Optimising High Resolution Measurements of Epidermal Growth Factor Receptor Oligomers in Cells Using Machine Learning Algorithms

Teodor Viktorov Boyadzhiev

Supervisors:

Simon Ameer-Beg

Marisa Martin-Fernandez

A thesis submitted to King's College London

in partial fulfilment of the degree of Doctor of Philosophy

Division of Cancer Studies

Faculty of Life Science & Medicine

King's College London

30 October 2018

Declaration

I hereby declare that the work presented in this thesis is the result of my own investigations and has not been accepted in a previous application for a degree.

Abstract

The epidermal growth factor receptor (EGFR) is a cell surface receptor, which controls cell growth and division. Mutations affecting the receptor expression could lead to cancer. Analysis of EGFR interactions with living cells requires measuring separations between 5 and 60nm. The separations are calculated by analysing time-series of diffraction limited spots, generated by labelled EGFRs. Finding such time-series manually is time consuming and non-reproducible. This project uses machine learning algorithms in combination with understanding of the data collection process and analysis requirements to optimise the data selection process, by automatically rejecting non-analysable time-series. The comparison to the manual process shows that the automated process significantly decreases the time required for data selection and decreases the uncertainty in the distance measurements.

In the last chapter of the thesis the refined data selection process is used as part of the analysis of EGFR experiments, which studies the effects of phorbol myristate acetate (PMA) on the receptor dimerisation. The results from the experiment show consistencies with data from electron microscopy and also that the PMA drives the EGFR to form smaller clusters as compared to the control experiment, however there is large uncertainty in the result due to limited data set.

Acknowledgements

I would like to express my gratitude to my colleague Dr. Michael Hirsch, and my supervisors Dr. Simon Ameer-Beg and Dr. Marisa Martin-Fernandez for their excellent advice, helpful guidance and patience during my studies.

I want to thank all members of Dr. Marisa Martin-Fernandez's group (Functional Biosystem Imaging team of the Central Laser Facility, Harwell Science and Innovation Campus) who provided all the help I needed to complete my work on this project.

Also I want to thank my wife Jiyang for all the support and encouragement during the hardest parts of this project, without whom I would not be able to complete my studies.

Authors Contributions

1. Laura C Zanetti-Domingues, Michael Hirsch, Christopher J Tynan, Daniel J Rolfe, Teodor V Boyadzhiev, Kathrin M Scherer, David T Clarke, Marisa L Martin-Fernandez, and Sarah R Needham. Determining the geometry of oligomers of the human epidermal growth factor family on cells with 7 nm resolution. *Progress in biophysics and molecular biology*, 118(3):139–152, 2015

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

activation function	In the context of artificial neural network activation function is the function applied to the activation potential of a node.	65
antibody	Any of a large number of proteins of high molecular weight that are produced normally by specialized B cells after stimulation by an antigen and act specifically against the antigen in an immune response, that are produced abnormally by some cancer cells, and that typically consist of four subunits including two heavy chains and two light chains	10
bias	In the context of artificial neural a bias is a constant value added to the weighed sum of the input connections to a unit before application of the activation function.	65
confusion matrix	A matrix which shows the counts of true positives, false positives, true negatives, and false negatives of a classifier.	58

connection	In the context of artificial neural network connection is an edge in a graph representing the structure of the network	65
diffraction limit	Minimum distance at which two point sources can be distinguished	9
diffraction limited spot	The pattern in an image created by one or more point sources which are closer together than the diffraction limit	10
dimer	A compound formed by the union of two molecules of a simpler compound	11
evanescent field	An electric and/or magnetic field that does not propagate as an electromagnetic wave but whose energy is spatially concentrated in the vicinity of the source	23
extrapolated	A feature which is not detected and is not part of the track. These features are assumed to exist and are before and after the track. When this is done the track is called extrapolated.	31
feature	In the context of bioimaging this refers to a diffraction limited spot. In the context of machine learning this term refers to a property of the classified objects.	21
feature space	The vector space associated with feature vectors	56

feature vector	An n dimensional vector of numerical features which represent an object	55
fluorophore	A chemical compound that can re-emit light upon light excitation.	11
frame	A microscope image made as part of ordered sequence of images	21
heterodimer	A protein composed of two polypeptide chains differing in composition in the order, number, or kind of their amino acid residues	10
hidden layer	All layers of neural network apart from the input layer and the output layer.	66
hidden unit	A unit from a hidden layer a the neural netowk.	66
homodimer	A protein composed of two polypeptide chains that are identical in the order, number, and kind of their amino acid residues	10
inhibitor	An agent that slows or interferes with a chemical action	10
input layer	The first layer of a neural network.	66
input unit	A unit from the input layer a the neural netowk.	66
interpolated	A feature which is part of a track however it is not detected.	31
level	A part of a track in which there is a constant number of active fluorophores and the intensity of the track is constant (apart from the noise).	34

level 1	The level in a track with the lowest intensity.	43
level 2	The level in a track with the second lowest intensity.	43
oligomer	A compound formed by the union of several molecules of a simpler compound	11
output layer	The last layer of a neural network.	66
output unit	A unit from the output layer a the neural netowk.	66
photobleaching	Irreversible loss of fluorescence upon continuous light excitation	21
photobleaching step	A sharp intensity change, separating two levels. It is not necessary to be due to photobleached fluorophore. It could just go to a dark state.	43
point source	A source of lights which is much smaller than the light's wavelength	9
track	An ordered collection of features	22
unit	In the context of artificial neural network a unit is a node in a graph representing the structure of the network	65

Acronyms

ADAM	Adaptive Moment Estimation	72
ANN	Artificial Neural Networks	65
AUC	Area Under the Curve	61
BIC	Bayesian information criteria	213
BM-I	Bisindolylmaleimide I	208
BM-I+PMA	BM-I and PMA	208
BSA	bovine serum albumin	209
CD	Contrastive Divergence	83
CDF	Cumulative Density Function	85
CHO	Chinese hamster ovary	209
CN	Condition Negative	58
CP	Condition Positive	58
EGF	Epidermal Growth Factor	209
EGFR	Epidermal Growth Factor Receptor	10
FFNN	Feed-Forward Neural Network	65

FLImP	Fluorophore Localisation Imaging with Photobleaching	9
FN	False Negative	58
FP	False Positive	58
FRET	Fluorescence Resonance Energy Transfer	11
KS	Kolmogorov-Smirnov	85
LDA	Linear Discriminant Analysis	41
MCMC	Monte Carlo Markov Chain	83
PBS	phosphate buffered saline	210
PCA	Principal Component Analysis	76
PFA	paraformaldehyde	210
PMA	phorbol myristate acetate	208
PN	Prediction Negative	58
PP	Prediction Positive	58
PSF	Point Spread Function	9
RBM	Restricted Boltzmann Machine	78
ROC	Receiver Operating Characteristic	61
ROI	Region of Interest	25
SHRImP	Single-molecule High-Resolution Imaging with Photobleaching	49

TIRF	Total Internal Reflection Fluorescence	23
TN	True Negative	58
TP	True Positive	58
WT EGFR	wild type epidermal growth factor receptor	208

Chapter 1

Introduction

Automation of repetitive tasks in data analysis can improve the efficiency of the process, reduce errors, and improve the reproducibility of the results. Removing manual steps from the analysis can reduce the need to train staff and hence allow better portability.

This project aims to reduce human involvement in Fluorophore Localisation Imaging with Photobleaching (FLImP) [2]. This technique is designed to measure fluorophore separations lower than the minimum resolvable distance of a fluorescence microscope in the native setting of cell membranes.

In an imaging system, an object which is much smaller than the wave length of the light is called point source. The function which describes the intensity of the image of a point source is called Point Spread Function (PSF). If the system is focused at the point source, the PSF is the Airy function, where the centre is the position of the point source. Therefore, a point source will be seen on the image as an Airy disk. This implies that there is minimum distance at which two point sources of light can be distinguished. The minimum resolvable distance, also referred to as diffraction limit, is given by

$$d = \frac{\lambda}{2\mathrm{NA}} \tag{1.1}$$

where λ is the wave length of the light, and NA is the numerical aperture [3]. For fluorescence microscope, typical values for the numerical aperture is NA $= 1.45$, and the wavelength of the light is between 488 and 647nm. Such system would have a diffraction limit between 168 and 223nm. When there are one or more point sources in an image which are closer to each other than the diffraction limit the pattern they create is called diffraction limited spot. The diffraction limited spot can be described as a sum of one or more Airy disks.

One of the applications of FLImP is extracting information about the Epidermal Growth Factor Receptor (EGFR). EGFR a trans-membrane receptor which is responsible for cell growth and division [4]. It is a member of the ErbB family receptors which comprises of HER1 (EGFR), HER2, HER3 and HER4. All of these receptors have an extracellular binding domain, a single transmembrane domain, and an intracellular domain [5]. In presence of a ligant the receptors form homodimers or heterodimers. The dimerisation of the receptor has been confirmed by crystallography [6] [7] [8] [9]. There are ligands for HER1, HER3, and HER4, and there are no known ligands for HER2. HER2 however, is the preferred receptor for heterodimerisation [10].

Members of the ErbB family are believed to be responsible for many forms of cancer, as overexpression or mutation of the receptors is observed in these cancers [11]. In breast cancer there is overexpression in 60% of the cases, in head and neck cancer in over 80% of the cases, and around 60% of the cases in lung cancer [5].

In cancer cells the members of the ErbB family are in activated states. Therefore, the drugs aim to disrupt the signalling chain which the active receptor triggers. There are two types of drugs targeting EGFR. They are monoclonal antibodies and tyrosine kinase inhibitors. The antibodies bind with the receptor without causing dimerisation and prevent the ligand from binding with the receptor. The tyrosine kinase inhibitors act on the intracellular domain of the receptor and disrupt the signalling chain. These drugs have limited success and the cancers develop resistance to them [12].

The limited success of the drugs can be due to insufficient understanding of the EGFR behaviour. X-ray crystallography requires the over-expression and purification away from the biological context [13]. This method provides accurate information about the structure of the protein, but it is believed that contextual information, such as the location on the membrane, whether there are clusters, and whether the protein forms oligomers, is critical for developing effective treatments. Therefore, techniques, such as FLImP, are needed to extract information about protein complexes in native membrane settings [14] [15] [16].

Several studies using various microscopy techniques have suggested that there are clusters of EGFR in cancer cells before and after introduction of the activation signal [17] [18] [19] [20].

The FLImP technique was developed to find more information about the EGFR clusters and to investigate whether larger oligomers exist. This is done by labelling the EGFR complexes with fluorophores and measuring the distance between these labels. Complexes with larger size are evidence for larger oligomers. Therefore, FLImP targets distances between 5 and 60 nm, which are too small for super-resolution microscopy and too large for Fluorescence Resonance Energy Transfer (FRET). FLImP has found evidence suggesting more complex structures than dimers [21] [22].

1.1 Super-Resolution Microscopy

The field of super-resolution microscopy aims to measure structures smaller than the diffraction limit of the visible light. Typically this is done by using chemical compounds, called “dyes”, which can absorb light in some wavelength and then emit light in different wavelength. Then the dyes are excited with a laser, and their activation is controlled so that some analysis method can estimate the dye’s positions. The super-position of many dyes can reveal details about biological structures which would be invisible if traditional

microscope is used.

Beyond visualisation, the dyes can be used to measure sizes of macro-molecule structures, track proteins and determine whether they interact with one another, and so on.

STED

One of the first developments of super-resolution microscopy is a technique is called stimulated emission depletion (STED) microscopy. The technique was proposed by Hell and Wichmann [23] in 1994 and then experimentally tested by Hell and Klar in 1999 [24]. This idea was also independently patented in 1986 by Okhonin [25] (patent SU 1374922) who worked in the Institute of Biophysics, USSR Academy of Sciences. It is believed that the patent was unknown to Hell and Wichmann at the time of development of the technique.

STED microscopy works by using two light beams, an excitation and a depletion beam. The excitation beam, which is Gaussian shaped, causes fluorescence and the depletion beam, which is doughnut shaped, depletes the excited state and, thereby, the fluorescence observed. The two beams are arranged in a concentric configuration, which allows fluorophores in the central region to fluoresce, while keeping the near-by fluorophores in a non-emitting state. The beams scan the sample exciting individual fluorophores consecutively and suppressing nearby fluorophores. Finally the image is reconstructed from the super-position of all of the fluorophores. The new resolution can be expressed by the modified Abbe equation [26],

$$d = \frac{\lambda}{2\mathrm{NA}\sqrt{1+\sigma}} \tag{1.2a}$$

$$\sigma = \frac{I_d}{I_e} \tag{1.2b}$$

where I_d is the maximum intensity of the depletion beam, I_e is the intensity of the excitation beam, λ is the wavelength, and NA is the numerical aperture.

This method is a member of wider class of methods which work on the principle of reversible saturable optical fluorescence transitions (RESOLFT) [27] [28].

SOFI

Frequently the super-resolution methods require special microscope set-ups and might have restrictions on what kind of dyes are used. The type of dye could interfere with the experiment. Therefore, it might be useful to have a method which doesn't impose such restrictions. A development by Dertiger et al. from 2009 provides super-resolution without imposing such restrictions [29]. The development is called super-resolution optical fluctuation imaging (SOFI). SOFI estimates the auto-correlation of the natural fluctuation of the fluorophore emissions for each pixel to find the centroids of multiple overlapping fluorophores. The paper has reported a resolution of 55nm. However, it requires analysis of series of frames acquired over long period of time, which makes it unsuitable for imaging of fast changing processes.

Another super-resolution development which doesn't require special microscope set-up or specific fluorophores is uper-resolution radial fluctuations (SRRF), pronounced "surf", [30]. This method finds correlation between radial fluctuations. It is able to work in low signal to noise conditions and with different microscopes, such as widefield, confocal or TIRF. SRRF achieved a resolution spatial resolution of 60 nm and temporal resolution of 1s on live using conventional fluorophores and low intensity light. The authors also provided a GPU enabled ImageJ plugin.

Single Molecule Localisation

A widely used class of super-resolution methods is single molecule localisation microscopy (SMLM) [31] [32]. These methods can achieve very high resolutions, which is only limited by the signal to noise ratio of the microscope. The theoretical limit of the precision is called Cramér–Rao lower bound [33], which can be expressed as

$$\sigma_{loc} \geq \frac{\sigma_0}{\sqrt{N}} \tag{1.3}$$

where σ_{loc} is one standard deviation of the error of the x, y coordinates, σ_0 is the standard deviation of the PSF when it is modelled by a 2D Gaussian function, and N is the number of detected photons. These methods can achieve very high resolution, between 10 and 30 nm [34] [35]. However, they require long imaging times and therefore can only capture very slow changing processes.

These methods work by temporally isolating individual emitters and the fitting 2D Gaussian function to them to find the centroid. This can be done trough least squares optimisation [36] or maximum likelihood estimator [37], which is more accurate [38]. These approaches are slow due to the iterative algorithms. There is a way to estimate the centroids of the PSF, based on centre of mass (QuickPALM) [35].

One of the most popular techniques for single molecule super-resolution microscopy is to stochastically switching on just few emitters at a time. If the emitters are far enough from each other a Gaussian profile can be fitted for each of them to estimate the position of the emitter with high accuracy. In the next frame another set of emitters is switched on and the process is repeated. After enough iterations the image can be reconstructed by superimposing all of the centroids. This method is called stochastic optical reconstruction microscopy (STORM) and it is developed by Rust et al. in 2006 [39]. In the development by Rust pairs of dyes Cy3-Cy5 are used and they are switched on with waves of green and red light. The original paper demonstrated a resolution of 20nm.

Since STORM imposes restrictions on the fluorophores used for the imaging, it is possible these fluorophores to interfere with the experiments due to toxicity. Therefore a simplified version of STORM is developed, which can work with large variety of fluorophores, such as Alexa Flour 647, instead of fluorophore pairs and requires only one laser source [40] [41] [42]. These methods are known as direct STORM (dSTORM).

The dSTORM can also achieve spatial resolution as low as 20nm [43].

Another similar development by Betzig et al. in 2006 is called photo-activated localization microscopy PALM [31]. This technique uses photoconvertible proteins to switch them from emitting one wavelength, for example 516nm, to another wavelength, for example 581nm. During the conversion not all of the proteins are switched, but just small, well separated subset of proteins. This allows for easy estimation of their centroids and integration over time to build a super-resolution image. The PALM microscopy was able to image the Z-ring in E. coli with spatial resolution of 35nm [43].

3D Single Molecule Localisation

Frequently in biology the three dimensional information is important. Therefore, the SMLM methods have been extended to extract also depth information about the emitters. This can be done by PSF engineering [38] [34]. This technique would modify the optical system so that the PSF would have specific distortion when the light source is below or above the focal plane. Such techniques include astigmatism [44], double-helix PSF[45], phase ramp [46].

The astigmatism is achieved by adding cylindrical lens to the optical system, which will cause the PSF to become elongated along the X or the Y axis, depending whether the emitter is below the focal plane or above. Huang et al. achieved depth resolution of 67 nm along depth of 3 µm, by combining multiple acquisitions at different depths.

Pavani et al. used optical modification to achieve the double helix PSF, which results in two visible lobes. Emitters at different depth cause the lobes to be rotated around their midpoint, which is used to achieve lateral resolution of 20 nm and axial resolution of 50 nm over a z range of 2 µm.

Baddeley et al. use linear phase gradient placed between the two halves of the objective pupil plane, which causes the PSF to split into two lobes whose position relative to each other depend on the depth of the emitter [46]. This technique achieved

resolution of 20 nm in the x direction, 16 nm in the y direction, and 42 nm in the z position.

Another technique is the BiPlane detection, used with FPALM, which uses two planes to analyse the changes of the PSF in each plane [47]. This technique achieved 30 nm lateral resolution and 75 nm axial resolution. It has advantage over other techniques such as astigmatism, that the resolution doesn't decays as much with the increased depth.

Aquino et al. use 4Pi microscope to achieve less than 10 nm resolution along all three axis, over depth of 650 nm [48].

Bourg et al. created super-critical angle fluorescence recovery [49], which uses evanescent field produced by total internal refraction of light between two mediums with different optical density to excite fluorophores. The exponential decay of the evanescent field causes fluorophores at different depths to have different intensity, which can be used to extract their depth. This methods is combined with dSTORM and achieves 20nm axial resolution over depth of 150nm under the coverslip.

Other techniques include dual-objective interferometry [50], which achieves sub-20 nm, and Zernike optimized localization approach in 3D (ZOLA-3D) [51], which achieves 32 - 40nm laterally and 36nm axially.

A study by Xu et al. in 2012 combined astigmatism with a dual-objective scheme. They achieved around 9 nm lateral resolution and 19 nm axial resolution over depth of 150nm with the fluorophores Alexa Fluor 647.

Multi-Emitter MLSM

Techniques such as STORM require that there is only single emitter within a diffraction limited spot so the position of the emitter can be reliably estimated [52]. Therefore, each diffraction limited spot needs to be tested causing a lot of data to be rejected. To analyse diffraction limited spots generated by multiple emitters a different class of methods needs to be employed.

One way analyse a diffraction limited spot, generated by multiple emitters, is to fit a model such as Gaussian mixture model.

An example of such methods SHRImP [53], which was developed in 2004. This method works by collecting frames of the diffraction limited spot until all of the fluorophores are photobleached. Then the frames in which there is just one fluorophore and the frames in which there are just two fluorophores are selected manually. Then a mixture of two Gaussians,

$$I_{pre}(x,y) = Ae^{-\frac{(x-x_0)^2}{w_{x1}^2}-\frac{(y-y_0)^2}{w_{y1}^2}} + Be^{-\frac{(x-x_0+\delta_x)^2}{w_{x2}^2}-\frac{(y-y_0+\delta_y)^2}{w_{y2}^2}} + z_0 \tag{1.4}$$

is fitted to the frames with two fluorophores, and a single Gaussian is fitted to the frames with one fluorophore,

$$I_{post}(x,y) = Ae^{-\frac{(x-x_0)^2}{w_{x1}^2}-\frac{(y-y_0)^2}{w_{y1}^2}} + z_1 \tag{1.5}$$

simultaneously. In these formulas w_{x1}, w_{y1}, w_{x2}, and w_{y2} are the widths for the x, and y components of the Gaussians for each emitter, z_0, and z_1 are the background intensities for each set of images, A, and B are the intensities of each emitter, and δ_x, and δ_y are the offsets for the second molecule, relative to the first one. SHRImP can achieve 5 nm precision in measuring separations between molecules and has been successfully used in biological studies [54] [55].

Simsonson et al. extended SHRImP in 2011 to generalized single molecule high-resolution imaging with photobleaching (gSHRImP) which can handle multiple fluorophores [40].

Another similar technique is nanometer-localized multiple single-molecule (NALMS) presented by Qu et al. [56] in 2004. This technique also uses two dimensional Gaussian fit to evaluate difference between the centroids. It uses TIRF to illuminate the fluorophores (Cy3) and manages to achieve 2.5nm precision on DNA rulers.

Similar technique to SHRImP and NALMS is Fluorophore Localisation Imaging with

Photobleaching (FLImP) [2] which can provide also confidence intervals around the separations, based on bootstrapping.

FLImP selects a set of frames which have two fluorophores and frames which have only one fluorophores. Then it treats them as independent and identically distributed measurements of the intensity profile of the diffraction limited spot. Then the set is resampled 1200 times, the bootstrapping technique, and for each sample a mixture of Gaussians is fitted. From each fit a separation can be calculated, which results of a set of 1200 samples of the separation between the two molecules. Then a Rice distribution can be fitted to the separation samples to provide a probability distribution of a separation, given the input data.

The FLImP technique is used for measuring the size of EGFR oligomers in fixed cells [57] [22] [1]. With the correct setup FLImP can achieve a separation measurement with confidence interval (smallest interval containing 67% of the probability density) of 7nm.

A different approach to Gaussian mixture fitting is subtraction of point spread functions. This technique would subtract the frames from the time interval when there is one fluorophore from the frames when there are two fluorophores to obtain the position of the second fluorophore. Then this is repeated for 3 fluorophores and so on.

An example of PSF subtraction microscopy is bleaching/blinking assisted localization microscopy (BaLM), developed Burnette et al in 2011 [58]. This technique takes a series of images of a diffraction limited spot and subtracts each image from the previous. At time points of photobleaching the subtracted image would have an intensity profile consistent with PSF function from single source. Then a Gaussian profile is fitted to the subtracted images at the points of photobleaching to estimate the positions of individual fluorophores. This method managed to achieve resolution of 48nm.

These techniques measure the fluorophore positions over large time intervals to produce measurements with low confidence interval. This requires the cells to be immobilised and the data to be corrected for the global drift of the sample. This would limit

the use of such techniques for processes which are slow. Another source of error is fluorophores which are at different depth, in which case the technique would underestimate the separation.

On theory it is possible to calculate depth from the evanescent decay, however if fixed cells are imaged the cell membrane would touch the coverslip some places and other it won't, making it impossible to know the optical density of the medium and therefore calculate the depth. It might be possible to use PSF engineering such as astigmatism to extract some depth information.

Chruchman et al. tried to address the temporal resolution limitations of the multi-emitter SMLM methods by introducing single-molecule high-resolution colocalization (SHREC) in 2004 [59]. This method uses two colour dyes (Cy3 and Cy5) to label two ends of a molecular complex. Then each to colour is fitted a Gaussian function to find the centroid. This allows to measure small distances with resolution of under 10nm.

Probabilistic Single Molecule Localisation

It is possible to use probabilistic model to describe the microscope data and calculate the most likely centroids of the emitters.

An example of such approach is developed by Huang et al. in 2011 [52], which uses maximum likelihood to find the most likely centroids. The PSF is modelled as a Gaussian function,

$$PSF(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma_0^2}} \tag{1.6}$$

where σ_0 is the size of the PSF in pixels. Assuming that the pixels are rectangular, the expected photons per pixel can be calculated by integrating Formula 1.6. Then the expected photon counts at pixel k with coordinates (x, y) can be expressed as

$$\mu_k(x,y) = I_0 \Delta E^{(x)}(x,y) \Delta E^{(y)}(x,y) + b_0 \tag{1.7a}$$

$$\Delta E^{(x)}(x,y) = \frac{1}{2}\left(\operatorname{erf}\left(\frac{x - x_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}\right) - \operatorname{erf}\left(\frac{x - x_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}\right)\right) \tag{1.7b}$$

$$\Delta E^{(y)}(x,y) = \frac{1}{2}\left(\operatorname{erf}\left(\frac{y - y_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}\right) - \operatorname{erf}\left(\frac{y - y_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}\right)\right) \tag{1.7c}$$

where x_0, y_0, are the centroid coordinates, I_0 is the expected intensity at the centre of the PSF, and b_0 is the background.

However, this model needs to allow for multiple emitters, therefore Formula 1.7a is redefined as

$$\mu_k(x,y) = b_0 + I_0 \sum_{n=0}^{N} E_n^{(x)}(x,y)\, E_n^{(y)}(x,y) \tag{1.8}$$

Having a model for the expected number of photons for each pixel then the probability of the number of photons for each pixel can be expressed with the Poission distribution,

$$P(D|\boldsymbol{\theta}) = \prod_{k=1}^{K} \frac{\mu_k(x,y)^{d_k} e^{-\mu_k(x,y)}}{d_k!} \tag{1.9}$$

where k iterates over all of the analysed pixels, d_k is the photon count observed at pixel k, and $\boldsymbol{\theta}$ is the parameters such as the centroids. From this model the log-likelihood is calculated and maximised using the Newton-Raphson method.

Another development by Cox et al. 2012 uses Bayesian analysis to find the emitter positions [60]. They model the entire data set as a set of emitters and the model allows for blinking and bleaching events. The method is based on factorial hidden Markov model. For inference a hybridized method, forward algorithm and Monte Carlo Sampling (MCMC), was used. This method achieved spatial resolution of 50 nm and temporal resolution of 4s.

A method by Tang et al. addresses the issues with false positive emitter detection in SMLM parameter free solution, called Auto-Bayes [61]. All single molecule localisation algorithms need to decide whether a diffraction limited spot is detected or the image

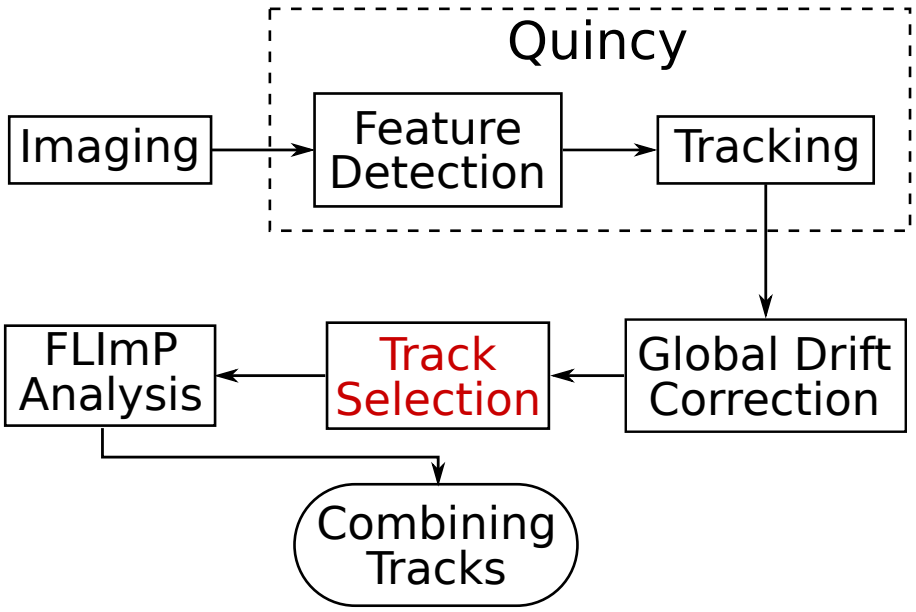


Figure 1.1: The FLImP pipeline.

contains just background noise. Frequently such methods require user-defined or hard-coded thresholds. Auto-Bayes tries to derive this threshold by using a Bayesian inference and was tested on a simulated dataset provided by ISBI online challenge in 2015.

1.2 FLImP Overview and Project Objectives

FLImP [22] measures distances lower than the diffraction limit by observing the change of the intensity profile of a diffraction limited spot during fluorophore photo-bleaching. This process can be represented as a flowchart which has 7 stages, Figure 1.1.

The first stage of the FLImP method is imaging, which uses fluorescent microscopy to produce series of images, also called frames, of diffraction limited spots. Each diffraction limited spot can be generated by one or more fluorophores and is referred to as feature in this project.

The second and third stage of the FLImP method are handled by Quincy [62], which

detects the features, "Feature Detection" and tracks them over time, "Tracking", to create ordered collections of features, called tracks.

The next stage of the FLImP method is "Global Drift Correction". During this stage frame ranges in which the drift of the sample can be corrected are selected. This information is used in the following stage called "Track Selection", in which suitable tracks are selected to be analysed by FLImP.

After suitable tracks have been selected they are analysed by FLImP and a probability distribution of fluorophore separation is produced, using the bootstrapping method [63].

In the last stage of FLImP the bootstrap samples of all the tracks are combined to produce the final result. Normally this is probability distribution with several peaks. Each peak corresponds to a fluorophore separation, which describes a type of oligomer.

This project presents further developments on the "Track Selection" process, which is shown in red on Figure 1.1. The rest of the processes are under the management of the Octopus group which is part of Science and Technology Facilities Council, and they were not modified during this project.

Prior to the work presented in this thesis, the track selection was primarily undertaken by manual processes. The aims of the project is to reduce the manual involvement, which will improve the reproducibility of the results, reduce the human error, and increase the efficiency of the method.

Chapter 1 gives overview of super-resolution methods and describes the FLImP method as it is used to analyse EGFR in fixed cells. Each analysis step is discussed alongside with limitations. Chapter 2 describes the analysis methods and techniques used throughout the thesis. Chapter 3 describes the simulations used to verify that the analysis methods works as expected against the ground truth. In Chapter 4 are presented and compared several algorithms which find time intervals in which the diffraction limited spots have constant intensity. In Chapter 5 the manual track selection is evaluated and in Chapter 6 a track selection process with high degree of automation is presented. Then

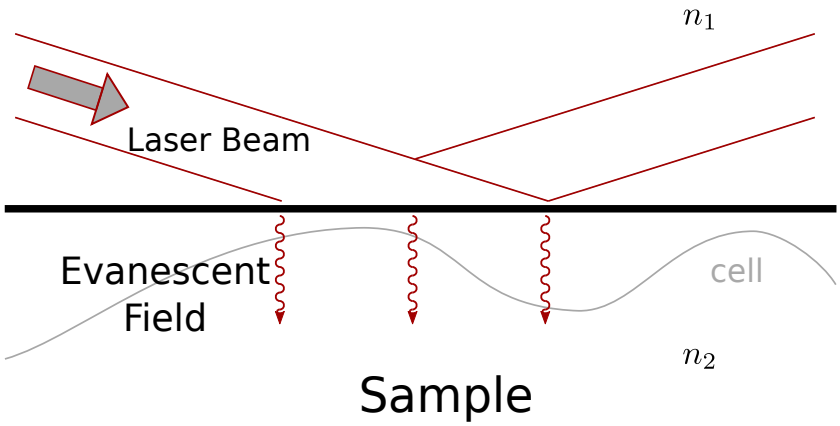


Figure 1.2: TIRF microscopy works by using the evanescent field at the point of total refraction to excite fluorophores close to the sample surface. The sample typically is a fixed cell with labelled EGFR

in Chapter 7 the refined track selection method was used as a part of the FLImP method to explore the dimerisation of EGFR under different treatments. In the last chapter are summarised the main conclusions from the thesis.

1.3 Imaging

The imaging stage of the FLImP method collects sequence of images of fixed cells. Each image, or frame, has exposure time of few hundred milliseconds.

The imaging technique which the FLImP [2] method uses is Total Internal Reflection Fluorescence (TIRF) [64]. The aim of this technique is to illuminate only the surface of the sample, typically several hundred nanometers, and therefore reduce the fluorescence coming from fluorophores deeper in the cell. This allows to be achieved better signal to noise ratio and hence measurements with higher accuracy[65].

TIRF works by shining a laser beam at large enough angle to achieve total internal reflection. At the point of the reflection there is an evanescent field which has decreasing intensity with the depth, Figure 1.2.

The intensity of the evanescent field decreases exponentially with the depth, z, according to

$$I(z) = I_0 e^{-\frac{z}{d}}, \tag{1.10}$$

where I_0 is the intensity at the point of refraction, the decay coefficient, d, is given by

$$d = \frac{\lambda}{4\pi n_1 \sqrt{\sin(\theta)^2 - \frac{n_2}{n_1}}}, \tag{1.11}$$

and n_1 is the refractive index of the substrate, n_2 is the refractive index of the liquid medium, θ is the laser angle, and λ is the wavelength of the laser[66]. Normally the refractive index of the substrate is larger than the refractive index of the sample, $n_1 > n_2$.

The exponentially decaying evanescent field excites fluorophores which are only several hundred nanometers away from the sample surface. In general it is possible to calculate the fluorophore depth from its intensity, however it is very hard to predict the refractive index of the sample, n_2. This is because when live cells are imaged the membrane touches the glass at some places and other places there are pockets of the liquid medium, hence the refractive index is non-uniform across the field of view. Therefore different techniques should be used to determine the depth of the fluorophores.

One possible way to extract depth of the fluorophore is to use PSF modification, which encodes a z position. Since the FLImP experiments target a layer with thickness of few hundred nanometers, astigmatism can be used. With this technique resolution in the z axis between 10nm and 12nm has been achieved [67], [68] [69]. Another point spread function which can be used to determine the z position is the double helix [70].

The resolution of all these methods depends on the number of detected photons. There are other PSF designs as well and they have different advantages and disadvantages [71].

1.4 Feature Detection and Tracking

The FLImP analysis requires a diffraction limited spot to be detected and tracked until all of the emitters which generated it transition to a dark state. Therefore, efficient and reliable algorithm is required for the FLImP analysis tool-chain.

The feature detection has been approached by Sergé et al. using maximum likelihood method and peak subtraction [72]. Different way is to address the problem of detection and tracking simultaneously by using Monte Carlo Bayesian approach [73], [74], [75]. The advantage of such approach is that it takes advantage of spatial and temporal prior.

In FLImP the single molecule detection is addressed by using Bayesian evidence-based feature detection, which is more reliable than maximum likelihood approach. For tracking is used a simpler method, based on nearest neighbour in time and space. This combination of these techniques allows Quincy to detect and track large number of features over multiple channels faster than fully Bayesian solutions and more reliably than simple maximum-likelihood methods.

1.4.1 Feature Detection

Feature detection in FLImP is the task of finding diffraction limited spots, also called features, and calculating their properties, $\boldsymbol{\phi} \in \mathbb{R}^5$. These properties are feature intensity, I, error in the feature intensity $\sigma(I)$, x position, y position, and background intensity, b. Each feature is generated by one or more fluorophores.

In FLImP feature detection is done by Quincy [62], which is based on earlier work by Hobson [76]. Quincy detects features by comparing the probability of two hypothesis, given the observed data in a Region of Interest (ROI) around each pixel. The first hypothesis, H_1, is that there is a feature together with a background noise in the ROI, and the second hypothesis, H_2, is that there is only background noise in the ROI around a pixel, Figure 1.3. The ROI for a pixel is a square centred at that pixel with side equal

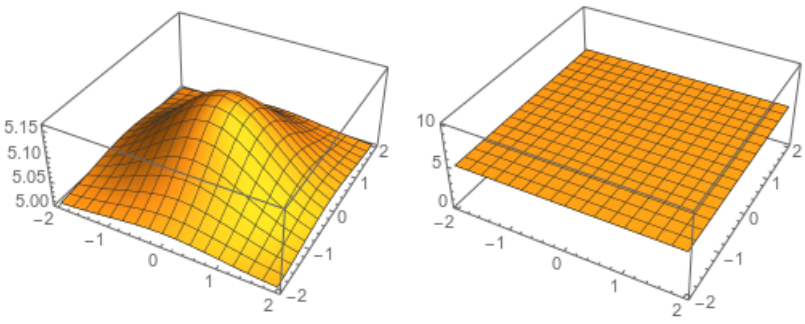


Figure 1.3: Quincy compares two hypothesis to detect a feature. The first model (H_1, left) is a uniform background plus 2D Gaussian profile, which describes a feature, and the second model (H_2, right) is just uniform background and describes absence of a feature.

to four times the width of the point spread function at half of the maximum intensity.

For each pixel in the image the probability of H_1 is compared to the probability of H_2. This can be expressed by the inequality

$$\frac{P\left(\boldsymbol{d}|H_1\right)}{P\left(\boldsymbol{d}|H_2\right)} > \frac{P\left(H_2\right)}{P\left(H_1\right)} \tag{1.12}$$

where $\boldsymbol{d} \in \mathbb{R}^n$ is a vector composed by the pixel values in the ROI. The data vector, $\boldsymbol{d}$, has $n = (4d)^2$ dimensions, where d is the diameter of the point spread function at half of the maximum intensity. If the inequality holds, then the pixel is a part of a diffraction limited spot. The probability of the observed data given H_1, feature plus background, is

$$P(\boldsymbol{d}|H_1) = \int\int P(\boldsymbol{d}|I, b_1, H_1) P(I, b_1|H_1) dI db_1 \tag{1.13}$$

where I is the intensity of the detected feature and b_1 is the intensity of the background. The probability of the observed data given b_2, just background, is

$$P(\boldsymbol{d}|H_2) = \int P(\boldsymbol{d}|b_2, H_2)P(b_2|H_2)db_2 \tag{1.14}$$

where b_2 is the intensity of the background.

The joint probability of feature intensity and background intensity given hypothesis H_1, is $P(I, b_1|H_1) = \pi_I(I)\pi_B(b_1)$. The probability of background intensity given hypothesis H_2, is $P(b_2|H_2) = \pi_B(b_2)$. The probabilities $\pi_I(I)$ and $\pi_b(b)$ are uniform between 0 and parameters $I_{\max}$ and $B_{\max}$.

$$\pi_I(I) = \begin{cases} \frac{1}{I_{\max}} & \text{if } 0 \leq I \leq I_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{1.15a}$$

$$\pi_b(b) = \begin{cases} \frac{1}{b_{\max}} & \text{if } 0 \leq b \leq b_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{1.15b}$$

After the position of a feature is determined Quincy can evaluate the most probable feature intensity I together with the confidence interval, $\sigma(I)$, around that intensity, using Bayesian parameter estimation. It also calculates the most probable background intensity, b_1.

Quincy also estimates the error of the measurement of intensity in each pixel $\sigma(I(x, y, t))$, which is different from the error on the feature intensity $\sigma(I)$. Quincy approximates the distribution of the pixel intensity with Gaussian distribution, and it estimates the standard deviation, $\sigma(I(x, y, t))$ in each pixel with coordinates (x, y) at time t which has intensity $I(x, y, t)$. This is done by smoothing the image with normalised square box smoothing kernel with size of 20 pixels.

$$I_{\text{smooth}}(x, y, t) = \frac{1}{441} \sum_{x'=x-10}^{x+10} \sum_{y'=y-10}^{y+10} I(x', y', t) \tag{1.16}$$

Then the smoothed image is subtracted from the original image, after which for each

pixel is calculated the error, $\sigma(I(x,y,t))$, as the mean square in an square box around the pixel.

$$\sigma(I(x,y,t)) = \frac{1}{441} \sum_{x'=x-10}^{x+10} \sum_{y^{prime}=y-10}^{y+10} (I(x',y',t) - I_{\text{smooth}}(x',y',t))^2 \qquad (1.17)$$

Assumptions and Limitations

The feature detection algorithm makes several assumptions. The first assumption is that the point spread function follows a Gaussian profile. However, as we know, it follows the Airy function. This assumption is shown not to influence the position calculation [53].

A diffraction limited spot may be generated by several fluorophores. This means that the true profile of the feature will be sum of Airy functions, which can be approximated by the sum of Gaussian functions. The side effect of this is that Quincy does not calculate a position of a fluorophore but an average position of all fluorophores. When one of the fluorophores photobleaches there is a shift in the position of the feature, until there is only one active fluorophore and the position of the feature is the position of the that fluorophore. The positions of the individuals fluorophores are calculated later by FLImP analysis.

Another assumption which Quincy makes during the feature detection is that the background is spatially uniform. When there is non-uniform fluorescence in the background Quincy could underestimates the feature intensity and overestimates the background intensity. The background fluorescence is inspected during track selection and tracks which have non-uniform background fluorescence are excluded from FLImP analysis.

Quincy also assumes that the noise in each pixel follows Gaussian distribution, however the experimental noise has components of Poisson distribution and specific detector properties. This means that the error in feature intensity is also approximated. Develop-

ing more accurate noise model can improve the accuracy of the parameters, calculated by Quincy such as feature intensity, I, and the confidence interval of the intensity $\sigma(I)$. This assumption also affects the algorithm, which decides whether there is change in the feature intensity or not.

The feature detection and the parameter estimates also depend on the thresholds $I_{\max}$, and $B_{\max}$ and the assumption of uniform probabilities π_I, and π_B. These thresholds need to be set manually. Also it is possible for the probability of background intensity π_B and feature intensity π_I, given a hypothesis, to be best described by different distribution from the uniform.

1.4.2 Tracking

After the features are detected in all frames, they are tracked over time by Quincy to create tracks.

In the first frame a track, τ_i, is created for each feature, $\phi_i^1 \in \mathbb{R}^5$. In every consecutive frame, f, a connection probability $P\left(\phi_i^f \middle| \tau_j\right)$ is calculated for each pair of feature, i, and track, j, based on the distance in space and time to the last detected feature in the sequence j, Figure 1.4. The probability is calculated using the formula

$$P\left(\phi_i^f \middle| \tau_j\right) = \begin{cases} N\left(\Delta p_{ij}, \Delta t_{ij} | a, b\right) & \text{if } \Delta p_{ij} < a \text{ and } \Delta t_{ij} < b \\ 0 & \text{otherwise} \end{cases} \tag{1.18}$$

$$N\left(\Delta p_{ij}, \Delta t_{ij} | a, b\right) \propto e^{-\left(\frac{\left(\Delta p_{ij}\right)^2}{a^2} + \frac{\left(\Delta t_{ij}\right)^2}{b^2}\right)} \tag{1.19}$$

$$\Delta p_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{1.20}$$

where (x_i, y_i) is the position of feature i, (x_j, y_j) is the position of the last detected feature of track j, Δt_{ij} is the difference in time between the frame in which feature i is detected and the last frame in which the sequence j has a detected feature, and a and

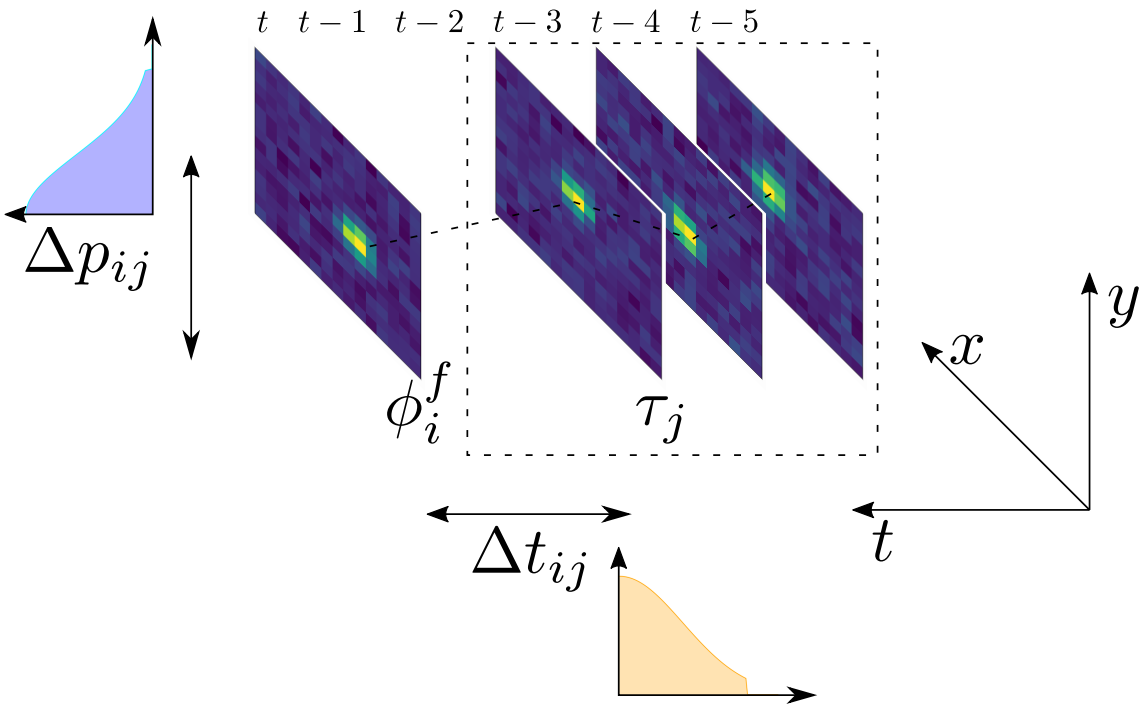


Figure 1.4: Quincy builds tracks by maximising the probability of a feature to belong to a track, given distance, in space and time, to the last feature in that track

b are user defined thresholds.

If the probability is 0, then a new sequence is created, otherwise the feature is assigned to the sequence where the probability is the highest.

Assumption and Limitations

This tracking algorithm relies on the assumptions that fluorophores travel small distance between the duration of each frame. It also does not assume any model of motion. In FLImP fluorophores are required to be fixed, therefore any motion of the features should be caused by either noise in the system, by a fluorophore photobleaching, which will cause shift in the feature position, or by the global drift in the sample.

Since the tracking algorithm does not use model of fluorophore movement it cannot distinguish between a fluorophore jumping from one position to another due to some

vibration and one fluorophore photobleaching and another coming out of a dark state nearby. Therefore the analysis would work better if precautions are taken to ensure that the sample is stable. In such case calculation of the threshold for the maximum distance, a, between two consecutive features in a track should be based on the system noise and the size of the point spread function.

Another limitation is that the probability of feature disappearing from a track for several frames is modelled as a truncated Gaussian distribution. Setting the threshold, b, would not be a straight forward task. The tracking algorithm could benefit from more explicit modelling of that probability.

There are two reasons that a track would have several frames without a detected feature. The first reason is that the feature detection algorithm fails to detect a feature, an instance of false negative (for more information on false negatives check Section 2.1.1). The second reason is that a feature is generated by only one fluorophore which has gone to a dark state. In both cases when the feature is detected again it should be regarded as part of the same track. The probability of false negative and information about the state transition model of the fluorophores can be used to model the probability of a fluorophore belonging to a track after several frames.

Interpolation and Extrapolation

When there are some gaps between detected features in a track, the position of the feature in the gap is assumed to be straight line between the last detected feature before the gap and the first detected feature after the gap. Then the intensity of the feature is calculated the same way as when the feature is detected. The feature in these frames is termed interpolated.

It is also possible to use the position of the first and the last detected feature in the track as a reference point to interpolate the features before and after the track. In this case the track is extrapolated. The intensity of the extrapolated features should be zero,

because the background is subtracted already. If it is different from zero it is possible that there is a feature before or after the track which is not detected or that background intensity is overestimated, usually due to additional fluorophore in close proximity to the track.

1.5 Global Drift Correction

During FLImP experiment the sample moves due to temperature changes, vibrations and other reasons. However, the model used in the FLImP analysis assumes that the fluorophores do not move. Therefore it is necessary to account for the instrumental drift during evaluation of the fluorophore separations.

The first step of the global drift correction is to evaluate the position of the sample for each frame. The algorithm which evaluates the sample position in each frame is described with Pseudocode 1.

The second step of the drift correction is to fit a model to the position estimates. The model is a polynomial of second degree for the position x and y

$$X(t) = at^2 + bt + c \tag{1.21a}$$

$$Y(t) = at^2 + bt + c \tag{1.21b}$$

The model of the motion aims to capture drift caused by the thermal expansion of the sample. A linear model would not be able to account for slow down of the sample drift when thermal equilibrium is reached, since it assumes constant speed. A polynomial of second degree will be able to capture the slowdown of the sample drift as thermal equilibrium is reached. A higher order polynomial is not used, because it is believed that sample drift caused by sources different from the thermal expansion can be avoided.

input : List tracks $\tau = \left\{\tau^{(n)}\right\}$

output: List of positions $p = \{(x_t, y_t)\}$

$\tau \leftarrow \texttt{SortTrackByLengthDecreasing}(\tau)$

$\mathcal{F} \leftarrow$ frame ids of track $\tau^{(1)}$

${x_{\text{offset}}}^{(1)} = \frac{1}{\texttt{Length}(\mathcal{F})} \sum_{t \in \mathcal{F}} \tau^{(1)}(x)_t$

${x_{\text{offset}}}^{(1)} = \frac{1}{\texttt{Length}(\mathcal{F})} \sum_{t \in \mathcal{F}} \tau^{(1)}(y)_t$

$\mathsf{xAVG}_t = \tau^{(1)}(x)_t - {x_{\text{offset}}}^{(1)}\,,\ t \in \mathcal{F}$

$\mathsf{yAVG}_t = \tau^{(1)}(y)_t - {x_{\text{offset}}}^{(1)}\,,\ t \in \mathcal{F}$

$\mathsf{nVALS}_t = 1 \quad \mathsf{t} = \mathsf{firstId}\ ..\ \mathsf{lastId}$

for $n \leftarrow 2$ *to* $\texttt{Length}(\tau)$ **do**

$\mathcal{F} \leftarrow$ frame ids in both $\tau^{(n)}$ and xAVG

${x_{\text{offset}}}^{(n)} = \frac{1}{\texttt{Length}(\mathcal{F})} \sum_{t \in \mathcal{F}} \tau^{(n)}(x)_t - \mathsf{xAVG}_t$

${y_{\text{offset}}}^{(n)} = \frac{1}{\texttt{Length}(\mathcal{F})} \sum_{t \in \mathcal{F}} \tau^{(n)}(y)_t - \mathsf{xAVG}_t$

$\mathsf{xAVG}_t = \frac{1}{\mathsf{nVALS}_t + 1} \left(\mathsf{xAVG}_t * \mathsf{nVALS}_t + \tau^{(n)}(x)_t - {x_{\text{offset}}}^{(n)}\right)\,,\ t \in \mathcal{F}$

$\mathsf{yAVG}_t = \frac{1}{\mathsf{nVALS}_t + 1} \left(\mathsf{yAVG}_t * \mathsf{nVALS}_t + \tau^{(n)}(y)_t - {y_{\text{offset}}}^{(n)}\right)\,,\ t \in \mathcal{F}$

$\mathsf{nVALS}_t = \mathsf{nVALS}_t + 1$

end

for $t \in$ *frame ids of* nVALS **do**

$\mathcal{N} \leftarrow$ track ids which have frame with id t

$x_t = \frac{1}{\texttt{Length}(\mathcal{N})} \sum_{n \in \mathcal{N}} \tau^{(n)}(x)_t - {x_{\text{offset}}}^{(n)}$

$y_t = \frac{1}{\texttt{Length}(\mathcal{N})} \sum_{n \in \mathcal{N}} \tau^{(n)}(y)_t - {y_{\text{offset}}}^{(n)}$

end

Pseudocode 1: Calculating aligned track $\{(x_t, y_t)\}$.

The parameters of the model are evaluated by the least square fit

$$\arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} (x_t - X(t))^2 \tag{1.22a}$$

$$\arg\min_{\boldsymbol{\theta}} \sum_{t=1}^{T} (y_t - Y(T))^2 \tag{1.22b}$$

where the parameter vector $\boldsymbol{\theta} = (a, b, c)$ contains the polynomial coefficients.

The last step is to correct the position of the fluorophores. This is done by the formula

$$\hat{x_t} = x_t - X(t) \tag{1.23a}$$

$$\hat{y_t} = y_t - Y(t) \tag{1.23b}$$

Before analysis of a data set it is important to make sure that the global drift can be described by the assumed model. It is possible vibrations from the acquisition equipment to interfere with the sample. In such case the data set should be discarded because the global drift model $X(t)$ and $Y(t)$ cannot accurately describe the sample movement and FLImP analysis will produce invalid results.

Before each experiment the global drift of the sample is inspected manually to make sure that it can be described by the assumed model. There are downsides to manual inspection such as it is time consuming and prone to error. Ideally this step should be automated, however this is outside the scope of this project.

1.6 Track Selection

Track selection is described below as it stood prior to the research project described in this thesis. The limitations are discussed and provide the basis for the work presented in later chapters. It has four stages, Figure 1.5.

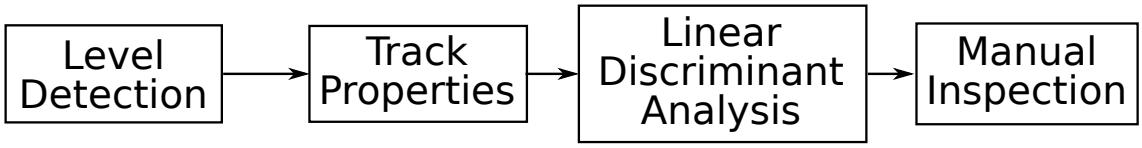


Figure 1.5: The track selection process has 4 steps, one of which is manual.

- **Level Detection**. Levels are frame ranges in which a constant number of fluorophores are active and the track has a constant intensity (apart for the noise).
- **Track Properties**. Based on the detected levels some properties of the tracks are calculated, such as mean intensity in levels, difference of position between levels, and so on. Full list of properties can be found in the Appendix A
- **Linear Discriminant Analysis**. This stage uses linear filter to discard tracks which are not likely to be suitable for FLImP analysis.
- **Manual Inspection**. Tracks are manually inspected and some of them are selected for FLImP analysis, based on pre-determined assumptions.

1.6.1 Level Detection

The FLImP analysis is applied on sections of the track where the fluorophore configuration does not change. In these sections there are either two active fluorophores or one active fluorophore. Therefore, finding such sections is important to the analysis process. A frame range in which there is a constant number of active fluorophores is called level.

Based on the their definition the track levels are expected to have several properties. They are supposed to have constant intensity. Any increase or decrease of intensity can invalidate the FLImP model and cause erroneous results. Another implication is that the statistical properties of any subsection of a level should be the same as the entire level.

The level detection algorithm first finds statistically significant intensity changes, called steps in the track. An intensity transition between frames t and $t+1$ is considered to be a step if

$$\mathrm{erf}\left(\frac{|I_t - I_{t+1}|}{\sqrt{2\left(\sigma_{I_t}^2 + \sigma_{I_{t+1}}^2\right)}}\right) > 0.95 \tag{1.24}$$

where I_t is the intensity of the feature in frame t, σ_{I_t} is the uncertainty in the measurement of the intensity of the feature in frame t. Quincy assumes that the measurement of the intensity of the diffraction limited spot follows Gaussian distribution. Therefore, if the intensity measured from two frames is subtracted the result will also follow Gaussian distribution with mean $I_t - I_{t+1}$ and standard deviation $\sqrt{\left(\sigma_{I_t}^2 + \sigma_{I_{t+1}}^2\right)}$. This means that if the result of Equation 1.24 is more than 0.95 there is 95% probability that the difference between I_t and I_{t+1} is not zero. This can be deduced from the fact that $\mathrm{erf}(x) = 2\Phi_{0,\sqrt{\frac{1}{2}}}(x) - 1$, where $\Phi_{\mu,\sigma}(x)$ is the cumulitive density function of a random variable which follows Gaussian distribution with mean μ and standard deviation σ [77].

These quantities are calculated by Quincy, using Bayesian parameter estimation. The erf is

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{1.25}$$

After all the steps are identified the frames between these steps are assigned to levels. The levels are merged, based on the mean and the standard deviation of the level intensity. Each two levels, i and j, are merged together if

$$\mathrm{erf}\left(\frac{|\langle I_i \rangle - \langle I_j \rangle|}{\sqrt{2\left(\sigma_{l_i}^2 + \sigma_{I_j}^2\right)}}\right) < 0.95 \tag{1.26}$$

The quantity $\langle I_i \rangle$ is the mean intensity in level i,

$$\langle I_i \rangle = \frac{1}{|l_i|} \sum_{t \in l_i} I_t \tag{1.27}$$

where l_i is the set of indexes of all frames assigned to level i and $|l_i|$ is the number of frames assigned to level i. The quantity σ_{l_i} is the standard deviation of the intensity in level i,

$$\sigma_{l_i}^2 = \frac{1}{|l_i|} \sum_{t \in l_i} (I_t - \langle I_i \rangle)^2 \tag{1.28}$$

Details about the algorithm can be found in the pseudo-code, which is split into two parts Pseudocode 2 and Pseudocode 3.

Assumptions and Limitations

The level detection algorithm merges levels based on their standard deviation and difference of intensity. This sometimes groups level together, when they should be in different levels, Figure 1.6.

This error can cause tracks which violate the FLImP assumptions to go through analysis and generate invalid results, an example of False Positive. Another way in which the same error can affect FLImP is to cause data to be thrown away and decrease the efficiency of the method, an example of False Negative, Figure 1.7.

Another limitation is that the level detection algorithm does not check for gradient in the levels. Increase of the intensity of a feature can be caused by overcrowding of fluorophores. Then the background intensity is overestimated and therefore the feature intensity to be underestimated. When the background fluorophores photobleach the background intensity would decrease, which would cause the feature intensity to be increased. Such tracks would have to be rejected because of the background fluorophores in the region of interest around the feature. This mistake is unlikely to cause FLImP to produce erroneous estimations, however it may create unnecessary manual work.

input : List of most likely intensities I of length T;

List of intensity confidence intervals σ of length T

output: List of levels, L. Each level is list of intervals $\{(\min, \max)\}$

for $t \leftarrow 1$ *to* $T - 1$ **do**

  if $\operatorname{erf}\left(\frac{|I_t - I_{t+1}|}{\sqrt{2\left(\sigma_{I_t}^2 + \sigma_{I_{t+1}}^2\right)}}\right) > 0.95$ **then**

    steps $\leftarrow t$;

  end

end

if steps *is empty* **then**

  levels $\leftarrow \{(1,\mathsf{T})\}$;

  else

    levels $\leftarrow \{(1, \text{steps}_1)\}$;

    for $t \leftarrow 1$ *to* $\texttt{Length}(\text{steps}) - 1$ **do**

      levels $\leftarrow \{(\text{steps}_t + 1, \text{steps}_{t+1})\}$

    end

    levels $\leftarrow \{(\text{steps}_1 + 1, T)\}$

  end

end

Pseudocode 2: Level detection algorithm. (Part 1)

```
while true do
    minProbDifferent ← 1.;
    for i ← 1 to Length(levels) do
        for j ← 1 to Length(levels) do
            if i == j then Continue;
            probDifferent ← erf( (⟨I_i⟩ − ⟨I_j⟩) / sqrt(2(σ²_{I_i} + σ²_{I_j})) );
            if probDifferent < minProbDifferent then
                minProbDifferent ← probDifferent;
                mergeI ← i;
                mergeJ ← j
            end
        end
    end
    if minProbDifferent < 0.95 then
        Append (levels, Concatenate (levels_mergeI, levels_mergeJ));
        Delete (levels, {mergeI, mergeJ});
        else
            Break;
        end
    end
end
```

Pseudocode 3: Level detection algorithm. (Part 2)

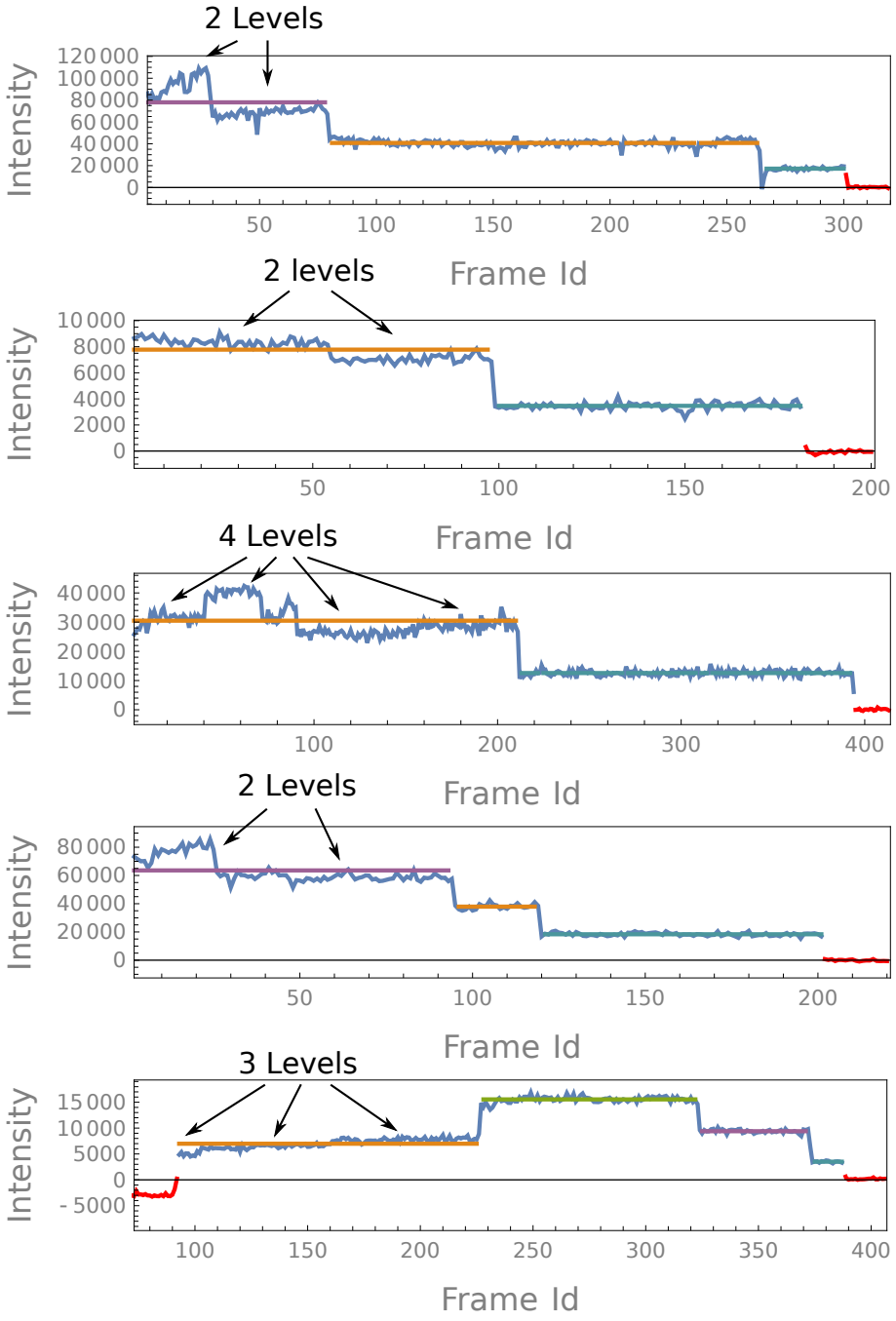


Figure 1.6: The level detection algorithm sometimes makes mistakes. The above figures are examples of cases when level segments are grouped into the same level, however they belong to different levels

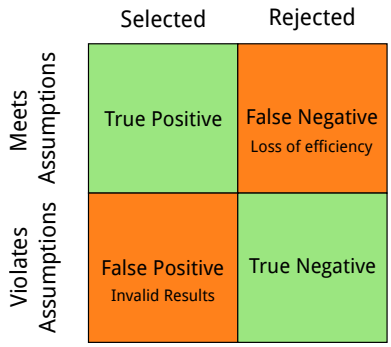


Figure 1.7: Wrongfully grouping level segments into levels can cause tracks which violate the FLImP assumptions to be analysed and tracks which meet the assumptions to be rejected.

1.6.2 Linear Discriminant Analysis

Some tracks are rejected before manual inspection based on their length and number of levels. If a track is shorter than 20 frames or has less than two levels it is automatically rejected. If a track is less than 20 frames it is considered too short to produce good confidence interval. If it has less than two levels then it cannot be analysed by FLImP.

After rejection of tracks based on their length and number of levels, a track filter is applied to reject some of the tracks, which will not be selected. This filter calculates some properties of the tracks, and uses a plane to separate the positive and negative tracks. The filter can be described with the formula

$$\boldsymbol{w}^T\boldsymbol{x} > \tau \tag{1.29}$$

where $\boldsymbol{w}$ and τ are the parameters of the plane and $\boldsymbol{x}$ is a property vector of a track. The properties are chosen manually and the parameters of the plane are chosen using LDA and previous track selections as labels. This filter is referred to as "LDA filter". A full list of the track properties and the parameters of the filter, $\boldsymbol{w}$ is shown in Table 1.1, and the meaning of the track properties can be found in Appendix A. The parameter τ

is set manually in the range $[0.010, 0.014]$.

For a track to pass the automatic rejection step, it needs to meet all of the conditions described below:

1. **Track is longer than 20 frames.** Track which has less than 20 frames is considered too short to have good confidence interval.

2. **Track has at least two levels.** Track which has only one level cannot be analysed by FLImP.

3. **The value of the LDA filter is larger than 0.01.** This condition is not related to FLImP assumptions. However, it is used to reduce the number of tracks that are manually inspected. While it reduces the number of tracks from several thousand to several hundred, it still produces way too many false positive tracks (Track which are identified as analysable by the filter but are not. More information in Section 2.1.1), as only a few percent of the tracks that pass the filter, are selected for analysis.

This filter assumes that tracks which have less than 20 frames will produce measurements with very large confidence intervals. The confidence interval of the result depends on the number of observations, which is the length of the levels in this case. It also depends on the signal to noise ratio. The signal is the number of photons detected for the duration of a frame from some fluorophore, and therefore it follows Poisson distribution. This means that the signal to noise ratio is proportionate to the brightness of the fluorophore. Therefore, a very bright track which is less than 20 frames long may still produce good confidence interval. Such mistakes can decrease the efficiency of FLImP.

Another assumption is that the tracks which are selected for analysis and these which are not selected, when represented in the property space, can be separated by a plane. This is not necessary true. Such mistake may produce a lot of false positives, which will

$\boldsymbol{x}_i$	approximated $\boldsymbol{w}_i$
sigovermlsig	1.2111e-03
level1points	1.1774e-05
pos_density_12	8.7591e-07
level2points	7.7686e-07
signal_lvl1	-2.0787e-10
signal_lvl2	-2.3563e-12
semdr12	3.8136e-03
leveliness	-1.6384e-04
cts_loc_sn_l1	1.3171e-05
cts_loc_sn_l2	-1.427e-06
maxstepfrac	9.9832e-04
dr12	1.3069e-03
ratio_real_points	4.4687e-03
displacement	-2.766e-04
cts_slm_l1	1.1455e-08
cts_slm_l2	1.8737e-07
meanovermlsig	-2.1767e-04
levelrchisq	-1.5365e-04
background	4.6931e-04
shortest_range	1.6090e-05

Table 1.1: This table shows the track properties and the corresponding parameter used in the LDA filter. The meaning of the track properties is explained in Appendix A.

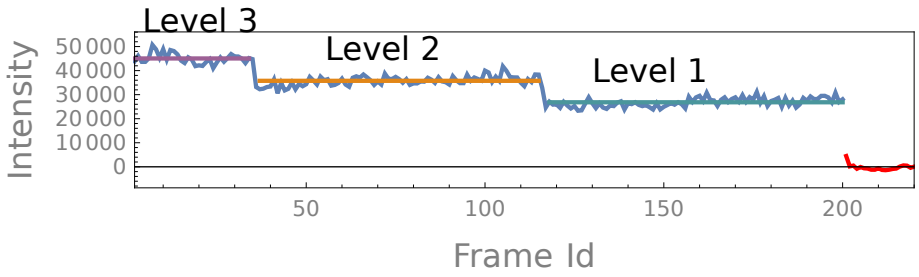


Figure 1.8: The track levels are numbered from least bright, called level 1, to the brightest, called level n if the track has n levels.

increase the manual work which needs to be done, and many false negatives, which will decrease the efficiency of FLImP.

1.6.3 Manual Inspection

FLImP analyses section of the track where there are two or one active fluorophores. These sections are the track level with the lowest mean intensity, also called level 1, and the track level with the second lowest mean intensity, called level 2. The rest of the levels are also numbered according to their intensity, Figure 1.8. FLImP model assumes that level 1 is generated by only one active fluorophore and level 2 is generated by the same fluorophore which generated level 1 and one more active fluorophore. Level 1 and level 2 are separated by a sharp intensity change, called photobleaching step.

The tracks that pass the automatic rejection step, need to be manually inspected before being selected for analysis. During the manual inspection it is possible to cut parts of the track so that only the last two or the first two levels are used. Then the order of levels could change. For example if there is track with level 1, followed by level 3, followed by level 2, as shown on Figure 1.9, the first part of the track can be removed so that only the last two levels are analysed. Then level 3 will become level 2 and level 2 will become level 1.

This way of analysing tracks is not consistent with the LDA filter, because the LDA

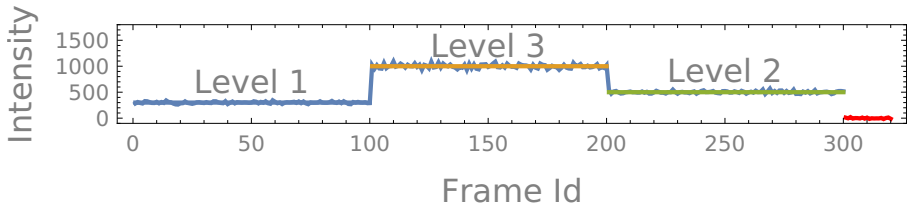


Figure 1.9: Example of a track in which the level order will change after selected for analysis.

filter will be applied to the entire track, which means that the level 1 used in the LDA filter may be different level from the level 1 selected by a person. The issues coming out of this inconsistency is that some tracks which could have been selected for analysis may be rejected by the LDA filter and also some tracks which are not rejected by the LDA filter may not be appropriate for analysis.

In the case of the track shown on Figure 1.9 level 3 will be analysed as level 2 and level 2 will be analysed as level 1. It is important that the last two levels are analysed instead of the first and the last level. In FLImP it is assumed that after track is cut in level 2 there are 2 fluorophores and in level 1 there is 1 fluorophore even if before cutting there is another level with lower intensity than level 1.

In the condition describing what type of tracks are selected for analysis it is assumed that the level 1 and level 2 are the levels with lowest and second lowest intensity after the track is cut, so that it only has the last 2 or the first 2 levels.

For a track to be selected for analysis, it needs to meet all of the conditions described below:

1. **Track ends with level 2 followed by level 1 or starts with level 1 followed by level 2.** These two types of tracks can give high confidence that in level 2 two fluorophores are active and in level 1 only one of these two fluorophores is active.

2. **If level 1 comes first, then the track needs to start after the beginning of the experiment.** If the track start before the first frame, it is not clear whether

there are other levels or intermediate intensities before level 1. Therefore, it is unclear how many fluorophores are in level 1.

3. **If level 1 comes last, then the track needs to end before the end of the experiment.** If the track ends after the last frame, it is not clear whether there are other levels or intermediate intensities after level 1. Therefore, it is unclear how many fluorophores are in level 1.

4. **Level 1 or level 2 should be within the frame ranges, in which the global drift can be processed.** FLImP can only account for global drift that can be described by a polynomial of second degree. Therefore, levels that are outside the frame ranges, where the global drift can be accounted for, cannot be processed.

5. **Level 1 or level 2 should have more than 10 non-interpolated frames.** This condition is related to the probability of the track to produce a measurement with low confidence interval. Too many interpolated frames can be an indication that the feature is too dim or there is interference with the feature.

6. **If level 1 comes first, then when the track is extrapolated, the intensity before the beginning of the track should be zero; if level 1 comes last, then when the track is extrapolated, the intensity after the end of the track should be zero.** If there is intensity different from zero at the position of the feature after or before the track, this could be an indication that either there is an additional feature that is not detected by Quincy, or there is background fluorescence different from the uniformly distributed noise.

7. **The intensity within a level should be constant. If there are increases or decreases over time within a level, the track should be rejected.** According to the FLImP model, the fluorophores are in the emitting state all the time during an intensity level. This means that if there are no additional interference, the

intensity should be constant.

Usually slow increase of the intensity is caused by a feature, around which there are many fluorophores active in the background. These fluorophores would cause Quincy to overestimate the background intensity and therefore underestimate the feature intensity. When the background fluorophores photobleach, the background intensity estimate decreases and the feature intensity becomes closer to the real intensity.

8. **Levels with too many interruptions should not be analysed.** According to the current understanding of the fluorophores used in the FLImP experiments, they can go from active to dark state and then back to active state. However, the probability of this happening is very low, therefore if there are many level interruptions, it is more likely that it is caused by some other process.

9. **Level 2 should have twice the intensity of level 1.** The fluorophores used in a FLImP experiment are of the same type, therefore their emission intensity would be the same, provided that the intensity of the light, to which they are exposed to, is the same. However, because the fluorophores are excited by the evanescent field, fluorophores at different depths would have different intensities.

   If the intensity of level 2 is different from twice the intensity of level 1, then either level 2 has more than two fluorophores, or the fluorophores that generated the track are at different depths. In either case the track cannot be analysed, because the measured distance will be either between a fluorophore and the mean position of two other fluorophores, or the projection of the real distance on the sample surface.

10. **There may be some small gaps between levels (several frames). However, if the levels are too far away from each other in time, the track should be rejected.** The FLImP model cannot explain such gaps, but sometimes the

level detection algorithm erroneously produces shorter levels. These gaps usually are less than 10 frames. However, if there are gaps larger than 10 frames, then it is likely that this track region is not part of a level.

11. **There cannot be more than 1 frame between level 1 and level 2 which have intensities that are not consistent with the intensities of these levels.** If there are more than 1 frame with intermediate intensities between level 1 and level 2, this could be an indication that there are more than two active fluorophores in level 2.

12. **If level 1 comes first, there cannot be any intensities before level 1 that are not consistent with the level; if level 1 comes last, there cannot be any intensities after level 1 that are not consistent with the level.** If there are intensities before or after level 1, this is an indication that there might be more than one active fluorophore in level 1.

13. **There cannot be any fluorescence in the neighbourhood of the diffraction limited spot, apart from uniformly distributed noise.** FLImP cannot model fluorescence that is different from the background noise, which is distributed uniformly across space, or the detected feature, which is modelled with one Gaussian distribution in level 1 and two Gaussian distributions in level 2.

14. **The positions of the track during levels 1 and 2 should be constant.** FLImP model assumes that the fluorophores do not change position apart from the change caused by the noise in the system, which should be treated as a random error. Therefore it takes each frame from a level as a independent and identically distributed measure of the fluorophore positions.

    The fluorophores are attached to the EGFR and the cells are fixed. Beyond the global drift of the sample, there shouldn't be any directed movement. FLImP

software can account for the global drift of the sample before processing the tracks.

Another source of motion could come from the anitbodies or the molecular domain, which will randomly walk around the their mean positions. Since FLImP takes average measurements over large period of time the this motions will be averaged out and the mean position will be taken into account.

15. **The shape of the feature should be a circle.** If the microscope is not focused at the surface of the cell, the point spread function of the diffraction limited spot would not be the Airy function and therefore it cannot be modelled by a Gaussian distribution.

16. **The selected tracks should be likely to result in measurements with confidence interval less than 10nm and separation less than 60nm**. The tracks which are selected should have

    - long levels
    - small variance in the position of level 1 and level 2
    - well separated intensities of level 1 and level 2
    - small shift in position during the photobleaching step.

    These conditions are not related to the validity of the result, but they aim to minimise the computational cost of the FLImP analysis, by trying to avoid analysing tracks that will not be included in the final results.

More details of the manual selection process can be found in Zanetti-Domingues et al [1].

1.7 FLImP Analysis

The FLImP technique is inspired by Single-molecule High-Resolution Imaging with Photobleaching (SHRImP) [53] and it calculates the distance between fluorophores, by analysing level 1 and level 2 in a track, Figure 1.10.

The fluorophores are numbered as fluorophore 1 and fluorophore 2. The intensity I of the diffraction limited spot in each frame t is modelled as a dependency of position by a mixture of two two-dimensional Gaussian distributions

$$I(x, y, t, \boldsymbol{\theta}, b(t)) = b(t) + \sum_{i=1}^{2} \frac{f_i(t) I_i}{2\pi\sigma^2} e^{\left(-\frac{1}{2}\frac{(x-x_i)^2+(y-y_i)^2}{\sigma^2}\right)} \tag{1.30}$$

where x_1 , y_1 , x_2 and y_2 are the x and y positions of fluorophores 1 and 2 and the I_1 and I_2 are their intensities. The parameter $f_i(t)$ is either 0 or 1, depending whether a fluorophore i is active in frames t. The parameter σ is the size of the PSF and $b(t)$ is the value of the background. The background is assumed to be uniformly distributed across space (has the same value for each pixel), however can change during time. This model is applied in a ROI around the centre of a feature. All the model parameters can be represented as a vector $\boldsymbol{\theta} = (x_1, y_1, x_2, y_2, I_1, I_2)$.

Since the model is continuous and the observations are discrete (pixels) the intensity of each pixel, with coordinates (X, Y) and centred at $(X + \frac{1}{2}, Y + \frac{1}{2})$, is the integral of the model over the area of the pixel

$$F(X, Y, t, \boldsymbol{\theta}, b(t)) = \int_{x=X}^{X+1} \int_{y=Y}^{Y+1} I(x, y, t, \boldsymbol{\theta}) dx dy \tag{1.31}$$

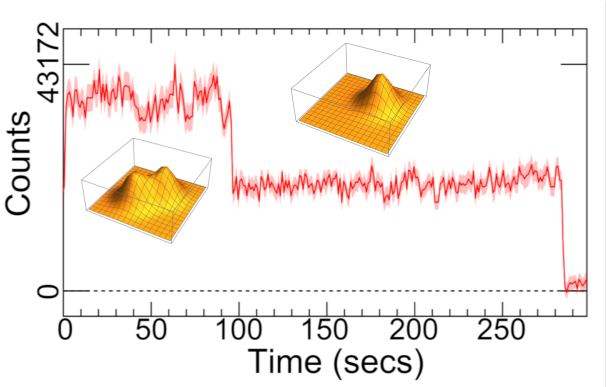


Figure 1.10: The FLImP analysis fits a sum of two 2D Gaussian functions in level 2 and one 2D Gaussian function in level 1 in order to calculate the separation between the two sources of light. Figure 1A from [2]

which when solved becomes

$$F(X,Y,t,\boldsymbol{\theta},b(t)) = b(t) + \frac{1}{4}\sum_{i=1}^{2} f_i(t) I_i \Delta X_i \Delta Y_i \tag{1.32a}$$

$$\Delta X_i = \operatorname{erf}\left(\frac{X+1-x_i}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{X-x_i}{\sigma\sqrt{2}}\right) \tag{1.32b}$$

$$\Delta Y_i = \operatorname{erf}\left(\frac{Y+1-y_i}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{Y-y_i}{\sigma\sqrt{2}}\right) \tag{1.32c}$$

The model parameters $\boldsymbol{\theta}$ are evaluated by minimising the error function

$$\chi^2(\boldsymbol{\theta}, b(t), t) = \sum_X \sum_Y \frac{\left(F(X,Y,t,\boldsymbol{\theta},b(t)) - D(X,Y,t)\right)^2}{\sigma^2(D(X,Y,t))} \tag{1.33}$$

where $D(X,Y,t)$ is the value of pixel (X,Y) at time t, and $\sigma(D(X,Y,t))$ the expected error in that measurement, which is assumed to be Gaussian. The error is set to one, $\sigma(D(X,Y,t)) = 1$, so that the problem becomes least square fit.

To simplify the calculations the background value $b(t)$ is expressed in terms of the rest of the parameters by solving the equation

$$\frac{\partial \chi^2\left(\boldsymbol{\theta}, b(t), t\right)}{\partial b(t)} = 0 \tag{1.34}$$

which gives a formula for $b(t)$

$$b(t) = \frac{1}{4k} \sum_X \sum_Y \left(D(X,Y,t) - \sum_i f_i(t) I_i \Delta X_i \Delta Y_i \right) \tag{1.35}$$

where k is the number of pixels in the ROI.

This quantity is substituted in 1.30 to produce error function with less parameters.

$$\mathcal{E}(\boldsymbol{\theta}) = \sum_X \sum_Y \sum_t \left(F(X,Y,t,\boldsymbol{\theta}) - D(X,Y,t) \right)^2 \tag{1.36}$$

The minimum of this function is found by the downhill-simplex algorithm [78]. After the model parameters $\boldsymbol{\theta}$ are calculated, the fluorophore separation can be calculated from the difference between the (x, y) positions.

1.7.1 Bootstrapping and Confidence Interval

FLImP estimates the probability distribution of the separation by using the bootstrapping method [63]. This method estimates the probability distribution of some parameter, θ, of a population $p = \{x_i\}\,,\ i = 1..n$, where $\theta = f(p)$. In the case of FLImP the population is the region of interest around a feature in all analysed frames and the parameter is the fluorophore separation.

The bootstrap method draws samples from the population p with replacement to create a new population p_1^*. Then from the new population it estimates the parameter $\theta_1^* = f(p_1^*)$. The method repeats this B times to create a set of parameters $\{\theta_i^*\},\ i = 1..B$. From this population can be estimated a confidence interval of the parameter θ.

FLImP estimates the parameters of the model on the original data set, which is called the original fit, s_0. Then it would re-sample the data set 1200 times and estimate the parameters of the model for each sample of the original data set, creating the bootstrap samples, s_1 .. s_n.

The number of bootstrap samples is chosen as a trade off between parameter accuracy and computational cost.

The probability density function $P(s)$ is estimated using kernel density estimation with Gaussian kernels. The standard deviation of the kernel is chosen by using Silverman's rule of thumb. Then the distance between each bootstrap sample and the original estimate is calculated and scaled by the probability of each sample.

$$d_i = \frac{|s_i - s_0|}{P(s_i)} \tag{1.37}$$

Using the scaled distance d_i, the confidence interval is defined as the range, which contains 68% of the closest samples to the original fit [2].

Tracks which generate separations with too large confidence intervals are removed. Such tracks are considered too unreliable. Often confidence intervals above 7nm or 10nm are removed [2]. The bootstrap samples of the remaining tracks are combined and mixture of Rice distributions is fitted. The optimal number of Rice components is chosen using Bayesian information criteria.

1.7.2 Assumption and Limitations

The FLImP model has several assumptions about the diffraction limited sequences, used for the parameter estimation.

- The sequence is generated by at least two fluorophores.
- The sequence has at least two sub-sections: level 1, where only one fluorophore is active and level 2, where both fluorophores are active.
- Time frames are assumed to be independent measurements of the fluorophore intensities.
- The fluorophores can have different positions, but don't change over time.

- The positions and intensities of the fluorophores are constant (apart from the noise in the system).
- The fluorophores are at the same distance from the sample surface.
- The background intensity has the same value for each pixel (there are no additional fluorophores in the background).
- The both fluorophores have a point spread function, PSF, with the same size, which is assumed to have a Gaussian profile.
- The background intensity can change over time.
- The noise of the measurement of the values of each pixel is Gaussian.

The model requires both levels, because it relies on the photobleaching step to estimate the difference of the position with high accuracy.

The fluorophores are assumed to be at the same depth, because FLImP measures the distance between the fluorophores projected on the sample surface. If the fluorophores are at different depth, then FLImP would estimate the separation as a 2D projection of the 3D inter-molecular distance.

The fluorophores are assumed not to move, because they are attached to the EGFR and the proteins are fixed. This allows us to treat each frame as an independent measure of the positions, which results in high accuracy estimations. The assumptions that each frame is an independent measure of the fluorophore, requires assumption that the fluorophores do not change their intensity, apart from when they photobleach. This assumptions is reasonable, since the fluorophores do not change their depth and the laser intensity is also kept constant.

The PSF of the fluorophores is best described as the Airy function. However, for the purposes of estimating the positions of the fluorophores, it is reasonable to use a

Gaussian approximation. The advantage of the Gaussian function is that it is much more computationally efficient to estimate the parameters.

The probability distribution of the separation is subject to several parameters. The first is the number of bootstrap samples. In FLImP 1200 samples are used, however this number can be changed. The number of bootstrap samples can change the accuracy of the separation and confidence interval. According to [79] 1000 bootstrap samples are enough when estimating confidence interval, therefore the choice of 1200 samples is reasonable.

The work presented in this thesis will improve the reliability, reproducibility and the efficiency of the track selection stage of FLImP. To do this a new level detection algorithm will be used, presented in Chapter 4, and large parts of the selection process will be handled automatically, presented in Chapter 6.

Chapter 2

Classification Using Machine Learning

This chapter describes the algorithms and methods used in the project. Such algorithms include methods for evaluating the performance of binary classifiers, neural networks which are used for classification, dimensionality reduction methods, and statistical tests.

Classification is the process of assigning categories to objects. In this project the problem of automatic classification is approached using machine learning techniques. All the uses of classification involves only two categories, which is known as binary classification.

2.1 Binary Classifiers

A binary classifier [80, Chapter 1.2] assigns objects either positive or negative category. Automatic classification can be done by forming some numerical description of the objects which are classified. This numerical description can be represented as a vector, called feature vector, with some dimensionality. The vector space which this vector is

member of is called feature space. Then the classification becomes function approximation problem, where the input of the function is the feature space and the output is the categories. This can be expressed as

$$\kappa(x) = c \qquad x \in \Psi, \text{ and } c \in \{+, -\} \tag{2.1}$$

where Ψ is the feature space of the classified objects and $\{+, -\}$ are the categories. For example if grey-scale images with particular resolution are classified, Ψ could be the space of all possible grey-scale images with that resolution. It is not necessary for the input data of a classifier to occupy the entire volume of the feature space. In such case it is possible to do dimensionality reduction to avoid difficulties with training.

Most binary classifiers have probabilistic interpretation. This means that they output not only the predicted category, $c \in \{+, -\}$, of a sample, $x \in \Psi$, but also the confidence of prediction, $p\,(c|x)$. Since the predicted categories are only two, the ratio of the probability of positive category given sample, and negative category given sample, can be compared with the number 1. Then the process of assigning a category to a sample is described by Formula 2.2.

$$\kappa(x) = \begin{cases} + & \text{if } \frac{p(+|x)}{p(-|x)} > 1 \\ - & \text{otherwise} \end{cases} \tag{2.2}$$

The classifier models the probability of sample given a class, $p(x|-)$ and $p(x|+)$, instead of the probability of class given sample. Therefore the Bayesian theorem is used to convert the condition in Formula 2.2 to Formula 2.3, where the threshold $\tau = \frac{p(-)}{p(+)}$ represents the ratio of the class priors.

$$\frac{p(x|+)}{p(x|-)} > \tau \tag{2.3}$$

By varying τ, the class priors can be changed and therefore the decision boundaries of the classifier will also change.

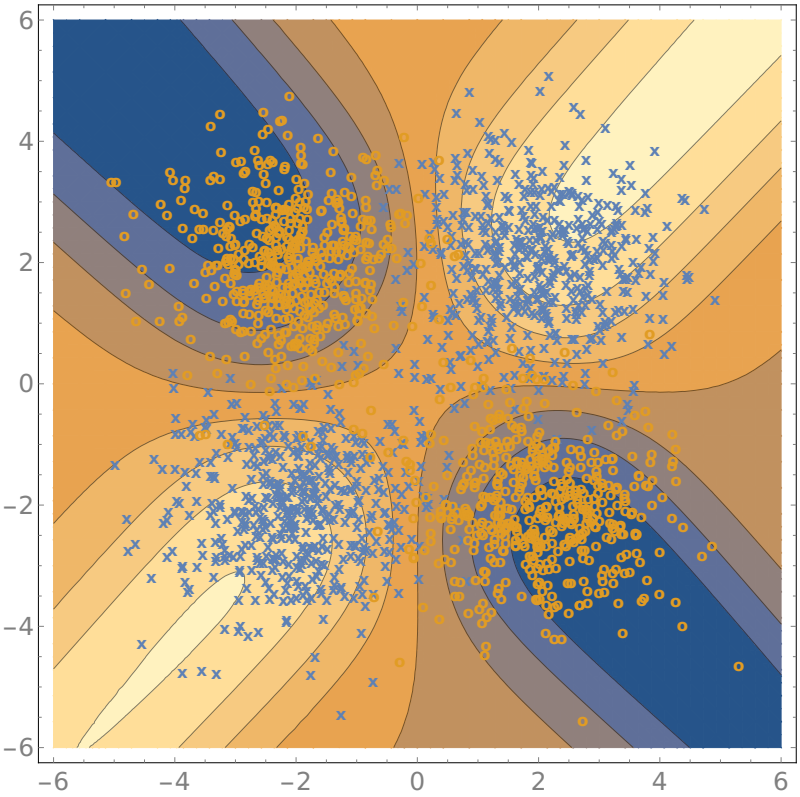


Figure 2.1: Different decision boundaries of a classifier, trained against the XOR data set, based on different thresholds applied to the ratio of probabilities of sample given a class. The different boundaries are shown with different colours. The training samples are shown with orange and blue.

A good example is a classifier trained on the XOR data set which consists of two classes shown in Figure 2.1. The positive class is shown with orange "o" and the negative class is shown with blue "x". The both classes are generated by randomly choosing between two 2D Gaussian distributions for each sample. The parameters of the positive class are $\mu_1^+ = (-2, 2)$, $\mu_2^+ = (2, -2)$, $\Sigma_1^+ = \Sigma_2^+ = I$ and the negative class are $\mu_1^- = (2, 2)$, $\mu_2^- = (-2, -2)$, $\Sigma_2^- = \Sigma_2^- = I$, where I stands for the identity matrix. In Figure 2.1 different decision regions are shown with different colours, based on different values of the threshold τ.

Different values of τ would shift the boundary in favour of one of the two classes. This may be useful when error of wrongly assigning a sample to one of the classes is much more expensive than wrongly assigning sample to the other class. For example in FLImP analysable tracks can be classified as likely to have low confidence interval, class *yes*, and likely to have large confidence interval, class *no*. The error for classifying track from the *yes* class as class *no* is more significant error than classifying track from the *no* class as class *yes*, because preparation of FLImP experiments is more expensive in terms of time than analysing tracks.

2.1.1 Evaluating Performance

Evaluating the performance of a binary classifier depends on the application domain and the desired properties of the classification system. Therefore different measures can be developed.

Figure 2.2 shows a confusion matrix [81] of a binary classifier. The columns of the matrix, which are named as Condition Positive (CP) and Condition Negative (CN), and are coloured in yellow, represent the true labels of the data. The rows, which are named as Prediction Positive (PP) and Prediction Negative (PN), and are coloured in blue, represent the classifier predictions. The cells True Positive (TP) and True Negative (TN), coloured in green, are the correctly classified samples and the cells False Positive (FP) and False Negative (FN), coloured in pink, are the misclassified samples.

The confusion matrix can be used to calculate some basic performance measures. There are four basic measures which are based on the condition positive and condition negative.

- **Recall and Miss Rate** The recall, also called sensitivity, can be expressed as

$$\mathrm{Recall} = \frac{\mathrm{TruePositive}}{\mathrm{ConditionPositive}} \tag{2.4}$$

		Data Labels			
	Total Population	Condition Positive (CP)	Condition Negative (CN)		
Classifier Output	Predicted Positive (PP)	True Positive (TP)	False Positive (FP)	Precision = TP/PP	False Discovery Rate (FDR) = FP/PP
	Predicted Negative (PN)	False Negative (FN)	True Negative (TN)	False Omission Rate (FOR) = FN/PN	Negative Predictive Value (NPV) = TN/PN
		Recall (Sensitivity) = TP/CP	Fall-out = FP/CN	Accuracy (TP+TN)/Total	
		Miss Rate = FN/CP	Specificity = TN/CN		

Figure 2.2: Confusion matrix for binary classifier. The rows represent classifier outputs and the columns represent the correct labels.

This measure shows what fraction of the positive class can be recognised as positive. The complementary measure is called Miss Rate, which shows what fraction of the positive class is not recognised. It can be expressed as

$$\mathrm{MissRate} = \frac{\mathrm{FalseNegative}}{\mathrm{ConditionPositive}} \tag{2.5}$$

The miss rate shows the fraction of the positive class which is recognised as negative. The recall and the miss rate are connected with the dependency

$$\mathrm{Recall} = 1 - \mathrm{MissRate} \tag{2.6}$$

In ideal classifier the miss rate will be 0 and the recall will be 1.

- **Fallout and Specificity** Fallout shows what fraction of the negative class is classified as positive. It can be expressed as

$$\mathrm{Fallout} = \frac{\mathrm{FalsePositive}}{\mathrm{ConditionNegative}} \tag{2.7}$$

The complementary measure is called specificity and it shows what fraction of the negative class is classified as negative. Is can be expressed as

$$\mathrm{Specificity} = \frac{\mathrm{TrueNegative}}{\mathrm{ConditionNegative}} \tag{2.8}$$

In ideal classifier the fallout will be 0 and the specificity will be 1.

The measures recall and specificity have the same function, which is to show what fraction of the corresponding class has been classified correctly. Also the miss rate and the fallout have the same function, which is to show what fraction of the corresponding class has been misclassified.

There are four more measures which are based on the prediction positive and prediction negative classes.

- **Precision and False Discovery Rate** The precision shows what fraction of the predicted positive is from the positive class. This can be expressed as

$$\mathrm{Precision} = \frac{\mathrm{TruePositive}}{\mathrm{PredictedPositive}} \tag{2.9}$$

The complementary measure is false discovery rate and it shows what fraction of the predicted positive is part of the negative class. This can be expressed as

$$\mathrm{FalseDiscoveryRate} = \frac{\mathrm{FalsePositive}}{\mathrm{PredictionPositive}} \tag{2.10}$$

In ideal classifier the precision will be 1 and the false discovery rate will be 0.

- **False Omission Rate and Negative Predictive Value**. Negative predictive value shows what fraction of the predicted negative are part of the negative class. This can be expressed as

$$\mathrm{NegativePredictiveValue} = \frac{\mathrm{TrueNegative}}{\mathrm{PredictedNegative}} \tag{2.11}$$

The complementary measure is the false omission rate. This measure shows what fraction of the predicted negative are part of the positive class. This can be expressed as

$$\mathrm{FalseOmissionRate} = \frac{\mathrm{FalseNegative}}{\mathrm{PredictedNegative}} \tag{2.12}$$

In ideal classifier the negative predictive value is 1 and the false omission rate is 0.

The both measures precision and negative predictive value show the fraction of the corresponding prediction which is correctly classified. The measures false discovery rate and false omission rate show the fraction of the corresponding prediction which is misclassified.

One way to measure performance is to use accuracy. This is the ratio of correctly classified samples to all samples, Formula 2.13. The downsides of accuracy is that it depends on the ratio of the positive and negative samples [82]. For example accuracy of 0.5 on balanced data set is the worst possible accuracy, because this is equivalent to assigning everything to the same category. On the same data set an accuracy of 0.9 would be considered an improvement, because minimum 80% of either class needs to be classified correctly. However an accuracy of 0.9 on data set with 10% positive samples can by achieved by simply assigning all samples to the majority class. This means that the accuracy underestimates the miss rate, $\frac{\mathsf{FN}}{\mathsf{CP}}$.

$$A = \frac{\mathsf{TP} + \mathsf{TN}}{\mathsf{CP} + \mathsf{CN}} \tag{2.13}$$

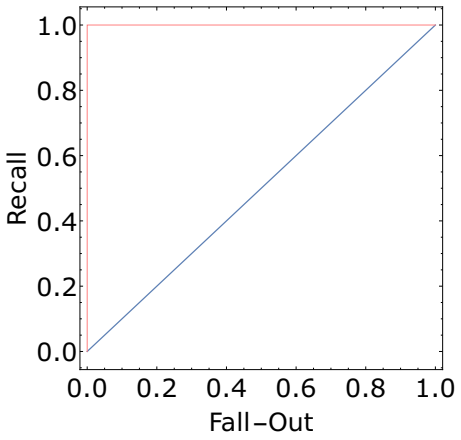
The issue with underestimating the miss rate can be addressed by using different way to measure the classifier performance. Such measure is the f1-score [83] [84]

$$F1 = 2\frac{\mathsf{precision} * \mathsf{recall}}{\mathsf{precision} + \mathsf{recall}} \tag{2.14}$$

This measure is designed to evaluate how well the minority class is represented by binary classifier in unbalanced data set. Another way to address the issues with the accuracy is by using a measure, called Area Under the Curve (AUC) [85], which represents area under the Receiver Operating Characteristic (ROC) curve. The ROC curve can be generated for every binary classifier, which has a probabilistic interpretation.

The ROC curve is defined as the pairs of $(\mathsf{fallout}, \mathsf{recall})$, measured for different values of the threshold, applied on the ratio of probability of sample to come from positive or negative class, τ (Formula 2.3). Formula 2.15 shows a formal definition of the ROC

Figure 2.3: The orange ROC curve is generated by a perfect classifier and the blue curve is generated by the worst possible binary classifier.



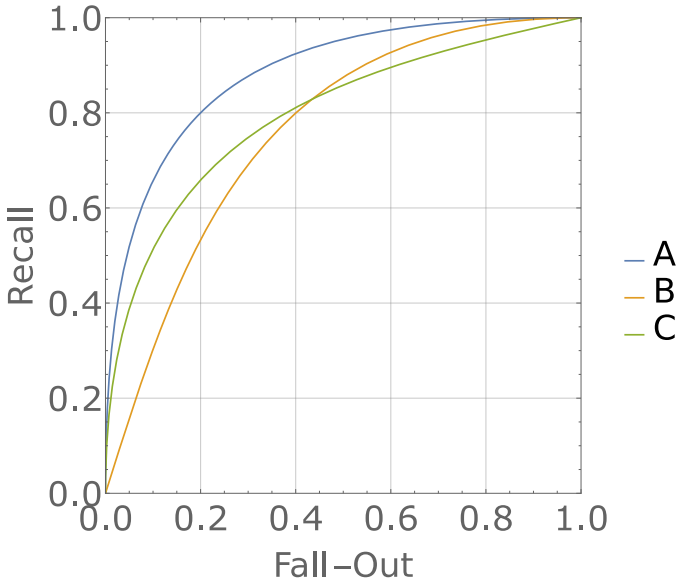
curve.

$$\text{ROC} = \{(\text{fallout}(\tau), \text{recall}(\tau)) | \tau \in \mathbb{R}\} \tag{2.15}$$

A ROC curve of a classifier with the worst possible performance would be a diagonal line between $(0, 0)$ and $(1, 1)$, which is the blue curve on Figure 2.3. A classifier with the best possible performance would have a ROC curve, which consists of two straight lines between $(0, 0)$ and $(0, 1)$, and between $(0, 1)$ and $(1, 1)$, which is the orange curve on Figure 2.3.

The ROC curve is very informative measure of classification performance, however when comparing two classifiers, some problems might occur. For example if all points of one ROC curve is to the left and above all points of another curve, as in the case of curve A and B in Figure 2.4 then clearly the classifier which generated the first curve has better performance than the classifier which generated the second curve. In this case it is said that curve A dominates curve B. However if the curves cross as in the case of curve B and C in Figure 2.4, it is not clear which classifier has better performance.

Figure 2.4: All points of curve A are to the left and above all points of curve B, therefore curve A dominates curve B. However curve B and C cross and cannot be concluded which curve represents better classification.



2.1.2 Choosing Threshold for Binary Classifier

After a classifier is constructed and a ROC curve is generated, a threshold, τ, needs to be chosen in order for the classifier to be deployed and used. There are different methods which can used to choose the threshold, such as maximising the accuracy, maximising the f1-score, or choosing the point where the ROC curve intersects the diagonal line between the points $(0, 1)$, and $(1, 0)$ [86].

To find out how these methods relate to one another, the accuracy, A, and the f1-score, $f1$, are expressed in terms of fall-out x, recall y, and the ratio between positive and all samples $r_+ = \frac{CP}{CP+CN}$.

$$A(x, y, r_+) = yr_+ + (1 - x)(1 - r_+) \tag{2.16}$$

$$f1(x, y, r_+) = \frac{2yr}{r_+ + yr_+ + x(1 - r_+)} \tag{2.17}$$

Figure 2.5 shows contour plots for accuracy and f1 score for balanced data set and for data set where the positive class is 10 % of the data ($r_+ = 0.1$). When unbalanced

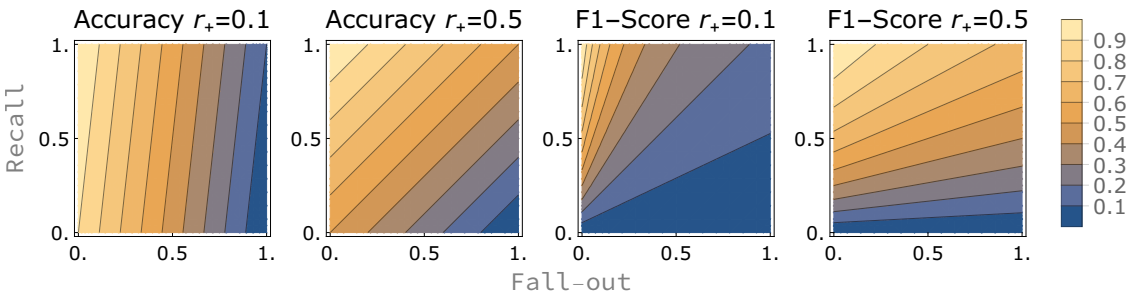


Figure 2.5: Contour plot of accuracy and f1-score as a function of fall-out and recall for different ratio of positive to all samples.

data set is used, the f1-score has very low value when the recall is low, and the accuracy can have high value when the recall is very low. However when the data set is balanced, the accuracy accounts better for the fall-out.

If the ROC curve is smooth, then the point of highest accuracy/f1-score would correspond the point, where the tangent to the curve is orthogonal to the gradient of the accuracy/f1-score. However the curve is not smooth, and the tangent cannot be calculated. In this case the highest accuracy/f1-score corresponds to the point of the ROC curve, where the line orthogonal to the gradient of the accuracy/f1-score is closest to the top left corner (0, 1).

In order to choose threshold for the binary classifier, the impact of false positives and false negatives has to be studied. This is done by assigning a cost measure to each type of error. If the cost of each type of error is combined with the probability of that error to occur, the expected cost of the classification can be estimated. Then this cost can be minimised with respect to the classifier threshold. This method is borrowed from the utility theory [87, Chapter 16].

If the false positives have the same cost as the false negatives, it is desired to achieve the same false positive rate (fallout) as the miss rate ($1-$recall). This is the intersection of the line between the points $(0, 1)$ and $(1, 0)$, and the ROC curve [86]. In the case of balanced data set, this point would correspond to the highest accuracy.

In the case when the fallout and the miss rate have different cost, using accuracy and f1-score is inappropriate. In such case the error rates can be weighted by their cost. If $\ell(x)$ is the label of sample x, then the fallout is the probability of negative sample to be misclassified $p_{\text{err}-} = p\left(\kappa(x) = +|\ell(x) = -\right)$ and the miss rate is the probability of positive sample to be misclassified $p_{\text{err}+} = p\left(\kappa(x) = -|\ell(x) = +\right)$. Using these probabilities the expected cost of errors can be calculated using Formula 2.18.

$$U = p_{\text{err}+} cost(\text{err}+) + p_{\text{err}-} cost(\text{err}-) \tag{2.18}$$

Then the expected cost U should be minimised to pick the most optimal threshold.

2.2 Neural Network

Artificial Neural Networks (ANN) are computing systems, inspired by biological neural networks. ANN are typically used for approximation of non-linear functions, $\mathbb{R}^n \mapsto \mathbb{R}^m$. Classification problems can easily be redefined as function approximation, therefore ANN can be used for classification as well.

An artificial neural network can be represented as a directed graph, where the edges are called connections and the nodes are called units. Units are usually organised in layers, and the units within a layer only have connections to the previous layer, the next layer or back to themselves (loops). Each connection is associated with a weight, $w \in \mathbb{R}$. Each unit has an activation function and a bias. It sums up all the input signals, multiplied by the weight of the corresponding connection and adds the bias. This is called activation potential. Then the activation potential is passed as the input of the activation function. The output of the activation function is the output of the unit.

If there are no loops in the network, then it is called Feed-Forward Neural Network (FFNN). This type of network typically handles independent and identically distributed data. If there are loops in the network, then it needs to "remember" outputs of some

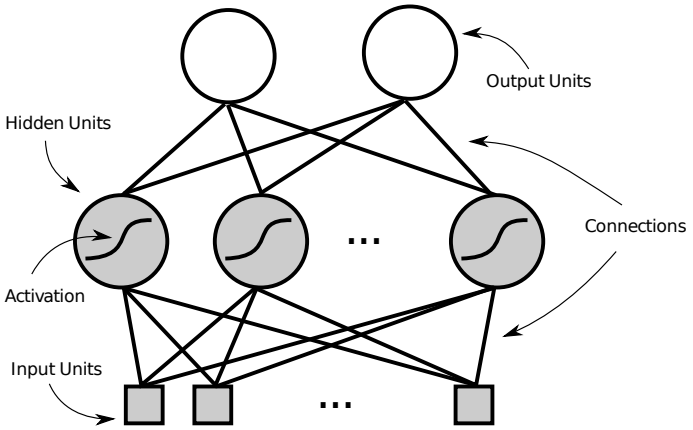


Figure 2.6: Feed-forward neural network has a input layer, one or more hidden layers with non-linear activation function and a output layer.

units until the next input is processed [88], [89], [90], [91]. Such networks are called recurrent neural networks and can process data with correlation between the data points, such as biological sequences [92] and speech [93], [94].

FFNN have an input layer, one or more hidden layers and an output layer, Figure 2.6. Usually the layers are fully connected and the hidden layers have a non-linear activation function. When some data is processed, the values of the input units are set to the corresponding values of each dimension of the input data, and then these outputs are passed forward to the hidden units and the output units.

The layers of the network can be numbered from 0 to L, where layer L is the output layer, layer 0 is the input layer, and layers 1 to $L-1$ are the hidden layers. The activation function for layer l is $\psi^l : \mathbb{R}^n \mapsto \mathbb{R}^n$ and is applied element-wise. The outputs of the input layer are the same as the input to the network $\boldsymbol{o^{(0)}} = \boldsymbol{x}$. The weights of each layer $l = 1..L$ are a matrix $\boldsymbol{W^{(l)}} \in \mathbb{R}^{n \times m}$, where the weight $\boldsymbol{W^{(l)}}_{ij}$ is associated with the connection between unit j from layer $l-1$ and unit i from layer l. The biases are represented by a vector $\boldsymbol{b^{(l)}} \in \mathbb{R}^m$, where $\boldsymbol{b}_i^{(l)}$ is the bias of unit i from layer l. The activation potentials are represented as vector $\boldsymbol{v}^{(l)}$, where $\boldsymbol{v}_i^{(l)}$ is the activation potential of unit i from layer l. Using these notations FFNN can be evaluated using the recursive

formula

$$\boldsymbol{o}^{(l)} = \begin{cases} \psi^l(\boldsymbol{v}^{(l)}) & \text{if } l \neq 0 \\ \boldsymbol{x} & \text{if } l = 0 \end{cases} \tag{2.19a}$$

$$\boldsymbol{v}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{o}^{(l-1)} + \boldsymbol{b}^{(l)} \tag{2.19b}$$

It is proven that ANN can approximate any function within a finite region with arbitrary accuracy, provided that enough hidden units are used [95]. The approximation works with any non-linear activation function, ψ^l, however experience shows that functions which are symmetric around zero work better than non-symmetric functions [96], [97]. Therefore in this project hyperbolic tangent will be used.

2.2.1 Training Algorithms

Supervised learning is a technique which tries to derive parameters $\boldsymbol{\theta}$ of some model f based on a set of pairs of inputs and outputs $\{\boldsymbol{x}, \boldsymbol{y}\}$. For example if the model f is a neural network used as a classifier, the classified objects can be represented as real vectors $\boldsymbol{x} \in \mathbb{R}^n$, and the outputs y will be the category.

The quality of the model parameters $\boldsymbol{\theta}$ is measured by an error function $\mathcal{E}(\theta)$, which has low values for good parameters and high values for bad parameters. An example of error function is the mean squared error

$$\mathcal{E}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N}\left(f(x_n, \boldsymbol{\theta}) - y_n\right)^2 \tag{2.20}$$

After the error function is defined, finding good parameters for the model f becomes function minimisation problem. For the function minimisation can be used a class of algorithms which use the gradient information. These algorithms find the minimum of a function by starting at a random location in the input space and then use the gradient

information to make a step downhill. Then they recalculate the gradient at the new location and repeat the process.

The simplest gradient algorithm is called "gradient descent" and uses the information of the first derivative to find the direction of the learning step. The learning process can be interpreted as a particle θ, which moves trough the domain of the error function. Frequently the particle is given mass, in which case the position of the particle can be calculated by the formulas

$$v_t = \omega v_{t-1} + \epsilon \nabla \mathcal{E} \tag{2.21a}$$

$$\theta_{t+1} = \theta_t + v_t \tag{2.21b}$$

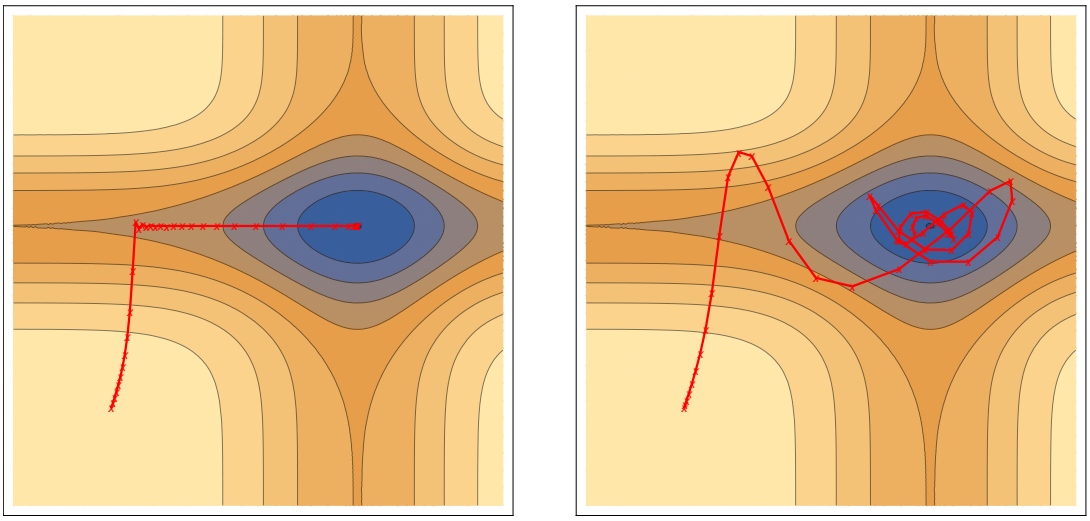
where $\epsilon \in (0, 1)$ is the learning rate and controls the size of the steps. Very large steps will cause the algorithm to "jump over" the minimum and diverge. Very small learning rate might take too long time to converge to the minimum. The parameter ω controls the velocity of the particle. If the particle has no mass, then $\omega = 0$. If velocity is used when training neural networks, the performance of the network is improved, because it helps the algorithm to escape saddle points. Frequently ω is in the range $[0.7, 0.9]$.

Figure 2.7 shows the trajectories of a particle without mass, Figure 2.7a, and with mass, Figure 2.7b. The algorithm was applied on the Michalewicz's function [98],

$$f(x, y) = sin(x) sin\left(\frac{x^2}{\pi}\right)^{20} + sin(y) sin\left(\frac{2y^2}{\pi}\right)^{20} \tag{2.22}$$

In both cases the algorithm starts from the same point. When no mass is used the particle travels north, then turns east and converges at the minimum. When mass is used the particle oscillates around the minimum before convergence.

A major issue with this class of algorithms is that they get trapped in the local minimum. Therefore, the gradient algorithms are very sensitive to the initialisation.



(a) (b)

Figure 2.7: The learning steps of gradient descent applied on the Michalewicz's function are plotted against the contour plot of the function. On the left is the trajectory generated by algorithm where the particle doesn't have mass and the right is the trajectory when mass is used.

2.2.2 Gradient Descent for Neural Networks

The parameters of the neural network can be derived from a set input-output pairs, $(\boldsymbol{x}^n, \boldsymbol{y}^n)$, where $n = 1..N$ and $\boldsymbol{y} \in \mathbb{R}^K$, by using the gradient descent algorithm. The error function can be defined as a sum of error of each sample

$$\mathcal{E} = \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}^n \tag{2.23}$$

where n is the index over the training samples. Then the error of each sample can be defined as a sum of errors of each dimension

$$\mathcal{E}^n = \sum_{k=1}^{K} e_k^n \tag{2.24}$$

where k is the index over the dimensions of the output of the network. The function e_k^n has to express the discrepancy between the desired outputs, $\boldsymbol{y}^n$, and the actual outputs of the network, $\boldsymbol{o^{(L)}}^n$. For simplicity e^n will be abbreviated as e. An example of error function is the squared distance

$$e_k = \frac{1}{2} \left(\boldsymbol{o^{(L)}}_k - y_k \right)^2 \tag{2.25}$$

Another example, which is useful in classification, is the cross-entropy. The outputs of the network are interpreted as the probability of each class. The data is labelled using 1 of K labelling. If the class of sample x^n is k', then $y_k = 0$ for $k \neq k'$ and $y_{k'} = 1$. The negative log likelihood would result in the cross-entropy error function

$$e_k = -y_k log \left(softmax \left(\boldsymbol{o^{(L)}} \right)_k \right) \tag{2.26}$$

In this case there has to be a softmax layer after output layer, so that the outputs of the network can represent probability of a class.

$$softmax \left(\boldsymbol{o} \right)_k = \frac{e^{\boldsymbol{o}_k}}{\sum_{k'} e^{\boldsymbol{o}_{k'}}} \tag{2.27}$$

After the error function is defined, the minimum point is found by following the opposite direction of the gradient of the error function,

$$\Delta \boldsymbol{W}^{(l)} = \epsilon \frac{1}{N} \sum_{n=1}^{N} \Delta \boldsymbol{W}^{(l)^n} \tag{2.28a}$$

$$\Delta \boldsymbol{b}^{(l)} = \epsilon \frac{1}{N} \sum_{n=1}^{N} \Delta \boldsymbol{W}^{(l)^n} \tag{2.28b}$$

where $\epsilon \in (0, 1)$ is the learning rate and

$$\Delta \boldsymbol{W}^{(l)^n} = -\frac{\partial \mathcal{E}^n}{\partial \boldsymbol{W}^{(l)}} \tag{2.29a}$$

$$\Delta \boldsymbol{b}^{(l)^n} = -\frac{\partial \mathcal{E}^n}{\partial \boldsymbol{b}^{(l)}} \tag{2.29b}$$

In order to calculate the derivative of the error function with respect to each element, the error needs to be propagated, backward through the network layers. The result of the derivations are the recursive formulas

$$\frac{\partial \mathcal{E}^n}{\partial \boldsymbol{W}^{(l)}} = \boldsymbol{\delta}^{(l)} \boldsymbol{o}^{(l-1)^T} \tag{2.30a}$$

$$\frac{\partial \mathcal{E}^n}{\partial \boldsymbol{b}^{(l)}} = \boldsymbol{\delta}^{(l)} \tag{2.30b}$$

$$\boldsymbol{\delta}^{(l)} = \begin{cases} \boldsymbol{o}^{(L)'} \circ \boldsymbol{e}' & \text{if} \quad L = l \\ \boldsymbol{o}^{(l)'} \circ \left(\boldsymbol{W}^{(l+1)^T} \boldsymbol{\delta}^{(l+1)} \right) & \text{otherwise} \end{cases} \tag{2.30c}$$

$$\boldsymbol{e}'_k = \frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{o}_k^{(L)}} \tag{2.30d}$$

$$\boldsymbol{o}_k^{(l)'} = \frac{\partial \boldsymbol{o}_k^{(l)}}{\partial \boldsymbol{v}_k^{(l)}} \tag{2.30e}$$

The superscript n is omitted for convenience. This is the simplest gradient descent algorithm for neural network. There are many variations of this algorithm. According

to comparison of variations of gradient descent [99], the best performing algorithms is ADAM [100], which is similar to RMSProp [101], however it also includes a momentum.

RMSProp

The difference between standard gradient descent and RMSProp (Root Mean Square Propagation) is that gradient descent has the same learning rate for all parameters and RMSProp has different rate for each parameter, based on the decaying average of the magnitude of the previous gradients of that parameter. Since all gradient algorithms are iterative, the iterations can be interpreted as discrete time. The algorithm would start from initial point at time $t = 0$ and every iteration is an increment in time. If all the parameters of the network at time t are represented as a vector $\theta(t)$, then the update rule for parameter $\theta_i(t)$ is

$$\theta_i(t+1) = \theta_i(t) - \frac{\eta}{\sqrt{\mu_i(t) + \epsilon}} \frac{\partial \mathcal{E}}{\partial \theta_i(t)} \tag{2.31}$$

Where ϵ is a smoothing constant, typically in the range of 10^{-6}, and $\mu_i(t)$ is the decaying average of the gradient at time t

$$\mu_i(t) = \gamma \mu_i(t-1) + (1-\gamma) \left(\frac{\partial \mathcal{E}}{\partial \theta_i(t)} \right)^2 \tag{2.32}$$

According to the authors of the algorithm, good values for $\gamma = 0.9$, and $\eta = 0.001$.

ADAM

Adaptive Moment Estimation (ADAM) is another algorithm which uses decaying average to estimate the learning rate of each parameter. However it also keeps expo-

nentially decaying average of previous gradients, similar to momentum.

$$m(t) = \beta_1 m(t-1) + (1-\beta_1)\frac{\partial \mathcal{E}}{\partial \theta(t)} \tag{2.33a}$$

$$\mu(t-1) = \beta_2 \mu(t-1) + (1-\beta_2)\left(\frac{\partial \mathcal{E}}{\partial \theta(t)}\right)^2 \tag{2.33b}$$

Where β_1 and β_2 are real numbers in the interval $(0, 1)$. The algorithm is designed to have larger gradients at the beginning of the training than the end

$$\hat{m(t)} = \frac{m(t)}{1-\beta_1^t} \tag{2.34a}$$

$$\hat{\mu(t)} = \frac{\mu(t)}{1-\beta_2^t} \tag{2.34b}$$

The update rule for the parameters of the neural network is

$$\theta(t+1)_i = \theta(t)_i - \frac{\eta}{\sqrt{\hat{\mu(t)}_i + \epsilon}}\hat{m(t)}_i \tag{2.35}$$

The authors of the algorithm [100] suggest default values for $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$

2.2.3 Choosing Initial Parameters

A common disadvantage of all gradient based methods is that they could converge into local minima of the error function, which makes them very sensitive to the starting point of the algorithm. For neural networks there are several methods which are designed to avoid this problem.

Nguyen and Widrow

Nguyen and Widrow [102] suggest a method which is specifically designed for neural networks with one hidden layer and non-linear activation function. A good choice for

the activation function is the hyperbolic tangent

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.36}$$

The purpose of the algorithm is to set up the weights and biases, so that for each input there would be at least one hidden unit, which is approximately linear. The $tanh$ is approximately linear in the range $(-1, 1)$, therefore the activation potential v_i needs to be in this interval. The neurons should be distributed randomly across the input field, therefore the directions weight vectors of each unit need to be random.

The algorithm assumes that the inputs are in the range of $(-1, 1)$. If the network has H hidden units and N input units, then the weights and bias of hidden unit i can be initialised using the formulas

$$\hat{\boldsymbol{W}}_{ij} \sim N(0, 1) \tag{2.37a}$$

$$\boldsymbol{W}_{i,\cdot} = \hat{\boldsymbol{W}}_{i,\cdot} \frac{H^{\frac{1}{N}}}{\left|\hat{\boldsymbol{W}}_{i,\cdot}\right|} \tag{2.37b}$$

$$\boldsymbol{b}_i \sim U\left(-\left|\boldsymbol{W}_{i,\cdot}\right|, \left|\boldsymbol{W}_{i,\cdot}\right|\right) \tag{2.37c}$$

where $U(a, b)$ is uniform distribution in the interval (a, b) and $N(\mu, \sigma)$ is Gaussian distribution with mean μ and standard deviation σ. The notation $\boldsymbol{W}_{i,\cdot}$ is the i^{th} row of matrix $\boldsymbol{W}$, and $|\cdot|$ is the norm of a vector.

Xavier

Glorot and Bengio [103] designed an initialisation algorithm, which is appropriate for deeper networks. The purpose of the algorithm is to preserve the variance of the input signal, and the variance of the error back-propagation through the layers.

The algorithm assumes that the input data has mean 0, the derivative of the activation function ψ at 0 is 1, $\frac{d\psi(0)}{dv} = 1$, and the activation function is symmetric around 0.

All the analysis is done with linear activation, however the paper also demonstrates that the algorithm works well with $tanh$ activation.

If $n^{(l)}$ is the number of units in layer l, then the Xavier initialisation sets the weights, so that their variance is reverse proportionate to the sum of the input and output units.

$$Var\left[\boldsymbol{W}_{ij}\right] = \frac{2}{n^{(l)} + n^{(l-1)}} \tag{2.38}$$

The authors suggest to use either Gaussian distribution or Uniform distribution. In the case of Gaussian distribution each weight of layer l is generated by

$$\boldsymbol{W}_{ij}^{(l)} \sim N\left(0, \sqrt{\frac{2}{n^{(l)} + n^{(l-1)}}}\right) \tag{2.39}$$

and in the case of uniform distribution

$$\boldsymbol{W}_{ij}^{(l)} \sim U\left[-\frac{\sqrt{6}}{\sqrt{n^{(l)} + n^{(l-1)}}}, \frac{\sqrt{6}}{\sqrt{n^{(l)} + n^{(l-1)}}}\right] \tag{2.40}$$

2.3 Dimensionality Reduction

When classifying data, the more information the classifier is provided, the better would be the classification. Therefore, frequently the raw data is classified directly, instead of extracting features. For example when working with the MNIST data set [104], some of the best classifiers [105], [106], [107] work directly with the raw images.

This approach has the advantage that avoids loss of information, because it doesn't extract features and provides all the information to the classifier. However there is a disadvantage, the size of the training set required for training a classifier grows exponentially with the number of dimensions of the data. This problem is commonly known as the curse of dimensionality [108], [109] and can be addressed by using dimensionality reduction techniques. These methods remove redundant dimensions, by projecting the data to lower dimensional space.

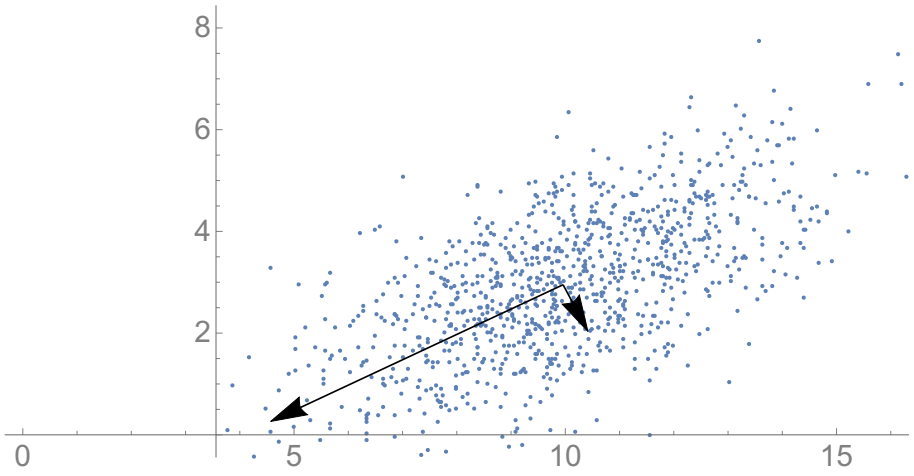


Figure 2.8: The eigenvectors of the co-variance matrix of a data set, which show the directions of the largest variance.

2.3.1 Principal Component Analysis

Principal Component Analysis (PCA) [110, Chapter 12] is a way of removing correlations in data and projecting it to lower dimensional space. The method calculates the eigenvectors of the co-variance matrix of the data, re-orders them by magnitude of the eigenvalues, and projects the data on the first several eigenvectors with the largest eigenvalues. Typically, the number of components is chosen, based on the number of eigenvalues that have sum of at least 95 % of the sum of all eigenvalues. This would capture 95% of the variance in the data.

The method works with the assumption that the data is a linear combination of "building blocks", which are the principal components.

For example, if the data is 2 dimensional, $x_i \in \mathbb{R}^2$, and the first and second dimension have correlation, the method would be able to project the data on one dimension. Figure 2.8 shows a data set generated by 2D Gaussian distribution and the direction of the eigenvectors of the co-variance matrix. Each eigenvector is scaled proportionate to its eigenvalue. In this case the data will be projected on the first eigenvector.

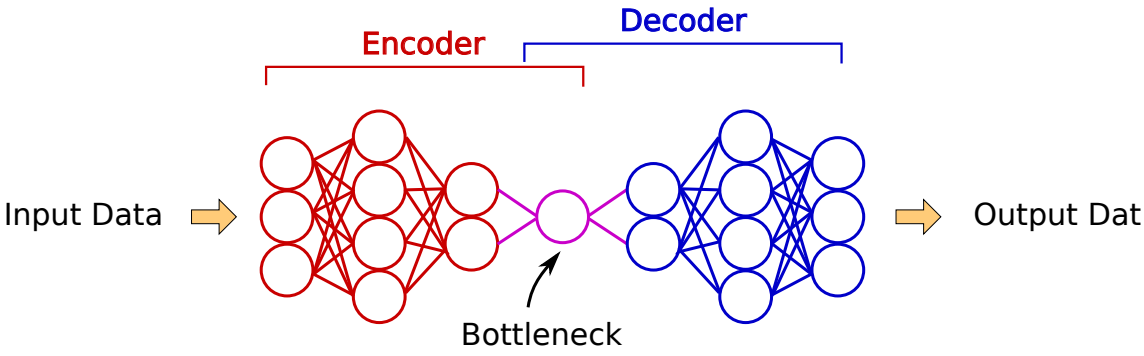


Figure 2.9: The autoencoder is a feed-forward neural network with multiple hidden layers. The number of units is symmetric with respect the central layer, which is called "bottleneck". The part of the network from the input layer to the bottleneck is called encoder, and the rest is called decoder.

A limitation of the method is that it can capture only linear dependencies between the dimensions of the data.

2.3.2 Autoencoder

Autoencoders are used for dimensionality reduction of non-linear data sets [111] [112]. The autoencoder is a feed-forward neural network with several hidden layers, and it is symmetric with respect to the middle layer. The middle layer has lower number of units than the input layer. The first part of the network until the middle layer is called encoder, and the rest of the network is called decoder, Figure 2.9. The network is trained to learn the identity operation. Since the middle layer has fewer units, a successful training of the networks would achieve lower dimensional representation of the data.

If the autoencoder has one hidden layer, then the dimensionality reduction becomes linear, and the network behaves like principal component analysis[113] [114]. Gradient descent applied on deep networks does not work well, because the magnitude of the error signal decays to zero, or becomes exponentially large, when it is propagated backward through many layers [115]. If the signal becomes too small, the network cannot be trained

in reasonable amount of time; if it becomes very large, it may cause the algorithm to "overshoot", which causes oscillations in the weights with increasing magnitude. Therefore, if gradient descent is used on deep networks, it will produce the average of the data[112]. Using meta-heuristic algorithms such as particle swarm optimisation or genetic algorithm to train auto-encoders is not efficient due to the large number of parameters to optimise.

Training Autoencoder

A good way to train autoencoder is by using gradient descent (GD) applied on pre-trained network[112] [116] – Figure 2.10. During the pre-training for each layer, a Restricted Boltzmann Machine (RBM) [117] [118] is created, so that the inputs to the layer are the visible units, and the outputs are the hidden units. The hidden units of each RBM are mapped to the visible units of the next one. The input units of the autoencoder are the visible units of the first RBM, and the output of the bottleneck layer are the hidden units of the last RBM. After the RBMs are trained, the weights of the encoder part of the network are set to the weights of the RBM, and the weights of the decoder part of the network are set to the transposed weights of the RBMs. The visible biases are the biases of the encoder layers, and the hidden biases are the biases of the decoder layer.

Boltzmann Machine

The Boltzmann machine is a stochastic fully connected neural network [80] [119] [120]. The neurons of the Boltzmann machine are binary and can be either in state $+1$

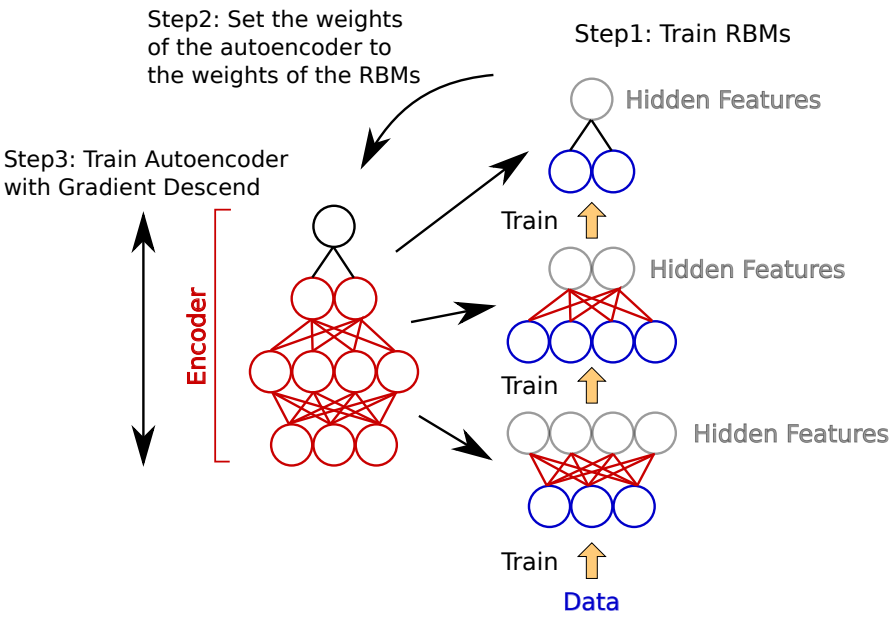


Figure 2.10: The autoencoder is trained by segmenting it into RBMs, training each RBM on the input data, and then using the weights of the RBM as a starting point for gradient descent.

or in state -1. The network with N neurons is in a state $\boldsymbol{x} \in \{-1, +1\}^N$ with probability

$$P(\boldsymbol{x}) = \frac{1}{Z} \mathrm{e}^{-E(\boldsymbol{x})} \tag{2.41a}$$

$$Z = \sum_{\boldsymbol{x}} \mathrm{e}^{-E(\boldsymbol{x})} \tag{2.41b}$$

where Z is the partition function, which makes sure that everything sums up to 1, $\sum_{\boldsymbol{x}}$ is a sum over all possible states of $\boldsymbol{x}$, and $E(\boldsymbol{x})$ is the energy function. The energy function is defined as

$$E(\boldsymbol{x}) = -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x} \tag{2.42}$$

where $\boldsymbol{W}$ is a symmetric matrix and $\boldsymbol{b}$ are set of biases.

The units of the network change states one at a time in a random sequence. This update schedule is asynchronous.

The probability of unit k to be equal to $+1$, given the rest of the units is

$$P\left(x_k = +1 | \boldsymbol{x}^c\right) = \frac{P\left(x_k = +1, \boldsymbol{x}^c\right)}{P\left(x_k = +1, \boldsymbol{x}^c\right) + P\left(x_k = -1, \boldsymbol{x}^c\right)} \tag{2.43}$$

where $\boldsymbol{x}^c$ is the states of all other units except from k. When these probability are written in terms of the energy function and x_k is substituted the probability becomes

$$P\left(x_k = +1 | \boldsymbol{x}^c\right) = \frac{1}{1 + e^{2\left(\sum_{i \neq k} W_{ki} x_i + b_k\right)}} \tag{2.44}$$

This expression is the sigmoid function, $\sigma(x)$, therefore the probability of unit k to be $+1$ can be expressed as

$$P\left(x_k = +1 | \boldsymbol{x}^c\right) = \sigma\left(-2\left(\sum_{i \neq k} W_{ki} x_i + b_k\right)\right) \tag{2.45}$$

This rule can be used to update the units of the Boltzmann machine, because the partition function Z, which is infeasible to evaluate, was cancelled out.

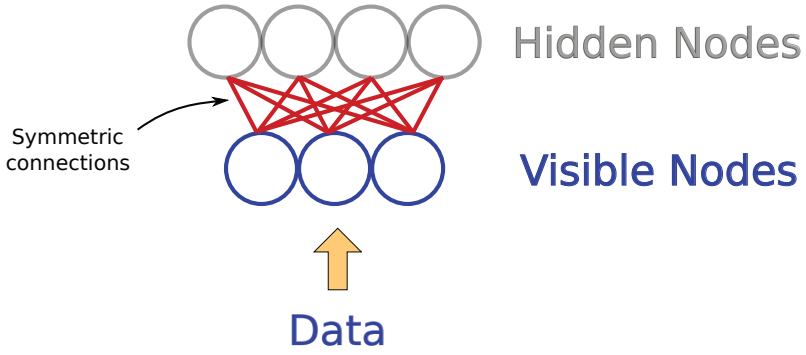


Figure 2.11: RBM has hidden and visible nodes. Every hidden node is connected with every visible node with a symmetric connection.

During the pretraining of auto-encoder are used restricted Boltzmann Machines instead of fully connected Bolzmann machines. This network has two types of units which are visible $\boldsymbol{v}$ and hidden $\boldsymbol{h}$. Every visible units is connected with every hidden unit with symmetric connections, Figure 2.11.

There are no connections between the hidden units and also there are no connections between the visible units. This means that each visible unit is independent on all other visible units and each hidden unit is also independent on all other hidden units. This means that in the connection matrix $\boldsymbol{W}$ there are zeros along the diagonal and all weights between any two visible units and any two hidden units. Having this assumption the energy function can be expressed as

$$E\left(\boldsymbol{v}, \boldsymbol{h}\right) = -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} - \boldsymbol{a}^T \boldsymbol{v} - \boldsymbol{b}^T \boldsymbol{h} \tag{2.46}$$

where $\boldsymbol{W}$ is the part of the matrix $\boldsymbol{W}$ from Formula 2.42 which connects the visible with the hidden units, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are the parameters in $\boldsymbol{b}$ from Formula 2.42 which corresponds to the visible and hidden units.

Since there are no connections between the hidden units and between the visible units, all hidden units can be updated at the same time with probability

$$P(h_k = +1|\boldsymbol{v}) = \frac{1}{1 + e^{2\left(\sum_j W_{ik}^T v_i + b_k\right)}} \tag{2.47}$$

After the hidden units are updated the visible units can be updates as well with probability

$$P(v_k = +1|\boldsymbol{h}) = \frac{1}{1 + e^{2\left(\sum_j W_{kj} h_j + a_k\right)}} \tag{2.48}$$

Training Restricted Boltzmann Machine

Pre-training step of the auto-encoder involves training RBMs. The RBMs defines the probability of the observed in a similar way as the Boltzmann machine, however the hidden units are marginalised out. This can be expressed as

$$P(\boldsymbol{v}) = \frac{1}{Z} \sum_{\boldsymbol{h}} e^{E(\boldsymbol{v}, \boldsymbol{h})} \tag{2.49}$$

where Z is the partition function and $\sum_{\boldsymbol{h}}$ is a sum over all possible states of the hidden units.

The purpose of training and RBM is to minimise the negative log-likelihood over the training data

$$\arg\min_{\boldsymbol{\theta}} \left(-\mathcal{L}(\boldsymbol{\theta})\right) \tag{2.50}$$

where

$$-\mathcal{L}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} P(\boldsymbol{v}_n, \boldsymbol{h}|\boldsymbol{\theta}) \tag{2.51}$$

The derivative of Formula 2.50 with respect to parameter θ_i (weight or bias) is

$$-\frac{1}{N}\frac{\partial}{\partial \theta_i}\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{v}_n)\frac{\partial}{\partial \theta_i}E(\boldsymbol{v}, \boldsymbol{h}) - \sum_{\boldsymbol{v},\boldsymbol{h}} P(\boldsymbol{v}, \boldsymbol{h})\frac{\partial}{\partial \theta_i}E(\boldsymbol{v}, \boldsymbol{h}) \tag{2.52}$$

The update rules for each parameter θ_i can be derived from the derivative of the negative log likelihood function

$$\Delta\theta_i = \epsilon \left(\left\langle \mathbb{E}\left[-\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}_n, \boldsymbol{h}) \middle| \boldsymbol{v}_n \right] \right\rangle_{n=1}^{N} - \mathbb{E}\left[-\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}, \boldsymbol{h}) \right] \right) \tag{2.53}$$

where the constant ϵ is the learning rate, $\langle \cdot \rangle$ is the average over data set, $\mathbb{E}[\cdot]$ is expectation, and $\mathbb{E}\left[-\frac{\partial}{\partial \boldsymbol{\theta}_i} E(\boldsymbol{v}, \boldsymbol{h}) \middle| \boldsymbol{v} \right] = \sum_{\boldsymbol{h}} p\left(\boldsymbol{h}|\boldsymbol{v}\right) \frac{\partial}{\partial \boldsymbol{\theta}_i} E(\boldsymbol{v}, \boldsymbol{h})$. The summation over all possible values of the visible and hidden nodes makes calculation of the updates unfeasible. Therefore, RBM is trained using Contrastive Divergence (CD) [121] [122]. Contrastive divergence is a way to approximate the maximum likelihood approach by calculating average over observed data and sampled data instead of the expectation.

The first expectation $\left\langle E\left[-\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}_n, \boldsymbol{h}) \middle| \boldsymbol{v}^p \right] \right\rangle_n$ is approximated with the average over the values of the derivative of the energy function for each data point where the values of the visible units are set to the corresponding value of the data point and the hidden units are samples, given the visible units.

The second expectation $E\left[-\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}, \boldsymbol{h}) \right]$ can be approximated by setting the values of visible units to each data point and sampling hidden units given the visible units and then the visible units given the hidden units k times, Figure 2.12. Each sample can be accepted according to the Monte Carlo Markov Chain (MCMC) rules. Since only probability ratio needs to be calculated the partition function cancels out. According to Hinton [122] only one step of sampling is enough to find the direction to minimise the negative log-likelihood. So the approximate update for each parameter is

$$\Delta\theta_i = \epsilon \left(\left\langle -\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}(\boldsymbol{0})_n, \boldsymbol{h}(\boldsymbol{0})_n) \right\rangle_n - \left\langle -\frac{\partial}{\partial\theta_i} E(\boldsymbol{v}(\boldsymbol{k})_n, \boldsymbol{h}(\boldsymbol{k})_n) \right\rangle_n \right) \tag{2.54}$$

where $\boldsymbol{v}(\boldsymbol{0})$ is the value of the visible units when set to the training data, $\boldsymbol{h}(\boldsymbol{0})$ the value of the hidden units when sampled given $\boldsymbol{v}(\boldsymbol{0})$, and $\boldsymbol{v}(\boldsymbol{k})$ and $\boldsymbol{h}(\boldsymbol{k})$ are the values of the visible and the hidden units after k samples. In contrastive divergence usually

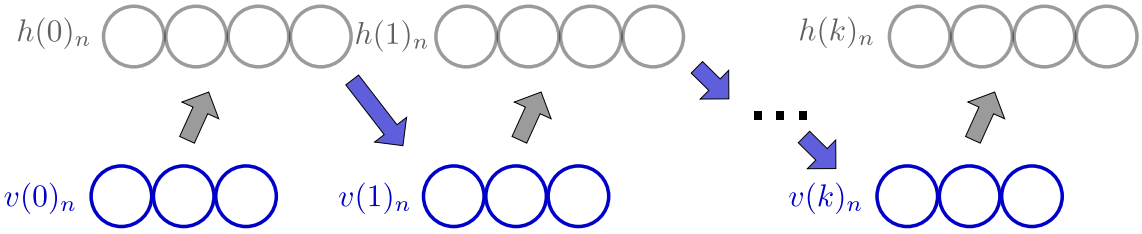


Figure 2.12: The values of the visible units are set to a data point at time-step 0 $v(0)_n$, then the values for the hidden units $h(0)_n$ are sampled from the visible units. At time-step 1 the values of the visible units $v(1)_n$ are sampled from the values of the hidden units in the previous time step, $h(0)_n$, then the values of the hidden units $h(1)_n$ are sampled from the values of the visible units $v(1)_n$. This is repeated k times.

$k = 1$.

Calculating the derivatives of the energy function, shown in Formula 2.46, and replacing in Formula 2.54 results in the update rules for the weight matrix W and the biases a and b.

$$\Delta W = -\epsilon \left(\left\langle \boldsymbol{v}(\boldsymbol{o})^T \boldsymbol{h}(\mathbf{0}) \right\rangle_{n=1}^{N} - \left\langle \boldsymbol{v}(\boldsymbol{k})^T \boldsymbol{h}(\boldsymbol{k}) \right\rangle_{n=1}^{N} \right) \tag{2.55a}$$

$$\Delta a = -\epsilon \left(\langle \boldsymbol{v}(\mathbf{0}) \rangle_{n=1}^{N} - \langle \boldsymbol{v}(\boldsymbol{k}) \rangle_{n=1}^{N} \right) \tag{2.55b}$$

$$\Delta b = -\epsilon \left(\langle \boldsymbol{h}(\mathbf{0}) \rangle_{n=1}^{N} - \langle \boldsymbol{h}(\boldsymbol{k}) \rangle_{n=1}^{N} \right) \tag{2.55c}$$

The learning is often more efficient if velocity α is introduced into the learning rules [123]

$$\theta_i(t+1) = v\Delta\theta_i(t) + \Delta\theta_i(t+1) \tag{2.56}$$

Continuous Autoencoder

When pre-training an auto-encoder, the units of the RBM need to be continuous. Also the activation function of the hidden units of the RBMs needs to be the same as the

activation function of corresponding hidden layer of the encoder part of the autoencoder. Similarly the activation function of the visible units of the RBMs needs to be the same as the activation function of the corresponding hidden layer of the decoder part of the autoencoder. The activation function used in this project is the hyperbolic tangent function, Formula 2.57.

$$Tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.57}$$

With continuous units randomness can be added by adding random value drawn from Gaussian distribution with mean 0 and standard deviation σ to each unit before applying the activation function.

The $Tanh$ function is chosen over sigmoid, because it is symmetric around zero, which makes the training easier [96], [97].

After the autoencoder is constructed by using the trained RBMs, it is trained using error back-propagation.

An autoencoder was used to compress non-linear data, Figure 2.13. The data was generated by adding Gaussian noise to the sin function. The figure shows compression and reconstruction done by the autoencoder and by principal component analysis (PCA). The PCA failed to capture the non-linear nature of the data and mapped all data points to a line, while the autoencoder learned the sin curve, managed correctly to reconstruct the data and ignored the Gaussian noise.

2.4 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test checks if a set of one dimensional data points, $D = \{d_i\}, d_i \in \mathbb{R}, i = 1..n$ comes from a given probability distribution, $f(x)$, [124]. The test returns a p-value where the null hypothesis is that the data set comes from the suggested probability distribution. A low value of the test would suggest that the data

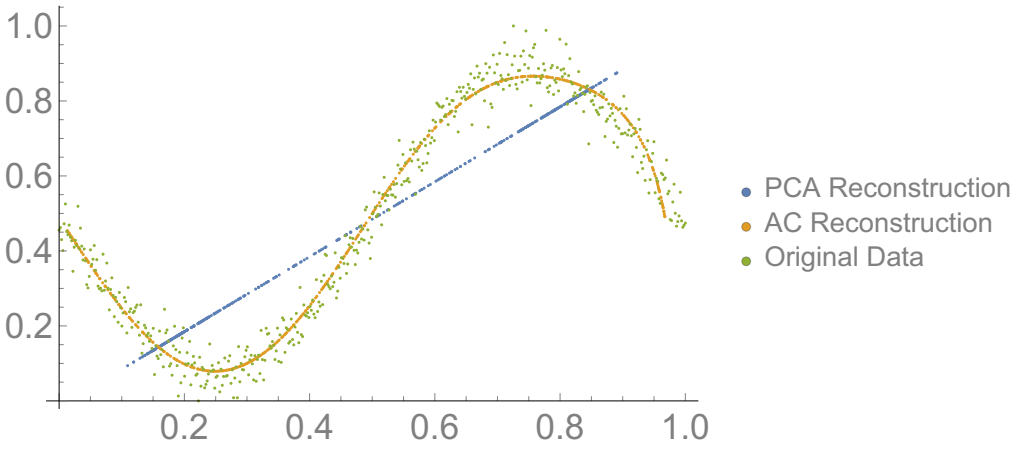


Figure 2.13: The original data (green dots) is compressed to one dimension and then reconstructed with autoencoder (orange dots), and PCA (blue dots).

set D does not come from $f(x)$.

First the KS test estimates the empirical Cumulative Density Function (CDF), $\hat{F}_D(d)$, of the process which generated the data set D

$$\hat{F}_D(d) = \frac{1}{n}\sum_{i=1}^{n} I(d_i < d) \tag{2.58}$$

where I return 1 if the argument is true and 0 otherwise

$$I(x) = \begin{cases} 1 & \text{if } x = \text{true} \\ 0 & \text{otherwise} \end{cases} \tag{2.59}$$

Then it finds the largest difference between the empirical CDF and the CDF of the suggested probability distribution, $F_X(x)$

$$M = \max_{-\infty<x<\infty} \left|\hat{F}_D(x) - F_X(x)\right| \tag{2.60}$$

After the largest difference between the empirical CDF and the suggested CDF is found the KS test converts this difference to a p-value using the formula

$$P = Q_{\mathrm{KS}}\left(\left(\sqrt{n} + 0.12 + \frac{0.11}{\sqrt{n}}\right) M\right) \tag{2.61}$$

where Q_{KS} is complement of the CDF of the Kolmogorov-Smirnov distribution, $Q_{\mathrm{KS}} = 1 - F_{\mathrm{KS}}(x)$, and

$$F_{\mathrm{KS}}(x) = 1 - 2\sum_{j=1}^{\infty}(-1)^{j-1}e^{-2j^2x^2} \tag{2.62}$$

This series converges very quickly so only few of the terms are enough to estimate an approximate value for $F_{\mathrm{KS}}(x)$.

It is possible to use the KS test to check if two data sets, $X = \{x_i\}$, $x_i \in \mathbb{R}$, $i = 1..n_x$ and $Y = \{y_i\}$, $y_i \in \mathbb{R}$, $i = 1..n_y$, come from the same distribution. In this case both CDFs, $\hat{F}_X(x)$ and $\hat{F}_Y(x)$, have to be calculated like in Formula 2.58. Then the largest difference between the two CDFs is calculated by

$$M = \max_{-\infty<x<\infty}\left|\hat{F}_X(x) - \hat{F}_Y(x)\right| \tag{2.63}$$

The largest difference between the two CDFs is converted to a p-value with the formula

$$P = Q_{\mathrm{KS}}\left(\left(\sqrt{n_e} + 0.12 + \frac{0.11}{\sqrt{n_e}}\right) M\right) \tag{2.64}$$

where n_e is the effective number of data points,

$$n_e = \frac{n_x n_y}{n_x + n_y} \tag{2.65}$$

.

2.5 Applications of Machine Learning to Single Molecule Analysis

Deep learning algorithms provide very high degree of flexibility and therefore allow solving of complex problems when large dataset sets are available. Naturally this allowed

scientists to successfully apply deep learning algorithms to various problems in super-resolution microscopy.

2.5.1 Super-Resolution

For STORM to work it requires to fit a model of PSF, usually a Gaussian fit, on a diffraction limited spot which is generated by a single emitter. Usually statistical tests are employed to verify that the PSF is generated by a single emitter. The stochastic switching provides diffraction limited images with well separated single emitters, however there is a requirement for low density labelling to reduce co-occurrences. Deep learning was successfully to mitigate this problem, thus increasing the emitter density.

Nehme et al, who developed an extension of STORM, called Deep Storm [125]. DeepSTORM uses deep learning to produce resolved images from input diffraction limited images, which allows to process frames where there are overlapping emitters, and therefore overcoming the requirement for low emitter density. The methods uses the stochastic switching of emitters in STORM to reduce overlap and allow for even higher emitter density, average of around $6\frac{\text{emitters}}{\mu m^2}$. The method works by training a deep fully convolutional neural network to produce resolved images from diffraction limited images. The network was trained and verified by a simulated dataset, where the emitter positions were known in advance. The same authors extended the method for 3D localisation, DeepSTORM3D [126]. The method used the Tetrapod PSF [127] to extract the depth information and can work with overlapping PSFs.

A similar approach to DeepSTORM is the work by Speiser and Müller, which is called DEep COntext DEpendent (DECODE) [128] [129]. In this approach they use a deep neural network, based on UNet, to predict the locations of the emitters in high density diffraction limited image. However, they also predict the probabilities an emitter to exist in each pixel, it's intensity, the exact coordinates of the emitter with respect of the centre of the pixel, and the uncertainties in the positions of each emitter and it's

intensity. Their network can integrate information over several frames, to take advantage of the sequential nature of the problem, and uses custom build error function. As with the previous work it is also trained on simulated data. The authors claimed that their work outperformed the other contestants on 12 out of 12 datasets from the 2019 SMLM challenge [130].

Another direction for improvement is exploiting the context sensitive properties of the neural networks to reconstruct super-resolution images from much fewer frames in STORM, therefore improving its temporal resolution.

Ouyang et al. used deep neural network to obtain a super-resolution image from a diffraction limited image [131]. This approach exploited the structural redundancies in the most of the biological images to reconstruct the super-resolution image from up to two orders of magnitude fewer frames than usually required by PALM or STORM. The spatial resolution of the images ranged from 20nm to 2nm.

Gaire et al. used deep learning to improve the speed multicolour STORM [132]. The method first images samples and uses STORM to find localisations from 10^4 images. This problem can be reformulated as image in-painting task, at which deep neural network have been very successful [133], [134], [135], [136], [137]. Then a deep convolutional network was trained to reconstruct the localisations from much fewer diffraction limited images, 300 - 1000, and thus allowing for much higher temporal resolution. This method was trained on different structures such as microtubules, mitochondria, and peroxisome. Using the deep neural network's ability to extract, remember and generalise over structures in images means that the results might depend on the structure imaged and lead to bias in case of localisation of randomly placed emitters or different structures.

Another problem which was successfully addressed by machine learning algorithms is detection of diffraction limited spots. The machine learning approach redefines this problem as a classification problem, where it tries to minimise the false positives and false negatives.

Colabrese et al. approached the diffraction limited spot detection as a classification problem [138]. They used support vector machines (SVM) to determine whether each pixel is a centre of a diffraction limited spot or not. After that they used a Gaussian fitting to determine the sub-pixel localisation. This algorithm was tested on simulated data from the 2013 IEEE ISBI SMLM challenge [139] where it achieved 87.2% on the Jaccard metric, $\frac{\mathrm{TP}}{\mathrm{TP+FP+FN}}$.

Cascarano et al. use deep learning for single molecule localisation [140], however they extend the method by making use of the CEL0 reguliser [141]. The new approach provides a parameter free solution, which is more flexible and showed improved results on the datasets from the 2013 IEEE ISBI SMLM challenge [139]. The performance was measured in terms the Jaccard metric $\frac{\mathrm{TP}}{\mathrm{TP+FP+FN}}$, which had best score of 61.48%, sensitivity $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, 70.12%, and specificity $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$, 99.96%.

2.5.2 Semantic Segmentation and Instance Detection

In biological studies frequently structure identification is required after imaging. Sometimes this is done manually, however it is very time consuming task and the field would benefit from automation. The task of biological structures identification and tracking can be approached as semantic or instance segmentation task.

Ronnenberger et al. developed fully convolutional network for semantic segmentation of biological images, called UNet [142]. The network outperformed other methods for segmentation of neuronal structures in electron microscope stacks. It also used for cell segmentation in light miscroscopy and it won the 2015 ISBI cell tracking challenge.

Schmidt, Weigert et al. developed a instance segmentation algorithm for detecting highly crowded cell nuclei in 2D and 3D [143] [144]. The method is based on UNet architecture and can detect individual instances. However, unlike Mask R-CNN it doesn't use axis-locked bounding boxes but star-convex polygons instead. This change avoids the problem with overlapping instances and can work with images with high density. The

method was compared with the Mask R-CNN method and it outperformed it, regardless that it uses simpler and easier to train architecture.

In another development by Hollandi et al. used instance clustering to find nuclei of cells [145]. This method used Mask R-CNN to find the approximate sizes of the nuclei of the cells. Then resizes the cells so that all of the cells can have approximately the same size nuclei and again Mask R-CNN architecture is used for instance segmentation. Finally the boundaries of the segmented nuclei are refined using UNet architecture. The method uses style transfer based on conditional GANs for data augmentation.

2.5.3 Image Enhancement

Single molecule localisation methods can achieve increased resolution if the diffraction limited images are pre-processed appropriately. Deep learning has been successfully applied to image denoising [146] and background removal.

Möckl et al. used deep learning to remove structured background from the PSF neighbourhood [147]. The method works by training a deep neural network, based on the UNet, to predict background only, when supplied with an image of the PSF together with the background. Then the background estimate was subtracted from the original image to obtain just the PSF without the background. The network was trained on simulated data with different PSF shapes. The experiments included the standard open aperture PSF, the double-helix PSF, and the Tetrapod PSF to test the suitability for 3D microscopy. The experiments also included a random PSF to test the performance on non-symmetric shapes. The methods managed to improve substantially the localisation precision.

Weigert et al. used UNet architectures to different single molecule related tasks [148] by exploiting the ability of the deep networks to remember biological structure. The method is called content-aware image restoration (CARE). One of the problems they tackled with the encoder-decoder architecture is the image denoising. They used

a microscope with two lasers to acquire images with high and low signal to noise ratio of the same sample. Then they used the deep network to reconstruct the high signal to noise images from the low signal to noise images. Another problem they wanted to address was enhancing the axial resolution of microscope images. That was done by taking 2D images and modifying them through realistic simulations to represent PSF distortions and then trained the CARE network to find the depth from the FPS distortion.

Wang et al. used deep learning to enhance diffraction limited images from confocal microscope [149]. Their work uses pairs of experimentally acquired high resolution images and low resolution images to train generative adversarial networks (GANs). The resulting network was general enough to enhance resolution of different microscope setups. The algorithm transformed images acquired by a wide-field microscope using a $10\times/0.4$-NA objective into images which matched the resolution that can be achieved by $20\times/0.75$-NA objective. The same algorithm then enhanced a diffraction limited images acquired by a STED microscope by improving the PSF size from 290nm to 110nm.

2.5.4 Other

Apart from single molecule localisation, machine learning has been used to aid molecule structure identification through single molecule nano-pore sensing [150] (SMNPS). In SMNPS two containers are connected trough nano-pore with diameter of few nanometres and filled with liquid. Then an electrical field drives charged molecules trough the nano-pore. When the electric current is measured at the time of the molecule passing trough the nano-pore the value would follow a pattern specific for a particular structure. In this work detecting these patterns before analysis was done using convolutional neural network. The advantage over other algorithms used in the field for SMNPS is that this method is non-parametric and doesn't require supervision.

Another successful application of deep learning to single molecule microscopy is extracting particle traces from kymographs. Jakobs et al. presented a software package,

called KymoButler, which extracts particle traces from kymographs automatically [151]. The package can handle particles which disappear, reappear, merge, cross each other's path, move in any direction, change speed, immobilise, and reverse direction. For this purpose the package uses deep fully convolutional neural network, based on the UNet architecture.

Ounkomol et al. used a UNet based architecture to generate fluorescence from transmitted light images [152]. The same method is also able to predict immunofluorescence images from electron micrograph inputs. The method works by training a UNet architecture on a pairs of input and output images, where the inputs are transmitted light images and the outputs are fluorescence images.

2.5.5 Software Products

Sometimes it can be hard for biological teams to put together the entire tool-chain for training and verification of deep learning models. Therefore, a team lead by G. Jacquemet and R. Henriques created a could based software called ZeroCostDL4Mic [153]. This software can be accessed through a web interface and provides the hardware and software support for training a wide variety of architectures for different microscopy related tasks, such as semantic segmentation, super-resolution from diffraction limited images, image denoising, and image to image translation. The software is organised as Jupyter notebooks and contains number of architectures, which are trained for different tasks and are also available for training on user specific task from scratch or using transfer learning:

- Super-resolution. For this task the DeepStorm architecture is implemented and trained [125].
- Semantic segmentation and object detection.
  - UNet [142]

  - StarDist [143] [144]
  - YOLOv2 [154]
- Denoising
  - CARE [148]
  - Noise2Void [146]
- Image 2 Image translation
  - Cycle GAN
  - pix2pix
  - fnet

2.5.6 Shortcomings of Deep Learning Algorithms

While successfully applied to many applications, the deep learning algorithms have shortcomings. It is important these to be known during integration into a microscope data processing tool-chain, so they can be addressed appropriately.

In general all deep learning methods for single molecule localisation are susceptible to introducing bias in case they are trained on structured data and then deployed to analyse random data or data with different structure. This issue is called the hallucination problem [155]. A good example for such problem was provided by Belthangady et. al. [156] where they showed that if UNet was trained to reconstruct degraded images of English words it works fine when an English word is reconstructed. However, if it was trained to reconstruct symbols of different alphabet, it fails to provide correct reconstruction.

Another issue with the neural networks is the generalisation ability. In case the training set is incomplete or too small the neural network will "remember" the training

set and would perform poorly on a novel data. Such issues can be addressed by using the dropout technique or L-regularisation.

Deep networks can become very sensitive to the inputs to the point where imperceptible perturbations of the input can case dramatic changes to the output, the adversarial fragility problem [156]. Synthetically adding noise which can be expected in a real word data can help mitigate such problems. Other techniques involve modifying the error function to include the norm of the derivative of the neural network with respect to the inputs, like in contractive auto-encoders [157]. Lower norm of the derivative value would mean that the network would be less sensitive to the input changes.

Chapter 3

Simulations

Simulated tracks are good way to verify that the algorithms work as expected, because the ground truth is known. Simulations are used in Chapter 4 to evaluate the quality of level detection algorithms, Chapter 5 to verify that the data representation can distinguish between analysable and non-analysable tracks, and Chapter 6 to verify the new track selection process.

This chapter provides a detailed description of all of the simulations used in this project. It shows details about the implementation of the simulations, provides justification for different choices and shows how the simulation parameters are deduced.

The chapter starts with Section 3.1 where a description of the imaging process is provided. Then Section 3.2 describes the architecture of the EMCCD and explains the different sources of noise. Section 3.3 describes the model of the point spread function of the microscope and the noise model of the sensor. Section 3.4 shows how the simulation parameters are chosen and Section 3.5 provides details of the implementation of the simulations. Finally Section 3.6, Section 3.7, and Section 3.8 describe the set-ups and the use-cases for the simulations used in chapters "Level Detection", "Track Classification", and "Refining Track Selection Process".

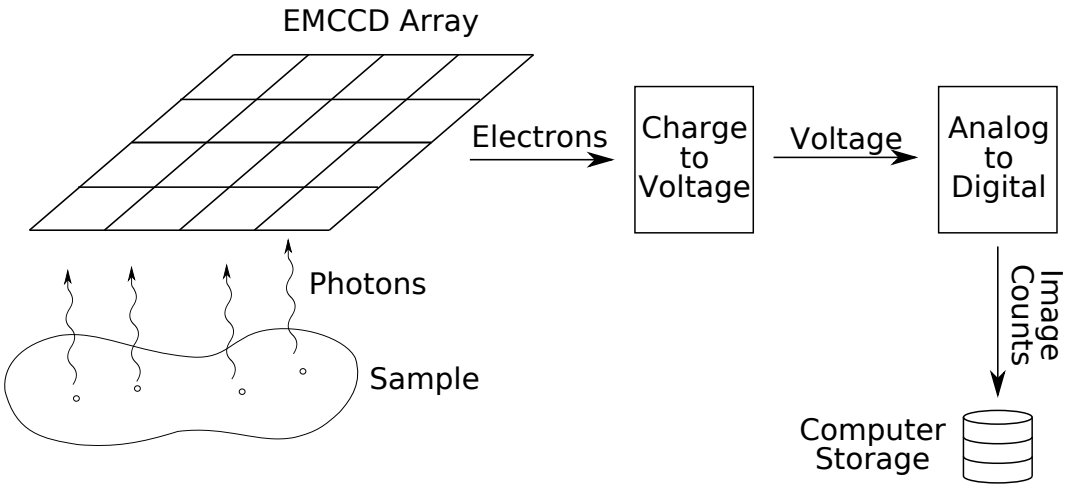


Figure 3.1: Imaging process in FLImP.

3.1 Imaging Process

During the imaging process the fluorophores in the sample are excited with a laser, and they emit light with some frequency. Photons from the light sources (fluorophores) are absorbed by the pixels of the detector, electron multiplier charge coupled device (EMCCD), where they are converted to electric charge. This charge is then converted to a voltage, which is read out then sent to analogue to digital converter and stored as integer value, called image count, Figure 3.1. This process generates noise, which has to be modelled by the simulation.

The simulation starts with the position of a fluorophore, then models the point spread function, which determines how many photons are expected to arrive in each pixel for the duration of the frame. Then based on the expected number of photons for each pixel and the EMCCD parameters it adds noise to generate the image counts for each pixel. Then these images can be processed by Quincy and analysed by FLImP the same ways real microscope images are handled.

3.2 EMCCD Architecture and Sources of Noise

The first step of the imaging process is converting photons into electrons [158]. This process happens in a device called *phototransistor*. A diagram of a phototransistor is shown on the left-hand side of Figure 3.2. It consists of a region of p-type semiconductor, a layer of non-conductive material (SiO_2), and metal electrode. This structure is also called MOS capacitor. The semiconductor side is connected to the ground and positive electrical pressure is applied to the metal electrode.

When a photon arrives at the photogate it creates electron-hole pair. The hole disappears into the ground and the electron is attracted to the positive charge and is stored at the semiconductor region. The probability of this event to occur is called *quantum efficiency*.

In the p-type semiconductor region of the MOS capacitor sometimes there are free electrons without a photon arrival. The probability of free electrons to occur is proportionate to the temperature. These electrons are called *spurious charge* and can cause *dark current* to occur in devices such as diodes. The spurious charge is considered to be part of the noise in the system.

These phototransistor are arranged into CCD array, so that the charges can be moved through the array into a readout register. This process is controlled by changing the electrical pressure at the gates. This is shown in the middle and the right of Figure 3.2.

After the electrons are moved to the read-out register they are moved to *electron multiplier* device, left hand side of Figure 3.3. This device works on the principle of impact ionisation. In this process a high electrical pressure is applied, which accelerates the electrons. When they collide with a molecule from the semiconductor they transfer part of their kinetic energy to another electron which becomes a free electron. This way the charge carriers multiply in an avalanche process. The ratio of input to output electrons in such device is called *EM gain*.

After the electrons go through the EM register they are converted to voltage, right

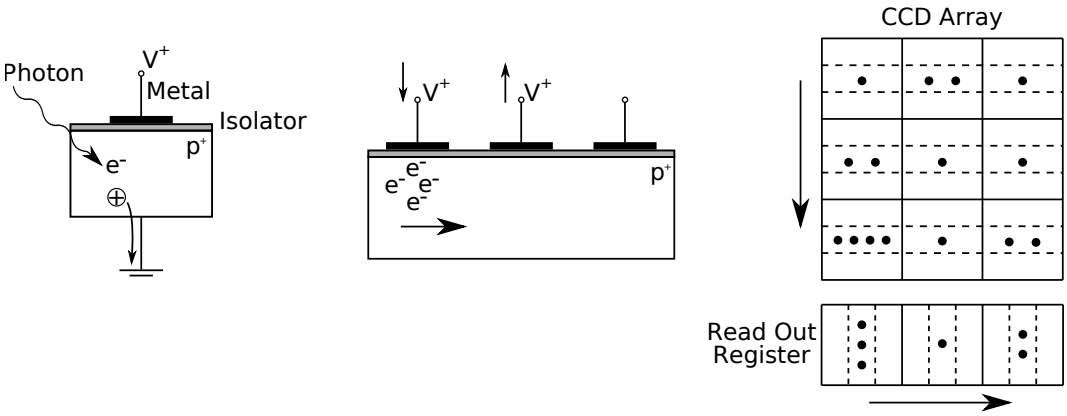


Figure 3.2: A diagram of a photogate (left), charge moving trough neighbouring photogates (middle), and the CCD array (right) [158].

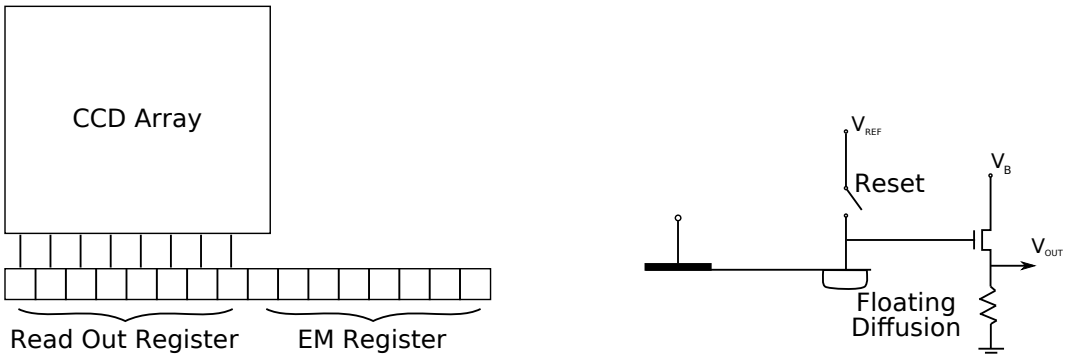


Figure 3.3: Electron Multiplier Register (left) and electrons to voltage converter (right) [158].

hand side of Figure 3.3, where the voltage is linearly dependant on the number of electrons. During this process there is a *read out noise*.

The voltage then goes through analogue to digital converter, where it is assigned an integer number. The digital number, referred to as *image value* has 14 bits, which means there are 2^{14} distinct image values. The number of electrons per one image count is called *Analogue/Digital factor*.

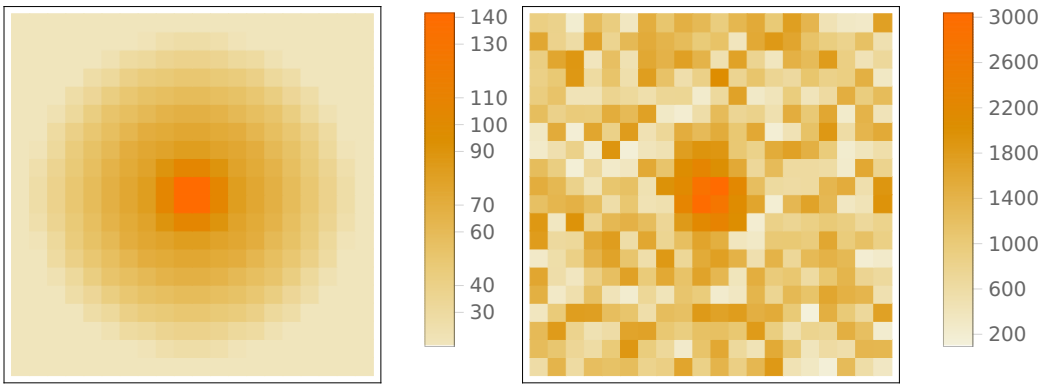


Figure 3.4: A simulated diffraction limited spot before adding noise (left) and after adding noise (right).

3.3 Diffraction Limited Spot

Simulation of a track starts with simulating background and diffraction limited spot. The background has uniform intensity. The diffraction limited spot is approximated with the Gaussian function. More realistic model of the diffraction limited spot is the Airy disk, however Gaussian approximation is close enough and it is used in FLImP [62] and other single molecule simulations[159].

The Gaussian profile is scaled so that it has maximum value of 1. Then in each simulation this is multiplied by the brightness of the fluorophore, measured in photons per frame. For example if the fluorophore has brightness of 60 photons per frame, the highest value of the diffraction limited spot is 60.

The left hand side of Figure 3.4 shows an image of diffraction limited spot and background before adding the noise.

3.3.1 Microscope Noise

When images are generated before adding the noise the each pixel value is the expected number of photons for that pixel. Then a noise model for the EMCCD is used

Parameter	Meaning	Value	Units
q	quantum efficiency	0.92	electrons per photon
c	spurious charge (dark current and clock induced charge)	0.02	electrons
g	gain of EM	250	dimensionless
r	read out noise	50	electrons
f	Analogue/Digital factor	12.7	electrons per image value

Table 3.1: EMCCD parameters and their values for a FLImP experiment [160]

to estimate the image values [160]. The value of each pixel is a random sample drawn from the probability of image counts, x, given the EMCCD parameters, (q, c, g, r, f), and the expected number of photon, n_{ph}, for that pixel

$$p\left(x|q,c,g,r,f,n_{ph}\right) = \frac{1}{\sqrt{2\pi}r} e^{-\lambda - \frac{(fx)^2}{2r^2}} + e^{-\lambda - \frac{fx}{g}} \sqrt{\frac{\lambda}{fxg}} I_1\left(2\sqrt{\frac{\lambda fx}{g}}\right) \tag{3.1a}$$

$$\lambda = n_{ph}q + c \tag{3.1b}$$

where I_1 is modified Bessel function of first kind, which can be expressed as

$$I_1(x) = \frac{x}{2} \sum_{j=0}^{\infty} \frac{\left(\frac{x^2}{4}\right)^j}{j!\Gamma(j+2)} \tag{3.2}$$

The noise model, Equation 3.1a, consists of a sum of two exponential terms. The first exponential term models the readout noise and the A/D converter. The second exponential term models the process of conversion of photons in the CCD array and the process of the signal amplification in the EM register.

The EMCCD parameters meaning and their values for a FLImP experiment are given in Table 3.1.

3.4 Fluorophore and background intensity

To generate a simulated frame the fluorophore and background intensity need to be specified. Choosing such intensities can be done by simulating tracks which start with region with two active fluorophores, followed by a region with one active fluorophores and comparing the track intensities with the intensity of real tracks. The real tracks which were compared to the simulations are tracks selected during year 2014 for papers publishing the results of FLImP experiments. Figure 3.5 shows kernel density estimation of the distributions of the signal to noise ratio when only one fluorophore is active for simulated track with different brightness and real tracks. Figure 3.6 shows the kernel density estimation of the intensity of tracks when only one fluorophore is active for simulated and real tracks. Based on the distributions shown on these figures the fluorophore brightness can be set to 60 photons. It is also possible to set fluorophores with higher brightness, for example 120, when higher signal to noise ratio is desired. When simulating fluorophore which is deeper in the cell it is possible to user lower than 60 values such as 30. Such values can be used to test the level detection algorithm or the FLImP filter.

Figure 3.7 shows the kernel density estimation of the distribution of background for simulated tracks and tracks processed during 2014. Based on these distributions, the background intensity can be set to 30, which is the value used for the most of the simulations.

3.5 Implementation

The code which generates the simulated images is implemented in "Wolfram Mathematica" version 11.3. Then the images are processed by Quincy, which does feature detection and tracking. The tracks generated by Quincy are read by code written in c++ and exported to a database, "PostgreSQL", from where they can be imported into

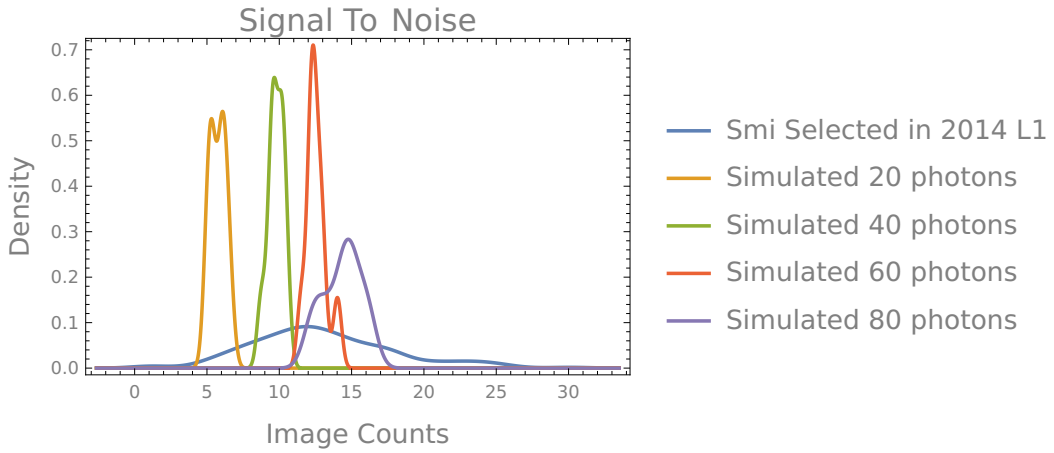


Figure 3.5: Kernel density estimation of the distribution of the signal to noise ration for level 1 of simulated and real tracks.

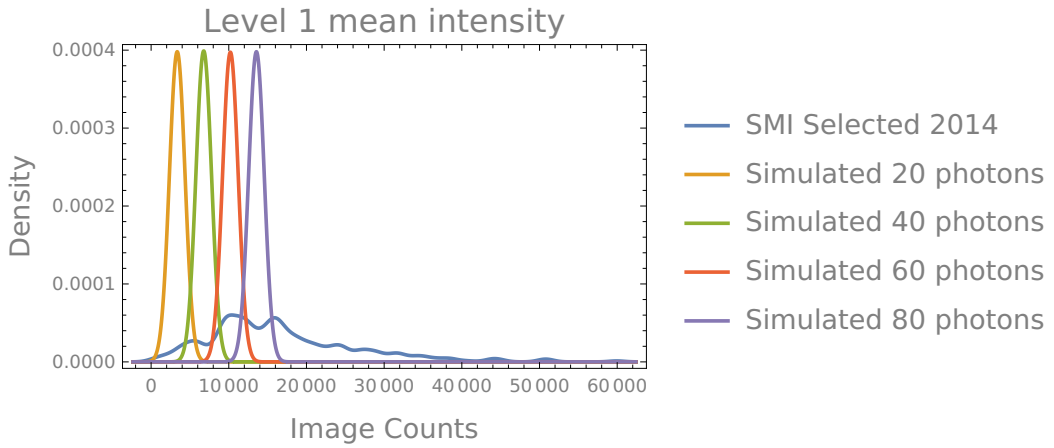


Figure 3.6: Kernel density estimation of the distribution of the intensity when only one fluorophore is active for simulated and real tracks.

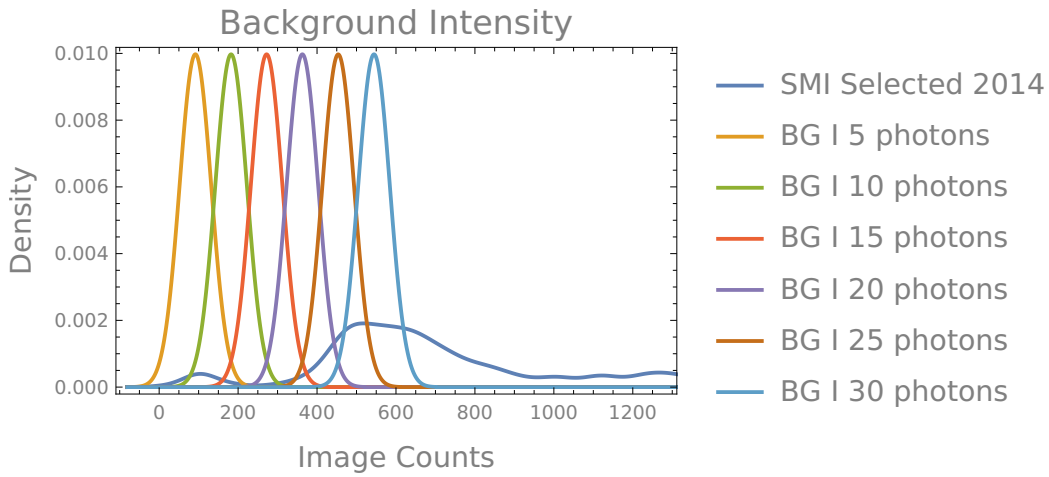


Figure 3.7: Kernel density estimation of the intensity of the background for simulated tracks and real tracks.

Mathematica for experiments such as feature extraction, classification, level detection and so on. This process is shown in Figure 3.8

The simulations typically are square image of 100x100 pixels. The size of each pixel is assumed to be 160nm, which is the pixel size in the FLImP data sets. Unless otherwise states in one such image there are 9 diffraction limited spots organised in a grid of 3x3, where each diffraction limited spot is separated with 25 pixels from each neighbour or

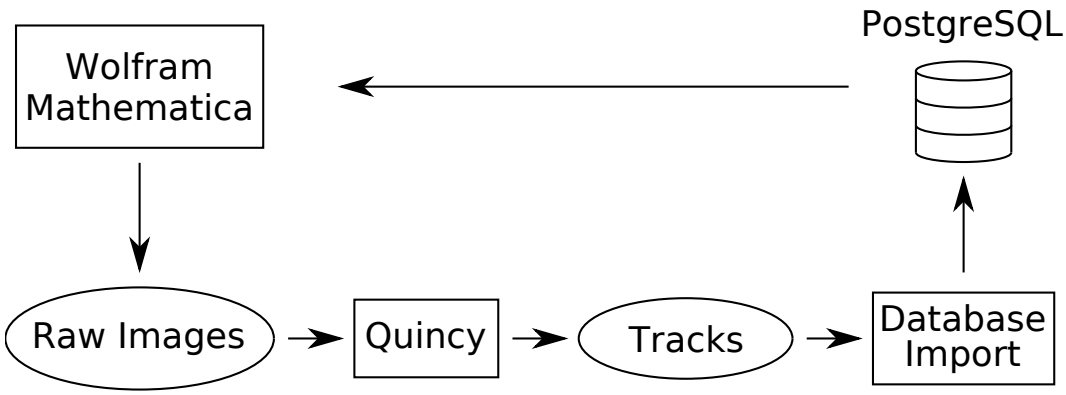
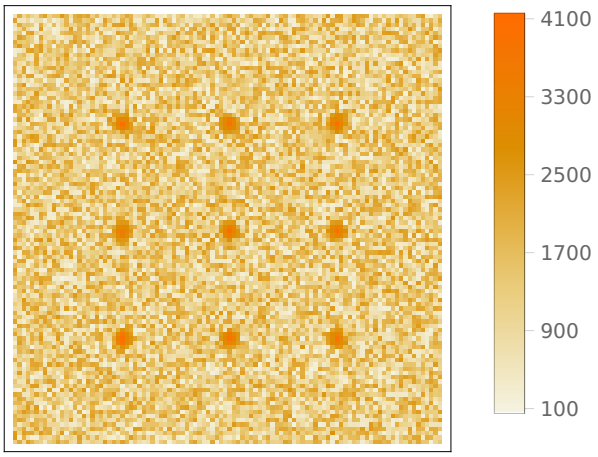


Figure 3.8: Simulation processing pipeline

Figure 3.9: Typical frame has 9 diffraction limited spots.



the end of the image, Figure 3.9.

Each diffraction limited spot in a frame can be generated by one or more fluorophores. Each fluorophore can be either in emitting or non-emitting state. When generating a data set, there are many frames generated as a movie. The states of the fluorophores in each frame can be set manually or by Markov chain. Usually all the fluorophores in a frame are in the same sate. Therefore, in a data set all tracks go trough the same states, and the only difference is the noise.

Image counts are drawn from the noise model by the library *RandomVariate* from the Mathematica framework.

Since the image counts are dependant on the brightness of each pixel, each image value has to be drawn from separate distribution. This includes numerical integration to calculate the cumulative density function, which has to be inverted. This is expensive operation, which makes simulations very slow, especially if it has to be repeated for each pixel in each image. If an experiment has 400 frames and each frame has 100x100 pixels this means that there has to be drawn 1 random value from $4 * 10^6$ distributions.

This process can be optimised, because many pixels have the same values. For example the pixels at equal distance from the centre of the PSF are with the same value (in photons). Also frames which have the same states of the fluorophores are identical.

Therefore, a map is used, where each key is image value in photons and the value is list of random samples for this photon counts.

3.6 Level Detection

Simulated tracks are used for training a level detection algorithm and also testing the performance of 3 level detection algorithms.

Each simulated track contains 3 fluorophores. One of these fluorophores is always in *emitting* state until the end of the track, when it goes to *photobleached* state. The states of the other two fluorophores are determined by a Markov chain with three states, which are *emission*, *dark*, and *photobleached*. Each track is long 400 frames. After the end of each track there are 20 frames where all of the fluorophores are not in emitting state.

There are 3 types of tracks generated by a Markov chain.

- *Large intensity change.* In this data set the fluorophores are bright enough so that there is large intensity change, compared to the level noise, during transitioning to *dark* or *photobleached* states. The fluorophore intensities are 60, 120, and 80 photon per frame. The last fluorophore is always on until the end of the track. The probability of state transition for the Markov chain are shown in Table 3.2.

  This type of tracks are easiest for the level detection algorithm to process. Example of this type of tracks is shown in Figure 3.10

- *Small intensity change.* In this data set one of the fluorophores is assumed to be deeper in the cell so it is very dim. The fluorophore intensities are 30, 80, and 80 photons per frame. The last fluorophore is always on until the end of the track. The probability of state transition for the Markov chain are shown in Table 3.2.

  This set of tracks produces small intensity changes, which are easy to mistake for

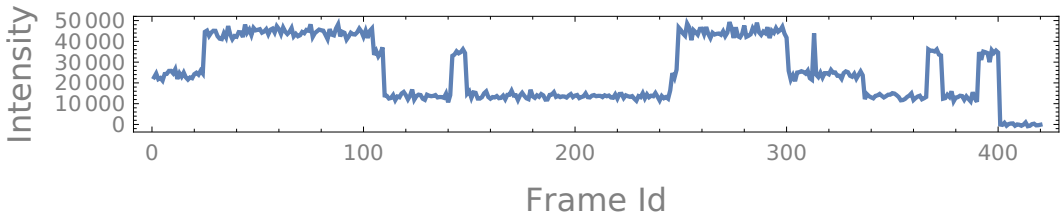


Figure 3.10: Example of a tracks used for the training set which has large intensity changes.

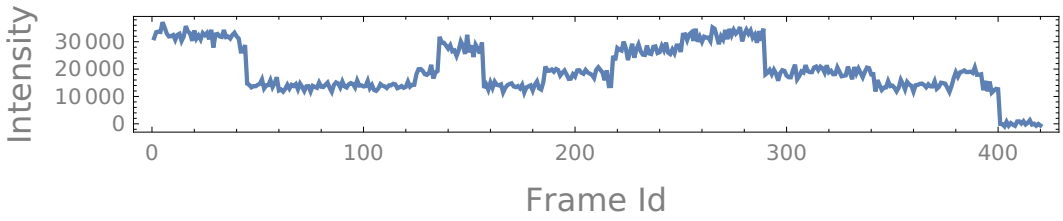


Figure 3.11: Example of a track used for the training set which has small intensity step.

noise. Example of this type of tracks is shown in Figure 3.11

- *Short levels.* This data set has tracks which have quick transition between states, resulting in very short levels. The fluorophore intensities are 60, 120, 80 photons per frame. The last fluorophore is always on until the end of the track. The probabilities of state transitions for the Markov chain are shown in Table 3.2. Example of this type of tracks is shown in Figure 3.12.

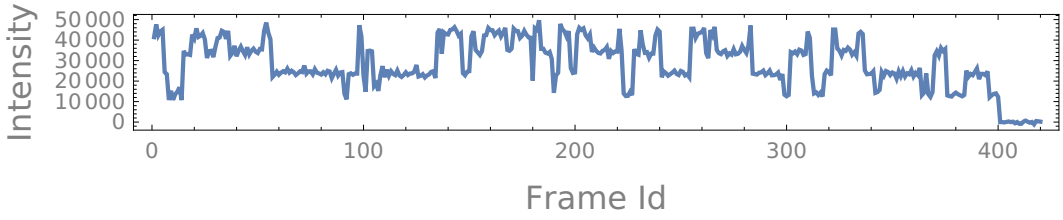


Figure 3.12: Example of a track used for the training set which has short levels.

		To Emission	To Dark	To Photobleach
Large Int. Change, Small Int. Change	From Emission	0.98	0.018	0.002
	From Dark	0.02	0.98	0
	From Photobleach	0	0	0
Short Levels	From Emission	0.92	0.078	0.002
	From Dark	0.08	0.92	0
	From Photobleach	0	0	0

Table 3.2: Transition probabilities for Markov Chain used to generate simulated tracks for level detection.

3.7 Track Classification

The experiments in Chapter 5 require classification of simulated tracks to make sure that the data representation and the classifier used in that chapter are appropriate for distinguishing between analysable and non-analysable tracks.

There are 2 types of analysable tracks. The first type starts with zero intensity for 20 frames. Then one fluorophore is active, which creates level 1, after which another fluorophore becomes active, which creates level 2, and both fluorophore are active until the end of the experiment.

The other type of positive track starts with both fluorophores active, level 2, then one of them becomes dark and only one is active, level 1. After that the second fluorophore becomes dark and there are no active fluorophore for 20 frames.

Both types of tracks are generated with levels of 50 frames each and 100 frames each. On the top of Figure 3.13 is shown track which starts with level 1 and on the bottom of the figure is shown track which starts with level 2. The extrapolated regions are shown in red.

In the positive data sets both fluorophores have intensity of 80 photons per frame

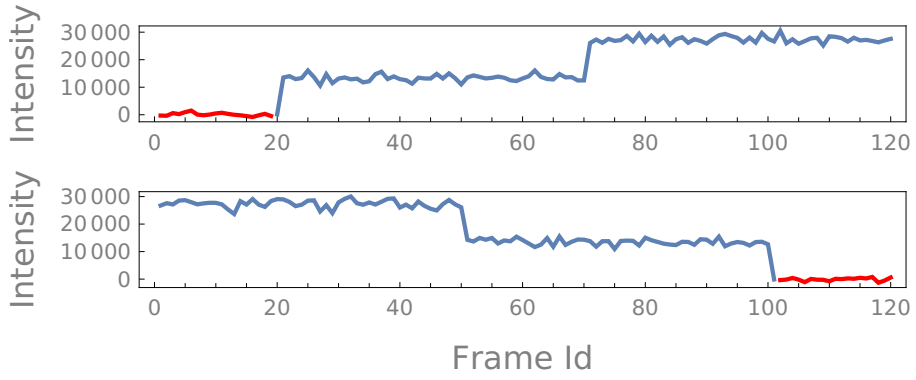


Figure 3.13: Simulated analysable tracks. Level 1 is first in the top track and level 2 is first in the bottom track.

(This is the highest value of the PSF) and the background has intensity of 30 photon.

In the set of negative tracks there are 9 different types. Each type violates different requirement of the track selection criteria. There is a separate data set for each type of tracks. In a data set there are 9 tracks of the same type. In each data set the background intensity is 30 photons per frame.

- *Step Ratio* This data set simulates fluorophores at different depth. Each track is generated by 2 fluorophores in level 2 and one fluorophore in level 1. The tracks start with level 2, which lasts for 100 frames, followed by level 1, which lasts for 100 frames, and then followed by 20 frames in which no fluorophore is active.

  The distance between the two fluorophores for each track range from 5 to 45nm at intervals of 5nm.

  The fluorophore which photobleaches second, fluorophore 2, has intensity of 80 photons per frame at the centre of the PSF. The intensity of the fluorophore which photobleaches first, fluorophore 1, is calculated as a coefficient multiplied by the intensity of the first fluorophore, $\alpha * 80$, where $\alpha \sim N(0.5, 0.1)$ for 4 tracks and $\alpha \sim N(1.7, 0.1)$ for 5 tracks. Figure 3.14 shows an example of a track in which fluorophore 1 has larger intensity (it is closer to the sample surface) than

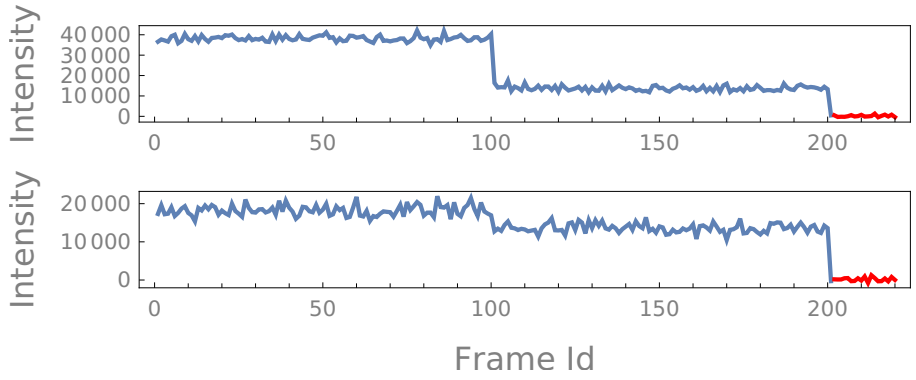


Figure 3.14: Simulated tracks in which the fluorophores are at different depth.

fluorophore 2 (top) and an example of a track in which fluorophore 1 has smaller intensity (it is deeper in the sample) than fluorophore 2 (bottom).

- *Additional Fluorophore* In this type of tracks there is an additional fluorophore in the diffraction limited spot. There are two cases for this data set. The first case there is small intensity for 3 frames before level 1, then follows level 1 and then follows level 2. The second case there is level 2, followed by level 1, which is followed by 3 frames with small intensity. In these tracks there are 3 fluorophores in level 2 and two fluorophores in level 1. Level 1 and level 2 last for 100 frames each.

  The fluorophores which photobleach in level 2, fluorophore 2, and level 1, fluorophore 1, have intensity of 80 photons per frame. The additional fluorophore has random intensity drawn from uniform distribution between 20 and 40 photons, $U(20, 40)$. The distance between fluorophore 1 and fluorophore 2 is 10nm and the additional fluorophore is in the middle between the two fluorophores. Figure 3.15 shows an example of two tracks in this data set.

- *Extrapolated Intensity* In this type of tracks the extrapolated intensity after (or before) the track is different from zero. There are two cases for this type. The

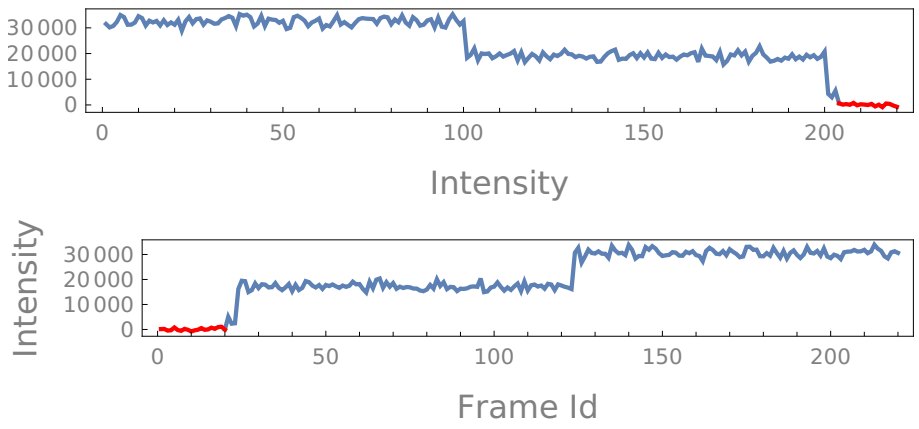


Figure 3.15: Simulated tracks in which there is an additional fluorophore.

first case the track starts with level 1, then follows level 2 and the second case the track starts level 2 and then is level 1. Level 1 and level 2 last 100 frames each.

There are 4 tracks in which there is additional fluorophore in the diffraction limited spot. In these tracks the fluorophores which photobleach in level 2 and level 1 have intensity of 80 photons per frame and the additional fluorophore, which is in the middle between the other 2 fluorophores, has intensity drawn from uniform distribution between 5 and 7 photons per frame. These tracks have extrapolated intensity larger than zero.

The other 5 tracks in the data set have neighbouring fluorophore which has intensity drawn from uniform distribution between 80 and 90 photons per frame. The neighbouring fluorophore is 4 pixels away from the two centre of the diffraction limited spot which generated the track. When these tracks are extrapolated the intensity is lower than 0. On Figure 3.16 is shown extrapolated intensity larger than 0 on top and smaller than 0 on the bottom.

- *Interruptions* In this type of tracks the fluorophores switch on and off more frequently than usual. The states of the fluorophores in this type are generated by a Markov chain. The tracks in one data set have the same states of the fluorophores

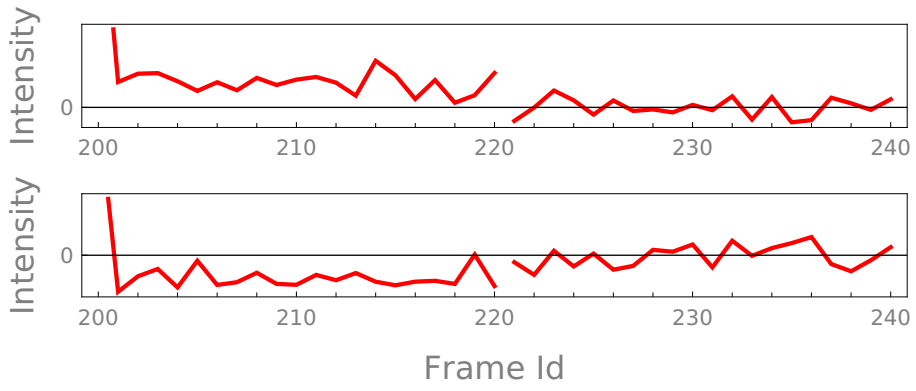


Figure 3.16: Extrapolated intensity from tracks with additional fluorophore (top) and neighbouring fluorophore (bottom).

	to E	to D	to P
from E	0.95	0.045	0.005
from D	0.2	0.8	0.0
from P	0.0	0.0	1.0

Table 3.3: The state transition probabilities of the Markov chain used to generate the states of the fluorophores for tracks of type *Interruption*.

and only differ by the noise. The tracks between different data sets have different states of the fluorophores. Each fluorophore can be in either emitting state (E), dark state (D) or photobleached (P). The probabilities used for the Markov chain are shown in Table 3.3.

An example of track of this type is shown on Figure 3.17

- *Level 1/2 Gap Intensity* The tracks from this type have a gap between level 1 and level 2 of 3 frames with intensity which is different from the levels. In a data set in 3 of the tracks the intensity of that gap is higher than level 2, in the next 3 tracks it is between level 2 and level 1 and in the last 3 tracks it is lower than

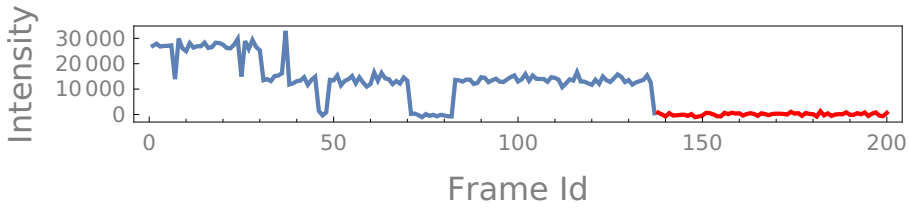


Figure 3.17: Example of a track in which fluorophores change states too frequently.

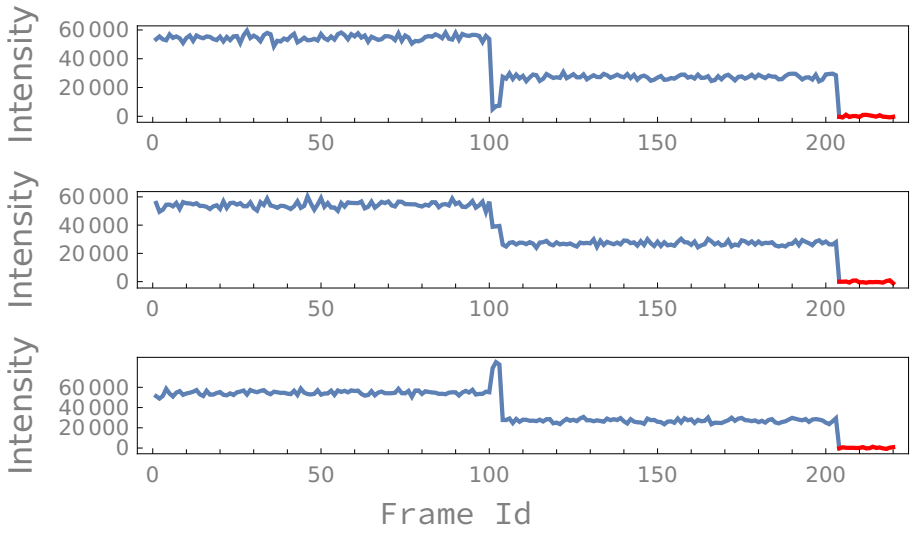


Figure 3.18: Example of a track which has gap between level 1 and level 2 the intensity of the frames in the gap is different from the intensities of level 1 and level 2.

level 1. The fluorophores which photobleach in level 1 and level 2 have intensities of 160 photons per frame. The levels are 100 frames long. The distance between the fluorophores in level 1 and 2 is 10nm. Figure 3.18 shows an example track of each case.

- *Neighbour* The tracks from this type have a fluorophore in the neighbourhood. The neighbouring fluorophore is 5 pixels away in a random direction from each track. The separation between the fluorophores in each track is 10nm and the levels are 100 frames long. The intensity of the fluorophores in each track are 80 photons per frame and the intensity of the neighbouring fluorophore is random

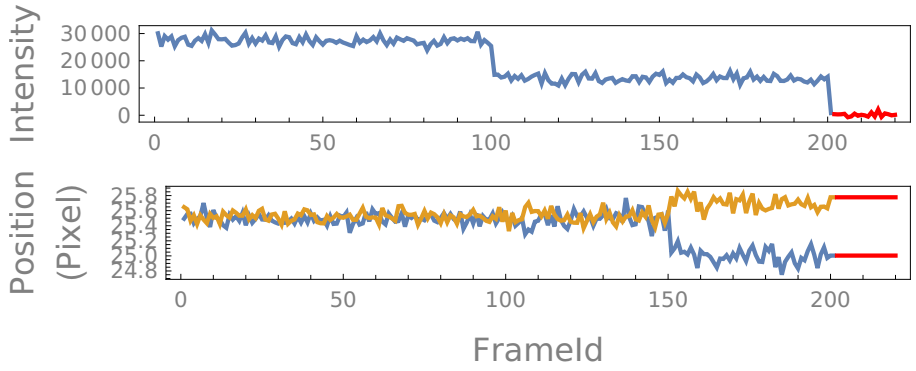


Figure 3.19: Example of a track in which the fluorophore in level 1 moves around 80nm.

number drawn from a Gaussian distribution, $N(60, 10)$.

- *Position Shift* The tracks from this type have a change in the position of the fluorophore which is active during level 1. The fluorophore moves in a random direction random distance. The direction is drawn from a uniform distribution and the distance is drawn from a Gaussian distribution, N(80nm, 5nm). The brightness of the fluorophores is 80 photons per frame, the distance between the fluorophores in the beginning of the track is 10nm and the levels are 100 frames long. An example of track of this type is shown in Figure 3.19

3.8 Refining Track Selection Process

The track selection process is verified by using simulated tracks. The simulations have 9 types of tracks and one data set with 9 tracks of each type was created. Eight types of tracks are not supposed to pass through the filter. Six types are not analysable and two types can be analysed but have too large confidence interval or separation. One type of tracks is analysable with good confidence interval and separation and therefore it should pass through the filter.

- *Step Ratio* Each track of this type is generated by two fluorophores at different depth. The same simulation conditions were used for verification of the track classification in Chapter 5. Details of this simulation are described in Section 3.7.

- *Extrapolated Intensity* Each track of this type is generated by 3 fluorophores. Two of the fluorophores are very bright and form level 2 and level 1. The third fluorophore is always on even after the end of the track and it is very dim, so that it is not detected. When the track is extrapolated however there is intensity after level 1 which is different from zero.

  In this simulation there are 2 cases. The first case the track starts with level 1 and then follows level 2. The second case the track starts with level 2 and then it is followed by level 1. In the first case the track starts after the beginning of the experiment and the extrapolated intensity is before level 1. In the second case the track ends before the end of the experiment and the extrapolated intensity is after level 1. A data set is created for each case.

  The same simulation conditions were used for verification of the track classification in Chapter 5. Details of this simulation are described in Section 3.7.

- *Neighbour* This type of tracks is generated by two fluorophores at the same depth and separated by distance of 10nm. However at a random position within the neighbourhood of the fluorophores, which is 5 pixels away, there is one or two additional fluorophores, which are on all the time.

  The fluorophores which generated the track have intensity of 80 photons per frame at the centre of the PSF, and the additional fluorophore has intensity drawn from Gaussian distribution with mean 60 and standard deviation of 10, $N(60, 10)$. The both levels are 100 frames long. Figure 3.20 shows a frame from a simulated data set of this type.

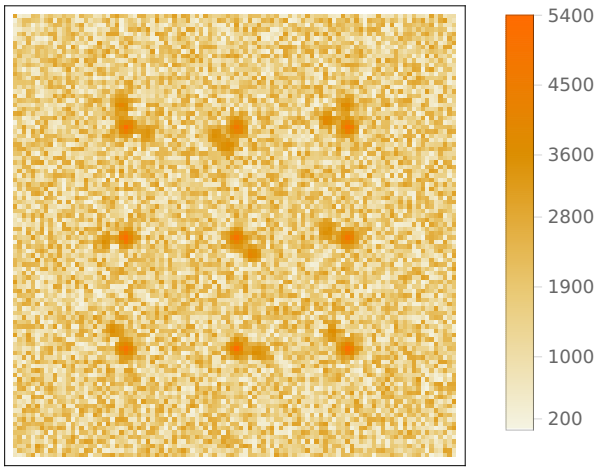


Figure 3.20: A frame from the data set of type "Neighbour". There are fluorophores within the neighbourhood of each track.

- *1 level* Track of this type are generated by only one fluorophore and have only one level. The fluorophore brightness is 80 photons per frame at the centre of the PSF. The level length ranges from 10 to 90 frames at intervals of 10 frames (there are 9 tracks in a data set).

- *Ends with level 3* Tracks of this type do not end with level 1. Normally the level order is level 2 followed by level 1 followed by level 3. The tracks are generated by 3 fluorophores, called fluorophore 1, fluorophore 2, and fluorophore 3. During level 2 fluorophore 1 and fluorophore 2 are active. Both of those fluorophore have brightness of 80 photons per frame. Then during level 1 only fluorophore 1 is active. During level 3 fluorophore 1 and fluorophore 3 are active. Fluorophore 3 has random intensity drawn from Gaussian distribution with mean 120 and standard deviation of 20, $N(120, 20)$, Figure 3.22.

  Distance between fluorophore 1 and fluorophore 2 is 10 nm, and the distance between fluorophore 1 and fluorophore 3 is also 10 nm, Figure 3.21

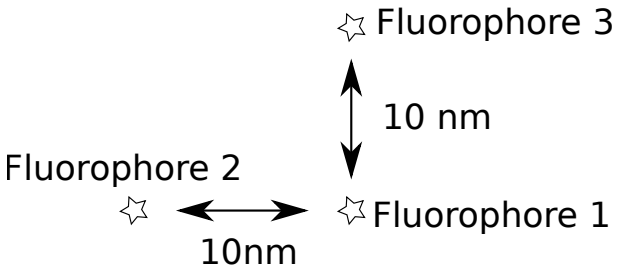


Figure 3.21: Position of the fluorophores.

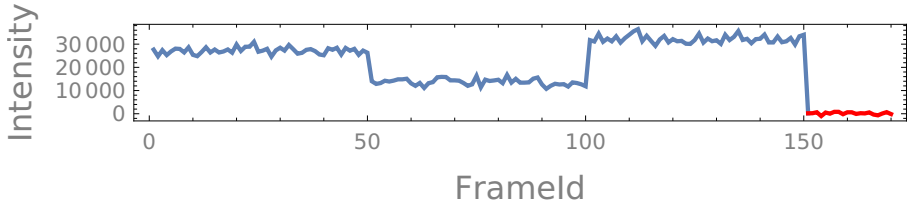


Figure 3.22: Track which ends with level 3 and level 1 is in the middle.

- *Large Confidence Interval* This type of tracks can be analysed by FLImP, however they are likely to produce very large confidence intervals as the levels are very dim and very short. Each track is generated by two fluorophores, which have intensity of 30 photons per frame at the centre of the PSF. The separation between the fluorophores is different for each track ranging from 5 to 40nm at intervals of 5nm. The separation for the last track is 50nm. The levels are 20 frames long.

- *Large Separation* This type of tracks can be analysed by FLImP, however the separations are very large. The separation range between 80 to 160nm at intervals of 10nm, which makes 9 tracks in total. Each track is generated by two fluorophores with intensities of 80 photons per frame at the centre of the PSF each. The levels are 100 frames long.

- *Used* This type of tracks should pass the filter successfully, since they are analysable and have low confidence interval and low separation. Each track is generated by two fluorophores each of which has intensity of 80 photons per frame at the centre

of the PSF. The separations range between 5 and 40nm at intervals of 5nm. The separation of the last track is 50nm. The levels are 100 frames long.

3.9 Conclusion

This chapter provided an overview of the imaging process used for FLImP and a short description of the EMCCD architecture. It described the noise model which was chosen for the simulations, and explained all of its parameters. The chapter described the simulation software, which was developed for this project, and gave some details about the implementation.

The emitter brightness was determined experimentally so that the simulated images were consistent with the real data obtained for FLImP. The experiments compared the simulated and real images based on the signal-to-noise ratio, mean intensity of a single fluorophore and the background intensity. The emitter intensity and the background intensity of the simulated images were evaluated by Quincy the same way as the real data. The parameter values of the noise model used for these experiments were adjusted according to the specifications of the EMCCD cameras used in the FLImP setup.

Finally the chapter provides detailed information about the simulation setup, the conditions, and the goals of the simulations used for each chapter.

Chapter 4

Level Detection

The task of identifying regions in the track with constant intensity is called level detection. Part of the FLImP pipeline involves already existing level detection algorithm, which is referred to as "Bayesian" in this chapter. Details of the algorithm are shown in the introduction, Section 1.6.1. This algorithm makes mistakes as shown in the introduction, therefore new algorithm needs to be designed to reduce the error during level identifications.

In this chapter two new level detection algorithms are proposed, which explore different ideas. One of the new algorithms uses the variance of the intensity to decide whether to split the level and it is referred to as "Level Noise". The other algorithm uses neural network to find level changes and it is referred to as "Neural Network". The performance of these algorithms is compared to the existing level detection algorithms on simulated and real data.

The results of the comparison on the simulated data showed that the algorithm which uses neural network has best performance, however the comparison on the real data showed that best performance is with the algorithm which measures the intensity variance.

After inspecting 5 real tracks on which the algorithms have the worst performance

the algorithm called "Level Noise" was adopted for this project.

4.1 Steps and Level Segments

The transitions of a fluorophore between different states takes less than a nanosecond [161], and a time frame in the FLImP experiments typically lasts hundreds of milliseconds. Therefore, state transitions can be assumed to be instant and cannot occur within more than one frame. Since the duration of the state transitions of the fluorophores is negligible compared to the duration of a frame, the level segments typically start and end with sharp intensity change, called step.

The track intensity for each frame is the integrated intensity of the diffraction limited spot over the duration of the frame. Depending on the part of the frame in which the fluorophore goes into a different state, the intensity of the "transition" frame may have different values anywhere between the two levels before and after the transition – Figure 4.1.

Figure 4.2 shows an example track with two levels. In *level 2* a fluorophore goes to dark state for less than the duration of a frame, which is called blinking event. Then one of the fluorophores goes to dark state, and there is only one active fluorophore – *Level 1*. After that the fluorophore goes back to emitting state, and *level 2* is observed, followed by a photo-bleaching event. After that *level 1* is observed, where only one fluorophore is active, followed by another photo-bleaching event. Since levels can have interruptions due to blinking event, one level can have many constant intensity areas, called level segments.

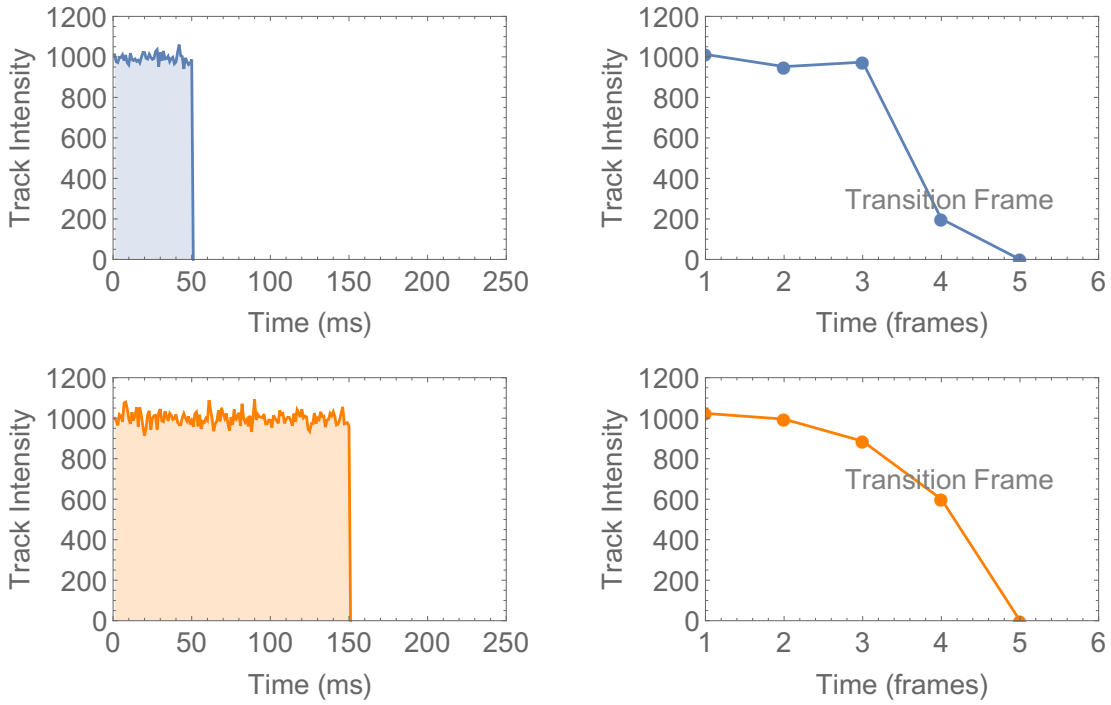


Figure 4.1: The left part of the figure shows a fluorophore transitioning to dark state, or photo-bleaching, in different part of the same frame. The right part of the figure shows track intensity plot for each case.

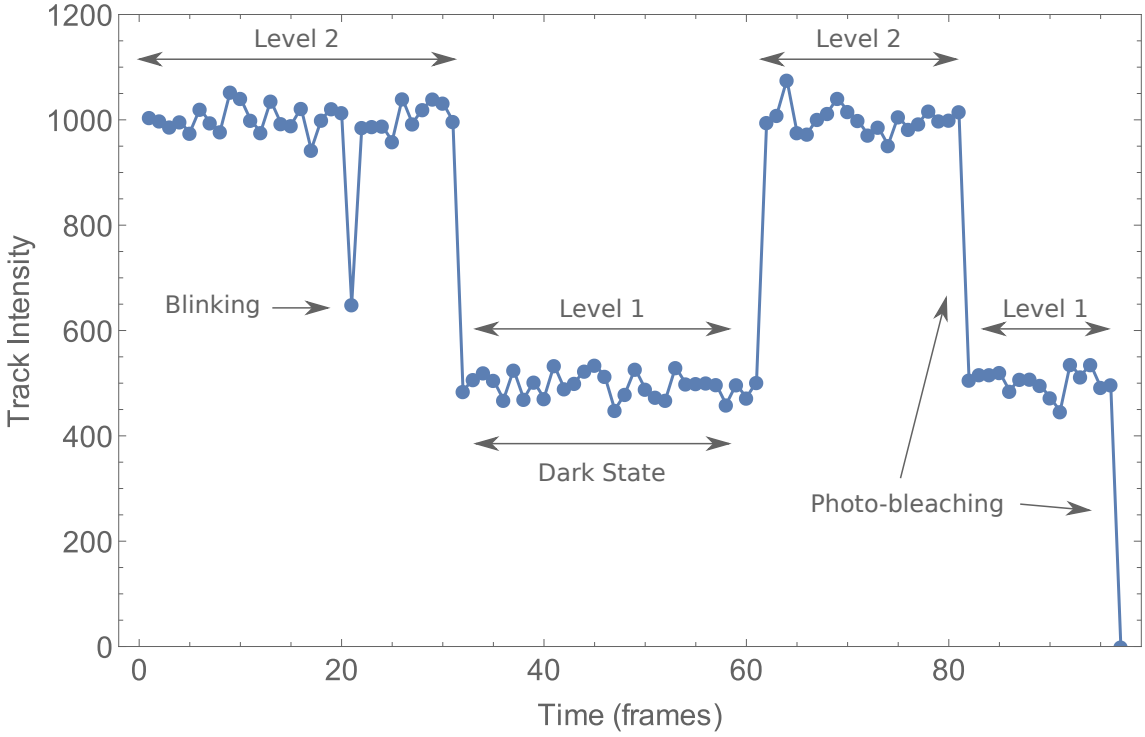


Figure 4.2: An example track with two levels – *level 1* and *level 2*. The track starts with a segment of *level 2*, interrupted by a blinking event, then segment of *level 1* (possibly as result of a fluorophores going into dark state), then another segment of *level 2*, photo-bleaching event, last segment of *level 1* and second photo-bleaching step.

4.2 Level Noise Algorithm

Depending on the depth of the fluorophores and the signal-to-noise ratio, the steps can be larger or smaller than three times the standard deviation of the levels. In the first stage of the level detection, the algorithm finds large steps and splits the tracks into sub-tracks, based on these steps. Then for each sub-track, there are 5 stages in which level segments are identified and modified. The last step combines the level segments of all sub-tracks into levels, based on statistical tests applied to the intensity of the segments. In total, there are seven stages, Figure 4.3:

1. Split into sub-tracks.
2. Identify level segment.
3. Extending segments.
4. Merge adjacent segments.
5. Background Detection. Remove frames, in which there is a non-uniform fluorescence in the background – In this step the background assessment is combined with the level detection.
6. Reject segments which fails statistical tests or have increasing or decreasing intensity.
7. Group level segments into levels.

Stage 5 of the level detection algorithm uses the algorithm described in Section 6.2 to exclude frames, in which there is non-uniform background noise from the level segments.

4.2.1 Identifying Sub-Tracks

The first part of the level detection is a recursive algorithm, which identifies steps and uses them to split the track into sub-tracks.

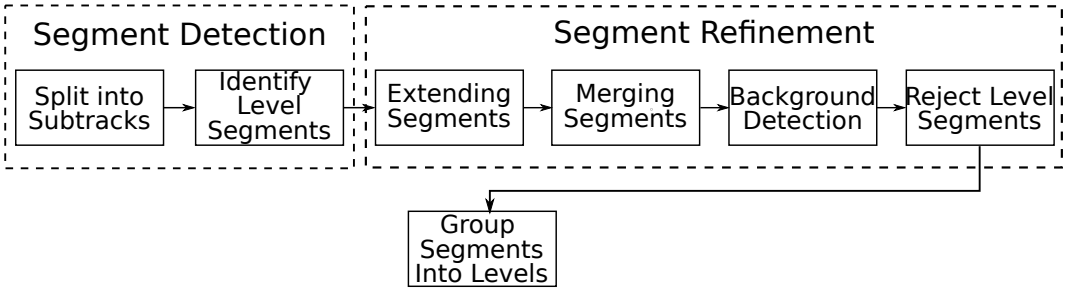


Figure 4.3: The level detection is done in 6 stages.

First in a region of length T are calculated the intensity transitions

$$\Delta I_t = |I_{t+1} - I_t| \;\; t = 1, 2, ..T - 1 \tag{4.1}$$

Then each intensity transition is checked whether it is a step. An intensity transition at time t is a step if

$$|\Delta I_t - \langle \Delta I \rangle_t| > 3\sigma \tag{4.2}$$

where $\langle \Delta I \rangle_t$ is the average of ΔI_t, and σ is the standard deviation. If the inequality holds, then there is more than 99.7% chance that there is a step, and therefore it is assumed that a fluorophore changed its state. Details about the procedure can be found the pseudo-code shown in Pseudocode 4.

Part of finding sub-tracks is converting list of steps to regions. Details of this procedure can be found in Pseudocode 5. This algorithm ignores sub-tracks which are less than 3 frames long.

4.2.2 Identifying Level Segments

Detecting statistically significant steps is a good way to find transitions from a level with two active fluorophores which have equal intensity to a level with one active fluorophore. However, if the fluorophores are at different depth, it may be difficult to detect the level transition this way, as the difference between the two levels could be smaller than what is considered statistically significant step.

input : List of intensities, I, of length T;

Beginning of the range: start;

End of the range: stop

output: List of frame ranges

if stop $-$ start < 5 **then**
  return $\leftarrow \{\}$

end

$\Delta I \leftarrow I_{\text{start}+1:\text{stop}} - I_{\text{start:stop}-1}$

for $t \leftarrow$ *1 to* $\texttt{Length}(\Delta I) - 1$ **do**

  if $|\Delta I_t - \langle \Delta I \rangle_t| > 3\sigma$ **then**
    $\texttt{Append}(\text{steps}, \text{start} + t)$

  end

end

tmpRanges $\leftarrow \texttt{ConstructRanges}(\text{steps}, \text{start}, \text{stop})$

if `Length` *(*tmpRanges*)* $==$ *0* **then**
  return $\leftarrow \{\}$

end

if `Length` *(*tmpRanges*)* $==$ *1* **then**
  return $\leftarrow$ tmpRanges

end

for range *in* tmpRanges **do**

  for newRange *in* $\texttt{FindSubTracks}(I, \text{range.start}, \text{range.stop})$ **do**
    `Append` (ranges, newRange)

  end

end

return ranges

Pseudocode 4: FindSubTracks

```
input  : List of steps;
Beginning of the range, start;
End of the range, stop
output: List of ranges
rangeBegin ← start
for i ← 1 to Length(steps) do
    if steps_i − rangeBegin > 1 then
        Append(ranges, (steps_i, steps_{i+1}))
    end
    rangeBegin ← steps_i + 1
end
if stop − rangeBegin > 1 then
    Append (ranges, (rangeBegin, stop))
end
return ranges
```

Pseudocode 5: ConstructRanges

To find small steps, a different algorithm is applied to each sub-track. This algorithm relies on the assumption that a level has fixed number of active fluorophores, in which case the noise in the level would be the same as the noise in any part of the level.

The algorithm applied to the sub-tracks works by splitting each sub-track into sections (or windows) of 10 frames and calculates the standard deviation of the track intensity in each window, σ_i. Then it starts building a level segment from left to right, by trying to merge each window. For a window to be added to a level segment, it needs to meet several requirements.

- The first requirement is that the average of standard deviation of the candidate window and the previous window should be consistent with the standard deviation of the intensity in the section of the track covered by the two windows.

$$\left|\frac{1}{2}\frac{\sigma_i + \sigma_{i+1}}{\sigma_{i\cup i+1}} - 1\right| < 0.5 \tag{4.3}$$

  where σ_i and σ_{i+1} are the standard deviation of the track intensity of the frames covered by window i and window $i+1$, and $i \cup i+1$ is the track intensities for the frames covered by both windows i and $i+1$.

- The second requirement is that the average standard deviation of windows in the level segment together with the candidate window should be consistent with standard deviation of the track intensity of the frames covered by the current level together with the candidate window.

$$\left|\frac{1}{N}\frac{\sum_{i\in l}\sigma_i}{\sigma_{\cup_{i\in l} i}} - 1\right| < 0.5 \tag{4.4}$$

  where l is the set of window indexes which are in the current level together with the candidate window, and $\cup_{i\in l} i$ is the track intensities of the time frames covered by the windows in the current level segment together with the candidate window.

- The last condition is that the level intensities should come from Gaussian distribution, which is tested using Kolmogorov-Smirnov test with rejection threshold of 0.01.

If any of these three tests fails, the new window will not be added to the level segment, the level segment will be added to the detected segments, and new segment containing the new window will be created.

Details about finding the level segments can be found in Pseudocode 6.

4.2.3 Refining Level Segments

After the level segments are detected, they are modified and some of them are removed. The modifications are applied in the following order:

1. **Segment Extending.** The windows do not always align with the segment boundaries, which means that there might be up to 9 frames which are part of the segment, but are not included. Therefore, nine frames before and after the segment are tested whether they are part of the segment, using the inequality

$$|I_t - m_{seg}| < 2\sigma_{seg} \tag{4.5}$$

   where I_t is the intensity in frame t, m_{seg} is the median of the segment intensities, and σ_{seg} is the standard deviation of the segment intensities. Segments are extended until any frame fails the test.

2. **Background Assessment.** The background assessment algorithm, described in Section 6.2, is applied to the region of interest of each frame in the level segments, and the frames which fail the test are excluded from the segment.

3. **Segment Merging.** An intensity level always contains fixed number of active fluorophores, which means that intensity of each frame should come from the same

input : A list of track intensities I

output: A list of frame ranges

winSize $\leftarrow 10$

T $\leftarrow \texttt{Length}(I)$

winN $\leftarrow \texttt{Floor}(\frac{\mathsf{T}}{\mathsf{winSize}})$

if winN ≤ 1 **then**

  return {(1, Length (I))}

end

for $i \leftarrow 1$ *to* Length(I) **do**

  Append(winStd, StandardDeviation($\mathsf{I}_{(i-1)*\mathsf{winSize}+1}, \mathsf{I}_{i*\mathsf{winSize}}$)

end

segmentOn $\leftarrow False$ segmentStart $\leftarrow 1$

for $i \leftarrow 2$ *to* winN **do**

  segmentCounts $\leftarrow \mathsf{I}_{(\mathsf{segmentStart}-1)*\mathsf{winSize}+1:i*\mathsf{winSize}}$

  lastTwoWinCounts $\leftarrow \mathsf{I}_{(i-2)*\mathsf{winSize}+1:i*\mathsf{winSize}}$

  consistentSegment $\leftarrow \frac{\texttt{Mean}(\mathsf{winStd}_{\mathsf{segmentStart}:1})}{\texttt{StandardDeviation}(\mathsf{segmentCounts})} - 1 < 0.3$

  consistentLastTwo $\leftarrow \frac{\texttt{Mean}(\mathsf{winStd}_{i-1:i})}{\texttt{StandardDeviation}(\mathsf{lastTwoWinCounts})} - 1 < 0.3$

  isGaussian $\leftarrow \texttt{KolmogorovSmirnovTest}\left(\frac{\mathsf{segmentCounts}-Mean(\mathsf{segmentCounts})}{\texttt{StandardDeviation}(\mathsf{segmentCounts})}\right) > 0.01$

  if consistentSegment & consistentLastTwo & isGaussian **then**

    segmentOn $\leftarrow True$

  end

  else if segmentOn **then**

    Append(segments, ((segmentStart $-$ 1) $*$ winSize $+$ 1, $i *$ winSize))

    segmentOn $\leftarrow False$

    segmentStart $\leftarrow i$

  end

  else

    segmentStart $++$

  end

end

if segmentOn **then**

  Append(segments, *((*segmentStart-*1)**winSize *+1*, winN **winSize*))*

end

return segments

Pseudocode 6: FindSegments

statistical distribution. Two or more segments are merged if they are adjacent and come from the same statistical distribution. The segments are tested whether they come from the same distribution by the Kolmogorov-Smirnov test with rejection threshold of 0.05.

4. **Segment Rejection.** The segment will be excluded from a level if it meets any of the conditions:

   (a) **Segment has gradient.** Since levels are defined as constant intensity levels, any tendency of decreasing or increasing intensity with time will disqualify a segment. Therefore, each segment is tested whether it has gradient. This has been done by using linear regression [162] to fit a line to the segment intensities and calculate the standard error of the fit. If the fit is more than 2 standard errors away from zero, this means that there is 95% probability that the segment does not have a constant intensity.

   The first step is to evaluate the parameters of the linear model. If the x axis is assumed to be the time frames, the y axis is assumed to be the track intensity, and in each frame there is also Gaussian noise, $\epsilon \sim N$, then the intensity can be modelled by a line

$$y_t = \alpha x_t + \beta + \epsilon_t \tag{4.6}$$

   The parameters of the model, α and β, are calculated by minimising the error function, ξ,

$$\xi(\alpha, \beta) = \sum_t \left(y_t - (\alpha x_t + \beta)\right)^2 \tag{4.7}$$

   The minimum of the error ξ is at the point where the derivatives with respect to α and β are equal to zero, which is at

$$\alpha = \frac{E[(x - \overline{x})(y - \overline{y})]}{E[(x - \overline{x})^2]} \tag{4.8}$$

$$\beta = \overline{y} - \alpha\overline{x} \tag{4.9}$$

Assuming that the noise in the model is Gaussian, it is possible to evaluate the standard error of the slope α,

$$\sigma_{\langle\alpha\rangle} = \sqrt{\frac{\sum_{t=1}^{T}\left(y_t - (\alpha x_t + \beta)\right)^2}{(T-1)\sum_{t=1}^{T}\left(x - \langle x\rangle\right)^2}} \tag{4.10}$$

where T is the length of the segment, and $\langle x\rangle$ is the mean of the time, x.

After the mean estimate of the slope α and its standard error $\sigma_{\overline{\alpha}}$ are calculated, the probability of the slope to be different from zero can be calculated. If the estimate α is more than two standard errors from zero, this means that there is 95% probability that the segment has increasing or decreasing intensity

$$\frac{\alpha}{\sigma_{\overline{\alpha}}} > 2 \tag{4.11}$$

(b) **Segment has less than 3 frames.**

4.2.4 Combining Level Segments into Levels

After level segments are identified, they need to be combined into levels. Since all segments in a level are generated by the same fluorophores and under the same conditions, they need to come from the same statistical distribution. To assign two or more segments into a level, they are tested for similarity using two sample Kolmogorov-Smirnov test with a rejection threshold of 0.05. Details about the procedure can be found in Pseudocode 7.

4.3 Neural Network

An alternative method of finding steps in the intensity trace of the diffraction limited spot is to train a neural network to find such steps. The neural network will decide for

input : A list of pairs,(start, stop), of frame Ids: unmergedSegments ;

An intensity sequence: I

output: A list of levels

while Length(unmergedSegments) > 0 **do**
    Append(MergedSegments, unmergedSegments[1])
    level $\leftarrow I_{\text{MergedSegments}[-1].\text{start:MergedSegments}[-1].\text{stop}}$
    for $i \leftarrow 2$ *to* $Length$(unmergedSegments) **do**
        currentSegmentCounts $\leftarrow I_{\text{unmergedSegments}[i].\text{start:unmergedSegments}[i].\text{stop}}$
        if KolgomorovSmirnovTest(level, currentSegmentCounts) > 0.05 **then**
            Append(MergedSegments, unmergedSegments[i])
            level $\leftarrow$ Join(level, currentSegmentCounts)
        end
        else
            Append(newUnmergedSegments, unmergedSegments[i])
        end
    end
    unmergedSegments $\leftarrow$ newUnmergedSegments
    Append(levels, MergedSegments)
end
return levels

Pseudocode 7: CreateLevels

each frame whether it is a step based on its neighbourhood. The input of the network is a window of intensities, which spans N frames before and after a frame. Therefore the input has dimensionality of $2 * N + 1$.

This algorithm has 4 stages:

1. **Step Detection**. This section uses neural network to find steps.

2. **Segment Generation**. This section find level segments based on the identified steps. The pseudo-code doing this can be found in Pseudocode 5.

3. **Background Assessment.** The background assessment algorithm, described in Section 6.2, is applied to the region of interest of each frame in the level segments, and the frames which fail the test are excluded from the segment.

4. **Reject Segments**. This stage rejects segments which have increasing or decreasing intensity or don't come from Gaussian distribution (Kolmogorov-Smirnov test with threshold of 0.05). The procedure is described in Section 4.2.3, list item 4.

5. **Combining segments into levels**. This procedure is described in Section 4.2.4

4.3.1 Training Set

Creating training set of existing tracks will be a problem, because it is not clear which intensity changes are steps and which is due to noise. Therefore, simulated tracks can be used for training. Although the simulations may not capture every scenario observed with real tracks, they have the advantage that the ground truth is known which will help to avoid mis-labelling.

The training set is created by labelling simulated tracks. There are 3 types of simulated tracks, in which the fluorophore states are controlled by Markov chain. These are called "Large Intensity Change", "Small Intensity Change", and "Short Levels". Details of how these tracks are generated can be found in Section 3.6.

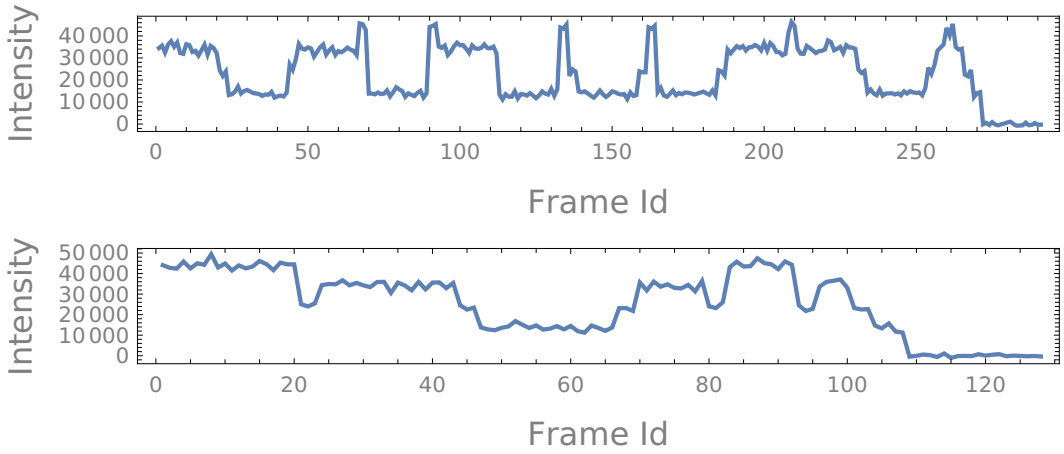


Figure 4.4: The training set contains two tracks which have manually chosen states.

The data set contains 40 sequences of each type, which make 120 simulated tracks in total. These tracks are split into training set which has 84 tracks and testing set which has 36 tracks. Two more tracks are added to the training set. These tracks have states which are set manually so that they can capture some more cases of intensity transition. These tracks are shown in Figure 4.4.

After tracks are split into training and testing set, from each track are extracted the data points. Each data point is intensity of a frame together with the intensities of its neighbourhood of N frames in each direction.

The next step the training and testing set are joined together to calculate the mean and standard deviation for each dimension and these values are used to whiten the data. The whole process is shown in Figure 4.5.

4.3.2 Representation

To find how large the neighbourhood of a frame should be a neural network with different number of hidden units was trained and tested on a labelled data set extracted with 5, 10, and 15 frames in each direction.

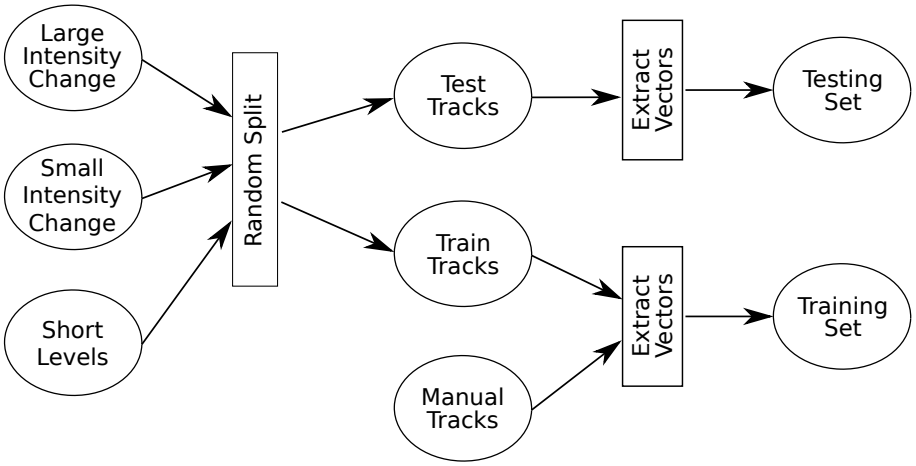


Figure 4.5: Construction of the training and testing set.

Methods

For each combination of hidden units and number of frames in the neighbourhood the data set was split randomly and the neural network was trained and tested, using the function *RandomSample* from Mathematica 11.3. This process was repeated 10 times.

The neural network has one hidden layer. The activation function of the hidden layer is the hyperbolic tangent and the activation function of the output layer is the softmax function. The error function is the cross-entropy.

The network was initialised with the Xavier method. The weights were generated by Gaussian distribution, with variance equal to 2 over the mean of the number of input units, $n^{(l-1)}$ and output units, $n^{(l)}$, for each layer, $\sigma^2 = \frac{2}{n^{(l)}+n^{(l-1)}}$.

The network was trained with the ADAM algorithm for 1000 rounds. The ADAM algorithm used the recommended values for its parameters, $\beta_1 = 0.9$, $\beta_2 = 0.999$. These parameters are described in Section 2.2.2.

The implementation of the initialisatoin and the training algorithm was proided by Mathematica 11.3.

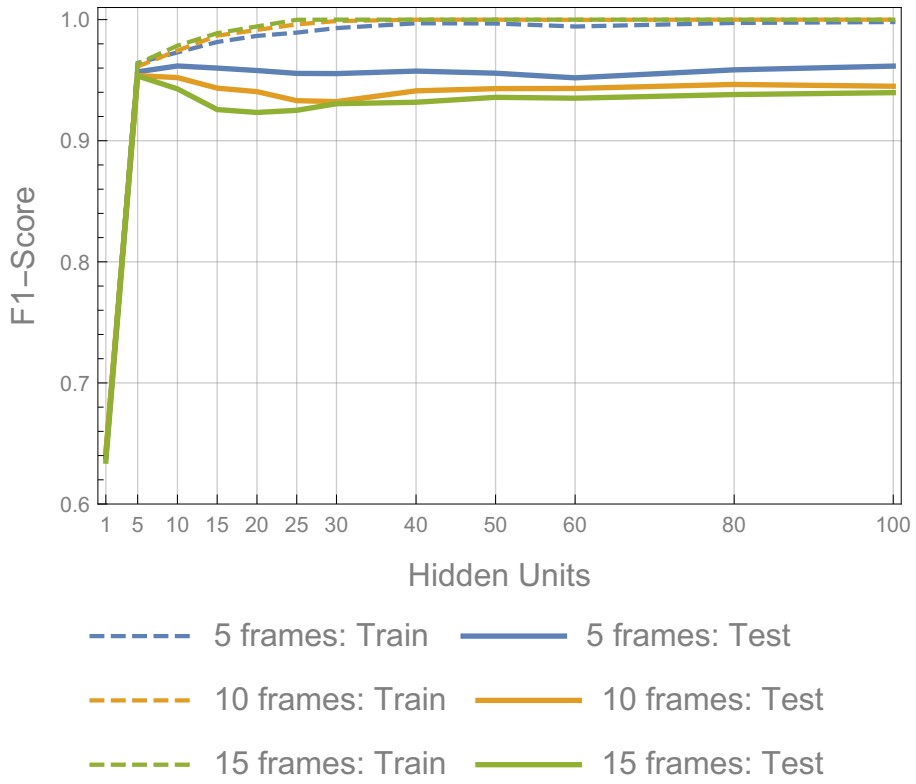


Figure 4.6: Average F1-Score for neural network trained with different number of hidden units and different data sets.

Results

Figure 4.6 shows the average F1-Score for each number of hidden units and number of frames in the neighbourhood. The result shows that the best representation of the data extracted using 5 frames in each direction. Using more frames does not seem to improve the results. Figure 4.7 shows more details of the average F1-Score of neural network trained on a data set extracted with neighbourhood of 5 frames in each direction. The error bars are one standard deviation long. The F1-Score for 1 hidden unit is 0.6234 on the training set and 0.6144 on the testing set and it is not shown on the plot for convenience.

Figure 4.7: Average F1-Score for neural network trained with different number of hidden units on data set extracted with neighbourhood of 5 frames in each direction.

Conclusion

The results show that the best performance is achieved when a neighbourhood of 5 frames in each direction is used and neural network with 11 hidden units. Therefore the data which is used in the next section will extracted by taking 5 frames in each direction around each frame.

4.3.3 Dimensionality Reduction

Dimensionality reduction may help with the classification by removing dimensions which correlate. The results in the previous section showed that the best f1-score was achieved on a data set extracted using 5 frames in each direction of the neighbourhood. Therefore, the original dimensionality of the data is 11. The data will be projected to 6, 4, and 2 dimensions using two methods. The first method, which uses linear projection, is principal component analysis and the second method, which can handle non-linear dependencies, is auto-encoder.

For comparing the algorithms mean squared distance between the original and the recovered data point was used,

$$\mathcal{E} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n)^T (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n) \tag{4.12}$$

where N is the number of data points, $\boldsymbol{x}$ is the original data point, and $\hat{\boldsymbol{x}}$ is the recovered data point. The vectors in this formula are column vectors.

To check if dimensionality reduction is beneficial for the classification a neural network with different number of hidden units was trained and tested on data projected to 2, 4, and 6 dimensions by PCA and auto-encoder. These results are compared to the results on uncompressed data.

Projected Space Dimensionality	Auto-Encoder Architecture
6	11-8-6
4	11-8-6-4
2	11-8-6-4-2

Table 4.1: Number of layers and number of hidden units for each layer of the encoder part of the auto-encoder for each dimensionality.

Methods

The data used in dimensionality reduction was split into training and testing data to check if the auto-encoder over-fits the data. The split was done as described in Section 4.3.1

The number of layers and the number of hidden units in each layer in the encoder part of the auto-encoder are shown in Table 4.1. The auto-encoder is trained in two parts. First each and RBM is created for each layer and trained as described in Section 2.3.2. Each RBM was trained for 1000 rounds with learning rate of 0.01 and momentum of 0.9. The implementation of the training algorithms was custom. The activation function of the hidden and visible units of the RBMs is the hyperbolic tangent. The initial weights and biases were drawn from Gaussian distribution with mean zero and standard deviation 0.3, $N(0, 0.3)$.

Then the parameters of the RBMs are used to create deep network as described in Section 2.3.2 and trained using ADAM with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These parameters are described in Section 2.2.2. The deep networks were trained for 1000 rounds. The implementatoin of the ADAM algorithm was provided by Mathematica 11.3.

Before training with auto-encoder the data was scaled so that the maximum value is 0.95 and the minimum value is -0.95. After compression and decompression each data

point was transformed to the original scale so that the error can be compared to the PCA.

Before training classifier the labelled data is split randomly to training and testing set by the function *RandomSample* from Mathematica 11.3. Then it is projected to a subspace with lower dimensionality. After projection the data was whitened to make the training of the neural network more efficient. Then the training and testing F1-Score is recorded. This process is repeated 10 times.

The network was initialised with the Xavier method. The weights were generated by Gaussian distribution, with variance equal to 2 over the mean of the number of input units, $n^{(l-1)}$ and output units, $n^{(l)}$, for each layer, $\sigma^2 = \frac{2}{n^{(l)}+n^{(l-1)}}$.

The network was trained with the ADAM algorithm for 1000 rounds. The ADAM algorithm used the recommended values for its parameters, $\beta_1 = 0.9$, $\beta_2 = 0.999$. These parameters are described in Section 2.2.2.

Results

Figure 4.8 shows the error for each method of projection and each projected dimensionality. The error of the auto-encoder is slightly better than the error of the PCA, which implies that the auto-encoder may capture the dependencies between the dimensions better than the PCA.

Figure 4.9 shows the average F1-Score measured on the testing set. The results imply that dimensionality reduction is not beneficial for the classification, as the best result remains on the raw data set. The F1-Score for the neural network trained on data projected with auto-encoder to 2 dimensional space is higher than the data projected to the same space by PCA. This result is consistent with the lower reconstruction error found for auto-encoder when the data was projected to 2 dimensional space.

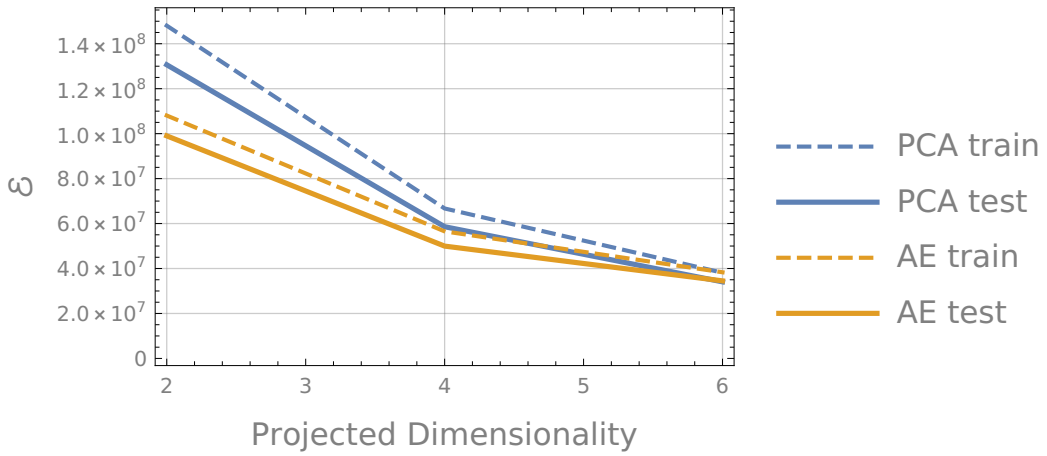


Figure 4.8: Training and testing error for each compression method and each projected dimensionality.

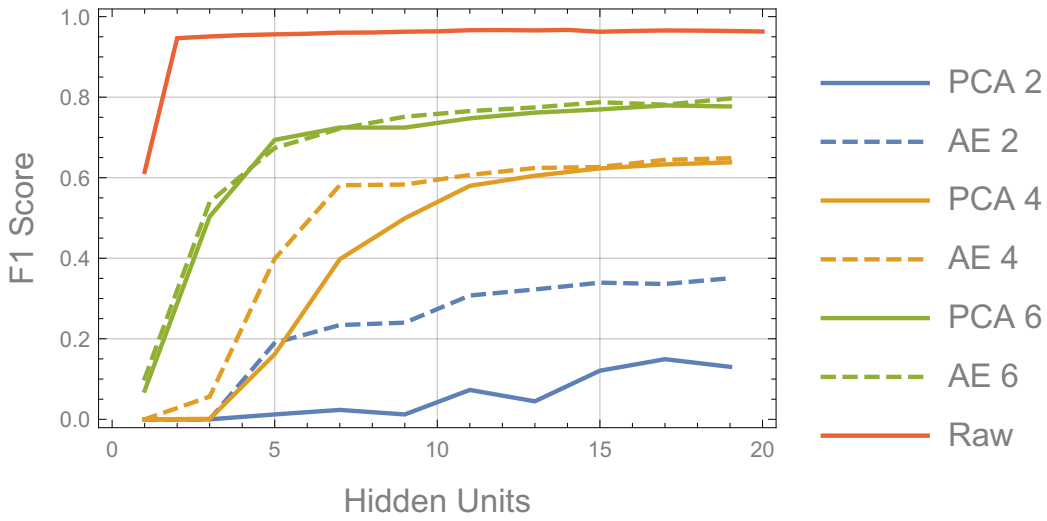


Figure 4.9: Average F1 Score for neural network with different number of hidden units trained on data, projected via Principal component analysis (PCA) and auto-encoder (AE), to 2, 4, and 6 dimensional subspace and the original representation (Raw).

4.3.4 Introducing Class Bias

In this application the false positive errors are less significant than the false negative, because if a frame is erroneously identified as step, the algorithm will create two level segments. Later these level segments are likely to be assigned to the same level, because they have similar intensity. This is not a major problem because FLImP allows levels to be interrupted.

However, if a there is a false negative, the algorithm will assign frames from two separate levels to the same level. This may cause analysable track to be rejected and therefore reduce the efficiency of FLImP or a non-analysable track to be analysed, which will cause invalid results.

Therefore a class bias will be introduced. This is done applying the softmax function to the output from the last layer of the neural network. This causes all the outputs to sum up to 1 and to be between 0 and 1, which allows for probabilistic interpretation of the outputs.

Since the output of the network have probabilistic interpretation, there can be introduced a class bias. The ratio of the probability of positive class, given the sample and the negative class, given the sample can be compared to a threshold. If the threshold is 1 then there is no bias introduced. Since in this application the false positives are less significant than the false negatives a threshold of 0.01 is introduced, which makes the positive class more likely than the negative class. When this threshold is used the recall is 0.9934 and false positive rate, also called fall-out, is 0.0366 on the testing set.

4.3.5 Discussion

The results from dimensionality reduction and training neural network show that the best performance is achieved on data extracted with neighbourhood of 5 frames in each direction. The compression does not benefit the classification accuracy as the highest

F1-Score, 0.9663, was achieved on neural network trained on uncompressed data set with 11 hidden units. Finally the significance of the false positive and false negative errors were taken into account to shift the decision boundary so that the recall can be maximised. This was done by applying the softmax function to the outputs $\boldsymbol{o}$ of the last layer and comparing the ratio $\frac{\text{softmax}(\boldsymbol{o})_1}{\text{softmax}(\boldsymbol{o})_2}$, which can be interpreted as $\frac{p(\text{step}|\boldsymbol{x})}{p(\neg\text{step}|\boldsymbol{x})}$, to a threshold of 0.01. This results in recall of 0.9934 and fall-out of 0.0366.

4.4 Comparing Level Detection Algorithms

The performance of the 3 level detection algorithms "Bayesian", "Level Noise", and "Neural Network" was compared on simulated and real tracks. The set of simulated tracks consists of 3 types of tracks, large intensity step, small intensity step, and short levels. These types of tracks are described in details in Section 4.3.1. The real data was extracted from FLImP experiments and the levels of the tracks were selected manually.

4.4.1 Performance Measures

The performance of the algorithms was evaluated according to 2 measures. The first measure evaluates how close the detected levels are to the original levels. It works by assigning two values to each frame, i. The first value is the mean intensity of the original level which the frame belongs to, $\mu_o(i)$, and the second value is the mean intensity of the detected level, which the frame is belong to, $\mu_d(i)$. Then the error of the level detection is

$$\mathcal{E}^1 = \frac{1}{N}\sum_{i=1}^{N}|\mu_d(i) - \mu_o(i)| \tag{4.13}$$

where N is the length of the track and $|\bullet|$ is the absolute value.

Some frames would not be assigned to a level by the level detection algorithm. This happens because these frames are falsely identified as steps, false positive, and excluded

from levels. These frames are excluded from the error $\mathcal{E}^1$. This means that the error $\mathcal{E}^1$ does not account for false positive.

For example if there is a track with length of 400 frames and the level detection algorithm identified just 5 frames as part of a level and the rest were left unassigned the error $\mathcal{E}^1$ would give very low error, but it is clear that the level detection algorithm failed to find good levels.

Therefore, a second measure is required, which shows how many frames are not assigned to a level but they are part of the original levels, and how many frames are assigned to a level but they are not part of the original levels. If the number of frames which the algorithm assigned to a level but they are not part of the original levels is N_{FP}, and the frames which the algorithms didn't assign to a level but are part of the original levels is N_{FN} then the second measure can be expressed as

$$\mathcal{E}^2 = \frac{N_{FP} + N_{FN}}{N} \tag{4.14}$$

where N is the total number of frames.

4.4.2 Methods

The verification set was assembled from 3 types of tracks, called large intensity step, small intensity step, and short levels. Each type is described in details in Section 4.3.1. For each type of track there are 10 data sets. Each data set contains 9 tracks which are generated by fluorophores in the same state in a frame. This means that the only difference of the tracks within the same data set is the noise. Tracks between different data sets have different state sequences. In total there are 270 tracks in the verification set.

The real data set consists of 100 tracks taken from various experiments. The reference levels of these tracks were selected manually.

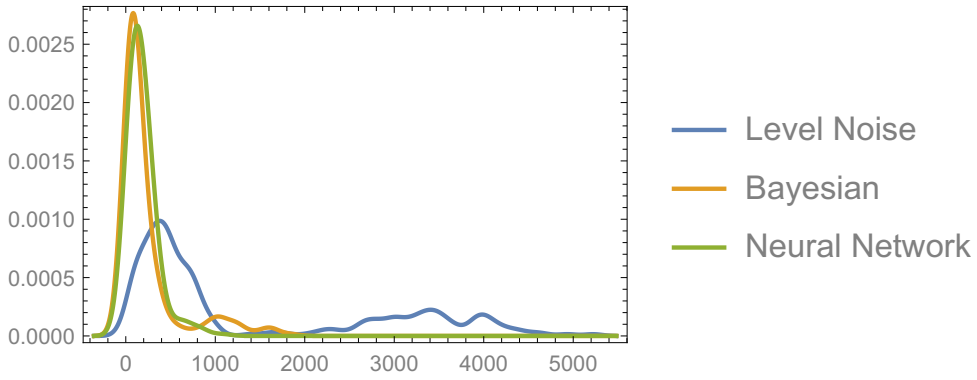


Figure 4.10: Kernel density estimation of the error $\mathcal{E}^1$ for each algorithm evaluated on the simulated data set.

All level detection algorithms are implemented in C++ as part of the processing pipeline of FLImP. Then for each track the levels are detected by each algorithm and then they is imported into a database, where they is read by Mathematica.

The performance measures are implemented in Mathematica. The performance of each algorithm is evaluated for each track, and then kernel density estimation plots are generated using the function *SmoothHistogram* from the Mathematica framework, with bandwidth of 100 for $\mathcal{E}^1$ and 0.02 for $\mathcal{E}^2$ and Gaussian kernel.

4.4.3 Results

Figure 4.10 shows the kernel density estimation of the error $\mathcal{E}^1$ evaluated on the validation set for each algorithm. The error for the algorithm "Level Noise" is much higher than the other two algorithm. The algorithm "Neural Network", with average error of 180.617, performed slightly better than "Bayesian",which has average error of 235.381. When compared the error $\mathcal{E}^2$, shown in Figure 4.11, the algorithm "Bayesian" failed to assign to a level many more frames than "Neural Network". The reason for this is that the algorithm "Bayesian" has too many false positives.

Figure 4.12 shows the kernel density estimation of the error $\mathcal{E}^1$ evaluated on real

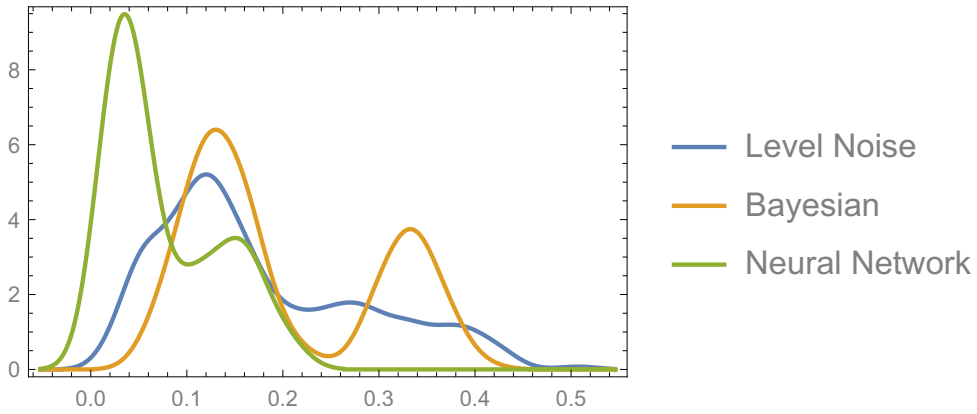


Figure 4.11: Kernel density estimation of the error $\mathcal{E}^2$ for each algorithm evaluated on the simulated data set.

data set for each algorithm. The outliers with values over 2000 are not shown on the plot. The algorithm "Level Noise" has best performance with average error of 244.542, the algorithm "Neural Network" has the second best performance with average error of 389.913, and the "Bayesian" algorithm has the worst performance on the real data set with average error of 555.704.

Figure 4.13 shows the kernel density estimation of the error $\mathcal{E}^2$ evaluated on real data set for each algorithm. The outliers with values over 0.2 are not shown on the plot. The best performance are for algorithm "Bayesian" with average error of 0.026, the second best algorithm is "Neural Network" with average error of 0.096, and the worst performance is of the algorithm "Level Noise" with average error of 0.122.

Figure 4.14, Figure 4.15, and Figure 4.16 show 5 tracks with highest error $\mathcal{E}^1$ for algorithms "Bayesian", "Level Noise", and "Neural Network". The tracks are ordered by the value of the error associated with each track, placing the highest error at the top in each figure. From the figures can be seen that the errors of algorithms "Bayesian" and "Neural Network" are much more significant than the errors of algorithm "Level Noise". In all of the examples algorithm "Bayesian" merged levels which clearly are distinct.

The 3 tracks with highest error processed by "Neural Network", Figure 4.16, also

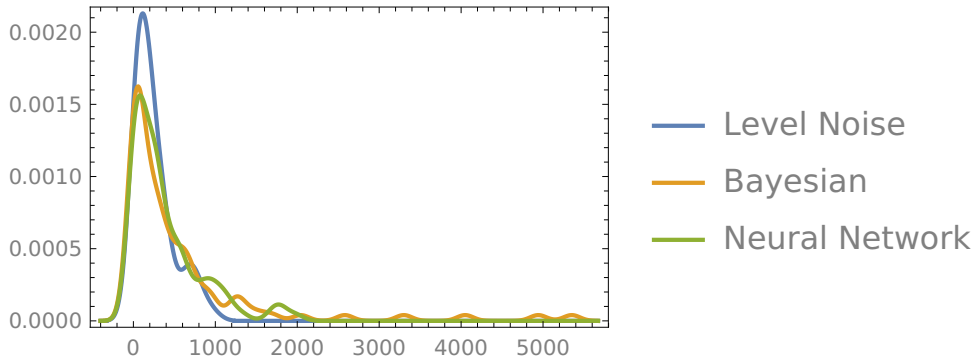


Figure 4.12: Kernel density estimation of the error $\mathcal{E}^1$ for each algorithm evaluated on the real data set.

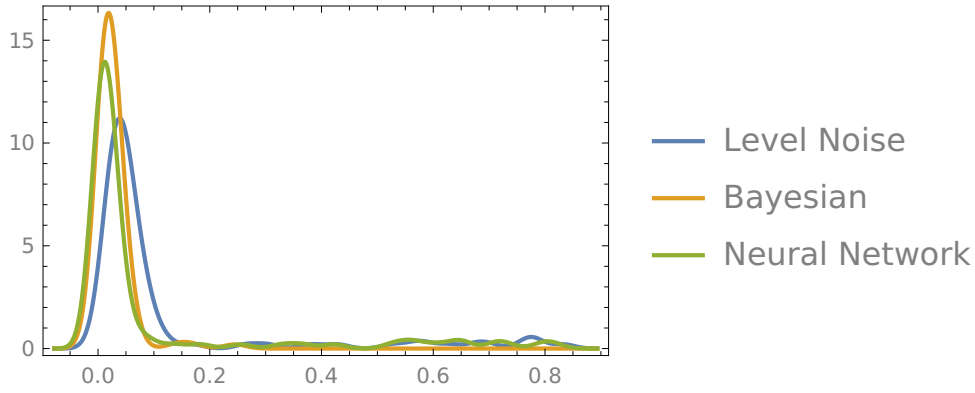


Figure 4.13: Kernel density estimation of the error $\mathcal{E}^2$ for each algorithm evaluated on the real data set.

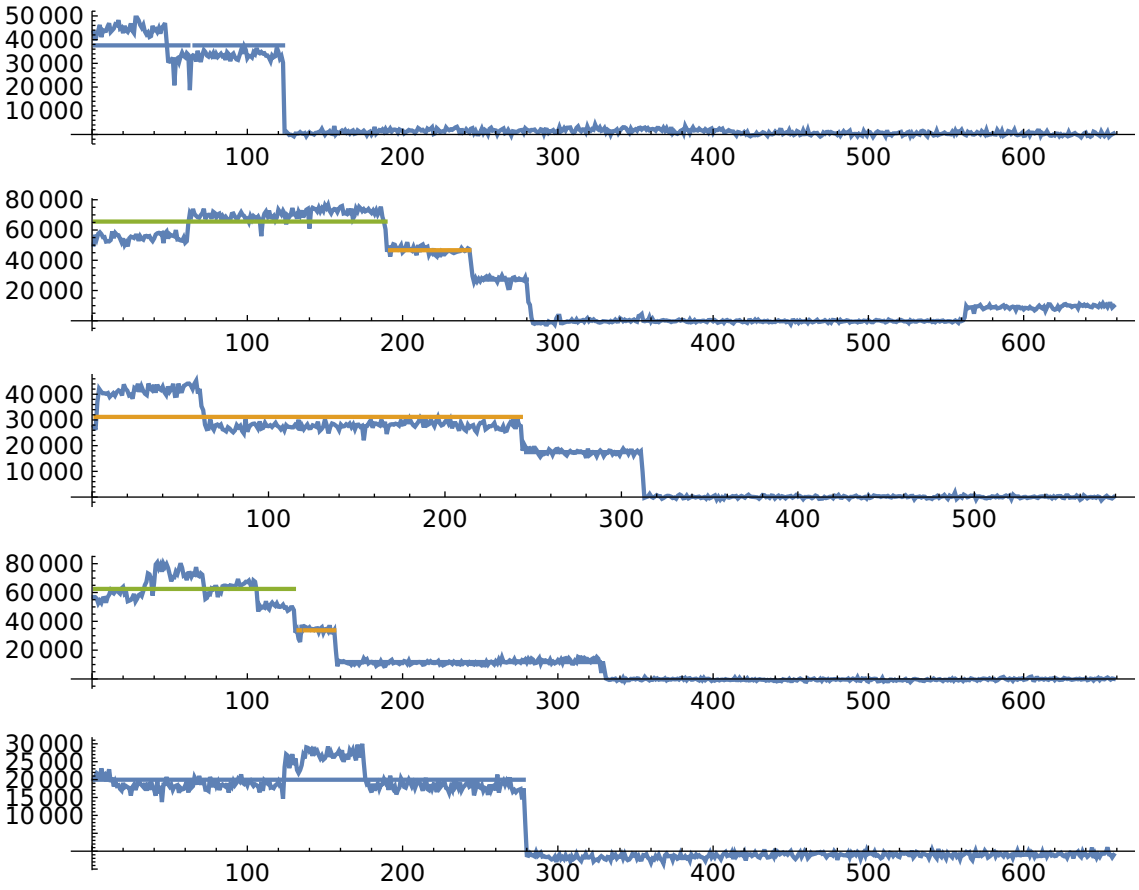


Figure 4.14: Identified levels of 5 tracks with highest error $\mathcal{E}^1$ for algorithm "Bayesian"

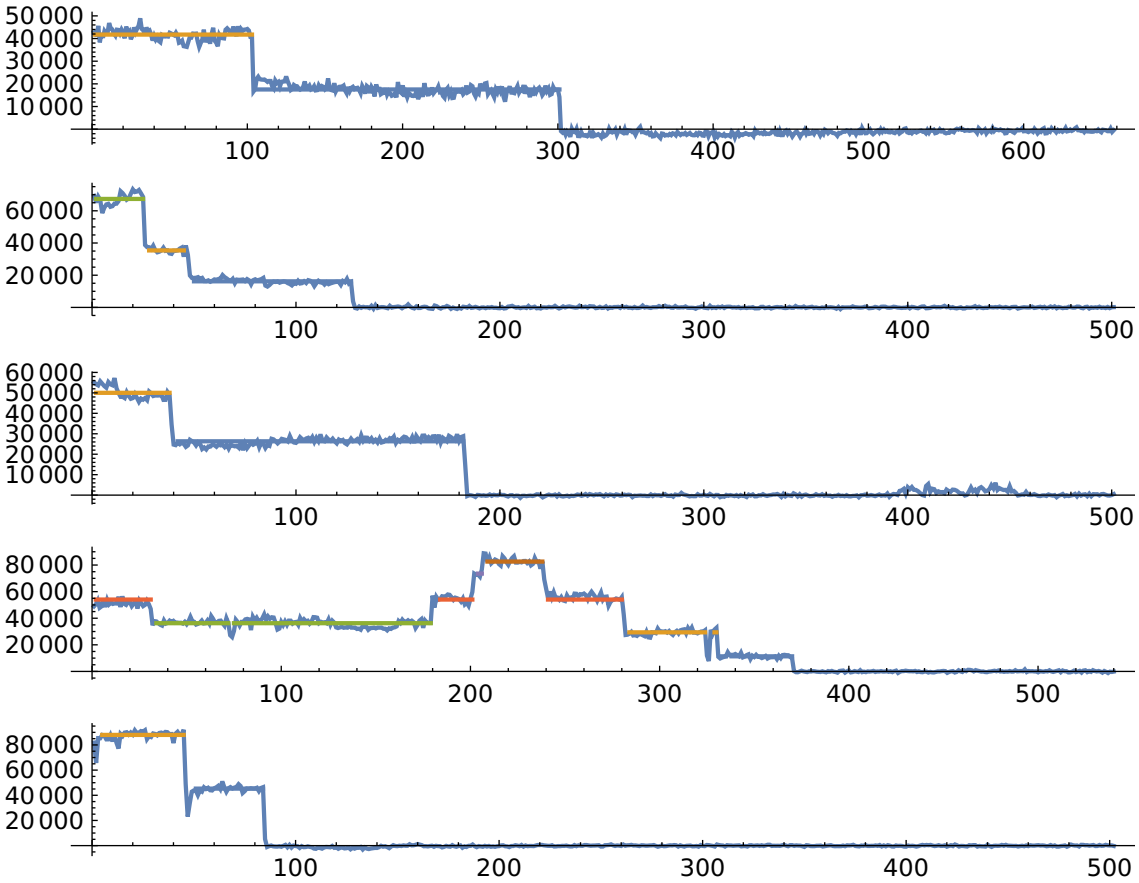


Figure 4.15: Identified levels of 5 tracks with highest error $\mathcal{E}^1$ for algorithm "Level Noise"

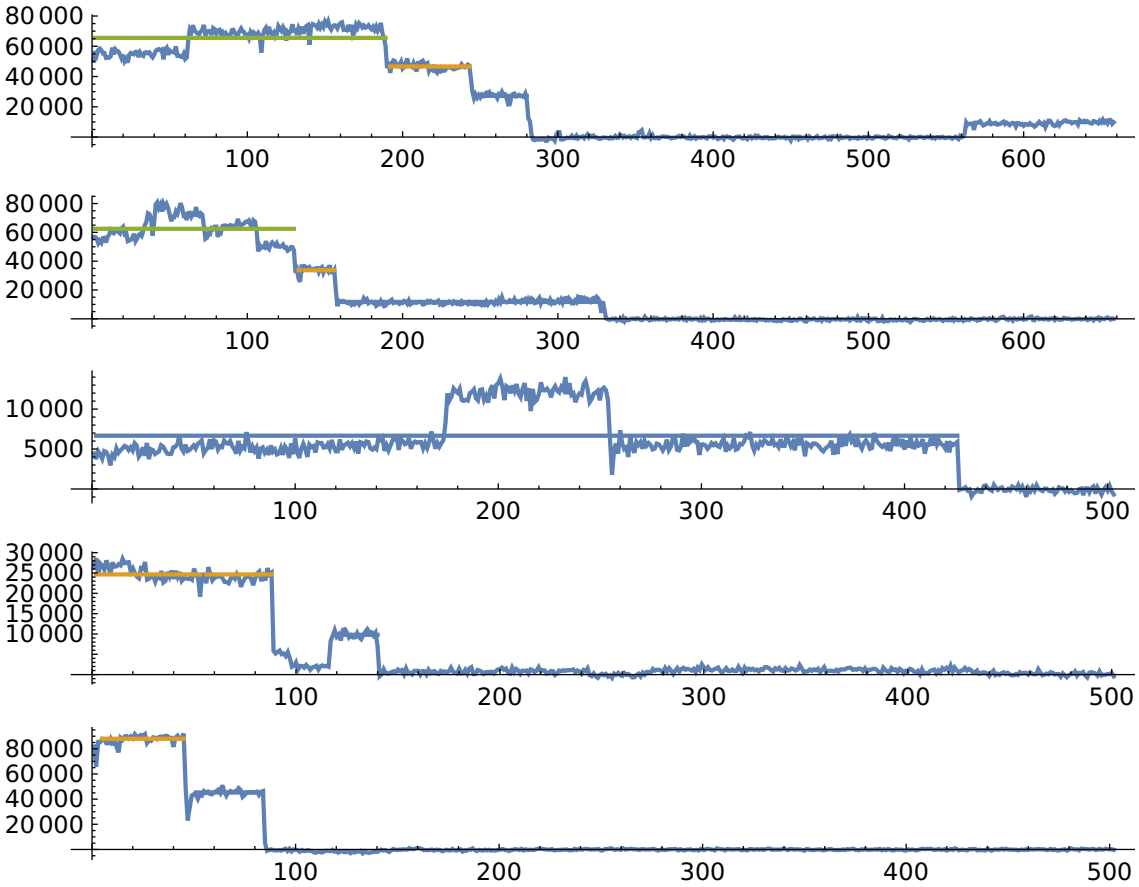


Figure 4.16: Identified levels of 5 tracks with highest error $\mathcal{E}^1$ for algorithm "Neural Network"

have levels grouped together which are clearly very different. In the fourth track the first few frames are likely to belong to different level, however they are also merged together with the tracks from the rest of the level. The last track the levels are correctly identified, however their length is different from the manually selected levels with just few frames.

The 4 tracks with highest error processed by "Level Noise" have only minor errors. The frames which are grouped together have very small intensity difference, and it is not clear whether they are generated by different fluorophore configuration or their intensity is different due to background fluorescence. The last track is the same as the last track of algorithm "Neural Network". Again the levels are correctly identified, however have different length from the manually selected levels.

4.4.4 Discussion

The comparison of the level detection algorithms on the simulated and the real data sets showed that the best algorithm on the simulated data set is "Neural Network" and the best algorithm on the real data set is the "Level Noise". The simulations do not model different aspects of the photophysics of the fluorophores such as FRET or polarisation. Also in the simulations a uniformly distributed background noise is assumed, while the background in the real data often has additional fluorophores or non-uniform background. These inconsistencies between the real data and the simulated data may cause the difference in the performance of the algorithms.

On the other hand the downside of the real data is that the ground truth is not known. However, when inspecting the 5 tracks with highest error, it is clear that the algorithms "Bayesian" and "Neural Network" group levels which do not belong together. The frames which are grouped in the same levels by the algorithm "Level Noise", but are assigned to different levels by the manual level selection, have very close intensities. Therefore it is not clear whether the difference in the intensity comes from a fluorophore

going in an off state or some other process, such as background fluorescence. This ambiguity can be resolved with more realistic simulations, however this is outside of the scope of this thesis.

4.5 Training Neural Network on Real Data

When comparing the level detection algorithms on real and simulated data the algorithm using neural network performed the best on simulated data but not so well on real data. The reason for this behaviour could be differences in the noise properties between the real data and the simulated data. Therefore it might be helpful to train the neural network on the real data and test its performance.

4.5.1 Training Set

The training and testing set is constructed by randomly selecting 100 tracks from those tracks which were analysed in FLImP experiments. Then frames which are just before a transition between levels occurs or during a transition are selected as transition frames (positive class), Figure 4.17

After selection of the transition frames a neighbourhood around each frame is extracted. Based on the results in Section 4.3 a window of 5 frames in each direction is used. This means that each data point is 11 dimensional vector, $\boldsymbol{x} \in \mathbb{R}^{11}$.

The set was randomly split into training and testing set before extraction of the vectors, so that 70 tracks were used for extraction of the training set and 30 tracks were used for extraction of the testing set. After extraction there are 43 376 samples in the training set and 18 645 samples in the testing set.

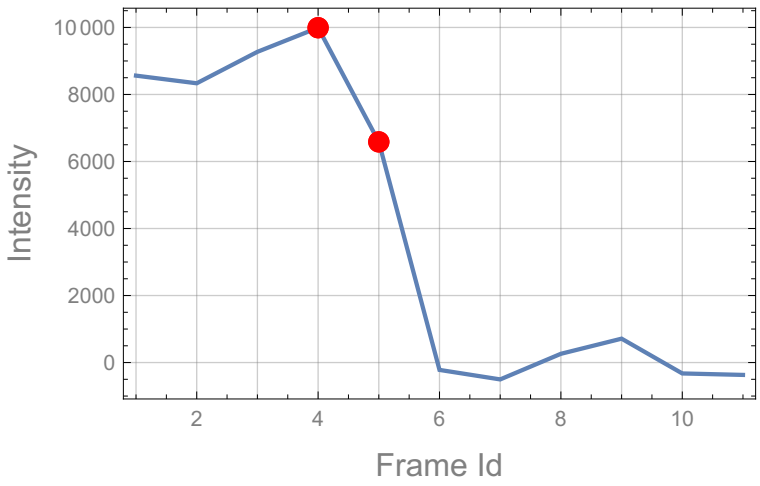


Figure 4.17: Frames which are just before transition or during a transition are selected as transition frames (red dots).

4.5.2 Classification

Since the results of the classification of the simulated data in Section 4.3 showed that compression does not improve the classification accuracy the classifier will be applied on the raw uncompressed data. The data is whitened before classification so that each dimension has mean 0 and standard deviation of 1.

The positive class is much smaller than the negative class. In the training set there are 660 positive samples and 42 716 negative samples. Therefore the performance of the classifier is measured by the f1 score on the positive class.

Methods

For each combination of hidden units the data set was split randomly before the neural network was trained and tested, using the function *RandomSample* from Mathematica 11.3. This process was repeated 10 times.

The neural network has one hidden layer. The activation function of the hidden layer is the hyperbolic tangent and the activation function of the output layer is the softmax

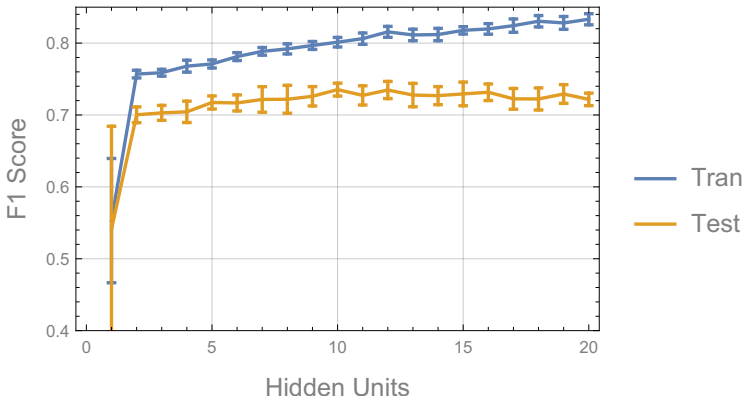


Figure 4.18: Average training and testing f1 score for neural network with different number of hidden units.

function. The error function is the cross-entropy.

The network was initialised with the Xavier method. The weights were generated by Gaussian distribution, with variance equal to 2 over the mean of the number of input units, $n^{(l-1)}$ and output units, $n^{(l)}$, for each layer, $\sigma^2 = \frac{2}{n^{(l)}+n^{(l-1)}}$.

The network was trained with the ADAM algorithm for 1000 rounds. The ADAM algorithm used the recommended values for its parameters, $\beta_1 = 0.9$, $\beta_2 = 0.999$. These parameters are described in Section 2.2.2.

The implementation of the initialisation and the training algorithm was provided by Mathematica 11.3.

Results

Figure 4.18 shows the mean training and testing f1 score over 10 repeats for neural network with different number of hidden units. The error bars are one standard deviation long. The results show that the maximum f1 score on the testing set is 0.7353 and is for neural network with 10 hidden units.

4.5.3 Discussion

The results from the classification of frames into transitional and non-transitional based on their neighbourhood showed that the neural network could not distinguish between the two types of frames.

The difference in the performance of the network on the simulated and the real data could be due to much higher variance in the real data which is not captured by the simulations. Such variance could be larger range in the total intensity of the levels, more diversity in the ratio between levels (larger or smaller steps) and more complex transition patterns. The simulations also do not account for the possibility of one or more transitional frames, which is something that occurs in the real data, and for fluorescence resonance energy transfer (FRET).

The poor f1 score could be caused by deficiencies in the labelling, since ground truth is not known in advance and the labelling is done manually. Other reasons could be inappropriate data representation, and using too short range around the transition frame. In future developments could be explored different data representations which can minimise the in-class variance and maximise the variance between classes.

4.5.4 Conclusion

Based on the results shown in this chapter the algorithm "Level Noise" will be used for the rest of the experiments in this thesis and will be deployed as part of the new track selection process for FLImP.

Chapter 5

Track Classification

This chapter aims to evaluate the reliability of the manual data selection process in FLImP. The analysis process in FLImP requires tracks with specific properties. Selecting tracks for analysis from all available tracks is done manually, however this process hasn't been verified. The work in this chapter will verity the quality of the manual track selection process. Here we show that the manual track selection process is not reliable and prone to errors.

The results from classifying simulated tracks show that the data representation and the classification algorithm can distinguish between the analysable and non-analysable tracks. However, when the classification is applied to FLImP data labelled according to the manual selections, the classifier is unable to distinguish the two classes. The manual selections are inspected and many errors are found.

Since this chapter demonstrates that the manual track selection process is prone to error a new track selection process is designed, which has high degree of automation. The improved track selection process is described in Chapter 6.

This chapter consists of five sections. The first section describes the representation of the tracks used in the classification. In the second section the analysable tracks are inspected to verify whether they meet some of the criteria for analysis. The third section

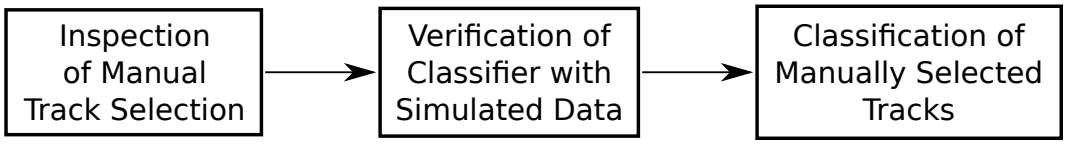


Figure 5.1: Flowchart showing the steps in the analysis done for this chapter.

presents the results from the classification of simulated tracks and in the fourth section presents the results of the classification of the manual track selections, Figure 5.1. The last section is a summary of the results and discussion.

5.1 Data Representation and Extraction

During the track selection process some tracks are rejected automatically before manual inspection. Therefore not all tracks are included in the data set used for training and testing the classifier. Tracks which did not fit the conditions for automatic rejection of tracks, including the LDA filter, were excluded from the analysis, so that manual track selection could be evaluated. These conditions are described in Section 1.6. There are 4 additional conditions added to the automatic track rejection, regarding the levels order and the global drift ranges.

1. Track ends with two levels where the last level has lower intensity than the one before the last or starts with two levels where the first level has lower intensity than the second level. The level with lower intensity will be called level 1 in this chapter and the level with higher intensity will be called level 2. These are not necessary the levels with the lower and second lowest intensity but the last two levels, or the first two levels.

   This condition is designed to be consistent with how people cut tracks during manual track selection.

2. If track begins with the two analysed levels, then it has to start after the beginning of the experiment.

3. If track ends with the two analysed levels, then it has to end before the end of the experiment.

4. Both levels are in the global drift ranges. Since FLImP can only account for a sample motion with specific properties, tracks which are outside the acceptable global drift ranges cannot be analysed.

The level detection algorithm used in the automatic track rejection is "Bayesian", Section 1.6.1, which is the same algorithm used in the track selection process described in Section 1.6. This is done so that the classified tracks are as close as possible to the tracks which were manually inspected during analysis of the data.

Each track is represented as a collection of properties and can be mapped to a point of a feature space. Then these properties, also called feature vectors, are classified by a neural network. These properties are calculated over the last two or the first two levels of the track rather than the entire track. This is important for tracks which have level 1 somewhere in the middle, because for such tracks level 1 will be cut out during the manual track selection and only the last two levels will be used.

The properties are extracted with the two level detection algorithms, "Levels Noise" and "Bayesian", creating a data set for each algorithm. Then each data set is classified.

For the categories to be represented correctly, it is important to capture all the information required for the classification. This can be done by inspecting all the conditions of the track selection process and use one or more properties to capture the information required for testing the condition.

All of the track selection conditions need to be true for a track to be analysed. A detailed explanation of each condition can be found in the introduction, Section 1.6.3.

Some properties are calculated for a moving window over a level or a track. For

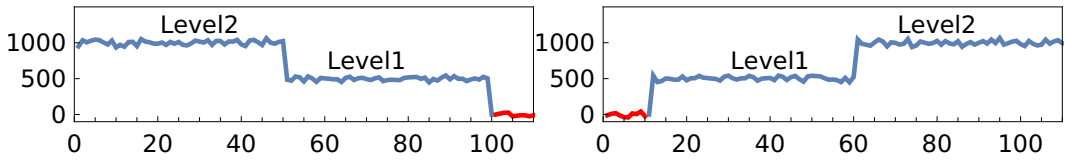


Figure 5.2: Track structures which can be analysed. The red area is extrapolated intensity.

example, if the mean intensity is calculated for each window, it is called local mean intensity; or if the standard deviation of the position is calculated for each window, it is called local standard deviation of position.

For each selection condition, several track properties are listed, which are designed to capture the information required to check if the condition holds. The automatic track rejection ensures that some conditions hold, and hence these conditions are not represented by properties.

1. **Track should have at least 20 frames.** Addressed by the automated rejection of tracks, described in Section 1.6, condition 1.

2. **Track has at least two levels.** Addressed by the automated rejection of tracks, described in Section 1.6, condition 2.

3. **Track ends with level 2 followed by level 1 or starts with level 1 followed by level 2.** These two structures are shown in blue on Figure 5.2.

   Addressed by the automated rejection of tracks, described in Section 5.4.1, condition 1.

4. **If level 1 is first, then track needs to start after the beginning of the experiment.** The right hand side of Figure 5.2 shows that the track starts from frame 11 and before that the intensity is zero.

Addressed by the automated rejection of tracks, described in Section 5.4.1, condition 2.

5. **If level 1 is last, then track needs to end before the end of the experiment.** The left hand side of Figure 5.2 shows that the track ends at frame 100 and after that the intensity is zero.

   Addressed by the automated rejection of tracks, described in Section 5.4.1, condition 3.

6. **Level 1 or level 2 should be inside the frame ranges, in which the global drift is acceptably small** Addressed by the automated rejection of tracks, described in Section 5.4.1, condition 4.

7. **Level 1 or level 2 should have more than 10 non-interpolated frames.**

   - *ratio_real_points_l1* Ratio of detected frames in level 1 and the duration of level 1
   - *ratio_real_points_l2* Ratio of detected frames in level 2 and the duration of level 2

8. **If level 1 is first, then when the track is extrapolated, the intensity before the beginning of the track should to be zero. This is the red intensity shown on the right hand side of Figure 5.2. If level 1 is last, then when the track is extrapolated, the intensity after the end of the track should be zero. This is the red intensity shown on the left hand side of Figure 5.2**

   - *goes_to_zero* If level 1 is first, a region of 10 frames before the beginning of the track is taken. If level 1 is last, a region of 10 frames after the end of the track is taken. Then the distance between the mean intensity of that region and 0 is measured and scaled by the standard deviation of the intensity in that region.

9. **The levels intensity should be constant. If there are increases or decreases over time within a level, the track should be rejected.**

    - *count_sigma_mls_12* The maximum between the ratio of the standard deviation of the intensity and the mean local standard deviation of the intensity in level 1 and level 2.
    - *cts_slm_l1* Standard deviation of the local mean of level 1.
    - *cts_slm_l2* Standard deviation of the local mean of level 2.

10. **Levels with too many interruptions should not be analysed.**

    - *fragmentation_l1* Number of frame ranges in level 1
    - *fragmentation_l2* Number of frame ranges in level 2
    - *longest_range_l1* Length of the longest range in level 1
    - *longest_range_l2* Length of the longest range in level 2

11. **Level 2 should have twice the intensity of level 1.**

    - *stepratio_l12* Measures how far is the ratio between mean intensity of level 2 and mean intensity of level 1 from 2. Usually in the FLImP experiments, the threshold used for this score is 0.2, but it may need to be changed, depending on the experimental conditions.

12. **There may be some small gaps between levels (several frames). However, if the levels are too far away from each other in time, the track should be rejected.**

    - *l12_gap* Length of the gap between level 1 and level 2

13. **There cannot be any intensities between level 1 or level 2 which are not consistent with the intensities of these levels. Such intensity is shown in green on Figure 5.3.**

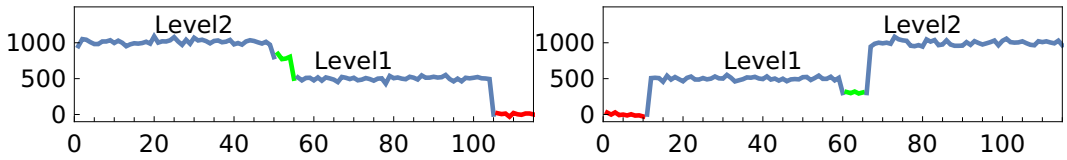


Figure 5.3: There is additional intensity between level 1 and level 2 in green.

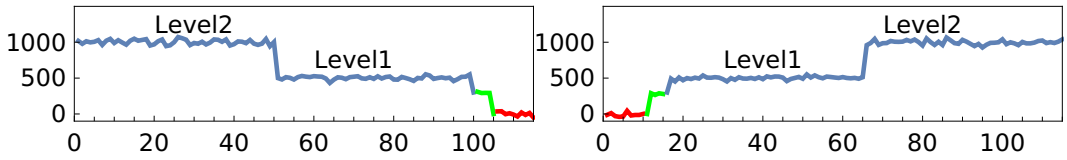


Figure 5.4: There is additional intensity after level 1 (left) and before level 1 (right).

- *l12_off_level* Three properties are generated, based on mean, maximum, and minimum. For each point between level 1 and level 2, the distance between the point and the closer level is measured in terms of standard deviation of the intensity in the corresponding level.

14. **If level 1 is first, there cannot be any intensities before level 1 which are not consistent with the level, the green intensity on the right hand side of Figure 5.4. If level 1 is last, there cannot be any intensities after level 1 which are not consistent with the level, the green intensity on the left hand side of Figure 5.4**

    If level 1 is last, the region between the last frame of level 1 and the last frame of the track is taken. If level 1 is first, the region between the first frame of the track and the first frame of level 1 is taken. In the following two properties, this region is referred to as "the region".

    - *int_before_after_l1* For each frame in the region the distance between the point and level 1 is measured in terms of standard deviation of level 1. Three properties are generated, based on the maximum, minimum, and mean.

- *frames_before_after_l1* The length of the region

15. **There cannot be any fluorescence in the neighbourhood of the diffraction limited spot different from uniformly distributed noise.**
    - *background_l1* Measures the uniformity of the background in level 1.
    - *background_l2* Measures the uniformity of the background in level 2.
16. **The positions of the track during levels 1 and 2 should be constant.**
    - *pos_sigma_mls_12* The larger ratio of the standard deviation and mean local standard deviation of the positions between level 1 and level 2
    - *pos_sigma_locmean_12* The larger standard deviation of local mean position between level 1 and level 2.
17. **The shape of the feature should be a circle.**
    - *feature_shape_l1* Measures the roundness of the feature in level 1.
    - *feature_shape_l2* Measures the roundness of the feature in level 2.
18. **The selected tracks should be likely to result in measurements with confidence interval less than 10nm and separation less than 60nm**. The track which are selected should have long levels with small variance in the position of level 1 and level 2, well separated intensities of level 1 and level 2, and small shift in position during the photobleaching step.
    - *signal_to_noise_l1* Signal to noise in level 1
    - *signal_to_noise_l2* Signal to noise in level 2
    - *sec_longest* Number of frames in the shorter level between level 1 and level 2
    - *sec_most_pts* Number of detected frames in the shorter level between level 1 and level 2

- *cts_loc_sn_l1* Mean local signal to noise ratio of level 1
- *cts_loc_sn_l2* Mean local signal to noise ratio of level 2
- *semdr12* Standard error in mean of difference between position of level 1 and position of level 2
- *pos_density_12* Number of detected features in level 1 and 2 divided by the area which they occupy
- *pos_error_1* Standard error of mean position of level 1
- *pos_error_2* Standard error of mean position of level 2
- *dr12* Distance between mean position of level 1 and level 2
- *clustering* Measures how well the intensity of level 1 and level 2 are separated, based on difference between the mean intensities and the standard deviation.

Detailed description of each property can be found in Appendix A.

5.2 Inspection of Manual Selection

This section will inspect whether the manually selected tracks meet some conditions. The first condition is whether a track has zero intensity when extrapolated before or after level 1, depending on whether a track starts with level 1 or ends with level 1. The second condition is whether level 2 is twice the intensity of level 1.

The scores used in this section are extracted using the level detection algorithm Levels Noise, Section 4.2.

Figure 5.5 shows the histogram of the property *goes_to_zero* measured on the tracks selected for analysis. This property measures the ratio of the mean intensity to the standard deviation of an extrapolated region before or after the track, ρ_3, depending on whether the track starts or ends with level 1,

$$\frac{\langle I(\rho_3)\rangle}{\sigma_{I(\rho_3)}} \tag{5.1}$$

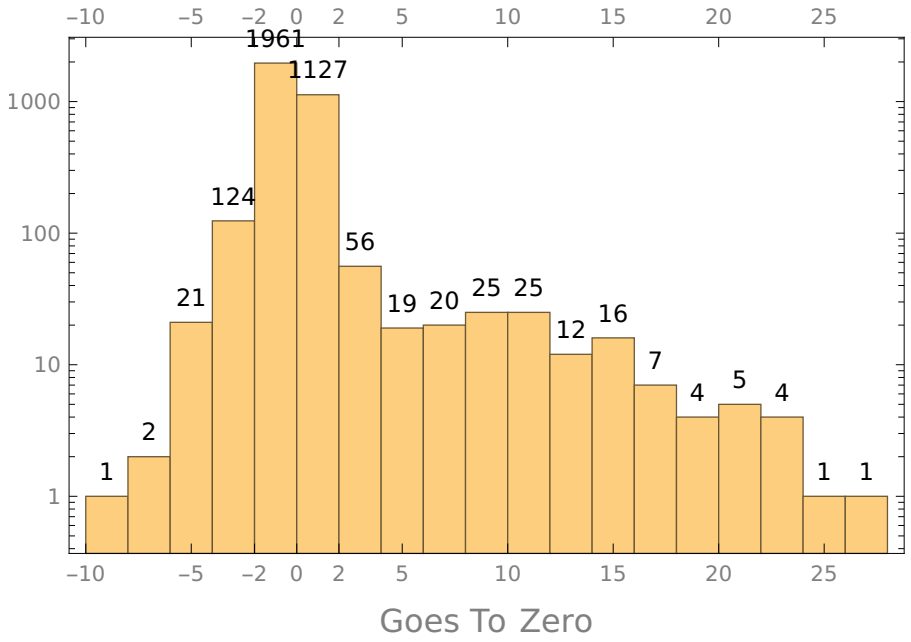


Figure 5.5: Histogram of the property *goes_to_zero* of all analysed tracks from the data set extracted from the manual selections.

There are 343 values, corresponding to 9.9% of the positive class, larger than 2 or smaller than -2, which means that there is 5% or lower chance that these tracks have intensity of zero before or after the track.

Figure 5.6 shows the plot of the intensity of a track, which was selected for FLImP analysis in 2014, for which the value of the property *goes_to_zero* is 6.77. The track starts from level 1, followed by level 2 and a photobleaching step. It can be seen that there are several frames before level 1 and after level 2 with intensity different from level 1 and level 2. When the track is extrapolated, shown with the red frames, the intensity of the extrapolated frames is similar to the intensity of the frames before level 1 and after level 2. This is an indication that there could be additional fluorophore in the feature during the duration of level 1 and level 2.

The bottom part of Figure 5.6 shows the neighbourhood of the position of the first detected feature in frames from 150 to 153, and it can be seen that there is an additional

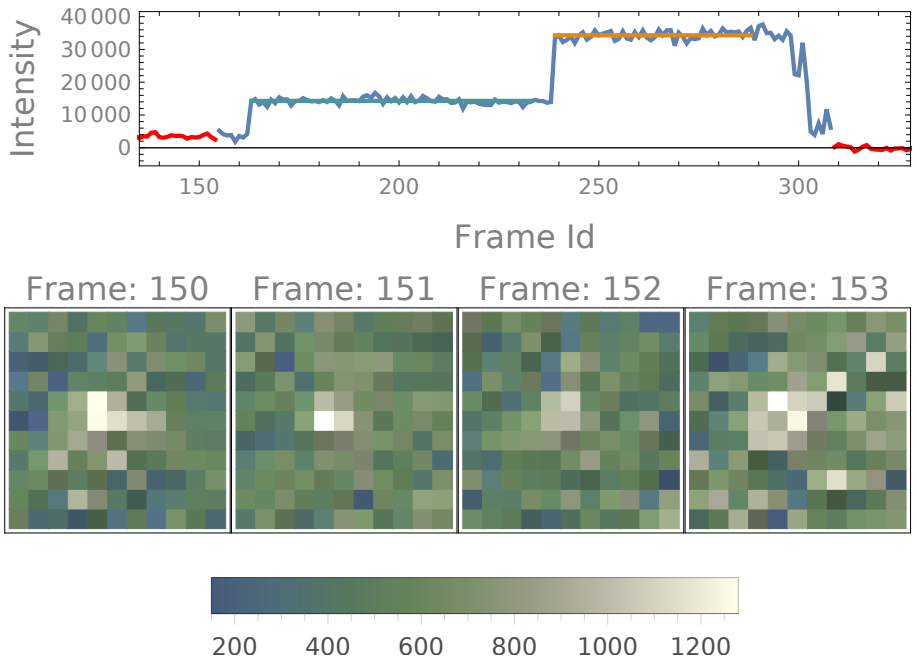


Figure 5.6: The top shows the intensity of a track selected for analysis, which has non-zero intensity before the beginning of the track. The extrapolated regions are shown in red, and the levels are show with blue and orange. The bottom shows the image around the position of the first detected feature for the frames from 150 to 153.

fluorophore. This implies that in level 1 there are minimum 2 fluorophores, and in level 2 there are minimum 3 fluorophores. Therefore, this track should not be analysed.

Another example of a track which should not be analysed, but was selected for analysis in 2014, is shown on the top of Figure 5.7. The values of *goes_to_zero* for that track is -3.41, which means that the intensity after the end of the track is lower than zero. Since the feature detection algorithm, Quincy, calculates the feature intensity by subtracting the background intensity, it is possible that there is a background fluorescence in the neighbourhood of the feature, which causes Quincy to overestimate the background intensity and hence produce negative values. The bottom part of Figure 5.7 shows the image of the neighbourhood the feature for frames from 170 to 173. It can

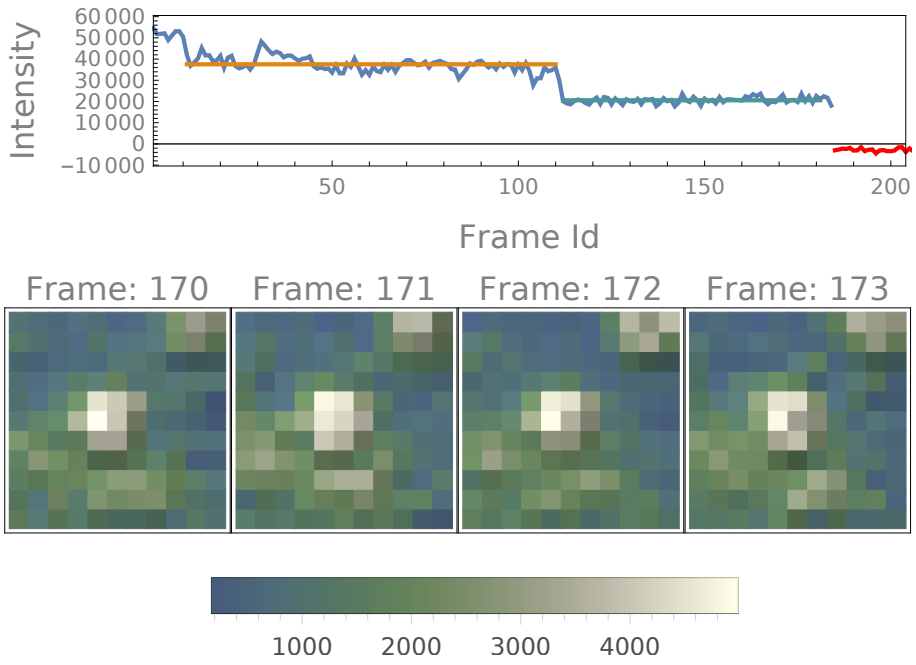


Figure 5.7: The top shows the intensity of a track selected for analysis, which has non-zero intensity after the end of the track. The extrapolated regions are shown in red, and the levels are show with blue and orange. The bottom shows the image of the neighbourhood around the detected feature from frames from 170 to 173.

be seen that there is a lot of background fluorescence different from zero. Therefore, such track shouldn't be analysed.

The other condition which can be checked before classification is whether the mean intensity of level 2 is twice the mean intensity of level 1. This can be done by examining the property *stepratio_lvl12*, which shows how much the mean intensity of level 2 differs from twice the mean intensity of level 1.

$$\left| \frac{\langle I(l2) \rangle}{\langle I(l1) \rangle} - 2 \right| \tag{5.2}$$

Figure 5.9 shows a histogram of that property for all the analysed tracks from the data set extracted from the manual track selections during the year 2014 and 2015.

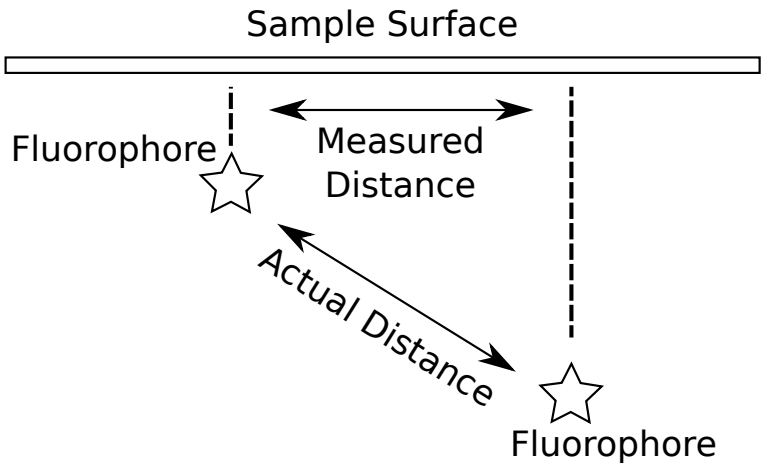


Figure 5.8: When fluorophores are at different depth FLImP will measure the projection to the surface of the sample rather than the actual distance.

The property *stepratio_lvl12* measures how similar are the intensities of the fluorophores in level 2. If these fluorophores have different intensity, this means that they are at different depth, and FLImP will measure the projection to the surface of the sample, rather than the distance between the fluorophores, Figure 5.8.

In FLImP analysis, the threshold used for this ratio is either 0.2 or 0.3, but there are 2301 tracks which have value larger than 0.2, corresponding to 67% of the positive class, and 1959 tracks which have value larger than 0.3, corresponding to 57% of the positive class.

There are two types of errors related to the property *stepratio_lvl12*. The first type of error is selecting tracks where level 2 has mean intensity different from twice the mean intensity of level 1. The top part of Figure 5.10 shows an example of such track.

The second type of error is in the level detection algorithm. The bottom part of Figure 5.10 shows such an example. The 2014 version of the level detection algorithm erroneously showed level 1 and level 2 as the same level and level 3 as level 2. Then during manual inspection, this error was not corrected. However, the improved version of the level detection algorithm found out that there was very high probability that the

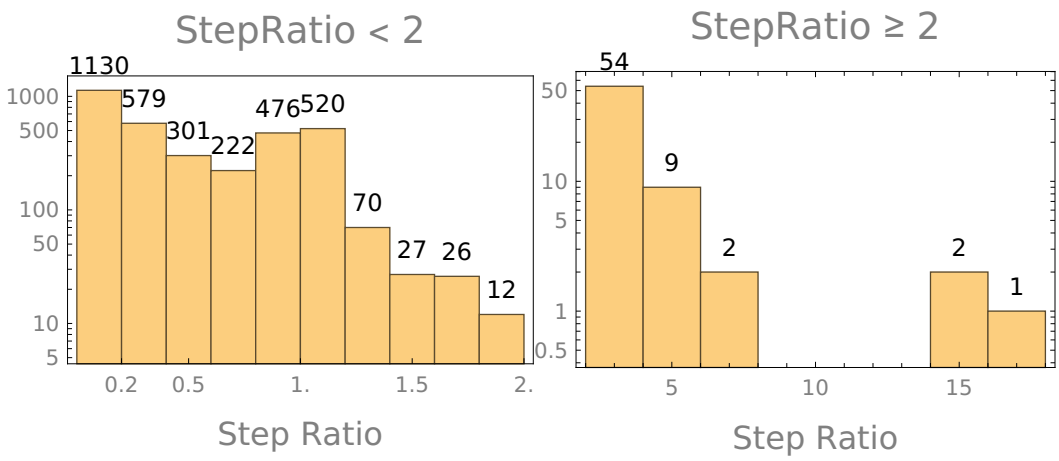


Figure 5.9: Histogram of the property *stepratio_lvl12* of all analysed tracks from the data set extracted from manual selections during 2014 and 2015.

intensity in level 1 and level 2 came from different statistical distributions and therefore identified them as different levels.

5.2.1 Conclusion

The inspection of the analysable tracks shows that there are some serious deficiencies in the manual track selection process. Provided that there are so many false positives (67% from the analysed tracks according to the measure *stepratio_lvl12*, and 9.9% from the analysed tracks according to measure *goes_to_zero*) it is likely that the classification of the manually selected data is not going to achieve very good performance.

5.3 Classification of Simulated Data

This section will verify that the representation and the classifier are appropriate for classification of tracks into analysable and non-analysable by classifying simulated tracks. The advantage of the simulated tracks is that the ground truth is known and there is no mislabelling.

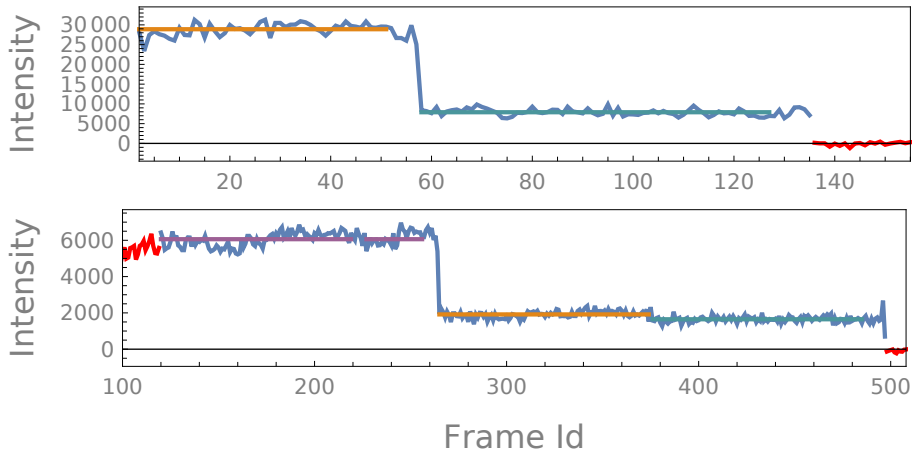


Figure 5.10: Examples of mistakes made during the manual selection process. The top track has mean intensity of level 2 too far from twice the mean intensity of level 1. On the bottom, level 1 and level 2 were selected as one level 1, and level 3 was selected as level 2. However, the improved level detection algorithm identified level 1 and level 2 as distinct levels.

5.3.1 Data Extraction and Labelling

The data set extracted with the refined level detection algorithm, called "Level Noise" and described in Section 4.2, has 316 analysable and 23148 non-analysable tracks. The data set extracted with the level detection algorithm called "Bayesian" has 316 analysable and 23043 non-analysable tracks. Both data set have ratio of analysable to non-analysable tracks very close to the ratio of the data set made of manually selected track, which is around 0.013.

The non-analysable tracks are generated by 9 cases, where each case violates different rule for manual track selection. More details about how the data is generated can be found in Section 3.7.

Both the analysable and non-analysable tracks pass the automated track rejection criteria, described in Section 5.1, apart from the condition for the Linear Discriminant

Analysis, to make sure that the simulated data set has the same properties as the real data. The global drift of the sample is assumed to be compatible with the FLImP requirements everywhere.

5.3.2 Classification

Each track was represented as a point in a vector space, $x \in \mathbb{R}^{37}$, using the representation described in Section 5.1 and the data is classified by neural network with different number of hidden units.

The classifier is applied on the raw data set and on a compressed data set (using PCA). Before classification the data is transformed so that it has mean 0 and standard deviation of 1 in each dimension.

Methods

The network is trained and tested 10 times and before each time the data set is randomly split into training and testing set, where the training set has 70% of the data. The random split was done using the function *random.shuffle* from *numpy 1.17.4*.

The neural network has one input layer, one hidden layer, and one output layer. The activation function is hyperbolic tangent and the error function is the cross-entropy. The labels follow one-of-N method.

The network is initialised with the Xavier method where the random weights are drawn from Gaussian distribution. It is trained with the ADAM algorithm, with parameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The networks was trained for 1000 rounds each time. For both initialisation and training was used the implementation of *tensorflow 2.0.0*.

Results

The PCA analysis shows that not all of the components are required to capture the variance of the simulated data. Figure 5.11 shows the percentage of variance captured by

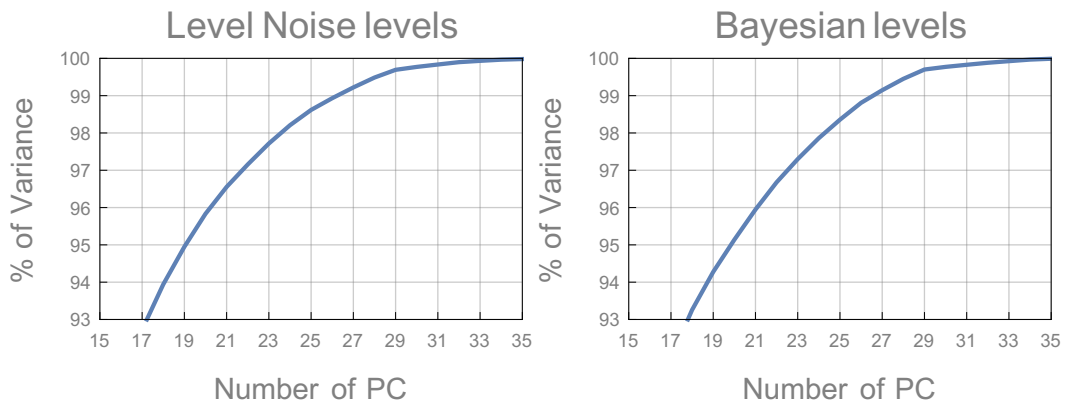


Figure 5.11: The percentage of variance captured by each number of principal components of simulated tracks

different number of principal components. The result shows that 20 principal components are sufficient to capture at least 95% of the variance of the data for the data sets extracted by both level detection algorithms.

The left hand side of Figure 5.12 shows the average f1 score of track classification by neural network with different number of hidden units, measured on the training and testing set extracted by both level detection algorithms. The error bars are one standard deviation long. The best testing f1 score for the data set extracted with the algorithm Levels Noise is 0.9597 with error bar of 0.009 and it is achieved by a neural network with 1 hidden unit. The best testing f1 score for data set extracted with the algorithm Bayesian is achieved by a classifier with 2 hidden units and is 0.8815 with error bar of 0.0372.

The right hand side of Figure 5.12 shows the average f1 score of track classification by neural network with different number of hidden units measured on compressed training and testing set, extracted by both level detection algorithms. The best testing f1 score for the data extracted with the algorithm Levels Noise is for neural network with 2 hidden units and is 0.9387 with error bar of 0.0211. The best testing f1 score for the data extracted with the algorithm Bayesian is for classifier with 3 hidden units and is

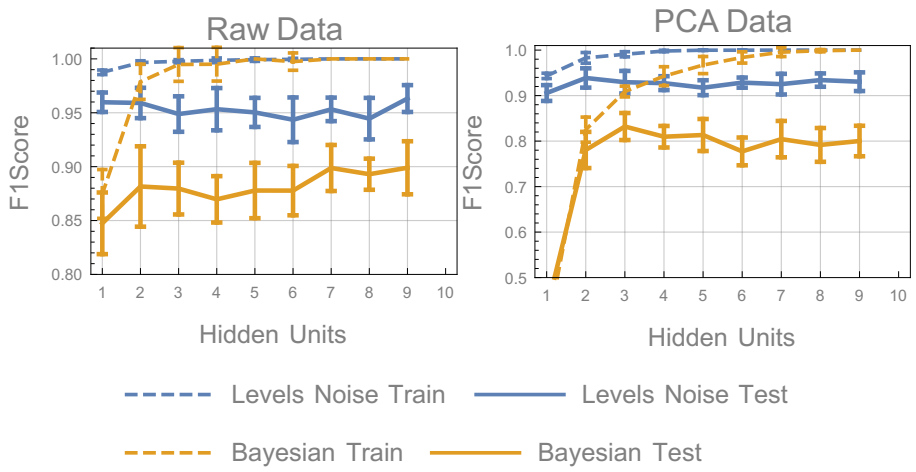


Figure 5.12: Training and testing f1-score for neural network with hidden units from 1 to 9.

0.832 with error bar of 0.0296.

Conclusion

The results presented in this section show that the features extracted with the improved level detection algorithm, Levels Noise, are significantly better and allow for higher f1 score than the features extracted using the algorithm Bayesian.

The high f1 score for the linear classifier applied on the raw data set, extracted by the algorithm Levels Noise, showed that the representation captures all the features of the analysable and non-analysable tracks and the classifier can easily distinguish between the two classes when they are labelled properly. These results rule out the case that low f1-score is caused by problems with the representation or the classifier.

Also the fact that the class imbalance was the same as the class imbalance in the data set made by manually selected tracks rules out that low f1 score can be caused by the class imbalance.

The similarity between the result on the raw data set and the data set projected on

the first 20 principal components shows that some of the scores correlate and it may not be necessary to be included in the feature space of the simulated tracks.

5.4 Classification of Manual Track Selection

In this section the quality of the manual track selection process will be tested using classification. The first subsection shows details of the extraction and labelling of the data set, and the second section presents the classification results.

5.4.1 Data Extraction and Labelling

The data set used to train the classifiers in this chapter is extracted from the data generated by 2064 FLImP experiments. The results of some of these experiments are published as part of a paper [22]. In all of experiments, cells from Chinese hamster ovary (CHO) are used and the receptors studied are HER1, and HER2. Details the preparations of the experiments can be found in Appendix B.

Before labelling some tracks were rejected by the automated rejection criteria described in Section 5.1 and the tracks which passed the rejection were labelled and classified. Tracks which were selected for analysis were labelled as positive, and the tracks which were not selected were labelled as negative.

Then the tracks were converted to feature vectors using the representation in Section 5.1. The data was extracted with two level detection algorithms, "Levels Noise" and "Bayesian", creating two datasets. The scores couldn't be extracted from some tracks, and therefore these tracks were also rejected. The data set extracted with the algorithm Level Noise had 3431 tracks in the analysable class and 247 196 tracks in the non-analysable class. The data set extracted with the algorithm Bayesian had 3379 tracks in the analysable class and 231 595 tracks in the non-analysable class.

5.4.2 Classification

Each track was represented as a point in a vector space, $x \in \mathbb{R}^{37}$, and the data was classified by feed-forward neural networks with different number of hidden units.

The classifier was applied on the raw data set and on a compressed data set (using PCA). Before each classification the data set was transformed, so that each dimension had mean 0 and standard deviation 1.

Methods

The network is trained and tested 10 times and before each time the data set is randomly split into training and testing set, where the training set has 70% of the data. The random split was done using the function *random.shuffle* from *numpy 1.17.4*

The neural network has one input layer, one hidden layer, and one output layer. The activation function is hyperbolic tangent and the error function is the cross-entropy. The labels follow one-of-N method.

The network is initialised with the Xavier method where the random weights are drawn from Gaussian distribution. It is trained with the ADAM algorithm, with parameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The networks was trained for 1000 rounds each time. For both initialisation and training was used the implementation of *tensorflow 2.0.0*.

Results

Figure 5.13 shows how much of the variance of the data is captured by each number of principal components for each data set. A common practice is to use the number of the principal components which capture at least 95% of the variance, which for the both data sets is 26 principal components.

Figure 5.14 shows the average f1 score for neural network with hidden units ranging from 1 to 9 trained on a data set extracted from manual selections. The features of the tracks were calculated using both level detection algorithms, "Level Noise" and

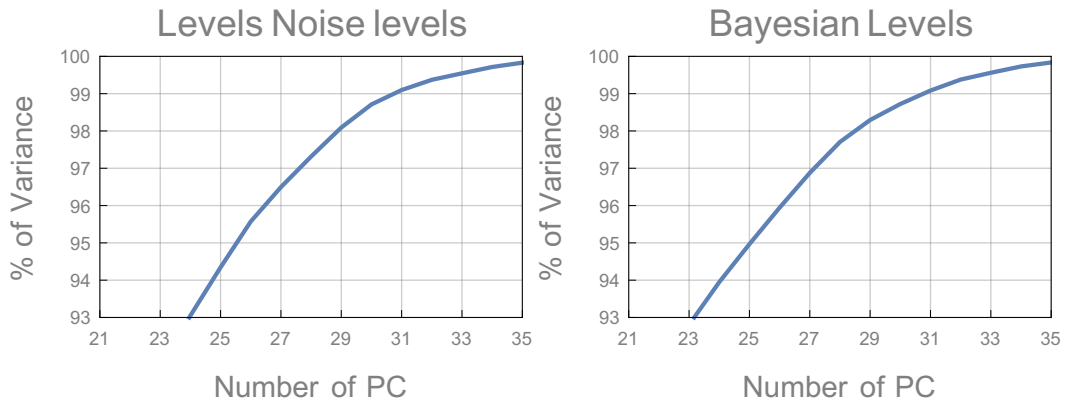


Figure 5.13: The percentage of variance captured by each number of principal components of tracks extracted from 2064 FLImP data sets

"Bayesian". The f1 score was calculated on both training and testing set. The classification was applied on the raw data set and also on the PCA compressed data.

The figure shows that the classification applied on a data set extracted with the algorithm "Bayesian" has slightly higher f1 score than the classification applied on the data set extracted with the algorithm "Levels Noise". All the testing f1 scores stayed below 0.3.

Figure 5.15 shows the average training and testing f1 score for a neural network with hidden units ranging from 10 to 30 with increment of 5. The network was trained on the raw and PCA compressed data set extracted with both level detection algorithms, "Levels Noise" and "Bayesian". The graphs show that the training f1 score increases as the number of hidden units increases however the testing f1 score does not increase for networks with more than 20 hidden units. The best testing f1 score remains around 0.31 which is by a neural network with 20 hidden units trained on the uncompressed data set extracted with the algorithm "Bayesian". However there is very large difference between the training and testing f1 score for this neural network, which means that it is likely that there is over-fitting.

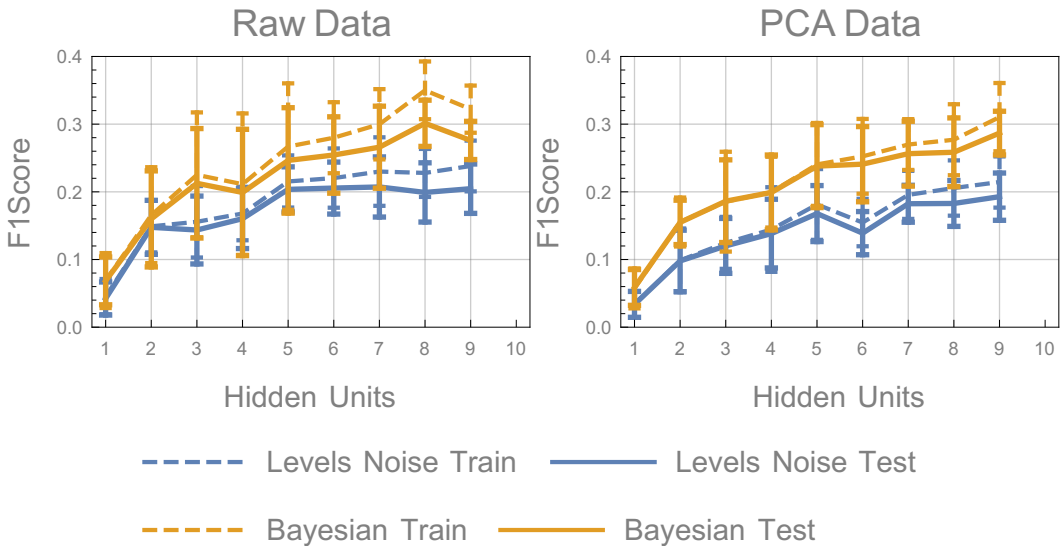


Figure 5.14: Training and testing f1-score for neural network with hidden units ranging from 1 to 9.

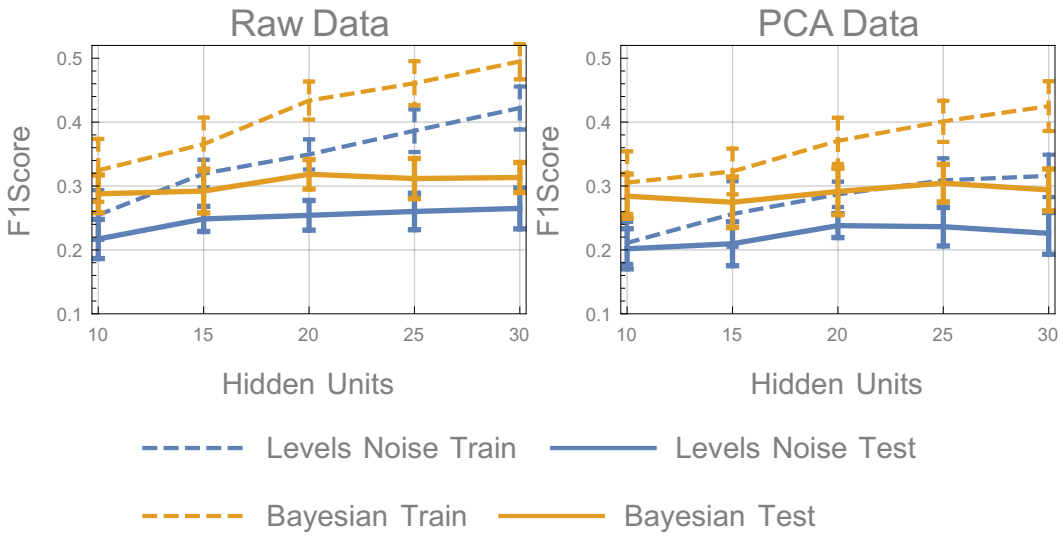


Figure 5.15: Training and testing f1 score for neural network with hidden units ranging from 10 to 30.

Conclusion

The results in this section showed that the classifier was unable to distinguish between the analysable and non-analysable tracks. The best classification which is not considered over-fitting was achieved when a neural network with 8 hidden units was trained on the uncompressed data set extracted with the algorithm "Bayesian". The testing f1-score of that classifier was 0.3012.

5.5 Discussion

The classification of the manually selected tracks produced very low f1-score. It is possible for the class imbalance to cause the classifier to be unable to learn the differences between classes. However, the high f1-score on the simulated data set, which has the same class imbalance, showed that this is not the case.

Another possibility is that the representation did not capture all the required information for the classification. The results from the simulation show that the classifier can distinguish between the analysable and non-analysable class when they are labelled properly. Therefore the representation is not a reason for the low f1 score in the manually selected tracks.

These results combined with the inspection of the analysable tracks in Section 5.2 shows that there are errors in the manual track selection process.

Errors in the track selection process can cause FLImP to underestimate or overestimate the separation of individual tracks, depending on the type of error.

The non-analysable category has many more tracks than the analysable category, therefore the FLImP users need to manually inspect large volumes of tracks to collect enough analysable tracks for a FLImP experiment. This could result in loss of concentration, which when combined with the complexity of the track selection process would cause errors.

Another reason for the large number of errors is that some of the conditions for selecting track are vaguely defined. For example, Condition 18 from Section 5.1 is not specific and is likely to produce many inconsistencies.

An easy way to resolve this problem is to automate as much from the track selection process as possible. Chapter 6 describes a new track selection process which has high degree of automation. FLImP users working with the refined track selection process would need to inspect around 70 times fewer tracks, and for each track there is much shorter list of conditions to be manually checked.

Chapter 6

Refining Track Selection Process

The FLImP analysis requires tracks with specific properties which necessitates a track selection process. In the previous chapter it was shown that the track selection process used in FLImP is unreliable and prone to errors. Therefore a refined track selection process needs to be created.

This chapter introduces a refined track selection process, which has high degree of automation. The comparison with the previously used track selection process shows that the refined process reduces the human effort during a FLImP experiment approximately 70 times. The new process also reduces the computation time as fewer tracks are selected for analysis and increases the number of usable tracks as more tracks are within the targeted confidence interval and separation. Another advantage of the refined process is that it has high degree of automation and therefore is less prone to error.

As a result of these improvements the new track selection process was adopted as part of the FLImP method.

This chapter has 6 sections. The first section shows an overview of the refined track selection process. The second, third, fourth, and fifth section show the background assessment, identification of track structure, track selection, and manual inspection. The sixth section shows comparison between the old track selection process and the

refined version.

6.1 Track Selection Overview

The refined track selection process has five stages, Figure 6.1. The first four stages are automated and the last one is manual. The manual step involves checking whether the tracks structure meets several criteria.

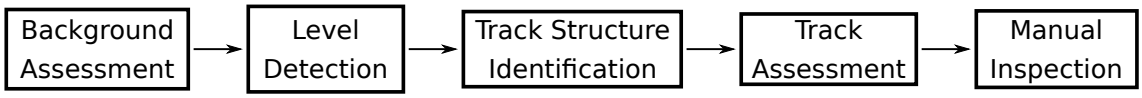


Figure 6.1: Overview of the refined track selection process.

1. **Background Assessment.** For each track the image of the neighbourhood of each feature is examined to determine whether the background fluorescence is a uniformly distributed noise or it has additional fluorophores or other non-uniform fluorescence. The automatic classification is done by a neural network, which is trained on a real data set, which is manually labelled in advance.

2. **Level Detection.** A level detection algorithm is applied to each track to find the constant intensity levels. The level detection algorithm used as part of the refined track selection process is called "Levels Noise". It is described in Section 4.2. Frames in which there is background fluorescence are excluded from the levels.

3. **Track Structure Identification.** After levels are identified, tracks with structure which can be analysed by FLImP are selected. This includes levels order, whether the intensity of the region before or after the track is zero when extrapolated, and the ratio of the mean intensity of level 2 and level 1. This section is semi-automated. There are three conditions which need to be checked manually during Stage 5

4. **Track Assessment.** This stage of the track selection process aims to reduce the computation cost of the FLImP experiments and the manual effort. This section uses supervised machine learning algorithms to classify tracks, based on their properties. The data set used for training the algorithm is constructed by tracks which passed the previous three stages and are analysed by FLImP. Then the confidence interval and the separation of each track is used for creating the data labels.

5. **Manual Inspection.** Tracks need to be inspected for problems which were not detected during the track structure identification. There are several guidelines which have to be followed.

6.2 Background Assessment

One of the assumptions of FLImP is that the background of the feature is uniformly distributed noise over the x, y plane. Deviations from such noise within 5 pixels away from the centre of the feature, similar to one shown in Figure 6.2, can skew the data[2]. Tracks which have frames with background fluorescence can still be analysed as long as the frames where this fluorescence occurs are excluded from the analysis.

The automatic exclusion of frames, in which the background is not uniform, is approached as a classification problem. A region of 11 by 11 pixels from the microscope image around the centre of the feature in each frame is extracted and rearranged as 121-dimensional vector, $\mathbb{R}^{121}$. Then these vectors are classified by a neural network.

The training set for the classifier, which is created by manually inspecting and labelling diffraction limited spots, has 3512 samples with equal number of samples in the negative and positive class.

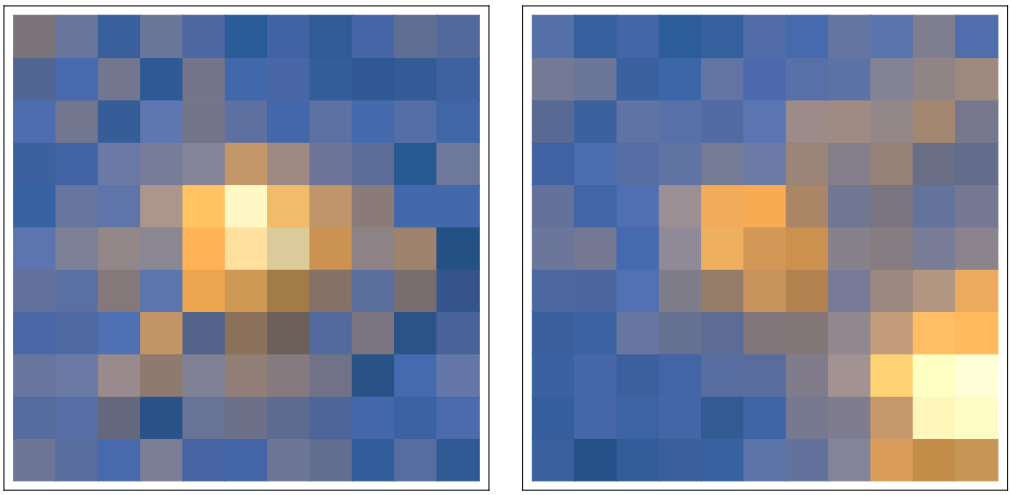


Figure 6.2: Two examples of the region of interest around a diffraction limited spot. The example on the left has uniformly distributed background and the one on the right does not.

6.2.1 Dimensionality Reduction

Classifying high dimensional data requires prohibitively large data set to reliably train a classifier. On the other hand the set of all features does not occupy the entire volume of $\mathbb{R}^{121}$, which means that the data can be projected to a lower dimensional space.

Two different dimensionality reduction techniques were compared – a principal component analysis (PCA) and an autoencoder (AC). Since the autoencoder is non-linear dimensionality reduction, it is possible for the algorithm to over-fit the data, and therefore the data set was split into training and testing set with 70% of the data in the training set and 30% in the testing set. It is also possible to use the dropout technique [163] [164] to avoid over-fitting.

The autoencoder and PCA were trained on the training set and the reconstruction error was measured on the testing set. The reconstruction error is the mean squared difference between the original samples, x_n and the reconstructed samples r_n

$$\mathcal{E} = \frac{1}{N}\sum_{n=1}^{N}(x_n - r_n)^T(x_n - r_n) \tag{6.1}$$

This test was repeated for projection to 25, 15, 12, 10, 8, 6, 5, 4, 3, and 2 dimensional space.

Methods

The architecture of the autoencoder was chosen manually, so that the hidden units in each layer were half of the previous layer. The autoencoder was trained as described in Section 2.3.2. Each RBM was trained for 5000 iterations with learning rate of 0.01 and velocity of 0.9. The initial values of the weights and biases of the RBMs were chosen from Gaussian distribution with zero mean and standard deviation of 0.3, $N(0, 0.3)$. The activation function of the RBM hidden and visible units was the hyperbolic tangent. For the training algorithm for the RBM was used a custom implementation. The algorithm was implemented in the wolfram language, Mathematica 11.3. Links to the implementation can be found in Appendix C.

Then the pre-trained autoencoder was trained with ADAM for 10 000 iterations with parameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The activation function of the hidden units and the output units of the autoencoder was the hyperbolic tangent. The error function was the mean squared distance. The implementation of the ADAM algorithm which was used was the one provided by Mathematica 11.3.

The data was scaled so that the minimum and maximum values are between -1 and 1.

Results

Figure 6.3 shows the reconstruction error for training and testing data set. The horizontal axis shows the projected dimensionality and the vertical axis is the reconstruction error. The reconstruction error on the training set is lower for the auto-encoder than

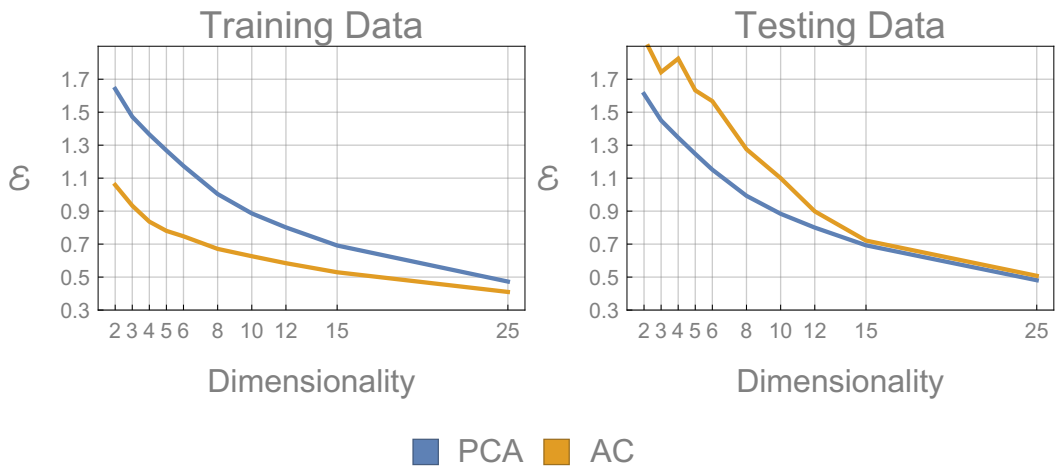


Figure 6.3: Autoencoder and PCA are trained to project the data to different dimensionality, and the reconstruction error is measured on the training and testing set.

the PCA. However, when the error was measured on the testing set, the PCA had lower reconstruction error.

Conclusion

The lower reconstruction error on the training set and the higher reconstruction error on the testing set shows that the autoencoder possibly remembers the noise in the data rather than just the structure. Therefore, PCA will be used for dimensionality reduction in this classification problem.

Figure 6.4 shows the first 8 principal components of the training set, which capture 61.79% of the variance of the data.

6.2.2 Classification

After the data dimensionality was reduced a neural network with different number of hidden units was trained for each compression level.

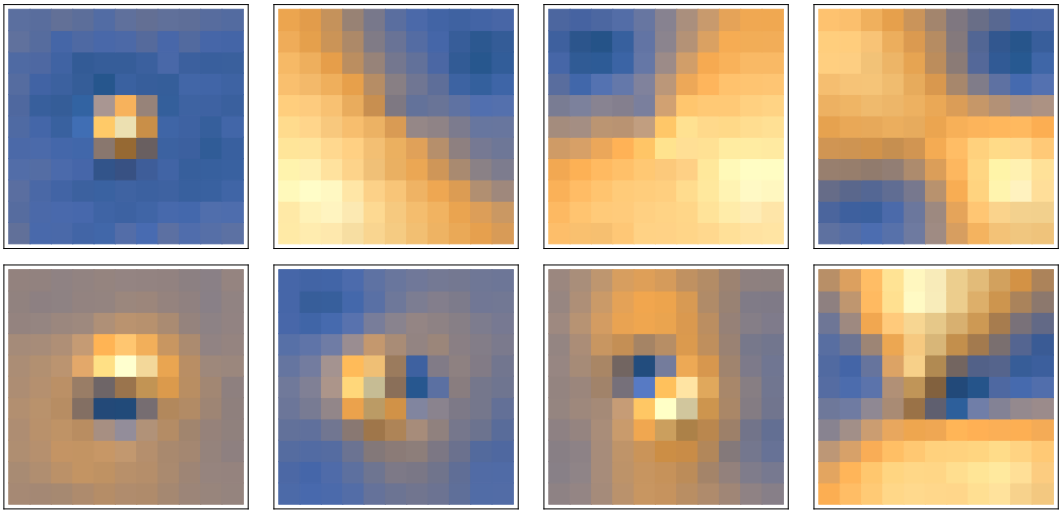


Figure 6.4: The first 8 principal components of the data set, plotted as images

Methods

The training algorithm was RMSProp with parameters $\gamma = 0.95$ and momentum $\theta = 0.9$. Each network was trained for 15000 iterations. The initial weights of the networks were random orthogonal matrix. The network had one hidden layer with hyperbolic tangent as activation function. The activation function of the output layer was the softmax function. The error function was the cross-entropy. Each network was trained and tested 30 times, and the labelled data set was randomly shuffled and split into training set, 70%, and testing set, 30%, before every repeat. The random shuffle was done by the function *RandomSample* from Mathematica 11.3. The implementation of the training and initialisation algowithm was provided by Mathematica 11.3.

Results

Figure 6.5 shows the average training and testing accuracy of a neural network with different number of hidden units. The horizontal axis is the number of hidden units for each network and the vertical axis is the mean classification accuracy. The error bar has

length of one standard deviations.

For the low dimensional projections the training accuracy did not reach 1.0, and the testing accuracy increased with the number of hidden units, reached the highest point, and then decreased. This behaviour is typical when too many hidden units are used and the model over-fits the data.

For projections to 10 and higher dimensional space, the training accuracy reached 1.0 very fast, but the testing accuracy remained lower. For these projections, the testing accuracy increased, reached a peak, dropped, and then increased again. The increase of accuracy cannot be attributed to better model, but to over-fitting due to insufficient data. Another observation is that for high dimensional projections fewer hidden units were required to reach the first peak in the testing accuracy.

Figure 6.6 shows a plot of the highest testing accuracy which is not considered over-fitting against the dimensionality of the data projection. The highest testing accuracy was achieved on data projected to the first 8 principal components at 95.4 % average testing accuracy with 1.4% error bar (one standard deviation). The neural network used for this classification had 7 hidden units. Since this is the model with highest testing accuracy it will be used for classification of diffraction limited spots.

Conclusion

The results show that the best classifier for classifying the diffraction limited spots into uniformly and non-uniformly distributed background is a neural network with 7 hidden units trained on a data set projected to the first 8 principal components by PCA. The average accuracy of such classifier is 95.4% which is considered good enough to be used in the FLImP track selection process.

Since there is 4.6% error rate, the track selection process still requires human assistance to avoid analysis of features which have non-uniform background fluorescence. It is important to avoid analysing such features, because they may introduce systematic

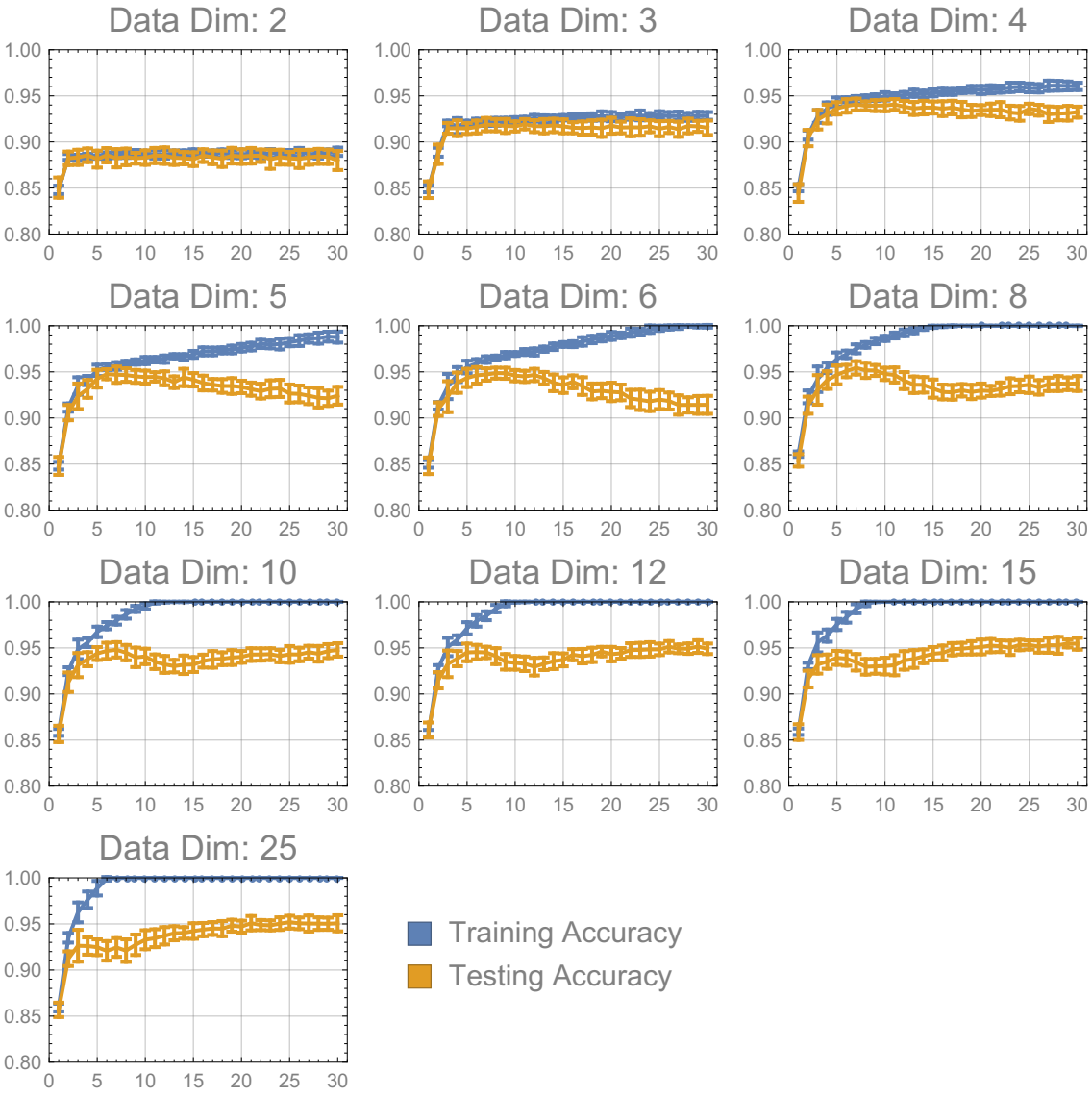


Figure 6.5: The background samples projected onto different number of principal components and classified with neural network. The horizontal axis are the number of hidden units and the vertical axis is the mean classification accuracy over 30 training attempts.

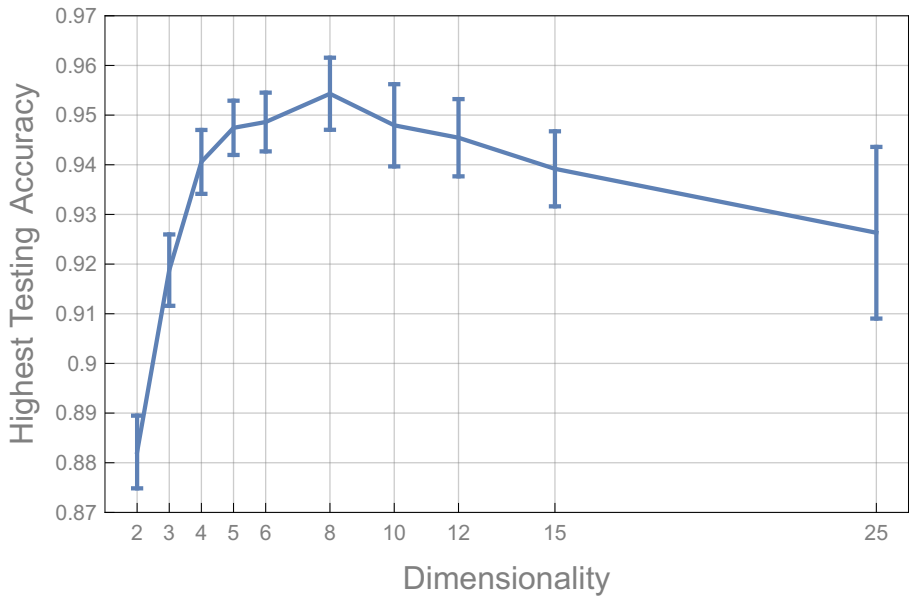


Figure 6.6: For each data projection, the highest testing accuracy is plotted. The horizontal axis is the dimensionality of the projection, and the vertical axis is the highest testing accuracy.

error to the measurement.

6.3 Identify Track Structure

After the constant intensity levels are identified and the frames with non-uniform noise are excluded, the track selection process needs to find tracks which meet the rest of the FLImP assumptions. Therefore, tracks with specific structure need to be selected.

1. Track should have at least two levels – *level 1* and *level 2*.
2. Track should either
   (a) finish with two levels, where the last level should have lower intensity (*level 1*) than the previous level (*level 2*) and then the intensity should be zero. An example is shown on the left-hand side of Figure 6.7
   (b) have zero intensity and then start with two levels where the first level has lower intensity (*level 1*) than the second level (*level 2*). An example sequence is shown on right-hand side of Figure 6.7.

   For this condition only the last two levels are examined. This means that *level 1* might not be the least bright level in the track but the one with lower intensity from the last two levels.
3. The intensity of *level 2* should be twice the intensity of *level 1*.
4. The position during the levels should be constant.
5. There should be no gap between *level 1* and *level 2*.
6. If the tracks starts with *level 1*, there cannot be any intensities before *level 1* different form the intensity level. If the track ends with *level 1*, there cannot be any intensities after *level 1* different from the intensity of the level.

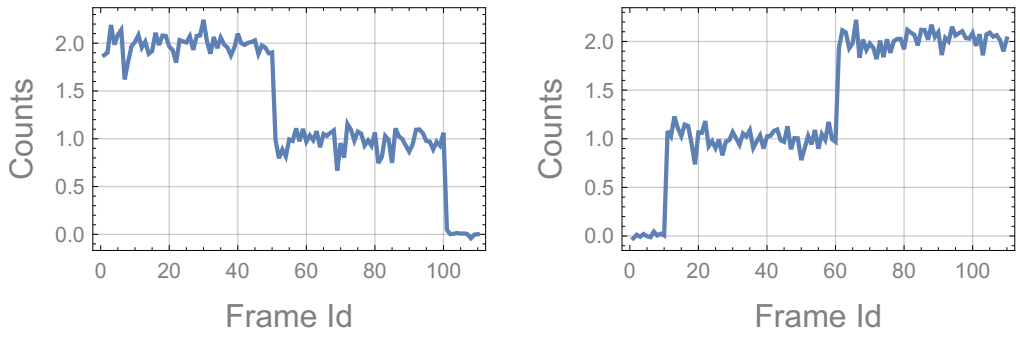


Figure 6.7: The levels order for analysable tracks can be either *level 2* followed by *level 1* and then zero intensity, or zero intensity followed by *level 1* and then *level 2*.

If conditions 1, 2, 5, and 6 are violated this is an evidence that there might be different number from 2 active fluorophores in *level 2* and one active fluorophore in *level 1*. It is important that the intensity after the end of *level 1* in case it is last, or before *level 1* in case it is first, to be zero. Otherwise, it is possible that there is another fluorophore at larger depth, which is hard to detect, and it may skew the distance measurement.

Condition 3 guarantees that FLImP will not underestimate the separation. Condition 4 is relevant to the FLImP assumption that the fluorophore positions do not change during levels. If a track does not meet that condition, this means that it is possible that the level detection algorithm has made a mistake, and the track should be either rejected all together or just a subsection of the level should be selected, depending on the case.

If a track meets all these conditions it is assumed that there are 2 active fluorophores in *level 2* and only one of these fluorophore is active in *level 1*. Such assumption is the basis for FLImP [2], SHRImP [53], and gSHRImP [40]. Having such assumption allowed some papers to be published with FLImP [1], [165], [22].

Tracks which do not meet conditions 1, 2, and 3 are rejected automatically, Section 6.3.2, and Section 6.3.1. Tracks which do not meet conditions 5, 4, and 6 are inspected manually, Section 6.5.

6.3.1 Fluorophore Depth Difference

FLImP uses total internal reflection (TIRF) to excite fluorophores [64] which are close to the surface of the sample. This technique uses the evanescent field at the point of reflection to excite the fluorophores. The evanescent field decays exponentially with the depth, which means that if fluorophores are at different depth, they will have different intensities. Therefore, in a track generated by two fluorophores, the intensity of *level 2*, where both fluorophores are active, should be twice as large as the intensity of level 1, where there is only one active fluorophore.

Measuring distances between fluorophores at different depth should be avoided, because FLImP measures the distance projected on the sample surface. If the fluorophores are at different depth, FLImP will measure distance, m, which is

$$\frac{m}{\sqrt{m^2 + \Delta d^2}} \tag{6.2}$$

times smaller than the real distance, where Δd is the depth difference between the fluorophores. The depth difference can be calculated by using the formula for exponential decay

$$\frac{I_1}{I_2} = e^{A\Delta d} \tag{6.3}$$

where I_1 is the intensity of the fluorophore which is active in *level 1* and *level 2*, I_2 is the intensity of the fluorophore which is active in *level 2*, A is the decay constant, and $\Delta d = d_1 - d_2$. Using the mean level intensities, Formula 6.3 can be rewritten as

$$\frac{\langle I(l2)\rangle}{\langle I(l1)\rangle} - 2 = e^{-A\Delta d} - 1 \tag{6.4}$$

where $\langle I(l2)\rangle$ is the mean intensity of *level 2*, and $\langle I(l1)\rangle$ is the mean intensity of *level 1*. The quantity on the left of the equation is the track property "stepratio_lvl12", which is used in Section 5 for the single molecule imaging data classification.

The decay constant A depends on the parameters of the experiment, and Δd depends on the application of FLImP. From these parameters, a threshold $\tau_{\text{stepratio\_lvl12}}$ can be calculated and used to reject tracks. In FLImP usually a threshold of 0.2 or 0.3 is used.

6.3.2 Level Order

Tracks need to have specific intensity pattern so it can be assumed that each level has the expected number of fluorophores. There are two patterns which are analysed by FLImP, Figure 6.7.

The first intensity pattern requires *level 2*, followed by a photobleaching step, followed by level 1, and then followed by another photobleaching step. After the last photobleaching step, the intensity at the position of the last observed diffraction limited spot should be zero.

The second pattern which can be analysed starts with zero intensity, then a fluorophore comes into emitting state to form *level 1*. Then *level 1* is followed by another fluorophore coming into emitting state, which forms *level 2*.

The first intensity pattern is much more frequently observed than the second intensity pattern. The levels can be interrupted by a drop in the intensity for one or two frames, which usually is due to a fluorophore going into dark state and then coming back into emitting state.

Both intensity patterns require that there is zero intensity before *level 1* for the second pattern, and after *level 1* for the first pattern. The reason for this is that if there is another fluorescence at the position of the diffraction limited spot, for example generated by a fluorophore deeper in the sample which is not detected, it may skew the FLImP measurement.

The requirement of the intensity to be zero is checked using statistical test, because there is a noise in the system. The intensity of 20 frames before or after a track, depending on which intensity pattern is detected, are used to calculate the mean intensity,

$\langle I \rangle$, and the standard error in mean $\sigma_{\langle I \rangle}$. Then if the test

$$\left| \frac{\langle I \rangle}{\sigma_{\langle I \rangle}} \right| < 3 \tag{6.5}$$

fails, the intensity is assumed to be different from zero.

6.4 Track Selection

Not every track which can be analysed by the FLImP algorithm is relevant to the biological study. For example, measurements with confidence interval of 20 nm cannot be used to distinguish between separations of 14 and 20 nm, and separations of 200 nm could not be measured from an EGFR oligomer. Therefore, additional step is required, called "Track Selection", in which the analysable tracks need to be filtered, so that the number of tracks which are analysed but are not relevant to the project is minimised, hence the manual effort is also reduced. Tracks with confidence interval larger than 10 nm or separation larger than 60 nm are considered irrelevant to the study and need to be rejected.

6.4.1 Data Extraction and Representation

The training set was created by selecting and analysing 746 tracks with FLImP. The track selection was done by applying the first 3 steps of the FLImP filter, "Background Assessment", "Level Detection", and "Track Structure Assessment". Every track which meets the FLImP assumptions was analysed, regardless of the position shift at the point of photobleaching, or the noise in the levels.

The tracks which had confidence interval lower than 10nm and separation less than 60nm were put in category "used", or positive, and the rest were put in category "unused", or negative.

The prediction of the confidence interval and the separation of a track is based on 10 properties.

- Standard error in mean difference between position of level 1 and level 2. This property is related to the confidence interval.

$$\sigma_{\overline{p_{12}}} = \sqrt{\frac{\left(\overline{x_1} - \overline{x_2}\right)^2 \left(\sigma^2_{\overline{x_1}} + \sigma^2_{\overline{x_2}}\right) + \left(\overline{y_1} - \overline{y_2}\right)^2 \left(\sigma^2_{\overline{y_1}} + \sigma^2_{\overline{y_2}}\right)}{\left(\overline{x_1} - \overline{x_2}\right)^2 + \left(\overline{y_1} - \overline{y_2}\right)^2}} \tag{6.6}$$

- Standard error in mean of the position, $\boldsymbol{p} \in \mathbb{R}^2$, of level l.

$$\sigma_{\overline{p_l}} = \frac{\sigma_{p_l}}{\sqrt{T_l}} \tag{6.7}$$

- Standard deviation of the position, $\boldsymbol{p} \in \mathbb{R}^2$, of level l.

$$\sigma_{p_l} = \sqrt{\frac{1}{T_l - 1} \sum_{t=1}^{T_l} \left(\left(x_l^t - \overline{x_l}\right)^2 + \left(y_l^t - \overline{y_l}\right)^2\right)} \tag{6.8}$$

- Difference between the mean position of level 1 and level 2. Large difference indicates large separation.

$$\Delta p_{12} = \left(\overline{x_1} - \overline{x_2}\right)^2 + \left(\overline{y_1} - \overline{y_2}\right)^2 \tag{6.9}$$

- Clustering. This property measures how well the intensities of level 1 and level 2 are separated.

$$C = \frac{\sigma_{I_1} + \sigma_{I_2}}{\overline{I_1} - \overline{I_2}} \tag{6.10}$$

- Ratio of noise to signal of level 1 and level 2. Large noise-to-signal ratio indicates large confidence interval.

$$S = \frac{\sigma_{\overline{I_1}}}{\overline{I_1}} + \frac{\sigma_{\overline{I_2}}}{\overline{I_2}} \tag{6.11}$$

- The sum of all intensities in level l. Tracks which are brighter or longer generate shorter confidence intervals.

$$Is_l = \sum I_l \tag{6.12}$$

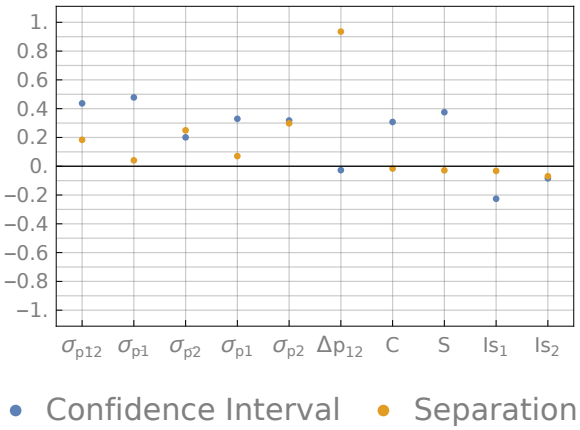


Figure 6.8: Correlation between each property and the confidence interval and separation of the analysed tracks.

The index l takes the values 1 and 2, I_l is the set of intensities of level l, and x_l and y_l are the set of x and y positions for level l.

Figure 6.8 shows the correlation between each property and the confidence interval and the separation of the analysed tracks.

6.4.2 Dimensionality Reduction and Classification

It is possible for the data to occupy a subspace in the property space. Therefore, two dimensionality reduction methods were applied, PCA and autoencoder. The labelled data set was projected to lower dimensional space with each method and the projected data was classified by a neural network.

Methods

Before applying PCA, the data set was transformed, so that it had mean 0 and standard deviation of 1 in each dimension.

The autoencoder was trained as described in Section 2.3.2. The activation function of the hidden and visible units of the restricted Boltzmann machine was the hyperbolic tangent, and therefore the data was transformed, so that the maximum value in each dimension was 0.95 and the minimum value was -0.95. Since there were no outliers, the

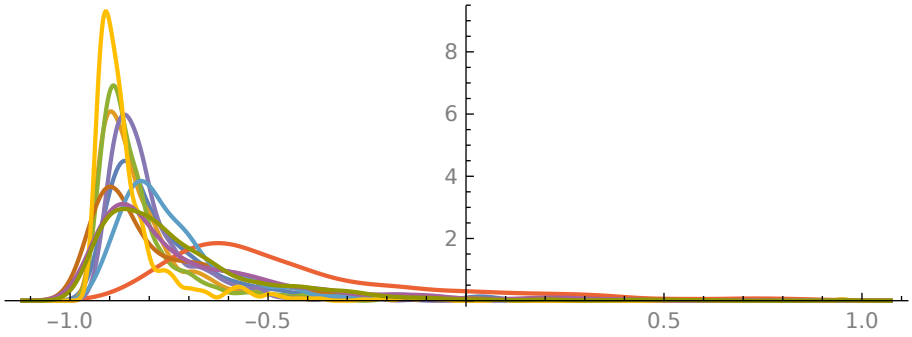


Figure 6.9: Kernel density estimation of the data for each dimension after transformation, so that the maximum value is 1.95 and the minimum values is -1.95.

transformation did not cause very small variance in any of the dimensions, Figure 6.9. Each restricted Boltzmann machine was trained for 5000 iterations with learning rate of 0.01 and velocity 0.9. The initial values of the weights and biases of the RBMs were chosen from Gaussian distribution with zero mean and standard deviation of 0.3, $N(0, 0.3)$. For the training algorithm for the RBM was used a custom implementation. The algorithm was implemented in the wolfram language, Mathematica 11.3. Links to the implementation can be found in Appendix C.

The second step of the training was done with the ADAM algorithm, for 10000 iterations, parameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$, and the error function was the mean squared difference. The encoder layers were 10-8-6-4. The activation function of the hidden layers and the output layer was hyperbolic tangent.

After compression, the data was transformed again, so that it had mean 0 and standard deviation 1 in each dimension, because the weights of the neural network were initialised with the Xavier method using Gaussian distribution and this transformation would aid the training process. The training algorithm was ADAM with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The implementation of the Xavier method and the ADAM algorithm was the one provided by Mathematica 11.3.

The training was repeated 10 times for 10000 iterations, and before each repeat the

data set was split randomly to training and testing set, where the training set was 70% of the data. The random split was done by the function *RandomSample* from Mathematica 11.3. A neural network with different number of hidden units was trained and tested for each repeat. The hidden units started from 1 to 10 with increments of 1, then from 12 to 20 with increments of 2, and then 25 and 30. The activation of the hidden units was hyperbolic tangent and the error function was the cross-entropy. This classification was also done on the uncompressed data set, which was transformed before classification so that it had mean 0 and standard deviation of 1 in each dimension.

Results

Figure 6.10 shows the reconstruction error measured on training and testing set for different levels of compression when PCA and autoencoder (AC) was used. The error was calculated according to Formula 6.1. The autoencoder had larger difference between the training and testing reconstruction error than the PCA, when the data was projected to 2 dimensions. This means that the autoencoder tends to over-fit the data more when compressing it to 2 dimensions than when compressing it to 4 and 6 dimensions.

Figure 6.11 shows the mean F1-Score measured on training and testing data set for each number of hidden units, each dimensionality, and each projection method. It can be seen that in 2 dimensions, the PCA performs better than the autoencoder, but in higher dimensions, there is no difference between the projection methods. This is consistent with the results shown in Figure 6.10, which imply that the autoencoder over-fits the data when projecting it to 2 dimensions.

The projection done by autoencoder to 4 and 6 dimensions requires a neural network with more hidden units to learn the class boundaries than the projection done by PCA. It can be seen that classifiers trained on data projected to higher dimensional space perform better than classifier trained on data projected to lower dimensional space.

Figure 6.12 shows the mean F1-Score measured on the testing data set for neural

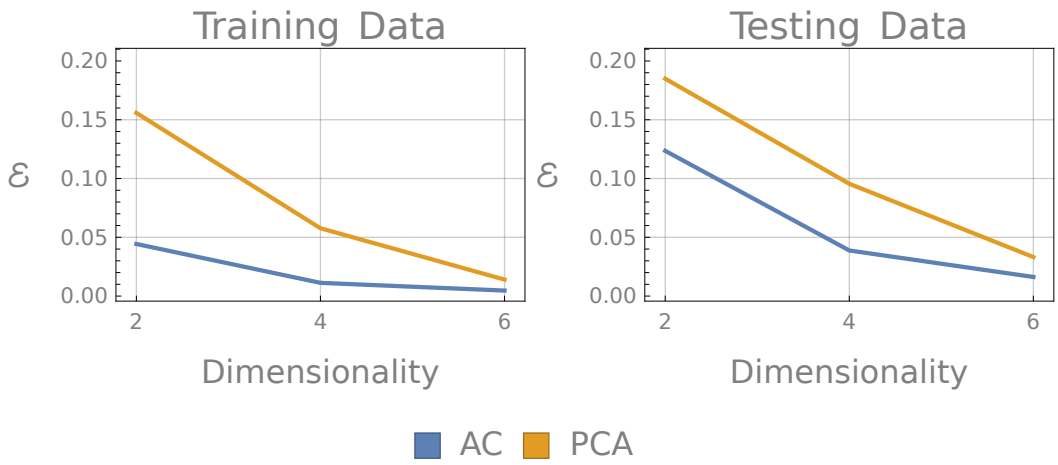


Figure 6.10: Autoencoder and PCA were trained to project the data to different dimensionality, and the reconstruction error was measured on the training and testing set.

network trained on the uncompressed data set, which has 10 dimensions, and the projected data set to 8 dimensions by PCA and autoencoder. The best performance was for 1 hidden unit, which corresponds to a linear boundary, on the uncompressed data set, which has f1 score of 0.8582. When the data was projected to lower dimensional space, the performance of the classifier also dropped. This means that lower dimensional representations lose information which is important to the classification.

Figure 6.13 shows the ROC curve of a linear classifier evaluated on the testing set. False positive samples have lower cost than false negative, because FLImP evaluations of the tracks are much cheaper than preparation of experiments. Therefore, the threshold for the classifier can be set, so that the false positive rate is 0.15 and the recall is 1.

Conclusion

The results show that the best classifier is a linear classifier applied on the uncompressed data set and it achieved f1 score of 0.8582. Taking into account the ROC

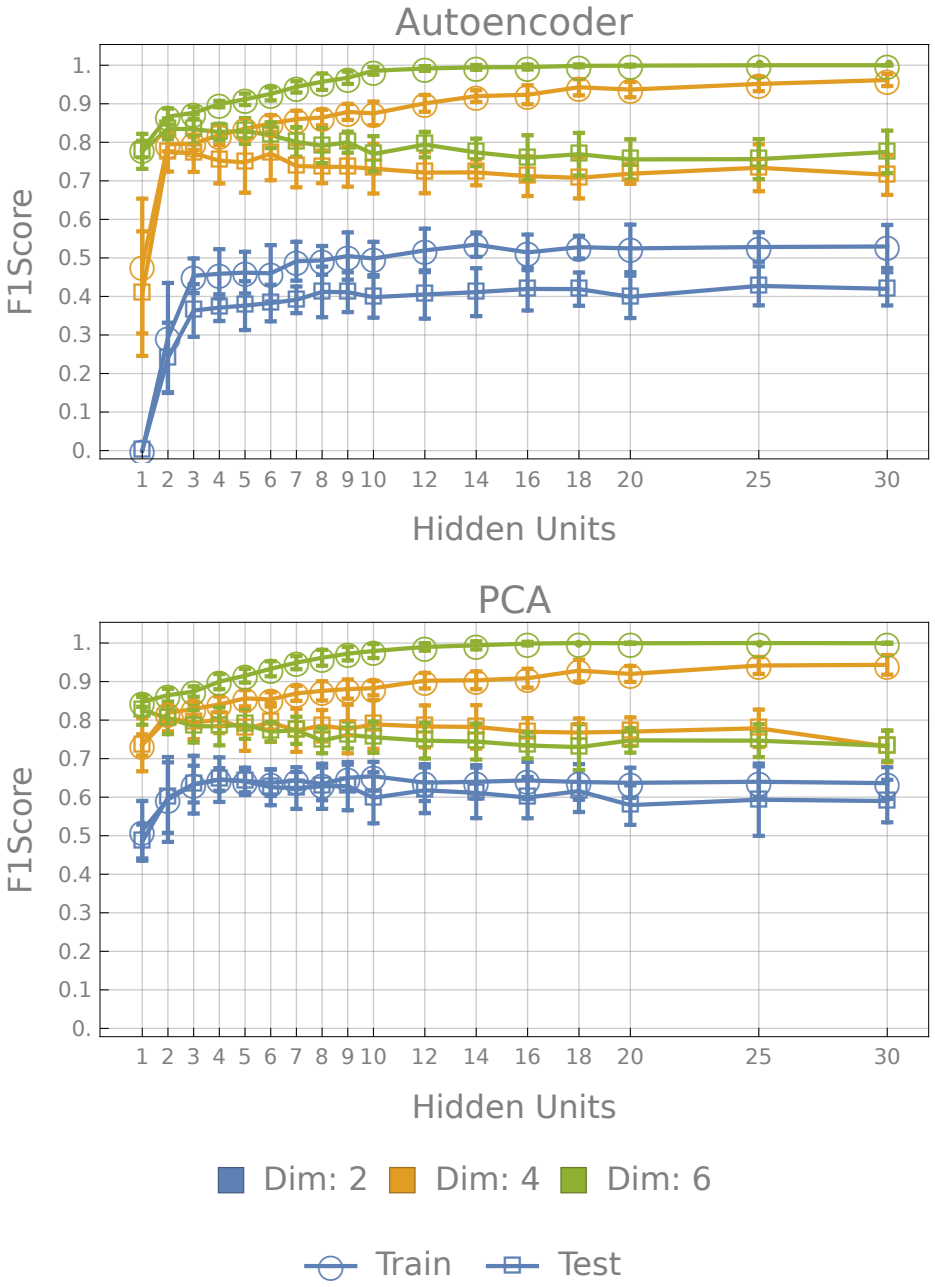


Figure 6.11: Training and testing F1-Score of neural network with hidden units from 1 to 30, trained on data set projected to 2, 4, and 6 dimensions. The plot on the top shows the results for data projected with autoencoder and the plot on the bottom shows the result for data projected by PCA.

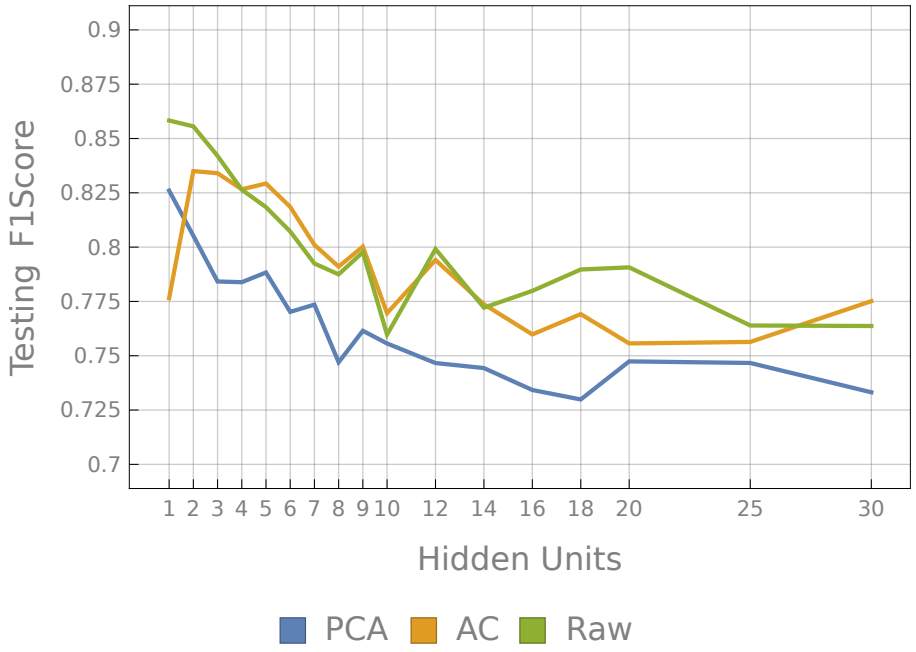
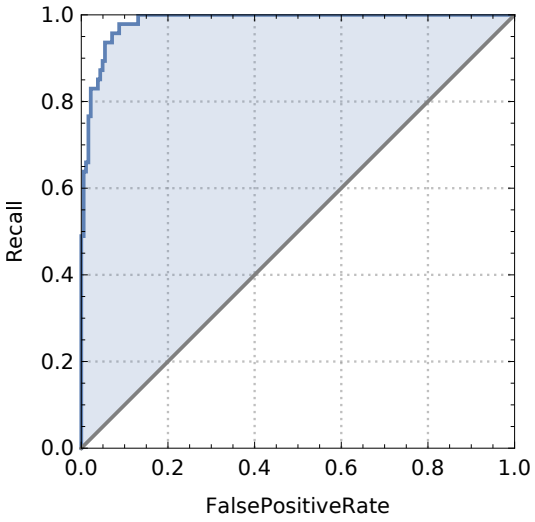


Figure 6.12: Testing F1-Score of neural network with hidden units from 1 to 30 trained on original data set with 10 dimensions and projected data set to 8 dimensional space by autoencoder and PCA.

Figure 6.13: ROC curve evaluated on testing set of a linear classifier.



curve of that classifier and the costs of experimental preparation and processing cost the threshold for that classifier is adjusted so that it has recall of 1 and false positive rate of 0.15.

This stage of the track filter optimises the FLImP analysis process by finding tracks which are likely to produce measurement which are relevant to the study. Since FLImP is used frequently for studying the EGFR, this section is optimised for finding tracks with separations and confidence intervals specific for EGFR. It is possible to use FLImP for studying of different types of oligomers, for which may be typical different separations. The stage of the track filter described here can be adapted for different application by changing the data labels based on the targeted separation and confidence interval and re-training the classifier.

6.5 Manual Inspection

This stage of the track selection process involves manual inspection of the tracks. There are four criteria which the tracks need to meet to be selected for analysis.

1. **Gap between levels** There should be no gaps between *level* 1 and *level 2*. Any intensity different from the levels indicates that the assumption that there is only one active fluorophore in *level 1* and there are two active fluorophores in *level 2* and of these fluorophores is the same as the one in *level 2* is violated. Examples of such tracks are shown in Figure 6.14.

2. **Intensity after or before level 1** If the track starts with *level 1*, and then it is followed by *level 2*, there should not be any frames with intensity different from the intensity of *level 1* before the level. In case the track ends with *level 1*, there should not be any such frames after the level. Example of tracks which have such frames is shown in Figure 6.15.

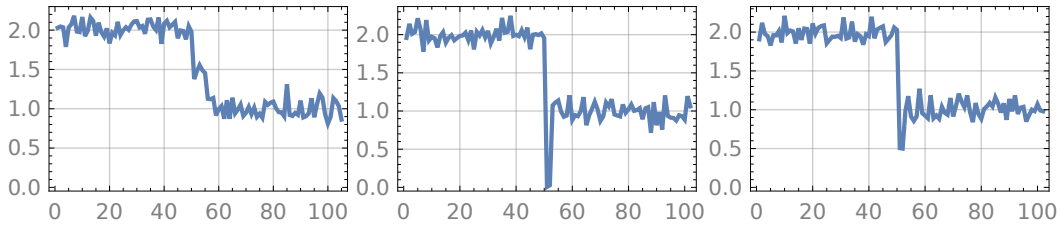


Figure 6.14: Examples of tracks which have intensity between level 1 and *level 2*

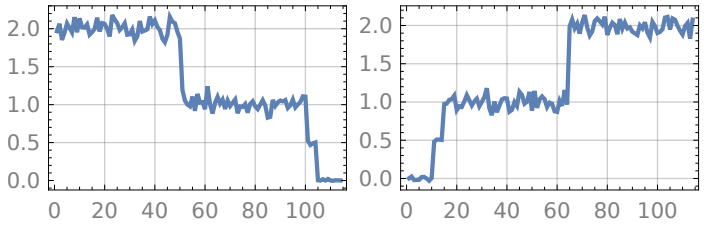


Figure 6.15: Example of tracks which have intensity before or after *level 1*.

3. **Background fluorescence** The neural network used for background assessment has small percentage false positives. Therefore, the area around the diffraction limited spot needs to be checked for non-uniform background fluorescence. If any such frames are found, they should be excluded from the analysis.

4. **Constant position** During constant intensity levels, the position of the diffraction limited spot should be constant. Any changes in the position during levels is an indication that there has been an error in the tracking or the level detection algorithm. Example of such tracks is shown on Figure 6.16.

6.6 Selecting Simulated Tracks

This section will apply the automated track selection on simulated tracks to verify that it works correctly. The filter is applied to 9 data sets each of which has 9 tracks. Eight of the data sets have tracks which should not be selected by the filter and in one

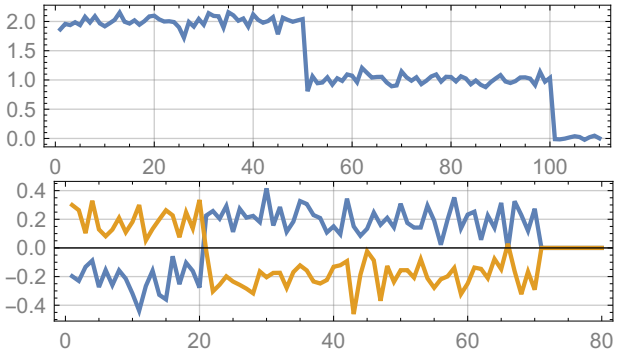


Figure 6.16: Example of a track which has a position change during constant intensity level.

data set should the tracks should be selected. Each of the 8 data sets has different reason why it shouldn't be selected by the filter. Each type of simulated tracks is described below:

- *Step Ratio* This type of tracks is generated by fluorophores at different depth.
- *Extrapolated intensity* In this type of tracks the intensity before or after level 1, depending on the levels order, is different from zero when extrapolated. Two data sets are generated. One when level 1 is first and one when level 1 is last.
- *Neighbour* There is one or two fluorophores in close proximity to each track.
- *1 level* These tracks have only one level.
- *Ends with level 3* In this type of tracks level 1 is in the middle. Before and after level 1 there is a level with higher intensity.
- *Large confidence interval* This type of tracks can be analysed, however the tracks have very large confidence interval when analysed.
- *Large separation* This type of tracks can be analysed, however the separation of the tracks is too large to be of interest to the project.

Data Set	Selected Tracks
Step Ratio	0
Extrapolated Intensity (level 1 last)	0
Extrapolated Intensity (level 2 last)	0
Neighbour	0
1 level	0
Ends with level 3	0
Large confidence interval	0
Large separation	0
Used	9

Table 6.1: Number of tracks which were selected for analysis by the automatic track filter.

- *Used* This type of tracks can be analysed and the confidence interval and separation are of interest. These tracks should pass the filter.

Details of the parameters of the simulations can be found in Section 3.8.

6.6.1 Results

Table 6.1 shows for each data set how many tracks were selected when the filter was applied. All of the data sets which contain non-analysable tracks had their tracks rejected by the filter. The data sets which had analysable track, however the confidence interval or the separation were too large also had their tracks rejected. The only data set which had all the tracks selected was the analysable tracks with small enough confidence interval or separation.

Receptor	Mutation	Ligand	Concentration	Treatment	Time Period
HER1	L680N	EGF	4 nM	Methyl Beta Cyclodextrin	25.04.2017 - 02.05.2017
HER1	I942E	Affibody	4 nM		18.08.2016 - 15.03.2017

Table 6.2: Experimental conditions and dates for data set used in filter evaluation.

6.6.2 Conclusion

The results in Table 6.1 show that the automatic track selection process works as intended. All the track which should have been rejected were rejected and the tracks which had to pass the filter passed it.

6.7 Evaluation of Refined Track Selection Process

The refined track selection process is compared with the previously existing track selection process in terms of number of tracks required to be manually inspected, number of tracks selected for analysis, and number of tracks used in the final result.

6.7.1 Methods

The comparison was done on two experimental conditions, under which 459 data sets were collected and analysed. The experimental conditions are shown in Table 6.2.

6.7.2 Results

Table 6.3 shows the results of how many tracks were manually inspected, how many tracks were analysed, and how many tracks have confidence interval less than 10nm and

separation less than 60nm for each version of the track selection process. The refined process handled many conditions automatically, therefore many tracks were rejected without human assistance. It required around 70.31 times fewer tracks to be manually processed than the original process. When the refined process was used, there were fewer tracks selected for analysis than when the original process was used. However, the tracks which have low confidence interval and separations in the desired range are more than those selected by the original process.

	Inspected	Analysed	Used (ci $\leq$ 10nm, sep $\leq$ 60nm)
Original Process	32766	253	97
Refined Process	466	231	108

Table 6.3: Comparison between the refined track selection process and the original track selection process, based on the number of inspected tracks, the number of analysed tracks, and the number of tracks used in the FLImP studies

6.7.3 Conclusion

These results show that the refined track selection process requires fewer tracks to be manually inspected and fewer tracks to be analysed by FLImP. It also finds more tracks from the same number of data sets. It can be concluded that the refined process is more efficient than the original version, because it reduces the false positive and false negative samples.

Chapter 7

Applications of Refined Track Selection Process

The experiment presented in this chapter tries to determine the effects of phorbol myristate acetate (PMA) when introduced to wild type epidermal growth factor receptor (WT EGFR), treated with Bisindolylmaleimide I (BM-I). The PMA is known to cause phosphorylation of the EGFR and it can be blocked by BM-I [166].

The experiment measured sizes of oligomers formed in cells expressing WT EGFR treated with BM-I only and cells treated with both BM-I and PMA.

Although the results of the experiment showed that more data and a more sophisticated model are required to reveal details about the effects of PMA on the WT EGFR, the large repeatable distances observed in the dataset treated with BM-I were not observed in dataset treated with BM-I and PMA (BM-I+PMA) in favour of large quantity of small distances. Therefore, it could be concluded that the PMA causes the WT EGFR to form smaller oligomers. Also the results implied existence of higher order oligomers, which was confirmed by electron microscopy data.

This chapter is organised in 5 sections. Section 7.1 describes the experimental conditions and preparation of the experiments. Section 7.2 describes the Rice model used for

interpreting the FLImP data (the combined bootstrap samples of all analysed tracks) and the Bayesian information criteria used to find how many distinct distances are observed. Then Section 7.3 shows the results from the experiment, and Section 7.4 discusses the significance and the validity of the results. Finally Section 7.5 presents the conclusions, and implications of the results. It also suggests collecting mode data and improvements to the Rice model to provide greater details about effects of the PMA on the WT EGFR.

7.1 Methods

There were 8 samples prepared for the experiments described in this chapter, Table 7.1. All of the samples were seeded with Chinese hamster ovary (CHO) cells expressing WT EGFR. The fluorophores used for the imaging is CF640R and the ligand was Epidermal Growth Factor (EGF). Each EGF had one CF640R molecule.

Two of the samples were controls for the experiments. Number 1 was a blank to show that when treated with BM-I there was no background, and the other blank (sample 2) was treated with PMA to show that it doesn't introduce any background or artefacts. All of the other six samples were WT EGFR treated with EGF-CF640R. Three of the samples were treated with PMA. All eight samples were treated with BM-I.

BM-I treated datasets came from dishes 3 to 5 inclusive, and BM-I+PMA treated datasets came from dishes 6 to 8 inclusive. The samples were prepared as follows:

- CHO cells (10^5 cell per dish) expressing EGFR under an inducible Tet-ON promoter were a gift from Prof Linda Pike (Washington University). Cells were grown in 5% CO_2 in air at 37 °C in phenol-red-free DMEM supplemented with 10% (v/v) fetal bovine serum, 2 mmol glutamine, 1% penicillin-streptomycin, 100 mg/ml hygromycin and 100 mg/ml geneticin. 50 ng/ml Doxycycline was also included to induce expression of EGFR. Cells were seeded onto 1% bovine serum albumin (BSA) coated glass bottomed dishes (MatTek Corporation)

Sample	2 nmol EGF-CF640R	PMA	BM-I
1	-	-	x
2	-	x	x
3	x	-	x
4	x	-	x
5	x	-	x
6	x	x	x
7	x	x	x
8	x	x	x

Table 7.1: All the samples used for the experiment presented in this chapter.

- The cells were grown for 48 h.
- Media was changed to 0.1% serum with 50 ng/ml doxycycline for 2 h
- The samples were washed with phosphate buffered saline (PBS) and treated with 2 µmol BM-I for 20 min at 37 °C.
- Washed the samples three times with PBS
- Left the samples on ice at 4 °C for 10 min.
- Labelled samples 3 to 8 inclusive with 2 nmol EGF-CF640R on ice at 4 °C for 1 h.
- Washed 3 times with ice cold PBS (the tube of PBS was kept in ice)
- Fixed samples with 3% (of the total solution) paraformaldehyde (PFA) + 0.5% glutaraldehyde. Incubated for 15 min on ice then 15 min at room temperature.
- Washed 3 times with PBS.

7.2 Rice Model

A FLImP experiment normally includes a number of distance measurements N_d. Each distance measurement corresponds to a molecular structure, which can be a dimer, trimer or larger oligomer. Therefore the measured distances will be a set of discrete distances each of which will be observed one or more times.

The model used to describe the FLImP separations is a mixture of Rice distributions with N_c components,

$$P\left(r|\boldsymbol{w},\boldsymbol{\sigma},\boldsymbol{R_T},N_c\right)=\sum_{c=1}^{N_c}w_c\text{Rice}\left(r|R_{Tc},\sigma_c\right) \tag{7.1}$$

where R_{Tc} is a size of molecular structure, of which there can be several measurements, σ_c is the spread, and w_c is the weight if each component. In this model using Gaussian distribution for describing the molecule separations is not appropriate, because the Gaussian distribution would overestimate small separations [167]. On the other hand if two emitters have Gaussian localisation error, the measurement of the distance between these emitters would follow the Rice distribution as shown in the supplementary information of the paper published by Needham et al. in 2016 [22].

Details of the model and the fitting procedure described in this section can be found in the supplementary information of [22]

FLImP uses the bootstrap technique to provide a sample of a probabilistic distribution over each distance, Chapter 1.7.1. For each separation FLImP generates N_b bootstrap samples. Normally N_b is set to 1200. Each bootstrap sample of each observed separation is treated as independent and identically distributed random variables. Therefore the probability of the observed data $D=\{r_{b,d}\},\ b=1..N_b,\ d=1..N_d$ given the model parameters is

$$P\left(D|\boldsymbol{w},\boldsymbol{R_T},\boldsymbol{\sigma},N_c\right)=\prod_{b=1}^{N_b}\prod_{d=1}^{N_d}P\left(r_{b,d}|\boldsymbol{w},\boldsymbol{\sigma},\boldsymbol{R_T},N_c\right) \tag{7.2}$$

.

The probability of the parameters, given the data can be estimated using the Bayesian theorem. Since the data does not change for a given experiment, the non-normalised probability of parameters given data is maximized,

$$P\left(\boldsymbol{w}, \boldsymbol{R_T}, \boldsymbol{\sigma}, N_c|D\right) \sim P\left(D|\boldsymbol{w}, \boldsymbol{R_T}, \boldsymbol{\sigma}, N_c\right) P\left(\boldsymbol{w}, \boldsymbol{R_T}, \boldsymbol{\sigma}|N_c\right) \tag{7.3}$$

where

$$P\left(\boldsymbol{w}, \boldsymbol{R_T}, \boldsymbol{\sigma}|N_c\right) = \prod_{j}^{N_c} \pi_{w_j} \pi_{\sigma_j} \pi_{R_T j} \tag{7.4}$$

is the probability of parameters given the number of components. The parameter π_{w_j} is

$$\pi_{w_j} = \begin{cases} 1 & \text{if } \frac{w_j}{\sum_{i=1}^{N_c} w_i} > \frac{0.9}{N_d} \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

.

This ensures that no component in the model can correspond to fewer than 0.9 measurements. In principle no component should correspond to less than one measurement, but the model allows for a small margin of error. The parameter π_{σ_j} is

$$\pi_{\sigma_j} = \begin{cases} 1 & \text{if } \min\left(\boldsymbol{\sigma}_{\mathsf{FLImP}}\right) \leq \sigma_j \leq \sqrt{2}\max\left(\boldsymbol{\sigma}_{\mathsf{FLImP}}\right) \\ 0 & \text{otherwisre} \end{cases} \tag{7.6}$$

where $\boldsymbol{\sigma}_{\mathsf{FLImP}}$ is the list of maximum likelihood Rician fit σ of each included FLImP separation distribution. The parameter $\pi_{R_T j}$ is

$$\pi_{R_T j} = \begin{cases} 1 & \text{if } R_{\mathsf{min}} \leq R_{Tj} \leq R_{\mathsf{max}} \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

Normally the parameters R_{min} is set to 0 nm and R_{max} is set to 60 nm.

Finding the maximum of the probability of parameters given data is non-linear multi-modal optimisation problem. Therefore, the search is performed 800 times, using the uphill simplex algorithm and the maximum value is used as a result. This procedure is repeated 50 times, where each time the data set D is re-sampled by replacement, to obtain 50 estimates of the most likely parameters. During each re-sample the most likely parameters are estimated for different number of Rice mixtures.

Number of Rice Components

After the parameters are estimated for different number of components it has to be decided which is the most optimal number of components, and therefore will determine how many distances are observed. To estimate the most optimal number of components FLImP uses Bayesian information criteria (BIC)

$$BIC = -2\ln\left(P\left(\boldsymbol{w}, \boldsymbol{R_T}, \boldsymbol{\sigma}, N_c|D\right)\right) + k\ln\left(N_d N_b\right) \tag{7.8}$$

where k is the number of free parameters, $k = 3N_c - 1$ (the minus 1 is because the weights of the components $\boldsymbol{w}$ have to sum up to 1). In FLImP an improvement of the BIC value of 10 is considered to be significant. To find how many Rice components should be assumed the BIC value is calculated for each number of components in each of the 50 estimates (as described in the previous section). This results in 50 curves. Then for each number of components it is calculated the fraction of the curves which have significantly better BIC value for that number of components than all the previous number of components. The optimal number of components is assumed to be the highest number of components for which that fraction is at least 50%, and there is no lower number of components for which the fraction is lower than 50%.

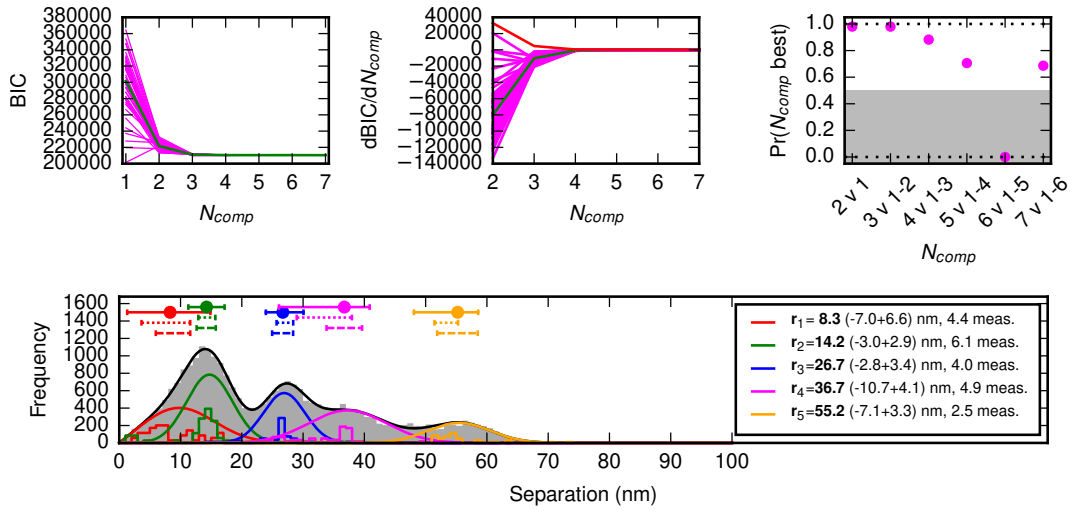


Figure 7.1: BIC curves and Rice mixture model fit for distances extracted from the data set BM-I.

7.3 Results

Samples from 3 to 8 were imaged and 89 data sets were extracted, 47 of which come from samples 3 to 5 and 42 come from samples 6 to 8. The datasets coming from samples 3 to 5 will be called BM-I and the datasets coming from samples 6 to 8 will be called BM-I+PMA. All of the datasets were analysed by Quincy, Section 1.4.2 and Section 1.4.1, and tracks were selected using the method described in Chapter 6. Then all of the tracks were analysed with the FLImP method, described in Section 1.7. Tracks which have confidence interval larger than 10 nm and separation larger than 70 nm were rejected. After filtering there were 35 tracks extracted from the data sets BM-I and 23 tracks extracted from the data sets BM-I+PMA. Then the bootstrap samples of the tracks were combined and a mixture of rice distributions was fit to it.

The top of figure 7.1 shows the BIC curves for different number of components estimated on the data set BM-I. From the fraction of curves which have better BIC value than the previous components can be seen that for this data set the best number of components is 5. Below is shown the Rice mixture model fit with 5 components for

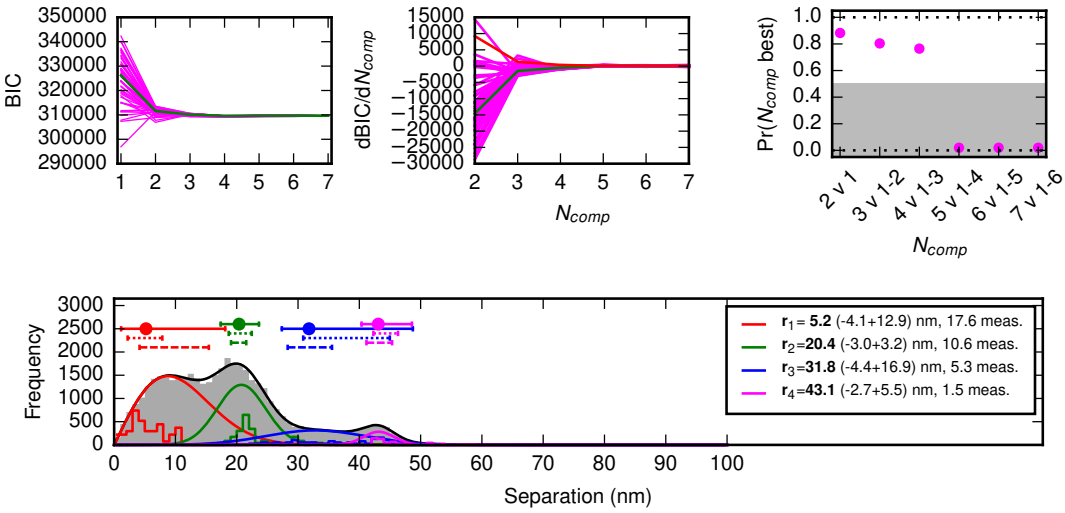


Figure 7.2: BIC curves and Rice mixture model fit for distances extracted from the data set BM-I+PMA.

that data-set. There are 5 peaks all of which seem to be responsible for relevantly few distances.

The first component has a mean of 8.3 nm, a confidence interval of 13.3 nm, and it is composed of about 4.4 distances. The second component has a mean of 14.2 nm, a confidence interval of 6 nm, and it is composed of the most distances, around 6.1. The third component has a mean of 26.7 nm, a confidence interval of 6.2 nm, and is responsible for 4 distances. The fourth component has a mean of 36.7 nm, a confidence interval of 14.8 nm, and is responsible for 4.9 distances. The fifth components has a mean of 55 nm, a confidence interval of 10.4 nm, and it is composed by 2.5 distances.

Figure 7.2 shows the BIC curves for different number of components estimated on the data set BM-I+PMA. From the fraction of curves which have better BIC value than the previous components can be seen that for this data set the best number of components is 4. Below is shown the Rice mixture model fit with 4 components for that data-set.

The first distance has a mean of 5.2 nm, a confidence interval of 17 nm, and it is composed of 17.6 distances. The second component has a mean if 20.4 nm, a confidence

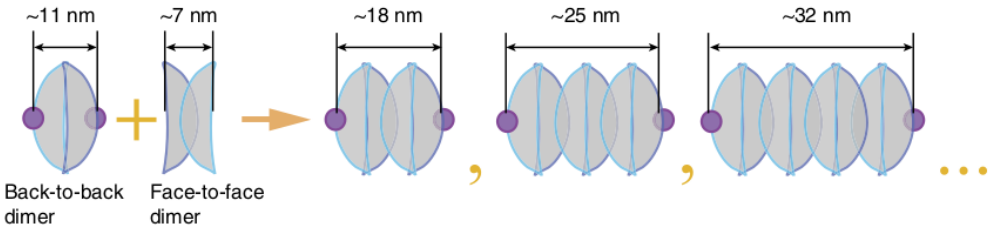


Figure 7.3: Models of hypothetical EGFR oligomers, [22]

interval if 6.2 nm, and it is composed of 10.6 distances. The third component has a mean if 31.8 nm, a confidence interval of 21.3 nm, and it is composed of 5.3 distances. The last component has a mean of 43.1 nm, a confidence interval of 8.2 nm, and it is composed of 1.5 distances.

7.4 Discussion

According to Needham et al. 2016 [22] there are several hypothetical configurations of the oligomers, Figure 7.3. When the size of the dye is added to the model, the sizes will increase with about 2 nm. Therefore the expected observed distances will be 9 nm, 13 nm, 20 nm, 27 nm, and 34 nm. Using this model it is possible to extrapolate even larger distances, such as 41 nm, 48 nm, and 55 nm. These hypothetical sizes assume that the dimers are arranged in a straight line.

Table 7.2 shows the components from the datasets BM-I and BM-I+PMA, associated with the closest hypothetical dimer.

7.4.1 BM-I

The first row of the Table 7.2 shows that there is a component from the BM-I dataset with mean of 8.3 nm however the confidence interval is 13.6 nm. This component is composed of around 4.4 distance. Based on the confidence interval it could be associated

Hypothetical Dimers			BM-I				BM-I+PMA			
Type	№Dimers	Length	S	M	E	№D	S	M	E	№D
?	?	?					1.1	5.2	18.1	17.6
FF	1	9	1.3	8.3	14.9	4.4				
BB	1	13	11.2	14.2	17.1	6.1				
BB	2	20					17.4	20.4	23.6	10.6
BB	3	27	23.9	26.7	30.1	4.0				
BB	4	34	26.0	36.7	40.8	4.9	27.4	31.8	48.7	5.3
BB	5	41					40.4	43.1	48.6	1.5
BB	6	55	48.1	55.2	58.5	2.5				

Table 7.2: This table shows the results from the rice fits and the hypothetical oligomers which they could match. The leftmost column is type dimer which is either face-to-face (FF) or back-to-back (BB). There are 2 sections, one for the dataset BM-I and the other is for the dataset BM-I+PMA. Under each dataset there are 4 columns which represent a Rice components from the fits. The first column is the start of the confidence interval of the component (S), the second column is the mean of the component (M), the third column is the end of the confidence interval (E), and the last column (№D) is how many distances are responsible for that component.

with the face-to-face dimer which would have a size of 9 nm. Also the back-to-back dimer at about 13 nm is within the confidence interval however it is less likely the cause for component. In general the confidence interval of this component is very large and the number of distances it is composed of is small. The confidence interval does not allow to distinguish clearly between the face-to-face and the back-to-back dimer. Therefore, this component is not a strong evidence for a face-to-face dimer.

The second component of the BM-I data set has a mean of 14.3 nm and confidence interval of 4.9 nm. This is consistent with the back-to-back dimer with size of 13 nm. Since the interval is relevantly low and the number of distances is around 6 it could be associated that a back-to-back dimers.

The next two components estimated from the BM-I data set have means of 26.7 nm, and 36.7 nm, with confidence intervals of 6.2 nm, and 6.8 nm, and composed of around 4 and 5 distances. These components can be associated with 3 and 4 dimers arranged in a line. These components have a relevantly small confidence intervals, however are composed of just few distances. Also arrangement of dimers in a line is not very likely.

The last Rice component estimated from this data set is with a mean of 55.2 nm, has a confidence interval if 10.4 nm, and it is composed of just 2.5 distances. Large confidence interval and so few distances make it very unlikely that there is an observed oligomer composed of 6 dimers arranged in a line. It is more likely that these are just some distances measured from fluorophores belonging to different dimers, which are just in proximity to one another and not necessarily part of the same structure.

7.4.2 BM-I+PMA

The component with lowest distance has mean of 5.2 nm, with confidence interval of 17 nm, and composed of 17.6 distances. This mean is lower than the hypothetical 9 nm, however the confidence interval is very large and therefore the distance can be associated with either the 9 nm front-to-front dimer or the 13 nm back-to-back dimer. It is also

possible that there could be distances measured from two side-to-side dimers, which could be shorter than 9 nm. Since this mean is based on about 17 distances it is quite likely that there is an observation in the range between 1.1 nm and 18.1 nm, however the large confidence interval makes it hard to distinguish which dimer is observed.

The second component has a mean of 20.4 nm, with confidence interval of 6.2 nm, and it is composed of 10.6 distances. This measurement and the low confidence interval is consistent with oligomer of 2 back-to-back dimers.

The third component has a mean of 31.8 nm, with confidence interval of 21.3 nm, and it is composed of 5.3 distances. This could be associated with the oligomer composed of 4 and 5 dimers with sizes of 31 nm and 41 nm. Since the large confidence interval and the low number of distances it is unclear what structure caused this measurement.

The last component has a mean if 43.1 nm, a confidence interval of 8.2 nm, and it is composed of 1.5 distances. Since the low number of distances it is unlikely that this component represents an EGFR structure in the cell. It is possible that this is just a random occurrence or to fit some noise from the rest of the measurements.

7.5 Conclusion

The measured distances for the data set BM-I of 14.3 nm, 26.7 nm, and 36.7 nm, and the measured distance from BM-I+PMA of 20.4 nm seem to represent a repeatable structures. According to data coming from electron microscopy [168] there are larger structures, however they are not arranged in a line. The data from the electron microscope shows distances of 32 nm, 36 nm, 40 nm, 48 nm, 50 nm, and 56 nm. These distances cannot be directly associated with the distances measured from the FLImP data, because the labelling methods are different. However, the distances estimated in this experiment show the possible existence of higher order oligomers, but the linear models presented in [22] are unlikely, since there were no EGFR arranged in a line in

the electron microscopy data. Another implication is that when PMA was added to the experiment more shorter distances were recorded.

The data used in this experiment has too few data points to be able to draw meaningful conclusions. Also the model assumes that there will be only repeatable distances such as dimers or larger oligomers. However if there is no dimerisation in the cells it is likely that the estimated separations will based on random monomers. Therefore, in such case it will be expected that the distances will follow random distribution. Another possibility is that there will be some number of distances which are measured from fluorophores which are in different dimers in proximity. This will introduce additional randomness and increase confidence intervals. It is also possible that there is equilibrium between dimerised and monomeric state of the EGFR [169], or periodicity in the ratio of dimers and monomers [170]. Therefore the model might benefit from including possibility for a uniformly distributed component, to account for random distances which are not estimated from a repeatable structure.

While some evidence exists that clusters of EGFR exist, and the PMA causes smaller structures to form, the experiment will require larger datasets and improvements to the model to bring more reliable information about effects of the PMA.

Chapter 8

Discussion

The super-resolution methods from the SHRImP family, which also includes FLImP, can measure very small separations, such as 9nm in fixed cells with confidence interval of 7nm. This gives them an advantage over other methods such as STORM, which provides a resolution of 20nm [43]. The higher accuracy is achieved by using large number of frames during which the fluorophores are always in emitting state. This provides much more measurements than STORM, where each emitter is in a fluorescent state in just a few frames.

However, methods such as FLImP and SHRImP are susceptible to erroneous measurements due to the large number of assumptions made by the model of the diffraction limited spot. Therefore, it is critical to select the correct data for analysis. When imaging live cells, pitfalls such as fluorophore depth difference, incorrectly identified levels, areas of high crowding, and non-uniform background fluorescence need to be taken into account. This project developed a semi-automated track selection process, which reduced the volume of the data for manual inspection around 70 times and greatly simplified the manual inspection criteria. The results showed that the refined process reduced the error rate of the methods, thus making them more reliable.

The developments from this thesis can be used for automation of other techniques

of the SHRImP family which use manual data processing steps [54] [53].

8.1 Simulations

Since there are many unknowns when imaging molecular complexes on the membrane of fixed cells, simulations are used to verify the correctness of the algorithms developed during the project and also as controls for the experiments. The simulations used in this project, documented in Chapter 3, allow the user to specify 2D fluorophore positions and their intensities. The fluorophore transitions is modelled by a Markov chain in order to generate a sequence of frames. Then the EMCCD noise model is used to generate image counts for each pixel.

The PSF model used in the simulations is a Gaussian function, which is different from the theoretical Airy function. The Gaussian function is good enough approximation for the purposes of the simulations in this project. It is also used for other simulations [140] and due to its simplicity is preferred over the Airy function.

The EMCCD model developed by Hirsh et al. [160] was chosen because of its good theoretical basis and completeness. It is a fully probabilistic model of the processes in the CCD matrix, the electron multiplier register, and the readout noise. The model also allows to adjust all of the parameters of the microscope sensor according to the technical specifications. A downside of the model is that it is hard for integration, and therefore computationally expensive numerical methods are used. Also it contains Poisson components, which means that when simulating complicated fluorophore configuration each image pixel is drawn from different distribution, consequently slowing down the simulations even further.

The simulations used in this project have very detailed noise model, which is an improvement over other simulations which use only Poisson noise [40] [171] or a combination between Poisson and Gaussian noise [147] [140]. A simulation used by Spieser et

al. for training neural networks also uses a realistic noise model for the EMCCD camera, which takes into account more details of the imaging process [128]. Other simulations have more sophisticated PSF models than the one used in this project, such as the Airy function or other PSFs used for 3D localisation [147].

The simulations are realistic enough for the most of the applications within this project, however, as shown in Chapter 4, there are limitations. It is possible to improve on the current simulations by using more accurate PSF model. Another area of improvement is to use realistic transition probabilities for the Markov chain which models the fluorophore behaviour. The simulations also don't take into account the effects of förster resonance energy transfer when there is a crowding of fluorophores.

8.2 Level Detection

The level detection algorithm used in FLImP was prone to errors, as demonstrated in Chapter 1, which necessitates the development of new algorithm for successful automation of the track selection. Chapter 4 presents two new developments of a level detection algorithm, "Level Noise" and "Neural Network", and compares them with the existing level detection algorithm, called "Bayesian".

The "Bayesian" algorithm uses a probabilistic approach to test for changes within traces of intensities of diffraction limited spots. This algorithm makes several very restrictive assumptions, such as Gaussian distributed intensities, which would fail when the intensity of the emitters is low. Other limitations are that it relies on the estimates of Quincy, which assumes uniform background, and this assumption frequently fails.

These issues are addressed by the algorithm "Level Noise", which is more flexible in terms of assumptions. The algorithm relies on the idea that the variance within a constant intensity level should remain the same, unless a change of the emitter states occurs. It also uses KS tests to ensure that the intensities within a level come from the

same distribution. The algorithm also tests for gradients within the levels, which can be an indication for background fluorescence.

The third approach, "Neural Network" uses neural network trained on simulated data to detect the level transitions.

The three algorithms were compared on simulated data and on real data. The "Neural Network" had best performance when tested on simulated data, however when tested on real data the "Levels Noise" algorithm outperformed the rest. A possible reason for the discrepancy between the results obtained from the simulated and the real data could be the simplicity of the simulations. The neural network was trained on simulated data, however the real data follows more complex distribution and therefore the algorithm performed poorly.

The level detection algorithm is self-contained and can be adapted to work for other single-molecule studies, where flexibility is required. Other developments, such as those by Watkins and Yang [172] and Andrec et al. [173], use probabilistic models to determine the most likely number of levels and transition points. However, these approaches are designed to work with Time-Correlated Single Photon Counting. Another approach provided by gSHRImP tries to find photobleaching steps by analysing the diffraction limited images directly [40].

8.3 Track Classification

The reliability of the manual track selection process was evaluated in Chapter 5 by inspecting different properties of tracks which were used in FLImP studies for previously published papers. The results showed that very high percentage of the manually selected tracks are unfit for analysis. This claim was further reinforced by a classifier which managed to correctly classify simulated tracks, but failed to distinguish between selected and rejected tracks.

The manual track selection process needs to account for all of the FLImP assumptions, which makes it very complex and difficult to follow. In addition the human judgement is subjective when compared to statistical tests. For example, when a track is extrapolated, is the intensity zero in the region after the last frame? If several people look at a plot, they might come to different conclusions due to their biases. Also the judgement of the same person varies based on the mental state, tiredness and so on. FLImP experiments required inspection of very large number of intensity traces. Chapter 6 showed that just for a few experiments around 32000 tracks were inspected. Completing such a task without loosing concentration and quality of judgement is for a person extremely difficult, therefore it is natural for mistakes to start slipping in.

The results from the experiments presented in Chapter 5 show that automation of the FLImP tool-chain is critically important for a successful study.

8.4 Refined Track Selection and Applications

Chapter 6 presented a refined and automated track selection process. The correctness of the process was verified by simulated data. The efficiency of the refined process was compared to the old manual process on already analysed FLImP data in terms of number of manually inspected tracks and number of useful tracks. The results showed that the refined process requires around 70 times fewer tracks to be inspected manually. Also the manual inspection is greatly simplified which reduces the possibility for error.

Later the refined track selection process was used to analyse novel EGFR data, Chapter 7. The results presented in that chapter suggested that the EGFR oligomerisation is influenced by phorbolmyristate acetate.

Chapter 9

Conclusion

The project described in this thesis aims to automate the track selection process. Automation of that stage of FLImP will improve the time and cost efficiency of the method, as human involvement will be reduced. The track selection process with reduced human involvement will be less prone to errors, which will cause the final results to have shorter confidence intervals, and therefore improved resolution. Other benefits of the automation is that the FLImP method will require less staff training and using it will be easier.

In Chapter 4 two novel algorithms were presented, which find areas with constant intensity in time-series (Level Detection). The novel algorithms, "Neural Network", and "Levels Noise", were compared to the existing level detection algorithm, called "Bayesian". The comparison of the algorithms' performance on real and simulated tracks showed that the best performing algorithm was "Levels Noise", which was adopted through the rest of the thesis and is currently deployed as part of the FLImP pipeline.

In Chapter 5 the FLImP tracks were classified into two categories, used and unused. The tracks were converted into feature vectors using domain specific representation, and a neural network was used for the classification. The classifier and the representation were verified by showing that they classified simulated tracks correctly, based on the

track selection requirements. However, when manually selected tracks from 2064 FLImP experiments were classified, the classifier was unable to distinguish between the tracks selected for analysis and the tracks which were rejected. The results from the experiments and inspection of the track selections showed that there are critical deficiencies in the manual track selection process. The results from this chapter showed the necessity of improved and automated track selection process.

Chapter 6 presented an automated track selections process. The refined process has several steps:

- **Background Assessment and Level Detection.** At this stage, diffraction limited spots which have non-uniform background fluorescence in the neighbourhood are excluded from analysis and regions with constant intensity, called levels, are detected.
- **Track Structure Identification.** Tracks which do not have suitable intensity pattern or good level ratio are rejected.
- **Track Assessment** Properties of the tracks which were selected during the previous two steps are calculated and used to classify tracks into those which are suitable for the biological study and those which are not.
- **Manual Inspection** At this step, several criteria are used to reject tracks which have passed the previous steps but are not considered analysable.

The validity of the refined track selection process was verified by simulations. The new process was also compared to its predecessor, and it reduced the time required for data selection around 70 times, increased the number of tracks extracted from the data sets by 11.3%, and reduced the FLImP analysis time by 8.6%. Also, the confidence intervals were on average shorter.

The results presented in this thesis showed that manually selecting single molecule imaging data is unreliable and inefficient. Manual selection is not optimal for inspecting

large volumes of data, because people become tired and prone to mistakes. Therefore, automation should be used whenever it is possible. I have demonstrated an approach which is a viable and automated alternative.

The track selection process described in this thesis can be used in other single molecule methods such as SHRImP [53]. Also, the level detection algorithm can be used in other studies as well. In this project it is used for intensity traces, however it can be easily adapted to any time-series. Therefore, it can be used in different single-molecule studies such as [174] [175] [176] [177].

9.1 Future Development

The new data selection process has a limitation that it has manual steps. Further developments on the FLImP analysis process can seek to fully automate the data selection process to eliminate errors made during manual selection and reduce the analysis time further.

Another area where the selection process can be improved is the track structure identification. In the refined process, the intensity levels should follow one of two patterns. The first pattern is high intensity, followed by low intensity, followed by zero intensity, and the second pattern is zero intensity, followed by low intensity, followed by high intensity. It is possible to use the probability of the fluorophore to change between different states to estimate the most likely number of fluorophores in each level, based on the observed intensity. This would allow FLImP to analyse larger variety of intensity patterns and hence increase the number of analysable tracks.

FLImP requires the fluorophores which generate an analysed track to be at the same depth in the sample, because the measured distance is the projection of the real distance on the sample surface. This problem is addressed in the refined process by using a threshold on the ratio of the intensity of the fluorophores to reject tracks where the

depth difference would skew the result. This threshold depends on the application of FLImP and the experimental conditions.

Appendix A

Track Properties

Part of the FLImP analysis is track construction and track selection, which involves identification of frame ranges, in which the track has constant intensity, called levels, and calculating track properties.

A level is constructed of several regions with constant intensity. The set of frame indexes which are part of a level n with K segments is

$$ln = \{t | t \in [\mathsf{start}_k, \mathsf{end}_k], k \in [1, K]\} \tag{A.1}$$

Based on the set of frame indexes, which are part of a level n, some other abbreviations are used

$$I(ln) = \{I_t | t \in ln\} \tag{A.2}$$

where I_t is the intensity of the diffraction limited spot at frame t,

$$x(ln) = \{x_t | t \in ln\} \tag{A.3}$$

where x_t is the x position of the diffraction limited spot at frame t,

$$y(ln) = \{y_t | t \in ln\} \tag{A.4}$$

where y_t is the y position of the diffraction limited spot at frame t, and

$$bg(ln) = \{bg_t | t \in ln\} \tag{A.5}$$

where bg is the background intensity around in the neighbourhood of the diffraction limited spot at frame t.

Another useful notation is the set of (x, y) positions, $\boldsymbol{p} \in \mathbb{R}^2$, in a level, n is

$$p(ln) = \{\boldsymbol{p}_t | t \in ln\} \tag{A.6}$$

The mean of the intensity in level n is defined as

$$\langle I(ln) \rangle = \frac{1}{T(ln)} \sum_{t \in ln} I_t \tag{A.7}$$

and the standard deviation of level n is defined as

$$\sigma(I(ln))^2 = \frac{1}{T(ln) - 1} \sum_{t \in ln} (I_t - \langle l1 \rangle)^2 \tag{A.8}$$

where $T(ln) = |l1|$ is the number of frames in *level n*.

Similarly the mean position in level n is defined as

$$\langle \boldsymbol{p}(ln) \rangle = \frac{1}{T(ln)} \sum_{t \in ln} \boldsymbol{p}_t \tag{A.9}$$

and the standard deviation of the position in the same level is defined as

$$\sigma(\boldsymbol{p}(ln)) = \sqrt{\frac{1}{n} \sum_{t \in ln} \left(\boldsymbol{p}_t - \langle \boldsymbol{p} \rangle_{t \in ln}\right)^T \left(\boldsymbol{p}_t - \langle \boldsymbol{p} \rangle_{t \in ln}\right)} \tag{A.10}$$

Some descriptors involve splitting the region into small segments, called windows, and calculating some properties over these windows. The windows in a level n, which has K regions, are 11 frames long and are centred around frame w, where

$$w \in [\mathsf{start}_k + 5, \mathsf{end}_k - 5], \ k \in [1, K] \tag{A.11}$$

If the mean value of the window is calculated, it is called local mean, μloc, and if the standard deviation is calculated, it is called local standard deviation, σloc. The local mean calculated for the intensity of level n is

$$\mu loc(I(ln))_w = \frac{1}{11}\sum_{i=-5}^{5} I_{w+i} \tag{A.12}$$

and the local standard deviation of the intensity calculated over the same region is

$$\sigma loc(I(ln))_w = \sqrt{\frac{1}{11}\sum_{i=-5}^{5}\left(I_{w+i} - \mu loc(I(ln))_w\right)^2} \tag{A.13}$$

Similar formulas are used for the local mean and local standard deviation of the position.

$$\mu loc(p(ln))_w = \frac{1}{11}\sum_{i=-5}^{5} \boldsymbol{p}_{w+i} \tag{A.14}$$

$$\begin{aligned} \sigma loc(p(ln))_w &= \sqrt{\frac{1}{11}\sum_{i=-5}^{5} \boldsymbol{d}_i^T \boldsymbol{d}_i}, \\ \boldsymbol{d}_i &= (p_{w+i} - \mu loc(I(ln))_w) \end{aligned} \tag{A.15}$$

- **background_l1**, **background_l2**, **background** Background homogeneity for *level 1*, *level 2*, or the entire track. For each frame t in the corresponding frame interval

  1. Create a vector, $\mathcal{I}$, with the intensity of each pixel, which has distance to the centre of the feature in the range $[3.5\sigma_{PSF} - \sqrt{2}, 5.5\sigma_{PSF} + \sqrt{2}]$, where σ_{PSF} is the standard deviation of the Gaussian distribution, used to model the diffraction limited spot.
  2. Split the vector into quadrants $\mathcal{J}_i$, $i = 1..4$.
  3. Calculate $\mathcal{B}_t = 4\frac{\sigma(\mathcal{I})}{\sum_{i=1}^{4}\sigma_{\mathcal{J}_i}}$

4. Return the mean of the 10 largest values for $\mathcal{B}_t$.

- **feature_shape_l1**, **feature_shape_l2** Measures the roundness of the feature in *level 1* and *level 2*. For each frame t in the corresponding level

  1. Crop area of 5 pixels in each direction around the centre, creating image, img of 11 by 11 pixels with intensity at pixel with coordinates x, and y of $img(x, y)$, where $x \in \{-5..5\}$, and $y \in \{-5..5\}$.
  2. Create set of points in $\mathbb{R}^2$, $D = \{[x, y] img(x, y) | x \in \{-5..5\}, y \in \{-5..5\}\}$
  3. Calculate the ratio of eigenvalues λ_1, and λ_2, where $\lambda_1 > \lambda_2$ of co-variance matrix of the set D.
  4. Calculate the value $r_t = \frac{\lambda_1}{\lambda_2}$

  Then sort the ratios r_t and find the mean of the 10% largest values.

- **fragmentation_l1** num frame ranges in $\text{l1} - 1$
- **fragmentation_l2** num frame ranges in $\text{l2} - 1$
- **longest_range_l1** Number of frames in the longest segment in *level 1*.
- **longest_range_l2** Number of frames in the longest segment in *level 2*.
- **level1points** Number of detected frames *level 1*.
- **level2points** Number of detected frames *level 2*.
- **ratio_real_points_l1** $\frac{\text{Number of detected frames level 1}}{\text{Number of frames in level 1}}$
- **ratio_real_points_l2** $\frac{\text{Number of detected frames level 2}}{\text{Number of frames in level 2}}$
- **ratio_real_points** $\frac{\text{Number of detected frames}}{\text{Number of frames}}$
- **sec_longest** Number of frames in the shortest of *level 1* and *level 2*.
- **sec_most_pts** Number of detected frames in the shortest of *level 1* and *level 2*.
- **shortest_range** Number of frames in the shortest level segment.

- **leveliness** $\frac{\text{num frames assigned to levels}}{\text{duration}}$
- **I12_gap** The length of a range ρ_1.

  If the track starts with *level 1*, the range ρ_1 ends at the first frame of the first range of *level 2* and starts at the last frame of the preceding range. If tracks are filtered by the automatic track rejection, described in Section 5.4.1, this range is part of *level 1*.

  If the track ends with *level 1*, the range ρ_1 starts at the last frame of the last range of *level 2* and ends at the first frame of the following frame range. If tracks are filtered by the automatic track rejection, described in Section 5.4.1, this range is part of *level 1*.
- **frames_before_after_I1** The length of range ρ_2.

  If the track starts with *level 1*, the range ρ_2 starts at the first frame of the track and ends at the first frame of the first range of *level 1*.

  If the track ends with *level 1*, the range ρ_2 starts at the last frame of the last range of *level 1* and ends at the last frame of the track.

Intensity

- **goes_to_zero** Define a set of frames ρ_3. If the track starts with *level 1*, then ρ_3 ends at the first frame of the track and starts 10 frames earlier or at the first frame of the experiment, depending which is later.

  If the track ends with *level 1*, then ρ_3 starts at the last frame of the track and ends 10 frames later or at the last frame of the experiment, depending which is earlier.

  The property is the ratio of the mean extrapolated intensity in the interval ρ_3 and the standard deviation, $\frac{\langle I(\rho_3)\rangle}{\sigma_{I(\rho_3)}}$.

- **l12_off_level** Define a set

$$m = \left\{ \min_{i \in \{1,2\}} \frac{|I_t - \langle I(li) \rangle|}{\sigma_{I(li)}} \middle| t \in \rho_1 \right\}$$

  where ρ_1 is the frame range as defined by the property *l12_gap*. Then calculate the minimum, maximum, and mean value of the set m – $\min(m)$, $\max(m)$, $\mu(m)$

- **int_before_after_l1** Define a set

$$m = \left\{ \frac{|I_t - \langle I(l1) \rangle|}{\sigma_{I(l1)}} \middle| t \in \rho_2 \right\}$$

  where ρ_2 is a frame range as defined by the property *frames_before_after_l1*. Then calculate the minimum, maximum, and mean value of the set m – $\min(m)$, $\max(m)$, $\mu(m)$.

- **signal_to_noise_l1** Ratio of the mean intensity to standard deviation of the intensity of *level 1*, $\frac{\langle I(l1) \rangle}{\sigma_{I(l1)}}$

- **signal_to_noise_l2** Ratio of the mean intensity to standard deviation of the intensity of *level 2*, $\frac{\langle I(l2) \rangle}{\sigma_{I(l2)}}$

- **signal_lvl1** The sum of all intensities in *level 1*, $\sum_{i \in l1} I_i$.

- **signal_lvl2** The sum of all intensities in *level 2*, $\sum_{i \in l2} I_i$

- **clustering** Measures separation between intensity of *level 1* and *level 2*, $\frac{\langle I(l2) \rangle - \langle I(l1) \rangle}{\sigma_{I(l2)} + \sigma_{I(l1)}}$

- **count_sigma_mls_12** $\max_{i \in \{1,2\}} \frac{\sigma_{I(li)}}{\langle \sigma loc(I(li)) \rangle}$

- **cts_loc_sn_l1** Mean local signal-to-noise ratio, $\left\langle \frac{\mu loc(I(l1))_w}{\sigma loc(I(l1))_w} \right\rangle$

- **cts_loc_sn_l2** Mean local signal-to-noise ratio, $\left\langle \frac{\mu loc(I(l2))_w}{\sigma loc(I(l2))_w} \right\rangle$

- **cts_slm_l1** Standard deviation of the local mean for *level 1*, $\sigma_{\mu loc(I(l1))}$

- **cts_slm_l2** Standard deviation of the local mean for *level 2*, $\sigma_{\mu loc(I(l2))}$

- **stepratio_lvl12** $\left| \frac{\langle I(l2) \rangle}{\langle I(l1) \rangle} - 2 \right|$

- **meanovermlsig** Ratio of the mean intensity and mean local standard deviation $\frac{\langle I \rangle}{\langle \sigma loc(I) \rangle}$.
- **sigovermlsig** $\frac{\sigma(I)}{\langle \sigma loc(I) \rangle}$
- **maxstepfrac** Ratio of the largest step to the maximum intensity, $\frac{\delta I_{max}}{I_{max}}$
- **levelrchisq** $\frac{1}{TL-3nlevels} \sum_{i=1}^{NL} \sigma\left(I(li)\right)$, where NL is the number of levels, and TL is the number of frames assigned to a level.

Position

- **displacement** Largest distance to the mean position of the track, $\max_{t=1..T} \sqrt{\left(x_t - \langle x \rangle\right)^2 + (y_t - \langle$
- **dr12** Distance between mean position of *level 1* and *level 2*,

$$\sqrt{\left(\langle x(l1) \rangle - \langle x(l2) \rangle\right)^2 + \left(\langle y(l1) \rangle - \langle y(l2) \rangle\right)^2}$$

- **pos_density_12** Number of detected features in *level 1* and *level 2* divided by the area, which they occupy, $\frac{level1points+level2points}{occupiedarea}$.
- **pos_error_1** Standard error of mean position of *level 1*

$$\frac{1}{\sqrt{T(l1)}} \sqrt{\frac{1}{T(l1)} \sum_{t=l1_{start}}^{l2_{end}} (x_t - \langle x \rangle)^2 + (y_t - \langle y \rangle)^2}$$

  where $T(l1)$ is the length of *level 1*, $l1_{start}$ is the first frame of *level 1*, and $l1_{end}$ is the last frame of *level 1*.

- **pos_error_2** Standard error of mean position of *level 2*.

$$\frac{1}{\sqrt{T(l2)}} \sqrt{\frac{1}{T(l2)} \sum_{t=l2_{start}}^{l2_{end}} (x_t - \langle x \rangle)^2 + (y_t - \langle y \rangle)^2}$$

  where $T(l2)$ is the length of *level 2*, $l2_{start}$ is the first frame of *level 2*, and $l2_{end}$ is the last frame of *level 2*.

- **pos_sigma_mls_12** The largest between the ratio of the standard deviation and mean local standard deviation of the positions in *level 1* and *level 2*, $\max_{i\in\{1,2\}} \frac{\sigma(p(li))}{\langle \sigma loc(p(li))\rangle}$.
- **pos_sigma_locmean_12** The largest between the standard deviation of the sequence of local mean position in *level 1* and *level 2*, $\max_{i\in\{1,2\}} \sigma\left(\mu loc(p(li))\right)$.
- **semdr12** Standard error in mean of difference between position of *level 1* and position of *level 2*

$$\frac{\sqrt{\left(\langle x(l1)\rangle - \langle x(l2)\rangle\right)^2 \left(\frac{\sigma^2_{x(l1)}}{T(l1)} + \frac{\sigma^2_{x(l2)}}{T(l2)}\right) + \left(\langle y(l1)\rangle - \langle y(l2)\rangle\right)^2 \left(\frac{\sigma^2_{y(l1)}}{T(l1)} + \frac{\sigma^2_{y(l2)}}{T(l2)}\right)}}{\mathrm{dr12}}$$

  where $T(l1)$ is the length of *level 1* and $T(l2)$ is the length of *level 2*.

Appendix B

FLImP Experimental Conditions

The experimental conditions, targeted receptors, mutations, and Ligands for the FLImP experiments used to train a classifier in Chapter 5.

Receptor	Mutation	Ligand	Concentration	Treatment	Comment
HER1	CC EGFR	EGF	4 nM		EGF is epidermal growth factor which is a ligand for EGFR and has the fluorophore attached (so distance is between two EGF's on different receptors).

HER1	L680N	Affibody	4 nM		HER1 Affibody binds to the receptor at domain 3 but does not activate it.
HER1	WT EGFR	EGF	10 uM		
HER1	WT EGFR	EGF	20 uM		
HER1	WT EGFR	EGF	4 nM	Bisindoly-maleimide, Phorbol Myristate Acetate	Bisindoly-maleimide inhibits protein kinase C and phorbal myristate acetate is an activator of PKC.

HER1	WT EGFR	EGF	4 nM	PH	The PH domain binds to PI(4, 5)P2 (a negatively charged signalling lipid) in the inner membrane leaflet and sequesters it, preventing EGFR from forming a complex with it and altering its structure (as well as other membrane processes)

HER1	WT EGFR	EgB4 nanobody	4 nM	9G8 nanobody	9G8 nanobody keeps the receptor in a bent (tethered) conformation. The fluorphore is on EgB4 nanobody (to measure the distance between domain 1s of the receptor).
HER2	HER2	HER2 Affibody	4 nM		
HER1	Delta C	EGF	4 nM		
HER1	WT EGFR	EGF	4 nM	Bisindoly-maleimide	Bisindoly-maleimide inhibits protein kinase C
HER1	WT EGFR	Affibody	100 nM		

HER1	WT EGFR	Affibody	4 nM	Erlotinib	The cells are treated with the erlotinib drug which may change the structure of the receptor.
HER1	WT EGFR	EgB4 nanobody	4nM	EGF	The receptor is stimulated with EGF (to activate it)
HER1	WT EGFR	EGF	400 nM		
HER1	C'973	Affibody	4 nM		
HER1	L8585R	Affibody	4 nM		
HER1	WT EGFR	Affibody	4 nM		A reduced % of paraformalde-hyde was used to prove the fixative was not effecting results
HER1	K618V	EGF	4 nM		

HER1	WT EGFR	EGF	4 nM	Methyl Beta Cyclodextrin	Methyl beta cyclodextrin is used to deplete cholesterol from the cell membrane.
HER1	K618V	Affibody	4 nM		
HER1	WTEGFR	EGF	30 nM		
HER1	L858R and T790M	Affibody	4 nM		
HER1	WT EGFR	EGF	4 nM	Erlotinib	The cells have been treated with the erlotinib drug which may change the structure
HER1	WT EGFR	Affibody	4 nM	Methyl Beta Cyclodextrin	Methyl beta cyclodectrin is used to deplete cholesterol from the cell membrane.
HER1	C'973	EGF	4nM		

HER1	WT EGFR	Affibody	4 nM	PH	The PH domain inserts a wedge so the intracellular region of the receptor can't bind to the membrane.
HER1	K721A	EGF	4 nM		
HER1	WT EGFR	Affibody	4 nM	Labelled at 37C	Labelled at 37C instead of 4C
HER1	Delta C	Affibody	4 nM		
HER1	WT EGFR	Affibody	4 nM	"Labelled at room Temperature"	Labelled at room temp instead of 4C
HER1	K721A	Affibody	4 nM		
HER1	L680N	EGF	4 nM		
HER1	WT EGFR	Affibody	4 nM	9G8 nanobody	9G8 nanobody keeps the receptor in a bent (tethered) conformation.
HER1	WT EGFR	EGF	100 nM		
HER1	WT EGFR	EGF			

HER1	C'698	EGF	4 nM		
HER1	WT EGFR	Affibody	4 nM		

Appendix C

Code Repositories

Code relevant to the thesis can be found in the repositories below:

- Wolfram code. This repository contains the code from the experiments and simulations in Chapter 3, Chapter 4, Chapter 5, Chapter 6. It contains several directories described below:
  - *background*. The code used in the experiments for the automatic background fluorescence detection, Section 6.2.
  - *classify-smi*. The code used for classification of tracks, Chapter 5.
  - *level-detection*. Part of the level detection implementation and comparison. Used in Chapter 4.
  - *simulations* These are the simulations used in Chapter 4, Chapter 5, and Chapter 6. The simulations are described in Chapter 3.
  - *track-selection* This code is used for the experiments in Section 6.4.
  - *filter-comparison* Compares the refined track selection process with the manual track selection process, Section 6.7
  - *common* Code that is shared by many experiments.

  - *retired* Code which was used as a proof of concept for some ideas that did not make it into the thesis.

  This repository uses data which is not uploaded to GitHub, because it is around 150GB, however it is available at request. When requested a file named *code_data.tar.gz* will be provided. Then extract the file:

```
$ cd $project_root
$ git clone https://github.com/teodorvb/phd-wolfram-code.git code
$ tar -xf code_data.tar.gz
```

- Track Import. Contains scripts for importing tracks from the FLImP experiments into a database which is used by this project.
- Deployment Libraries Contains deployment libraries for the FLImP filter. This repository is private. To gain access please contact the Octopus Group.
- MSMMBayesian. Contains the implementation of the FLImP method. Part of the repository is the implementation of the automated track filter. This repository is private. To gain access please contact the Octopus Group.
- Track Filter and Level Detection Implementation of the track filter and the level detection algorithm. Contains files which are part of the MSMMBayesian codebase.

Data relevant to the project is available at request:

- *code_data.tar.gz* Data and simulations used by the wolfram code.
- *final_database.tar.gz* PostgreSQL database dump of tracks and simulations used in the thesis.
- *others.tar.gz* Other files such as tracks selections used for filter comparison and data analysis.

Bibliography

[1] Laura C Zanetti-Domingues, Michael Hirsch, Christopher J Tynan, Daniel J Rolfe, Teodor V Boyadzhiev, Kathrin M Scherer, David T Clarke, Marisa L Martin-Fernandez, and Sarah R Needham. Determining the geometry of oligomers of the human epidermal growth factor family on cells with 7 nm resolution. *Progress in biophysics and molecular biology*, 118(3):139–152, 2015.

[2] Sarah R Needham, Michael Hirsch, Daniel J Rolfe, David T Clarke, Laura C Zanetti-Domingues, Richard Wareham, and Marisa L Martin-Fernandez. Measuring egfr separations on cells with 10 nm resolution via fluorophore localization imaging with photobleaching. *PloS one*, 8(5):e62331, 2013.

[3] Ariel Lipson, Stephen G Lipson, and Henry Lipson. *Optical physics*. Cambridge University Press, 2010.

[4] Mark A Lemmon and Joseph Schlessinger. Cell signaling by receptor tyrosine kinases. *Cell*, 141(7):1117–1134, 2010.

[5] Sean P Kennedy, Jordan F Hastings, Jeremy ZR Han, and David R Croucher. The under-appreciated promiscuity of the epidermal growth factor receptor family. *Frontiers in cell and developmental biology*, 4:88, 2016.

[6] Thomas PJ Garrett, Neil M McKern, Meizhen Lou, Thomas C Elleman, Timothy E Adams, George O Lovrecz, Hong-Jian Zhu, Francesca Walker, Morry J

Frenkel, Peter A Hoyne, et al. Crystal structure of a truncated epidermal growth factor receptor extracellular domain bound to transforming growth factor α. *Cell*, 110(6):763–773, 2002.

[7] Hideo Ogiso, Ryuichiro Ishitani, Osamu Nureki, Shuya Fukai, Mari Yamanaka, Jae-Hoon Kim, Kazuki Saito, Ayako Sakamoto, Mio Inoue, Mikako Shirouzu, et al. Crystal structure of the complex of human epidermal growth factor and receptor extracellular domains. *Cell*, 110(6):775–787, 2002.

[8] Xuewu Zhang, Jodi Gureasko, Kui Shen, Philip A Cole, and John Kuriyan. An allosteric mechanism for activation of the kinase domain of epidermal growth factor receptor. *Cell*, 125(6):1137–1149, 2006.

[9] Diego Alvarado, Daryl E Klein, and Mark A Lemmon. Structural basis for negative cooperativity in growth factor binding to an egf receptor. *Cell*, 142(4):568–579, 2010.

[10] Monilola A Olayioye, Richard M Neve, Heidi A Lane, and Nancy E Hynes. The erbb signaling network: receptor heterodimerization in development and cancer. *The EMBO journal*, 19(13):3159–3167, 2000.

[11] Nicola Normanno, Antonella De Luca, Caterina Bianco, Luigi Strizzi, Mario Mancino, Monica R Maiello, Adele Carotenuto, Gianfranco De Feo, Francesco Caponigro, and David S Salomon. Epidermal growth factor receptor (egfr) signaling in cancer. *Gene*, 366(1):2–16, 2006.

[12] Parthasarathy Seshacharyulu, Moorthy P Ponnusamy, Dhanya Haridas, Maneesh Jain, Apar K Ganti, and Surinder K Batra. Targeting the egfr signaling pathway in cancer therapy. *Expert opinion on therapeutic targets*, 16(1):15–31, 2012.

[13] Alexander McPherson. Introduction to protein crystallization. *Methods*, 34(3):254–265, 2004.

[14] Mohammed Kaplan, Siddarth Narasimhan, Cecilia de Heus, Deni Mance, Sander van Doorn, Klaartje Houben, Dušan Popov-Čeleketić, Reinier Damman, Eugene A Katrukha, Purvi Jain, et al. Egfr dynamics change during activation in native membranes as revealed by nmr. *Cell*, 167(5):1241–1251, 2016.

[15] Nelson P Barrera and Carol V Robinson. Advances in the mass spectrometry of membrane proteins: from individual proteins to intact complexes. *Annual review of biochemistry*, 80:247–271, 2011.

[16] Rebecca F Thompson, Matt Walker, C Alistair Siebert, Stephen P Muench, and Neil A Ranson. An introduction to sample preparation and imaging by cryo-electron microscopy for structural biology. *Methods*, 100:3–15, 2016.

[17] Andrew HA Clayton, Maria L Tavarnesi, and Terrance G Johns. Unligated epidermal growth factor receptor forms higher order oligomers within microclusters on a431 cells that are sensitive to tyrosine kinase inhibitor binding. *Biochemistry*, 46(15):4589–4597, 2007.

[18] Saveez Saffarian, Yu Li, Elliot L Elson, and Linda J Pike. Oligomerization of the egf receptor investigated by live cell fluorescence intensity distribution analysis. *Biophysical journal*, 93(3):1021–1031, 2007.

[19] Abedelnasser Abulrob, Zhengfang Lu, Ewa Baumann, Dusan Vobornik, Rod Taylor, Danica Stanimirovic, and Linda J Johnston. Nanoscale imaging of epidermal growth factor receptor clustering effects of inhibitors. *Journal of Biological Chemistry*, 285(5):3145–3156, 2010.

[20] Nicholas Ariotti, Hong Liang, Yufei Xu, Yueqiang Zhang, Yoshiya Yonekubo, Kerry Inder, Guangwei Du, Robert G Parton, John F Hancock, and Sarah J Plowman. Epidermal growth factor receptor activation remodels the plasma membrane lipid

environment to induce nanocluster formation. *Molecular and cellular biology*, 30(15):3795–3804, 2010.

[21] Laura C Zanetti-Domingues, Dimitrios Korovesis, Sarah R Needham, Christopher J Tynan, Shiori Sagawa, Selene K Roberts, Antonija Kuzmanic, Elena Ortiz-Zapater, Purvi Jain, Rob C Roovers, et al. The architecture of egfr's basal complexes reveals autoinhibition mechanisms in dimers and oligomers. *Nature communications*, 9(1):4325, 2018.

[22] Sarah R Needham, Selene K Roberts, Anton Arkhipov, Venkatesh P Mysore, Christopher J Tynan, Laura C Zanetti-Domingues, Eric T Kim, Valeria Losasso, Dimitrios Korovesis, Michael Hirsch, et al. Egfr oligomerization organizes kinase-active dimers into competent signalling platforms. *Nature communications*, 7:13307, 2016.

[23] Stefan W Hell and Jan Wichmann. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. *Optics letters*, 19(11):780–782, 1994.

[24] Thomas A Klar and Stefan W Hell. Subdiffraction resolution in far-field fluorescence microscopy. *Optics letters*, 24(14):954–956, 1999.

[25] VA Okhonin. Method of investigating specimen microstructure. *Patent SU*, 1374992, 1986.

[26] Hans Blom and Hjalmar Brismar. Sted microscopy: increased resolution for medical research? *Journal of internal medicine*, 276(6):560–578, 2014.

[27] Michael Hofmann, Christian Eggeling, Stefan Jakobs, and Stefan W Hell. Breaking the diffraction barrier in fluorescence microscopy at low light intensities by using reversibly photoswitchable proteins. *Proceedings of the National Academy of Sciences*, 102(49):17565–17569, 2005.

[28] Stefan Bretschneider, Christian Eggeling, and Stefan W Hell. Breaking the diffraction barrier in fluorescence microscopy by optical shelving. *Physical review letters*, 98(21):218103, 2007.

[29] Thomas Dertinger, Ryan Colyer, Gopal Iyer, Shimon Weiss, and Jörg Enderlein. Fast, background-free, 3d super-resolution optical fluctuation imaging (sofi). *Proceedings of the National Academy of Sciences*, 106(52):22287–22292, 2009.

[30] Nils Gustafsson, Siân Culley, George Ashdown, Dylan M Owen, Pedro Matos Pereira, and Ricardo Henriques. Fast live-cell conventional fluorophore nanoscopy with imagej through super-resolution radial fluctuations. *Nature communications*, 7(1):1–9, 2016.

[31] Eric Betzig, George H Patterson, Rachid Sougrat, O Wolf Lindwasser, Scott Olenych, Juan S Bonifacino, Michael W Davidson, Jennifer Lippincott-Schwartz, and Harald F Hess. Imaging intracellular fluorescent proteins at nanometer resolution. *Science*, 313(5793):1642–1645, 2006.

[32] William E Moerner and Lothar Kador. Optical detection and spectroscopy of single molecules in a solid. *Physical review letters*, 62(21):2535, 1989.

[33] Steven M Kay. *Fundamentals of statistical signal processing: Practical algorithm development*, volume 3. Pearson Education, 2013.

[34] Ismail M Khater, Ivan Robert Nabi, and Ghassan Hamarneh. A review of super-resolution single-molecule localization microscopy cluster analysis and quantification methods. *Patterns*, 1(3):100038, 2020.

[35] Ricardo Henriques, Mickael Lelek, Eugenio F Fornasiero, Flavia Valtorta, Christophe Zimmer, and Musa M Mhlanga. Quickpalm: 3d real-time photoactivation nanoscopy image processing in imagej. *Nature methods*, 7(5):339–340, 2010.

[36] Michael K Cheezum, William F Walker, and William H Guilford. Quantitative comparison of algorithms for tracking single fluorescent particles. *Biophysical journal*, 81(4):2378–2388, 2001.

[37] Anish V Abraham, Sripad Ram, Jerry Chao, ES Ward, and Raimund J Ober. Quantitative study of single molecule location estimation techniques. *Optics express*, 17(26):23352–23373, 2009.

[38] Sébastien Herbert, Helena Soares, Christophe Zimmer, and Ricardo Henriques. Single-molecule localization super-resolution microscopy: deeper and faster. *Microscopy and microanalysis*, 18(6):1419–1429, 2012.

[39] Michael J Rust, Mark Bates, and Xiaowei Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). *Nature methods*, 3(10):793–796, 2006.

[40] Paul D Simonson, Eli Rothenberg, and Paul R Selvin. Single-molecule-based super-resolution images in the presence of multiple fluorophores. *Nano letters*, 11(11):5090–5096, 2011.

[41] Jonas Fölling, Mariano Bossi, Hannes Bock, Rebecca Medda, Christian A Wurm, Birka Hein, Stefan Jakobs, Christian Eggeling, and Stefan W Hell. Fluorescence nanoscopy by ground-state depletion and single-molecule return. *Nature methods*, 5(11):943–945, 2008.

[42] David Baddeley, Isuru D Jayasinghe, Christoph Cremer, Mark B Cannell, and Christian Soeller. Light-induced dark states of organic fluochromes enable 30 nm resolution imaging in standard media. *Biophysical journal*, 96(2):L22–L24, 2009.

[43] Bonnie O Leung and Keng C Chou. Review of super-resolution fluorescence microscopy for biology. *Applied spectroscopy*, 65(9):967–980, 2011.

[44] Bo Huang, Sara A Jones, Boerries Brandenburg, and Xiaowei Zhuang. Whole-cell 3d storm reveals interactions between cellular structures with nanometer-scale resolution. *Nature methods*, 5(12):1047–1052, 2008.

[45] Sri Rama Prasanna Pavani, Michael A Thompson, Julie S Biteen, Samuel J Lord, Na Liu, Robert J Twieg, Rafael Piestun, and William E Moerner. Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function. *Proceedings of the National Academy of Sciences*, 106(9):2995–2999, 2009.

[46] David Baddeley, Mark B Cannell, and Christian Soeller. Three-dimensional sub-100 nm super-resolution imaging of biological samples using a phase ramp in the objective pupil. *Nano Research*, 4(6):589–598, 2011.

[47] Manuel F Juette, Travis J Gould, Mark D Lessard, Michael J Mlodzianoski, Bhupendra S Nagpure, Brian T Bennett, Samuel T Hess, and Joerg Bewersdorf. Three-dimensional sub–100 nm resolution fluorescence microscopy of thick samples. *Nature methods*, 5(6):527–529, 2008.

[48] Daniel Aquino, Andreas Schönle, Claudia Geisler, Claas v Middendorff, Christian A Wurm, Yosuke Okamura, Thorsten Lang, Stefan W Hell, and Alexander Egner. Two-color nanoscopy of three-dimensional volumes by 4pi detection of stochastically switched fluorophores. *Nature methods*, 8(4):353–359, 2011.

[49] Nicolas Bourg, Céline Mayet, Guillaume Dupuis, Thomas Barroca, Pierre Bon, Sandrine Lécart, Emmanuel Fort, and Sandrine Lévêque-Fort. Direct optical nanoscopy with axially localized detection. *Nature Photonics*, 9(9):587–593, 2015.

[50] Gleb Shtengel, James A Galbraith, Catherine G Galbraith, Jennifer Lippincott-Schwartz, Jennifer M Gillette, Suliana Manley, Rachid Sougrat, Clare M Waterman, Pakorn Kanchanawong, Michael W Davidson, et al. Interferometric fluores-

cent super-resolution microscopy resolves 3d cellular ultrastructure. *Proceedings of the National Academy of Sciences*, 106(9):3125–3130, 2009.

[51] Andrey Aristov, Benoit Lelandais, Elena Rensen, and Christophe Zimmer. Zola-3d allows flexible 3d localization microscopy over an adjustable axial range. *Nature communications*, 9(1):1–8, 2018.

[52] Fang Huang, Samantha L Schwartz, Jason M Byars, and Keith A Lidke. Simultaneous multiple-emitter fitting for single molecule super-resolution imaging. *Biomedical optics express*, 2(5):1377–1393, 2011.

[53] M. P. Gordon, T. Ha, and P. R. Selvin. Single-molecule high-resolution imaging with photobleaching. *Proceedings of the National Academy of Sciences*, 101(17):6462–6465, 2004.

[54] Hamza Balci, Taekjip Ha, H Lee Sweeney, and Paul R Selvin. Interhead distance measurements in myosin vi via shrimp support a simplified hand-over-hand model. *Biophysical journal*, 89(1):413–417, 2005.

[55] Bärbel I de Bakker, Frank de Lange, Alessandra Cambi, Jeroen P Korterik, Erik MHP van Dijk, Niek F van Hulst, Carl G Figdor, and Maria F Garcia-Parajo. Nanoscale organization of the pathogen receptor dc-sign mapped by single-molecule high-resolution fluorescence microscopy. *Chemphyschem*, 8(10):1473–1480, 2007.

[56] Xiaohui Qu, David Wu, Laurens Mets, and Norbert F Scherer. Nanometer-localized multiple single-molecule fluorescence microscopy. *Proceedings of the national academy of sciences*, 101(31):11298–11303, 2004.

[57] Stephen ED Webb, Michael Hirsch, Sarah R Needham, Benjamin C Coles, Kathrin M Scherer, Selene K Roberts, Laura C Zanetti-Domingues, Christopher J

Tynan, Marisa L Martin-Fernandez, and Daniel J Rolfe. Nanometric molecular separation measurements by single molecule photobleaching. *Methods*, 88:76–80, 2015.

[58] Dylan T Burnette, Prabuddha Sengupta, Yuhai Dai, Jennifer Lippincott-Schwartz, and Bechara Kachar. Bleaching/blinking assisted localization microscopy for superresolution imaging using standard fluorescent molecules. *Proceedings of the National Academy of Sciences*, 108(52):21081–21086, 2011.

[59] L Stirling Churchman, Zeynep Ökten, Ronald S Rock, John F Dawson, and James A Spudich. Single molecule high-resolution colocalization of cy3 and cy5 attached to macromolecules measures intramolecular distances through time. *Proceedings of the National Academy of Sciences*, 102(5):1419–1423, 2005.

[60] Susan Cox, Edward Rosten, James Monypenny, Tijana Jovanovic-Talisman, Dylan T Burnette, Jennifer Lippincott-Schwartz, Gareth E Jones, and Rainer Heintzmann. Bayesian localization microscopy reveals nanoscale podosome dynamics. *Nature methods*, 9(2):195–200, 2012.

[61] Yunqing Tang, Johnny Hendriks, Thomas Gensch, Luru Dai, and Junbai Li. Automatic bayesian single molecule identification for localization microscopy. *Scientific reports*, 6(1):1–11, 2016.

[62] Daniel J Rolfe, Charles I McLachlan, Michael Hirsch, Sarah R Needham, Christopher J Tynan, Stephen E D Webb, Marisa L Martin-Fernandez, and Michael P Hobson. Automated multidimensional single molecule fluorescence microscopy feature detection and tracking. *European biophysics journal*, 40(10):1167–1186, 2011.

[63] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[64] Daniel Axelrod. Cell-substrate contacts illuminated by total internal reflection fluorescence. *The Journal of cell biology*, 89(1):141–145, 1981.

[65] Russell E. Thompson, Daniel R. Larson, and Watt W. Webb. Precise nanometer localization analysis for individual fluorescent probes. *Biophysical Journal*, 82(5):2775–2783, 2002.

[66] Peter Huang, Kenneth S Breuer, and Jeffrey S Guasto. Evanescent wave microscopy. *Encyclopedia of Microfluidics and Nanofluidics*, pages 1050–1059, 2015.

[67] Christopher H Bohrer, Xinxing Yang, Zhixin Lyu, Shih-Chin Wang, and Jie Xiao. Improved single-molecule localization precision in astigmatism-based 3d superresolution imaging using weighted likelihood estimation. *BioRxiv*, page 304816, 2018.

[68] Alondra Escobar and Christopher M Yip. Optimizing astigmatism for 3d stochastic optical reconstruction microscopy. *Biophysical Journal*, 116(3):440a, 2019.

[69] H Pin Kao and AS Verkman. Tracking of single fluorescent particles in three dimensions: use of cylindrical optics to encode particle position. *Biophysical journal*, 67(3):1291–1300, 1994.

[70] Ginni Grover, Keith DeLuca, Sean Quirin, Jennifer DeLuca, and Rafael Piestun. Super-resolution photon-efficient imaging by nanometric double-helix point spread function localization of emitters (spindle). *Optics express*, 20(24):26681–26695, 2012.

[71] Alex von Diezmann, Yoav Shechtman, and WE Moerner. Three-dimensional localization of single molecules for super-resolution imaging and single-particle tracking. *Chemical reviews*, 117(11):7244–7275, 2017.

[72] Arnauld Sergé, Nicolas Bertaux, Hervé Rigneault, and Didier Marguet. Dynamic multiple-target tracing to probe spatiotemporal cartography of cell membranes. *Nature methods*, 5(8):687–694, 2008.

[73] Ji Won Yoon, Andreas Bruckbauer, William J Fitzgerald, and David Klenerman. Bayesian inference for improved single molecule fluorescence tracking. *Biophysical journal*, 94(12):4932–4947, 2008.

[74] Christoph Bräuchle, Don Carroll Lamb, and Jens Michaelis. *Single particle tracking and single molecule energy transfer*. John Wiley & Sons, 2009.

[75] Toshio Yanagida and Yoshiharu Ishii. *Single molecule dynamics in life science*. John Wiley & Sons, 2008.

[76] MP Hobson, Graça Rocha, and Richard S Savage. Bayesian source extraction. *Bayesian Methods in Cosmology*, page 167, 2009.

[77] Joram Soch. Solution for the indefinite integral of the standard normal probability density function. *arXiv preprint arXiv:1512.04858*, 2015.

[78] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[79] Philip M Dixon. The bootstrap and the jackknife. *Design and analysis of ecological experiments*, pages 267–288, 2001.

[80] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[81] David Martin Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[82] Charles E Metz. Basic principles of ROC analysis. In *Seminars in nuclear medicine*, volume 8, pages 283–298. Elsevier, 1978.

[83] Yutaka Sasaki et al. The truth of the f-measure. *Teach Tutor mater*, 1(5), 2007.

[84] CJ Van Rijsbergen. Information retrieval. dept. of computer science, university of glasgow. *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*, 14, 1979.

[85] Charles X Ling, Jin Huang, and Harry Zhang. AUC: a Better Measure than Accuracy In Comparing Learning Algorithms. In *IN PROC. OF IJCAI{\rq}03*, pages 329–341. Springer, 2003.

[86] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. AUC: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151, 2008.

[87] Peter Norvig Stuart Russel. *Artificial Intelligence, A modern Approach*. Prentice Hall Series in Artificial Intelligence. Pearson, 1 Lake Street, Upper Saddle River, NJ 07458, 2010.

[88] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[89] Geoffrey E Hinton and Terrence J Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 448–453. Citeseer, 1983.

[90] Geoffrey E Hinton and Terrence J Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(282-317):2, 1986.

[91] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[92] John Hawkins and Mikael Bodén. The applicability of recurrent neural networks for biological sequence analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(3):243–253, 2005.

[93] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[94] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. pages 1–28, 2015.

[95] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[96] Yann LeCun, Ido Kanter, and Sara A Solla. Second order properties of error surfaces: Learning time and generalization. In *Advances in neural information processing systems*, pages 918–924, 1991.

[97] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[98] Hartmut Pohlheim. Examples of objective functions. *Retrieved*, 4(10):2012, 2007.

[99] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.0, 2016.

[100] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[101] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[102] D Nguyen and B Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. 3:21–26 vol.3, 07 1990.

[103] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[104] Corinna Cortes Christopher J C Burges Yann LeCun. THE MNIST DATABASE of handwritten digits. \url{http://yann.lecun.com/exdb/mnist/}.

[105] Ueli Meier, Dan Claudiu Ciresan, Luca Maria Gambardella, and Jurgen Schmidhuber. Better digit recognition with a committee of simple neural nets. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1250–1254. IEEE, 2011.

[106] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.

[107] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962, 2003.

[108] Richard Bellman. *Adaptive control process: a guided tour*. 1961.

[109] Gerard V Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on pattern analysis and machine intelligence*, (3):306–307, 1979.

[110] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 233 Spring Street, New York, NY 10013, USA, 2006.

[111] Yuandong Zhao and Zhaohua Hu. A Novel Tracking and Recognition Algorithm Using" Continuous Autoencoder" Network. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–5. IEEE, 2009.

[112] G E Hinton and R R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[113] Juha Karhunen and Jyrki Joutsensalo. Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, 8(4):549–562, 1995.

[114] Nathalie Japkowicz, Stephen Jose Hanson, and Mark A Gluck. Nonlinear autoassociation is not equivalent to PCA. *Neural computation*, 12(3):531–545, 2000.

[115] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[116] Philip Graff, Farhan Feroz, Michael P Hobson, and Anthony N Lasenby. SKYNET: an efficient and robust neural network training tool for machine learning in astronomy. *Mon. Not. Roy. Astron. Soc.*, 441(2):1741–1759, 2014.

[117] Hsin Chen and Alan F Murray. Continuous restricted Boltzmann machine with an implementable training algorithm. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 150, pages 153–158. IET, 2003.

[118] Max Welling, Michal Rosen-zvi, and Geoffrey E Hinton. Exponential Family Harmoniums with an Application to Information Retrieval. In L K Saul, Y Weiss, and L Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1481–1488. MIT Press, 2005.

[119] Simon S Haykin, Simon S Haykin, Simon S Haykin, and Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Education Upper Saddle River, 2009.

[120] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[121] Oliver Woodford. Notes on Contrastive Divergence.

[122] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[123] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[124] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[125] Elias Nehme, Lucien E Weiss, Tomer Michaeli, and Yoav Shechtman. Deep-storm: super-resolution single-molecule microscopy by deep learning. *Optica*, 5(4):458–464, 2018.

[126] Elias Nehme, Daniel Freedman, Racheli Gordon, Boris Ferdman, Lucien E Weiss, Onit Alalouf, Tal Naor, Reut Orange, Tomer Michaeli, and Yoav Shechtman. Deepstorm3d: dense 3d localization microscopy and psf design by deep learning. *Nature methods*, 17(7):734–740, 2020.

[127] Yoav Shechtman, Steffen J Sahl, Adam S Backer, and William E Moerner. Optimal point spread function design for 3d imaging. *Physical review letters*, 113(13):133902, 2014.

[128] Artur Speiser, Lucas-Raphael Müller, Ulf Matti, Christopher J Obara, Wesley R Legant, Anna Kreshuk, Jakob H Macke, Jonas Ries, and Srinivas C Turaga. Deep learning enables fast and dense single-molecule localization with high accuracy. *bioRxiv*, 2020.

[129] Artur Speiser, Lucas-Raphael Müller, Ulf Matti, Christopher J Obara, Wesley R Legant, Jonas Ries, Jakob H Macke, and Srinivas C Turaga. Teaching deep neural networks to localize single molecules for super-resolution microscopy. *arXiv preprint arXiv:1907.00770*, 2019.

[130] Daniel Sage, Thanh-An Pham, Hazen Babcock, Tomas Lukes, Thomas Pengo, Jerry Chao, Ramraj Velmurugan, Alex Herbert, Anurag Agrawal, Silvia Colabrese, et al. Super-resolution fight club: assessment of 2d and 3d single-molecule localization microscopy software. *Nature methods*, 16(5):387–395, 2019.

[131] Wei Ouyang, Andrey Aristov, Mickaël Lelek, Xian Hao, and Christophe Zimmer. Deep learning massively accelerates super-resolution localization microscopy. *Nature biotechnology*, 36(5):460–468, 2018.

[132] Sunil Kumar Gaire, Yang Zhang, Hongyu Li, Ray Yu, Hao F Zhang, and Leslie Ying. Accelerating multicolor spectroscopic single-molecule localization microscopy using deep learning. *Biomedical Optics Express*, 11(5):2705–2721, 2020.

[133] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016.

[134] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6721–6729, 2017.

[135] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[136] Ugur Demir and Gozde Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

[137] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, and Shiguang Shan. Shift-net: Image inpainting via deep feature rearrangement. In *Proceedings of the European conference on computer vision (ECCV)*, pages 1–17, 2018.

[138] Silvia Colabrese, Marco Castello, Giuseppe Vicidomini, and Alessio Del Bue. Machine learning approach for single molecule localisation microscopy. *Biomedical optics express*, 9(4):1680–1691, 2018.

[139] IEEE. 2013 isbi grand challenge localization microscopy. `http://bigwww.epfl.ch/smlm/challenge2013/index.html`, 2013.

[140] Pasquale Cascarano, Maria Colomba Comes, Andrea Sebastiani, Arianna Mencattini, Elena Loli Piccolomini, and Eugenio Martinelli. Deepcel0 for 2d single molecule localization in fluorescence microscopy. *arXiv preprint arXiv:2107.02281*, 2021.

[141] Emmanuel Soubies, Laure Blanc-Féraud, and Gilles Aubert. A continuous exact \ell_0 penalty (cel0) for least squares regularized problem. *SIAM Journal on Imaging Sciences*, 8(3):1607–1639, 2015.

[142] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[143] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 265–273. Springer, 2018.

[144] Martin Weigert, Uwe Schmidt, Robert Haase, Ko Sugawara, and Gene Myers. Star-convex polyhedra for 3d object detection and segmentation in microscopy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3666–3673, 2020.

[145] Reka Hollandi, Abel Szkalisity, Timea Toth, Ervin Tasnadi, Csaba Molnar, Botond Mathe, Istvan Grexa, Jozsef Molnar, Arpad Balind, Mate Gorbe, et al. nucleaizer: a parameter-free deep learning framework for nucleus segmentation using image style transfer. *Cell Systems*, 10(5):453–458, 2020.

[146] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.

[147] Leonhard Möckl, Anish R Roy, Petar N Petrov, and WE Moerner. Accurate and rapid background estimation in single-molecule localization microscopy using the deep neural network bgnet. *Proceedings of the National Academy of Sciences*, 117(1):60–67, 2020.

[148] Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018.

[149] Hongda Wang, Yair Rivenson, Yiyin Jin, Zhensong Wei, Ronald Gao, Harun Günaydın, Laurent A Bentolila, Comert Kural, and Aydogan Ozcan. Deep learning enables cross-modality super-resolution in fluorescence microscopy. *Nature methods*, 16(1):103–110, 2019.

[150] Karolis Misiunas, Niklas Ermann, and Ulrich F Keyser. Quipunet: convolutional neural network for single-molecule nanopore sensing. *Nano letters*, 18(6):4040–4045, 2018.

[151] Maximilian AH Jakobs, Andrea Dimitracopoulos, and Kristian Franze. Kymobutler, a deep learning software for automated kymograph analysis. *Elife*, 8:e42288, 2019.

[152] Chawin Ounkomol, Sharmishtaa Seshamani, Mary M Maleckar, Forrest Collman, and Gregory R Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature methods*, 15(11):917–920, 2018.

[153] Lucas von Chamier, Romain F Laine, Johanna Jukkala, Christoph Spahn, Daniel Krentzel, Elias Nehme, Martina Lerche, Sara Hernández-Pérez, Pieta K Mattila, Eleni Karinou, et al. Democratising deep learning for microscopy with zerocostdl4mic. *Nature communications*, 12(1):1–18, 2021.

[154] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[155] Mohamadreza Fazel. Analysis of single molecule fluorescence microscopy data. *arXiv preprint arXiv:2103.11246*, 2021.

[156] Chinmay Belthangady and Loic A Royer. Applications, promises, and pitfalls of deep learning for fluorescence image reconstruction. *Nature methods*, 16(12):1215–1225, 2019.

[157] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Icml*, 2011.

[158] Gerald C Holst and Terrence S Lomheim. Cmos/ccd sensors and camera systems. *Reconstruction*, 9(5):2PFC, 2011.

[159] Ivo F Sbalzarini and Petros Koumoutsakos. Feature point tracking and trajectory analysis for video imaging in cell biology. *Journal of structural biology*, 151(2):182–195, 2005.

[160] Michael Hirsch, Richard J Wareham, Marisa L Martin-Fernandez, Michael P Hobson, and Daniel J Rolfe. A stochastic model for electron multiplication charge-coupled devices–from theory to practice. *PloS one*, 8(1):e53671, 2013.

[161] Joseph R Lakowicz. *Principles of Fluorescence Spectroscopy*. Springer, University of Maryland School of Medicine Baltimore, Maryland, USA, 2006.

[162] J F Kenney and E S Keeping. Linear Regression, Simple Correlation, and Contingency. In *Mathematics of Statistics*, volume 2, chapter 8, pages 199–237. Princeton, NJ: Van Nostrand, 2 edition, 1951.

[163] Wenjun Sun, Siyu Shao, Rui Zhao, Ruqiang Yan, Xingwu Zhang, and Xuefeng Chen. A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement*, 89:171–178, 2016.

[164] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[165] Sarah R Needham, Laura C Zanetti-Domingues, Kathrin M Scherer, Michael Hirsch, Daniel J Rolfe, Selene K Roberts, Marisa L Martin-Fernandez, David T Clarke, and Christopher J Tynan. Determining the geometry of oligomers of the human epidermal growth factor family on cells with¡ 10 nm resolution, 2015.

[166] Samson Amos, Patrick M Martin, Gregory A Polar, Sarah J Parsons, and Isa M Hussaini. Phorbol 12-myristate 13-acetate induces epidermal growth factor receptor transactivation via protein kinase cδ/c-src pathways in glioblastoma cells. *Journal of Biological Chemistry*, 280(9):7729–7738, 2005.

[167] L Stirling Churchman, Henrik Flyvbjerg, and James A Spudich. A non-gaussian distribution quantifies distances measured with fluorescence localization techniques. *Biophysical journal*, 90(2):668–671, 2006.

[168] Diana B Peckys, Jean-Pierre Baudoin, Magdalena Eder, Ulf Werner, and Niels De Jonge. Epidermal growth factor receptor subunit locations determined in hydrated cells with environmental scanning electron microscopy. *Scientific reports*, 3(1):1–8, 2013.

[169] Nicholas J Bessman, Daniel M Freed, and Mark A Lemmon. Putting together structures of epidermal growth factor receptors. *Current opinion in structural biology*, 29:95–101, 2014.

[170] Malgorzata Kluba, Yves Engelborghs, Johan Hofkens, and Hideaki Mizuno. Inhibition of receptor dimerization as a novel negative feedback mechanism of egfr signaling. *PLoS One*, 10(10):e0139971, 2015.

[171] Martin Ovesnỳ, Pavel Křížek, Josef Borkovec, Zdeněk Švindrych, and Guy M Hagen. Thunderstorm: a comprehensive imagej plug-in for palm and storm data analysis and super-resolution imaging. *Bioinformatics*, 30(16):2389–2390, 2014.

[172] Lucas P Watkins and Haw Yang. Detection of intensity change points in time-resolved single-molecule measurements. *The Journal of Physical Chemistry B*, 109(1):617–628, 2005.

[173] Michael Andrec, Ronald M Levy, and David S Talaga. Direct determination of kinetic rates from single-molecule photon arrival trajectories using hidden markov models. *The Journal of Physical Chemistry A*, 107(38):7454–7464, 2003.

[174] Ryohei Yasuda, Tomoko Masaike, Kengo Adachi, Hiroyuki Noji, Hiroyasu Itoh, and Kazuhiko Kinosita. The atp-waiting conformation of rotating f1-atpase revealed by single-pair fluorescence resonance energy transfer. *Proceedings of the National Academy of Sciences*, 100(16):9314–9318, 2003.

[175] Manuel Diez, Boris Zimmermann, Michael Börsch, Marcelle König, Enno Schweinberger, Stefan Steigmiller, Rolf Reuter, Suren Felekyan, Volodymyr Kudryavtsev, Claus AM Seidel, et al. Proton-powered subunit rotation in single membrane-bound f 0 f 1-atp synthase. *Nature structural & molecular biology*, 11(2):135–141, 2004.

[176] Robert M Dickson, Andrew B Cubitt, Roger Y Tsien, and William E Moerner. On/off blinking and switching behaviour of single molecules of green fluorescent protein. *Nature*, 388(6640):355–358, 1997.

[177] Ahmet Yildiz, Joseph N Forkey, Sean A McKinney, Taekjip Ha, Yale E Goldman, and Paul R Selvin. Myosin v walks hand-over-hand: single fluorophore imaging with 1.5-nm localization. *science*, 300(5628):2061–2065, 2003.