first stage, we upscale the input tensor into a tensor with a channel size equal to 8. Then we proceed the same way as in the 2D ResNet, working with 8x32x64x64 tensors in the first stage, 16x16x32x32 tensors in the second stage, and 32x8x16x16 tensors in the third stage.

This model has 719,008 parameters in total. More than 5 times fewer parameters than the 2D ResNet, because of the much smaller channel size 8 instead of 32.

**Three-dimensional ResNet - Training** We trained the 3D ResNet in 100 epochs, with batch size 12. The total number of image sequences used during the training is 1,638,400. We used cosine learning rate decay, starting at value $5 \cdot 10^{-5}$ and ending at 0 and Adam optimizer with otherwise default TensorFlow parameters.

The progress of the model during the training is shown in Figure 2.7.

Figure 2.7: The progress of our 3D ResNet through training epochs with the validation loss (left) and the validation accuracy (right).



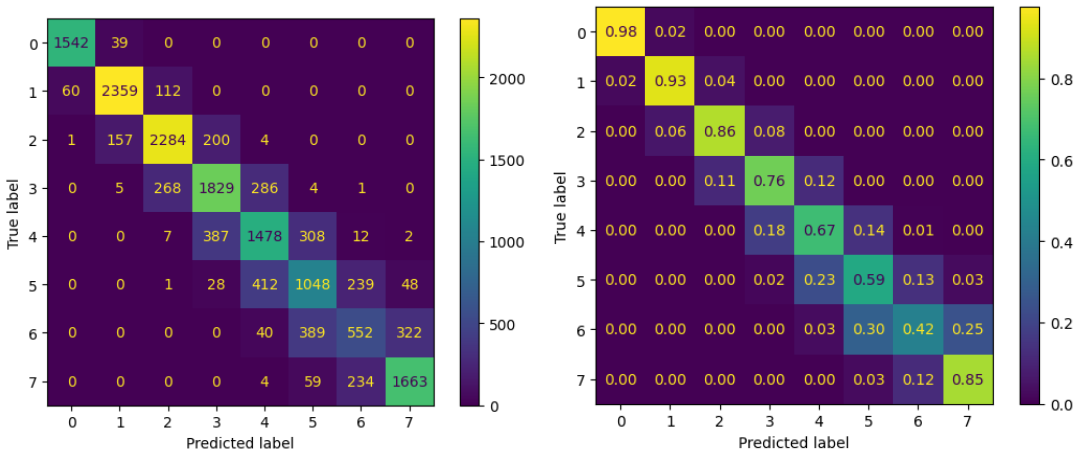

**Three-dimensional ResNet - Evaluation** The accuracy achieved by our 3D ResNet is 77.84% with a mean deviation of 0.2352

As we can see, the performance of our 3D ResNet is almost the same as the 2D ResNet. However, it is necessary to point out, that this is achieved with more than 5 times fewer parameters.

The reason for the small parameter count in this model is the increased time complexity of 3-dimensional convolutions. Each epoch took around 15 minutes to run, the total time of the whole training being around 25 hours.

The confusion matrix of the model on the validation dataset is shown in Figure 2.8.

Figure 2.8: The confusion matrix of our 3D ResNet with absolute values (left) and ratios with respect to the sample count of each class (right).




## Two-plus-one-dimensional ResNet

As we saw, the time complexity of 3-dimensional convolutions is very high. If we could lower the time complexity of the 3D ResNet, then we could increase its capacity and potentially surpass its performance.

To work around this problem we can try to split 3-dimensional convolutions into 2-dimensional convolutions in the spatial dimensions and 1-dimensional convolutions in the temporal dimension.

Now, when the execution and training of our model are more efficient, we can handle higher channel sizes.

We increased the channel size in the first stage three times to 24. So the first stage deals with 24x32x64x64 tensors, the second stage with 48x16x32x32 tensors, and the third stage with 96x8x16x16 tensors.

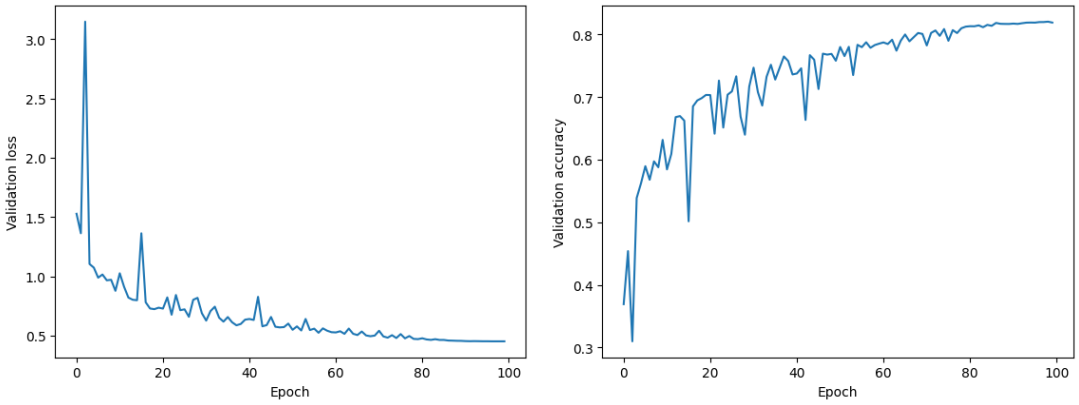
This model has 2,867,168 parameters in total, around four times more than the 3D ResNet.

**Two-plus-one-dimensional ResNet - Training** While the channel size increased three times, the training time increased by only one-half to 23 minutes per epoch or around 38 hours for the whole training.

We trained the (2+1)D ResNet using 100 epochs, with batch size 4. The total number of image sequences used during the training is 1,638,400. We used cosine learning rate decay, starting at value $5 \cdot 10^{-5}$ and ending at 0 and Adam optimizer with otherwise default TensorFlow parameters.

The progress of the model during the training is shown in Figure 2.9.

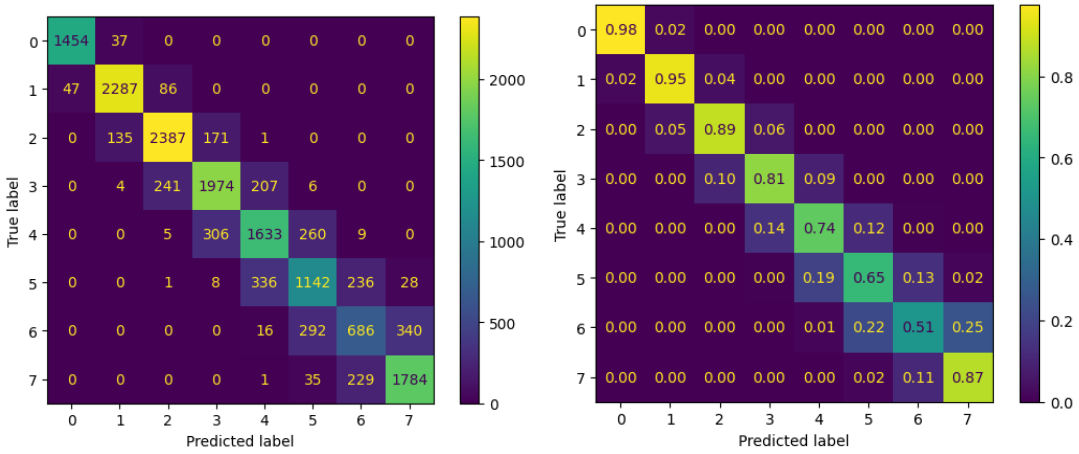
Figure 2.9: The progress of our (2+1)D ResNet through training epochs with the validation loss (left) and the validation accuracy (right).




**Two-plus-one-dimensional ResNet - Evaluation** The accuracy achieved by our (2+1)D ResNet is 81.47% with a mean deviation of 0.1924.

The confusion matrix of the model on the validation dataset is shown in Figure 2.10.

Figure 2.10: The confusion matrix of our (2+1)D ResNet with absolute values (left) and ratios with respect to the sample count of each class (right).




As we can see, this model with higher capacity, compared to 3D ResNet, achieved higher accuracy as we expected.

### 2.2.3 Localization

We would like to localize particles in the FOV in the 15th frame of an image sequence.

This problem can be formulated as a task of classifying image pixels into whether they contain or do not contain particles.

All of our models will have a training dataset of 16,384 image sequences updated during each epoch, and a validation dataset of 16,384 sequences generated only once in the beginning. As our loss function, we will select the Binary Crossentropy.

#### U-net

In this section, we will be modifying the U-net architecture. Here, we will use our 2D ResNet from the classification task as a backbone.

We extract feature maps of the last layers of each stage from our 2D ResNet. This way, we get 128x16x16, 64x32x32, and 32x64x64 tensors (feature maps). We use convolution and a transposed convolution layer on the 128x16x16 feature map in such a way as to change its dimensions to be the same as the next feature map, then we concatenate the result with the next feature map. We further process the result with 30 residual blocks. Then we use the same procedure for the next feature map as well.

Finally, we use a convolutional layer to get the output dimensions of 1x64x64, we also use a sigmoid activation after.

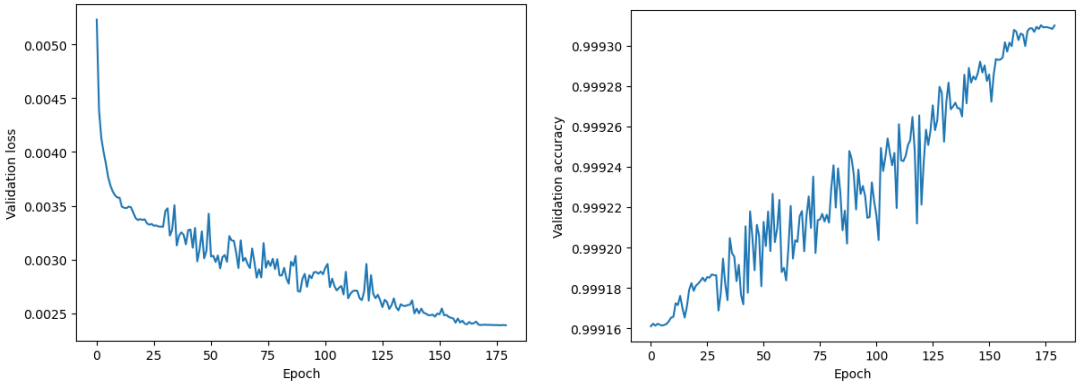
This model has 6,734,753 parameters in total.

**U-net - Training** We trained in two stages. The first stage took 30 epochs and the parameters of the backbone were frozen, so only the new parameters were being trained. The second stage took 150 epochs and the whole model was being trained. Both stages used cosine learning decay with starting learning rate of

$5 \cdot 10^{-5}$ and ending at 0 and Adam optimizer with otherwise default TensorFlow parameters. The total number of image sequences used during the training was 2,949,120.

The progress of the model during the training is shown in Figure 2.11.

Figure 2.11: The progress of our U-net through training epochs with the validation loss (left) and the validation accuracy (right).



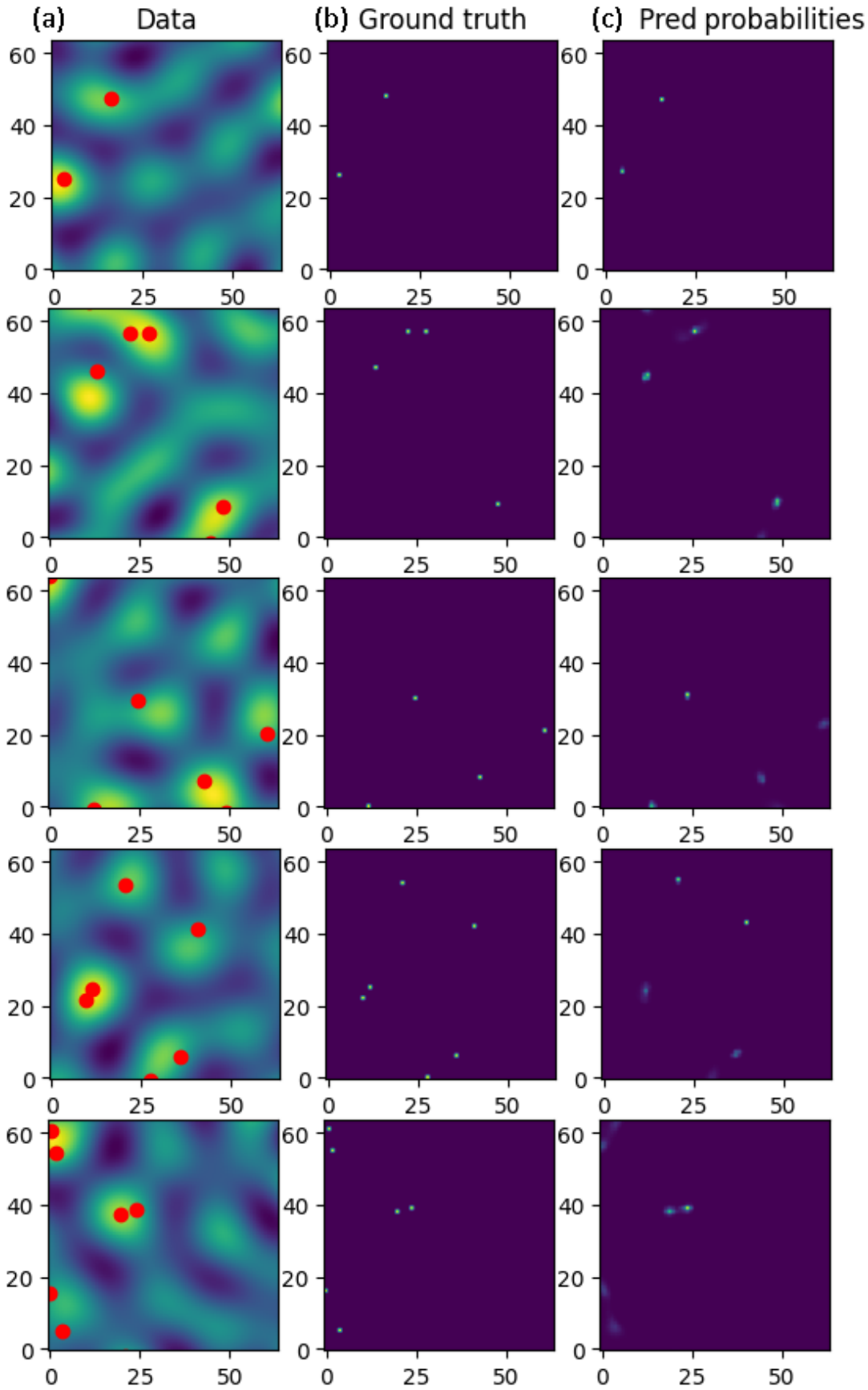

**U-net - Evaluation** Our U-net achieves an accuracy of 99.931%, which corresponds to 2.826 pixels incorrectly classified per sample on average.

Since the average particle count in the FOV is around 3.3, we can see that the model must localize particles correctly at least sometimes to achieve this accuracy.

However, when we plot the predictions and compare them with the correct locations (Figure 2.12), we see that our model's performance exceeds our expectations.

We think that the reason for this amazing performance might be a possible efficient use of the temporal information contained in image sequences.

Figure 2.12: (a) 15th frame of the analyzed image sequence, with red dots highlighting particle positions. (b) Correct particle positions in each frame. (c) Predicted probabilities of particle locations.



## 2.3 Code Availability

The code of our Python module **Sparse Subpixel Convolution**, for data simulation, can be found at `https://github.com/luzny274/Sparse_Subpixel_Convolution`. The module is currently compiled for NumPy 1.24.2, but it can be easily recompiled for other versions as well, using scripts *pyrebuild.bat* for Windows systems and *pyrebuild.sh* for Linux systems.

The code of the rest of our scripts, from PSF approximation to neural network training, can be found at `https://github.com/luzny274/DeepLearning_iSCAT`. Weights of the MLP are not included, because of their enormous size exceeding the Github file size limit.

# Conclusion

In conclusion, this thesis has demonstrated the potential of using deep neural networks (DNNs) for the analysis of speckle patterns in iSCAT microscopy.

In the review part, we briefly described the fundamentals of iSCAT microscopy and Deep Learning.

In the practical part, we developed a Python module in C++, that enables us to efficiently simulate diffusing particles and their images in the iSCAT microscope. Making it possible to generate large datasets in a short amount of time.

Then we successfully modified the ResNet architecture to classify simulated iSCAT image sequences with accuracy up to 81.47% and modified the U-net architecture to localize particles in the camera's field of view. We attribute the success of DNNs in these tasks to their ability to utilize the temporal information and prior knowledge of the studied sample.

Although the demonstrated performance on simulated data may not be directly applicable to real-life experiments yet, it showcases the capability of DNNs to investigate samples that were previously inaccessible for analysis.

# Bibliography


L. A. Amos and W. B. Amos. The bending of sliding microtubules imaged by confocal light microscopy and negative stain electron microscopy. *Journal of Cell Science. Supplement*, 14:95–101, 1991. ISSN 0269-3518. doi: 10.1242/jcs.1991.supplement_14.20.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, New York, NY, softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009) edition, 2016. ISBN 9781493938438.

Robert Brown. XXVII. *A brief account of microscopical observations made in the months of June, July and August* 1827, *on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. The Philosophical Magazine*, 4(21):161–173, September 1828. ISSN 1941-5850, 1941-5869. doi: 10.1080/14786442808674769. URL https://www.tandfonline.com/doi/full/10.1080/14786442808674769.

Mahyar Dahmardeh, Houman Mirzaalian Dastjerdi, Hisham Mazal, Harald Köstler, and Vahid Sandoghdar. Self-supervised machine learning pushes the sensitivity limit in label-free detection of single proteins below 10 kDa. *Nature Methods*, 20(3):442–447, March 2023. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-023-01778-2. URL https://www.nature.com/articles/s41592-023-01778-2.

Alexander P Demchenko. Photobleaching of organic fluorophores: Quantitative characterization, mechanisms, protection. *Methods and Applications in Fluorescence*, 8(2):022001, 2020. doi: 10.1088/2050-6120/ab7365.

Albert Einstein and R. Fürth. *Investigations on the theory of Brownian movement*. Bnpublishing.net, Place of publication not identified, 2011. ISBN 9781607962854. OCLC: 922648379.

Rolf Erni, Marta D. Rossell, Christian Kisielowski, and Ulrich Dahmen. Atomic-Resolution Imaging with a Sub-50-pm Electron Probe. *Physical Review Letters*, 102(9):096101, March 2009. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.102.096101. URL https://link.aps.org/doi/10.1103/PhysRevLett.102.096101.

Eric D. B. Foley, Manish S. Kushwah, Gavin Young, and Philipp Kukura. Mass photometry enables label-free tracking and mass measurement of single proteins on lipid bilayers. *Nature Methods*, 18(10):1247–1252, October 2021. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-021-01261-w. URL https://www.nature.com/articles/s41592-021-01261-w.

Franz J. Giessibl. Atomic Resolution of the Silicon (111)-(7×7) Surface by Atomic Force Microscopy. *Science*, 267(5194):68–71, January 1995. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.267.5194.68. URL https://www.science.org/doi/10.1126/science.267.5194.68.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Neha Goswami, Yuchen R. He, Yu-Heng Deng, Chamteut Oh, Nahil Sobh, Enrique Valera, Rashid Bashir, Nahed Ismail, Hyunjoon Kong, Thanh H. Nguyen, Catherine Best-Popescu, and Gabriel Popescu. Label-free SARS-CoV-2 detection and classification using phase imaging with computational specificity. *Light: Science & Applications*, 10(1):176, September 2021. ISSN 2047-7538. doi: 10.1038/s41377-021-00620-8. URL https://www.nature.com/articles/s41377-021-00620-8.

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

Tamara Heermann, Frederik Steiert, Beatrice Ramm, Nikolas Hundt, and Petra Schwille. Mass-sensitive particle tracking to elucidate the membrane-associated MinDE reaction cycle. *Nature Methods*, 18(10):1239–1246, October 2021. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-021-01260-x. URL https://www.nature.com/articles/s41592-021-01260-x.

Jekaterina Kazantseva, Roman Ivanov, Michael Gasik, Toomas Neuman, and Irina Hussainova. Graphene-augmented nanofiber scaffolds demonstrate new features in cells behaviour. *Scientific Reports*, 6(1):30150, July 2016. ISSN 2045-2322. doi: 10.1038/srep30150. URL https://www.nature.com/articles/srep30150.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs].

Frank B. Knight. On the random walk and Brownian motion. *Transactions of the American Mathematical Society*, 103(2):218–228, 1962. ISSN 0002-9947, 1088-6850. doi: 10.1090/S0002-9947-1962-0139211-2. URL https://www.ams.org/tran/1962-103-02/S0002-9947-1962-0139211-2/.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets, 2018.

K. Lindfors, T. Kalkbrenner, P. Stoller, and V. Sandoghdar. Detection and spectroscopy of gold nanoparticles using supercontinuum white light confocal microscopy. *Physical Review Letters*, 93(3):037401, July 2004. ISSN 0031-9007. doi: 10.1103/PhysRevLett.93.037401.

Ariel Lipson, Stephen Geoffrey Lipson, and Henry Lipson. *Optical physics*. Cambridge, 1981.

Reza Gholami Mahmoodabadi, Richard W. Taylor, Martin Kaller, Susann Spindler, Mahdi Mazaheri, Kiarash Kasaian, and Vahid Sandoghdar. Point spread function in interferometric scattering microscopy (iscat). part i: aberrations in defocusing and axial localization. *Opt. Express*, 28(18):25969–25988, Aug 2020. doi: 10.1364/OE.401374. URL https://opg.optica.org/oe/abstract.cfm?URI=oe-28-18-25969.

W. E. Moerner and L. Kador. Optical detection and spectroscopy of single molecules in a solid. *Physical Review Letters*, 62(21):2535–2538, May 1989. ISSN 0031-9007. doi: 10.1103/PhysRevLett.62.2535. URL https://link.aps.org/doi/10.1103/PhysRevLett.62.2535.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Marek Piliarik and Vahid Sandoghdar. Direct optical sensing of single unlabelled proteins and super-resolution imaging of their binding sites. *Nature Communications*, 5(1):4495, July 2014. ISSN 2041-1723. doi: 10.1038/ncomms5495. URL https://www.nature.com/articles/ncomms5495.

Malte Renz. Fluorescence microscopy-A historical and technical perspective: Fluorescence Microscopy. *Cytometry Part A*, 83(9):767–779, September 2013. ISSN 15524922. doi: 10.1002/cyto.a.22295. URL https://onlinelibrary.wiley.com/doi/10.1002/cyto.a.22295.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351, pages 234–241. Springer International Publishing, Cham, 2015. ISBN 9783319245737 9783319245744. doi: 10.1007/978-3-319-24574-4_28. URL http://link.springer.com/10.1007/978-3-319-24574-4_28.

Nicole Rusk. The fluorescence microscope: First fluorescence microscope, First epifluorescence microscope, The dichroic mirror. *Nature Cell Biology*, 11(S1): S8–S9, October 2009. ISSN 1465-7392, 1476-4679. doi: 10.1038/ncb1941. URL http://www.nature.com/articles/milelight04.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Pearson, Boston Columbus Indianapolis New York San Francisco Upper Saddle River Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo, third edition, global edition edition, 2016. ISBN 9781292153964.

John H. Seinfeld and Spyros N. Pandis. *Atmospheric chemistry and physics: from air pollution to climate change*. J. Wiley, Hoboken, N.J, 2nd ed edition, 2006. ISBN 9780471720171 9780471720188. OCLC: ocm62493628.

Russell E. Thompson, Daniel R. Larson, and Watt W. Webb. Precise Nanometer Localization Analysis for Individual Fluorescent Probes. *Biophysical Journal*, 82(5):2775–2783, May 2002. ISSN 00063495. doi: 10.1016/S0006-3495(02)75618-X. URL https://linkinghub.elsevier.com/retrieve/pii/S000634950275618X.

vdumoulin. conv_arithmetic, 2019. URL https://github.com/vdumoulin/conv_arithmetic.

Donna J. Webb and Claire M. Brown. Epi-Fluorescence Microscopy. In Douglas J. Taatjes and Jürgen Roth, editors, *Cell Imaging Techniques*, volume 931, pages 29–59. Humana Press, Totowa, NJ, 2012. ISBN 9781627030557 9781627030564. doi: 10.1007/978-1-62703-056-4_2. URL http://link.springer.com/10.1007/978-1-62703-056-4_2.

Alexander Weigel, Aleksandar Sebesta, and Philipp Kukura. Dark Field Microspectroscopy with Single Molecule Fluorescence Sensitivity. *ACS Photonics*, 1(9):848–856, September 2014. ISSN 2330-4022, 2330-4022. doi: 10.1021/ph500138u. URL https://pubs.acs.org/doi/10.1021/ph500138u.

# List of Figures