Real-time image-based cell identification

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium
(Dr. rer. nat.)

vorgelegt
der Fakultät Mathematik und Naturwissenschaften
der Technischen Universität Dresden

von

Maik Herbig

geboren am 29.08.1988 in Dresden, Deutschland

Vorgeschlagene Gutachter

Erstgutachter: Prof. Dr. Jochen Guck

Zweitgutachter: Prof. Dr. Michael Schlierf

Abstract

Identification of different cell types is an indispensable task of biomedical research and clinical application. During the last decades, much attention was given to molecular characterization, and many cell types can now be identified using established markers that bind to cell-specific antigens. The required staining process is a lengthy and costly treatment, which can cause alterations of cellular properties, contaminate the sample and therefore limit its subsequent use. For example, for photoreceptor transplantations, highly pure samples of photoreceptor cells are required, which can currently only be obtained using molecular labelling, rendering the resulting sample incompatible for clinical application. A promising alternative to molecular markers is the label-free identification of cells using mechanical or morphological features. Real-time deformability cytometry (RT-DC) is a microfluidic technique, which allows capturing both types of information simultaneously for single cells at high-throughput. In this thesis, I present machine learning methods which allow identifying different cell types, based on bright-field images from RT-DC. In particular, I introduce algorithms that are fast enough to be applied in real-time during the measurement (at >1000 cells/s), which can be used for image-based cell sorting. The performance of the algorithms is shown for the identification of rod precursor cells in retina-samples, indicating that image-based sorting based on those algorithms would allow enriching photoreceptors to a final concentration, applicable for transplantation purposes.

Kurzfassung

Die Identifizierung verschiedener Zellarten ist ein unabdingbarer Bestandteil in der biomedizinischen Forschung und klinischen Anwendung. Während der vergangenen Jahrzehnte bestand ein verstärktes Interesse an der molekularen Charakterisierung und viele Zellarten können nun mit bewährten Markern identifiziert werden. Der notwendige Färbeprozess ist eine langwierige und kostspielige Behandlung, welche Änderungen zellulärer Eigenschaften hervorrufen, die Probe kontaminieren und daher den Nutzen der Probe einschränken kann. Zum Beispiel werden für Photorezeptortransplantationen reine Proben von Photorezeptoren benötigt, welche gegenwärtig nur mithilfe molekularer Labels hergestellt werden können, was allerdings die Probe für den klinischen Anwendungsbereich unbrauchbar macht. Eine vielversprechende Alternative zu molekularen Markern ist die label-freie Identifizierung von Zellen durch mechanische oder morphologische Eigenschaften. Real-time deformability cytometry (RT-DC) ist eine Technik, basierend auf Mikrofluidik, welche beiderlei Information gleichzeitig auf Einzellzellbasis mit hohem Durchsatz messen kann. In dieser Arbeit führe ich Machine Learning Methoden ein, welche es erlauben, basierend auf Hellfeldbildern von RT-DC, unterschiedliche Zelltypen zu unterscheiden. Insbesondere werde ich Algorithmen aufzeigen, welche während des Experiments in Echtzeit (>1000 Zellen/s) und somit auch für bildbasierte Zellsortierung verwendet werden könnten. Die Genauigkeit dieser Algorithmen wird für die Erkennung von Vorläuferzellen von Stäbchen-Photorezeptoren in Retina-Proben gezeigt. Die bildbasierte Sortierung mit diesen Algorithmen zeigt die konkrete Möglichkeit auf, Proben von Photorezeptoren herzustellen, deren Reinheit für Transplantationszwecke geeignet ist.

Contents

Abstract iii

Kurzfassung iv

List of figures viii

List of tables x

1. Introduction 1
    - 1.1. Texture and mechanical properties: label-free markers 4
    - 1.2. The retina, diseases and cure by photoreceptor transplantation 5
    - 1.3. Technologies for label-free assessment of cells 8
2. Materials and Methods 10
    - 2.1. Experimental setup 10
        - 2.1.1. Chip design for RT-DC and RT-FDC 10
        - 2.1.2. Chip design for soRT-FDC 11
        - 2.1.3. Chip fabrication 13
        - 2.1.4. RT-DC, RT-FDC and soRT-FDC Setup 14
        - 2.1.5. Physics of surface acoustic wave mediated sorting 16
        - 2.1.6. Measurement buffer (MB) for RT-DC 17
    - 2.2. Online parameters 18
    - 2.3. Offline parameters 21
    - 2.4. Linear mixed models (LMM) 28
    - 2.5. Normality test using probability plots 31
    - 2.6. Gaussian mixture model (GMM) and Bayesian information criterion (BIC) 32
    - 2.7. Random forests 33
    - 2.8. Confusion matrix 34
    - 2.9. Deep learning 36
    - 2.10. Preparation of retina samples 43

2.11. Preparation of blood samples 44

2.12. Staining of neutrophils and monocytes 45

3. Results 46

3.1. Meta-analysis of RT-DC data 46

3.1.1. Correlations of area and volume 47

3.1.2. Correlations of deformation and inertia ratio 48

3.1.3. Further screening of correlations 50

3.1.4. Shape of distributions 52

3.1.5. Discussion 54

3.2. Characterization of retina cells in RT-DC 57

3.2.1. Maturation of retina cells 57

3.2.2. Comparing retina cell types using statistical tests 61

3.2.3. Discussion 63

3.3. Classification of retina cells using supervised machine learning 66

3.3.1. The dataset 66

3.3.2. Cell classification using optimized area gating 72

3.3.3. Cell classification using random forests 74

3.3.4. Cell classification using deep neural nets 80

3.3.5. Improving DNN accuracy using image augmentation 85

3.3.6. Tuning of final models and classification performance 93

3.3.7. Visualization of model attention 98

3.3.8. Discussion 100

3.4. Software tools to train and apply deep neural nets for sorting 105

3.4.1. AIDeveloper 105

3.4.2. Sorting Software 113

3.4.3. Discussion 114

3.5. Sorting experiments 117

3.5.1. Sorting of rod precursor cells 117

3.5.2. Sorting of neutrophils 120

3.5.3. Discussion 125

4. Conclusion and outlook 128

A. Appendix 131

I. Comparison of dense and convolutional layer 131

Bibliography 133

Acronyms 148

Acknowledgements 150

Erklärung 152

List of figures

Figure 1.1 Bright-field images of white blood cells captured using RT-DC 4

Figure 1.2 Eye, retina and photoreceptor transplantation 6

Figure 2.1 RT-(F)DC chip design and soRT-FDC setup components.................................. 11

Figure 2.2 Design of the sorting chip 13

Figure 2.3 Online image processing pipeline 18

Figure 2.4 Parameters derived from contour and bright-field image 21

Figure 2.5 Lookup-table (LUT) 23

Figure 2.6 Contour reconstruction using elliptical Fourier coefficients 28

Figure 2.7 Area of SSCs and MG-63 29

Figure 2.8 Example of a decision tree 34

Figure 2.9 Confusion matrix and normalized confusion matrix 35

Figure 2.10 Multilayer perceptron, activation functions and dropout 39

Figure 2.11 Convolution, maxpooling, feature maps, and residual block 41

Figure 2.12 Generative adversarial networks (GANs) and principle of Cycle-GAN 43

Figure 3.1 Correlations of area vs. volume and deformation vs. inertia ratio 50

Figure 3.2 Correlation Matrix 52

Figure 3.3 Examples of probability plots 53

Figure 3.4 Shape of distributions 54

Figure 3.5 Area and deformation of retina cells at different maturation stages 58

Figure 3.6 GMM for P04 and development of cell size 60

Figure 3.7 Application of three statistical tests to artificial and experimental data 62

Figure 3.8 Boxplots for area and deformation and statistical analysis using LMM 63

Figure 3.9 Unsorted Nrl-GFP retina cells in RT-FDC 68

Figure 3.10 Area/ fluorescence gating and fl.-peak assignment 71

Figure 3.11 Cell classification using area gating 74

Figure 3.12 Performance of random forest that uses online features 75

Figure 3.13 Feature importance of online and offline features 77

Figure 3.14 Performance of random forest that uses 15 online and offline features 80

Figure 3.15 Screening of MLP architectures 83

Figure 3.16 Screening of CNN architectures 85
Figure 3.17 Image distortion by lithium niobate substrate 86
Figure 3.18 Image augmentation by rotating and flipping 87
Figure 3.19 Image transformation using Cycle-GANs 90
Figure 3.20 Training MLPs and CNN using 0% and 20% generated data 91
Figure 3.21 Performance of DNN based cell classification 94
Figure 3.22 Confusion matrices for DNNs 95
Figure 3.23 Performance of models on validation and testing set 96
Figure 3.24 Rod-identification in full range of fluorescence values using MLP_1 98
Figure 3.25 Appearance of correctly predicted cells 99
Figure 3.26 Correlation between DNN output and inertia ratio 100
Figure 3.27 Main window of AID 107
Figure 3.28 Training window of AID 110
Figure 3.29 Reviewing a training history in AID 111
Figure 3.30 Assessing a trained model in AID 112
Figure 3.31 Preventing accidental sorting of multiple cells 114
Figure 3.32 Performance of MLP_1 on validation set 118
Figure 3.33 RT-FDC analysis after sorting rod precursor cells 120
Figure 3.34 Manual gating for different blood cell types 122
Figure 3.35 MLP screening and training 124
Figure 3.36 RT-FDC analysis after sorting neutrophils 125
Figure A.1 Inference time of dense vs. convolutional layer 132

List of tables

Table 1 Activation functions 38

Table 2 Data leveraged for meta-analysis 47

Table 3 Nr. of cells when using GMM-based fluorescence gates 69

Table 4 Nr. of cells when using histogram-based fluorescence gates 69

Table 5 Nr. of small GFP$^+$ and small GFP$^-$ cells in training, validation and testing set 72

Table 6 Feature importance values of random forest model that uses 35 features 78

Table 7 Validation accuracy with and without generated data 92

Table 8 Comparison of c_{40} for all classification methods 97

Table 9 Training and validation data of retina cells 118

Table 10 Training and validation data of different blood cells 122

1. Introduction

Cells are the smallest building blocks of live. Different cells have different functions implicating different structure, protein content, surface molecules and stiffness. The ability to measure such properties allows distinguishing different cell types and changes occurring, for example during maturation, differentiation, infection or drug treatment. Identification of subpopulations and description of their unique properties is an essential first step in biomedical research. For refined analysis, drug testing, or transplantation, often purified samples of certain cell types are required, evocating the need for cell sorting [1,2]. Popular sorting methods facilitate molecular labelling, in which subpopulations are identified based on specific antigens being expressed by the cells. These antigens are then used to bind either magnetic or fluorescent labels to it and to sort the corresponding cells using magnetic or fluorescence activated cell sorting (MACS, FACS), respectively [3–5]. Such labelling can be costly with respect to time and reagents and it cannot be ignored that cells might alter due to the label or the labelling process, rendering the sorted sample incompatible for certain applications such as transplantation. Hence, there is an increasing interest in alternative sorting techniques that employ intrinsic properties of cells. Existing techniques such as elutriation [6], filtration-based approaches [7,8] and deterministic lateral displacement (DLD) [9,10] allow to enrich cells with respect to their density, size, and deformability. Those techniques are bulk-sorting devices (such as MACS), which work passively and therefore reach preeminent throughput. The disadvantages are that the sorting logic is hard-wired into the system and that the specificity of molecular labeling is not met. In contrast to bulk sorters, flow cytometers such as fluorescence activated cell sorters (FACS) assess each cell individually and a sorting decision can be made dependent on an arbitrary combination of the available features. FACS can actually also be used in a label-free manner by employing forward and side scatter on unstained samples, allowing to get an indirect measure of cell size and granularity [1,11].

Real-time deformability cytometry allows obtaining such measures more directly by analyzing bright-field images of single cells. RT-DC is a microfluidic technique, in which cells flow through a narrow channel where they are aligned, deformed, and captured by a high-speed camera [12]. Acquired images are analyzed in real-time at rates of 1000 cells/s, but the data is also stored on hard-disk, allowing to derive any mechanical or texture property in an offline analysis (a more detailed introduction to mechanical and texture features is provided separately in section 1.1).

Recently, real-time deformability and fluorescence cytometry (RT-FDC) was introduced, which extends the existing RT-DC technique by the capability to measure a fluorescence signal simultaneously to the known RT-DC parameters [13]. This allows to link the fluorescence signal from established markers with label-free quantities such as cell size and deformation which are derived from bright-field images. RT-DC and RT-FDC stand out from other label-free measurement techniques (see section 1.3) due to the real-time analysis capability. Real-time analysis plays a key role for prospective sorting, since sorting can only be triggered after certain cell-parameters are computed. Due to the flow cytometry-like approach of RT-FDC, it is possible to implement a sorting unit subsequent to the measurement region, as published recently [14]. This device is called sorting real-time fluorescence and deformability cytometry (soRT-FDC). Typically, in RT-FDC only a very limited number of features such as cell size, deformation and transparency are computed in real-time. In this thesis I want to explore the potential of further label-free features, obtained from bright-field images from RT-DC, RT-FDC or soRT-FDC in the context of cell classification and prospective sorting. The aspect of cell sorting highlights the need for classification in real-time. Further methods for label-free cell characterization are addressed separately in section 1.3.

Cell transplantation is a major motivation for label-free sorting because transplantation material should not be contaminated with labeling compounds in a clinical setting. Therefore, this thesis will use the example of photoreceptor transplantation, for which highly pure samples of photoreceptors are required. A more detailed introduction to photoreceptor transplantation is provided separately in section 1.2. Sources for the required cells could be other retinae (from a primary source) or organoids originating from induced or pluripotent embryonic stem cells. In either case, such a source will contain a mixture of different cells. Some sources of retina cells are already well established and fluorescent labels for certain subpopulations are known. Such known labels can serve as a ground truth to benchmark the performance of new (label-free) classification methods. In this thesis I will aim at reaching the specificity of a molecular marker for rod precursor cells called Nrl-GFP, using bright-field images of the cells only.

After the introduction, all results are presented in five sections of chapter 3. Section 3.1 will highlight some specific features, computed from RT-DC measurements and a meta-analysis using data from several thousands of experiments is performed to assess correlations and distribution properties. In section 3.2, these features are used together with machine learning to characterize retina samples at different maturation stages. Statistical analysis of differences between maturation stages is performed using linear mixed models. In section 3.3 supervised machine learning methods are presented, which allow identifying photoreceptor cells based on bright-field images. Several methods including random forests and deep neural nets (DNNs) are screened for an algorithm which provides sufficient accuracy and a computational speed suitable for sorting. In section 3.4.1, a software called AIDeveloper (AID) is introduced, which allows to train DNNs and to convert the resulting models into a format that can be used by the software, which runs the soRT-FDC setup (the Sorting Software is courtesy of Martin Nötzel, see section 3.4.2). In section 3.5, I go beyond the scope of this thesis and use all developed methods and tools to actually perform image-based sorting of rod photoreceptors. Furthermore, in a second experiment, I sort neutrophils from human blood, showing that the methods can be applied for multiple experimental settings.

The experimental setup evolved continuously during the work on this project and the transitions are reflected by individual sections in chapter 3. Sections 3.1 and 3.2 leverage data from the basic RT-DC setup [12]. RT-FDC [13], which allows a simultaneous measurement of fluorescence intensity parallel to the known RT-DC parameters is introduced in chapter 3.3. Finally, in section 3.5, soRT-FDC [14] is utilized, which is a combined setup, integrating a sorting capability into the RT-FDC setup.

1.1. Texture and mechanical properties: label-free markers

All cells contain cytoplasm, ribosomes and DNA and eukaryotic cells also have membrane enclosed organelles to fulfill particular functions. All of these contents have a different structure, with different absorption, diffraction and refraction behavior leading to a specific appearance. As a result, some cell types can be distinguished by eye based on bright-field images, captured with the help of a microscope. For example, the amount of granules which have a very low transmission rate are usually distinct in neutrophils, monocytes and eosinophils which leads to a recognizable brightness difference of the cells as shown in Figure 1.1 [15]. Furthermore, texture properties such as Haralick features, scale-invariant feature transform, local binary patterns or threshold adjacency statistics can be used to predict pathological conditions such as skin-cancer and malaria or even protein expressions [16–19].

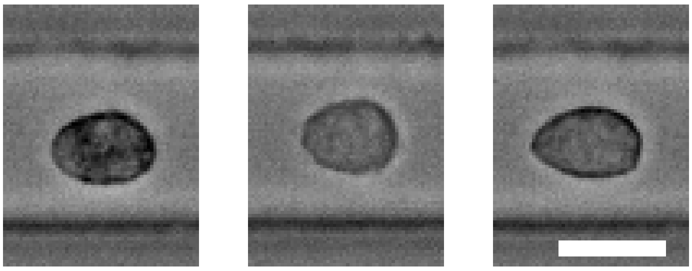


Figure 1.1 Bright-field images of white blood cells captured using RT-DC

Eosinophils (left) have the highest number of granules, causing a dark appearance in a bright-field image, captured using RT-DC at 40x magnification. Monocytes (middle) appear bright since they have an even lower number of granules than neutrophils (right). Scale bar: 10 µm. Measurement was performed using blood of a healthy donor in a microfluidic chip for RT-DC with 20 µm channel width and a flowrate of 0.04 µl/s.

The cell integrity and shape are maintained by the cytoskeleton, a dynamic meshwork of actin, microtubules and intermediate filaments that is not at equilibrium. The amount and architecture of the meshwork components influences the ability of the cell to migrate and deform upon mechanical stress. Mechanical properties of cells can be used to detect pathological alterations of monocytes after inflammation, of red blood cells during plasmodium-infection, or of neutrophils during psoriasis [20–23]. The ability of cancerous cells to migrate is linked to elasticity and allows recognition of metastatic cells [24,25]. Furthermore, stem cells, mature cells, as well as cells differentiating along certain lineages show differences in their mechanical properties [26,27].

Textures as well as mechanical features are measurable without the need of any labelling, which causes a high interest in methods that leverage these physical properties for quantification of cells.

1.2. The retina, diseases and cure by photoreceptor transplantation

Visual perception is arguably one of the most important senses of humans. The eye is the organ which allows us to capture light from our environment, which is then processed by the brain. The optical system of the eye consisting of cornea, lens, and vitreous body guides the light onto the retina, a tissue that layers the back of the eye as shown in Figure 1.2 A. In the retina, electromagnetic waves (light) are converted into electrochemical signals by photoreceptors. These signals are then transmitted and post-processed by bipolar, amacrine and ganglion cells and exit the retina through the optic nerve (see Figure 1.2 B). These different cell types are embedded into the retina which shows a stratified structure.

Color vision on the one hand and scotopic vision on the other hand is achieved by two different types of photoreceptor cells: cones and rods. Cone photoreceptors require high light levels to operate and are densely distributed at the macula and provide high spatial acuity. The human eye usually contains approximately 6 million cones. Black and white contrast and visual perception at low-light conditions is provided by rod photoreceptors, which are more abundant (approximately 120 million in a human eye) [28]. Rods are widely distributed over the whole extend of the retina, but are absent in the macula.

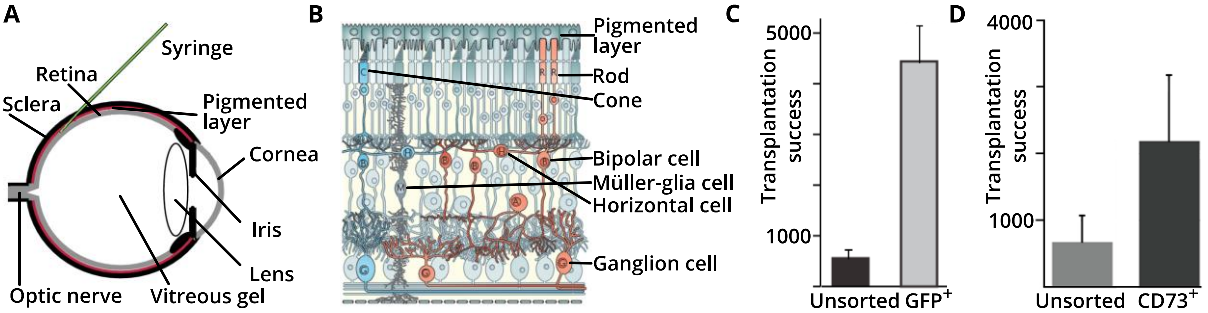


Figure 1.2 Eye, retina and photoreceptor transplantation

(A) Anatomy of the eye and a syringe indicating the aimed subretinal space for photoreceptor transplantation. Light is guided through the eye by cornea, lens and vitreous body and projected onto the retina. Image is adapted from [29].

(B) The retina shows different layers and contains two types of light sensitive cells called rods and cones. Light would enter from the bottom. Image is adapted from [30].

(C) The transplantation success (measured as number of GFP$^+$ cells counted in the host retina) is increased by transplanting purified photoreceptor cells. Purification was achieved utilizing fluorescence activated cell sorting (FACS) to sort for the GFP tagged rod precursor cells of the Nrl-GFP mouse line. Image is adapted from [29].

(D) Another method to purify rod photoreceptors is magnetic activated cell sorting (MACS), utilizing the CD73 antigen, which is expressed on the surface of rod photoreceptors. Image is adapted from [31].

Diseases such as age-related macula disease (AMD) or retinitis pigmentosa (RP) result in an impairment of cone and rod photoreceptor function, respectively. A promising therapeutic approach to recover vision is the transplantation of photoreceptor precursor cells into the subretinal space (between retina and retinal pigment epithelium, see Figure 1.2) of the host retina [32]. Using a mouse model, it was shown that after transplantation, the precursor cells differentiated to mature rod photoreceptors and resulted in improved vision [29,33,34]. The best transplantation success was observed when using retina cells from mice at postnatal day four (P04) [2,32,35]. In the corresponding studies, a transgenic mouse line was used, which expresses green fluorescent protein (GFP) in rod precursors via a neural retina leucine (Nrl) zipper gene promoter [2,29,31–36]. The GFP tag allows locating transplanted rods in the host retina after transplantation. Furthermore, fluorescence-activated cell sorting (FACS) can be used for sorting rods to produce highly pure samples for transplantation. It has been shown that transplanting FACS-enriched rod photoreceptors results in higher transplantation success as compared to transplanting whole unsorted retina (see Figure 1.2 C). A second purification method utilizes the CD73 antigen, which is expressed by rods. Magnetic beads, which are linked to a CD73 antibody, are mixed into the dissociated retina sample, where they bind to the CD73 antigens on rods. This allows separating rods using magnetic-activated cell sorting (MACS). This purification method also results in an improved integration of rod photoreceptors as shown in Figure 1.2 D [31].

Both sorting methods require to add labelling molecules which could contaminate the sample, cause alterations of the cells, and are therefore not applicable in a clinical setting for example to collect cells photoreceptor transplantation. The MACS-based approach requires binding an antibody to a CD73 antigen, which could cause reactions and alterations of the cell. Furthermore, the CD73 antigen is not a functional marker for rods, but is present on other cells as well, for example B and T lymphocytes. For FACS, a fluorescent label is necessary, which could also be facilitated using the CD73 antigen, giving rise to the same problems mentioned for MACS. Alternatively, cells can be genetically modified to express a GFP tag, but the genetic modification could lead to side effects. For clinical research and cell therapy, aimed for the application on human tissue, the transplantation of mouse cells is inappropriate since xenotransplantations often fail due to immune barriers. Additionally, the transplantation of genetically modified cells into humans is ethically problematic. An allotransplantation could be realized using induced or embryonic pluripotent stem cells (PSC) of human origin that are directed to differentiate towards retinal fate in vitro [37–42]. This approach would also not yield a uniform distribution of highly concentrated photoreceptors of the same maturation stage because the cell differentiation pathway and cell cycle cannot be synchronized for all cells simultaneously. Hence, material from such organoids would need to be enriched for photoreceptors before transplantation and the required labelling process would result in the same implications, as mentioned above. Therefore, a label-free cell identification and sorting technology would also be required for retina cells originating from embryonic or induced PSCs.

In this thesis the Nrl-mouse line whose rod photoreceptors express a GFP tag will be used as a model system to characterize retina cells and to develop methods for label-free cell identification for prospective sorting.

1.3. Technologies for label-free assessment of cells

While flow cytometry is commonly used in combination with fluorescent labels, it also allows to use the scatter signal to capture information linked to the refractive index and cell size without need of fluorescent labels [43], but the information content is very limited. Imaging flow cytometers allow to obtain bright and dark-field images of the cells, but a mechanical readout and sorting capabilities are lacking [44,45]. Further label-free measurement methods are Raman scattering, to detect chemical substances [46,47] and quantitative phase imaging, to measure distributions of the refractive index within the cell [48–51]. Mechanical properties have emerged as a promising feature for label-free discrimination of cells and to detect malignant changes for example during cancer or malaria [15,52]. Available techniques include Brillouin scattering [53,54], micropipette aspiration [55], optical tweezer [56], optical stretcher [57] and atomic force microscopy [58]. These methods reach a throughput of approximately 1 cell/minute, rendering sorting for transplantation material unfeasible since several tens to hundred thousand cells are required.

A significant increase in throughput has been achieved by techniques leveraging microfluidics. Micro-constriction arrays utilize tight constrictions, smaller than the nucleus of cells and measure the entry time for approximately 3 cells/second [20,59]. Even higher throughputs were achieved by a similar method called suspended microchannel resonator. There, the microfluidic chip with constrictions is placed onto an oscillating cantilever to speed up the measurement of the entry and passage time employing changes of the resonance frequency. While the buoyant mass of cells can be deducted, a bright-field image of the cells is not captured [60]. Hydropipetting achieves contact free deformation of cells by a rapidly accelerating them using perpendicular flows, which is captured using a high speed camera [61]. Similarly, deformability cytometry (DC) uses extensional flow to decelerate cells, to cause deformation [62]. Hydropipetting and DC achieve a throughput similar to commercial flow cytometers (Hydropipetting approximately 65,000 cells/second and DC approximately 2000 cells/second), but data is stored on camera during the experiment and analyzed afterwards, rendering cell sorting impossible.

Intelligent image-activated cell sorting (iIACS) is also a microfluidic technique, in which cells flow through a channel and bright-field as well as 2D fluorescence images (reconstructed from one dimensional profiles) are captured [63]. Analysis is performed in real-time of up to 100 cells/s and optionally, a sorting unit is triggered to separate individual cells. Since the computational time for image reconstruction and analysis is 32 ms and the speed of cells is 1 m/s, cells travel a distance of 3.2 cm from imaging to sorting region. The number of sorted cells, reported in the original publication range from several hundred to 5000 cells. To find back those cells for post-analysis, a centrifugation-based device for cell-counting needed to be developed. The throughput of iIACS is certainly sufficient to collect cells for prospective single cell omics analyses or cultivation, but for transplantation purposes such as photoreceptor transplantation, approximately 10 to 50 times more cells are required.

Since RT-DC allows a throughput of more than 1000 cells/s and latency is approximately 1 ms, it is a promising tool to perform label-free assessment and sorting of cells at quantities, suitable for transplantation. Furthermore, such an image-based sorting device allows to separate mechanically or morphologically distinguishable cells for example for prospective single cell omics analyses [64,65], cell culture, or creation of particular drugs [64,66,67].

2. Materials and Methods

The previous chapter already mentioned techniques for measuring label-free properties of cells, including RT-DC and RT-FDC. In this chapter, RT-DC and RT-FDC are introduced in detail, as well as an additional sorting module. Data analysis techniques are introduced, which allow extracting more information from individual cells or full measurements. Furthermore, statistical analysis techniques for the comparison of datasets and machine learning methods for cell classification are presented.

2.1. Experimental setup

2.1.1. Chip design for RT-DC and RT-FDC

The microfluidic design of Real-time deformability cytometry (RT-DC) and Real-time deformability and fluorescence cytometry (RT-FDC) chips is shown in Figure 2.1. Two inlets (I_1 and I_2 in Figure 2.1) allow inserting tubing carrying sheath fluid (dark blue in Figure 2.1) and the suspended cells (cyan in Figure 2.1). Filter units close after both inlets (‘P_1’ and ‘P_2’ in Figure 2.1) help to block large particles to avoid clogging of the chip. The chip design for the cell suspension is tightened to 100 µm, a region called ‘reservoir’ (‘ROI_2’ in Figure 2.1). This region can be used to capture slowly moving cells that are subjected to very low stress. Next, the cell suspension flow is focused by a sheath flow towards a constriction channel. At the backmost part of this 300 µm (RT-DC and RT-FDC) or 880 µm (soRT-FDC) long channel, cells are captured in the deformed state (‘ROI_1’ in Figure 2.1). In order to avoid contact of cells to the channel wall and to still have sufficient hydrodynamic forces, the channel width has to be chosen depending on the size of the cells (common channel widths are 5, 10, 15, 20, 30, and 40 µm). In an ordinary RT-DC chip, the fluid simply exits the chip at a single outlet. Figure 2.1 shows two outlets, which are required for cell sorting to direct target cells into a dedicated collection tube. Section 2.1.2 introduces details of the microfluidic design of sorting chips.

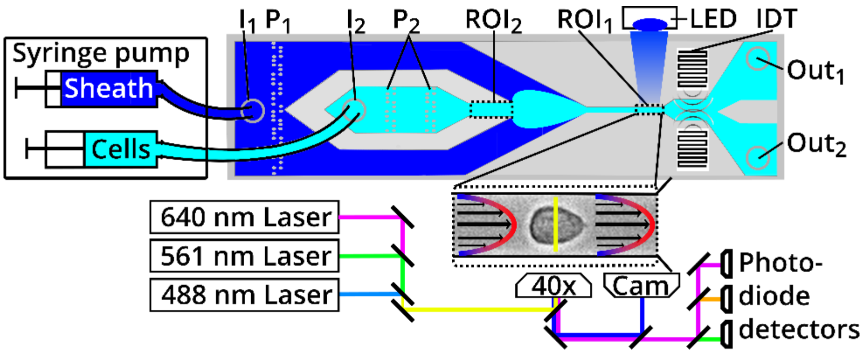


Figure 2.1 RT-(F)DC chip design and soRT-FDC setup components

Sketch shows the chip design and components of the soRT-FDC setup. The microfluidic chip is supplied with sheath flow (dark blue) and cell suspension (cyan) by two coupled syringe pumps. The tubing for the cell suspension is inserted at inlet 2 (I_2). The cell suspension flow is then filtered by pillars (P_2), pre-focused in a narrow region of 100 µm width (ROI_2) and hydrodynamically focused by the sheath flow (dark blue) towards a constriction channel. Usually, cells are captured at the end of the channel (ROI_1) and in the 100 µm wide reservoir (ROI_2). Illumination is achieved by a high-power LED and images are captured using a high-speed camera. Those parts are the basis for the RT-DC setup, (which is employed in section 3.1 and 3.2). In RT-FDC (used in section 3.3), there are additionally three lasers available, allowing to excite fluorescence, which is then detected by photodiode detectors. To add a sorting function to the setup, interdigital transducers (IDTs) are integrated into the chip design, allowing translocating cells using standing surface acoustic waves (SSAW). By default, cells travel towards outlet 1 (Out_1) and in case of a sorting event, cells are pushed into the channel leading to outlet 2 (Out_2). The corresponding setup is called soRT-FDC (used in section 3.5).

2.1.2. Chip design for soRT-FDC

Figure 2.1 displays the schematic design of a sorting chip, where the constriction channel is followed by a 200 µm long channel of 50 µm width. On both sides, interdigital transducers (IDT, see Figure 2.1) are neighboring this part of the chip. Upon actuation, the IDTs create standing surface acoustic waves (SSAW) on the substrate of the chip, which enable to shift the trajectory of cells towards outlet 2 (“Out_2” in Figure 2.1). A slight offset (5 µm) of the dividing wall between Out_1 and Out_2 ensures that cells travel towards outlet 1 when SSAW are not applied.

Typically, RT-(F)DC experiments last several minutes to maximum half an hour. Occasional clogging of the channel due to large particles is simply solved by either increasing the flowrate to force the objects through the channel or by replacing the chip, which only takes minutes. Especially samples, which required a dissociation process to obtain single cells (adherent cells and tissues), often still contain agglomerates of cells, which can simply be disregarded from the measurement by size gating. Hence, for RT-(F)DC, there is no immediate need to change the chip design to further decrease the chance of clogging or having agglomerates of cells in the channel. For sorting experiments, both events are actually highly problematic. Clogging or partial clogging of the channel disturbs the flow profile, likely causing cells to accidentally go to the target outlet. Similarly, parts of an agglomerate could potentially spill into the target outlet, contaminating the respective sample. Changing a sorting chip requires considerably more time since a slow inflow of the sample is required to avoid spilling into the target channel. Furthermore, an optimization of phase and frequency for actuation of the IDTs (see section 2.1.5) is required to achieve translocation of cells into the target channel. Hence, the chip design was reviewed to implement structures, which reduce the chance of clogging or agglomerates in the channel. First, several columns of pillar units with decreasing inner distance were placed in sheath as well as sample inlet (see red rectangles in Figure 2.2). In the final column of pillars, the inner distance between pillars is 15 µm in the sample flow and 10 µm in the sheath flow. Furthermore, assemblies of serpentines were placed into the sample flow (courtesy of Ahmad Ahsan Nawaz), which help to tear apart agglomerates and also increase the spacing between cells (see orange rectangles in Figure 2.2). Each serpentine has a width of 30 µm. The measurement channel has a width of 20 µm and a length of 880 µm, which is followed by a 50 µm wide and 200 µm long section, where SSAW are applied. In the same region there are pockets for IDTs at both sides of the channel.

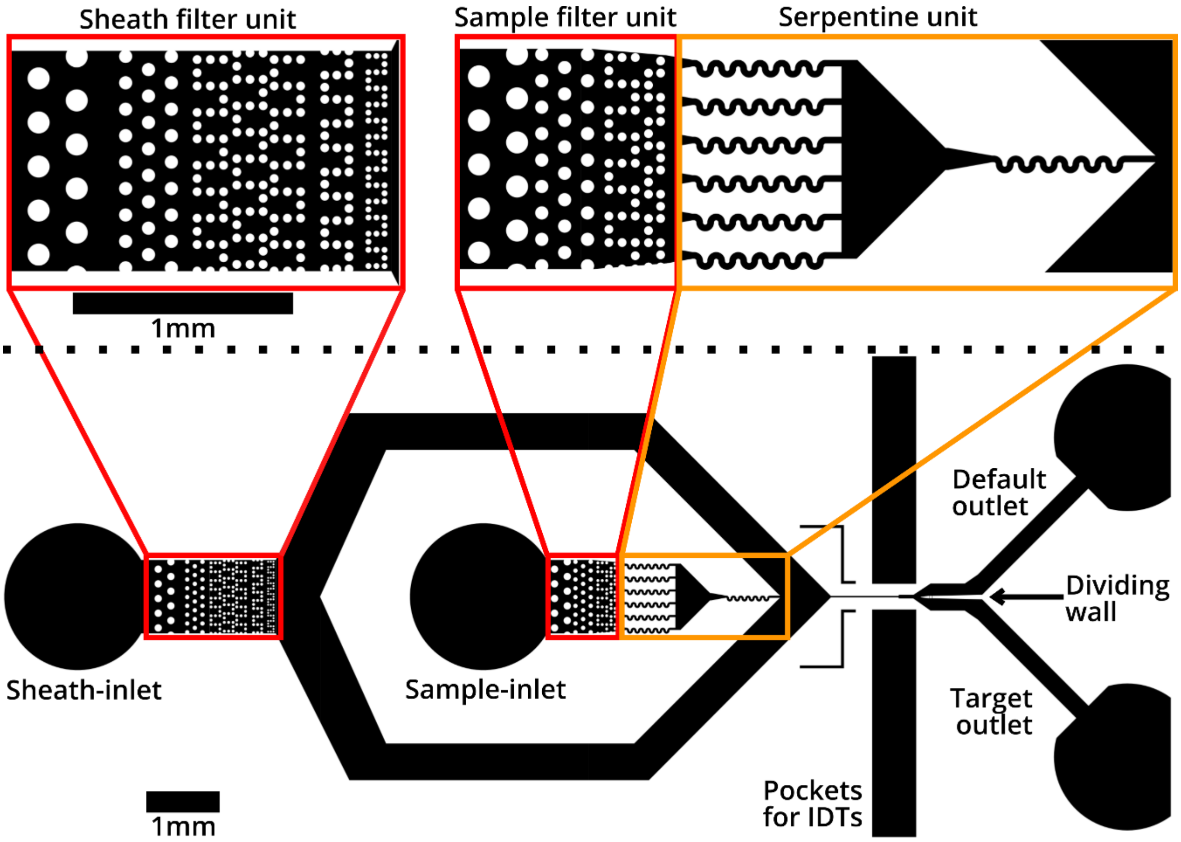


Figure 2.2 Design of the sorting chip

Figure shows the 2D-CAD design of the entire sorting chip and a zoomed in version of specific parts. The red rectangles indicate unique filter assemblies, which consist of a cascade of pillars with decreasing inner distance. The orange rectangles indicate a unit of several serpentines, which helps to tear apart clusters of cells and to increase the spacing between cells. The layout was designed using KLayout 0.25.3.

2.1.3. Chip fabrication

The two-dimensional design is first brought onto a photomask, which is then used to copy the structure to a silicon waver by photolithography [12]. Chips are produced using a combination of polydimethylsiloxane (PDMS, SYLGARD®, Dow Corning, USA) and curing agent (10:1 w/w), which is distributed on the master to create a 0.5 cm thick layer. Layer and master are separated again after curing the PDMS for 60 min at 65 °C, which allows using the master several times. Holes for connecting tubes at inlets and outlets (I_1, I_2, Out_1, and Out_2 in Figure 2.1) are cut using a biopsy puncher (Biopsy Punch with Plunger, size 1.5 mm, pfm medical AG, Germany).

In the final step of chip production, the structures in the PDMS layer need to be sealed. Chips that are not intended for sorting are sealed using a glass slide (thickness 2, Hecht, Germany) that is bound covalently to the PDMS by means of plasma activation (50 W, 30 sec, Plasma Cleaner Atto, Diener electronic, Germany). Instead of glass, sorting chips are sealed using a 128° Y-cut lithium niobate ($LiNbO_3$) substrate. Lithium niobate is piezoelectric, which allows generating surface acoustic waves by electrical excitation using interdigitated transducers (IDT, see Figure 2.1). The IDTs are composed of chromium and gold layers (Cr/Au, 10 nm/70 nm, respectively), which are brought onto the substrate by an evaporation process. The design of the IDTs (30 pairs of electrodes with an aperture of 200 µm and an inter-finger distance of 70 µm) results in an excitation frequency of 55.23 MHz and an acoustic power of the SSAW of -2.8dbm [14]. In addition, alignment markers are deposited onto the substrate during the evaporation process. These markers allow to precisely align structures on PDMS and substrate during chip assembly. The punching of holes for the two inlets (sheath and cells) and outlets (Out_1 and Out_2) as well as the bonding of PDMS and substrate by plasma activation is identical for sorting and non-sorting chips.

2.1.4. RT-DC, RT-FDC and soRT-FDC Setup

A stage with magnetic clamps is used to fix the microfluidic chip on an inverted microscope (Axio Observer Z1, Zeiss, Germany), equipped with a 40x objective (NA = 0.75, Neofluar, Zeiss, Germany) or alternatively a 20x objective (NA = 0.8, Plan-Apochromat, Zeiss, Germany). A high-speed camera (EoSens CL MC1362, Mikrotron, Germany) with a 1280x1024 pixels CMOS sensor captures bright-field images with a final resolution of 0.34 µm/pixel (40x magnification) or 0.68 µm/pixel (20x magnification), which are sent to a standard PC using a full camera-link frame grabber card (PCIe-1433, National Instruments, USA). To avoid image blurring the camera triggers an LED (CBT-120, Luminus Devices, USA) which sends 2 µs long light pulses for illumination of every captured frame.

Precise flowrates for sheath and cell suspension fluid are controlled by two linked syringe pump modules (NemeSyS, Cetoni, Germany). PEEK tubing (Postnova Analytics, Germany) is used to connect the syringes with the microfluidic chip. Only for sorting experiments a third syringe pump is used which withdraws fluid from the default outlet.

This allows adjusting the flow profile to optimize the trajectory of cells, such that they traverse into the default outlet (Out$_1$ in Figure 2.1), close to the dividing wall (see Figure 2.2).

To control the setup components and perform data acquisition, ShapeIn (Zellmechanik Dresden, Germany) is used (for RT-DC and RT-FDC experiments), which utilizes the OpenCV library [68] for image processing. Continuously, a rolling average of the last 100 frames is computed. This background image is then subtracted from the latest image. Next, the image is smoothed and binarized by thresholding to finally obtain the contour using a border following algorithm [69] (for more details see Figure 2.3).

Optionally, the setup is complemented by three lasers (OBIS 640 nm LX 40 mW, OBIS 561 nm LS 50 mW and OBIS 488 nm LS 60 mW, Coherent Deutschland, Germany), which allow to excite fluorescence. Each laser excitation beam is focused to form a light-sheet in the region of interest in the microfluidic channel (ROI$_1$ in Figure 2.1). Each cell traverses this sheet and the emitted fluorescence is collected by the objective and guided to a photodiode detector assembly (MiniSM10035, SensL Corporate, Ireland), resulting in up to three 1D fluorescence traces for each captured cell [13]. Bright-field images and fluorescence traces are acquired synchronously for each single cell.

To create standing surface acoustic waves for sorting, the signal of a surface acoustic wave generator (BSG F20, BelektroniG, Germany) is duplicated using two fast-switches (BPS-300, BelektroniG, Germany) and connected to each IDT. Instead of glass, microfluidic chips for sorting use a $LiNbO_3$ substrate, which is birefringent. To remove the resulting image distortion, a polarizer (Polarizer D, Zeiss, Germany) is used.

For controlling the sorting system, dedicated software was developed, based on the C++ code for RT-DC. In case of RT-(F)DC queues of events are temporarily stored on RAM and multiple CPU-cores analyze stored events in parallel. For sorting, such a queue would result in varying computational time for each event. Therefore, the sorting software was optimized to strictly process only single events as fast as possible. The computational time for a single event is approximately 150 µs and the total delay between image acquisition and sorting trigger is 1 ms [14]. Depending on the computed quantities (e.g. area and deformation) and user defined thresholds for sorting, a trigger signal is send from the frame grabber card to the SAW generator, inducing a sorting pulse. Further details on the Sorting Software are presented in section 3.4.2.

2.1.5. Physics of surface acoustic wave mediated sorting

The capability to sort cells based on real-time parameters is achieved using standing surface acoustic waves (SSAW). Upon actuation of the opposing IDTs at resonance frequency, counter propagating surface acoustic waves are generated due to the piezoelectricity of lithium niobate. Each IDT consists of a pair of comb-shaped electrodes that are interlocked, but not touching. The inner distance between electrodes defines the resonance frequency of the IDT: $f = v/\lambda$, where v is the speed of sound in lithium niobate v ($v = 3978.2\,m/s$ [70]) and λ is the wavelength, defined by the distance between two electrodes. When two opposing IDTs are excited simultaneously at the same frequency and amplitude, the counter-propagating SAW generate a standing surface acoustic wave (SSAW). In the design used for sorting chips, the wavelength of the SSAW is $\lambda = 70\ \mu m$ and the distance between the IDTs is 350 µm. Particles or cells interact with the SSAW via acoustic radiation force, which is described by [71]:

$$F_t = -\left(\frac{\pi p_0^2 V_c \beta_w}{2\lambda}\right)\phi(\beta,\varrho)sin(2kx), \qquad 2.1$$

with the acoustic contrast factor ϕ,

$$\phi(\beta,\varrho) = \frac{5\varrho_c - 2\varrho_w}{2\varrho_c + \varrho_w} - \frac{\beta_c}{\beta_w}, \qquad 2.2$$

acoustic pressure p_0 , wavelength λ, cell volume V_c, cell compressibility β_c, fluid compressibility β_w, cell density ϱ_c and fluid density ϱ_w. The acoustic factor determines if the cells travel towards the pressure node of the SSAW ($\phi > 0$) or towards the pressure antinode. Typically, cells have positive ϕ (assuming a water based surrounding fluid) and move towards the pressure node, on the one hand because cells contain protein, which increases their density in comparison to water ($\varrho_{Water} \approx 1.0\ g/m^3$, $\varrho_{Protein} \approx$

$1.3 \dots 1.4\ kg/m^3$) and on the other hand because cells have a higher compressibility than water ($\beta_w \approx 4.5\ GPa^{-1}$, $\beta_c \approx 4\ GPa^{-1}$) [72–74].

The current design of the IDTs causes a resonance frequency of 55.23 MHz, which corresponds to a wavelength of $\lambda = \frac{v}{f} = \frac{3978.2\ m/s}{55.23\ MHz} = 72.03\ \mu m$ for SSAW on lithium niobate substrate (using the speed of sound of $LiNbO_3$ of 3978.2 m/s [70]). Since the maximum translocation of a cell by a SSAW is only $\lambda/4 = 18.01\ \mu m$, it is important that the trajectory of cells goes very close to the dividing wall (see Figure 2.2) towards the default outlet (Out_1) during a sorting experiment. This fine adjustment of the trajectory of cells is achieved using a third syringe pump which operates at a negative flowrate, effectively drawing liquid out from the default outlet.

2.1.6. Measurement buffer (MB) for RT-DC

In principle, any cell medium could be used and flushed through the microfluidic chip, but it has several advantages to use media with an elevated viscosity. Firstly, higher viscosity increases the shear stress and therefore allows to deform eukaryotic cells already at low flowrates [12]. Secondly, higher viscosity reduces sedimentation which is necessary for longer experiment durations for example for cell sorting. Therefore, the viscosity of a given cell medium is increased to 15 mPas or 26 mPas (zero shear viscosity at 24°C) by adding 0.5% (w/w) or 0.6% (w/w) methyl cellulose (4000 cPs, Alfa Aesar, Germany), respectively. These buffers show a shear thinning effect, which causes a drop of the apparent viscosity when increasing the flowrate. The dependency of viscosity and flowrate has been shown in a publication for a PBS based MB with 0.5% and 0.6% methyl cellulose (MC) in microfluidic chips with a channel width of 20 µm and 30 µm, resulting in the following relations for viscosity:

$$\eta_{0.5\%MC} = 179 \cdot \left(7.922 \cdot \frac{Q}{l^3}\right)^{-0.323} \cdot \left(\frac{\vartheta}{23.2}\right)^{-0.866} \qquad 2.3$$

for buffer containing 0.5% MC, and

$$\eta_{0.6\%MC} = 360 \cdot \left(8.093 \cdot \frac{Q}{l^3}\right)^{-0.366} \cdot \left(\frac{\vartheta}{23.2}\right)^{-0.866} \qquad 2.4$$

for buffer containing 0.6% MC [75].

For the experiments in this thesis, measurement buffer based on phosphate-buffered saline without magnesium or calcium (PBS⁻) and 0.5% and 0.6% MC were prepared. Furthermore, a measurement buffer based on PBS⁻, complemented with 10% Leibovitz medium (Thermo Fischer, Germany) and 0.6% methyl cellulose was prepared. The viscosity of each buffer was adjusted using a falling ball viscometer (Haake, Germany).

2.2. Online parameters

The main improvement RT-DC introduced for microfluidics-based assessment of cell mechanics is the capability to analyze images in real-time. To make this possible, image analysis has to be performed at least at the same rate as image acquisition of the camera. This requirement is met by a camera, operating at thousand frames per second and an image analysis pipeline, which requires approximately 1 ms to process one image. Real-time analysis allows to only save frames which actually contain interesting objects, while images of the empty channel or debris can be omitted. Additionally, the experimenter can observe measured quantities while running the experiment, and, most importantly for the project of this thesis, parameters calculated in real-time can be used to trigger sorting. Due to a direct streaming of the data to a hard disk drive, the duration of the experiment is practically only limited by sample volume. The image analysis pipeline is sketched in Figure 2.3.

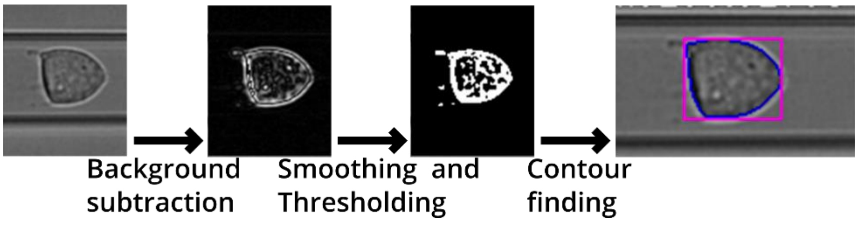


Figure 2.3 Online image processing pipeline

For fast image processing, RT-DC uses an image processing pipeline, which consists of three steps: background subtraction, binarization and contour finding. Figure is adapted from [76].

A rolling average of the last 100 images delivers a background image, which is subtracted from the current image. After smoothing and thresholding operations, the resulting binary image is used to perform a contour finding algorithm [69], which returns the contour of the cell as indicated in Figure 2.3. The contour and the bright-field image are used to compute online parameters, but are also stored on hard disk to allow for ancillary offline analyses. The following list summarizes parameters that are computed in real-time.

Area (A_{hull} or A)

The convex hull of the contour defines a cross-sectional area of the cell A_{hull}, which is a parameter linked to cell size (see Figure 2.4).

Length (L_x) and height (L_y)

The bounding box is determined using the contour and allows to estimate the length L_x and height L_y of the object (see Figure 2.4).

Aspect ratio (γ)

The aspect ratio is defined by the ratio of height L_y and length L_x: $\gamma = \frac{L_x}{L_y}$.

Circularity (C) and deformation (D)

Circularity expresses how well a shape matches a perfect circle and is defined by the area of the convex hull A_{hull} and the corresponding perimeter (P_{hull}) using the equation $C = \frac{2\sqrt{\pi A_{hull}}}{P_{hull}}$. The convex hull is used since dents or protrusions would cause an increase of the perimeter while reducing the area, which would result in a strong decrease of the circularity value. For example, the cell in Figure 2.4 shows a deep dent in the upper left. Deformation (D) is calculated as following: $D = 1 - C$.

Inertia Ratio (I)

The contour of an object describes how area is distributed in space. Contours of elongated objects contain more patches of area that are more distant from the centroid, which can be quantified by the second moment of area. The second moment with respect to the x-direction is $I_{xx} = \iint_A y^2 dx\, dy$ and with respect to the y-direction $I_{yy} = \iint_A x^2 dx\, dy$. When increasing the size of an object, both I_{xx} and I_{yy} will increase. Therefore, to get a measure of how deformed an object is, the inertia ratio $I = \frac{I_{yy}}{I_{xx}}$ is calculated. In contrary to circularity, inertia ratio is not computed using the convex hull, but the original contour (red in Figure 2.4).

Porosity (Ω)

Figure 2.4 shows a deep dent in the upper left part of the cell, causing a difference between the area of the convex hull and the area of the original contour. Porosity (Ω) quantifies this difference using $\Omega = \frac{A_{hull}}{A_{contour}}$. Related terms for porosity are “Area ratio” which is a synonym, and “solidity” which is the inverse of porosity $S = \frac{1}{\Omega}$.

x and y position (c_x and c_y)

The contour allows to compute the centroid (c_x, c_y), which is used to define the position of the cell.

Brightness (B) and standard deviation of brightness (B_{std})

Simple texture and transparency properties are obtained by calculating the mean and the standard deviation of the grayscale values inside the cell (see Figure 2.4) which are denoted as B and B_{std}, respectively.

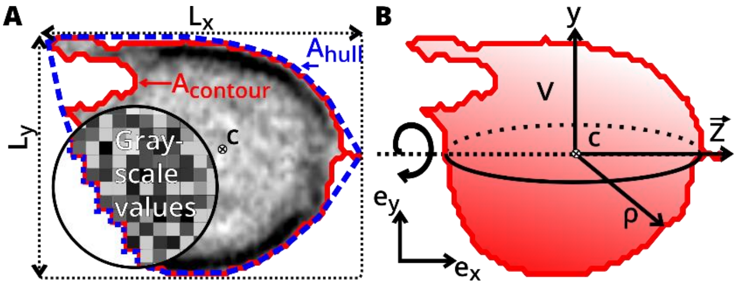


Figure 2.4 Parameters derived from contour and bright-field image

(A) Schematic shows grayscale values (pixels) of a recorded cell and the corresponding tracked contour. Centroid (c), area (A_{hull}), as well as the bounding box are derived from the contour. The bounding box defines the length L_x and height L_y of the cell. The grayscale values inside the contour are used to calculate mean and standard deviation of brightness.

(B) By rotating the contour around an axis $\vec{z}$ which is defined by the centroid and the flow direction, the volume of the cell can be estimated.

2.3. Offline parameters

The following list shows parameters that are not available in real-time, but can be computed after the experiment, using the stored contour and bright-field image.

Volume (V)

By revolving the contour around the central axis $\vec{z}$ ($\vec{z}$ is defined by the centroid of the cell and the flow direction as shown in Figure 2.4 B), the volume can be calculated under the assumption of rotational symmetry. In principle, this assumption would only be valid for cylindrical channels and not for channels with square cross-section, as used in RT-DC, but it was shown that the parabolic flow profile is a good assumption for cells that do not cover more than 90% of the channel width [77]. By revolving the upper and lower half ($\vec{z}$ is the dividing axis) of the contour around $\vec{z}$ individually, two volume estimations are obtained, which are averaged [78]. Green`s theorem and the Gaussian divergence theorem allow to formulate the volume as a line integral and convert the expression into an equation accepting discrete Cartesian coordinates [79,80]:

$$V = \frac{1}{3}\oint_{Contour} x[xdy - ydx] = \frac{2\pi}{3}\int_0^1 (\rho(s)[\rho(s)z'(s) - z(s)\rho'(s)])\,ds \tag{2.5}$$

with $\rho'(s) = d\rho(s)/ds$ and $z'(s) = dz(s)/ds$ and $0 \leq s \leq 1$.

In the equation on the left-hand side, the contour is expressed in Cartesian coordinates, while on the right-hand side the distance from every contour point to the centroid ρ and the coordinate along the central axis $\vec{z}$ is used. Primes in the equation denote derivatives. Because the contour is piecewise linear, derivatives are simply the differences between consecutive contour points. The application of this algorithm to RT-DC data has been published [81,82] and implemented into ShapeOut (courtesy of Paul Müller), the open source analysis software for RT-DC data [83].

Elastic modulus (E)

The elastic modulus (E) is a material parameter, which can be computed for RT-DC measurements, using an analytical model that leverages steady state hydrodynamics and linear elasticity theory and assuming that cells are fully elastic spheres flowing through a cylindrical channel [77]. The validity of this approach is limited to small deformations ($D < 0.02$) since changes of the fluid flow around strongly deformed objects are not regarded by the model. This effect is considered by a numerical model, which validates the analytical model and is applicable also for higher deformation values [84]. In this thesis, the elastic modulus will be abbreviated E_{ana} or E_{num}, to indicate if the analytical or numerical approach was used. Both approaches can be used to calculate the elastic modulus for given area and deformation values, but especially the numerical simulations require a lot of computational time. Therefore, a close-meshed lookup-table (LUT) was generated for both approaches, which can be used to quickly obtain E for whole measurements. LUTs are computed only for a given flowrate Q, channel width l and viscosity η, and the scaling relation $E' = \frac{Q'\eta' l^3}{Q\eta l'^3} E$ allows to convert the LUT to different conditions [77]. An exemplary use case of the lookup table and the scaling relation is shown in Figure 2.5.

For a valid transformation to elastic modulus, certain boundary conditions regarding cell size should be met. Firstly, cells should cover between 30% and 90% of the channel width. For smaller sizes, a minute change in deformation causes a large change of the elastic modulus, which results in large errors. Cells larger than 90% of the channel width could touch the channel wall, causing non-hydrodynamic forces, which are not considered by the models. Since cells are captured using a CMOS camera with a particular pixel size, even perfectly round objects appear pixelated and have a $D > 0$ [75]. This offset in deformation depends on cell size and has to be considered when computing the elastic modulus [75].

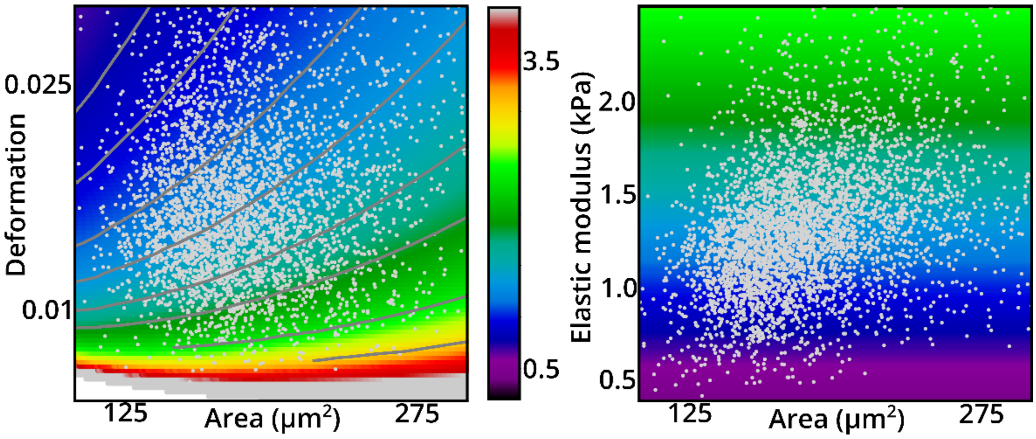


Figure 2.5 Lookup-table (LUT)

Each location in the area vs. deformation space is mapped to a corresponding elastic modulus, which is indicated by a color code from black (low elastic modulus) to red (high elastic modulus). The shown LUT was obtained using the numerical approach [84]. An exemplary measurement of the human breast epithelial cell line MCF10A is shown as scatterplot in area vs. deformation and also area vs. elastic modulus space (the measurement was performed in a 30 μm chip at a flowrate of 0.16 μl/s) [85].

Principal inertia ratio and orientation (φ)

Principal inertia ratio is derived from inertia ratio (section 2.2), and is rotation-invariant. To achieve rotation invariance, first, the orientation of the cell with respect to $\vec{e}_x$ is computed by [86]:

$$\varphi = \frac{1}{2} arctan\left(\frac{2I_{xy}}{-(I_{yy} - I_{xx})}\right) \tag{2.6}$$

with I_{xx} and I_{yy} as introduced in section 2.2 and the biaxial second moment of area

$$I_{xy} = -\iint_A xy\,dx\,dy. \tag{2.7}$$

Next, the contour which is given in Cartesian coordinates is translated into polar coordinates and rotated by $-\varphi$ and subsequently translated back into Cartesian coordinates to compute the inertia ratio as introduced in section 2.2. Principal inertia ratio was implemented into ShapeOut (courtesy of Paul Müller)[83].

Haralick texture features

The average or the standard deviation of the gray scale values are very simple features describing the texture of an object. But when we try to describe the look of a cell we would use adjectives such as "blurry" and "glossy" or argue using the number of speckles. The Haralick texture features quantify such texture properties by leveraging the so-called co-occurrence matrix. An image with p different grayscale values results in a co-occurrence matrix with $p \times p$ dimensions, where the matrix item at location (i, j) shows how many pixels with the grayscale value i are bordering on pixels with the grayscale value j. The Haralick features are 13 defined quantities obtained from this matrix [87]. In this thesis the implementation of Haralick texture features from the Python package "mahotas" was used [88].

Local binary pattern (LBP)

Local binary patterns (LBPs) quantify local texture properties by comparing the intensities of each pixel with its surrounding pixels [89]. First, the pixel i is compared to the neighboring pixel j on the top left, resulting in 0 if the pixel i has a larger grayscale value than the neighbor j, or 1 otherwise. This procedure is repeated for all 8 neighbors of pixel i by going clockwise. The resulting 8-digit binary number is then converted to a decimal number, the LBP value. Since there are 8 digits, there are $2^8 = 256$ possible LBP values. This LBP value is then obtained for every pixel in the image. Lastly, a histogram, showing the frequency of LBP values is computed and finally used as feature vector. Therefore, there are in principle 256 values representing texture properties, but only 36 of them are rotation invariant and used in practice. Since 3x3 pixel patches are used to compute LBP values, only very fine-grained image details are expressed. The LBP features are illumination independent, which could be of advantage for example when comparing images from different RT-DC experiments. In this thesis the implementation of LBP from the Python package “mahotas” was used [88].

Threshold adjacency statistics (TAS)

The threshold adjacency statistics (TAS) are also measures of texture, which are obtained after binarization of the image by thresholding [17]. In the binary image, the number of white pixels around each white pixel are counted. There are 9 possible outcomes (a white pixel can have 0,1,2,3,4,5,6,7 or 8 white neighboring pixels) and since the routine is carried out for each white pixel, a histogram is obtained, which shows how often a white pixel is surrounded by a certain number of white pixels. For example, when an image is almost entirely dark with only very few sparse white pixels, most likely, each white pixel would have 0 white neighboring pixels. The histogram is normalized by dividing it by the total number of white pixels in the binary image, resulting in 9 TAS feature values. The appearance of the binary image is dependent on the thresholding operation. Therefore, three different thresholding routines are applied, each resulting in 9 TAS feature values [90]. Additionally, each binary image is also inverted (black becomes white and vice versa) and TAS feature values are retrieved, which results in a total of $2 \cdot 3 \cdot 9 = 54$ TAS feature values. Since TAS are not illumination invariant, the input images were always normalized. Normalization was performed by multiplying each image with a factor, such that the resulting image has a background brightness (median brightness in the middle of the channel) of 100. In this thesis the implementation of parameter free TAS from the Python package “mahotas” was used [88].

Background brightness ($B_{back.median}$ and $B_{back.std}$)

Since there is no automatic focus and illumination adjustment, the brightness of images might alter which affects the mean brightness of cells. For normalization, the background brightness levels can be used. Getting information about the background is straight forward since the captured images of RT-DC are usually much larger than the cells. The background is evaluated in the center of the channel at a location without cell.

I decided to use window size of 20 pixels height (=6.8 μm) and 40 pixels length (=13.6 μm) to determine the background properties. Since most of the cells travel in the middle of the channel, the average y-position of an experiment was used to place the window in the center of the channel. To avoid having a cell inside the window, the x-position of the window was adjusted for each frame such that the window is located behind the cell. The grayscale values inside that window are then used to determine the median ($B_{back.median}$) and the standard deviation ($B_{back.std}$) of the background brightness.

Normalized and maximum brightness (B_{norm} and B_{max})

The median of the background intensity ($B_{back.median}$) should not change during an experiment (typically, focus and illumination are adjusted before an experiment and kept constant). Therefore, $B_{Back.Median}$ which is the median of all $B_{back.median}$ values is a good estimate to describe the illumination setting of an experiment and can be used to normalize the mean brightness value B of each event using:

$$B_{norm} = B \cdot \frac{100}{B_{Back.Median}}. \tag{2.8}$$

The maximum brightness B_{max} of an object was determined by sorting the grayscale values inside the contour and computing the median of the 10 brightest pixels (this is equal to the value of the 5th brightest pixel). This definition of the maximum is slightly more robust than just taking the maximum of all grayscale values, since the impact of single very bright pixels is smaller. Such bright pixels could appear due to noise, or when the contour includes regions of the bright halo which is typically situated around the cell (usually, the contour does not include the halo).

Elliptical Fourier features (EFFs)

The elliptical Fourier features allow a more detailed quantification of the shape of closed contours. Besides the elongation of tracked objects in RT-DC, certain contour properties, such as protrusions or bending curvatures could play an important role for cell classification. EFFs have been used in several biological studies for example to analyze the shapes of blood cells, lamina or leaves [91–93].

A closed contour $M(t)$ with $t = 1 \dots m$ (m is the number of discrete contour points) can be described by a sum of N ellipses, resulting in the following approximations for x and y coordinates of contour points [94,95]:

$$x(t) = A_0 + \sum_{n=1}^{\mathrm{N}} [a_n cos\phi + b_n sin\phi] \tag{2.9}$$

$$y(t) = C_0 + \sum_{n=1}^{\mathrm{N}} [c_n cos\phi + d_n sin\phi] \tag{2.10}$$

with $= \frac{2n\pi t}{T}$, the constants A_0, C_0 and Fourier descriptors a_n, b_n, c_n and d_n. When $N \rightarrow \infty$, the constants become $A_0 = \frac{1}{T}\int_0^T x(t)dt$ and $C_0 = \frac{1}{T}\int_0^T y(t)dt$. Since the functions $x(t)$ and $y(t)$ are piecewise linear, the Fourier coefficients are defined by the following equations [94]:

$$a_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{\mathrm{m}} \frac{\Delta x_i}{\Delta t_i} [cos\phi_i - cos\phi_{i-1}] \tag{2.11}$$

$$b_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{\mathrm{m}} \frac{\Delta x_i}{\Delta t_i} [sin\phi_i - sin\phi_{i-1}] \tag{2.12}$$

$$c_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{\mathrm{m}} \frac{\Delta y_i}{\Delta t_i} [cos\phi_i - cos\phi_{i-1}] \tag{2.13}$$

$$d_n = \frac{T}{2n^2\pi^2} \sum_{i=1}^{\mathrm{m}} \frac{\Delta y_i}{\Delta t_i} [sin\phi_i - sin\phi_{i-1}] \tag{2.14}$$

with $\phi_i = 2n\pi t_i/T$, $\Delta x_i = x_i - x_{i-1}$, $\Delta y_i = y_i - y_{i-1}$, $\Delta t_i = \sqrt{\Delta x^2 + \Delta y^2}$, $t_i = \sum_{j=1}^{i} \Delta t_j$ and $\mathrm{T} = t_m = \sum_{j=1}^{m} \Delta t_j$. These Fourier coefficients allow to describe and reconstruct contours with arbitrary precision, depending on N (see Figure 2.6 A). Higher N results in less deviation from the original contour (see Figure 2.6 B).

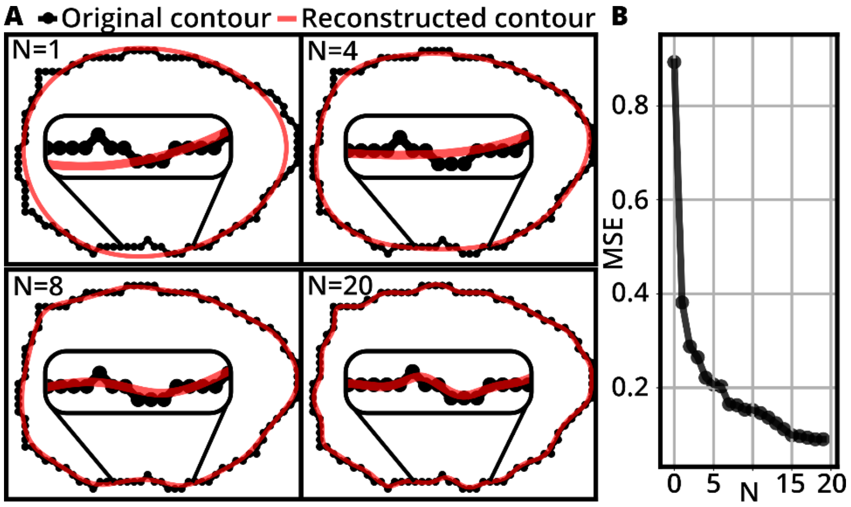


Figure 2.6 Contour reconstruction using elliptical Fourier coefficients

(A) Plots show the contour of a cell (black) from which the elliptical Fourier coefficients of the first N orders are computed. Next, the contour is reconstructed from the coefficients (red). Allowing only one order results in an elliptical fit. By increasing N, the reconstructed contour gets closer to the original contour until even small deviations of single pixels are resolved.

(B) Plot shows the mean squared error (MSE) which describes the difference between original and reconstructed contour. The error decreases the more orders N are used to encode the contour.

To obtain rotation, scale and translation invariant elliptical Fourier features (EFFs), the coefficients a_n, b_n, c_n and d_n are transformed by aligning the semi-major axis with the x-axis and dividing the coefficients by the magnitude of the semi-major axis. In this thesis the coefficients for $\mathrm{N} = 10$ orders were computed using the Python package PyEFD [96], resulting in a total of 40 EFFs (10 times a_n, b_n, c_n and d_n).

2.4. Linear mixed models (LMM)

Features from RT-DC datasets, for example those defined in sections 2.2 and 2.3 could be used as label-free markers to track the behavior of cells for example upon treatment with a specific drug or to compare different cell types. In many applications, a treatment only causes a small shift of the population, leading to the question if the change is significant or just a random fluctuation. Let’s consider experimental data for area (as defined in section 2.2) of two samples of human skeletal stem cells (SSC) and three samples of the human osteosarcoma cell line MG-63 as shown by boxplots in Figure 2.7

[82,97]. The task is to compare these cell types, which is interesting because MG-63 is often used as a model system for SSCs.

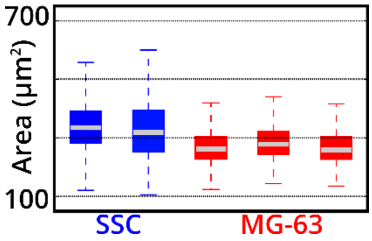


Figure 2.7 Area of SSCs and MG-63

Boxplot shows the cross-sectional area of a duplicate measurement of human skeletal stem cells (SSCs) and a triplicate measurement of the human osteosarcoma cell line MG-63, all measured using RT-DC in a 30 µm wide channel and a flowrate of 0.16 µl/s. Data was published in [82,97]. Boxes show the interquartile ranges (IQR), which are defined by the 25th percentile (Q_1) and the 75th percentile (Q_3): $IQR = Q_3 - Q_1$. White lines in the boxes show the medians. Whiskers represent the range of the data (lower bound: $Q_1 - 1.5 \cdot IQR$, upper bound: $Q_3 + 1.5 \cdot IQR$. This representation is commonly used for boxplots, and will be applied for each following boxplot in this thesis. Figure is adapted from [82].

To compare the two different cell types one could fit a linear model, which is defined by:

$$Y = X\beta + \varepsilon \tag{2.15}$$

Here, the output variable Y (measured values) is fitted assuming a linear relationship to the effect size β (also called fixed effect). The design matrix X encodes the categorical variable “cell type” which is either “SSC” or “MG-63” in this example. Additionally, there is the error term ε, which accounts for all possible fluctuations using a Gaussian distribution. Figure 2.7 indicates variation in the mean levels across the biological replicates, which is expected for biological samples. For example, the median (white line in the boxplots) of the first MG-63 dataset is lower than of the second. In a linear model, ε, would have to describe that.

Linear mixed models (LMM) take these fluctuations into account by adding an additional term to the model:

$$Y = X\beta + Zu + \varepsilon \tag{2.16}$$

which is called random effect term (Zu). In this term, systematic measurement errors (e.g. different cell confluency, slightly varying channel widths, viscosity, focus and illumination), which affect all replicates differently, can be explained. This essentially leads to a subdivision of the error into a systematic part which we can be explained (Zu)

and some random error, which cannot be further explained (ε). The design matrix Z describes, which data point corresponds to which replicate.

In the terminology of LMMs, the variation in mean values of a given parameter across replicates of a given sample is named a "random intercept". For example, there is a variation of the mean levels of SSCs (see Figure 2.7), and the random intercept term, which is the first part of the random effect term (Zu) accounts for that.

Furthermore, the difference between pairs of the same replicate number varies. For example, the difference between the mean of the 1st replicate of SSC and the 1st replicate of MG-63 is different than for the respective 2nd replicates. This second part of the random effect term is a so-called "random slope". For better explanation of the concept of random slope let me show a second example. Let us assume, a drug against hypertension is tested on three patients. For each patient, the blood pressure is measured before and after taking the drug and the reduction of the blood pressure is determined. If the decrease of the blood pressure is different for each patient, the random slope is high. If the decrease of the blood pressure is similar for each patient, the random slope is low.

The basic equation of an LMM showing all components allows a more comprehensive representation of an LMM with random intercept and random slope:

$$Y_{ij} = \beta_0 + x_j\beta_1 + u_{0i} + x_j u_{1i} + \varepsilon_{ij} \tag{2.17}$$

In the light of the above example (shown in Figure 2.7), Y_{ij} is the area of the i-th cell of cell type x_j (cell types are categorical variables: SSC or MG-63). β_0 is the fixed intercept and β_1 is the fixed slope. The random intercept and random slope are expressed by u_{0i} and u_{1i}, respectively, which are Gaussian distributions with a mean of zero and standard deviations, which express the variability of the means of one condition (e.g. MG-63) and the variability of the differences between two conditions (SSC and MG-63).

The R [98] package "lme4" [99] was used to define an LMM as shown in equation 2.17 and to fit it to experimental data. In order to design a statistical test (based on a likelihood ratio test), another LMM which is lacking the fixed effect term was fitted as well (a so-called "null-model"). After fitting each model, the maximized likelihood values L_{Model}

and $L_{NullModel}$ are returned, which are employed to compute the likelihood ratio $\Lambda = L_{Model}/L_{NullModel}$. This ratio expresses how much better a model fits the data if it assumes the presence of a fixed effect compared to a model without fixed effect. The corresponding p-value can be computed based on Wilks theorem, which states that $-2\log(\Lambda)$ approaches a χ^2 distribution for large sample sizes [100]. The degrees of freedom associated to the chi-squared distribution is the difference in degrees of freedom of the compared models, which equals to one in this case, since the models only differ by the fixed effect. The resulting p-value allows evaluating if the null hypothesis “model and null-model are identical” is true. If the p-value is below a given significance level (typically chosen to be 0.05), the null hypothesis can be rejected, which in this example would mean that the area of SSCs and MG-63 cells are considered to be significantly different.

The advantage of this statistical test is that it does not automatically return low p-values for very large sample sizes, since it takes the reproducibility of an effect across biological replicates into consideration. This is important for a meaningful statistical interpretation of data and discussed in further detail in section 3.2.2. The test was implemented as part of the work of this thesis. Moreover, the test was integrated into the analysis software ShapeOut [83] (courtesy of Paul Müller) and has already been used in several RT-DC related publications [13,15,22,23,85,97,101–106].

2.5. Normality test using probability plots

Several statistical tests such as the t-test are designed for data that follows a Gaussian distribution. To test data for normality, one of several normality tests can be employed. Here, a test based on probability plots is introduced. This test requires to first sort the measured n data points in ascending order and to compute the quantiles m according to Filliben’s estimate:

$$m(i) = 1 - 0.5^{1/n}, \text{for } i = 1 \tag{2.18}$$

$$m(i) = \frac{i - 0.3175}{n + 0.635}, \text{for } i = 2,3,\dots,n-1 \tag{2.19}$$

$$m(i) = 0.5^{1/n}, \text{for } i = n. \tag{2.20}$$

Next, the ordered data is plotted versus the quantile values, resulting in the so-called probability plot, which shows a linear relation if the data was normally distributed. This linearity can be quantified using the square of the Pearson coefficient of correlation R^2:

$$R^2 = \left(\frac{1}{n\sigma_x\sigma_y}\sum\left[(x_i - \mu_x)(y_i - \mu_y)\right]\right)^2 \qquad 2.21$$

with the mean of the values μ_x, the mean of the quantiles μ_y and the respective standard deviations σ_x and σ_y.

2.6. Gaussian mixture model (GMM) and Bayesian information criterion (BIC)

Gaussian mixture models (GMMs) belong to the group of unsupervised machine learning algorithms. A GMM is applied with the aim to model an $n \times k$-dimensional dataset X (n-number of events, k-number of parameters) using a superposition of K Gaussian distributions $\mathcal{N}(\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$ (with the k-dimesnional mean vector $\boldsymbol{\mu_i}$ and a $k \times k$-dimesnional covariance matrix $\boldsymbol{\Sigma_i}$) [107]:

$$X \sim \sum_{i=1}^{K} w_i \mathcal{N}(X|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}). \qquad 2.22$$

The contribution of each Gaussian i is modulated by a scalar weight w_i. The probability density function of a multivariate Gaussian is defined by:

$$\mathcal{N}(X|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) = (2\pi)^{-\frac{k}{2}}|\boldsymbol{\Sigma_\mathrm{i}}|^{-\frac{1}{2}}\, exp\left(-\frac{1}{2}(X - \boldsymbol{\mu_i})^T\, \boldsymbol{\Sigma_\mathrm{i}}\, (X - \boldsymbol{\mu_i})\right). \qquad 2.23$$

The expectation-maximation algorithm is an iterative optimization process and can be used to fit the GMM to a dataset and find the parameters w_i, $\boldsymbol{\mu_i}$ and $\boldsymbol{\Sigma_i}$ [108]. In this thesis the Python implementation for GMM from the sklearn package [109] was used. The resulting probability distributions can for example be used to define borders of subpopulations (clustering) or to quantify subpopulations in a dataset using their mean and covariance.

If the number of subpopulations K is not known, several models with different K have to be fitted and compared. The performance of each model can be quantified using the Bayesian information criterion (BIC), which should be minimized:

$$BIC = -2\ln(L) + t\ ln(n). \tag{2.24}$$

The first term in equation 2.24 is dependent on L, the maximized value of the likelihood function p

$$ln\ p(X|\boldsymbol{\mu}, \boldsymbol{\Sigma}, w) = \sum_{j=1}^{n} ln \left\{ \sum_{i=1}^{K} w_i \mathcal{N}\left(X_j | \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\right) \right\}. \tag{2.25}$$

Typically, increasing the number of Gaussians K in a GMM will increase L and the highest L corresponds to a GMM which has as many Gaussians as data points ($K = n$). To avoid such overfitting, the second term in equation 2.24 introduces a penalization of the number of parameters t, which implies constraining K in a GMM.

2.7. Random forests

Unlike GMM, random forest-based classification is a supervised machine learning technique which means the model is trained on labeled datasets. A random forest consists of several decision trees, which aim to predict the label, based on a chain of thresholds in multiple dimensions. Figure 2.8 exemplarily shows the sketch of a decision tree for a two-dimensional input. Training a decision tree implies finding optimal thresholds, which allow to correctly predict the majority of events in the training set. Since single decision trees tend to overfit quickly, several decision trees are trained using random subsets of the training data, hence the name random forest. For prediction, the input data is passed through each decision tree of the random forest, each returning a prediction. Class probabilities are computed based on the proportion of votes of the trees and classification is performed by taking the majority vote. Decision trees work by splitting up the data at a node using a certain feature and a threshold. A perfect feature would allow splitting up the entire dataset correctly into the existing classes. Less performant features only allow guiding a small portion of the data into the correct classes. Before a split, the data is usually mixed (impure) and after the split as pure as possible. Impurity is quantified using the Gini impurity index:

$$G = 1 - \sum_{i=1}^{n_c} p_i^2 \tag{2.26}$$

with the number of classes n_c and the fraction of events that belong to class i p_i ($p_i =$ number of events of class i divided by the total number of events). If a feature separates the data well, it decreases the Gini impurity index:

$$\Delta G = G_{parent} - \left(G_{Split1} + G_{Split2}\right). \tag{2.27}$$

Therefore, the G can be used to quantify the importance of a feature for the overall classification performance.

The bootstrap aggregation of decision trees used to create a random forest was shown to improve classification accuracy and robustness of the models. This is true not only for decision trees, but in general for any predictive model [110]. In this thesis the Python implementation for random forest classifiers from the sklearn package [109] was used, which employs the Gini impurity index to quantify feature importance.

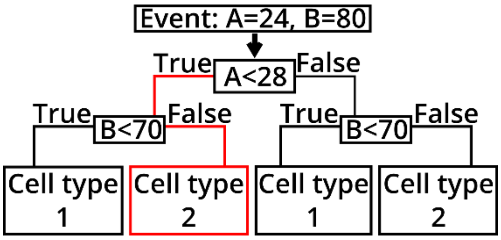


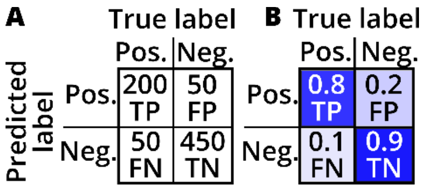
Figure 2.8 Example of a decision tree

Sketch of a decision tree, which takes two inputs to decide between two classes. Red lines indicate the decision process for the given example event.

2.8. Confusion matrix

Quantification of the performance of machine learning models is an essential part before choosing a final model. The method for performance quantification should be chosen depending on the application of the model. For example, a HIV-test should be tuned to result in more false positives (FP) instead of false negatives since the consequence of a wrongly detected HIV case (FP) only results in additional tests and maybe unnecessary treatment, but the consequence of a false negative would mean that a sick patient would not be treated.

Commonly, machine learning models are used to predict data into classes or give probabilities for certain classes. If probabilities are given, the prediction is performed using the maximum probability. For example, in a binary classification task, an event would be predicted to belong to class 1 of the probability for class 1 is larger than 50%. Such a prediction could either be correct (true positive - TP) or wrong (false positive - FP). A confusion matrix (CM) summarizes the prediction of several events and allows reading the number of true and false positives and negatives, respectively. Figure 2.9 A exemplarily shows the CM for a binary classification of 250 positive and 500 negative examples. Common quantities that are derived from the confusion matrix are $\text{accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$, $\text{sensitivity} = \frac{TP}{TP+FN}$, $\text{specificity} = \frac{TN}{TN+FP}$ and $\text{precision} = \frac{TP}{TP+FP}$. In the example in Figure 2.9 A, these metrics would be: $\text{accuracy} = 0.87$, $\text{sensitivity} = 0.8$, $\text{specificity} = 0.9$, and $\text{precision} = 0.8$. The normalized confusion matrix is obtained by dividing each matrix element by the total number of events of the respective class (see Figure 2.9 B), which especially helps to visualize the performance of the model, if there are different numbers of events in each class.



A

Predicted label	True label Pos.	True label Neg.
Pos.	200 TP	50 FP
Neg.	50 FN	450 TN

B

Predicted label	True label Pos.	True label Neg.
Pos.	0.8 TP	0.2 FP
Neg.	0.1 FN	0.9 TN

Figure 2.9 Confusion matrix and normalized confusion matrix

(A) Sketch shows an exemplary CM for a binary classification into two classes: positive (Pos.) and negative (Neg.). The upper and lower row of the CM show the prediction of 250 positive events and 500 negative events, respectively. Common quantities that are derived from the confusion matrix are accuracy, sensitivity, specificity and precision.

(B) Sketch shows the normalized CM, corresponding to the CM shown in (A). This representation is helpful, especially when the numbers of events for each class are not equal. Conditional coloring is added to visualize large and small values (white – small, dark blue – large)

2.9. Deep learning

Average brightness, Haralick texture features, LBP and TAS (see 2.2 and 2.3) are only a selection of image features and in principle infinitely many features could be computed. Some features might be better suited than others for a given classification task and the discovery or design of suitable features is a challenging task (often called ‘feature engineering’). Deep learning is a category of machine learning methods which automatize feature learning using artificial neural networks. Since these methods allow computers to learn suitable actions to perform a given task, the term “artificial intelligence” is also frequently used [111]. Artificial neural networks are loosely inspired by the functionality of brain-neurons and were first suggested in 1943 [112], but only became popular in 2012 after a neural network substantially outperformed all other machine learning techniques in the large scale image classification challenge [113]. Beside computer vision, which is most relevant for this thesis, deep learning is applicable in many fields for example natural language processing, stock price prediction or speech recognition.

A general characteristic of neural networks is that they process the input information in subsequent (sometimes interconnected) layers. The first layer of a network contains the input information (input layer). Next, the information is passed to the next layers which are called hidden layers. Hidden layers perform particular operations on the input data. The last layer is called output layer (see Figure 2.10 A). A selection of network types and the corresponding layers operations are introduced in the following.

Multilayer perceptron

Multilayer perceptrons (MLPs) are a kind of neural network which consists of a number of stacked layers with nodes (see Figure 2.10 A). The nodes of the first layer contain the input values x, which could be the single pixels of an image. For a 32x32 pixel image this would result in 1024 input nodes. Each node is then connected to all the nodes in the next layer, which is called a hidden layer h. In the example of 1024 input values, each node in the first hidden layer is defined by 1024 weights one bias and an activation function (see Table 1 and Figure 2.10 B). Let the matrix $W^{(1)}$ represent the weights of all nodes, $b^{(1)}$ all biases and s the activation function of the first hidden layer. Then, the operation, performed by the first hidden layer can be expressed in matrix notation:

$$h(x) = s\left(b^{(1)} + W^{(1)}x\right). \tag{2.28}$$

The weights allow each node of h to give more or less "attention" to specific input values (pixels). In each node of h, the weighted input and the bias are added together and the resulting sum is passed through an activation function, returning the final score of this node. The ensemble of nodes in the first hidden layer contains combinations of contributions of the pixels, which can be of a nonlinear fashion because of the activation function. In the simplest possible form, the MLP has only one hidden layer. Then, $h(x)$ is connected to the nodes in the output layer by another weight matrix $W^{(2)}$, bias $b^{(2)}$ and activation function G. The full MLP can be written in matrix notation:

$$f(x) = G\left(b^{(2)} + W^{(2)}\left(s\left(b^{(1)} + W^{(1)}x\right)\right)\right). \tag{2.29}$$

In an MLP, each node of a layer is connected to all nodes of the preceding layer. This kind of fully connecting layer is also called "dense layer". Nowadays, neural nets typically contain a cascade of multiple hidden layers, which allows an abstract representation of the input data. While neural nets are called "deep" when they have many layers, the "width" of a network refers to the number of nodes in each layer. If the architecture has no hidden layer, the input nodes are directly linked to the output nodes through weights. Such a model is just called perceptron [114]. When the final output of a perceptron is modulated by a sigmoid activation function (see Table 1 and Figure 2.10 B), the model is identical to a logistic regression model.

Without such modulation, even an MLP with hundreds of hidden layer would only be able to fit linear functions. To allow deep neural nets to fit non-linear functions, so-called activation functions are used, which are typically applied to modulate the output of each layer. Commonly used activation functions are the $sigmoid$, $tanh$, rectified linear unit ($ReLU$) and the $softmax$ function (see Table 1 and Figure 2.10 B). $Sigmoid$ and $tanh$ work like switches between two states (0,1 for $sigmoid$ and -1,1 for $tanh$) and a generalization to k states is given by the $softmax$ function:

Name	Function	Eq. nr.
$sigmoid$	$y = 1/(1 + e^{-x})$	2.30
$tanh$	$y = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	2.31
$ReLU$	$y = max(0, x)$	2.32
$softmax$	$y = \frac{\exp(x_i)}{\sum_{j=0}^{k} \exp(x_i)} \; for\ i = 0,1, \ldots k$	2.33

Table 1 Activation functions

Initially, the model parameters are set to random values and the error of the model is high. The error is quantified by a so-called “cost” or “loss” function such as the mean squared error: $MSE = \frac{1}{n}\sum_{i=0}^{n}\left(\widehat{Y_i} - Y_i\right)^2$, where $\widehat{Y_i}$ are the predictions and Y_i the desired values. Another loss function is the cross-entropy (also called “log-loss”): $crossentropy = -\sum_{c=1}^{M} \hat{Y}_{o,c} \log(p_{o,c})$, where M is the number of classes, $\hat{Y}_{o,c}$ is a binary indicator showing if the class label c is the correct classification for the observation o and $p_{o,c}$ is the corresponding probability, predicted by the model.

By computing the gradient of the loss with respect to each parameter of the model, it is possible to calculate how the parameters need to be changed in order to decrease the loss. This routine allows to iteratively optimize the model parameters and is called gradient descent. For gradient descent, the entire training dataset is used to compute the gradient for one parameter update, which is not feasible for large datasets. A solution is to use random batches of the training dataset to perform parameter updates. This method is called stochastic gradient descent (SGD).

Dropout layer

Due to the high complexity and large number of parameters of deep neural networks, overfitting becomes an issue. Overfitting means that models learn to perfectly predict the training data but fail to generalize to new data. Dropout is a regularization method which can be applied to prevent overfitting. In a dropout layer, a defined number of nodes of a layer is randomly switched off during training (see Figure 2.10 C), which essentially creates a different, less complex model in each training iteration [115]. Currently, dropout is perhaps the most used regularization technique in deep learning.

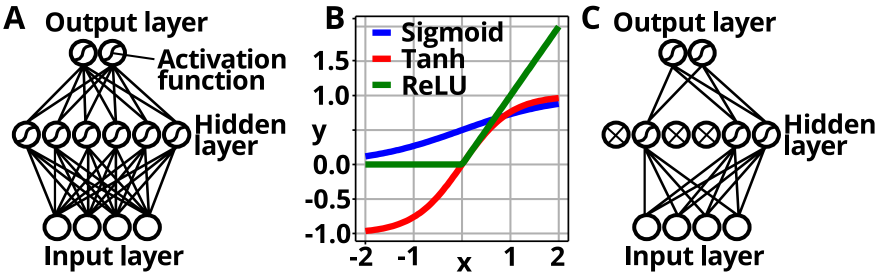


Figure 2.10 Multilayer perceptron, activation functions and dropout

(A) Sketch of a multilayer perceptron with one hidden layer. Values of the input layer are altered by a set of weights (indicated by lines) and biases. In the nodes (represented by circles), the altered values of all nodes of the preceding layer in the network are added up and modulated by an activation function.

(B) Examples of popular activation functions.

(C) Sketch showing an MLP, where the hidden layer is regularized by a dropout of 50% ($p = 0.5$).

Batch normalization layer

The batch normalization layer performs a normalization of the activations of the preceding hidden layer, which avoids that a few activations are much larger than others [116]. A domination of single activations would strongly determine the state of succeeding layers and cause instability of the network. Therefore, batch normalization helps to speed up the training process and also tends to regularize the network (prevents overfitting).

Convolutional layer

One of the most important developments of deep learning during the last few years is the convolutional layer, which gives the name for a class of neural nets called convolutional neural nets (CNNs). While in an MLP, every node of one layer is connected to every node in the next layer (“dense layer”), in a convolutional layer, so-called convolutional filters are used, which restrict the connection of nodes to a defined neighborhood [117,118]. This restriction is very helpful to learn representations of structured data such as images, sound and text. The principle of a convolutional layer is that convolutional filters scan over the input image and a dot product is computed at every position using the image patch at this position and the weights of the filter as exemplarily shown in Figure 2.11. During SGD, these weights (values in the convolutional filters) are learned. For images, the filters in the first convolutional layer usually learn to detect edges [119] as shown in Figure 2.11 A and B. By performing another convolution on the resulting feature maps, more complex features (see Figure 2.11 B) can be described and modern CNNs often consist of several or even thousands of succeeding convolutional layers. The first popular CNN contained three convolutional layers and two dense layers and attracted attention because of its performance in character recognition (“LeNet-5”) [117]. In recent years, the most widely used network architectures were often introduced during the annual “Large Scale Visual Recognition Competition” (ILSVRC). In 2015 a novel network architecture which introduced residual blocks (hence named “ResNet”) won this competition. In residual blocks, feature maps resulting from a convolutional layer are merged with data from an earlier layer (see Figure 2.11 C).

Maxpooling layer

Maxpooling is a subsampling operation in which a window of defined size is moved over the input image and at each position, only the maximum value within the window is kept. Figure 2.11 A shows maxpooling using a window size of 2x2 which reduces the number of pixels by a factor of four. While subsampling reduces the resolution, it is still heavily used in practice since it improves translation invariance and reduces the computational cost of the model.

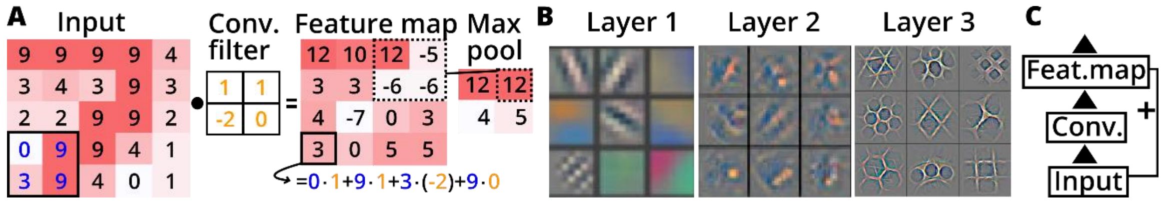


Figure 2.11 Convolution, maxpooling, feature maps, and residual block

(A) Exemplary calculation of a convolution using a 5x5 pixel input image and a 2x2 convolutional filter. The input image shows the number seven and the example convolutional filter changes the input image such that horizontal lines are highlighted in the feature map. This feature is still highlighted after a subsequent maxpooling operation. An exemplary computation of a convolution for a region in the input image (indicated by black rectangle and blue numbers) with the convolutional filter (yellow numbers) is shown.

(B) Visualization of feature maps of a trained CNN with multiple convolutional layers. The first convolutional layer captures simple image characteristics such as edges. Filters of subsequent layers capture increasingly complex features. Figure is adapted from [119].

(C) Principle of a residual block. The input data or data from an earlier layer is added to the result of a convolutional layer.

Generative adversarial networks (GANs): Cycle-GAN

Well trained deep neural networks should have learned representations of the data, allowing the network to correctly predict new data, which might deviate from the training data. In other words, the network should learn the basic nature of the input data. From here, one could imagine that these learned latent representations could be used to generate images, by basically inverting the neural net, such that data is inserted in the former output layer and the result is shown in the former input layer. The problem is that this is a one-to-many-mapping since different inputs to the original neural net could produce the same output. Hence, there is no close-formed expression for inverting a neural net.

GANs approach the issue of data generation by training two models. On the one hand, a model G is trained to generate realistic looking images ("generator") and on the other hand, a second model D is trained to distinguish real and generated images ("discriminator") (see Figure 2.12 A). In each training iteration both models are optimized individually to get better at generating realistic images and detecting generated images, respectively. The objective is to find an equilibrium (Nash equilibrium). GANs were introduced in 2014, showing for example the generation of images of human faces [120], using multilayer perceptrons for D and G, but the image quality was relatively low and the appearance partially unrealistic. The limited quality of the results arose from the problem of minimizing the loss of two models simultaneously, which can for example result in generators that do not learn to generate a distribution of images but only a given number of different images (mode collapse). Similarly, a generator can learn to generate images that are identical to images of the training set. Major improvements were shown by the Wasserstein GAN [121,122]. The basic principle of GANs is shown in Figure 2.12 A.

A related task in this domain is image to image translation for example to change the style of an image. Datasets that provide pairs of images for both domains would allow to learn the mapping between the domains (for example photograph to painting or sharp image to image with Bokeh-effect), but such datasets of paired images are rare and often difficult or impossible to generate. This issue is addressed by Cycle GAN which uses two generators, one generator G which translates images from domain X to domain Y and a generator F for mapping from Y back to X [123] (see Figure 2.12 B). The results of generator G and F are fed into discriminator D_Y and D_X, respectively. The discriminator D_Y promotes G to return images that look like images in domain Y and the other way around for D_X and F. Additionally, a cycle consistency loss is introduced to ensure that mapping an image I from X to Y using G and then mapping this image back using F, results in an image that is very similar to the initial image: $I \approx F\big(G(I)\big)$ and vice versa $I \approx G\big(F(I)\big)$. In the original paper the generators are neural nets with several convolutional layers and residual blocks. The discriminators are neural nets with four convolutional layers. This model architecture allowed the authors to create stunning results for a style transfer in images for example a translation of photographs to paintings with styles of different popular artists such as Monet and van Gogh.

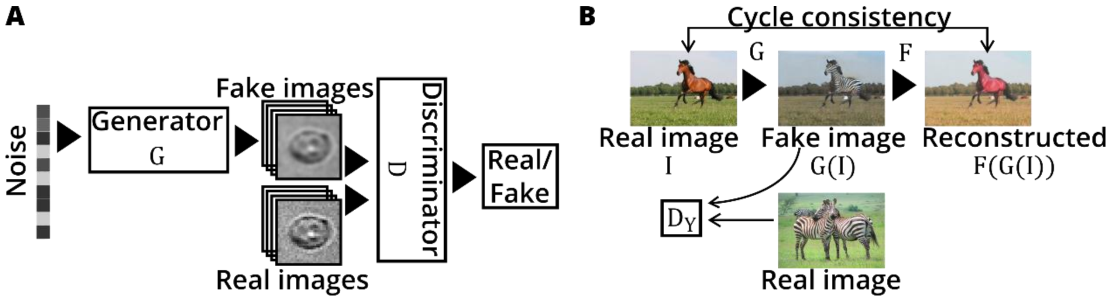


Figure 2.12 Generative adversarial networks (GANs) and principle of Cycle-GAN

(A) Sketch of the basic principle of generative adversarial networks. Usually, random noise is fed into a generator, which is a neural net, translating the input to images that look similar to real images. Another neural net takes generated (fake) images and real images and learns to distinguish them. Both neural nets are trained in adversarial fashion such that the performance of both increases from training iteration to training iteration.

(B) Sketch of the principle of Cycle-GAN. A generator G translates images I of domain X (e.g. horses) to domain Y (e.g. zebras). A discriminator D_Y learns to discriminate real and generated images in domain Y, promoting G to improve in performance, i.e. to get better at creating images that look like real images of Y. Additionally, generated images $G(I)$ are translated back to domain X by generator F, which allows to formulate a cycle consistency loss supporting that $I \approx F\big(G(I)\big)$. Equivalently, F is trained adversarial to a discriminator D_X using the respective cycle consistency $I \approx G\big(F(I)\big)$ (not shown in sketch). Images of horse, zebra, and generated zebra were taken from [123].

For better illustration, the principle will be explained using the example, where domain X are images of cells that were captured in sorting chip A, which results in a specific appearance of cells because of unique image distortion (see section 3.3.5, Figure 3.17) by the lithium niobate substrate. Equivalently, domain Y is defined by images of cells in sorting chip B with different image distortion. Since it is not possible to measure the same cell in both chips, we are dealing with an unpaired dataset. Hence, the Cycle-GAN algorithm can be used to fit a generator G, which transforms images from the phenotype of sorting chip A to the phenotype of sorting chip B. Similarly, the opposite direction can be performed using the generator F. Such image augmentations could allow to artificially increase the size of the dataset, which is especially of interest when training neural nets.

2.10. Preparation of retina samples

Rod photoreceptor cells were isolated from pups of the mouse line Nrl-GFP at the following maturation stages: embryonic day (E) 15.5 and postnatal days (P) 4, 10 and 20.

To retrieve retina cells at embryonic stage E15.5, the females were continuously checked for a vaginal plug. When a vaginal plug was found, the respective female was transferred to an individual cage and that day was counted as embryonic day (E) 0.5. At E15.5, the females were euthanized using cervical dislocation. Their abdomen was cleaned with 70% ethanol (vol/vol) and the embryos were taken out using scissors. After this step, the preparation is identical for E15.5, P04, P10 and P20. Following dissection of the eyes, the retinas were isolated, transferred to a Papain solution (Worthington Biochemical Corporation, USA) and incubated for 35 minutes at 37 ° C to dissociate the retinas into single cells as described previously [2]. After dissociation, cells were centrifuged for 5 minutes at 300 g and re-suspended in in a particular buffer suitable for the prospective measurement technique. For FAC-sorting, cells were re-suspended in FACS buffer (2mM EDTA and 1% fetal calf serum (FCS) in PBS) and passed through a 40 µm Nylon cell strainer (BD Biosciences, USA). For RT-DC and RT-FDC measurements the cells were re-suspended in a measurement buffer based on PBS complemented with 0.5% methyl cellulose after filtering using a 40 µm Nylon cell strainer. For SAW-sorting experiments, the cells were re-suspended in a measurement buffer based on PBS, complemented with 10% Leibovitz medium (Thermo Fischer, Germany), 0.6% methyl cellulose and 0.01 mg/ml DNase (Invitrogen, USA). (Without DNAse, cell-clumps appear after approximately one hour. Such cell clumps would congest the filters in the sorting chip or could accidentally flow into the target outlet during a sorting experiment.)

2.11. Preparation of blood samples

10 ml venous whole blood was drawn from healthy donors after informed consent (ethical approval EK89032013, granted by the ethics committee of the Technische Universität Dresden) into sodium-citrate tubes (S-Monovette® 10ml 9NC, Sarstedt, Germany). After adding 2.5ml of dextran solution (6% dextran (Dextran T500, Pharmacosmos A/S, Denmark) solved in Sodium chloride (0.9% Sodium Chloride Irrigation, Baxter Healthcare, Switzerland)), the sample was gently mixed and incubated at room temperature for 30 min allowing red blood cells (RBCs) to sediment. The

supernatant was moved to a new falcon tube and centrifuged at 900 rpm (≙120g) for 10 min (Universal 30RF, Hettich, Switzerland). After removal of supernatant, the pellet was resuspended in measurement buffer (MB) for RT-DC.

2.12. Staining of neutrophils and monocytes

Approximately 50,000 cells, (suspended in MB) were transferred into a 2 ml Eppendorf-tube and centrifuged at 3500 rpm (≙822g) for 5min (Mini Spin, Eppendorf, Germany). The pellet was resuspended in 40 µl of supernatant. Next, 1 µl of CD66 conjugated to PE (PE anti-human CD66a/c/e, Biolegend, USA) and 2 µl of CD14 conjugated to APC (Anti-Human CD14, Invitrogen, USA) were added. Sufficient distribution of staining molecules despite the viscous buffer was achieved by pipetting the sample up and down 10 times. The sample was incubated for 25 min at 25 °C in a block thermomixer (Thermomixer, Eppendorf, Germany) using 600 rpm.

3. Results

In RT-DC, bright-field images of cells are captured by a high-speed camera and parameters such as deformation and cell size are determined from the tracked contour. This chapter first assesses universal properties of such parameters. Next, the measured parameters of samples of retina cells at different maturation stages are compared. Knowledge about universal properties of parameters is applied to choose certain modeling approaches and statistical tests.

In section 3.3, multiple supervised machine learning techniques are employed to develop methods for the identification of photoreceptors in mixed retina samples. First, a classification algorithm is optimized to provide the highest classification accuracy possible. Next, an algorithm is optimized to deliver good classification accuracy at minimal computational time such that the algorithm could be applied for real-time analysis and sorting. The different algorithms are then compared and software is introduced, which eases the access to those methods and allows to quickly apply them for new datasets. Finally, in section 3.5, the developed methods and software are leveraged to perform sorting of rod photoreceptors from retina and of neutrophils from human blood.

3.1. Meta-analysis of RT-DC data

A multitude of label-free features is available in RT-DC. Some features are determined in real-time but other features which require more computational power can be calculated after the measurement. Ideally, all parameters would be orthogonal, meaning; they describe non-correlated information to avoid redundancy. In the set of features described in 2.2 and 2.3, this is clearly not the case as for example area and volume both express the concept “cell size”. When analyzing a single measurement, appearing correlations could originate from an artifact or a unique biological property. To discover the general nature of features as well as general trends and effects, it requires a large number of measurements of different cells-types. In order to do that, I ran a meta-analysis using the existing data from data from 21,000 experiments which were acquired by multiple users (members of Guck-lab and collaboration partners). The data was captured over a time range of approximately two years, multiple flowrates were used and many different cell types were measured. Table 2 shows an overview of the number of experiments and the number of measured events that were captured in 10, 15, 20, 30, 40 µm channels or in the reservoir. Especially when analyzing deformation, it is important to remove doublets, badly tracked cells, and cells that have a very irregular shape. For this purpose, typically porosity is used and all cells with $\Omega > 1.05$, or $\Omega > 1.15$ are removed. Table 2 shows the number of events remaining after applying each filter. For each case, there are thousands of experiments and several millions of events available.

Region	Nr. of experiments	Nr. events (total) in mio.	Nr. events ($\Omega \leq 1.05$) in mio.	Nr. events ($\Omega \leq 1.15$) in mio.
20 µm channel	10066	48.2	25.7	45.7
30 µm channel	6541	23.7	13.7	22.3
10, 15 and 40 µm channel	1322	10.7	5.4	10.2
Reservoir	3071	11.1	6.9	10.4

Table 2 Data leveraged for meta-analysis

3.1.1. Correlations of area and volume

Area, volume and deformation are features that originate from calculations based on the tracked contour. The dependency of deformation on cell size has been described by analytical and simulation approaches [75,77,84]. In brief, larger cells deform more in the channel as they are exposed to larger shear and normal forces [75,77,84]. To visualize this effect, measurements of human neutrophils are displayed in Figure 3.1 A using scatterplot-contours, which show a slight “tilt”, and align well with the iso-elasticity lines [77] (regions of equal elastic modulus according to the analytical model, depicted as gray lines in Figure 3.1 A). Scatterplot-contours are obtained by computing the event-density in a scatterplot and highlighting regions that refer to 95% (solid line) and 50% (dashed line) of the maximum density. When comparing the scatterplot-contours from measurements of different flowrates, there is a shift towards larger area for higher flowrates. This effect could occur because more deformed objects cover a larger area when being projected into two dimensions, as illustrated in Figure 3.1 B. When approximating the shape of a deformed cell as an ellipsoid, the corresponding volume is:

$$V = \frac{4}{3}\pi abc. \tag{3.1}$$

This equation can be rearranged to

$$\frac{3V}{4c} = \pi ab = A. \tag{3.2}$$

Equation 3.2 shows that the projected area A depends on the height (c) of the ellipsoid, allowing to construct ellipsoids of identical volume, but different projected A. As the height of cells in RT-DC is affected due to the channel height and hydrodynamic forces, A might not be the optimal measure for cell size. When using volume (V) instead of area, the shift of the scatterplot-contours (for different flowrates) is reduced as shown in Figure 3.1 C. Also, the scatterplot-contours appear to have less "tilt", indicating lower correlation of volume and deformation. To quantify correlation, the Pearson correlation coefficient R (see Materials and Methods 2.5) for area vs. deformation (A vs. D) and volume vs. deformation (V vs. D) was computed for all experiments that were performed in the channel region (17929 experiments) and displayed in a boxplot (see Figure 3.1 D) (only events with $\Omega \leq 1.05$ are used). The boxplots in Figure 3.1 D show an overall lower correlation for V vs. D, as compared to A vs. D, indicating that volume is less affected by deformation.

3.1.2. Correlations of deformation and inertia ratio

A measure that is frequently used in context of RT-DC is deformation (D, see section 2.2). D is calculated using the perimeter and area of the convex hull of a tracked object. The convex hull is chosen since protrusions or dents in the contour would cause a large increase of D. To exclude objects from the analysis, which are badly tracked or have a very irregular shape, typically, the porosity (Ω) is used and objects with $\Omega > 1.05$ are omitted from the measurement. After cleaning the data using this filter, there is still a positive correlation R between deformation and porosity ($R(D\ vs. \Omega)$) for many measurements, as shown in Figure 3.1 E. The median of $R(D\ vs. \Omega)$ is approximately

0.28. This indicates that D tends to be larger for contours with protrusions or dents, despite using the convex hull for computing D.

An alternative parameter for D is inertia ratio (I), which is computed using the raw contour. With a median correlation of 0.18, the correlation between inertia ratio and porosity $R(I\ vs.\Omega)$ is lower as compared to $R(D\ vs.\Omega)$ (see Figure 3.1 E). Hence, inertia ratio is more robust to quantify contours with irregular shapes.

Similar to volume, I is not invariant to rotation, which requires the user to manually align the channel axis to $\overrightarrow{e_x}$ before an experiment, to obtain reproducible results. Alternatively, the principal inertia ratio (I_p) could be computed, which is invariant to rotation, but requires more computational power. As indicated by the rightmost box in Figure 3.1 E, $R(I_p\ vs.\Omega)$ is very similar to $R(I\ vs.\Omega)$ (median of $R\left(I_p\ vs.\Omega\right) = 0.19$).

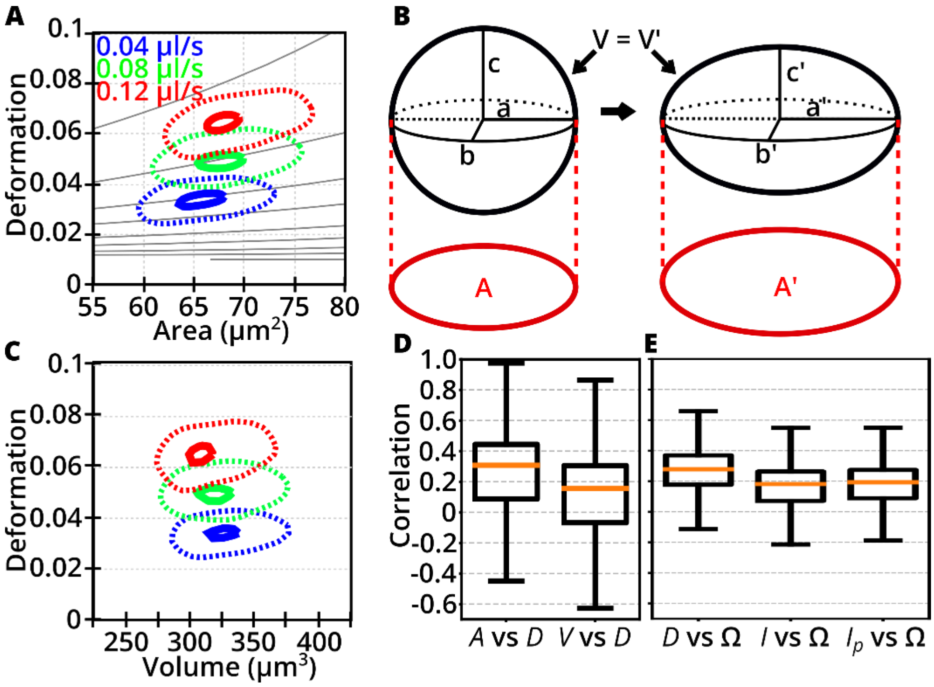


Figure 3.1 Correlations of area vs. volume and deformation vs. inertia ratio

(A) Plot shows RT-DC measurements of purified human neutrophils at different flowrates in a 20 µm channel. For better comparison between measurements, the density of scatter-dots was computed and scatterplot-contours show regions of 95% (solid line) and 50% (dashed line) of the maximum density.

(B) Schematic explanation how deformation affects area. Both ellipsoids (black) have the same volume, but the projected area (red) is different.

(C) Same measurements as in (A), but represented using D and V.

(D) Boxplots summarize the Pearson correlation coefficients of $A\ vs. D$ and $V\ vs. D$, obtained from 17,929 experiments (only experiments where cells are captured in the channel and events with Ω≤1.05 were used). Boxplot shows median, interquartile range and range of data, as introduced in Figure 2.7.

(E) Boxplots summarize the Pearson correlation coefficients of $D\ vs. \Omega$, $I\ vs. \Omega$, and $I_p\ vs. \Omega$ obtained from 17,929 experiments. There is trend, that inertia ratio is less affected by Ω, as compared to deformation. Boxplot shows median, interquartile range and range of data, as introduced in Figure 2.7.

3.1.3. Further screening of correlations

The previous section presented correlations of a couple of chosen parameters which are of special interest since they are employed in almost every publication that uses RT-DC. To pursue a more unbiased search for correlated features, the correlation matrix of the following features is determined: A, V, L_x, L_y, γ, D, E_{ana}, E_{num}, I, I_p, φ, Ω, c_x, c_y, B, B_{max}, B_{Std} (for abbreviations and explanation how features are computed see sections 2.2 and 2.3). For a set of two given features, the correlation coefficient was determined for all measurements that were performed in the channel region.

Subsequently, the median of the resulting values was determined. This procedure was repeated for each possible pairing of features, resulting in the correlation matrix, displayed in Figure 3.2 A. The shown correlation matrix reproduces aforementioned (see section 3.1.1 and 3.1.2) insights: $R(A\ vs. D) > R(V\ vs. D)$ (indicated by magenta rectangle) and $R(D\ vs. \Omega) > R(I\ vs. \Omega)$ (indicated by blue rectangle). Furthermore, a green rectangle in Figure 3.2 A highlights an astonishingly high correlation of all features describing cell size (A, V, L_x, L_y) with features related to transparency (B, B_{max}). For each matrix-element within the green rectangle, the correlation is positive, meaning large cells tend to be more transparent. To obtain a better understanding of this phenomenon, the underlying values of $R(A\ vs. B)$ from all measurements that were performed in the channel region and that have more than 100 events (in total 16,330 individual experiments) are plotted as a histogram in Figure 3.2 B (green). For highly negative ($R \approx -1$) and positive ($R \approx 1$) correlations, I only found measurements of beads and droplets, respectively (see I and II in Figure 3.2 B). To focus on data of biological cells, all experiments for beads and (silicone) droplets were removed and the remaining values (from 15,116 experiments) are visualized as a histogram in Figure 3.2 B (cyan). This histogram still shows a positive correlation for area and brightness for the majority of experiments, indicating that this could be a general property of cells.

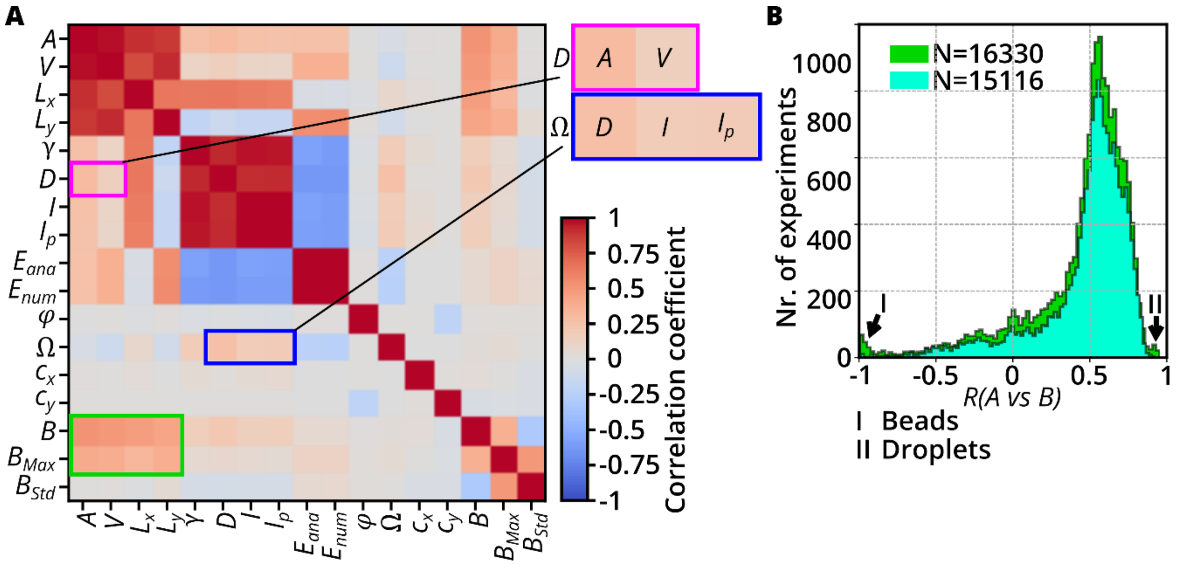


Figure 3.2 Correlation Matrix

(A) Confusion matrix shows the medians of the correlation coefficients from 16,330 experiments. The magenta rectangle indicates an elevated correlation of $A\ vs.\ D$ and $V\ vs.\ D$. The blue rectangle highlights a slightly elevated correlation of $D\ vs.\ \Omega$, $I\ vs.\ \Omega$, and $I_p\ vs.\ \Omega$ (discussed in 3.1.2). The green rectangle highlights an elevated correlation of parameters describing cell size with parameters describing transparency (brightness). The correlation matrix shows Pearson correlation coefficients which range from -1 (perfect negative linear correlation) to 1 (perfect positive linear correlation).

(B) Histogram shows the Pearson correlation coefficient for area vs. brightness of all 16,330 experiments (green). I and II indicate small peaks, which are caused by measurements of oil droplets and beads, respectively. When removing all experiments which contain the keywords “oil”, “drop”, and “bead”, 15,116 experiments remain, and the corresponding Pearson correlations are plotted as histogram (cyan).

3.1.4. Shape of distributions

Measurable properties of cells typically underlie an intrinsic variability, which originates for example from genetic or environmental differences. If such influences change the measured quantity by additive or multiplicative factors, the resulting spread will be better approximated by a Gaussian or lognormal distribution, respectively [124,125]. Furthermore, when a parameter has by definition a lower bound and most values gather close to the lower bound, the resulting distribution can also be modeled well by a lognormal distribution [125].

To determine whether either a normal or lognormal distribution is the underlying distribution of a parameter, so-called probability plots (see section 2.5) can be leveraged. Such plots are shown exemplarily in Figure 3.3 using an RT-DC measurement of dissociated Nrl-GFP retina cells that were FACS sorted for the GFP$^+$ fraction (rod photoreceptor precursor cells). In the upper left and lower left plot, original and log-transformed data of area (A) is shown. If the distribution was Gaussian, a linear function would be a perfect fit (red line in each plot). To quantify how well data aligns to the linear fit, the square of the Pearson correlation coefficient is given for original (R^2) and log-transformed data ($R^{2\prime}$). At low and high area values, there is a deviation between data and fitting function for the original data as well as for the log-transformed data, resulting in an R^2 and $R^{2\prime}$ of approximately 0.97 (see Figure 3.3). Such deviation (between data and lin. fit) is also present for the original deformation data (upper right plot in Figure 3.3, $R^2 = 0.957$), but not for the log-transformed data (lower right plot in Figure 3.3, $R^{2'} = 0.999$), indicating that the underlying distribution of deformation could be a lognormal function. The explanatory power of this finding is quite small as the analysis is based only on a single measurement, which could be erroneous. Artifacts such as technical error or presence of subpopulations could potentially skew the distribution, resembling a lognormal.

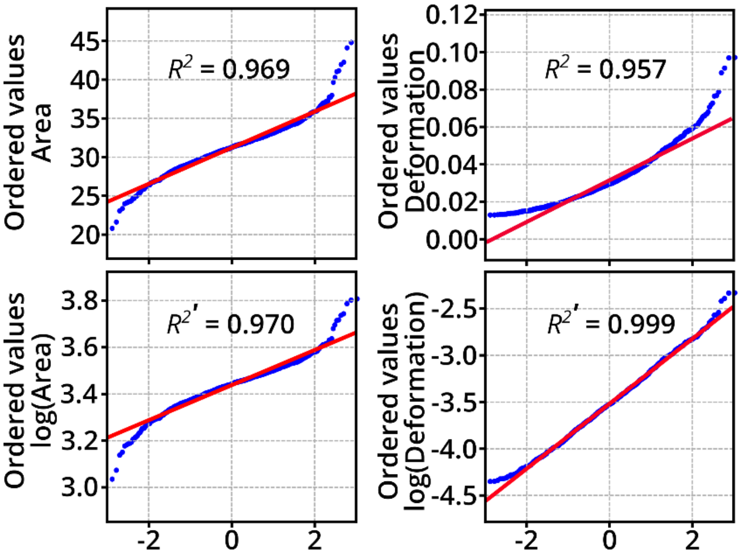


Figure 3.3 Examples of probability plots

Probability plots (introduced in section 2.5) show data (area, log transformed area values, deformation and log-transformed deformation values) of an RT-DC experiment of GFP$^+$ retina cells from an Nrl-GFP mouse at maturation stage P04. The measurement was performed in the channel region (20 µm channel) at 0.04 µl/s. Blue dots represent the experimental data and the red line is a linear fit. The Pearson correlation coefficient quantifies how well the linear function fits the original (R^2) and log-transformed ($R^{2\prime}$) values.

Therefore, R^2 and $R^{2\prime}$ was computed for each measurement (see Table 2) and each feature (A, V, D, I), (using only events where Ω≤1.05) allowing to make a more general statement about the distribution properties of those features. Scatterplots in Figure 3.4 show R^2 and $R^{2\prime}$ of individual measurements for area, volume, deformation and inertia ratio and the color of each dot represents the median value of the respective parameter. The line $R^2 = R^{2\prime}$ is drawn as a guide for the eye and the numbers of events lying above and below this line are given. For all features, there are more events above the line than below ($R^{2'} > R^2$), indicating that the spread of each feature is better modeled by lognormal than by a normal distribution.

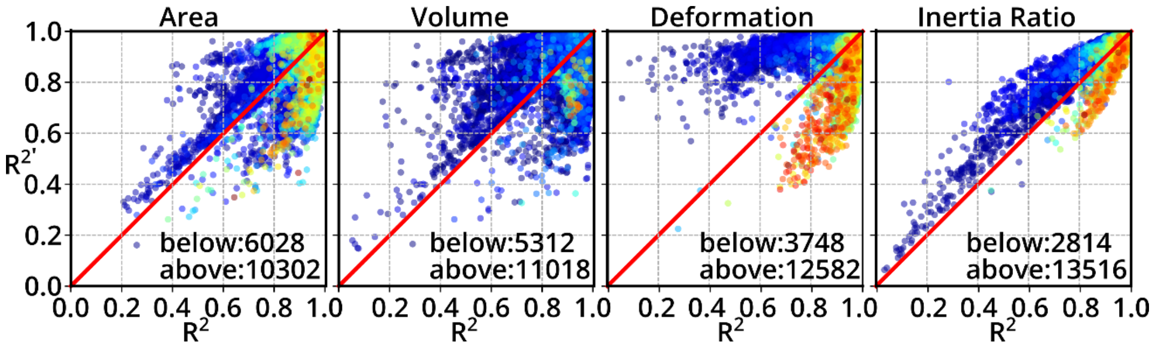


Figure 3.4 Shape of distributions

Scatterplots show the Pearson correlation coefficients of original (R^2) and log-transformed ($R^{2'}$) data for area, volume, deformation and inertia ratio. Each dot represents one experiment and the Pearson correlation coefficients were obtained from probability plots for the original (R^2) and log-transformed ($R^{2'}$) data. Furthermore, the median was computed for each experiment and used to colorize each scatter-dot (blue refers to low values and red to high values). The red line indicates $R^2 = R^{2\prime}$.

3.1.5. Discussion

In this chapter, a meta-analysis was performed in order to discover universal properties of chosen features that are important for RT-DC. A total of 93.7 million measured events from 21,000 experiments that were performed over a time-range of 2 years were used for this analysis. A correlation of area and deformation was found, which can be explained by an increase of the projected area when objects are deformed. Therefore, volume is a more robust parameter to describe cell size, but one has to consider that the current implementation to compute volume relies on a rotation of the tracked contour. Hence, rotational symmetry is assumed, which requires all cells to be aligned to $\overrightarrow{e_x}$, which is for example not the case for reservoir measurements. Furthermore, the channel has to be aligned to $\overrightarrow{e_x}$, which is adjusted manually in the current setup.

Like area and volume, also deformation is computed based on the tracked contour of the object. Imperfect tracking and protrusions or dents on cells can cause an increase of the area ratio. I showed that area ratio and deformation are correlated, indicating that deformation is not only a measure of how stretched a cell is, but also how smooth its contour is. The correlation of area ratio and inertia ratio is slightly lower, indicating that inertia ratio is less affected by non-smooth contours. This indicates that inertia ratio should be favored instead of deformation, especially for non-smooth cells such as macrophages.

After a broader screening of correlations between 17 features, a surprisingly consistent correlation of cell size and transparency emerged. A possible explanation of this effect could be a dilution of the cytoplasm during cell growth which was suggested in a recent publication [126]. Normally, one would expect that larger objects scatter more light and hence appear darker. Such a behavior appears to be true for multiple measurements of beads (I Figure 3.2), since they show a negative correlation between area and brightness in Figure 3.2. Beads such as polyacrylamide microgel beads tend to have a much higher refractive index as compared to cells.

Knowledge about the shape of a distribution can be important information for the choice of a statistical test or for data-modeling. Therefore, I analyzed the shape of the distributions for area, volume, deformation and inertia ratio. For the majority of experiments, area (63%), volume (67%), deformation (77%), and inertia ratio (83%) is better modeled by a lognormal distribution. The trend towards a lognormal distribution is clearer for D and I, which suggests that those features are modulated by multiplicative factors (which could for example be environmental or genetic factors). A more pragmatic explanation could be that errors in contour tracking will always cause an increase of deformation, causing a positive bias. Furthermore, by definition, deformation has a lower limit of 0, which creates a bias of the spread of the distribution towards larger deformation values. In order to design a symmetric deformation distribution (Gaussian) of a given width, one would need to shift the population towards larger values until there is enough space left and right of the population center. This effect can actually be observed in Figure 3.4: while experiments with low median deformation (blue dots) tend to have higher $R^{2'}$, experiments with high median deformation (orange-red dots) are more likely to result in a higher R^2 (indicating Gaussian behavior). The same trend can be observed for inertia ratio. In an identical analysis for D using an older and smaller set of RT-DC data, I came to the same result, which has been published earlier [82]. Area and volume can also not be negative, which might explain the trend that the distributions of A and V of most experiments are better modeled by a lognormal distribution.

Data used for the meta-analyses in this section was collected by multiple users and data-quality was not checked. Moreover, individual measurements could be heterogeneous (contain subpopulations) and the correlation of features could be non-linear. In both cases, the Pearson correlation coefficient would be distorted. In a future, subsequent analysis of this data, a rank correlation could be used, which is robust for non-linear behavior. Furthermore, heterogeneous datasets could be removed or each subpopulation could be gated and analyzed individually. As this meta-analysis is not the main objective of this thesis, only 17 of 140 features were used, indicating potential for further analyses.

3.2. Characterization of retina cells in RT-DC

Retina tissue of Nrl-GFP-mice is already well characterized for several maturation stages using FACS and MACS technology [2,32,34,36,127,128]. In this chapter, label-free morphological and rheological characteristics during retina maturation are assessed using RT-DC. RT-DC was chosen as its high throughput is beneficial to resolve subpopulations contained in retina samples and allows tracking small changes that might occur during maturation.

3.2.1. Maturation of retina cells

Retina tissue from Nrl-GFP mice at the following developmental stages was assessed: embryonic day (E) 15.5 (15.5 days after fertilization of the parent animal), postnatal day (P) 4, 10 and 20. An established FAC-sorting protocol [29,32] was used to obtain the GFP$^+$ (rod photoreceptors) or the GFP$^-$ fraction. For each maturation stage, three biological replicates of the unsorted, GFP$^+$ and GFP$^-$ fraction were measured in RT-DC (using chips with 20 µm constriction and a flowrate of 0.04 µl/s). One representative scatterplot for each condition is shown in Figure 3.5. The uppermost row of plots shows the maturation stage E15.5, which displays a wide population of cell sizes in the unsorted and GFP$^-$ sample, while the GFP$^+$ sample shows a narrow distribution of relatively small cells. A similarly narrow distribution of cell sizes does also appear for the GFP$^+$ fraction of all other maturation stages. Especially at P10 and P20 there are two clearly distinguishable distributions in the unsorted sample and one of them is located in the same region on the scatterplot like the events of GFP$^+$ samples, indicating that GFP$^+$ cells form a subpopulation that is distinct in cell size. But, when inspecting the GFP$^-$ samples, one can see that there are still those two populations, suggesting that some cells of the population with small cell size are not rod precursor cells. For P04 there is no clear separation of subpopulations in the unsorted sample because the population of GFP$^-$ cells is shifted even more towards the region where GFP$^+$ cells are located. For P10 and P20, the cells in the population with larger cell sizes seem to be more deformed in the GFP$^-$ sample as compared to the unsorted sample, indicating that FAC-sorting might altered the mechanical properties of those cells.

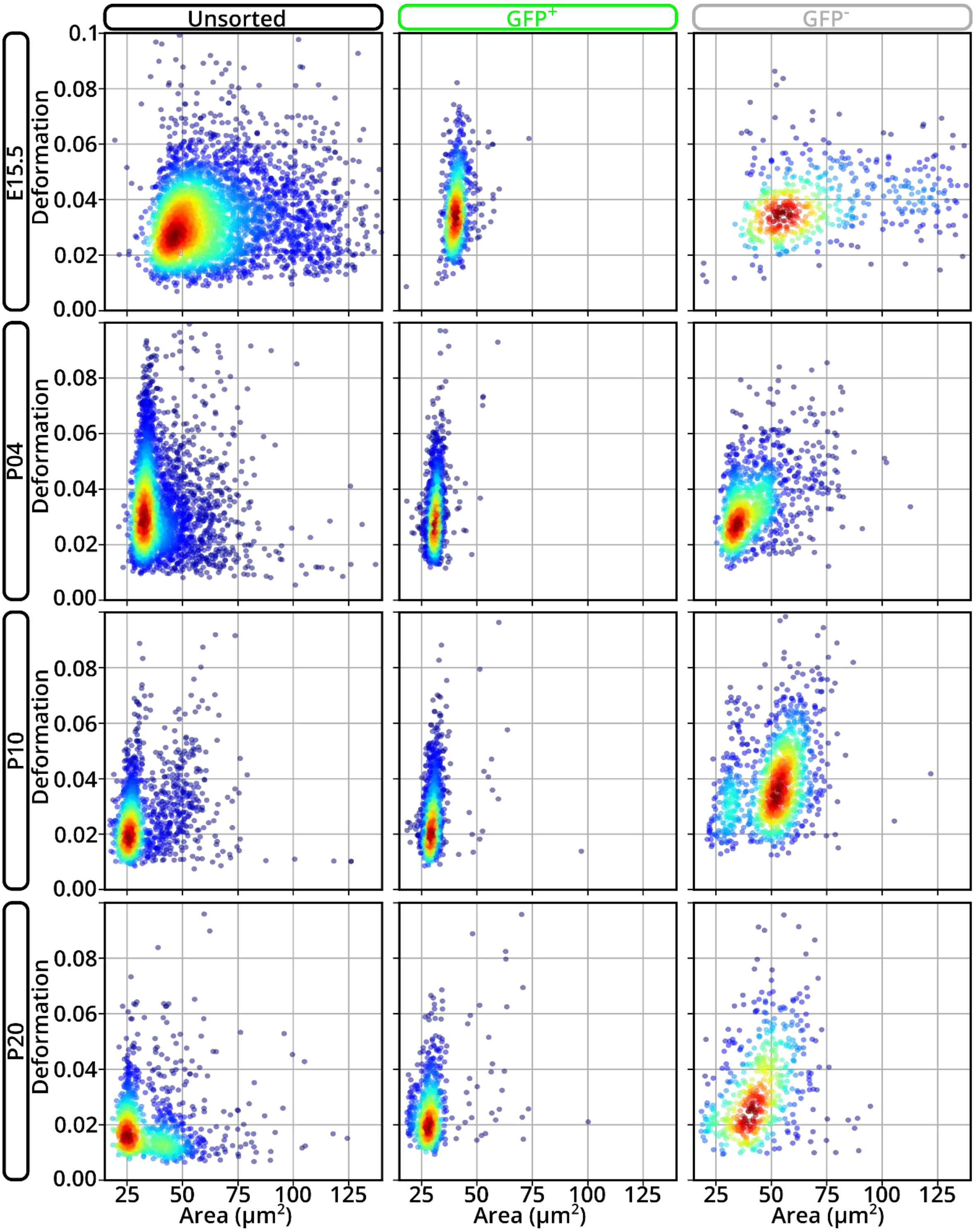


Figure 3.5 Area and deformation of retina cells at different maturation stages

Scatterplots show area and deformation of single cells, resulting from RT-DC measurements of dissociated retina samples. Unsorted as well as the Nrl-GFP$^+$ and Nrl-GFP$^-$ (obtained by FAC-sorting) were measured at multiple maturation stages (E15.5…P20). Color in the scatterplots represents the event-density. Measurements were performed using microfluidic chips with a 20 µm wide channel constriction, a flowrate of 0.04 µl/s and a measurement buffer with a viscosity of 15 mPas (PBS+0.5% methyl cellulose).

To evaluate each subpopulation in mixed samples individually, a two-dimensional Gaussian mixture model was used for clustering. For the unsorted and GFP$^-$ samples, a model with two Gaussians and for the GFP$^+$ samples a model with a single Gaussian was fitted. Since section 3.1.4 showed that area and deformation behave more like a lognormal distribution, the data was first normalized using a log-transformation. Most machine learning models need scaled data in order to avoid issues due to variables that are defined on very different scales. For example, area values are usually above 30 (μm^2), but deformation values are always below 1.0. Therefore, before fitting the mixture model, each variable was scaled by removing the mean and dividing by the standard deviation (the resulting values are often called *standard-score* or *z-score*). After fitting, the resulting model parameters were transformed back to the original scale. Figure 3.6 A exemplarily shows the result of GMMs for maturation stage P04. The found Gaussians define clusters, which are indicated by blue and red dots in the scatterplot and lines in the histogram projections. The superposition of both populations is shown using orange lines and yellow histograms. The mean of each subpopulation for area (μ_A) and deformation (μ_D) as well as the number of data points (N), which is assigned to each cluster, is shown in Figure 3.6 A. The boundary between the cluster of small cells (blue dots) and cluster of larger cells (red dots) is almost vertical, indicating that the GMM uses primarily area to distinguish both populations.

To get an overview about the development of the cell size, the same algorithm has been applied to all other maturation stages and replicates, and the result is shown as a scatterplot in Figure 3.6 B. The leftmost plot shows the result for unsorted samples, where a GMM with two Gaussians was fitted. The means of the population with smaller cell size (blue) correspond closely to the means of the GFP$^+$ sample for each maturation stage. This indicates that the population with small cell size, found by GMM in the unsorted sample corresponds to the GFP$^+$ fraction. Similarly, the mean values of the population with larger cell size in the unsorted sample are similar to the mean values of the population with larger cell size in the GFP$^-$ sample. The displayed error-bars visualize the standard error of the mean, which is quite small for P04, P10 and P20 compared to E15.5, indicating high reproducibility.

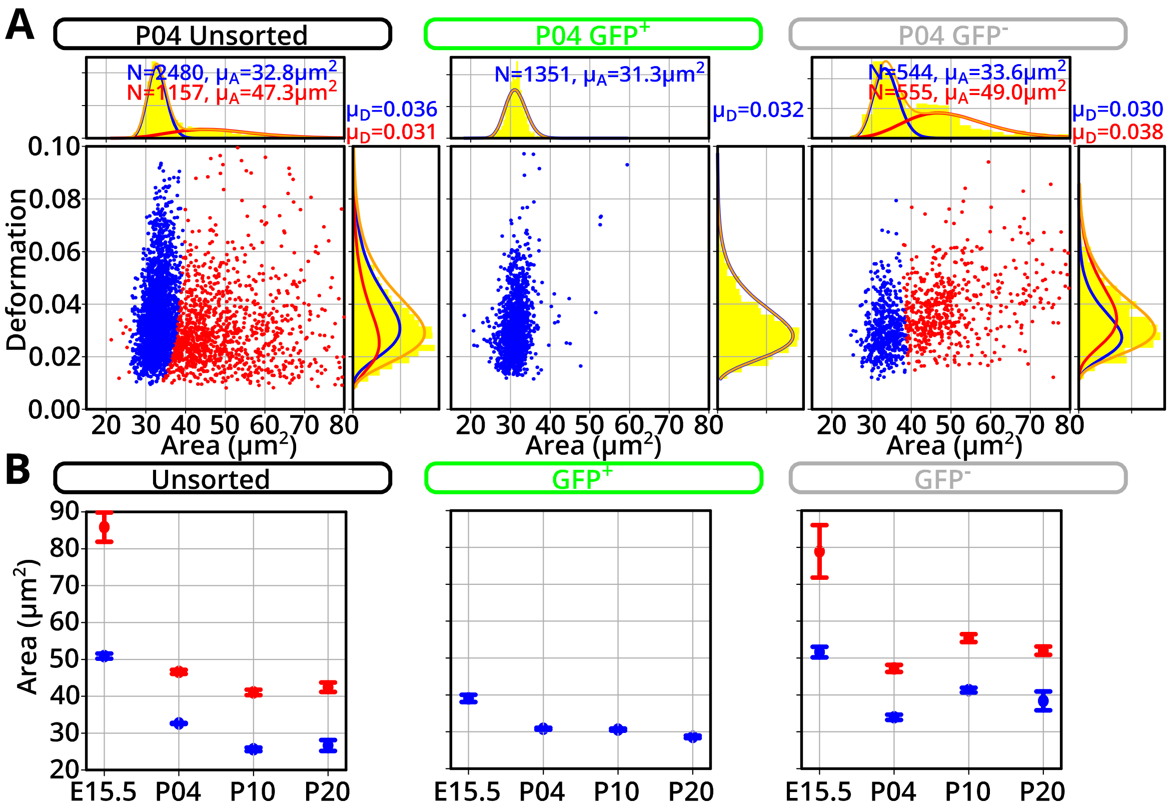


Figure 3.6 GMM for P04 and development of cell size

(A) Scatterplots show the result after fitting GMMs to the data of P04 measurements. The populations with smaller and larger cell size are shown in blue and red color, respectively.

(B) The scatterplot shows the mean area for four maturation stages for the population with small and large cell size (blue and red, respectively). For each maturation stage, three replicates were measured and a GMM was employed to find the mean of each subpopulation. Dots and error bars show the mean and standard error of the mean (SEM), computed using the means of each replicate.

When comparing the populations of GFP$^-$ cells across the developmental stages (see Figure 3.6) one can see that the population with larger cell size shifts towards smaller cell sizes during development. The cells of the GFP$^+$ fraction also show a slightly shift towards smaller sizes during development. The distance between the means of the population with small and large cell size is largest at E15.5 and smaller at P04, P10 and P20. Despite the large size gap between the clusters at E15.5, the clusters are highly overlapping at this maturation stage because of the wide spread (high standard deviation) of the GFP$^-$ distribution. Especially at P10 and P20, most of the cells in the range from 20 to 30 μm^2 are rods since the number of cells in that region in the GFP$^-$ sample is relatively small compared to the number of cells in the population with larger cell size. At P04 there is not such a clear distinction of subpopulations in the unsorted sample, but the GFP$^+$ sample still shows a very narrow size distribution. While the values for area are much smaller for the GFP$^+$ as compared to the GFP$^-$ fraction (compare red and blue cluster in Figure 3.6 A), it is not immediately clear whether the distributions of deformation are significantly different. This topic is highlighted in the next section.

3.2.2. Comparing retina cell types using statistical tests

Student's t-test [129], Mann-Whitney rank test [130] and Kruskal-Wallis H-test [131] are examples of tests, which can be used to compare two independent samples. Figure 3.7 shows the application of these tests to artificial and experimental data. The two artificial datasets were obtained by drawing 1500 values from a normal distribution generated by a random number generator with a mean of zero and standard deviation of one. The histogram on the left in Figure 3.7 shows both distributions (red and blue histogram in Figure 3.7), which are strongly overlapping. Let us assume that sampling and comparing using statistical tests is repeated 100 times. Then, one would expect to get a p-value below 0.05 in 20 cases due to the definition of the p-value [125]. Such an example of a significant difference despite drawing values from an identical random number generator is shown in Figure 3.7 and the table (Figure 3.7, bottom) shows the corresponding the p-values. The histograms on the right in Figure 3.7 show area-distributions of two GFP$^+$ rod photoreceptor samples, which were measured individually using RT-DC. Again, the distributions are highly overlapping, but the resulting p-values when comparing these distributions indicate a significant difference. Using these statistical tests, one is essentially asking for the probability that both samples were obtained by sampling from the same distribution (null hypothesis). In case of the Nrl-GFP data, cells of different mice were measured, which means that cells originate from different populations. Therefore, it makes sense that the tests return low p-values. Even when measuring identical cells from the same donor twice, there could be slight differences due to experimental noise (cells aged, room temperature changed,…). Such small differences are typically not resolved when for example measuring 30 cells, but in in RT-DC the sample size is normally on the order of thousands of cells, which allows detecting minute differences. Assuming, the second sample of Nrl-GFP$^+$ cells was for example drug-treated, it would not be possible to decipher if the statistically significant difference arose due to an effect of the treatment or due to experimental noise.

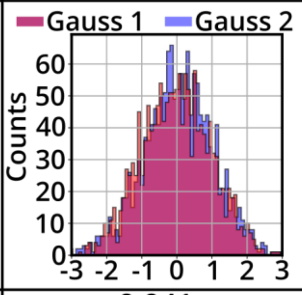
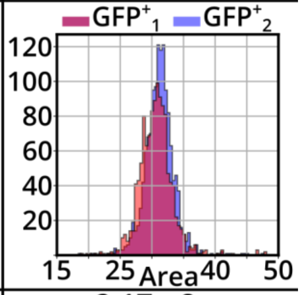
	2 perfect Gaussians	2 GFP$^+$ samples
Histogram		
t-Test	0.041	2.17e-8
Mann-Whitney	0.011	3.55e-13
Kruskal-Wallis	0.021	7.10e-13

Figure 3.7 Application of three statistical tests to artificial and experimental data

The histogram on the left shows two Gaussian distributions (red and blue) with 1500 data-points each, produced by a random number generator. The histogram on the right shows the population of GFP$^+$ cells from Nrl-GFP retina cells at P04 for two biological replicates (red and blue). The distributions in each histogram are compared and tested for significant differences using student's t-test, Mann-Whitney rank test, and Kruskal-Wallis H-test. The table states the corresponding p-values. Despite strongly overlapping populations, the resulting p-values are all below 0.05, indicating significant differences, especially for the Nrl-GFP data.

To compute meaningful significance levels for such large sample sizes, it is important to consider not only the difference between two populations but also how reliably this difference can be measured. Section 2.4 described a method based on linear mixed models and a likelihood ratio test that allows considering biological variation and reproducibility of the effect. To use this model, data from biological replicates is required. Hence, the GFP$^+$ as well as the GFP$^-$ fraction was obtained for three biological replicates by FACS sorting. Three biological replicates of each developmental stage were measured using RT-DC. Figure 3.8 shows boxplots for area and deformation for each measurement. Green and gray boxes show data from GFP$^+$ and GFP$^-$ samples, respectively and p-values (obtained using the LMM-based approach) in each plot indicate whether the difference between GFP$^+$ and GFP$^-$ is significant. Since GFP$^+$ cells are consistently smaller than GFP$^-$ cells for each replicate, p-values indicate a significant difference for area at each developmental stage. At E15.5, deformation of GFP$^+$ is considerably lower compared to GFP$^-$ for the first and second replicate, but not for the third, resulting in a p-value above 0.05 implying a non-significant difference (due to insufficient reproducibility). In contrast, the difference in deformation between GFP$^+$ and GFP$^-$ is very small for each replicate at P04, but this small difference is so similar across all replicates, that the LMM-based significance test returns a p-value of 0.0073. At P10, the differences are larger, but same as consistent, resulting in a p-value of 0.0019. At P20, the differences are also large for the first and second replicate, but not for the third, resulting in a p-value of 0.0276.

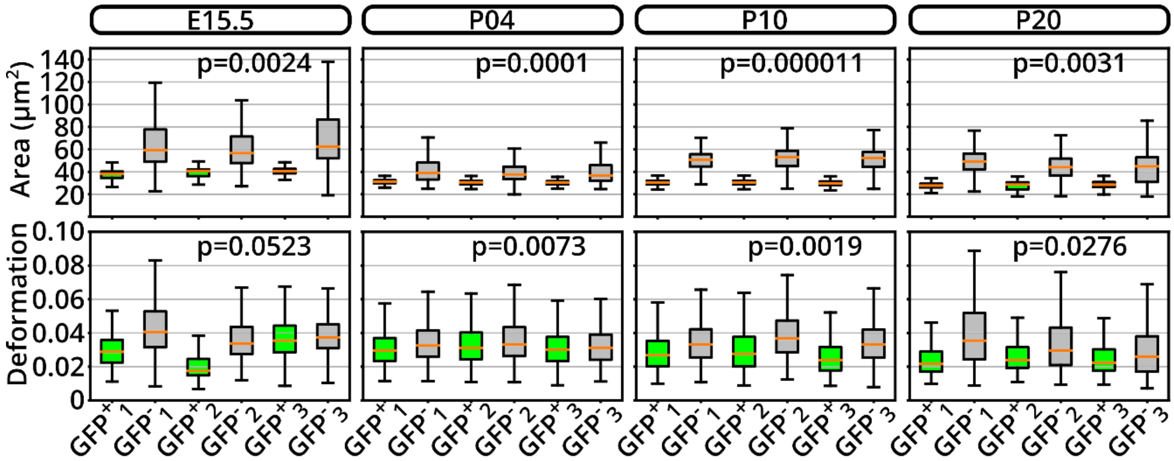


Figure 3.8 Boxplots for area and deformation and statistical analysis using LMM

Boxplots show area and deformation for triplicate measurements of GFP$^+$ (green boxes) and GFP$^-$ (gray boxes) samples at four developmental stages. Subscript numbers at x-axis indicate the replicate number. Deformation and area tend to be smaller for GFP$^+$ cells compared to GFP$^-$ cells and p-values in each plot indicate whether this difference is significant. The p-values were computed using a test based on linear mixed models, which allows taking reproducibility of a measurement into account by considering replicates. Boxplot shows median, interquartile range and range of data, as introduced in Figure 2.7.

3.2.3. Discussion

RT-DC measurements of Nrl-GFP retina samples from mice at different maturation stages reveal a continuous change of morphological and mechanical properties, which is expected since the retina develops rapidly at the chosen ages. Using FAC-sorting, the GFP$^+$ rods were isolated and measured individually in RT-DC, resulting in a narrow distribution of cell sizes for each maturation stage. Such a population of cells with a narrow area-range was also found in unsorted samples. Using a 2D GMM I showed that it is possible to predict the location of the GFP$^+$ fraction in an unsorted sample. Predicting the location of the GFP$^+$ cells is easier for maturation stage P10 and P20 as there are clearly distinguishable populations, but for transplantation, especially samples from maturation stage P04 are of interest due to a higher transplantation success [2]. Despite the narrow area-distribution of GFP$^+$ cells, in an unsorted sample at P04, this population is overlapped by GFP$^-$ cells. To fully discriminate GFP$^+$ and GFP$^-$ cells, more label-free features are required. Therefore, deformation was assessed, which shows a significant difference between GFP$^+$ and GFP$^-$ cells (at P04). For the significance analysis, a test based on linear mixed models was leveraged, which allows to include biological replicates. Other approaches (see Figure 3.7) such as the t-test tend to return lower p-values for larger sample sizes. Therefore, a low p-value can be obtained even for a very small effect by simply measuring more cells. In contrast, the LMM based approach considers if an effect is reproducible across biological replicates. This approach appears to be robust and was used in multiple RT-DC related publications [13,15,22,23,85,97,101–106].

Like many other statistical tests, the LMM based test requires data to follow a normal distribution, but it was shown that the test is robust and results in useful outcome also for considerably skewed distributions [132]. Especially for deformation one could alternatively use a generalized linear mixed model, which uses a log-link function to account for the lognormal behavior of deformation [133]. I implemented this alternative and it was integrated into ShapeOut (courtesy of Paul Müller). Furthermore, LMM requires equal variances of the residuals of the compared distributions (homoscedasticity), which is certainly not given for most biological cases. Recently, approaches that are robust for heteroscedasticity were published [134,135]. Therefore, I implemented a test using the more robust Bayesian hierarchical models (BHM) and compared the p-values resulting for several scenarios (several experiments and artificial datasets) to the p-values from the analogous LMM based test. In general, the p-values were very similar in all cases, but the computational time for BHM was orders of magnitude longer (while LMM took seconds, BHM required multiple minutes), rendering the application of BHM unfavorable, especially when dealing with large amounts of data or many experiments.

The LMM based significance test can also result in a very low p-value even for very small differences, if the effect is highly reproducible. This shows that the p-value only indicates that there is a difference between two states (e.g. between GFP$^+$ and GFP$^-$

sample), but not if the difference is large enough to distinguish single cells from those populations when samples were mixed. Since the goal of this thesis is to find parameters that allow distinguishing rod precursor cells from other retina cells in mixed samples, the next section presents more advanced methods for classification.

3.3. Classification of retina cells using supervised machine learning

The previous section showed that the target cells, which are rod photoreceptors are strongly overlapping with non-photoreceptors in size-deformation scatterplots, especially for retina samples from P04 mice, which are best suited for transplantation [2]. The introduced unsupervised learning method (GMM) allowed to get insight into the data and could be used for classification. A more promising approach to get accurate models for classification is supervised machine learning. For supervised machine learning, a labeled dataset is required, which means that for each captured cell, the Nrl-GFP expression needs to be measured. This is indeed possible with RT-FDC, and therefore, further samples from P04 mice were measured using this technique. The maturation stage P04 was chosen since due to its relevance for photoreceptor transplantation [2].

3.3.1. The dataset

A total of six biological replicates of (not FACS sorted) dissociated retinas from Nrl-GFP mice at postnatal day 4 (P04) were measured using RT-FDC in the channel region of sorting chips (see 2.1.2) and the 488 nm laser was employed to excite the GFP marker, allowing to measure the fluorescence expression. Each biological replicate originated from an individual litter and was measured on a different day. Data from the first three replicates will be used as a training dataset to fit machine learning models. The last three replicates were measured three months later, which allows using the datasets as testing set to check the performance of models in a realistic scenario. Figure 3.9 shows scatterplots of a measurement from the training set. Figure 3.9 A shows the label-free parameters deformation and area on the axes. The top histogram projects the area distribution, indicating two populations, as one would expect for an unsorted sample. The scatterplot also shows a population of more deformed and larger cells, indicating doublets. The scatterplot in Figure 3.9 B shows a combination of a label-free and a label-based parameter on the axes. The y-axis shows the GFP-fluorescence intensity and the x-axis the area. Rod photoreceptors show elevated fluorescence signals and can be found in a narrow area range from 25 μm^2 to 40 μm^2 (see cell type 1 in Figure 3.9 B).

Another population of larger cells with a high fluorescence signal is created due to the presence of doublets (cell type 2 in Figure 3.9 B). The distribution of fluorescence intensities does not show a clearly separate distribution of GFP$^+$ rods, but continuously extends towards lower intensities where a third population of small GFP$^-$ cells is defined (cell type 3 in Figure 3.9 B). This pattern is expected since it is also observed in FACS measurements of these cells [31]. The threshold to distinguish fluorescent and non-fluorescent cells was determined in two ways. First, a threshold was determined using the minimum of the histogram of fluorescence values (T_{hist} in Figure 3.9 B). A second threshold (indicated by T_{GMM} in Figure 3.9 B) was determined using a 1D Gaussian mixture model (GMM) with two Gaussian clusters, which was applied to the distribution of the log-transformed fluorescence values $\log(F)$. The GFP$^-$ population was split at 40 µm^2 in order to have small GFP$^-$ cells (cell type 3 in Figure 3.9 B) as a separate class in the same size region like the rod photoreceptors and an individual class for cells larger than 40 µm^2 (cell type 4 in Figure 3.9 B).

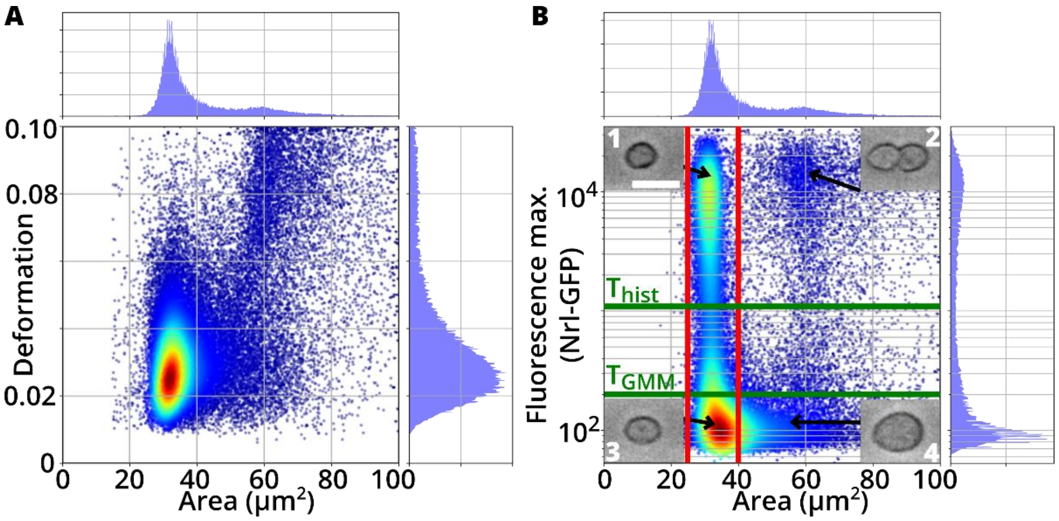


Figure 3.9 Unsorted Nrl-GFP retina cells in RT-FDC

(A) Scatterplot shows an example measurement from one of 6 biological replicates in area vs. deformation space. Cells were measured in the channel region of a 30 μm wide sorting chip at a flowrate of 0.1 μl/s. Debris and very irregular objects were removed from the dataset by gating out all objects with porosity $\Omega > 1.08$. The upper histogram shows area, which has a narrow peak at approximately 30 μm^2. A second population of large and strongly deformed cells is formed due to events of doublets and triplets of cells.

(B) Scatterplot shows the same measurement as in (A), but the y-axis now displays the maximum fluorescence intensity F [arbitrary unit]. In this plot, four subpopulations are distinguished (example images are given as inset images 1-4). In the upper left are highly fluorescent single cells, which can be interpreted as rod photoreceptors (1). Rods are concentrated within a narrow area range from 25 μm^2 to 40 μm^2 as indicated by red lines. Doublets are located in the upper right corner (2). In the lower left are non-fluorescent cells (3), which partly populate the same area-range like rods, but the populations also extend towards cell sizes larger than 40 μm^2 (4). The green lines indicated by T_{hist} and T_{GMM}, define two different thresholds between GFP$^+$ and small GFP$^-$ cells. Scale bar: 10 μm.

Table 3 gives a coarse idea about the number of cells in each population when gating using T_{GMM}. When not restricting to the range from 25 μm^2 to 40 μm^2, there are approximately $\frac{c_1+c_2}{c_1+c_2+c_3+c_4} = (49.2 \pm 4.6)\%$ GFP$^+$ cells in the samples, which is in good agreement with an earlier study, where a percentage of $53.0\% \pm 9.01\%$ was found [31]. The number of GFP$^+$ cells increases to $\frac{c_1}{c_1+c_3} = (53.0 \pm 6.4)\%$, when gating to the range from 25 μm^2 to 40 μm^2, showing that cell size could maybe be used to increase the concentration of rods, especially, when the sample preparation is optimized to avoid cell doublets. The fluorescence threshold found by the GMM-method tends to be very low and close to the population of small GFP$^-$ cells (see Figure 3.9 B) for all experiments. Therefore, an alternative threshold was defined using the minimum of the fluorescence histogram, indicated by T_{hist} in Figure 3.9 B. T_{hist} is considerably higher than T_{GMM} for all experiments. Despite this difference, the resulting cell numbers do not change dramatically ($\frac{c_1+c_2}{c_1+c_2+c_3+c_4} = 47.4\% \pm 3.5\%$ and $\frac{c_1}{c_1+c_3} = 54.7\% \pm 4.7\%$) but show a higher increase of the concentration of GFP$^+$ cells upon gating from 25 µm^2 to 40 µm^2, suggesting that area gating could be used to enrich rod photoreceptors. Area is already an online parameter, which can be used to trigger SAW to sort cells. MACS based enrichment of photoreceptors using CD73 allows to enrich rod photoreceptor precursor cells to $87.7\% \pm 4.7\%$ and shows a significantly higher transplantation success compared to transplanting unsorted cells [31]. Therefore, for label-free rod identification, a concentration of >80% rods is desired. The initial concentration of rods is approximately 50%, resulting in a required enrichment by a factor of approximately $E_{required} = \frac{80\%}{50\%} = 1.6$.

Repl.	c_1 [%]	c_2 [%]	c_3 [%]	c_4 [%]
1	32.7	17.8	22.3	27.2
2	38.2	19.4	25.4	17.0
3	30.3	15.4	31.2	23.1
4	33.9	22.6	20.2	23.3
5	13.1	15.2	44.4	27.2
6	37.2	19.5	21.4	21.8
Mean±SEM	30.9±3.8	18.3±3.8	27.5±3.8	23.3±1.5

Table 3 Nr. of cells when using GMM-based fluorescence gates

Repl.	c_1 [%]	c_2 [%]	c_3 [%]	c_4 [%]
1	30.7	13.9	24.3	31.1
2	27.6	13.5	36.0	22.8
3	23.3	11.8	38.2	26.6
4	35.5	16.7	18.6	29.2
5	36.2	20.6	21.3	21.9
6	36.7	17.9	21.9	23.4
Mean±SEM	31.7±2.2	15.7±1.3	26.7±3.4	25.8±1.5

Table 4 Nr. of cells when using histogram-based fluorescence gates

The cell numbers shown in Table 3 and Table 4 give an estimate of the initial mix of the cell types, which is approximately 50% rods vs. 50% non-rods. Shifting the fluorescence threshold results in a slight change of the classification of cells (rods become classified as non-rods and vice versa). The cells of interest for a prospective sorting application are single GFP$^+$ cells (1). Many GFP$^-$ cells can simply be gated out using the area threshold of 40 µm^2, but the more challenging task is to distinguish, small GFP$^+$ (1) and small GFP$^-$ (3) cells (highlighted in Figure 3.10 A). Therefore, all machine learning models presented in the following will focus on the discrimination of those two classes.

To avoid training of machine learning models on misclassified cells, events in a broad window of fluorescence values (gray regions in Figure 3.10 A) were deleted from the datasets. Misclassification could also occur when multiple cells are contained in the same frame as shown in Figure 3.10 B. Each captured event corresponds to an image with a length of roughly 87 µm. The fluorescence trace for each event (an example fluorescence trace is shown in Figure 3.10 C) has the same length. In case of multiple cells in one frame, it is important to check if florescence peaks are assigned to the correct cells to avoid misclassifications. After filtering for fluorescent cells, the peak positions and the x-positions should be highly correlated and most of the events should be close to a linear fit as shown in Figure 3.10 D. To determine events for which the x-position and fluorescence peak match, two parallel linear functions above and below the fit were defined by modulating the intercept of the fitting function using a factor of 1.15 and 0.85, respectively (dash-dot-lines in Figure 3.10 D). Only events within the both dash-dot-lines were kept for the training, validation and testing datasets.

It might be interesting to note that the slope of the linear fit allows computing the speed of cells since the fluorescence peak position is measured in µs and the x-position of the centroid of the cell in µm. In the example shown in Figure 3.10 D, the slope s of the linear fit ($s = 4.76\ \mu s/\mu m$) corresponds to a velocity of 0.21 m/s, which is close the theoretical velocity computed according to Mietke et. al. [77] using $u' = 2.0962 \cdot \frac{Q}{L^2} = 0.23\,\frac{m}{s}$, with flowrate $Q = 0.1\ \mu l/s$, channel width $L = 30\ \mu m$, and $K_1 = 2.0962$. The constant K_1 results from a representation of the flow field in a channel with squared cross-section using a Fourier series [77].

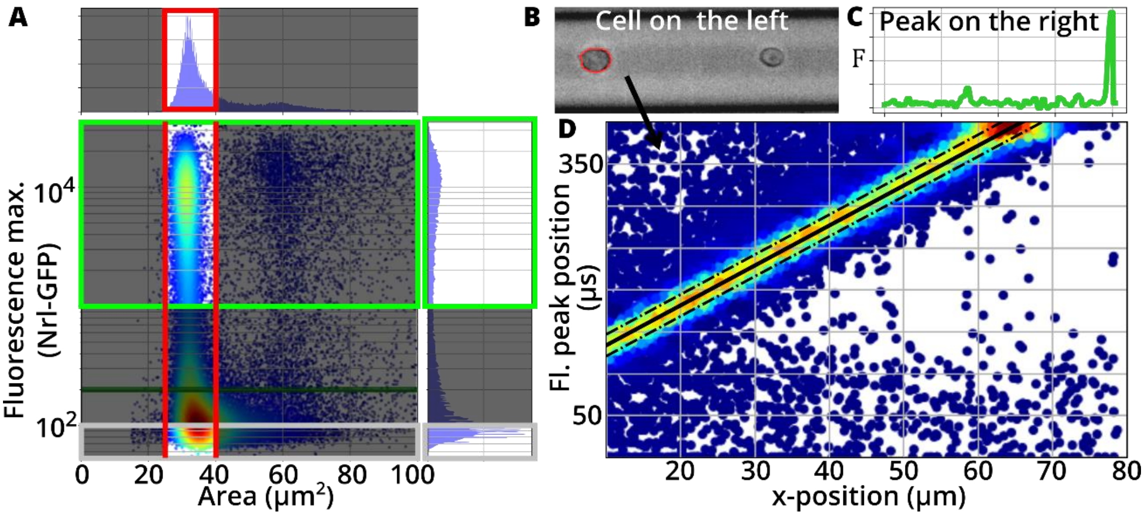


Figure 3.10 Area/ fluorescence gating and fl.-peak assignment

(A) Scatterplot and histograms show the same data like Figure 3.9 B, but additionally illustrate the area range as well as the fluorescence ranges, which were applied to filter the data to create training, validation and test datasets.

(B) Image of a captured event with two cells in the region of interest but only the left cell was analyzed. The corresponding fluorescence trace is shown in C, indicating a mismatch between location of the analyzed cell (indicated by red contour) and location of the fluorescence peak.

(C) The fluorescence trace corresponding to the frame shown in B shows one peak on the right. The maximum of this trace was used to quantify the fluorescence signal of the detected cell (left cell in B), which is a mismatch since the peak was actually caused by a second cell.

(D) Scatterplot showing the position of the fluorescence peak vs. the x-position of the cell. For correct peak-assignments both quantities should be highly correlated. The color-map shows the data-density, which illustrates this correlation. For fitting a linear function, only events with a density, higher than 40% of the maximum density were used.

In machine learning it is good practice to have three datasets, a training set to fit the model, a validation set to quantify the performance of the model on new data after each training iteration (e.g. to diagnose overfitting) and a testing set to check the performance of the final model in a realistic use-case scenario (new experiment, other biological replicate). A validation set was created by taking 167 random small GFP$^+$ cells (cell type 1 in Figure 3.9 B) and 167 small GFP$^-$ cells from the first, second and third dataset, resulting in a total of $167 \cdot 2 \cdot 3 = 1002$ cells. Table 5 shows the number of GFP$^+$ rods (n_1) and GFP$^-$ non-rods (n_3) in the training, validation and test sets after restricting the data to an area range (red lines in Figure 3.10 A), specific fluorescence regions (green lines in Figure 3.10 A) and deleting cells with invalid fluorescence peaks (see

Figure 3.10 D). The following sections will use this data to find algorithms, which allow distinguishing small GFP$^+$ (1) and small GFP$^-$ cells (3).

Training

Repl.	n_1	n_3
1	21670	9868
2	51748	25313
3	27748	33840
Sum	101166	69021

Validation

Repl.	n_1	n_3
1	167	167
2	167	167
3	167	167
Sum	501	501

Testing

Repl.	n_1	n_3
4	37924	11696
5	10201	814
6	18873	14069
Sum	66998	26579

Table 5 Nr. of small GFP$^+$ and small GFP$^-$ cells in training, validation and testing set

3.3.2. Cell classification using optimized area gating

By closely inspecting the distributions of the small GFP$^+$ and GFP$^-$ cells in the red window in Figure 3.10, one can already recognize that the center of the GFP$^+$ population tends to be at a lower cell size compared to the GFP$^-$ population. This results in the question, if an optimized area gating would already suffice to enrich rods to the desired concentration of above 80%. To asses this question, the histograms for area of GFP$^+$ and GFP$^-$ cells are plotted for one biological replicate (training set 1) in Figure 3.11 A and a red line indicates an upper area threshold. All cells below this threshold were classified to be rods and would in practice be sorted. Cells of type 1 (GFP$^+$ rod photoreceptors, green histogram) tend to be smaller than cells of type 3 (GFP$^-$ cells, grey histogram). By moving the upper area threshold (A_{thresh}) to values smaller than 40 μm^2, many GFP$^-$ cells can be gated out, resulting in an increase of the rod concentration in the target region, but also in a decrease of the total number of rods (yield). This analysis has been performed individually for the data of all three biological replicates in the training set and the second plot in Figure 3.11 A shows the mean and standard error of the mean for the yield and the concentration of rods in the target region for various thresholds. The dashed lines show the optimal upper area threshold, which is 32 μm^2 and results in the maximum concentration of rods of 66.48%±3.81%, but only yields 39.53%±11.3% of the available rods. Figure 3.11 B shows the normalized confusion matrix for the validation set when applying the optimal upper area threshold of 32 μm^2 as a classifier to predict. Since the validation set was obtained by randomly sampling from the training set, the area distributions are very similar, resulting in a very similar performance of the classifier in validation and training set. The upper left square shows that 40% of the rods are correctly predicted (true positive = TP), which equals the yield. Cells that are actually rods, but are classified to be non-rods are false negative events (FN). In general, yield is computed using the following equation:

$$yield = \frac{Nr.\ of\ correctly\ predicted\ rods}{Total\ nr.\ of\ rods} = \frac{TP}{TP + FN} = sensitivity. \qquad 3.3$$

Apparently, $yield$ equals $sensitivity$ (see section 2.8). While $sensitivity$ is commonly used in the jargon of statisticians, I decided to use the word “yield” as it easier to interpret, especially in the context of cell sorting.

Approximately 19% of the GFP^{-} cells have an area which lies in the target region and are wrongly predicted to be rods (false positive = FP), which results in a concentration of rods in the target region of $c_{target} = \frac{40}{40+19} = 68\%$. In more statistical terms this equation can be rewritten to

$$c_{target} = \frac{TP}{TP + FP} = precision. \qquad 3.4$$

Again, there is an established phrase ($precision$) that is commonly used by statisticians, but I will continue calling this metric concentration due to its meaning when applying the model for cell sorting.

Figure 3.11 C shows confusion matrices, illustrating the performance of the area gating method on the testing set. The initial mixing condition of 50% rods and 50% non-rods was created for the testing data for each biological replicate individually, by determining the difference in number of events and randomly deleting this number of events from the population, which has more events. The resulting balanced testing datasets were pooled to determine the confusion matrix, which indicates a yield of only 26%, but a similar concentration of rods in the target region of approximately 70%.

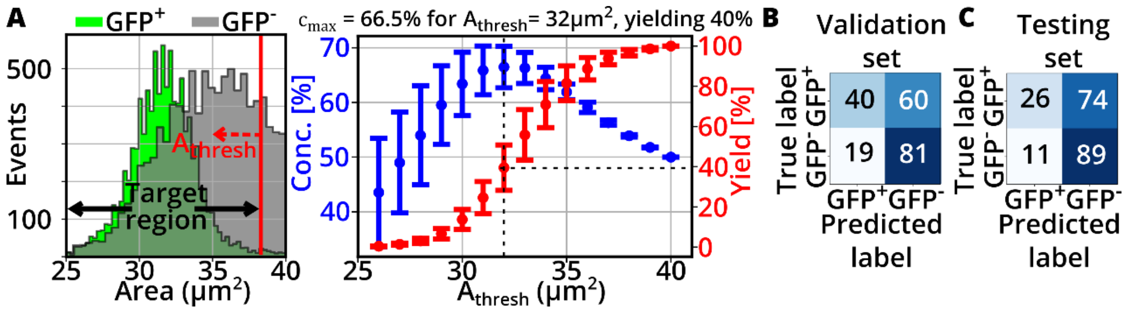


Figure 3.11 Cell classification using area gating

(A) Histogram showing the area distributions of GFP$^+$ and GFP$^-$ events in training set 1. Red line indicates the upper area threshold (A_{thresh}), defining the target region. This analysis has been performed individually for data of three biological replicates in the training set. The second plot shows the mean and standard error of the mean for the yield and the concentration of rods in the target region for various A_{thresh}. Dashed lines show the optimal upper area threshold, resulting in the maximum concentration of rods.

(B) The normalized confusion matrix for the validation set when applying the upper area threshold of 32 µm^2 as a classifier to predict. The concentration of rods in the target region is $69.20\% \pm 4.61\%$.

(C) The normalized confusion matrix resulting when using the area threshold of 32 µm^2 as classifier to predict data from the testing set. The concentration of rods in the target region is $70.30\% \pm 4.61\%$.

3.3.3. Cell classification using random forests

Beside area also deformation, inertia ratio, x-length, y-length, porosity, average brightness, and standard deviation of brightness are online parameters that could be used to trigger SAW for sorting. The optimal thresholds for such a multidimensional problem can be found by training a random forest. Training was performed using the training data shown in Table 5. The final random forest model was then used to compute the probability to be positively fluorescent P(GFP$^+$) for each cell in training set 1, resulting in histograms for GFP$^-$ and GFP$^+$ shown in Figure 3.12 A. The histogram for GFP$^-$ cells (gray) shows a pronounced peak at low P(GFP$^+$), indicating that a large number of cells can be confidently predicted to be not GFP$^+$. For classification of cells to either cell type, a certain threshold P(GFP$^+$)$_{thresh}$ has to be chosen. Cells with a probability below or above P(GFP$^+$)$_{thresh}$ were classified as GFP$^-$ and GFP$^+$ cells, respectively. The concentration of cells that are truly rods within the target region depends on this threshold as shown in the scatterplot (see Figure 3.12 A). By moving P(GFP$^+$)$_{thresh}$ towards higher values, the concentration of rods in the target region increases, but at the same time the total number of rods within the target region decreases. For better comparability, to the area gating method presented in section 3.3.2, P(GFP$^+$)$_{thresh}$ was adjusted such that an average yield of 40% is obtained on the training sets. The resulting concentration of rods in the target region is $c_{40} = 75.7\% \pm 4.09\%$. A more practical reason for the chosen yield of 40% arises from an approximation of the minimum yield as explained in the following. To estimate the minimum yield, required to sort 100,000 cells (necessary number of rods for transplantation) within 3 hours (duration of keeping cells in vitro should be minimized), let us assume that cells are analyzed at a constant rate of 50 cells/s. As shown in section 3.3.1, approximately 50% of the cells are rods, resulting in a frequency of 25 rods/s, which could potentially be sorted. Therefore, the minimum yield is: $yield_{min} = \frac{100{,}000\ rods}{25\ rods/s \cdot 3h} = 37.\% \approx 40\%$. The used PC is actually capable to analyze up to 300 cells/s which would allow using higher cell concentrations, but this would increase the risk of having two cells within the sorting region and accidentally sorting both.

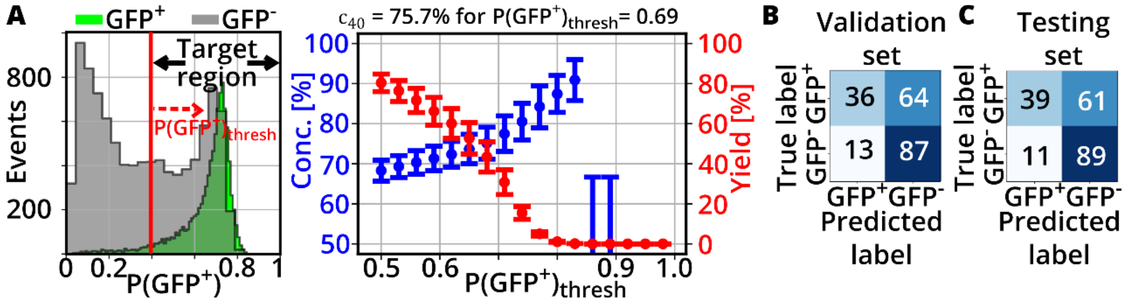


Figure 3.12 Performance of random forest that uses online features

(A) Distributions of the probability to be GFP$^+$ for GFP$^+$ (green) and GFP$^-$ (gray) cells from training set 1. All cells above the threshold P(GFP$^+$)$_{thresh}$ are predicted to be rods. The scatterplot (middle) shows the resulting concentration of rods in the target region and the yield for various P(GFP$^+$)$_{thresh}$. When adjusting P(GFP$^+$)$_{thresh}$ such that a yield of 40% is obtained, the resulting concentration of rods in the target region of three training datasets is $c_{40} = 75.7\% \pm 4.09\%$ (mean and SEM).

(B) The normalized confusion matrix for the validation set when using the random forest and P(GFP$^+$)$_{thresh}$=0.69 to predict. The concentration of rods in the target region is $72.29\% \pm 4.65\%$.

(C) The normalized confusion matrix for the testing set when using the random forest and P(GFP$^+$)$_{thresh}$=0.69 to predict. The concentration of rods in the target region is $77.61\% \pm 1.01\%$. For better comparison, the same data as shown in Figure 3.11 C was used.

While there are still misclassified cells, it is clear from the analyses and the histograms in Figure 3.11 A and Figure 3.12 A that the separation of GFP$^+$ and GFP$^-$ cells is better when using more features to perform the classification. This leads to the question if additional texture and shape descriptors would help to distinguish GFP$^+$ and GFP$^-$ cells even better to improve the classification performance. Therefore, volume, principal inertia ratio, orientation, median and standard deviation of the background intensity, normalized and maximum brightness, Haralick texture features, local binary pattern (LBP), threshold adjacency statistics (TAS), and elliptical Fourier features (EFFs) were computed for all cells listed in Table 5 using the stored contours and images of the events. Next, a random forest model was trained, which results in a validation accuracy of 0.746 and the feature importance values were examined as shown in Figure 3.13 A. “Online” features (red in Figure 3.13) are area, length, height, aspect ratio, deformation, inertia ratio, porosity, mean, and standard deviation of brightness. “Other” features (orange in Figure 3.13) are volume, principal inertia ratio, orientation, median and standard deviation of the background brightness, as well as the normalized and maximum brightness. “Haralick” (yellow in Figure 3.13) are the Haralick texture features. “TAS” (green in Figure 3.13) represents the threshold adjacency statistics. “Fourier” (blue in Figure 3.13) represents the elliptical Fourier features. “LBP” (magenta in Figure 3.13) represent the local binary pattern features.

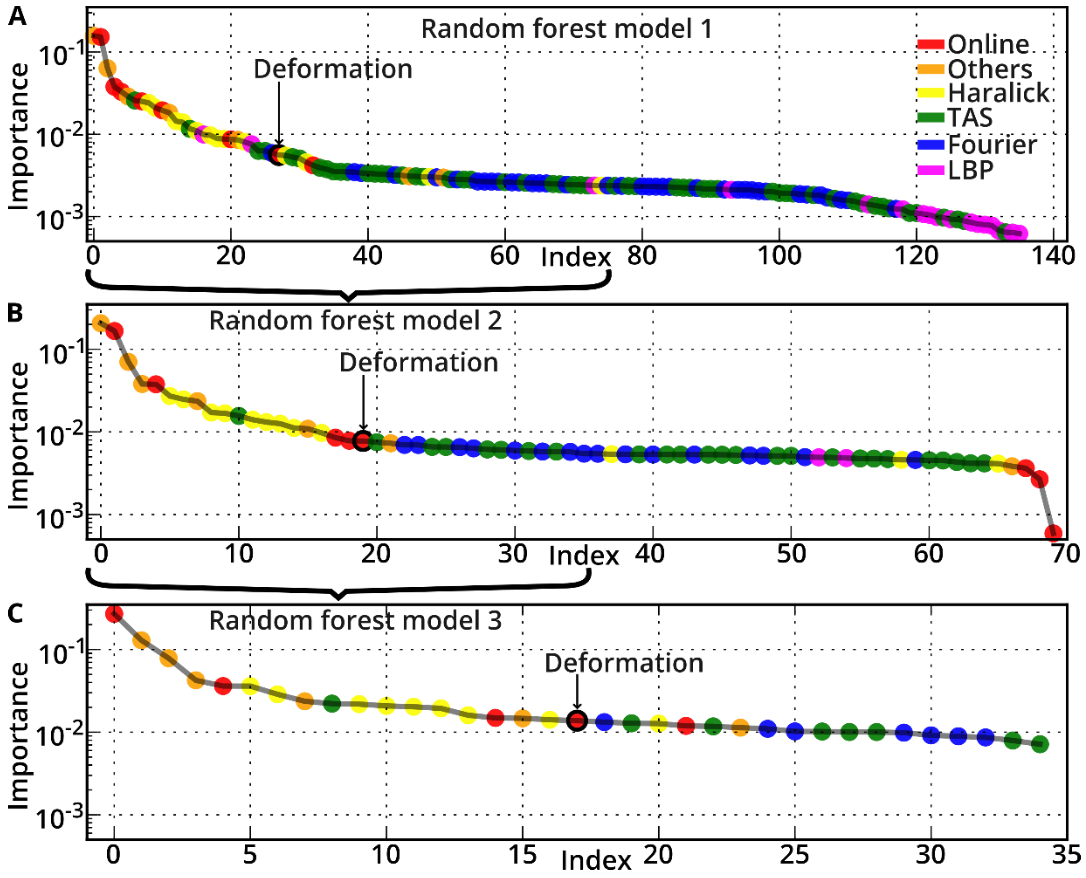


Figure 3.13 Feature importance of online and offline features

(A) Scatterplot shows the sorted feature importance values, resulting after training a random forest using 140 offline and online features. Deformation appears at index 27.

(B) A new model was optimized, which uses only those 70 features, which had the highest feature importance in model 1(A). Now, deformation appears already at index 19, indicating that collinear features were removed.

(C) For the third random forest model, the top 35 features with the highest feature importance of model 2 were used for training. Deformation appears at index 17. The feature names and exact values of the first 20 features are shown in Table 6.

At index zero in Figure 3.13 A one can see the most important feature, which is volume (belongs to “Others“). While online, others and Haralick features have in general high importance, Fourier features and local binary pattern (LBP) features carry only low importance. This could either mean that these features are not useful for classification, or that there is collinearity between the features, such that they have to share importance. For example, if deformation and several Fourier features described a very similar feature, which is helpful for classification, the resulting importance would be shared among those features. To avoid that, the feature importance values from the first random forest model were used to define 70 features with the highest importance, which were then used to train a second random forest. The resulting feature importance values are shown in Figure 3.13 B. Next, the 35 most important features of that model were used to train a third model (see Figure 3.13 C). The resulting feature importance values should reflect better the properties which distinguish photoreceptors from non-photoreceptors. An overview of the 20 most important features, resulting from model 3 (see Figure 3.13 C) is shown in Table 6.

Index	Feature-name	Importance
0	A	0.270
1	V	0.129
2	$B_{back.std}$	0.078
3	B_{norm}	0.042
4	B	0.036
5	$Haralick_6$	0.036
6	$Haralick_{12}$	0.028
7	B_{max}	0.024
8	TAS_{19}	0.022
9	$Haralick_3$	0.022

Index	Feature-name	Importance
10	$Haralick_{10}$	0.021
11	$Haralick_{11}$	0.020
12	$Haralick_{13}$	0.019
13	$Haralick_4$	0.016
14	B_{std}	0.015
15	$B_{back.median}$	0.015
16	$Haralick_2$	0.014
17	D	0.014
18	$Fourier_4$	0.013
19	TAS_{46}	0.013

Table 6 Feature importance values of random forest model that uses 35 features

The process of deleting features with low predictive power might seem like a reduction of information which could decrease the validation accuracy. In practice, often the opposite is the case. Here, the model, which uses all features, reached 74.6% validation accuracy while the models, which used 70 and 35 features, reached 75.7 and 76.2% validation accuracy, respectively. Apparently, a reduction of features continuously improves the validation accuracy. Therefore, the feature importance values from the third model were used to select the 20, 15, 10 and 5 most important features and a random forest model was trained for each case. The resulting optimized models reached a validation accuracy of 75.9% (20 features), 76.8% (15 features), 75.6% (10 features) and 75.3% (5 features), respectively. The corresponding concentrations of rods in the target region for a yield of 40% were $c_{40} = 80.0\%, 83.5\%, 77.6\%$ and 78.1%, respectively. Apparently, the model which uses 15 features achieved the best performance and was therefore chosen for further assessment (see Figure 3.14). The probability histogram in Figure 3.14 A clearly shows two separate peaks for the GFP^+ and GFP^- population and allows a much better separation of these two populations compared to the random forest which only uses online features (see Figure 3.12 A) or area-based classification (see Figure 3.11). The concentration and yield plot shows that the concentration of rods in the target region can even go beyond 90% (but at a yield below 5%). Also, the concentration of rods in the target region for the validation (see Figure 3.14 B) and testing set (see Figure 3.14 C) is higher as compared to the random-forest that uses only online features. Therefore, it can be concluded that including information about the background as well as texture properties of the object (primarily Haralick features and TAS as shown in Table 6) significantly helps to distinguish rods from non-rods.

This evaluation shows the potential of texture properties as a label-free marker, but the abundance of such texture properties is large and it is not clear whether the Haralick and TAS features are the optimal choice for all applications. Furthermore, it could be difficult to perform the computation of Haralick and TAS features and in real-time. The image processing library “mahotas” [88] provides C++ implementations for Haralick features, TAS and LBP, which require approximately 4 ms, 8 ms and 2 ms, respectively for a single image from an RT-DC dataset (on an Intel® Core™ i7-4810MQ @ 2.80 GHz). These numbers show that even computing LBP alone, would take approximately 10 times longer than computing the current online features (which takes approximately 150 µs). Long computational times could in principle be countered by increasing the distance between analysis and sorting region, but this would reduce the achievable sorting throughput. Therefore, it would be advantageous to have a machine learning tool which allows to compute a minimal number of very suitable features that are optimized for a given classification task. Deep neural nets are such a tool and therefore the next sections will introduce methods to optimize DNNs for real-time analysis.

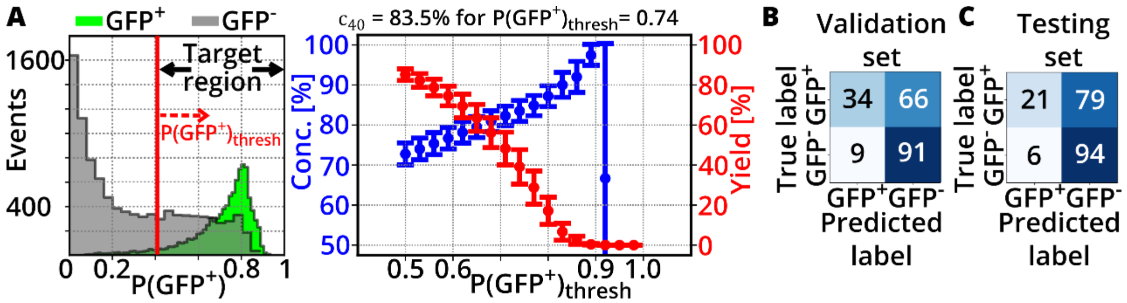


Figure 3.14 Performance of random forest that uses 15 online and offline features

(A) The best performing random forest uses only 15 particular online and offline features (see Table 6). The histogram shows the resulting probability distributions of GFP^+ and GFP^- events from training set 1. The gray histogram (GFP^-) as well as the green histogram (GFP^+) show pronounced and separated peaks. When adjusting $P(GFP^+)_{thresh}$ to get a yield of 40%, the concentration of rods in the target region is in average $c_{40} = 83.5\% \pm 2.63\%$ for the three training datasets.

(B) The normalized confusion matrix for the validation set when using the random forest and a threshold of $P(GFP^+)_{thresh}$=0.74 to predict rods. Here, the concentration of rods in the target region is $77.91\% \pm 1.79\%$.

(C) The normalized confusion matrix for the testing set when using the random forest and a threshold of $P(GFP^+)_{thresh}$=0.74 to predict rods. Here, the concentration of rods in the target region is $77.30\% \pm 0.69\%$. For better comparison, the same data as shown in Figure 3.11 C was used.

3.3.4. Cell classification using deep neural nets

Current development in the domain of deep learning mostly focuses on developing more complex and accurate algorithms, which often require a high-performance graphics processing unit (GPU) or even a GPU cluster to execute training or inference in a reasonable time. For image-based sorting, there is a restriction in terms of inference time (time to classify one image) which has to be below 1ms (ideally around 150 µs, matching the computational time of current online parameters). Such short inference times exclude the usage of online GPU clusters since sending the image to the cluster and receiving back the information is linked to long and varying latency times. Batch-wise processing would reduce the total time per image since GPUs are optimized for parallel computation, but this is not applicable for image-based cell sorting since inference has to be performed image-by-image. Therefore, on-board hardware has to be used for inference.

Inference time and accuracy of a deep neural net (DNN) are dependent on the complexity of the DNN. More complex DNNs can be more accurate, but at cost of more computational power and longer inference time. For real-time classification, an accurate and yet very fast model is required. Convolutional neural nets are currently very popular because of their enormous performance in image classification, audio signal processing and other tasks where they can even beat human performance [136]. A basic deep neural net architecture where each node of one layer is connected to each node in the next layer is called multilayer perceptron (MLP). MLPs are characterized by the number of hidden layers and the number of nodes in each hidden layer. Convolutional neural nets are just special cases of multilayer perceptrons. While each input node is connected to every node in the next hidden layer in MLPs, CNNs use filters to restrict the network to link neighboring pixels within a certain filter size. This technique tends to result in more accurate classification algorithms, especially for small datasets, but convolutional filters typically require longer computational times (see appendix A) due to implementation details (convolutions are realized using sparse matrix multiplications). Since the task of real-time classification requires fast inference, a screening of different MLP architectures was performed. Each screened MLP had an input layer, three hidden layers and an output layer as sketched in Figure 3.15 A. A total of 369 different network architectures were defined, which differed in the number of nodes in the hidden layers (n_1, n_2, n_3). The smallest network had 16 nodes and the largest had 72 nodes in each hidden layer. To limit the computational time, models were trained using a subset of all 6 datasets.

Another important factor which influences the inference time is the size of the input image. The larger the input image, the more multiplications are performed in the first hidden layer. Original images of RT-FDC are typically 256x96 pixels in size, but cells only cover a small region on that image. To avoid performing a large number of unnecessary computations, the images were cropped to 32x32 pixels such that the cells appear mass-centered in the image (cropping with respect to the centroid). Such cropped images are then used as input for the neural nets.

The uppermost scatterplot in Figure 3.15 B shows the maximum validation accuracy of each trained model vs. its numbers of parameters. As expected, more complex models with more parameters tend to achieve higher maximum validation accuracies. One model with 25458 parameters stands out since it has a higher validation accuracy compared to other models with a similar number of parameters. This MLP has 24 nodes in the first, 16 in the second and 24 nodes in the third hidden layer (magenta dot in Figure 3.15 B). Such model architectures with funnels (low number of nodes in a hidden layer in the middle of the network) are typically used in variational autoencoders [137] with the aim to get a compressed representation of the input data in a hidden layer which is then used to reconstruct the data. Here, we do not want to train models to compress and reconstruct, but to classify the data it into two classes, which might result in a similar task since the model has to learn the general phenotype of the cells which allows it to robustly classify new data. MLP_1 was chosen for further assessments (see next sections). A second MLP was chosen by considering Master's rule, which states that the number of nodes should decrease when going from input to output layer, forming a pyramid structure [138]. The chosen MLP has 64 nodes in the first 32 in the second and 16 nodes in the third hidden layer (red dot in Figure 3.15 B). The scatterplot in the middle of Figure 3.15 B shows the maximum validation accuracy of each trained model vs. its inference time (time to predict a single 32x32 pixel image) on a CPU (Intel® Core™ i7-3930K @ 3.20 GHz) in a Python environment. This plot shows less correlation, which suggests that two models with different numbers of parameters can still have the same inference time. This is possible due to the capability of the CPU to compute a certain number of parameters of the model in parallel. Nevertheless, the more accurate MLP_2 (t_1=0.11 ms) has a slightly longer inference time compared to MLP_1 (t_1=0.107 ms). In general, the inference time of these models is very small, such that both would be suitable for online analysis and sorting. The inference time when using a GPU (NVIDIA® GTX™ 1080) is shown in the scatterplot at the bottom of Figure 3.15 B. GPUs allow to process orders of magnitude more operations in parallel than CPUs. As a result, the scatterplot in Figure 3.15 B (bottom) shows no correlation between maximum accuracy and inference time. Since the data first has to be transferred to the GPU, there is an additional delay, which causes that the GPU inference times are all approximately six times longer than the CPU inference times.

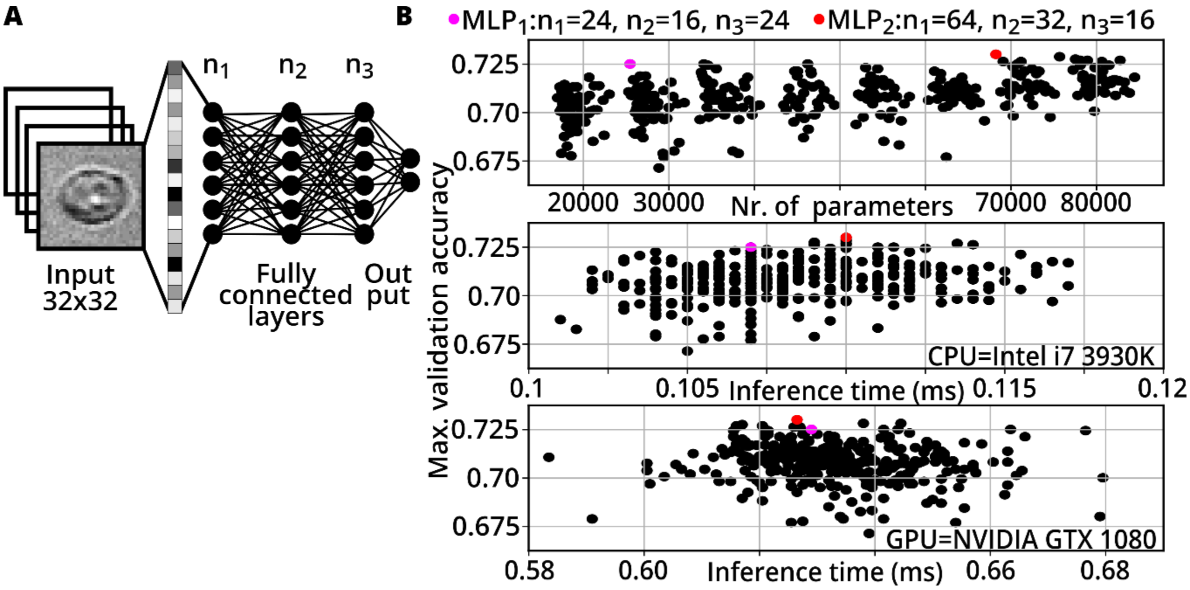


Figure 3.15 Screening of MLP architectures

(A) Basic architecture of the screened MLPs. The raw images (mass-centered body of the cell in 32x32 pixel image) are the input of the model. All architectures consist of three hidden layers with defined numbers of nodes (n_1, n_2, n_3) and two output nodes, which refer to two classes of cells (photoreceptor/ no photoreceptor).

(B) A total of 369 different MLP architectures were trained and the performance of each architecture was quantified using the maximum validation accuracy. The x-axis in the uppermost scatterplot shows the number of parameters of each MLP. MLPs with more parameters tend to deliver a higher accuracy. Two models are highlighted (magenta and red dot) because they deliver a higher accuracy compared to models with a similar number of parameters. The number of nodes in each hidden layer of these two MLPs is indicated above the plot.

The scatterplot in the middle shows the same data as the top scatterplot, but the x-axis was changed, which now shows the inference time for a single image on a CPU. The inference time for the model with fewer parameters (MLP_1) is lower.

The bottom scatterplot analogously shows the inferences time on a GPU, which are approximately 6 times longer compared to the inference times on CPU for all models.

To obtain a benchmark for the best classification performance, if there was no restriction in terms of inference time, also different CNNs were trained. To obtain the highest classification accuracy, very complex CNN architectures should be used, but they often tend to overfit. To prevent overfitting, the regularization technique dropout was used. Dropout is based on the idea that a certain number of nodes in the network is randomly switched off during training. A CNN with 6 subsequent blocks was designed (see Figure 3.16 A). The first four blocks consist of a convolutional, a batch normalization, a ReLU-activation, and a dropout layer. The number of convolutional filters is c_1 in the first and second block and c_2 in the third and fourth block. While the filter size of the first and second convolutional layer is 3x3, the filter size is increased to 5x5 in the third, and to 7x7 in the fourth convolutional layer, in order to increase the receptive field of a single filter (see Figure 3.16 A). Larger filter sizes (i.e. larger receptive fields) allow describing larger features in the image. In the fourth block an additional subsampling layer is inserted between ReLU-activation and dropout layer, which reduces the number of pixels in the feature map (FM) by a factor of 4 and helps to facilitate translation invariance of the model. The feature maps (FM) continuously decrease in size because the computation of a convolutional filter is stopped at the border of the image. The fifth and sixth block each consist of a fully connected layer with 512 and 265 nodes, respectively, a ReLU-activation layer, a batch normalization layer, and a dropout layer. An identical dropout rate of d is used in the dropout layers at the end of each block. To find a very performant model with ideal hyperparameters, a total of 772 CNNs with different d (range from 0.1 to 0.6) and different numbers of convolutional filters c_1, c_2 (range from 4 to 36, respectively) were trained. The training accuracy of most models quickly surpassed the validation accuracy (see Figure 3.16 B), which is a sign of overfitting. Training was terminated when the mean training accuracy over the last 100 training iterations was larger than the respective mean validation accuracy. The CNN with $d = 0.4$, $c_1 = 6$ and $c_2 = 36$ (CNN_1 in Figure 3.16 C) outperformed all other models and was therefore chosen for further assessment. This CNN has a maximum validation accuracy of above 0.81, while the best performing MLP does not even surpass a validation accuracy of 0.73.

While convolutional filters are only a special case of fully connected layers, CNNs tend to generalize better and obtain better performance with less data compared to MLPs. Therefore, optimized training routines, more data and image augmentation can help to further increase the performance, especially of the MLP architectures. In the following section, the training routine for the models which were highlighted in the MLP and CNN screenings are further optimized in order to improve the validation accuracy.

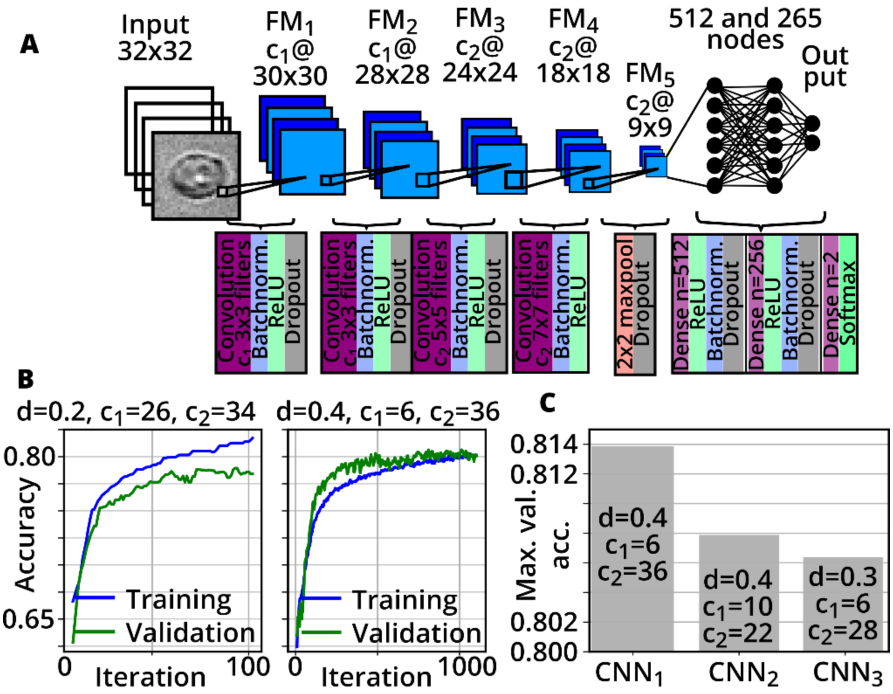


Figure 3.16 Screening of CNN architectures

(A) Sketch of screened model architectures. The architectures each consist of four blocks with convolutional layers with different filter sizes and two blocks with fully connected layers with 512 and 265 nodes, respectively. Each block also contains a batch normalization layer, a ReLU-activation layer and a dropout layer. After the last convolutional block, the feature maps are subsampled by a maxpooling layer with a 2x2 filter. The output is obtained by a layer with two nodes, which is modulated by a softmax activation function.

(B) Left plot: training and validation accuracy of a model that shows overfitting (training accuracy becomes larger than the validation accuracy). Most of the screened models showed overfitting characteristics. Right plot: Desirable behavior of training and validation accuracy: both curves converge and the validation accuracy tends to be slightly higher than the training accuracy.

(C) Bar plot showing the maximum validation accuracy of the three best models after screening of 772 CNNs. The model, with dropout rate $d = 0.4$, 6 convolutional filters in the first and second convolutional layer and 36 convolutional filters in the third and fourth convolutional layer outperformed all other architectures.

3.3.5. Improving DNN accuracy using image augmentation

A useful model for predicting photoreceptor cells needs to be robust for all changes that could occur between or during experiments. This includes changes in illumination and focus, which can result in substantial alterations of the image phenotype. Moreover, the alignment of the microfluidic chip could differ, resulting in images of slightly rotated cells. The lithium niobate ($LiNbO_3$) substrate, which is required for generating standing surface acoustic waves in microfluidic sorting chips is also a source of image distortion. Beside birefringence, which is corrected by a polarizer (at the cost of losing light), $LiNbO_3$ also causes scattering noise [139,140], which varies for different units of sorting chips (see Figure 3.17).

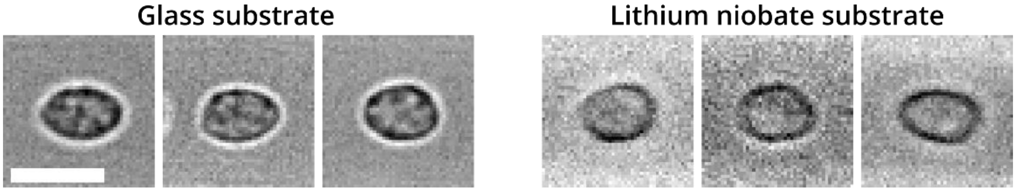


Figure 3.17 Image distortion by lithium niobate substrate

Images of cells, recorded using three different microfluidic chips with glass substrate (left) and three different microfluidic chips with $LiNbO_3$ substrate (right). Each chip has a channel width of 30 μm and the images are recorded in the channel at a flowrate of 0.1 µl/s using 40x magnification. Scale bar: 10µm.

Furthermore, the appearance of cells from different biological replicates differs, potentially resulting in a high variability of the data. The ideal dataset for training a robust model would consist of a large number of experiments, capturing all possible variations of illumination, focus, rotation, scattering noise and biological variation, but of course it would require an enormous effort to create such a dataset. Therefore, this section introduces methods for optimized data acquisition and data generation to reduce the required experimental time and the demand of biological replicates.

Optimized acquisition of training data

While, the focus is usually left constant during RT-DC experiments, it should be altered continuously during acquisition of the training dataset. This is a very simple, yet important task since otherwise the resulting data would support overfitting to particular focus settings. The range of focus alterations should be chosen similar to the range that appears in usual RT-DC experiments and also during future sorting applications. Similarly, the illumination should be continuously altered by changing the opening of the aperture on the microscope. Both, the focus and illumination altering has to be done gently in order to avoid problems during online tracking.

While the change of focus and illumination setting can be quickly realized in a continuous fashion during the experiment, the rotation of the chip in the chip holder under the microscope requires a stop of the acquisition and an opening of the microscope head, which takes time and was therefore only performed two to four times during the acquisition of the datasets used in section 3.3. Since the image distortion is different for different sorting chips, each biological replicate was measured in several sorting chips but since the change of a chip requires even more time, it was performed only up to three times for one biological replicate. Alteration of focus, brightness, rotation, and change of the sorting chip was performed for the all training datasets introduced in section 3.3.1.

Image augmentation

Image augmentation is a very common practice to improve performance and robustness of machine learning models for image classification. In order to obtain a dataset representing a continuous distribution of rotations, a mathematical image transformation was applied. Rotation of images only requires that the input image is slightly larger than the final image to avoid cutting off edges as shown in Figure 3.18. Moreover, also random vertical flipping can increase the information content of the dataset. In contrast, horizontal flipping would result in images in which it looks like cells would travel in the opposite direction, which never occurs in RT-DC and would therefore not help to train a more accurate model. Additionally, brightness augmentation of each image I was performed using a linear transformation with random numbers r_1 and r_2: $I' = r_1 \cdot I + r_2$. Augmentation was performed on the fly during the model training process.

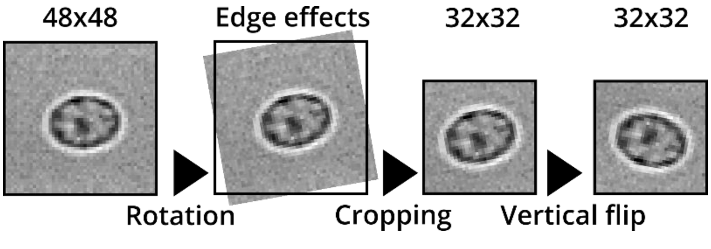


Figure 3.18 Image augmentation by rotating and flipping

Initially, all recorded images were cropped to 48x48 pixels with the cell mass-centered in the image. Next, the images were randomly rotated by ±3° (example shows 10° for better illustration) and finally cropped to 32x32 pixels. Additionally, vertical flipping was applied on a random basis.

The captured dataset reflects the phenotype of only a few biological replicates measured in some sorting chips with certain image distortion levels arising from the $LiNbO_3$ substrate. To augment the data to obtain a continuous distribution, covering all nuances of biological variation and image distortion, a mathematical transformation mimicking these differences is required. Ideally, a dataset to train such a function would contain a version of each cell for every possible combination. Since we only have an unpaired dataset with different cells at different conditions, a Cycle-GAN is an appropriate choice to learn the mapping between the image properties [123] (see section 2.9). Cycle-GANs show an impressive performance in transforming the style of an image reaching almost photorealistic quality.

Let us consider the image distortion of sorting chip A and B as style domains A and B. Then, a Cycle-GAN can be trained to transform images of domain A into images that look like they were measured in sorting chip B (style domain B). Training the Cycle-GAN, results in two generator functions G and F, mapping from style A to style B and from style B to style A, respectively. Similar to image distortion in sorting chips, also different biological replicates differ in appearance, defining individual style domains. As a result a multitude of transfer functions could be generated, which map between each possible pair of distortion levels and biological replicates. To limit the computational cost, a random selection of seven combinations of experiments was used to train corresponding Cycle-GANs. Some examples of transformed and reconstructed images were plotted after each training iteration as shown in Figure 3.19 A to allow for visual inspection of the image quality. Since the quality of some iterations was equally good, I kept up to four versions. The versions only differ in the number of training iterations, and show slightly different transformation results. Since the aim of image augmentation is to alter the training data in a multifarious but meaningful manner, each version which delivers good (according to visual inspection) results can be used to generate data. A total of 27 different Cycle-GANs was kept, mapping between the seven chosen style domains. While a Cycle-GAN was actually only trained to map images from one given style A to another style B (G) and vice versa (F), the transformation functions can also be applied to datasets of different styles C, D, F,… resulting in images with entirely new styles (C’, D’, F’,…) that were not contained in the initial dataset. The principle is sketched in Figure 3.19 B under the assumption styles could be defined in a two-dimensional space and a generator is defined by a vector, mapping from one style to another.

Consequently, by applying each Cycle-GAN to each dataset it is possible to generate images showing novel nuances of image distortions, which were not present in the initial training dataset, but could potentially occur in a new sorting chip or biological replicate (see Figure 3.19 C). Since the computational cost for the transformation of an image is quite large, only a random sample of 500 images of photoreceptors and non-photoreceptors was transformed from each dataset. Since there are 27 Cycle-GANs which were applied to three training datasets, a total of 81 new datasets was created, each containing 2000 images (500 images of photoreceptors transformed by G and by F and equivalently for 500 images of non-photoreceptors). This results in a total of 162,000 generated images, which can be used during training of the neural network.

Yann LeCun, one of the most respected researchers in the field of deep learning, thinks "Adversarial training is the coolest thing since sliced bread" [141], but GANs are still heavily studied to get them working reliably and practical applications reach from text to image translation [142] over image super resolution [143] to image manipulation for creating novel artworks [144]. To my knowledge, Cycle-GANs were not yet used to produce training data for the improvement of model performance. Therefore, the following subsection compares models that were trained with and without generated data.

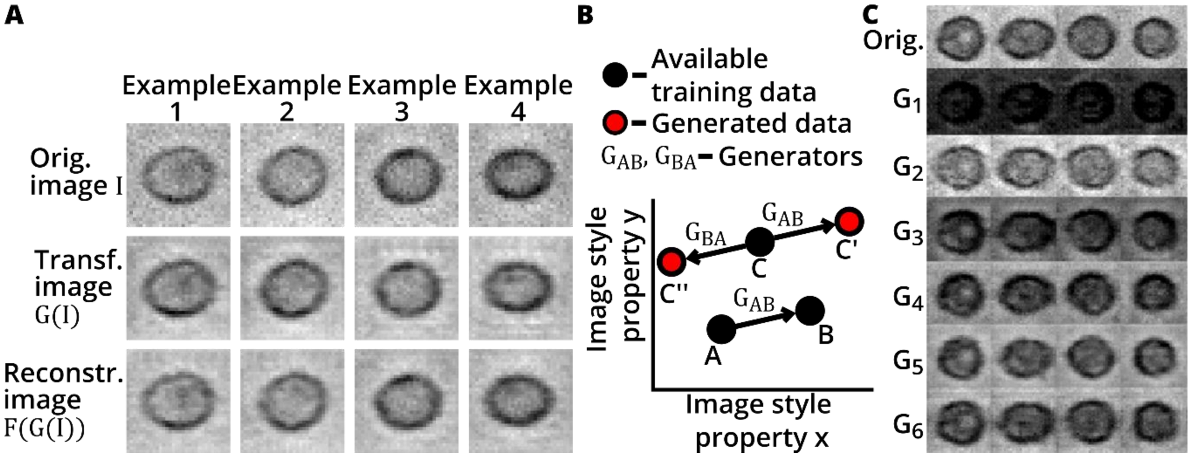


Figure 3.19 Image transformation using Cycle-GANs

(A) Four examples of images, which were transformed by a trained Cycle-GAN. The performance of the Cycle-GAN was checked by visually inspecting the quality of the transformed images. The transformed images show unique image distortions. The reconstructed images $F(G(I))$ do not look identical to the original images, indicating potential improvement of the Cycle-GAN.

(B) Sketch, indicating three training datasets (black dots A, B and C) in a 2D style domain space. A Cycle-GAN was trained to learn the transformation between A and B, resulting in the generators G_{AB} and G_{BA}, which are vectors mapping between points in the 2D style domain space. By applying G_{AB} and G_{BA} to the third dataset C, new datasets C' and C'' (red dots) with unique style properties were generated.

(C) Images of four random cells, which are transformed by six different generators ($G_1 \dots G_6$) resulting from trained Cycle-GANs. Each Cycle-GAN was trained to learn the mapping between different datasets.

Retraining models using augmented and generated data

In this section, the MLP and CNN network architectures, which appeared to be most suitable for the task of classifying retina cells (see section 3.3.4), are retrained using the aforementioned augmentation methods.

In each training iteration, 500 random cells of each class (photoreceptor and non-photoreceptor) and of each biological replicate in the training set (3 in total) were used, which results in a total of 3000 images. I decided to use the same number of cells from each training set for each training iteration since the total cell number differs for each replicate and it is important to avoid overfitting the algorithm to the largest dataset. Similarly, the number of rods and non-rods was kept equal for each training iteration to prevent the algorithm from finding the trivial solution which would mean that it simply classifies each event as non-rod since non-rods occur more often in the dataset. Models were trained individually, once with 0%, and once with 20% of the 3000 cells being replaced by random generated images. All images were additionally altered by random rotation, vertical flipping and brightness change. In summary, each training iteration used 3000 different images that are differently augmented, which effectively reduces the risk of overfitting (as shown in Figure 3.20). Figure 3.20 shows the rolling median (window size=250 iterations) of the training (transparent lines) and validation accuracy (opaque lines) of MLP_1, MLP_2 and CNN_1. The rolling median was calculated using the accuracies of 250 successive iterations, which allows visualize the overall convergence and training behavior of a model. Specific single training iterations can indeed result in models with classification accuracies higher than the median and in practice a single model would have to be chosen and implemented into an image-based sorting device. When training MLP_1 with generated data (red lines in Figure 3.20), the rolling median of the validation accuracy increases slower compared to the training without generated data (opaque blue line is above opaque red line for iteration<175,000). For MLP_2 the opposite is true (opaque red line is mostly above opaque blue line in Figure 3.20) and for CNN_1 there is not much difference.

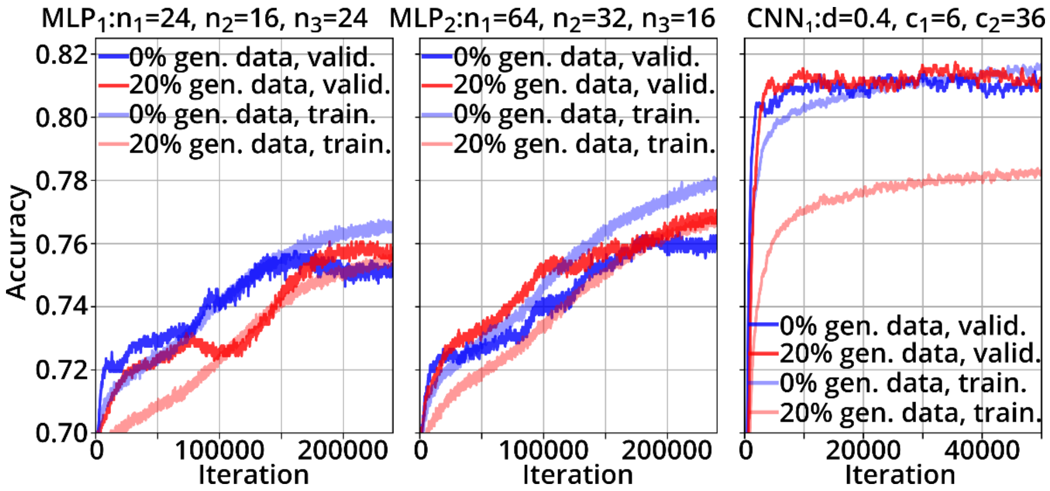


Figure 3.20 Training MLPs and CNN using 0% and 20% generated data

Plots show the rolling median of the accuracy (window size = 250 training iterations) for MLPs and CNN, which were trained without generated data (0% gen. data, blue) and with 20% generated data (red). MLPs were trained for 250,000 iterations, showing very slow but continuous improvement of the validation accuracy. The CNN was trained for only 50,000 iterations, showing a steep increase in accuracy for the first 10,000 iterations and a plateau afterwards. Overfitting (training accuracy>validation accuracy) appears for MLP_1 after 150,000 iterations, for MLP_2 after 50,000 iterations and for CNN_1 already after 30,000 iterations, when training without generated data (blue lines). When training with 20% generated data (red lines), the training accuracy is always below the validation accuracy, which shows that the generated data introduces variation, which represses overfitting.

Interestingly, the global maximum of the rolling median curve of the validation accuracy is slightly higher for each model when being trained with 20% generated data as summarized in Table 7. Furthermore, the maximum validation accuracy of a single model was higher, when training with 20% generated data. On the one hand one could find this surprising since the validation set does not contain generated data and therefore training on generated data would not introduce information that would help to learn classifying images in the validation set. On the other hand, generated data introduces variation into the training set, just like other image augmentation methods, which is known to often be helpful to train more robust models.

The optimized training routines caused a rise of the validation accuracy of MLP_1 from 0.7201 (see Figure 3.15 B) to 0.7854 (see Table 7) and of MLP_2 from 0.7241 (see Figure 3.15 B) to 0.7954 (see Table 7), which corresponds to a relative improvement by a factor of $\frac{0.7854}{0.7201} = 1.091$ and $\frac{0.7954}{0.7241} = 1.098$, respectively. The CNN does only benefit slightly from the optimized training routines, showing an increase in validation accuracy from 0.8139 (see Figure 3.16 C) to 0.8273 (see Table 7), which results in a relative improvement factor of $\frac{0.8273}{0.8139} = 1.016$. This behavior is expected since CNNs tend to deliver more robust and generalizable models, already for smaller datasets, compared to MLPs [117]. Since MLPs improved more than the CNN, their maximum validation accuracy is now close to the performance of the CNN. Table 7 shows the classification performance of MLP_1, MLP_2, and CNN_1 using the maximum of the curves of the rolling median of the validation accuracy (window size=250 training iterations) and also the maximum validation accuracy of a single iteration.

Model	MLP_1	MLP_1	MLP_2	MLP_2	CNN_1	CNN_1
% gen. data	0%	20%	0%	20%	0%	20%
Max. val. acc. rolling median	0.7575	0.7605	0.7635	0.7704	0.8144	0.8174
Max. val. acc. single iteration	0.7824	0.7854	0.7864	0.7954	0.8253	0.8273

Table 7 Validation accuracy with and without generated data

3.3.6. Tuning of final models and classification performance

During training, only the best performing models (in terms of validation accuracy) were saved. The models were trained to return P(GFP$^+$) – the probability that the input image is a GFP$^+$ cell. A perfect model would return P(GFP$^+$)=0.0 for each non-rod and P(GFP$^+$)=1.0 for each rod, but in reality, we obtain a broad distribution of probability values as shown in the histograms in Figure 3.21. The histograms were obtained by predicting the cells from training set 1 (see Table 5) using MLP$_1$ (top), MLP$_2$ (middle) and CNN$_1$ (bottom), which were trained without (0%, left histograms) and with 20% generated data (20%, right histograms). As expected, GFP$^+$ cells (green histograms) return larger P(GFP$^+$) values compared to GFP$^-$ cells (gray histograms), but there is a certain overlap of both distributions for all models.

The analysis routine and data is identical as used for Figure 3.11 (area gating), Figure 3.12 (random forest based on online features) and Figure 3.14 (random forest based on online and offline features), allowing to compare the methods. The overlaps of green and gray histograms are smaller for all DNNs (MLPs and CNN) as compared to all other methods (compare Figure 3.21 to Figure 3.11 A, Figure 3.12 A, and Figure 3.14 A), suggesting superior classification performance of these DNNs.

Usually, the threshold P(GFP$^+$)$_{thresh}$, upon which an event is predicted as GFP$^+$, is 0.5. By increasing P(GFP$^+$)$_{thresh}$ (indicated in upper left histogram in Figure 3.21 by a red line), the number of false positives within the target region decreases, resulting in an increase of the concentration of rods as shown by the plots in the middle in Figure 3.21. Simultaneously, the number of rods in the target region ("yield") decreases, as shown in the plots on the right in Figure 3.21. For all three DNNs, the concentration of rods increases until approximately P(GFP$^+$)$_{thresh}$≈0.9. Above 0.9, the yield drops to zero. MLPs trained with 20% generated data show improved performance in terms of rod-concentration and yield. This is not the case for CNN$_1$, but CNN$_1$ performs in general better than the MLPs. A concentration above 90% is obtained by MLP$_2$ for thresholds above 0.85 and by CNN$_1$ already for P(GFP$^+$)$_{thresh}$≥0.8. For the validation data and the testing data, the distributions might look different and one would maybe like to use a different threshold, but since P(GFP$^+$)$_{\text{thresh}}$ modulates the prediction, it belongs to the model parameters and has to be defined using the training data alone.

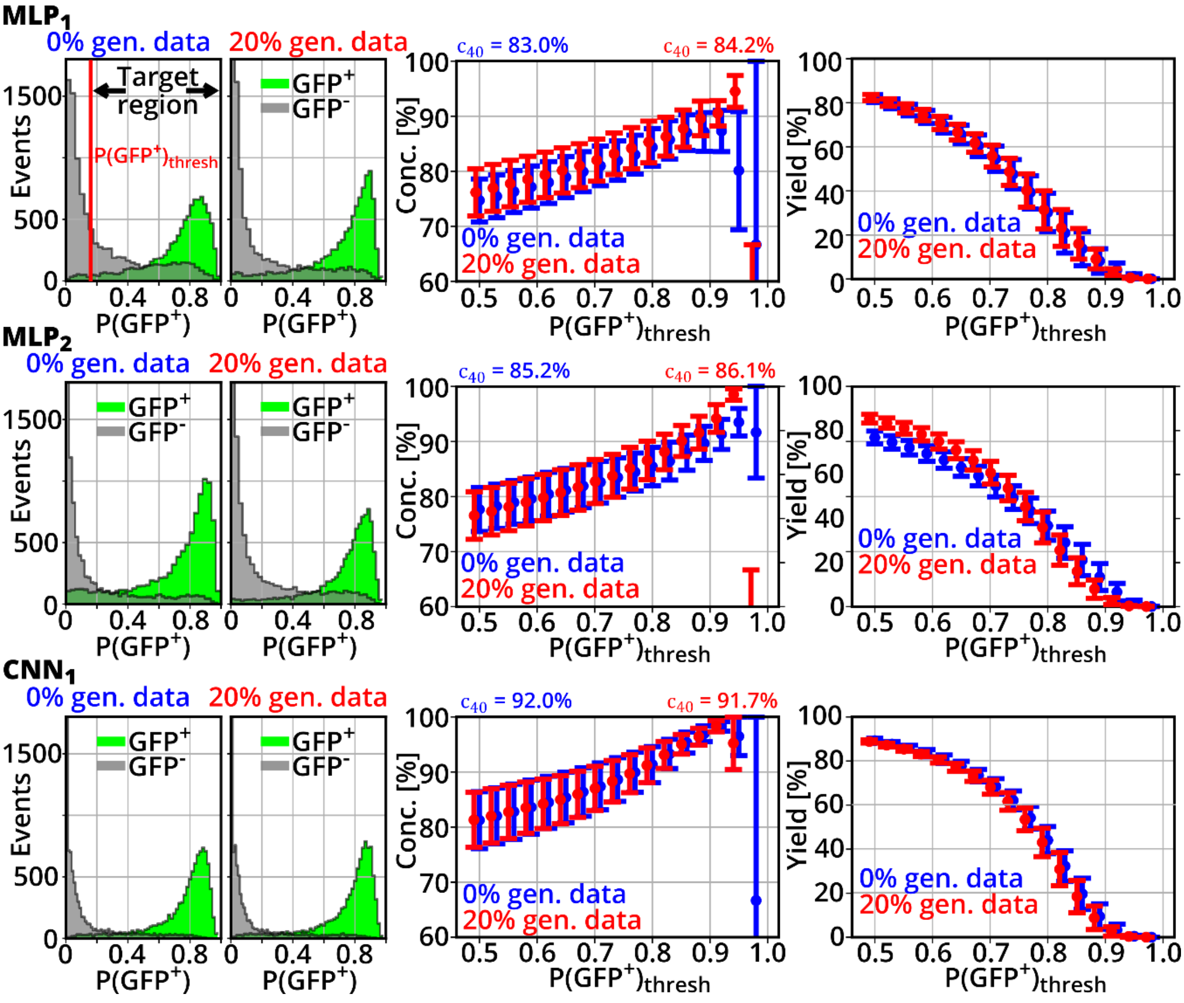


Figure 3.21 Performance of DNN based cell classification

Histograms show the probability distributions for GFP$^+$ (green) and GFP$^-$ (grey) cells resulting from training set 1 using MLP$_1$, MLP$_2$ and CNN$_1$, which were trained without generated data (0%, blue) and with 20% generated data (red), respectively. The red line indicates the threshold P(GFP$^+$)$_{\text{thresh}}$, which is the decision boundary between GFP$^+$ and GFP$^-$. The concentration and number of rods (yield) within the target region is dependent on the threshold as shown in the middle and right plots. The target concentrations and yields were determined for all three training sets individually and the plots show the resulting mean and standard error of the mean. This routine has been applied to models that were trained without generated data (0%, blue) and to models that were trained with 20% generated data (20%, red).

To allow comparison to the area gating method (see section 3.3.2) and random forest based classification (see section 3.3.3), the identical analysis-routine was applied, to obtain the confusion matrices: for each DNN, a certain P(GFP$^+$)$_{\text{thresh}}$ was determined, which delivers in average a yield of 40% on the training sets. This threshold is denoted as $P(GFP^+)_{thresh\ 40}$. The resulting confusion matrices when applying the models on the validation and test set are shown in Figure 3.22, which indicate that each DNN performs better on the validation set compared to the random forests (see Figure 3.14 B) or the area gating method (see Figure 3.11).

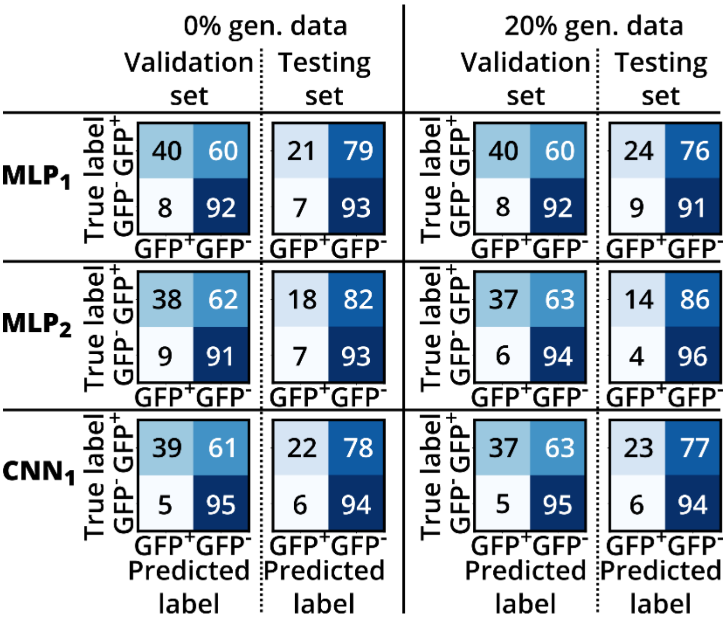


Figure 3.22 Confusion matrices for DNNs

Normalized confusion matrices show the classification performance of each DNN model. The adjusted classification threshold which results in a yield of 40% on the training set is used ($P(GFP^+)_{thresh\ 40}$).

An analysis of the concentration and yield in the target region for each model (MLP_1, MLP_2 and CNN_1) is shown in Figure 3.23. Besides the usual threshold of $P(GFP^+)_{thresh\ 40}$, also another threshold, which delivers a yield of 20% on the training sets (denoted as $P(GFP^+)_{thresh\ 20}$) is used in order to check, if higher concentrations could be achieved. As one would expect, the achievable concentration is higher for the validation data (dark gray bars) compared to the testing data (light gray bars) since the testing data corresponds to new biological replicates which might show slightly different phenotypes and therefore deviate from training and validation data. Furthermore, the testing data has been recorded several months after the training data and the measurement system was continuously used, adjusted and has altered (a different LED was implemented). Both, validation and testing dataset, contain data of three biological replicates, which were analyzed individually, allowing to compute a mean and standard error of the mean for concentration and yield, which is shown in Figure 3.23. The left plot in Figure 3.23 A shows the concentration of rods in the target when using $P(GFP^+)_{thresh\ 40}$. For the validation data the concentrations are above 80% for all MLPs and even above 85% for CNN_1. The concentrations for the testing set are in average approximately 8% lower than for the validation set. The right plot in Figure 3.23 A shows the same analysis, when using $P(GFP^+)_{thresh\ 20}$. Interestingly, MLP_2 reaches a concentration of 89.2% for the validation set, but at the same time, the yield drops by approximately 20% and 10% for the validation data and testing data, respectively (see Figure 3.23 B). So one can conclude that increasing $P(GFP^+)_{thresh}$ allows to increase the resulting concentration but at cost of a large decrease of the yield.

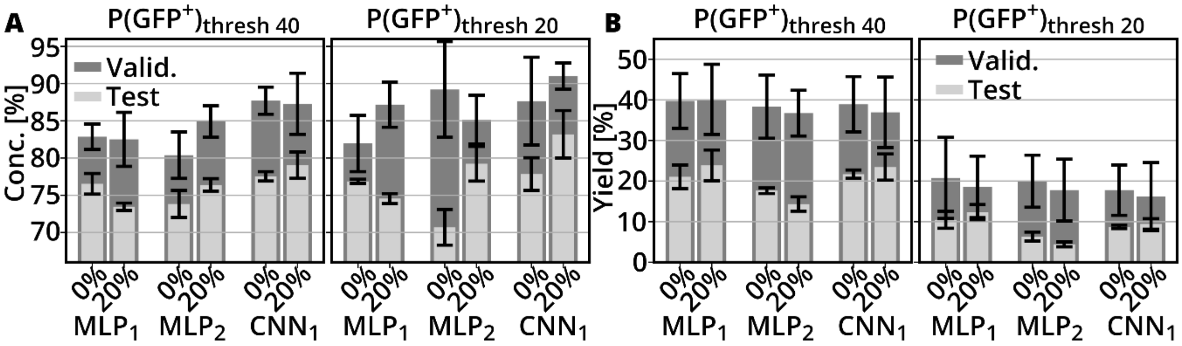


Figure 3.23 Performance of models on validation and testing set

(A) Barplots show the concentration of rods in the target region when the threshold $P(GFP^+)_{thresh}$ is adjusted such that a yield of 40% ($P(GFP^+)_{thresh\ 40}$, left) or 20% ($P(GFP^+)_{thresh\ 20}$, right) is achieved. Models were either trained using 0% or 20% generated data. Each model was applied to three validation and three testing datasets individually, resulting in a certain mean concentration and standard error of the mean, which is displayed by error bars.

(B) The corresponding yield for each case shown in A.

Table 8 summarizes c_{40} of each algorithm when being applied to the training and validation set:

Approach	c_{40} (training set)	c_{40} (validation set)
Area gating	$66.5\% \pm 3.8\%$	$69.20\% \pm 4.6\%$
Random forest (online features)	$75.7\% \pm 4.1\%$	$72.29\% \pm 4.7\%$
Random forest (online & offline features)	$83.5\% \pm 2.6\%$	$77.91\% \pm 1.8\%$
MLP_1 (0% gen. data)	$83.0\% \pm 3.5\%$	$82.9\% \pm 1.7\%$
MLP_1 (20% gen. data)	$84.2\% \pm 3.8\%$	$82.5\% \pm 3.6\%$
MLP_2 (0% gen. data)	$85.2\% \pm 3.5\%$	$80.4\% \pm 3.1\%$
MLP_2 (20% gen. data)	$86.1\% \pm 3.6\%$	$84.9\% \pm 2.1\%$
CNN_1 (0% gen. data)	$92.0\% \pm 3.1\%$	$87.7\% \pm 1.8\%$
CNN_1 (20% gen. data)	$91.7\% \pm 3.0\%$	$87.3\% \pm 4.1\%$

Table 8 Comparison of c_{40} for all classification methods

The training datasets were tightly restricted to very low (GFP^-) and very high (GFP^+) fluorescence intensities in order to avoid training on mislabeled cells (see section 3.3.1). In the following, the performance of the models is checked on unfiltered datasets, spanning the full range of fluorescence values. In the worst case, cells with moderate fluorescence levels would show a different phenotype and would be incorrectly classified by the models. The blue vertical histograms in Figure 3.24 show the distributions of GFP fluorescence expressions of two testing sets (without gating to fluorescence ranges as shown in section 3.3.1). Testing set 1 was captured at a higher laser power, which causes relatively higher fluorescence values compared to testing set 3. The blue horizontal histograms show the corresponding area distributions. Apparently, cells of testing set 1 tend to be slightly larger compared to testing set 3, which could be due to biological variation. The squared plots show scatterplot-contours at 95% (solid blue line) and 50% (dashed blue line) of the maximum event-density. Here, the initial data of both testing sets shows a population in high and low fluorescence intensity ranges. MLP_1 was used to predict which events are rods and the area and fluorescence distributions of the identified cells are shown by green histograms and contours. While the histogram of the fluorescence of the initial dataset (blue vertical histograms) showed two peaks, the distribution of the cells that are predicted to be rods has only one peak at high fluorescence intensity. Even though, there is a difference in cell size between training set 1 and 3, MLP_1 based selection of rods succeeds to shift the fluorescence distributions towards higher values for both datasets. This indicates that the model is robust for such differences in the phenotype.

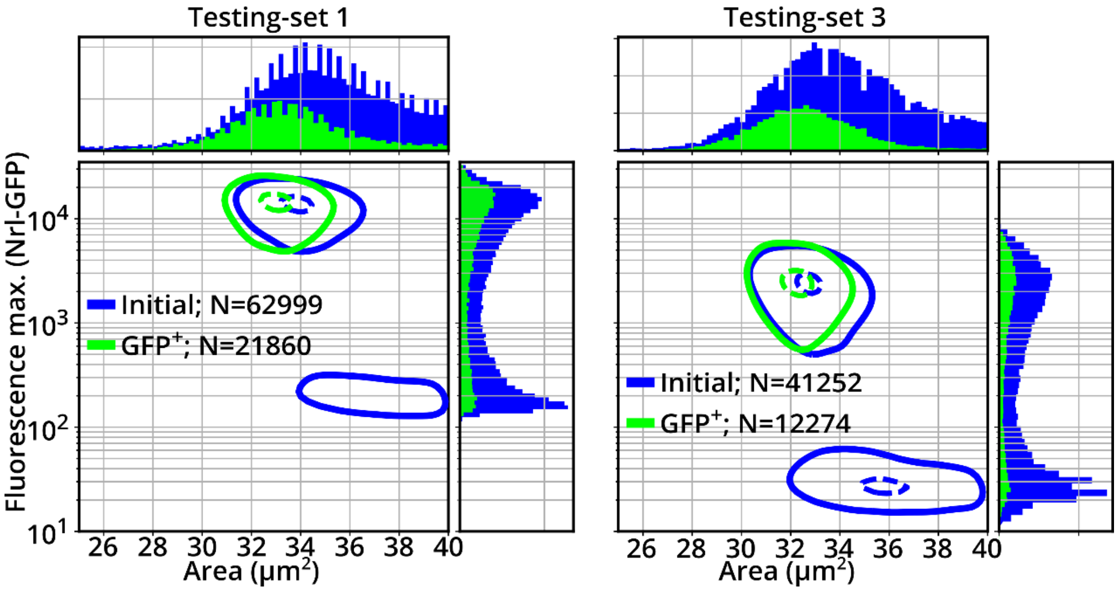


Figure 3.24 Rod-identification in full range of fluorescence values using MLP_1

The plots show the area (horizontal histograms) and fluorescence (vertical histograms) distributions of two testing sets. Squared plots between the histograms show the 95% (solid line) and 50% (dashed line) scatterplot-contour lines. The distribution of the initial data is shown in blue. MLP_1 was used to predict which events correspond to rods and the resulting populations are shown in green.

3.3.7. Visualization of model attention

Interpreting why a machine learning model returned a certain prediction is helpful to understand and optimize a model. While such interpretation is possible for machine learning models such as random forest, which use a set of pre-defined features, it is still a subject of current research to interpret the decision of a neural net. A simple approach to get an idea of the typical phenotype is to look for cells of each cell type that are very confidently and correctly classified. The upper row in Figure 3.25 A shows four rods, with the highest P(GFP$^+$) scores and similarly the lower row shows images of non-rods which have the lowest P(GFP$^+$) scores. Here, it seems the GFP$^+$ cells tend to have a thinner black border in the periphery and a brighter center compared to the GFP$^-$ cells. This pattern is also visible when averaging 10 cells with the highest or lowest P(GFP$^+$) scores as shown in Figure 3.25 B. A more involved approach is to create an image, which maximizes the probability for a certain class. This task is not straight forward since it requires to use the DNN in a reverse fashion, which is a one-to-many-mapping (one label could be caused by many different images). Figure 3.25 C shows the activation maps, which were produced using MLP_1 (trained using 20% generated data) and the "keras-vis" Python library [145]. Activation maps are generated images, which result in a high probability for a given class. Again, it appears that the generated image which maximizes the probability for a GFP^+ cell has a brighter center compared to the activation map for GFP^-.

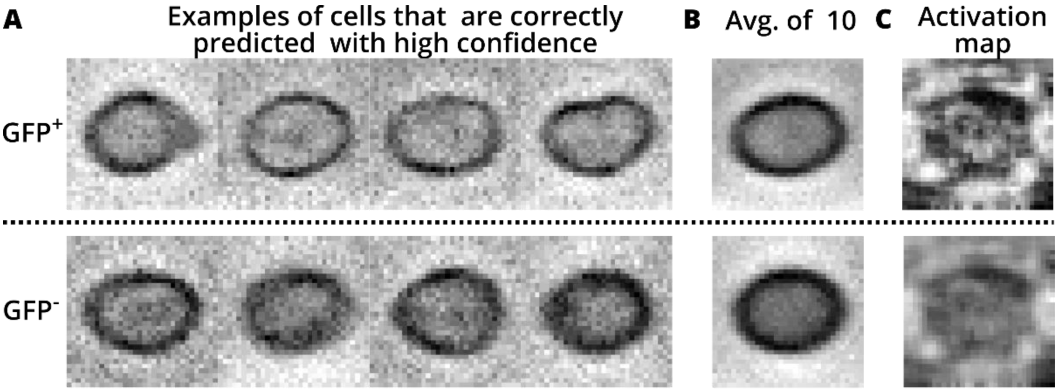

A
Examples of cells that are correctly predicted with high confidence
B
Avg. of 10
C
Activation map
GFP+
GFP-

Figure 3.25 Appearance of correctly predicted cells

(A) MLP_1, (trained using 20% generated data) was used to predict $P(GFP^+)$ for the images from training set 1. The images show examples of cells that are predicted correctly and with high confidence, which means $P(GFP^+)$ is very high for GFP^+ cells and very low for GFP^- cells. The GFP^+ cells seem to be slightly brighter in the center, and have a sharper, thinner dark ring on the outside compared to the GFP^- cells.

(B) The same phenomenon can be visualized by averaging 10 GFP^+ with the highest $P(GFP^+)$ and 10 GFP^- cells with the lowest $P(GFP^+)$.

(C) Activation maps show images created such that they cause the DNN to return maximized probabilities for a given cell type. The map which maximizes the probability for a GFP^+ cell shows more contrast between border and center of the cell compared to the map of for GFP^- event.

A distinct feature of RT-DC is the ability to measure a mechanical readout but it is not yet clear if that information is used by the DNNs. A computation of the elastic modulus would be linked to high error since cells cover only approximately 20% of the channel width (see section 2.3). Also, the usage of deformation or inertia ratio is problematic due to the significant size difference between GFP^+ and GFP^- cells. Therefore, I filtered the dataset for cells in a very narrow area range from 34 μm^2 to 36 μm^2 and determined $P(GFP^+)$ of these events. The scatterplots in Figure 3.26 show that there is a positive correlation between the inertia ratio and P(GFP$^+$), which indicates that the model tends to return higher P(GFP$^+$) values for more deformed cells.

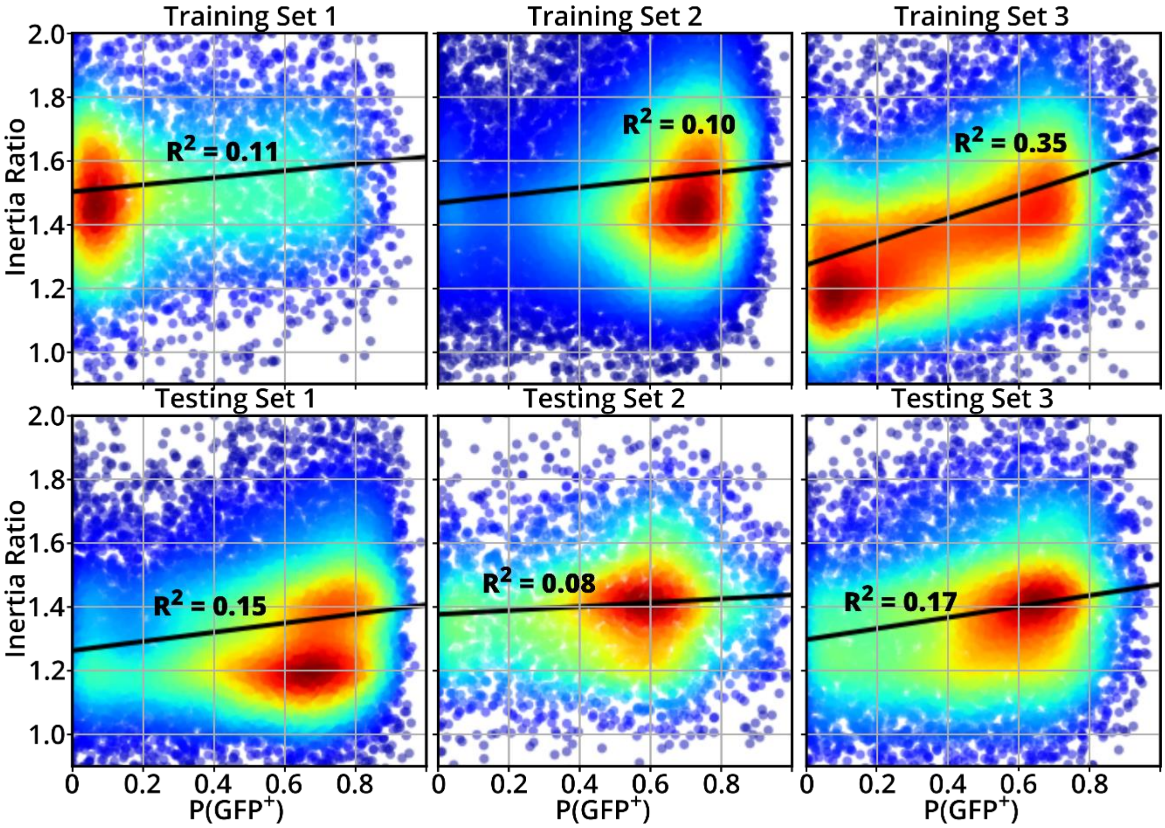


Figure 3.26 Correlation between DNN output and inertia ratio

The scatterplots show the probability, resulting from MLP$_2$ (probability that the event is a GFP$^+$ cell) and the inertia ratio for cells from three training and three testing datasets. Only cells in a range from 34 µm^2 to 36 µm^2 were used for that analysis to keep the influence of size differences small. In all cases there is a positive correlation, which is also visualized by a linear fit and quantified using the Pearson coefficient of correlation R^2.

3.3.8. Discussion

In this chapter, I showed different approaches to use machine learning for label-free image-based identification of rod precursor cells. Gating to a particular area range would be easiest since it already works in real-time in RT-DC (up to 1000 cells/s). Unfortunately, the expected concentration of rod precursors upon sorting is below 70%. Random forests allow considering more features for classification and would actually reach a final concentration c$_{40}$ of 83.5% (see Figure 3.14). Unfortunately, computing the required features takes approximately 8 ms, limiting the possible throughput to a maximum of 55 cells/s. The highest c$_{40}$ of 92.0% was achieved using a convolutional neural net, which required a computational time of 3ms (corresponds to a throughput of 150 cells/s). Despite a lower c_{40}, of 86.1% certain multilayer perceptron (MLP) architectures are the most promising algorithms for sorting because of a low computational time of approximately 100 µs (corresponds to a throughput of 10,000 cells/s). The given inference times were determined using an Intel® Core™ i7 processor, showing that standard PC hardware is sufficient to perform AI-based image classification at high-throughput.

Because of parameter sharing, CNNs typically generalize better than MLPs, resulting in a more robust classification performance upon rotation or translation of the objects the image [117]. In RT-DC, cells are aligned in the channel, and images are cropped such that the cell body is centered. Both processes reduce the degrees of freedom, which effectively also reduces the required computational complexity for image analysis. Likely for this reason, MLPs deliver acceptable performance for RT-DC data. MLP architectures are more prone to overfitting as compared to CNN architectures which perform the same number of multiplications. One approach to prevent overfitting is to use a larger dataset, which is often linked to high costs and effort. Therefore, I used image augmentation and data generation using a generative adversarial net and successfully trained MLPs and the CNN for tens of thousands of training iterations without overfitting. As a result, the validation accuracies of the MLPs increased considerably and reached almost the performance of the CNN. The contribution of generated data on the performance increase was assessed individually and it turned out that only MLP architectures did benefit slightly. The computational cost of data generation is quite high and the resulting images from GANs often simply look like the original image with noise added and different contrast. Since addition of noise and contrast changes are a simpler operations, they would be practical alternatives to using GANs.

The training dataset only consisted of three biological replicates and between the acquisition of training and testing set, the LED illumination assembly needed to be changed, which is not optimal as it changes the image properties. Nevertheless, the models achieved final rod concentrations between 74% and 79% on the testing set. By using more biological replicates for training, and keeping the system constant, one can expect to increase the accuracy and hence the theoretically achievable concentration to 87%. Even a concentration of 90% could be reached but at a reduction of yield, which in practice would require a longer time for sorting. Hence, using these algorithms for image-based sorting would allow enriching rod photoreceptors to similar concentrations like MACS ($87.7\% \pm 4.7\%$), which were already shown to be sufficient for photoreceptor transplantation [31].

A higher accuracy and concentration could theoretically be achieved using more complex neural nets, but this would require more powerful computational hardware to keep inference times at the same level. The screening of MLPs (see Figure 3.15), was limited to architectures with three hidden layers. One could extend this search by altering also the number of hidden layers, in order to find even better performing architectures. Alternatively, one could start to train a large network and subsequently identify nodes that are of low importance and remove those. Next, one would continue to train the reduced model and again remove unimportant nodes. The process is continued iteratively, until the validation accuracy falls below a given threshold. This approach is called “network pruning” and is subject of current research especially because quantification of the importance of nodes is not trivial [146]. The classification performance and robustness of models can also be expected to increase when improving the image quality. For example, by reducing the thickness of the $LiNbO_3$ substrate (currently 0.5 mm), the level of noise could be reduced. Furthermore, one could provide more information to the neural net, for example by increasing the framerate of the camera to capture multiple images of the same cell, which could then all be used for classification. Even more promising would be to add information of another physical (label-free) property, for example by simultaneously performing quantitative phase imaging (QPI) [51]. The resulting maps for refractive index or mass-density could be used in parallel to the bright-field image and neural nets could be designed to use all that information in parallel.

Currently, many companies and research institutes put a lot of effort into the acceleration of neural nets using dedicated hardware such as tensor processing units or field-programmable gate arrays (FPGAs). Furthermore, the optimization of software such as the Intel® Math Kernel Library for Deep Neural Networks results in a continuous increase of performance. Therefore, it is likely that deep CNNs will soon be feasible for real-time analysis on standard PCs as acceleration by a factor of 10 would already be sufficient.

A recent publication showed the application of FPGAs for image classification, but a relatively long inference time of 5.8 ms was reported and the authors assume an improvement when using a GPU instead [147]. For intelligent image activated cell sorting (iIACS), a GPU was used and inference times of 3 ms were reported [63]. While iIACS actually allows image-based sorting using DNNs, running this setup requires a team of specialized staff [148]. In contrast, the approach suggested in this section could be used on the existing soRT-FDC setup, which can be operated by a single person.

Despite the drawbacks of the current setup (MLP only, image distortion due to sorting chip), this section shows that remarkable classification performance is achievable using standard computational hardware. Therefore, image-based sorting using the existing setup is feasible and such a technology would not only be interesting for of label-free sorting of rod-progenitor cells but for example also for label-free sorting of induced or embryonic pluripotent stem cells or certain subpopulations of blood cells. Recent publications showed that discrimination of granulo-monocytes [15], prediction of differentiation lineages [149] and distinguishing B and T cells [150] can be performed in a label-free fashion based on bright-field images. In general, the specificity of established fluorescent labels can be employed to create labeled datasets using RT-FDC which can then be used to train DNNs which perform cell classification based on bright-field images. Besides image-based sorting, real-time classification by a DNN could be used as a diagnostic tool for example to detect malignant cells. These examples indicate the potential and wide applicability of image-based label-free cell identification and sorting.

Unfortunately, the need for programming skills excludes a lot of users from applying deep learning for image classification. Image processing software such as cell profiler [151] or ImageJ [152] did not yet implement such tools. Therefore, at the beginning of the next section, I will introduce software I developed that allows non-programmers to train, evaluate and apply deep learning for image classification. This software provides an easy access to the concepts introduced in section 3.3 and offers the possibility to convert the resulting DNNs to a format that can be used by soRT-FDC.

3.4. Software tools to train and apply deep neural nets for sorting

In this section, I present software, which allows even non-programmers to train deep neural nets for image classification and to convert the final models to a format accepted by the software, controlling the sorting device (called “Sorting Software”). Furthermore, the working principles of the Sorting Software (courtesy of Martin Nötzel) are explained.

3.4.1. AIDeveloper

Installing a programming language including all required libraries for deep learning and their dependencies can be a time consuming and complicated task. Furthermore, due to the rapid development in the field of deep learning, software libraries quickly change, loose compatibility to other libraries or are even stopped being supported at all. A prominent example is Theano [153], which used to be one of the most popular Python libraries for deep learning, until a major support was stopped in September 2017 since other libraries such as TensorFlow [154] from Google or Pytorch [155] from Facebook started to offer the same or even more functionality. Code, which runs on one machine does not necessary work right away on another machine and I must admit that I broke my own code several times upon updates of Python libraries. Such problems could delay or even inhibit sorting experiments if a particular model needs to be trained and would limit the accessibility of image-based sorting to people with sufficient programming skills and persistency.

Therefore, I created AIDeveloper (AID), a software suite which allows non-programmers to train, evaluate and apply deep learning for image classification. The software was frozen to a standalone executable which runs identically on each Windows 7 and 10 PC. AID guides the user through the entire working pipeline: starting from loading data, assembling training and validation set, setting sensible image augmentation parameters towards choice and training of a neural net. During the training process, model metrics such as accuracy and loss are displayed in real-time. Furthermore, hyperparameters (image augmentation parameters, learning rate, dropout rates,…) can be changed during the training process and all details are documented in an excel file. Support for

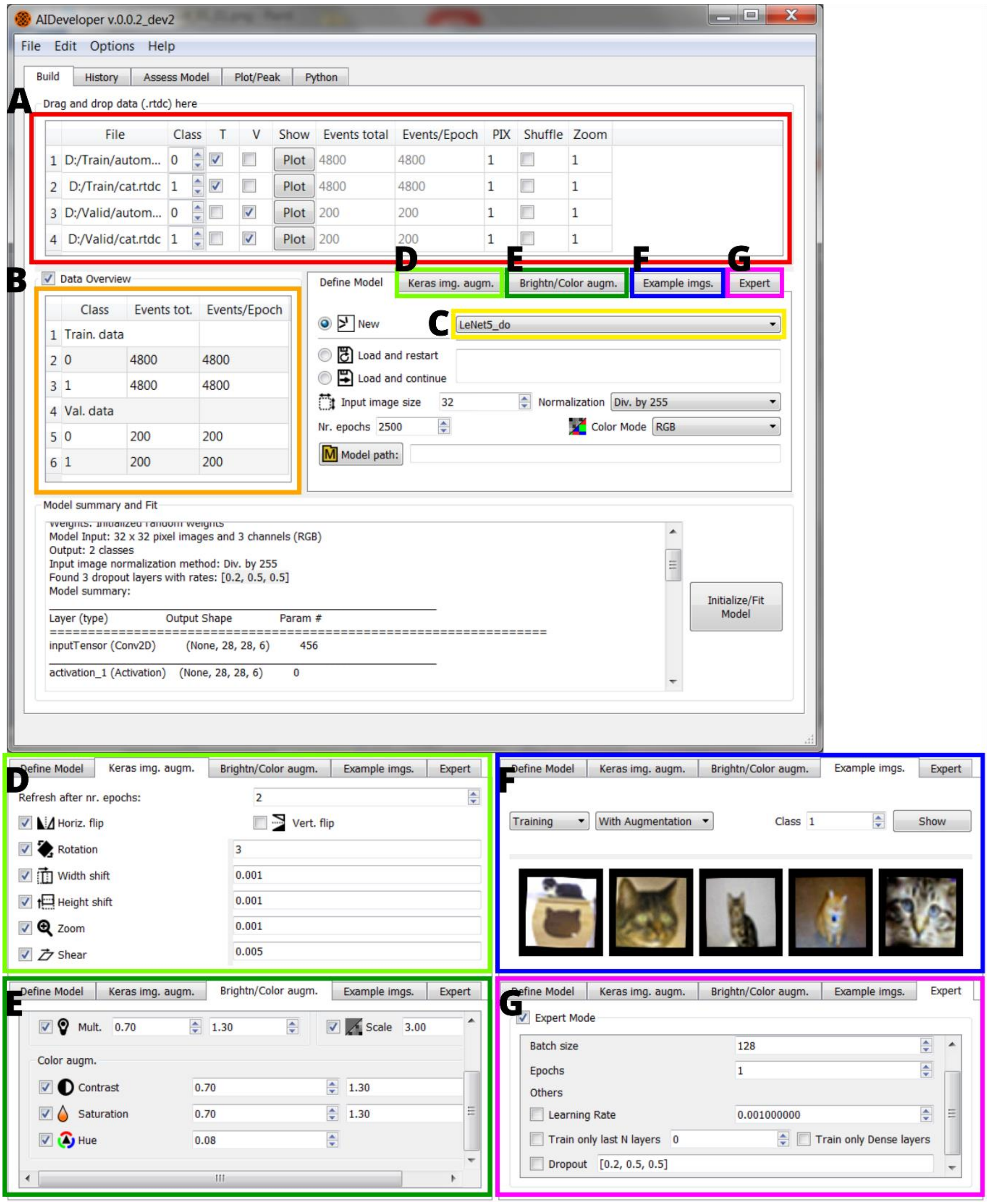
MLPs was implemented into the Sorting Software (courtesy of Martin Nötzel), and AID allows to convert models to the required format.

File loading

Figure 3.27 shows a screenshot of the main window of AID. Data is loaded into AID by dragging and dropping rtdc-files into a dedicated table (indicated by a red rectangle A in Figure 3.27). Alternatively, also folders containing images (.jpeg, .png, .bmp, .tif, .eps, .gif, .ico and many more) can be dropped into this region. The contents of each dropped folder would then automatically be converted to individual rtdc-files. The rtdc file format is based on the hdf5 format which allows for fast data loading. ShapeIn (Version ≥ 2.0.6, Zellmechanik Dresden GmbH, Germany) writes this file format and the open-source software ShapeOut [83] allows to post-process RT-DC experiment-files and to export data to rtdc-format. To showcase the functionality of AID, images of automobiles and cats from the CIFAR10 dataset are used [156].

AID lists the loaded files in a table (indicated by red rectangle (A) in Figure 3.27). The file location is displayed and the number of contained images. Upon double-clicking on the filename, an example image of the dataset is shown. Corresponding fields in the table allow users to set the class of each file and whether the data should be used for training or validation. Furthermore, the number of images that are used in each training iteration (epoch) are customizable. Optionally, a zooming factor can be applied, which is useful when images were captured using different magnification levels.

An overview box (orange box (B) in Figure 3.27) shows the total number of images in each class in training (T) and validation (V) set which allows for a quick sanity check and is especially helpful, when working with many files.



Drag and drop data (.rtdc) here

	File	Class	T	V	Show	Events total	Events/Epoch	PIX	Shuffle	Zoom
1	D:/Train/autom...	0	☑	☐	Plot	4800	4800	1	☐	1
2	D:/Train/cat.rtdc	1	☑	☐	Plot	4800	4800	1	☐	1
3	D:/Valid/autom...	0	☐	☑	Plot	200	200	1	☐	1
4	D:/Valid/cat.rtdc	1	☐	☑	Plot	200	200	1	☐	1

☑ Data Overview

	Class	Events tot.	Events/Epoch
1	Train. data		
2	0	4800	4800
3	1	4800	4800
4	Val. data		
5	0	200	200
6	1	200	200

● New LeNet5_do
○ Load and restart
○ Load and continue
Input image size 32
Normalization Div. by 255
Nr. epochs 2500
Color Mode RGB
Model path:

Model summary and Fit

```
Weights: Initialized random weights
Model Input: 32 x 32 pixel images and 3 channels (RGB)
Output: 2 classes
Input image normalization method: Div. by 255
Found 3 dropout layers with rates: [0.2, 0.5, 0.5]
Model summary:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputTensor (Conv2D)         (None, 28, 28, 6)         456
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 6)         0
```

Refresh after nr. epochs: 2
☑ Horiz. flip
☐ Vert. flip
☑ Rotation 3
☑ Width shift 0.001
☑ Height shift 0.001
☑ Zoom 0.001
☑ Shear 0.005

Class 1

☑ Mult. 0.70 1.30
☑ Scale 3.00
Color augm.
☑ Contrast 0.70 1.30
☑ Saturation 0.70 1.30
☑ Hue 0.08

☑ Expert Mode
Batch size 128
Epochs 1
Others
☐ Learning Rate 0.001000000
☐ Train only last N layers 0
☐ Train only Dense layers
☐ Dropout [0.2, 0.5, 0.5]

Figure 3.27 Main window of AID

Screenshots show the main window of AID. Colored rectangles indicate units with specific functions. Red rectangle (A): Region to load data by drag and drop. Table shows loaded data. Orange rectangle (B): Summary of the number of images in each class. Yellow rectangle (C): Menu to choose a neural net architecture. Light and dark green rectangles (D, E): Menus for image augmentation methods. Blue rectangle (F): Options to show example images with and without augmentation. Magenta rectangle (G): Access to hyperparameters such as batch size, epochs, learning rate and dropout rates.

Model definition

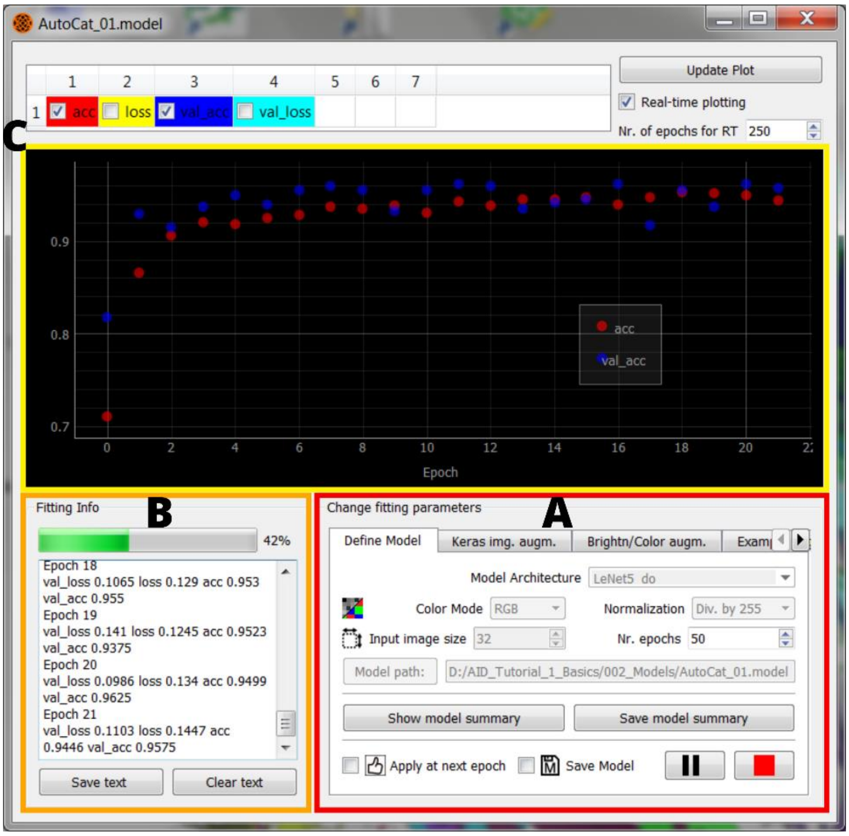
The yellow rectangle (C) in Figure 3.27 indicates a menu, which allows choosing a neural net architecture. Seven different multilayer perceptron (MLP) architectures, including MLP_1 (MLP-24-16-24) and MLP_2 (MLP-64-32-16) as well as 23 different convolutional neural nets of a wide range of complexities, are implemented. All neural nets are defined in a Python script (model_zoo.py), allowing customization and adding of models. Furthermore, loading of previously trained models into AID to continue or restart the training process is supported, allowing to re-use ("transfer learning" [157]) and share models. User input boxes for setting the input image size and color depth (grayscale or RGB) are available and the neural net is built accordingly. To obtain images of the requested image size, AID performs center cropping, using the centroid, which is always recorded for RT-(F)DC experiments (c_x and c_y). Optionally, one of the following data normalization methods can be chosen in a drop-down menu: division by 255, standard scaling using the mean and standard deviation of each individual image, standard scaling using the mean and standard deviation of the entire training set. Each of those normalization techniques was also implemented into the Sorting Software (courtesy of Martin Nötzel).

Image augmentation

The light-green (D) and dark-green (E) boxes in Figure 3.27 indicate tabs which allow defining parameters for image augmentation. The following augmentation methods are available: vertical and horizontal flipping, image rotation, width shift, height shift, zooming, shearing, additive and multiplicative brightness change, addition of Gaussian noise, Gaussian blurring, average blurring, motion blurring, as well as alteration of contrast, saturation (only for RGB images) and hue (only for RGB images). In the tab, indicated by a blue box (F) in Figure 3.27 random example images are shown, visualizing the effect of the chosen image augmentation parameters. The last tab (magenta box (G) in Figure 3.27) provides options to change learning rate, dropout rates (if applicable for the chosen neural net) and the trainability status of individual layers of the neural net. The last option is especially interesting for the application of transfer learning (loading a pre-trained neural net to optimize it for a different classification task) as it allows to unfreeze single layers when optimizing the model.

Starting and monitoring the training process

When all parameters are set, the training process is started upon the push of a button. The training process is carried out in a separate processing thread and a dedicated popup window allows interacting with it. The window shows information about the chosen model (see red box (A) in Figure 3.28) and provides the same options as the main screen (see Figure 3.27) allowing to adjust all hyperparameters during the training process. A text-box (indicated by orange rectangle (B) in Figure 3.28) displays metrics (such as accuracy and loss) of completed training iterations. Furthermore, all metrics are available for real-time plotting (indicated by a yellow rectangle (C) in Figure 3.28). Real-time plotting allows for example to quickly spot overfitting, which could immediately be countered by increasing image augmentation (e.g. by adding more Gaussian noise). When the training process is started, an excel file is created, listing information about files used, properties of the PC-system and all chosen parameters. Each change of parameters during the training process is documented in the excel file as well. After each training iteration, the model is automatically saved if the validation accuracy or validation loss reached a new maximum or minimum, respectively. The metrics (such as accuracy and loss) are also saved to the excel file for each training iteration.



1 ☑ acc ☐ loss ☑ val_acc ☐ val_loss

☑ Real-time plotting

Nr. of epochs for RT 250

42%

```
Epoch 18
val_loss 0.1065 loss 0.129 acc 0.953
val_acc 0.955
Epoch 19
val_loss 0.141 loss 0.1245 acc 0.9523
val_acc 0.9375
Epoch 20
val_loss 0.0986 loss 0.134 acc 0.9499
val_acc 0.9625
Epoch 21
val_loss 0.1103 loss 0.1447 acc
0.9446 val_acc 0.9575
```

Model Architecture LeNet5 do

Color Mode RGB

Normalization Div. by 255

Input image size 32

Nr. epochs 50

Model path: D:/AID_Tutorial_1_Basics/002_Models/AutoCat_01.model

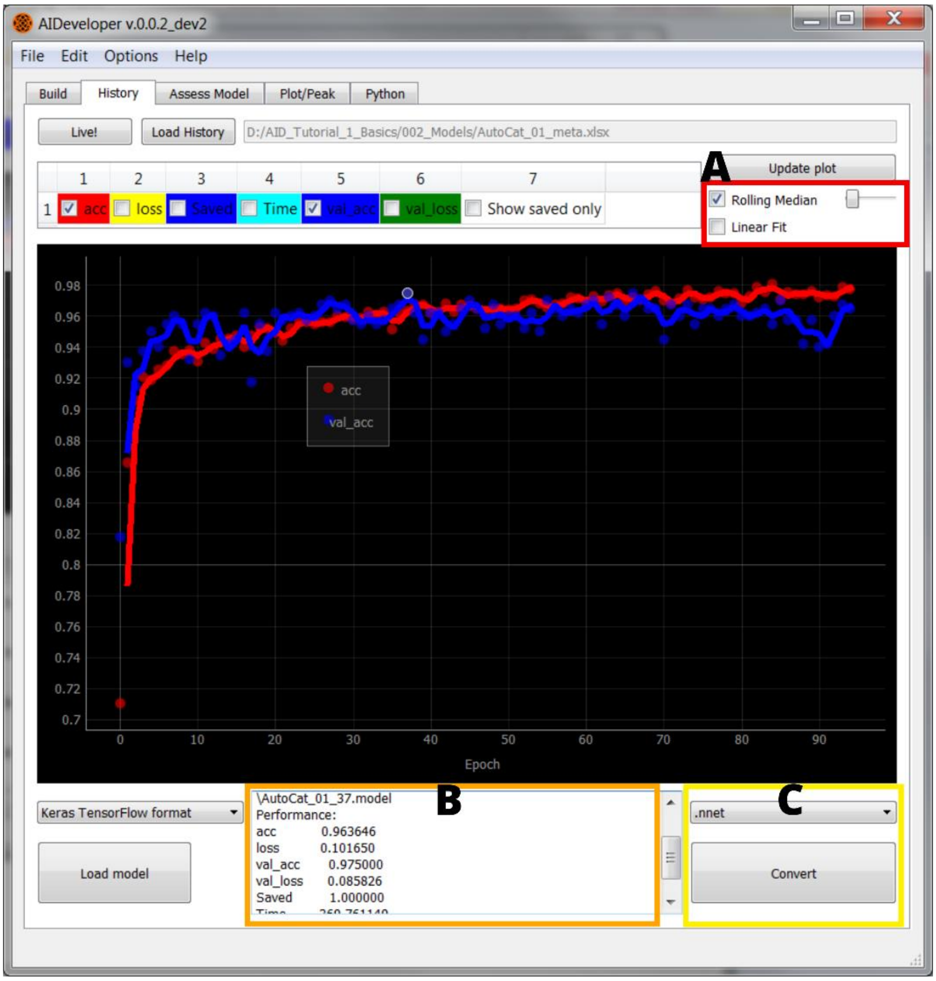
☐ Apply at next epoch ☐ Save Model

Figure 3.28 Training window of AID

During training, a separate window shows information about the model (highlighted by red rectangle, A) and allows to change hyperparameters. After each training iteration, information is displayed in a textbox (highlighted by orange rectangle, B). Model metrics (e.g. accuracy) are plotted in real-time (highlighted by yellow rectangle, C).

Reviewing a training history and preparing a model for the Sorting Software

A dedicated tab (see Figure 3.29) provides tools for loading and displaying metrics from a previous training session. Additionally, a rolling median and a linear fit can be added (see red rectangle (A) in Figure 3.29), which could for example help to figure out trends. Further information about a model is shown after clicking on the respective dot in the plot (see orange rectangle (B) in Figure 3.29). Furthermore, tools to convert the model to other formats such as protocol buffer (.pb), onnx (.onnx) [158] and to the format required by the Sorting Software (.nnet) are provided (see yellow rectangle (C) in Figure 3.29).



D:/AID_Tutorial_1_Basics/002_Models/AutoCat_01_meta.xlsx

1 ☑ acc ☐ loss ☐ Saved ☐ Time ☑ val_acc ☐ val_loss ☐ Show saved only

☑ Rolling Median

☐ Linear Fit

Keras TensorFlow format

```
\AutoCat_01_37.model
Performance:
acc        0.963646
loss       0.101650
val_acc     0.975000
val_loss    0.085826
Saved       1.000000
```

.nnet

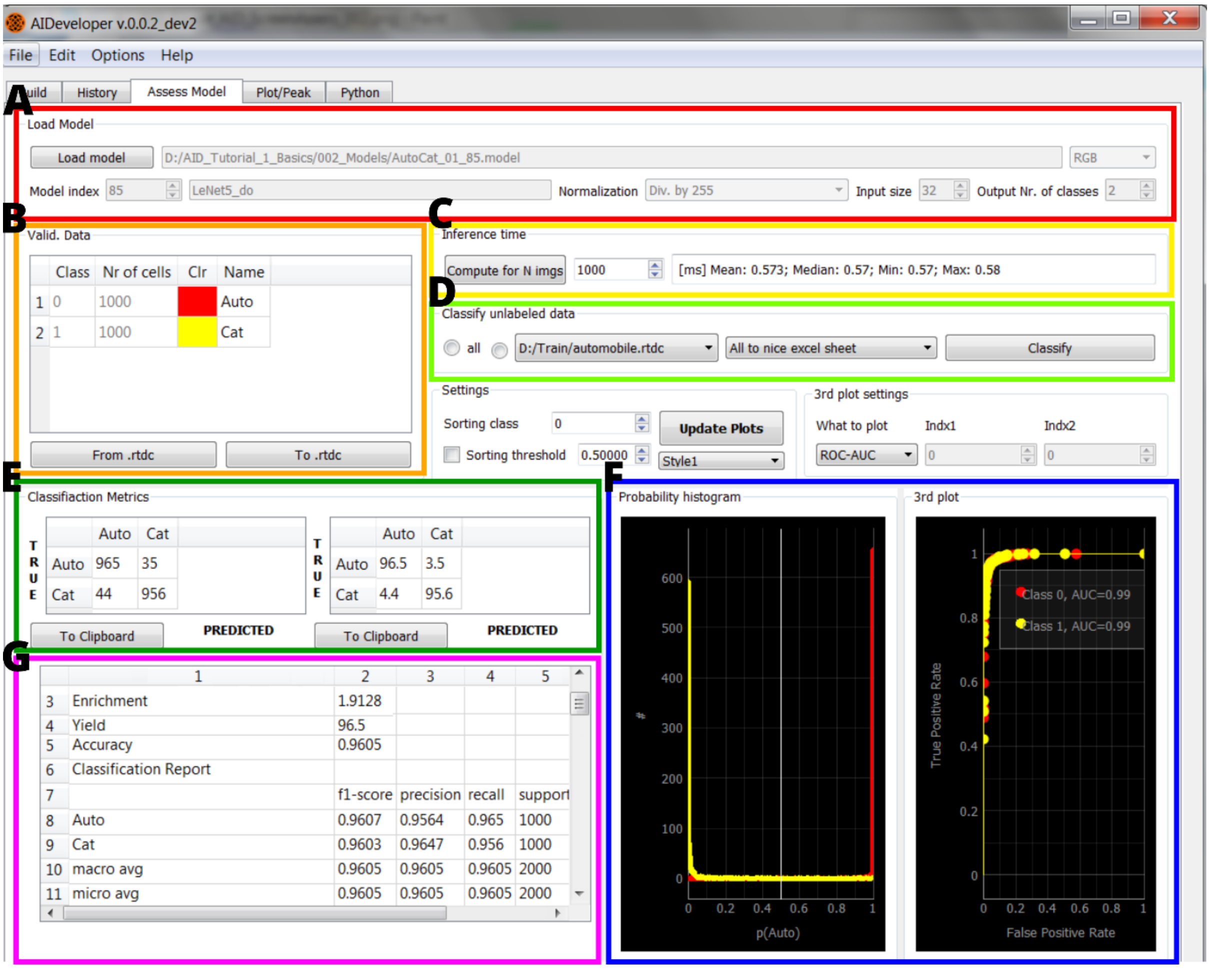
Figure 3.29 Reviewing a training history in AID

AID allows loading and displaying the history of a previous training session. Optionally, a rolling median and linear fit can be added to the plot (see red rectangle, A). Upon clicking on a dot, additional information about the corresponding model is displayed (see orange rectangle, B). Functions for converting models to other formats are provided (see yellow rectangle, C).

In-depth assessment of individual models

A dedicated tab (see Figure 3.30) provides tools to load a model (see red rectangle (A) in Figure 3.30) as well as data (see orange rectangle (B) in Figure 3.30) and to analyze the performance of the model. Tools to determine the inference time (processing time to predict the class of a single image) (see yellow rectangle (C) in Figure 3.30) as well as classification of unlabeled data (see light-green rectangle (D) in Figure 3.30) are provided. Interactive confusion matrices (see dark-green E rectangle in Figure 3.30) visualize the decision of the model for each class. By double-clicking on a confusion matrix item, the corresponding images can either be shown or saved to an rtdc-file.

Showing correctly and incorrectly classified images can help to figure out issues of the model. Standard plots such as receiver operating characteristic (ROC) and Precision-Recall curve [159] as well as sorting specific plots such as enrichment vs. sorting threshold (p_{thresh}) or yield vs. p_{thresh} are accessible (see blue (F) and magenta (G) rectangles in Figure 3.30). Export of each image or plot to.png and vector graphics file (.svg) is supported.



Load Model

Load model: D:/AID_Tutorial_1_Basics/002_Models/AutoCat_01_85.model; RGB

Model index: 85; LeNet5_do; Normalization: Div. by 255; Input size: 32; Output Nr. of classes: 2

Valid. Data

	Class	Nr of cells	Clr	Name
1	0	1000	🟥	Auto
2	1	1000	🟨	Cat

Inference time

Compute for N imgs: 1000 — [ms] Mean: 0.573; Median: 0.57; Min: 0.57; Max: 0.58

Classify unlabeled data

○ all ○ D:/Train/automobile.rtdc; All to nice excel sheet

Settings

Sorting class: 0

☐ Sorting threshold: 0.50000; Style1

3rd plot settings

What to plot: ROC-AUC; Indx1: 0; Indx2: 0

Classifiaction Metrics

	Auto	Cat
Auto	965	35
Cat	44	956

	Auto	Cat
Auto	96.5	3.5
Cat	4.4	95.6

	1	2	3	4	5
3	Enrichment	1.9128			
4	Yield	96.5			
5	Accuracy	0.9605			
6	Classification Report				
7		f1-score	precision	recall	support
8	Auto	0.9607	0.9564	0.965	1000
9	Cat	0.9603	0.9647	0.956	1000
10	macro avg	0.9605	0.9605	0.9605	2000
11	micro avg	0.9605	0.9605	0.9605	2000

Figure 3.30 Assessing a trained model in AID

Screenshot shows the tab in AID, which allows to load a trained model and data to assess the performance of the model.

Implementation details and availability of AIDeveloper

AID was programmed using Python 3.5. For deep learning, the Python packages Keras (version 2.2.2) [160] and TensorFlow (version 1.10) [154] were leveraged. The graphical user interface was created using PyQt (version 5.9.2, Riverbank Computing, United Kingdom).

PyQtGraph (version 0.10) was used for plotting and the standalone executable was generated using PyInstaller (version 3.5). AID is open-source software which is available as standalone executable and script: *https://github.com/maikherbig/AIDeveloper*. The standalone executable of AID currently leverages CPU implementations of TensorFlow to provide compatibility to many PC-systems. Running AID from script allows using individual Python installations with GPU support.

3.4.2. Sorting Software

The software to control sorting was written by Martin Nötzel. For the sake of completeness, I want to describe here the parts, which are important for image-based sorting using DNNs. The software is written in C++ and OpenCV [68] is used for image processing. Frames are continuously retrieved from the high-speed camera at 3000 frames per second. A background image is obtained by computing a rolling average of the last 100 frames, which is subtracted from each subsequent frame. After background subtraction, thresholding is applied to binarize the image. Next, dilation and eroding operations are applied to finally obtain smooth contours using a contour finding algorithm [69]. Optionally, computation is stopped at this point if the number of found contours exceeds a defined number. For sorting, it is advantageous to have only a single object in the region of interest to avoid sorting cells which are in the proximity of a target cell (see Figure 3.31). Furthermore, the time difference between the current and last captured event is tracked which can be used to prohibit sorting when events appeared too short after each other. Using this option one can avoid accidental sorting of multiple cells, which is especially helpful when working with samples that tend to form clusters (such as retina).

The contour of each object is used to compute the bounding box, returning the length, height and aspect ratio of the object. Computation is only continued if the object meets certain user defined criteria (min. and max. length, height and aspect ratio), which allows to stop computation for example when debris (small) or a red blood cell (high aspect ratio) is captured. Next, the original image (not the binary one) is cropped to a user defined region around the middle of the bounding box. The cropped image shows the object in the middle and has dimensions matching the requirements of the chosen neural net. The C++ library keras2cpp [161] is leveraged to forward the image through the trained neural net, which returns the probability for each class. A sorting pulse is triggered when the probability exceeds a defined threshold for a defined class. To decrease the computational time, saving of images and parameters is omitted entirely during image-based sorting.

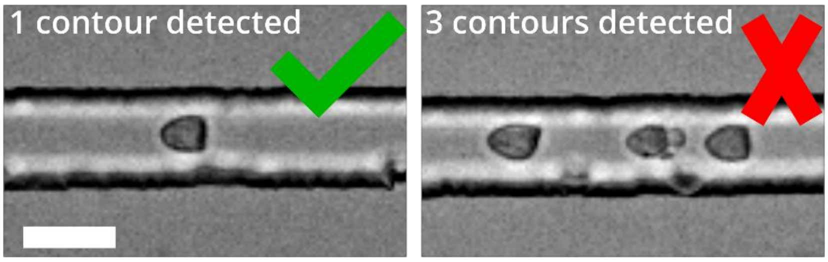


Figure 3.31 Preventing accidental sorting of multiple cells

Sorting pulses are omitted if multiple contours are detected within the region of interest. Scale bar: 20µm.

3.4.3. Discussion

Before image-based sorting can be performed, a suitable DNN needs to be trained which can be a substantial challenge, as programming skills are required. The most popular programming language for deep learning is Python and a large open source community drives a rapid software development, resulting in quickly improving software libraries, but also in a risk of losing compatibility to older code and a need of continuous software maintenance. Therefore, I developed AID, a software with graphical user interface, which allows to perform all required steps to obtain a model that can be used for soRT-FDC. Since AID was embedded into a standalone executable, it runs identically on each Windows 7 and 10 PC, allowing for reproducible analyses. AID provides several DNN architectures (including MLP_1 and MLP_2) which can be extended and customized. Methods for image augmentation are implemented and their effect is visualized by example images. Hyperparameters such as image augmentation parameters can be changed during the training process and the effect on metrics such as accuracy and validation accuracy is immediately displayed by interactive plots. AID eases the applicability of DNNs for image-based sorting using soRT-FDC because it provides tools to convert final models to a format that is accepted by the Sorting Software.

Other tools for image processing using machine learning are “Zen Intellesis” from Zeiss and ilastik [162], but both programs use fixed (non-trainable) DNNs for feature extraction and only support training of Random forests for image segmentation. The more popular image analysis tools Fiji [163] and ImageJ [152] offer only very limited support for neural nets. DIGITS™ from NVIDIA® provides a GUI and allows to train DNNs, but only very few DNNs are available, the input dimensions are fixed and access to optimization of hyperparameters is limited. The most complete GUI based program I found is “Deep learning studio” (DLS) from DeepCognition, which provides a flexible solution, allowing to train DNN models for many use cases (e.g. image classification, image segmentation, and natural language processing). In contrast, AID is just optimized for image classification due to its intention for image-based sorting. Unfortunately, DLS does not provide solutions to handle unbalanced datasets, which are quite common in biology (for example red blood cells are orders of magnitudes more abundant in blood than white blood cells). DLS supports training of models on certain GPUs which is currently not supported by AID (at least not in the standalone executable). Despite limitation to CPU power, training of models in AID is more than sufficiently fast, for example training MLP_1 and MLP_2 for a single iteration using 200,000 images (grayscale, 32x32 pixels) takes 3.6s and 4.4s, respectively (on an Intel® Core™ i7-4810MQ @ 2.80 GHz). More computational time (9s) is actually spent to perform affine image augmentations (random rotation, shift, zoom, shear and flip). Therefore, GPU support only becomes attractive when training of larger neural nets is proposed, but those are not yet of interest for soRT-FDC due to long inference times.

In conclusion, AID helps to accelerate and standardize the process of DNN training. Hence, the time between the first RT-FDC measurement of a sample and the image-based sorting of such a sample using soRT-FDC is shortened. AID allows everyone to use deep learning methods and train DNNs, which extends accessibility of DNN based image analysis and image-based sorting also to non-programmers. The combination of AID and basic MLP architectures allows for very fast training of models. Measuring a sample, training a neural net, and sorting using that neural net, is a routine that can be conducted by one person within a single day. To show that, the next section presents two examples of sorting experiments, where AID was used to perform training of the DNN.

3.5. Sorting experiments

While the scope of this thesis is the development of real-time analysis methods for image-based identification of cells, this section goes beyond that goal, showing successful sorting experiments. Gained knowledge about well performing deep neural nets (see section 3.3.4), AID (see section 3.4.1), the specific sorting-chip design (see 2.1.2), and the Sorting Software (see section 3.4.2) are employed. First, sorting of rod precursor cells from an Nrl-GFP retina sample (P04) is presented. Those cells originate from tissue, and chemical dissociation is required to obtain single cells. Furthermore, application of soRT-FDC to naturally suspended cells is shown by sorting neutrophils from human blood.

3.5.1. Sorting of rod precursor cells

Label-free identification of rod precursor cells in heterogeneous retina samples is one main motivation of this thesis. This section shows a practical sorting experiment in which rod precursor cells were successfully enriched.

Dataset assembly

Dissociated retina cells were resuspended in measurement buffer for RT-DC to a concentration of 20 million cells/ml (prepared by Karen Teßmer). The sample was loaded into a sorting chip (chip design is shown in 2.1.2) and RT-FDC, together with ShapeIn were used to capture training data. Normally a 40x magnification is used during RT-FDC experiments (see Figure 2.1). For sorting-experiments it is currently essential to use a 20x magnification to have a wider field of view, allowing to supervise the sorting-region. Therefore, training data was captured using 20x magnification. The resulting data shows the typical spread of GFP expressions, which was also seen for previous experiments, performed using 40x magnification (see Figure 3.9). ShapeOut [83] was employed to gate for events of the small GFP$^+$ and small GFP$^-$ fraction (similar as shown in Figure 3.10). A region with medium GFP expression was omitted since it is difficult to assign the events either to GFP$^+$ or GFP$^-$ class, creating a slight unbalance in the dataset. To obtain a balanced validation set, 2000 cells of each class were randomly selected (see Table 9). The numbers of cells for each class in training and validation set are shown in Table 9.

	Training	Validation	Balanced validation
Small GFP$^+$	2471	2048	2000
Small GFP$^-$	4512	4434	2000

Table 9 Training and validation data of retina cells

Training of MLP for rod identification using AID

The balanced validation set was loaded into AID (see section 3.4.1) and an MLP with 24, 16 and 24 nodes in the first, second and third hidden layer was trained (MLP$_1$). The performance of the resulting model, when using a threshold P(GFP$^+$)$_{thresh}$ of 0.6 is shown in the confusion matrix in Figure 3.32. In the balanced validation set, the initial concentration of GFP$^+$ cells is 50% and sorting for GFP$^+$ cells would theoretically result in a target concentration of $c_{GFP^+} = \frac{1238}{1238+490} \cdot 100 = 71.6\%$.

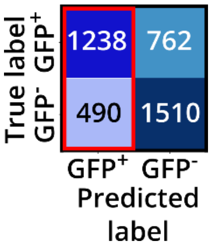


Figure 3.32 Performance of MLP$_1$ on validation set

Confusion matrix shows the performance of the final model (MLP$_1$) when being applied to the validation dataset, which contains 2000 images of small GFP$^+$ and small GFP$^-$ cells. A cell is only classified to be GFP$^+$, when the corresponding probability (P(GFP$^+$)$_{thresh}$) is ≥0.6 (red rectangle). Those cells would theoretically be sorted when using this model for an actual sorting experiment.

Application of final MLP to enrich rod precursor cells

The model was converted to .nnet using AID and loaded into the Sorting Software (version 1.556_rev1727, see section 3.4.2). A bounding box mediated gating for cells of length between 4 and 12 µm was applied to gate out events that are too small (debris) or too large. The SAW function generator was connected to the IDTs of the sorting chip and frequency as well as phase were adjusted such that pulses of 2 ms pushed single cells into the target outlet. AI-based sorting was carried out for 1 hour, effectively collecting 25,000 cells, which corresponds to an average sorting speed of approximately 7 cells/second.

For analysis of the target and initial sample, a normal glass-PDMS chip (with 20 µm channel) and 40x magnification (standard setting for RT-FDC) was used to obtain optimal fluorescence signals. The scatterplots in Figure 3.33 show the cell size and fluorescence expression for the initial and target sample, respectively. The color code of the scatter dots illustrates the event-density, suggesting a maximum density at a fluorescence intensity of approximately 300 and 4000 for the initial and target sample, respectively. Apparently, cells in the target sample tend to have higher fluorescence expression, which is also confirmed by the medians of the fluorescence intensity (M_{Init}=728 and M_{Targ}=1684, see Figure 3.33). The solid green rectangle in Figure 3.33 indicates a gating strategy for GFP$^+$ cells, which was chosen manually. The percentage of events within that gate is $c_{GFP+}^{Init} = \frac{3957}{7428} \cdot 100 = 53.2\%$ for the initial sample and $c_{GFP+}^{Targ} = \frac{1516}{2180} \cdot 100 = 69.5\%$ for the target sample. When omitting doublets from the count of GFP$^+$ cells (region indicated by dashed green rectangle in Figure 3.33), the concentration of GFP$^+$ cells in initial and target sample is $c_{GFP+}^{Init} = \frac{2949}{7428} \cdot 100 = 39.7\%$ and $c_{GFP+}^{Targ} = \frac{1187}{2180} \cdot 100 = 54.4\%$, respectively.

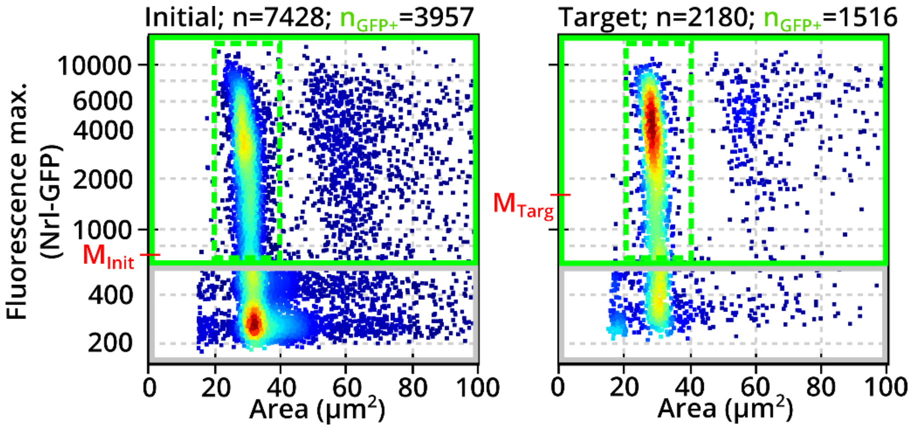


Figure 3.33 RT-FDC analysis after sorting rod precursor cells

The scatterplots show RT-FDC experiments of the initial and target sample. Axes display cells size and fluorescence expression and the color code represents the density of scatter dots. M_{Init} (=728) and M_{Targ} (=1684) show the locations of the medians of the fluorescence intensity. The solid green box indicates a gating strategy to select GFP^+ events, resulting in 53.2% and 69.5% GFP^+ cells in the initial and target sample, respectively. An alternative gating strategy, indicated by dashed lines, results in 39.7% and 54.4% GFP^+ cells in the initial and target sample, respectively.

The sorting process apparently caused a shift of the distribution of fluorescence expressions towards higher values, which means an enrichment of GFP^+ rod precursor cells. Each gating strategy (solid and dashed green rectangles in Figure 3.33) indicates an increase of GFP^+ cells by approximately 15%.

While the presented sorting experiment shows enrichment of rod precursor cells, it is still an open question whether the suggested MLP is capable to generalize for new biological samples and new sorting chips, given enough training-data. Furthermore, so far, the suggested MLP architecture was only applied for a binary classification task of retina cells (rod vs. non-rod) and it is not clear, whether the architecture would also work well for another specimen and more classes. Therefore, in the next section, a large number of existing datasets of human blood is leveraged to answer these questions.

3.5.2. Sorting of neutrophils

To support the claim that the presented methods for label-free sorting are also applicable for a different specimen than retina, this section describes the training of a neural net to distinguish different blood cell types (debris/thrombocytes, lymphocytes, red blood cells (RBCs), doublets, monocytes, neutrophils and eosinophils). In the following, the neural net is used to actively sort neutrophils from human RBC-depleted (mediated through dextran-sedimentation) blood using soRT-FDC.

Dataset assembly

Experimental data, suitable for the training and validation set are all whole blood measurements and measurements of dextran-based RBC-depleted blood, which were captured at 20x magnification in a sorting chip. In total, 100 measurements using blood from 20 donations were available. In multiple experiments, an alteration of the focus was performed, enriching the information about possible phenotypes of the cells. Each subpopulation was gated manually, using polygon-filters in ShapeOut. First, debris/thrombocytes, lymphocytes, RBCs and doublets were gated in an area vs. deformation scatterplot, as indicated by the red, orange, yellow and light-green polygons in Figure 3.34 A. The remaining granulo-monocyte fraction (dark-green rectangle in Figure 3.34 A) was then plotted in a brightness vs. standard deviation of brightness scatterplot and gated as indicated by blue, magenta and black polygons in Figure 3.34 B, which corresponds to the eosinophil, neutrophil and monocyte population, respectively [15]. This representation (B vs. B_{std}) allows a better discrimination of the individual subpopulations as compared to scatterplots showing area and brightness [15]. Even upon alteration of the focus, which affects the brightness, the subpopulations still appear well separated. In rare cases, doublets of cells fall into the gating regions, as shown by example-images in Figure 3.34 B. Since these events typically contain at least one RBC, they occur more frequently in whole blood measurements, making whole blood measurements ideal to get training data for doublets (and for RBCs as well). By checking 50 random images of eosinophils, neutrophils and monocytes, I estimated the ratio of doublets to approximately 4%.

Relatively bright and relatively dark neutrophils are difficult to classify due to their similarity to monocytes and eosinophils, respectively. Therefore, the corresponding regions (see light-gray and dark-gray polygons in Figure 3.34 B) were gated separately, which allows to use the corresponding images more frequently during training in AID. The data of each gated subpopulation of each measurement-file was exported to a format this is suitable for import into AID. Table 10 shows an overview of the available data, consisting in total of more than 3.4 million cells. Several smaller measurement files were chosen randomly to assemble a validation set. All other measurement files were used for training.

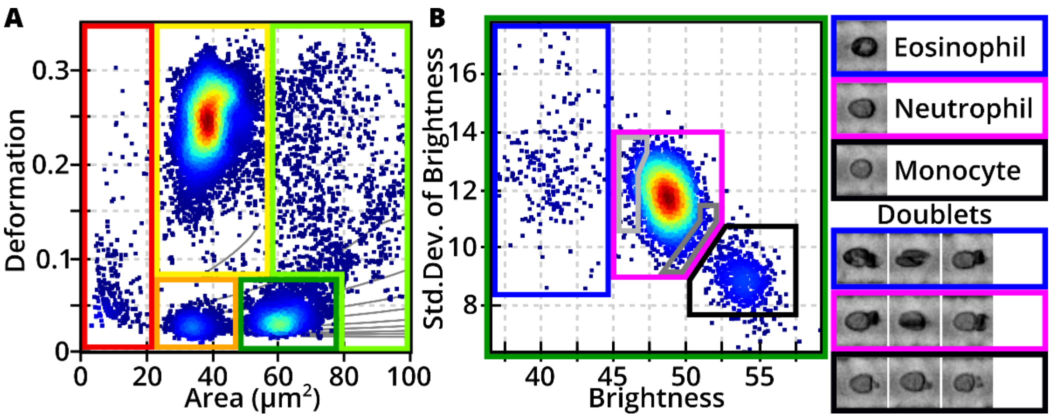


Figure 3.34 Manual gating for different blood cell types

(A) Scatterplot shows a measurement of RBC-depleted blood and the gating strategy for debris (red), lymphocytes (orange), RBCs (yellow), granulo-monocytes (dark-green), and doublets (light-green).

(B) Scatterplot shows the granulo-monocyte fraction from (A) plotted using B and B_{std}. Three populations appear, which can be assigned to eosinophils (blue), neutrophils (magenta) and monocytes (black) [15]. Example images of each subpopulation are shown at the upper right. In rare cases, the brightness levels of doublets match those of single eosinophils, neutrophils or monocytes. Example images are displayed at the lower right.

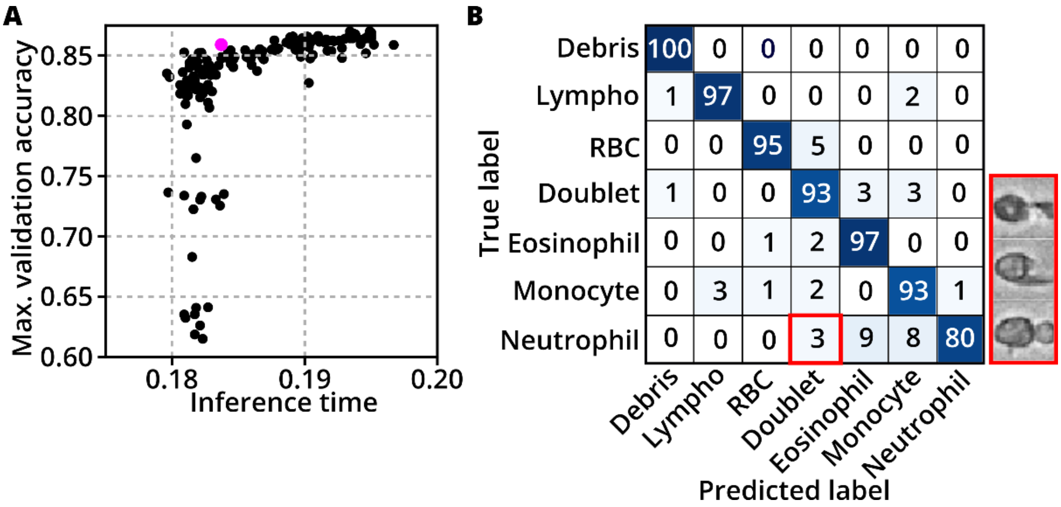
Class	Train	Valid
Debris	69959	770
Lymphocytes	363936	492
RBCs	207071	950
Doublets	93031	1063
Eosinophils	110201	1094
Monocytes	211808	1074
Neutrophils	2349493	3388

Table 10 Training and validation data of different blood cells

Training of MLP for neutrophil identification using AID

All files of gated cells were loaded into AID. To train a robust model, the data should reflect all alterations that occur from experiment to experiment. Measurements which run longer can contain orders of magnitude more events compared to other measurements. To avoid overfitting to conditions of experiments containing many events, only a certain number of random cells is used from such datasets in each training iteration.

Figure 3.15 showed a screening, which highlighted MLP architectures that performed well in predicting data from 6 biological replicates of retina. The blood dataset has a much higher variability because it contains more experiments and data of more donors. Furthermore, in the blood dataset, 7 classes of cells were defined, while in the retina dataset only two cell types were used. Therefore, the blood dataset is suited to repeat the MLP screening for a more demanding setting (more variability and more classes). AID allows training of collections of models on identically augmented data. This feature was used to perform a screening of 162 MLP architectures. Figure 3.35 A shows the maximum validation accuracy reached by each architecture after training for 500 epochs and the corresponding inference time (on an Intel® Core™ i7-3930K @ 3.2GHz). An MLP with 24, 16 and 24 nodes in the first, second and third hidden layer, respectively, appears to be a well performing architecture (magenta dot in Figure 3.35 A). This is in accordance with the previous MLP screening, documented in section 3.3.4 (see Figure 3.15), indicating that this MLP architecture could be a good choice in general. Hence, this MLP was trained further, reaching a validation accuracy of 0.94. In order to reduce the risk of falsely classifying an event as neutrophil, a threshold $P(\text{neutrophil})_{\text{thresh}} = 0.8$ was chosen. While the confusion matrix in Figure 3.35 B suggests that 3% of the neutrophils are wrongly classified as doublets (red box), example images of the corresponding events actually show doublets. Apparently, the model generalized well despite noisy training labels (see doublets in Figure 3.34). This behavior is expected since neural nets are robust to noisy labels, especially for large datasets [164].



True label \ Predicted label	Debris	Lympho	RBC	Doublet	Eosinophil	Monocyte	Neutrophil
Debris	100	0	0	0	0	0	0
Lympho	1	97	0	0	0	2	0
RBC	0	0	95	5	0	0	0
Doublet	1	0	0	93	3	3	0
Eosinophil	0	0	1	2	97	0	0
Monocyte	0	3	1	2	0	93	1
Neutrophil	0	0	0	3	9	8	80

Figure 3.35 MLP screening and training

(A) 162 different multilayer perceptron architectures were trained to distinguish seven different blood cell types for 500 epochs each. The scatterplot shows the maximum validation accuracy and the inference time of each architecture. One model is marked (magenta) since it offers a good trade-off between inference time and maximum validation accuracy. This model has 24, 16 and 24 nodes in the first, second and third hidden layer, respectively.

(B) This MLP (see magenta dot in (A)) was trained further and then applied to the validation set, resulting in the displayed normalized confusion matrix. A red box indicates that 3% of the cells that are labeled as neutrophils are predicted to be a 'Doublet'. Example images of these events are presented on the right, actually showing doublets.

Application of final MLP to enrich neutrophils

Typically, whole blood is measured in RT-DC using a 1:20 dilution of blood in MB [15]. The resulting frequency of capturing granulo-monocytes then lies between 0.2 and 0.8 cells/s (for a flowrate of 0.06 µl/s). In conclusion, sorting 50,000 cells would take at least 17 hours. Therefore, venous blood was drawn from a healthy donor and RBCs were depleted using dextran-based RBC sedimentation (see section 2.11). In biomedical research, dextran-based RBC depletion is a widely applied method, often as an initial step before density gradient centrifugation [165,166]. Cells were then resuspended in measurement buffer at a concentration of 200 mio. cells/ml. The SAW function generator was connected to the IDTs and phase as well as frequency were optimized such that cells exposed to SAW for 2 ms were pushed into the target outlet. The Sorting Software (version 1.556_rev1727, see section 3.4.2) was used to start and control the image-based sorting experiment using the trained MLP. To reduce unnecessary load on the CPU, cells with an aspect ratio above 2.0 (mostly RBCs), shorter than 6 µm (mostly debris), longer (L_x) than 18 µm (chunks) or higher (L_y) than 12 µm were gated out and were not analyzed by the MLP.

After sorting approximately 50,000 cells, the target as well as the initial sample was stained using CD14 and CD66 (see section 2.12). Each stained sample was measured using RT-FDC using a regular 20 µm glass-PDMS chip and a magnification of 40x. CD14 is a marker for monocytes and CD66 labels neutrophils [167]. The initial sample contained 3% CD14$^+$, 74.5% CD14$^-$/CD66$^-$ and 22.3% CD66$^+$ cells (see Figure 3.36 A). For the target sample, the populations of CD14$^+$ and CD14$^-$/CD66$^-$ cells reduced to 0.6% and 9.3%, respectively. In exchange, the concentration of CD66$^+$ cells increased to 89.9%. The experiment was repeated three times using blood from three different donors and three different sorting chips, (see barplot in Figure 3.36 C). For each sorting experiment, the same MLP was used.

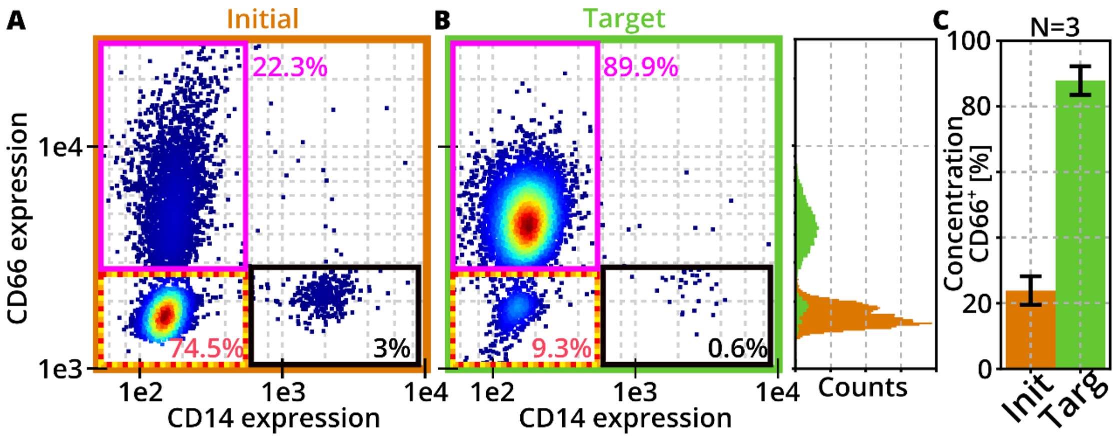


Figure 3.36 RT-FDC analysis after sorting neutrophils

(A) Scatterplot shows fluorescence expressions for CD14 and CD66 of the initial sample. The majority of cells (74.5%) is fluorescence negative, implying events of RBCs or lymphocytes.

(B) In the target sample, most cells are CD66 positive (89.9%), implying neutrophils. A vertical histogram visualizes the distribution of CD66 expression levels for the initial and target sample.

(C) The sorting experiment was repeated three times using blood from three different donors, resulting in a similar enrichment of neutrophils as shown by the barplot. The height of the bar represents the mean of three individual sorting experiments and the error bar shows the standard deviation.

3.5.3. Discussion

For the sorting of photoreceptors and neutrophils, only software with graphical user interface was used (ShapeIn, ShapeOut, AID and Sorting Software), showing that no programming skills are required to perform image-based sorting using soRT-FDC. Furthermore, with these experiments I show that the system can be operated by a single person, which allows access of the technology to a broader audience.

In the rod-sorting experiment a total of 25,000 cells were collected within one hour, which extrapolates to 100,000 cells within four hours. A sorting time of four hours is feasible and 100,000 cells already match the number of cells needed for a transplantation (50,000 per eye). The obtained concentration of rod precursor cells after sorting is only 70%, which reflects the precision of the model used. For a better sorting result, more data should be acquired and the model should be trained longer until a validation accuracy above 0.75 is reached. While a 40x magnification is standard for RT-DC and RT-FDC experiments, for sorting, currently a 20x magnification is essential, since the bifurcation point of the sorting chip needs to be observed during sorting. Due to the change of the objective from 40x (0.75 NA) to 20x (0.8 NA), the image appears different and cells are projected to less pixels. Hence, the trained models presented in section 3.3 could not be used for sorting. Instead, new data was captured right before the experiment and a new model was trained.

In contrast, for the neutrophil sorting experiments, a model was trained days/weeks in advance using existing data. For all three replicates, an increase of the concentration of CD66$^+$ cells to above 86% was achieved, which indicates an enrichment of neutrophils. Equally consistently, the concentration of CD14$^-$/CD66$^-$ cells was reduced to below 10% for each experiment, indicating a depletion of RBCs and lymphocytes. Monocytes were stained using CD14, which were reduced to below 1.15% in each experiment. Apparently, the chosen MLP architecture is capable to generalize to a sufficient bandwidth of experimental conditions (focus, brightness, different sorting chips and donors). To obtain the labeled dataset to train the model, data from existing RT-DC measurements were gated for the individual subpopulations using specific features (A, D, B and B_{std}, see Figure 3.34). This gating strategy implies that classification and sorting could also be achieved based on these features. Together with Ahsan Nawaz I indeed performed brightness-based sorting of neutrophils and recognized that adjustments of the brightness threshold for sorting are required due to slight shifts of brightness and focus during the long duration of the sorting experiment. Furthermore, doublets of cells can be located in the same brightness region like neutrophils as shown in Figure 3.34 B and would be sorted. This highlights the advantages of the MLP based sorting, as the MLP was trained to be robust for such cases.

While cells of the retina samples tended to clump together, especially for high cell-concentrations (>20 million cells/ml), blood cells mostly stayed separate for the entire time of sorting, even when using concentrations of approximately 200 million cells/ml. The resulting increase in throughput allowed to sort approximately 30 cells/s. Hence, 100,000 neutrophils could theoretically be sorted within approximately one hour. Thus, sorting of rarer subpopulations is feasible. For example, monocytes occur ten times less than neutrophils, which would theoretically allow sorting 10,000 cells within one hour when using the same cell concentration. As neutrophils and monocytes play an important role in immune defense, functional studies often require isolated but unaltered cells. Common purification methods include density gradient based methods [168], lysis of RBCs as well as molecular markers (immunomagnetic selection [169], FACS [170]). Each of these techniques implies a chemical exposition, potentially changing the properties of the cells [171–173]. While cell separation methods such as deterministic lateral displacement (DLD), elutriation, or acoustophoresis do not introduce any chemical contamination, these methods allow sorting only based on size (DLD), a combination of size and density (elutriation) or acoustic contrast (acoustophoresis) [6,174,175]. Intelligent image activated cell sorting (iIACS) is a novel technique, which would actually allow to perform label-free sorting using a DNN, but it offers a lower throughput compared to soRT-FDC and a team of researchers with multifarious skillset is required to run the setup [148]. As soRT-FDC can be operated by a single person it is a promising alternative which can be implemented relatively inexpensively, especially if an RT-DC or RT-FDC setup is already present in the lab.

4. Conclusion and outlook

In this thesis, I assessed the potential of bight-field images captured using real-time deformability cytometry (RT-DC) to distinguish different cell types. Using the example of retina cells, I showed that morphological information in the bright-field image allows identifying rod precursor cells. After assessing multiple classification techniques, certain deep neural net (DNN) architectures were identified, which provide sufficient classification accuracies at analysis rates of >1000 cells/s on commercially available computer hardware. Those DNNs are therefore suitable to be used for real-time analysis to trigger a cell sorting unit. The performance of these DNNs was tested for the identification of rod precursor cells in samples of whole dissociated retinae. DNNs can be tuned to deliver either higher yield or higher purity and would allow obtaining a photoreceptor concentration after image-based sorting that is comparable to MACS-based sorting (using CD73). Therefore, image-based sorting would allow obtaining the desired concentration for photoreceptor transplantations. The methods were successfully implemented into software tools with graphical user interface, which allows usage without need for programming. Exceeding the scope of this thesis, I leveraged all introduced tools to perform sorting experiments using two different specimens.

Sections 2.2 and 2.3, introduced parameters that are deduced from RT-DC data and section 3.1 highlighted universal properties of chosen parameters. Strong evidence for lognormal behavior of the deformation parameter was found after assessing 21,000 individual experiments, containing a total of 93.7 million measured events. The insights were then used in section 3.2 to characterize mechanical and morphological parameters of retina samples from different maturation stages. A major finding was that the population of rod precursor cells has a lower spread and is located in a region of smaller cell sizes compared to the distribution of non-rod precursor cells. A method for robust statistical analysis of RT-DC data based on linear mixed models was introduced and applied to characterize differences between subpopulations in retina, indicating size and mechanical differences. I implemented the statistical test using Python and R and it was already adopted by several other RT-DC related publications

[13,15,22,23,85,97,101–106,176]. In section 3.3.3, a total of 140 parameters, calculated from bright-field images was used to train random forests to distinguish photoreceptor and non-photoreceptor cells. Beside area, also several texture parameters (Haralick and TAS) were found to be important for the classification task. While LBP are also texture parameters, they appeared not to be important for this classification task. Deformation was under the 20 most important parameters, but its importance was 20 times lower than that of area. From this analysis one can conclude, that certain texture parameters are useful to distinguish photoreceptor and non-photoreceptor cells. Sections 3.3.4, 3.3.5, 3.3.6, and 3.3.7 focused on deep learning methods to decipher optimal texture descriptors. Hundreds of deep neural nets (DNNs) were screened to find a method that delivers a good trade-off between classification accuracy and inference time. Interestingly, certain architectures of multilayer perceptrons (MLPs), which are the simplest form of DNNs are more accurate than the random forest models and allow to classify a single image in 100 μs, which is fast enough for application in real-time. In section 3.4 I show a software tool with GUI that I developed to ease the training of DNNs for image classification. Due to the leverage of highly optimized Python libraries, the software provides a sufficient performance to train DNNs for sorting. As a result, DNN training becomes a standardized, reproducible task which is also feasible for non-programmers, allowing a broader audience to use soRT-FDC.

In section 3.5, I employed all introduced tools to perform an image-based sorting experiment using dissociated retinae where I achieved a final purity of 70% rods. Furthermore, I sorted unstained neutrophils from human blood, resulting in a target concentration above 90%. This shows that all introduced tools are practically working and the setup can even be handled by a single person. The achieved cell numbers after sorting for one hour were between 25,000 and 50,000 cells, indicating a throughput that is sufficient to collect cells for transplantation.

In this thesis, I developed image analysis methods that allow leveraging the specificity of established molecular markers to train DNNs for sorting using label-free characteristics, hence facilitating the enhancement of soRT-FDC to recognize and sort based on arbitrary patterns in cells. This approach opens up a wide field of potential applications, for example sorting of stem cells for transplantation, CFU assays or investigations of gene expression. Furthermore, unsupervised learning approaches such as UMAP or PCA could be used to discover and sort unknown cell types that can currently not be stained using molecular markers.

I showed the application of image-based sorting of photoreceptors from Nrl-GFP mice, but the final goal is the curation of retinal dystrophy in humans for which retina cells from human origin are required. To translate the approach to human cells, photoreceptors from human retinal organoids (created from embryonic or induced pluripotent stem cells) could to be used. Therefore, this project will continue in that direction, leveraging reporter lines of human origin, expressing fluorescence tags on photoreceptor cells (DFG grant 399422891 to Marius Ader).

The primary bottleneck limiting the sorting throughput is the time of 2 ms, which is currently required to exert sufficient force on cells to push them into the target channel. Using traveling surface acoustic waves, that time could be reduced. Higher classification accuracies could be obtained by improving the image quality. For example, by using a thinner substrate for the sorting chip or brighter LEDs, the image noise could be reduced. Furthermore, quantitative phase imaging could be integrated into the setup to use the resulting mass and density maps in addition to the bright-field image for classification. Even without sorting capability, real-time classification by specifically trained models could be employed for diagnostic purposes for example to detect and count interesting blood cell types or to identify malignant cells.

A. Appendix

I. Comparison of dense and convolutional layer

For sorting applications, highly optimized computational routines are required. After assessing multiple neural net architectures, convolutional neural nets (CNNs) resulted in the most accurate and multilayer perceptrons (MLPs) in the fastest algorithms (see section 3.3.4). The difference between such neural nets is only the presence of convolutional layers in CNNs, which are lacking in MLPs. While in a dense layer a node is connected to all nodes of the preceding layer, a convolutional layer limits the connections to certain neighborhoods, defined by convolutional filters. When computing a convolution, matrix multiplications using the input image and the values of the convolutional filter are performed. By screening over the image, each possible neighborhood region of the input image is assessed using the same convolutional filter (see Figure 2.11). Due to the limitation that each neighborhood region is manipulated by the same convolutional filter, the degrees of freedom is reduced, which in practice often results in robust classification performance of CNNs. The reduction in degrees of freedom reduces the complexity of the network and to compensate, more convolutional filters would need to be added.

If we neglect the reduction in degrees of freedom, the number of multiplications performed by a DNN, should be a proxy for the complexity of the DNN. The utmost processing power in MLPs and CNNs is created due to matrix multiplications. For a comparison of the dense and convolutional layer, let us design very simple architectures which only have an input layer of size 32x32 pixels which is connected to one dense or one convolutional layer. If the dense layer contains d_1 nodes, a forward pass through the network would result in $m_{MLP} = 32 \cdot 32 \cdot d_1$ multiplications. The numbers of multiplications and inference times of 100 architectures with $1 \leq d_1 \leq 100$ are shown in Figure A.1 A. Instead of the dense layer, let us now use a convolutional layer with c_1 convolutional filters of size 3x3. This convolutional filter is screened over the input image and is therefore applied $32 \cdot 32 = 1024$ times, resulting in $m_{CNN} = (32 \cdot 32) \cdot c_1 \cdot (3 \cdot 3)$ multiplications. The numbers of multiplications and inference times of 15 architectures with $1 \leq c_1 \leq 15$ are shown in Figure A.1 A. The range of d_1 and c_1 was chosen such that the number of multiplications of MLP_1 and MLP_2 are covered ($m_{MLP1} = 25024$ and $m_{MLP2} = 68096$ for an input size of 32x32). Despite having similar numbers of multiplications, the architectures containing a convolutional layer require approximately twice the inference time compared to architectures with a dense layer (see Figure A.1 A). As the same convolutional filter is used multiple times to modulate each neighborhood in the input image, the number of parameters of a convolutional layer is in general much lower than of a dense layer (see Figure A.1 B). While these properties give rise to the better generalization of CNNs, they also currently render convolutional layers unfavorable for DNNs for real-time application in RT-(F)DC or soRT-FDC.

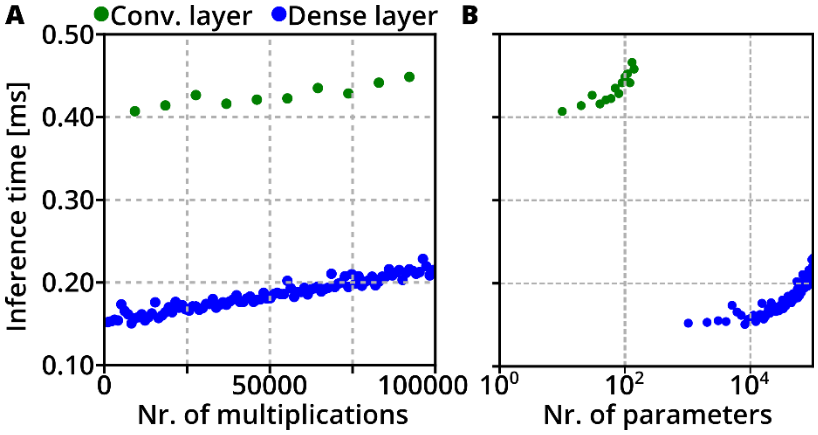


Figure A.1 Inference time of dense vs. convolutional layer

(A) Scatterplot shows the inference time and number of multiplications of neural net architectures that only contain an input layer and one dense (blue) or convolutional layer (green). Computations were performed on an Intel® Core™ i7-4810MQ @ 2.80 GHz.

(B) Scatterplot shows the same architectures as in (A), but the x-axis shows the number of parameters of the network (log. scale). The CNNs have orders of magnitude less parameters due to parameter sharing.

Bibliography

1. Shapiro, H. M. *Practical Flow Cytometry*. (John Wiley & Sons, Inc., 2003). doi:10.1002/0471722731

2. Santos-Ferreira, T. *et al.* Daylight Vision Repair by Cell Transplantation. *Stem Cells* **33**, 79–90 (2015).

3. Miltenyi, S., Müller, W., Weichel, W. & Radbruch, A. High gradient magnetic cell separation with MACS. *Cytometry* **11**, 231–238 (1990).

4. Šafařık, I. & Šafařıková, M. Use of magnetic techniques for the isolation of cells. *J. Chromatogr. B Biomed. Sci. Appl.* **722**, 33–53 (1999).

5. Bonner, W. A., Hulett, H. R., Sweet, R. G. & Herzenberg, L. A. Fluorescence Activated Cell Sorting. *Rev. Sci. Instrum.* **43**, 404–409 (1972).

6. Stroncek, D. F. *et al.* Counter-flow elutriation of clinical peripheral blood mononuclear cell concentrates for the production of dendritic and T cell therapies. *J. Transl. Med.* **12**, 241 (2014).

7. Preira, P. *et al.* Passive circulating cell sorting by deformability using a microfluidic gradual filter. *Lab Chip* **13**, 161–170 (2013).

8. Wang, G. *et al.* Microfluidic cellular enrichment and separation through differences in viscoelastic deformation. *Lab Chip* **15**, 532–540 (2015).

9. Holmes, D. *et al.* Separation of blood cells with differing deformability using deterministic lateral displacement. *Interface Focus* **4**, 20140011 (2014).

10. Beech, J. P., Holm, S. H., Adolfsson, K. & Tegenfeldt, J. O. Sorting cells by size, shape and deformability. *Lab Chip* **12**, 1048–1051 (2012).

11. Salzman, G. C. Light Scattering Analysis of Single Cells. in *Cell Analysis* 111–143 (Springer US, 1982). doi:10.1007/978-1-4684-4097-3_5

12. Otto, O. *et al.* Real-time deformability cytometry: on-the-fly cell mechanical phenotyping. *Nat. Methods* **12**, 199–202 (2015).

13. Rosendahl, P. *et al.* Real-time fluorescence and deformability cytometry. *Nat. Methods* **15**, 355–358 (2018).

14. Nawaz, A. A. *et al.* Using real-time fluorescence and deformability cytometry and deep learning to transfer molecular specificity to label-free sorting. *bioRxiv* 862227 (2019). doi:10.1101/862227

15. Toepfner, N. *et al.* Detection of human disease conditions by single-cell morpho-rheological phenotyping of blood. *Elife* **7**, e29213 (2018).

16. Das, D., Ghosh, M., Chakraborty, C., Maiti, A. K. & Pal, M. Probabilistic prediction of malaria using morphological and textural information. in *2011 International Conference on Image Information Processing* 1–6 (IEEE, 2011). doi:10.1109/ICIIP.2011.6108879

17. Hamilton, N. A., Pantelic, R. S., Hanson, K. & Teasdale, R. D. Fast automated cell phenotype image classification. *BMC Bioinformatics* **8**, 110 (2007).

18. Linder, N. *et al.* A malaria diagnostic tool based on computer vision screening and visualization of Plasmodium falciparum candidate areas in digitized blood smears. *PLoS One* **9**, e104855 (2014).

19. Adjed, F., Faye, I., Ababsa, F., Gardezi, S. J. & Dass, S. C. Classification of skin cancer images using local binary pattern and SVM classifier. in 080006 (2016). doi:10.1063/1.4968145

20. Lange, J. R. *et al.* Microconstriction Arrays for High-Throughput Quantitative Measurements of Cell Mechanical Properties. *Biophys. J.* **109**, 26–34 (2015).

21. Hosseini, S. M. & Feng, J. J. How malaria parasites reduce the deformability of infected red blood cells. *Biophys. J.* **103**, 1–10 (2012).

22. Koch, M. *et al.* Plasmodium falciparum erythrocyte-binding antigen 175 triggers a biophysical change in the red blood cell that facilitates invasion. *Proc. Natl. Acad. Sci.* **114**, 4225–4230 (2017).

23. Steffen, S. *et al.* Toll-Like Receptor-Mediated Upregulation of CXCL16 in Psoriasis

Orchestrates Neutrophil Activation. *J. Invest. Dermatol.* **138**, 344–354 (2018).

24. Guck, J. *et al.* Optical deformability as an inherent cell marker for testing malignant transformation and metastatic competence. *Biophys. J.* **88**, 3689–98 (2005).

25. Lekka, M. *et al.* Elasticity of normal and cancerous human bladder cells studied by scanning force microscopy. *Eur. Biophys. J.* **28**, 312–316 (1999).

26. Ekpenyong, A. E. *et al.* Viscoelastic Properties of Differentiating Blood Cells Are Fate- and Function-Dependent. *PLoS One* **7**, e45237 (2012).

27. Darling, E. M., Topel, M., Zauscher, S., Vail, T. P. & Guilak, F. Viscoelastic properties of human mesenchymally-derived stem cells and primary osteoblasts, chondrocytes, and adipocytes. *J. Biomech.* **41**, 454–464 (2008).

28. Baylor, D. A., Lamb, T. D. & Yau, K. W. Responses of retinal rods to single photons. *J. Physiol.* **288**, 613–34 (1979).

29. Pearson, R. A. *et al.* Restoration of vision after transplantation of photoreceptors. *Nature* **485**, 99–103 (2012).

30. Swaroop, A., Kim, D. & Forrest, D. Transcriptional regulation of photoreceptor development and homeostasis in the mammalian retina. *Nat. Rev. Neurosci.* **11**, 563–576 (2010).

31. Eberle, D., Schubert, S., Postel, K., Corbeil, D. & Ader, M. Increased Integration of Transplanted CD73-Positive Photoreceptor Precursors into Adult Mouse Retina. *Investig. Opthalmology Vis. Sci.* **52**, 6462 (2011).

32. MacLaren, R. E. *et al.* Retinal repair by transplantation of photoreceptor precursors. *Nature* **444**, 203–207 (2006).

33. Barber, A. C. *et al.* Repair of the degenerate retina by photoreceptor transplantation. *Proc. Natl. Acad. Sci.* **110**, 354–359 (2013).

34. Singh, M. S. *et al.* Reversal of end-stage retinal degeneration and restoration of visual function by photoreceptor transplantation. *Proc. Natl. Acad. Sci. U. S. A.* **110**,

1101–6 (2013).

35. Bartsch, U. *et al.* Retinal cells integrate into the outer nuclear layer and differentiate into mature photoreceptors after subretinal transplantation into adult mice. *Exp. Eye Res.* **86**, 691–700 (2008).

36. Santos-Ferreira, T. *et al.* Retinal transplantation of photoreceptors results in donor–host cytoplasmic exchange. *Nat. Commun.* **7**, 13028 (2016).

37. Eiraku, M. *et al.* Self-organizing optic-cup morphogenesis in three-dimensional culture. *Nature* **472**, 51–56 (2011).

38. Nakano, T. *et al.* Self-Formation of Optic Cups and Storable Stratified Neural Retina from Human ESCs. *Cell Stem Cell* **10**, 771–785 (2012).

39. Gonzalez-Cordero, A. *et al.* Photoreceptor precursors derived from three-dimensional embryonic stem cell cultures integrate and mature within adult degenerate retina. *Nat. Biotechnol.* **31**, 741–747 (2013).

40. Decembrini, S., Koch, U., Radtke, F., Moulin, A. & Arsenijevic, Y. Derivation of Traceable and Transplantable Photoreceptors from Mouse Embryonic Stem Cells. *Stem Cell Reports* **2**, 853–865 (2014).

41. Völkner, M. *et al.* Retinal Organoids from Pluripotent Stem Cells Efficiently Recapitulate Retinogenesis. *Stem Cell Reports* **6**, 525–538 (2016).

42. Santos-Ferreira, T. *et al.* Stem Cell–Derived Photoreceptor Transplants Differentially Integrate Into Mouse Models of Cone-Rod Dystrophy. *Investig. Opthalmology Vis. Sci.* **57**, 3509 (2016).

43. Green, R. E., Sosik, H. M., Olson, R. J. & DuRand, M. D. Flow cytometric determination of size and complex refractive index for marine particles: comparison with independent and bulk estimates. *Appl. Opt.* **42**, 526 (2003).

44. Barteneva, N. S., Fasler-Kan, E. & Vorobjev, I. A. Imaging flow cytometry: coping with heterogeneity in biological systems. *J. Histochem. Cytochem.* **60**, 723–33 (2012).

45. Doan, M. *et al.* Diagnostic Potential of Imaging Flow Cytometry. *Trends Biotechnol.* **36**, 649–652 (2018).

46. Hiramatsu, K. *et al.* High-throughput label-free molecular fingerprinting flow cytometry. *Sci. Adv.* **5**, 1–9 (2019).

47. Suzuki, Y. *et al.* Label-free chemical imaging flow cytometry by high-speed multicolor stimulated Raman scattering. *Proc. Natl. Acad. Sci.* **116**, 15842–15848 (2019).

48. Gorthi, S. S. & Schonbrun, E. Phase imaging flow cytometry using a focus-stack collecting microscope. *Opt. Lett.* **37**, 707 (2012).

49. Chen, C. L. *et al.* Deep Learning in Label-free Cell Classification. *Sci. Rep.* **6**, 1–16 (2016).

50. Merola, F. *et al.* Tomographic flow cytometry by digital holography. *Light Sci. Appl.* **6**, 1–7 (2017).

51. Min, J. *et al.* Quantitative phase imaging of cells in a flow cytometry arrangement utilizing Michelson interferometer-based off-axis digital holographic microscopy. *J. Biophotonics* 1–10 (2019). doi:10.1002/jbio.201900085

52. Guck, J. & Chilvers, E. R. Mechanics Meets Medicine. *Sci. Transl. Med.* **5**, 212fs41-212fs41 (2013).

53. Zhang, J., Nou, X. A., Kim, H. & Scarcelli, G. Brillouin flow cytometry for label-free mechanical phenotyping of the nucleus. *Lab Chip* **17**, 663–670 (2017).

54. Zhang, J., Fiore, A., Yun, S. H., Kim, H. & Scarcelli, G. Line-scanning Brillouin microscopy for rapid non-invasive mechanical imaging. *Sci. Rep.* **6**, 1–8 (2016).

55. Hochmuth, R. M. Micropipette aspiration of living cells. *J. Biomech.* **33**, 15–22 (2000).

56. Ashkin, A. Acceleration and Trapping of Particles by Radiation Pressure. *Phys. Rev. Lett.* **24**, 156–159 (1970).

57. Guck, J. *et al.* The optical stretcher: a novel laser tool to micromanipulate cells.

Biophys. J. **81**, 767–84 (2001).

58. Binnig, G., Quate, C. F. & Gerber, C. Atomic Force Microscope. *Phys. Rev. Lett.* **56**, 930–933 (1986).

59. Rosenbluth, M. J., Lam, W. A. & Fletcher, D. A. Analyzing cell mechanics in hematologic diseases with microfluidic biophysical flow cytometry. *Lab Chip* **8**, 1062–70 (2008).

60. Byun, S. *et al.* Characterizing deformability and surface friction of cancer cells. *Proc. Natl. Acad. Sci.* **110**, 7580–7585 (2013).

61. Dudani, J. S., Gossett, D. R., Tse, H. T. K. & Di Carlo, D. Pinched-flow hydrodynamic stretching of single-cells. *Lab Chip* **13**, 3728 (2013).

62. Gossett, D. R. *et al.* Hydrodynamic stretching of single cells for large population mechanical phenotyping. *Proc. Natl. Acad. Sci.* **109**, 7630–7635 (2012).

63. Nitta, N. *et al.* Intelligent Image-Activated Cell Sorting. *Cell* **175**, 266-276.e13 (2018).

64. Heath, J. R., Ribas, A. & Mischel, P. S. Single-cell analysis tools for drug discovery and development. *Nat. Rev. Drug Discov.* **15**, 204–216 (2016).

65. Tanay, A. & Regev, A. Scaling single-cell genomics from phenomenology to mechanism. *Nature* **541**, 331–338 (2017).

66. Islam, M. *et al.* Microfluidic cell sorting by stiffness to examine heterogenic responses of cancer cells to chemotherapy. *Cell Death Dis.* **9**, 239 (2018).

67. Poser, S. W. *et al.* Controlling distinct signaling states in cultured cancer cells provides a new platform for drug discovery. *FASEB J.* **33**, 9235–9249 (2019).

68. Bradski, G. The OpenCV library. *Dr Dobbs J. Softw. Tools* **25**, 120–126 (2000).

69. Suzuki, S. & Be, K. Topological structural analysis of digitized binary images by border following. *Comput. Vision, Graph. Image Process.* **30**, 32–46 (1985).

70. Müller, C. *et al.* Surface acoustic wave investigations of the metal-to-insulator

transition of V2O3 thin films on lithium niobate. *J. Appl. Phys.* **98**, 084111 (2005).

71. Nawaz, A. A. *et al.* Acoustofluidic Fluorescence Activated Cell Sorter. *Anal. Chem.* **87**, 12051–12058 (2015).

72. Görisch, S. M., Lichter, P. & Rippe, K. Mobility of multi-subunit complexes in the nucleus: accessibility and dynamics of chromatin subcompartments. *Histochem. Cell Biol.* **123**, 217–228 (2005).

73. Kell, G. S. Density, thermal expansivity, and compressibility of liquid water from 0.deg. to 150.deg.. Correlations and tables for atmospheric pressure and saturation reviewed and expressed on 1968 temperature scale. *J. Chem. Eng. Data* **20**, 97–105 (1975).

74. Hartono, D. *et al.* On-chip measurements of cell compressibility via acoustic radiation. *Lab Chip* **11**, 4072 (2011).

75. Herold, C. Mapping of Deformation to Apparent Young's Modulus in Real-Time Deformability Cytometry. (2017).

76. Rosendahl, P. M. Mechanical Characterization of Suspended Cells in Microfluidic Channels. (Technische Universität Dresden, 2013).

77. Mietke, A. *et al.* Extracting Cell Stiffness from Real-Time Deformability Cytometry: Theory and Experiment. *Biophys. J.* **109**, 2023–2036 (2015).

78. Geoff Olynyk. File Exchange - MATLAB Central - Mathworks. (2012). Available at: http://de.mathworks.com/matlabcentral/fileexchange/36525-volrevolve/content/volRevolve.m. (Accessed: 1st January 2017)

79. Green, G., Wheelhouse, T. & Lindsay, R. B. An Essay on the Application of Mathematical Analysis to the Theories of Electricity and Magnetism. *Phys. Today* **12**, 48–48 (1959).

80. Joseph Louis Lagrange. *Oeuvres de Lagrange*. (Gauthier-Villars, 1867).

81. Herbig, M. *et al.* Real-Time Deformability Cytometry: Label-Free Functional Characterization of Cells. in *Flow Cytometry Protocols* (ed. Teresa S. Hawley, R. G.

H.) **1678**, 347–369 (2018).

82. Herbig, M., Mietke, A., Müller, P. & Otto, O. Statistics for real-time deformability cytometry: Clustering, dimensionality reduction, and significance testing. *Biomicrofluidics* **12**, 042214 (2018).

83. Müller, P., Herbig, M. & Rosendahl, P. ShapeOut. (2017).

84. Mokbel, M. *et al.* Numerical Simulation of Real-Time Deformability Cytometry To Extract Cell Mechanical Properties. *ACS Biomater. Sci. Eng.* **3**, 2962–2973 (2017).

85. Tavares, S. *et al.* Actin stress fiber organization promotes cell stiffening and proliferation of pre-invasive breast cancer cells. *Nat. Commun.* **8**, 15237 (2017).

86. Richard, H. A. & Sander, M. *Technische Mechanik : Festigkeitslehre ; Lehrbuch mit Praxisbeispielen, Klausuraufgaben und Lösungen*. (Vieweg + Teubner, 2008).

87. Haralick, R. M., Shanmugam, K. & Dinstein, I. Textural Features for Image Classification. *IEEE Trans. Syst. Man. Cybern.* **SMC**-**3**, 610–621 (1973).

88. Coelho, L. P. Mahotas: Open source software for scriptable computer vision. *J. Open Res. Softw.* **1**, e3 (2013).

89. Ojala, T., Pietikainen, M. & Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**, 971–987 (2002).

90. Coelho, L. P. *et al.* Structured Literature Image Finder: Extracting Information from Text and Images in Biomedical Literature. in 23–32 (Springer, Berlin, Heidelberg, 2010). doi:10.1007/978-3-642-13131-8_4

91. Ali, J., Ahmad, A., George, L. E., Der, C. S. & Aziz, S. Red Blood Cell Recognition using Geometrical Features. *Int. J. Comput. Sci. Issues* **10**, 90–94 (2013).

92. Sanfilippo, P. G., Grimm, J. L., Flanagan, J. G., Lathrop, K. L. & Sigal, I. A. Application of Elliptic Fourier analysis to describe the lamina cribrosa shape with age and intraocular pressure. *Exp. Eye Res.* **128**, 1–7 (2014).

93. Chitwood, D. H. & Otoni, W. C. Morphometric analysis of Passiflora leaves: the

relationship between landmarks of the vasculature and elliptical Fourier descriptors of the blade. *Gigascience* **6**, 1–13 (2017).

94. Kuhl, F. P. & Giardina, C. R. Elliptic Fourier features of a closed contour. *Computer Graphics and Image Processing* **18**, 236–258 (1982).

95. Due Trier, Ø., Jain, A. K. & Taxt, T. Feature extraction methods for character recognition-A survey. *Pattern Recognit.* **29**, 641–662 (1996).

96. Blidh, H. PyEFD. (2016). Available at: https://pyefd.readthedocs.io/en/latest/.

97. Xavier, M. *et al.* Mechanical phenotyping of primary human skeletal stem cells in heterogeneous populations by real-time deformability cytometry. *Integr. Biol.* **8**, 616–623 (2016).

98. R-Core-Team. R: A Language and Environment for Statistical Computing. (2014).

99. Bates, D. M. Fitting linear mixed models in R. *R News* **5**, 27–30 (2005).

100. Wilks, S. S. The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *Ann. Math. Stat.* **9**, 60–62 (1938).

101. Urbanska, M. *et al.* Single-cell mechanical phenotype is an intrinsic marker of reprogramming and differentiation along the mouse neural lineage. *Development* **144**, 4313–4321 (2017).

102. Bartel, K. *et al.* V-ATPase inhibition increases cancer cell stiffness and blocks membrane related Ras signaling - a new option for HCC therapy. *Oncotarget* **8**, 9476–9487 (2016).

103. Mziaut, H. *et al.* The F-actin modifier villin regulates insulin granule dynamics and exocytosis downstream of islet cell autoantigen 512. *Mol. Metab.* **5**, 656–68 (2016).

104. Kräter, M. *et al.* Bone marrow niche-mimetics modulate HSPC function via integrin signaling. *Sci. Rep.* **7**, 2549 (2017).

105. Bekeschus, S. *et al.* Toxicity and Immunogenicity in Murine Melanoma following Exposure to Physical Plasma-Derived Oxidants. *Oxid. Med. Cell. Longev.* **2017**, 1–12 (2017).

106. Riesner, K. *et al.* Initiation of acute graft-versus-host disease by angiogenesis. *Blood* **129**, 2021–2032 (2017).

107. Reynolds, D. Gaussian Mixture Models. in *Encyclopedia of Biometrics* 827–832 (Springer US, 2015). doi:10.1007/978-1-4899-7488-4_196

108. Dempster, A. P., Laird, N. M. & Rubin, D. B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. R. Stat. Soc. Ser. B* **39**, 1–38 (1977).

109. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. (2012).

110. Breiman, L. Bagging Predictors. *Mach. Learn.* **24**, 123–140 (1996).

111. Brewka, G. Artificial intelligence—a modern approach by Stuart Russell and Peter Norvig, Prentice Hall. Series in Artificial Intelligence, Englewood Cliffs, NJ. *Knowl. Eng. Rev.* **11**, 78 (1996).

112. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).

113. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Proc. 25th Int. Conf. Neural Inf. Process. Syst.* **1**, 1097–1105 (2012).

114. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**, 386–408 (1958).

115. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).

116. Ioffe, S. & Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. (2015).

117. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).

118. Le Cun, Y. & Le Cun, Y. Generalization and Network Design Strategies. in *Connectionism in perspective* (eds. Pfeifer, R., Schreter, Z., Fogelman, F. & Steels, L.)

(Elsevier, 1989).

119. Zeiler, M. D. & Fergus, R. Visualizing and Understanding Convolutional Networks. (2013).

120. Goodfellow, I. J. *et al.* Generative Adversarial Networks. (2014).

121. Arjovsky, M., Chintala, S. & Bottou, L. Wasserstein GAN. (2017).

122. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. Improved Training of Wasserstein GANs. (2017).

123. Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. (2017).

124. Koch, A. L. The logarithm in biology 1. Mechanisms generating the log-normal distribution exactly. *J. Theor. Biol.* **12**, 276–290 (1966).

125. Motulsky, H. *Intuitive biostatistics : a nonmathematical guide to statistical thinking*.

126. Neurohr, G. E. *et al.* Excessive Cell Growth Causes Cytoplasm Dilution And Contributes to Senescence. *Cell* **176**, 1083-1097.e18 (2019).

127. Eberle, D., Santos-Ferreira, T., Grahl, S. & Ader, M. Subretinal Transplantation of MACS Purified Photoreceptor Precursor Cells into the Adult Mouse Retina. *J. Vis. Exp.* (2014). doi:10.3791/50932

128. Singh, M. S. *et al.* Transplanted photoreceptor precursors transfer proteins to host photoreceptors by a mechanism of cytoplasmic fusion. *Nat. Commun.* **7**, 13537 (2016).

129. THE PROBABLE ERROR OF A MEAN. *Biometrika* **6**, 1–25 (1908).

130. Mann, H. B. & Whitney, D. R. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Ann. Math. Stat.* **18**, 50–60 (1947).

131. Kruskal, W. H. & Wallis, W. A. Use of Ranks in One-Criterion Variance Analysis. *J. Am. Stat. Assoc.* **47**, 583–621 (1952).

132. Gelman, A. & Hill, J. *Data Analysis Using Regression and Multilevel/Hierarchical*

Models. *Journal of Educational Measurement* (Cambridge University Press, 2006). doi:10.1017/CBO9780511790942

133. Breslow, N. E. & Clayton, D. G. Approximate Inference in Generalized Linear Mixed Models. *J. Am. Stat. Assoc.* **88**, 9 (1993).

134. Koller, M. robustlmm : An R Package for Robust Estimation of Linear Mixed-Effects Models. *J. Stat. Softw.* **75**, (2016).

135. Bürkner, P.-C. Advanced Bayesian Multilevel Modeling with the R Package brms. *R J.* **10**, 395 (2018).

136. Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. in *2014 IEEE Conference on Computer Vision and Pattern Recognition* 1701–1708 (IEEE, 2014). doi:10.1109/CVPR.2014.220

137. Goodfellow, I., Bengio, Y. & Courville, A. *Deep learning*. (MIT Press, 2016).

138. Masters, T. Designing Feedforward Network Architectures. in *Practical Neural Network Recipies in C++* 173–185 (Elsevier, 1993). doi:10.1016/B978-0-08-051433-8.50015-X

139. Kösters, M., Sturman, B., Werheit, P., Haertle, D. & Buse, K. Optical cleaning of congruent lithium niobate crystals. *Nat. Photonics* **3**, 510–513 (2009).

140. Villarroel, J. *et al.* Analysis of photorefractive optical damage in lithium niobate: application to planar waveguides. *Opt. Express* **18**, 20852 (2010).

141. Session with Yann LeCun - Quora. Available at: https://www.quora.com/session/Yann-LeCun/1. (Accessed: 7th June 2018)

142. Reed, S. *et al.* Generative Adversarial Text to Image Synthesis. (2016).

143. Ledig, C. *et al.* Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 105–114 (IEEE, 2017). doi:10.1109/CVPR.2017.19

144. Zhu, J.-Y., Krähenbühl, P., Shechtman, E. & Efros, A. A. Generative Visual

Manipulation on the Natural Image Manifold. (2016).

145. Kotikalapudi, R. keras-vis. (2017). Available at: https://github.com/raghakot/keras-vis. (Accessed: 30th July 2018)

146. Molchanov, P., Tyree, S., Karras, T., Aila, T. & Kautz, J. Pruning Convolutional Neural Networks for Resource Efficient Inference. (2016).

147. Gu, Y. *et al.* Machine Learning Based Real-Time Image-Guided Cell Sorting and Classification. *Cytom. Part A* **95**, 499–509 (2019).

148. Isozaki, A. *et al.* A practical guide to intelligent image-activated cell sorting. *Nat. Protoc.* **14**, 2370–2415 (2019).

149. Buggenthin, F. *et al.* Prospective identification of hematopoietic lineage choice by deep learning. *Nat. Methods* **14**, 403–406 (2017).

150. Nassar, M. *et al.* Label-Free Identification of White Blood Cells Using Machine Learning. *Cytom. Part A* cyto.a.23794 (2019). doi:10.1002/cyto.a.23794

151. Jones, T. R. *et al.* CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics* **9**, 482 (2008).

152. Schneider, C. A., Rasband, W. S. & Eliceiri, K. W. NIH Image to ImageJ: 25 years of image analysis. *Nat. Methods* **9**, 671–675 (2012).

153. Al-Rfou, R. *et al.* Theano: A Python framework for fast computation of mathematical expressions. (2016).

154. Abadi, M. *et al.* TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. (2016).

155. Paszke, A. *et al.* Automatic differentiation in PyTorch. (2017).

156. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. (2009).

157. Tan, C. *et al.* A Survey on Deep Transfer Learning. (2018).

158. Bai, J., Lu, F., Zhang, K. & Others. ONNX: Open Neural Network Exchange. Available at: https://github.com/onnx/onnx. (Accessed: 17th July 2019)

159. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **27**, 861–874 (2006).

160. Chollet, F. and others. Keras. (2015). Available at: https://keras.io/.

161. Płoński, P. keras2cpp. Available at: https://github.com/pplonski/keras2cpp. (Accessed: 19th June 2019)

162. Berg, S. *et al.* ilastik: interactive machine learning for (bio)image analysis. *Nat. Methods* (2019). doi:10.1038/s41592-019-0582-9

163. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).

164. Rolnick, D., Veit, A., Belongie, S. & Shavit, N. Deep Learning is Robust to Massive Label Noise. (2017).

165. Quach, A. & Ferrante, A. The Application of Dextran Sedimentation as an Initial Step in Neutrophil Purification Promotes Their Stimulation, due to the Presence of Monocytes. *J. Immunol. Res.* **2017**, 1–10 (2017).

166. Bøyum, A. Separation of Blood Leucocytes, Granulocytes and Lymphocytes. *Tissue Antigens* **4**, 269–274 (2008).

167. Gustafson, M. P. *et al.* A method for identification and analysis of non-overlapping myeloid immunophenotypes in humans. *PLoS One* **10**, e0121546 (2015).

168. Dagur, P. K. & McCoy, J. P. Collection, Storage, and Preparation of Human Blood Cells. in *Current Protocols in Cytometry* 5.1.1-5.1.16 (John Wiley & Sons, Inc., 2015). doi:10.1002/0471142956.cy0501s73

169. Cotter, M. J., Norman, K. E., Hellewell, P. G. & Ridger, V. C. A novel method for isolation of neutrophils from murine blood using negative immunomagnetic separation. *Am. J. Pathol.* **159**, 473–81 (2001).

170. Lagasse, E. & Weissman, I. L. Flow cytometric identification of murine neutrophils and monocytes. *J. Immunol. Methods* **197**, 139–150 (1996).

171. McAfee, J. G., Subramanian, G. & Gagne, G. Technique of leukocyte harvesting and

labeling: Problems and perspectives. *Semin. Nucl. Med.* **14**, 83–106 (1984).

172. Haslett, C., Guthrie, L. A., Kopaniak, M. M., Johnston, R. B. & Henson, P. M. Modulation of multiple neutrophil functions by preparative methods or trace concentrations of bacterial lipopolysaccharide. *Am. J. Pathol.* **119**, 101–10 (1985).

173. Bignold, L. P. Effects of preparative technique on the rate of adoption of crawling-like movements by polymorphonuclear leukocytes. *Cell Biol. Int. Rep.* **11**, 19–25 (1987).

174. Huang, L. R., Cox, E. C., Austin, R. H. & Sturm, J. C. Continuous Particle Separation Through Deterministic Lateral Displacement. *Science (80-.).* **304**, 987–990 (2004).

175. Augustsson, P., Karlsen, J. T., Su, H.-W., Bruus, H. & Voldman, J. Iso-acoustic focusing of cells for size-insensitive acousto-mechanical phenotyping. *Nat. Commun.* **7**, 11556 (2016).

176. Guzniczak, E. *et al.* Deformability-induced lift force in spiral microchannels for cell separation. *Lab Chip* (2020). doi:10.1039/C9LC01000A

Acronyms

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
AI	Artificial intelligence
AID	AIDeveloper (official name). Do not use “artificial intelligence developer”
AMD	Age-related macular disease
APC	Allophycocyanin; a fluorophore
BIC	Bayesian information criterion
C++	This is no abbreviation but the official name of a programming language
CAD	Computer-aided design
CD14, 66, 73,…	Cluster of differentiation. Indicate surface molecules of cells.
CIFAR10	A dataset named after the Canadian Institute for Advanced Research
CM	Confusion matrix
CMOS	Complementary metal-oxide-semiconductor
CNN	Convolutional neural net
CPU	Central processing unit
DC	Deformability cytometry; a microfluidic technique [62]
DLD	Deterministic lateral displacement [9,10]
DNA	Deoxyribonucleic acid
DNN	Deep neural net
E15	Embryonic day 15, which means 15 days after fertilization
EDTA	Ethylenediaminetetraacetic acid; anticoagulant during for blood collection
EFF	Elliptic Fourier function
FACS	Fluorescence-activated cell sorting
FCS	Fetal calf serum (often also called Fetal bovine serum – FBS)
FM	Feature map
FN	False negative
FP	False positive
FPGA	Field-programmable gate array
GAN	Generative adversarial network
GFP	Green fluorescent protein
GMM	Gaussian mixture models
GPU	Graphics processing unit
GUI	Graphical user interface
HIV	Human immunodeficiency virus
IDT	Interdigital transducer
iIACS	Intelligent Image-Activated Cell Sorting [63]
IQR	Interquartile range
LBP	Local binary patterns [89]

LED	Light-emitting diode
LMM	Linear mixed model
LUT	Lookup table
MACS	Magnetic-activated cell sorting
MB	Measurement buffer
MC	Methyl cellulose
MCF10A	Michigan Cancer Foundation – 10A; a human breast epithelial cell line
MG-63	Human osteosarcoma cell line
mio.	Million
MLP	Multilayer perceptron
MSE	Mean squared error
NA	Numerical aperture
Nrl	Neural retina-specific leucine zipper protein
P04, P10,…	Postnatal day 4, 10, …
PBS	Phosphate-buffered saline
PC	Personal computer
PDMS	Polydimethylsiloxane
PE	Phycoerythrin; a fluorophore
PSC	Pluripotent stem cell
RBC	Red blood cell
ReLU	Rectified linear unit
RF	Random forest
RGB	Red green blue
ROC	Receiver operating characteristic
ROI	Region of interest
RP	Retinitis pigmentosa
rpm	Revolutions per minute
RT-DC	Real-time deformability cytometry [12]
RT-FDC	Real-time fluorescence and deformability cytometry [13]
SAW	Surface acoustic wave
SGD	Stochastic gradient descent
soRT-FDC	Sorting real-time fluorescence and deformability cytometry [14]
SSC	Human skeletal stem cells
Std	Standard deviation
TAS	Threshold adjacency statistics [17]
TN	True negative
TP	True positive

Acknowledgements

First of all, I want to thank Jochen Guck as he gave me the amazing opportunity to work in his group. Also, I want to thank him for all the fruitful discussions and guidance throughout this project. His style of leadership allowed me lots of scientific freedom to develop freely but provided also a clear vision directing the project towards a meaningful outcome. Through the years Jochen initiated so many collaborations which were truly advantageous for my work and broadened my knowledge each time. I want to thank Oliver Otto, who supervised me during the first years of the thesis. His guidance and motivation propelled this project forward. Without his inspiring nature I would have not gotten so far.

My work of its own would just be a bunch of boring analysis tools. Only in concert with the multifarious skillset of the team of gucklab and collaboration partners, it was possible to create meaningful outcome. Hence, I thank Ahsan Nawaz who patiently introduced me to the magic of soRT-FDC and all related bits and pieces. I thank Martin Nötzel who created a beast of a Sorting Software and even implemented for me the option to use neural nets for sorting. I thank Martin Kräter, who gave me lots of advice and help in biological questions and was always open for discussion or a gym session. Martin Kräter motivated me to improve AID to ease application by non-programmers and he had the idea for a corresponding publication which will be submitted soon. After hours of discussion and months of hard work we managed to finalize a manuscript. I thank Shada Abuhattum Hofemeier and Despina Soteriou who also contributed a lot to this manuscript. I want to thank Philipp Rosendahl and Christoph Herold for all the help, especially for maintaining the RT-FDC setup and providing me certain options in ShapeIn which enormously eased the acquisition of data when using sorting chips. Special thanks also go to Philipp Rosendahl for proofreading this thesis.

I thank Paul Müller for implementing the linear mixed model analysis into ShapeOut. Availability of this feature in ShapeOut speeded up the analysis routine (not only for me). I greatly acknowledge his continuous effort to improve ShapeOut and to solve my specific issues, making RT-DC data analysis easier and faster. I thank Marius Ader, Tiago Ferreira, Oliver Borsch, Silvia Llonch, Sylvia Gasparini and Karen Teßmer for discussions

and preparation of retina samples. Production of silicon masters was performed by Ahsan Nawaz and the Microstructure Facility and I also want to namely thank Salvatore Girardo and Ruchi Goswami for support and help. I thank Konrad Wauer for discussions and inspiration regarding icons for AIDeveloper.

In particular I want to thank the group assistant Heike Neumann and the technical assistants Isabel Richter, Cornelia Liebers, Claudia Mrosowski, and Christine Schweitzer. They essentially kept the lab alive and were always open to help. I acknowledge my Thesis Advisory Committee members, Jochen Guck, Marius Ader and Stefan Gumhold for constructive criticism and encouragement.

I also want to thank Jochen Guck, Christian Schreiber, and the Department for Patent and License Management of the MPG and the TU Dresden for the support during application of the patent for image-based sorting. Furthermore, I'm very grateful that Jochen Guck together with Marius Ader applied for a grant which will allow me it to continue working on that project (DFG grant 399422891 to Marius Ader).

Last but not least I want to express gratitude to my family, friends and my girlfriend Nadja for their continuous and unprecedented love, help and support.

This work was financially supported by the Alexander von Humboldt Foundation (Alexander von Humboldt Professorship to Jochen Guck), the Deutsche Forschungsgemeinschaft (FZT 111) and the ProRetina Stiftung (to Marius Ader).

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Es fanden keine früheren erfolglosen Promotionsverfahren statt. Ich erkenne die Promotionsordnung der Fakultät Mathematik und Naturwissenschaften vom 23. Februar 2011 an. Ich habe diese Arbeit von Januar 2015 bis Januar 2020 unter der Betreuung von Prof. Dr. Jochen Guck am Biotechnologischen Zentrum der Technischen Universität Dresden angefertigt.

Dresden, 28. Januar 2020