Computational Methods for *in vivo* Single Molecule Dynamics in Bacterial Cells



Rasched Haidari

Linacre College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2025

Disclaimer

I hereby declare that the work in this thesis is that of the candidate alone, except where indicated in the text, and as described below.

- Strains of sigma-54, sigma-70, and RNA Polymerase fused to PAmCherry in *E. coli* were developed by Hafez El Sayyed.
- Experimental data in Chapter 2 and data for sigma-70 and sigma-54 in Chapter 3 was acquired by Hafez El Sayyed. Experimental data in Chapter 3 for the case of DNA & RNA Polymerase was taken by myself.
- Single molecule microscope, as described in Chapter 2, was built by Oliver J. Pambos.
- Initial localisation, tracking, cell normalisation, and cell segmentation software was developed by Seamus Holden, Stephan Uphoff, Mathew Stracy, and Piers Turner respectively.

Rasched Haidari
January 2025

Abstract

This thesis develops and applies computational methods for analysing single molecule trajectories in bacterial cells obtained from super-resolution imaging, specifically Photoactivated Localisation Microscopy.

We begin by characterising the *in vivo* diffusive properties of two previously poorly understood proteins essential for bacterial transcription, sigma-54 and sigma-70. Here, we uncover distinct biological behaviours in various cellular environments, including stress conditions, using ensemble diffusion analysis. This provides insight into transcription processes in *Escherichia coli*.

We then develop two novel machine learning approaches for analysing single protein trajectories. First, we introduce a convolutional neural network for classifying protein states at long exposure times on a per frame basis. We use this method to calculate protein binding times to the DNA. This method addresses limitations in current traditional approaches, and provides a quantitative framework for determining when "long exposure" imaging becomes effective for different proteins. We validate our approach using single molecule data from a DNA-binding protein and estimate binding times for RNA Polymerase, sigma-70, and sigma-54.

Finally, we develop a novel recurrent neural network for analysing diffusive properties of heterogeneous single protein trajectories in complex cellular environments. This method infers diffusion coefficients, anomalous exponents, and biological states at each time point, while also subsequently detecting transitions between behaviours.

In this work, we also demonstrate how machine learning can enhance our understanding of protein dynamics beyond traditional statistical methods. The methods developed in this thesis are broadly applicable across different experimental setups and proteins, and provide new tools for investigating cellular processes at the single molecule level. This thesis contributes both to our fundamental understanding of bacterial transcription and to the methodological toolkit for analysing single molecule data.

Acknowledgements

I would like to thank my supervisor Achilles Kapanidis for his guidance, support, and generosity throughout the past few years.

I would also like to thank the Kapanidis Group for welcoming me when I first started and the good laughs throughout this journey. In particular I would like to thank Hafez El Sayyed for teaching me all about RNAP and the joy of working with him. Thank you to Piers Turner, who was always around to answer my machine learning queries and provided valuable comments on my work, and Jagadish Prasad Hazra for insightful discussions across life and science. I would also like to thank the rest of the Kapanidis Group (past and present): Stelios Chatzimichail for making lab chess a thing, Edward Wheeler for showing me how to play better chess by winning every game, Nicolas Shiaelis for more chess, Jacob Wright for checking in on my protein intake, Sammi Ta for amazing lab socials, Mirjam Kummerlin for amazing lab socials before retiring as lab social manager, Alison Farrar for training for a 10km run with me, Qing Zhao for taking over marking students when I couldn't anymore, Emma Lalande for laughs and chatter, Oliver Pambos for training me in using lab software and equipment, Christoff Hepp for inviting me to climbing although I never attended, Suchintak Dash for making very good baked rasgulla, Abhishek Mazumder for once doing the workout plan I gave him, and Anna Wang for organising lab runs. I am grateful to the Computational Discovery DTC for funding.

To my friends: Thank you for making my life more fun!
To my parents: Thank you for your love and support!

Rasched Haidari
January 2025

Publications

- **Pointwise prediction of protein diffusive properties using machine learning**. R. Haidari, A.N. Kapanidis. *Manuscript in submission.*
- **Quantitative evaluation of methods to analyze motion changes in single-particle experiments**. Muñoz-Gil *et al. Manuscript in submission.*
- **Determining DNA-Bound Times of Proteins *in vivo* using single-molecule tracking and deep learning**. R. Haidari, H. El-Sayyed, A.N. Kapanidis. *Manuscript in preparation.*
- **Single-molecule tracking reveals the functional allocation and *in vivo* interactions of sigma-70 and sigma-54 in *Escherichia coli*** (Provisional Title). R. Haidari, H. El Sayyed, A.N. Kapanidis. *Manuscript in preparation.*

Contents

1 Introduction **1**

1.1 Imaging Single Molecules . . . 1
1.2 Photoactivated Localisation Microscopy . . . 4
1.2.1 Imaging Modes for PALM . . . 4
1.2.2 Fluorophore Photophysics . . . 7
1.2.3 Localisation and Tracking . . . 8
1.2.4 Diffusion . . . 10
1.3 Machine Learning . . . 11
1.3.1 Neural Networks . . . 12
1.3.2 Optimisation . . . 14
1.3.3 Model Choice . . . 16
1.3.4 Applications to Single Particle Tracking . . . 16
1.4 Overview of Thesis . . . 21

2 Photoactivated Localisation Microscopy of Sigma Factors *in vivo* 24

2.1 Introduction . . . 25
2.1.1 Brief Overview . . . 25
2.1.2 Background on Transcription . . . 25
2.1.3 Background on Sigma Factors . . . 27
2.2 Methods . . . 29
2.2.1 Bacterial Strains . . . 29
2.2.2 Cell Preparation . . . 29
2.2.3 Microscope . . . 31
2.2.4 Localisation and Tracking . . . 32
2.2.5 Mapping Localisations onto Heatmaps . . . 33
2.2.6 Calculating Diffusion Coefficients . . . 34
2.2.7 Fitting Sub-populations . . . 35
2.3 Results . . . 36
2.3.1 Diffusion Properties in Rich M9 . . . 36
2.3.2 Heatmaps for Spatial Distribution in Rich M9 . . . 42
2.3.3 Nitrogen Starvation . . . 44
2.3.4 Overnight Stationary . . . 47

2.3.5 Serine Hydroxamate Treatment 49
2.4 Discussion 50

3 Capturing Protein Trajectories at Long Exposures 55
3.1 Introduction 56
3.1.1 Brief Overview 56
3.1.2 Protein Binding Times 56
3.2 Methods 59
3.2.1 Convolutional Neural Networks 59
3.2.2 Data Generation 60
3.2.3 Model Architecture 62
3.2.4 Training 64
3.2.5 Inference 66
3.2.6 Bound Time Calculations 68
3.2.7 Cell Preparation 71
3.2.8 Analysis of Classifier Output 72
3.3 Results 72
3.3.1 Simulations 72
3.3.2 Explaining Model Classification 75
3.3.3 Application to *in vivo* PALM 78
3.3.4 Breaking Point 85
3.3.5 Image Quality 87
3.4 Discussion 88

4 Diffusive Properties in Heterogeneous Trajectories 92
4.1 Introduction 93
4.1.1 Brief Overview 93
4.1.2 Difficulties in Determining Protein Properties 93
4.1.3 Anomalous Diffusion Challenge 96
4.2 Methods 98
4.2.1 Recurrent Neural Networks 98
4.2.2 Model Architecture 101
4.2.3 Feature Selection 103
4.2.4 Generating Training Data 105
4.2.5 Augmentations 107
4.2.6 Training 108
4.2.7 Post-Processing 109
4.2.8 Changepoint Detection 109
4.2.9 Evaluation Metrics 110
4.3 Results 111

4.3.1 Pointwise Inference of α, D, and s 112
4.3.2 Dependence on Trajectory Length 115
4.3.3 Changepoints 115
4.3.4 Dependence on Biological Behaviours 122
4.3.5 Shortcomings 123
4.4 Discussion 125

5 Conclusion and Future Outlook **127**
5.1 Summary 127
5.2 Future Work 129

References **132**

1 Introduction

Contents

1.1 Imaging Single Molecules 1
1.2 Photoactivated Localisation Microscopy 4
1.2.1 Imaging Modes for PALM 4
1.2.2 Fluorophore Photophysics 7
1.2.3 Localisation and Tracking 8
1.2.4 Diffusion 10
1.3 Machine Learning 11
1.3.1 Neural Networks 12
1.3.2 Optimisation 14
1.3.3 Model Choice 16
1.3.4 Applications to Single Particle Tracking 16
1.4 Overview of Thesis 21

1.1 Imaging Single Molecules

The ability to observe biological systems allows for the verification of scientific hypotheses at the smallest of scales. The earliest observations of unicellular organisms and bacteria was by van Leeuwenhoek in the 17th century [1]. His contemporary, Robert Hooke, was another pioneering figure in the field, who published observations of various small living objects, through his microscope, in his renowned book *Micrographia* [2].

The observations by van Leeuwenhoek and Hooke relied on the compound microscope, using two convex lenses (objective and eye lens), to enlarge a given object. In 1873, Ernst Abbe published the resolution limit of such diffraction-limited microscopes as given by [3]:

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

where λ is the wavelength of light, NA is the numerical aperture of the objective lens, and d is the minimum resolvable distance. For a $50\times$ magnification compound microscope, $NA \sim 0.7$. Taking $\lambda = 550\ nm$, we achieve $d \sim 400\ nm$ as the minimum resolvable distance between two objects. While traditional microscopes continued to improve, they were ultimately limited by the physics of Abbe's equation. This led to the development of alternative approaches, including confocal microscopy. Using an oil immersion objective, $NA \sim 1.4$ for a typical confocal microscope, such that $d \sim 200\ nm$.

One of the more powerful applications of microscopy is in fluorescent imaging of biological structures such as cells [4]. However, most biological samples require fluorescent markers to be visible. A breakthrough in this field came with the development and extraction of green fluorescent protein (GFP) from the bioluminescent jellyfish *Aequorea victoria* [5]. Genetic modification of cells by introducing GFP enables biological samples to become visible. When laser light of a specific wavelength illuminates genetically modified cells, it energizes electrons in the fluorescent proteins, causing them to transition to an excited state. Subsequently, the electron relaxes back to its ground state and emits light of a different wavelength, which is detected by sensors. This concept is what gave rise to the vast field of fluorescence microscopy. Due to losses of energy in transitions between excited and ground states, the energy of the emitted photon is less than that of the absorbed photon (known as the Stokes shift). Thus, the emitted photon has a longer wavelength than the wavelength of light used to excite the sample.

Shining light on biological samples with fluorescent proteins illuminates many proteins at once, perfect for viewing large cellular structures. However, this renders

imaging single molecules, which are smaller than the diffraction limit, impossible. The leap from resolving single cells to single molecules was pioneered with the development of advanced fluorescence labelling and protein localisation methods.

There were a number of advancements that made single molecule imaging possible. Firstly, the development of photoactivatable proteins [6]. These fluorescent proteins can be switched ON and OFF, allowing for more precise control of regions within the cell that are emitting light. Different super-resolution techniques emerged, including Photoactivated Localisation Microscopy (PALM), which uses photoactivatable proteins to image a few molecules in a cell at a time [7]. Alongside these various improvements, mathematical and computational work in the localisation of signals from samples finally allowed for "pushing beyond the diffraction limit", achieving $d \sim 20\ nm$ [8]. There have also been significant advancements in imaging technology, particularly with the development of highly sensitive detectors like the Electron Multiplying Charge Coupled Device (EMCCD) and the scientific Complementary Metal Oxide Semiconductor (sCMOS).

In addition to PALM, several other super-resolution imaging techniques are widely used, each with their unique strengths. dSTORM (direct Stochastic Optical Reconstruction Microscopy), similar to PALM, relies on the stochastic activation and localisation of single molecules but uses photo-switchable organic dyes instead of genetically encoded proteins. While dSTORM can reach comparable resolution, it is more commonly applied to fixed cell samples. STED (Stimulated Emission Depletion) uses a depletion laser to selectively suppress fluorescence around a focal point, enabling continuous imaging and minimal post-processing of acquired data, but it does not achieve sufficient resolution for single molecule localisation. MINFLUX (minimal fluorescence photon flux microscopy) offers the highest spatial resolution of current techniques (down to a few nanometres) by using a donut-shaped excitation beam and localising molecules based on minimal photon emission. However, it can be technically demanding and less accessible. PALM stands out for its use of genetically encoded fluorescent proteins, allowing for powerful high-resolution single molecule imaging and tracking in live cells.

In this work, we will be using PALM for single molecule imaging and fluorescent proteins as markers (interchangeably referred to as fluorophore). The remaining sections will provide brief introductions to PALM and Machine Learning, which form the basis of this thesis.

1.2 Photoactivated Localisation Microscopy

PALM is a super resolution microscopy technique that begins with tagging a protein of interest with another photoactivatable protein [7]. In this work, we use Photoactivated mCherry (PAmCherry), a $\sim 30\ kDa$ fluorescent protein derived from jellyfish, to visualise proteins in live *E. coli* cells (*in vivo*) [9]. The typical rod-shaped *E. coli* cell has a length of 2 μm and width 0.5 μm. For instance, PAmCherry is attached to sigma-70 in Chapter 2 , a protein involved in transcription initiation. Proteins are stochastically photoactivated using laser light of a specific wavelength. Activation is a necessary step as it places the protein into an excitable state, and occurs through irreversible changes in the molecular structure upon absorbing the laser light. Then, using laser light of a different wavelength, the molecules are cyclically excited and de-excited, with the emitted light picked up by a detector. By keeping track of the position of emitted light, we can construct movies of protein movement. This is depicted in Figure 1.1. PALM allows for minimum resolvable distances (d) of roughly 20 nm [7, 10, 11].

Note that the creation of these protein-protein fusions and subsequent insertion into cells can be achieved through experimental techniques, but is not the focus of this work.

1.2.1 Imaging Modes for PALM

PALM allows for several imaging modes based on the illumination angle to emphasise different sample properties. Of these, prominent imaging modes include wide-field (epifluorescence), highly inclined laminated optical sheet (HILO), and total internal reflection fluorescence (TIRF) illumination. These modes are shown in Figure 1.2. A cell sample is placed on top of a coverslip. Imaging is performed with an oil

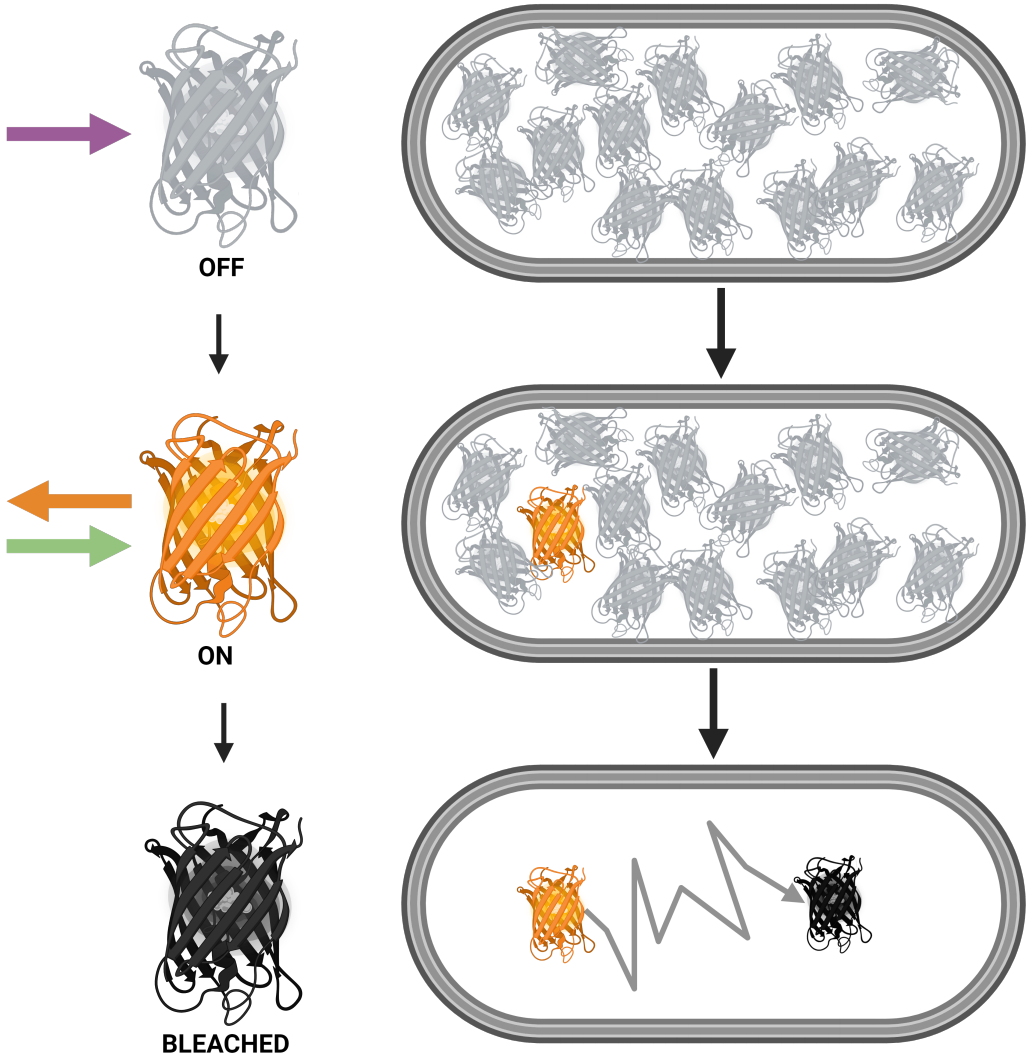


Figure 1.1: PALM process: Fluorophores need to be activated before they emit light. In the case of PAmCherry, we use a 405 *nm* laser (purple). The fluorophore can now be excited using a 561 *nm* laser (green) and it emits a 600 *nm* wavelength of light after de-excitation (orange). We repeat this process inside live cells and keep track of protein movement allowing us to build up trajectories. Eventually, the fluorophore can no longer emit light and is now bleached. Figure adapted from [12] and created using BioRender.

immersion between the objective and coverslip, increasing the numerical aperture of the objective (above 1 which is the limit for air).

Epifluorescence (Epi) involves illuminating the entire sample with laser light. This causes many fluorescent proteins inside the cell to stochastically activate and become excited. Epi works well for thick samples (>10μm) and is easy to align and setup. However, excitation of fluorophores at various depths produces a lot of light

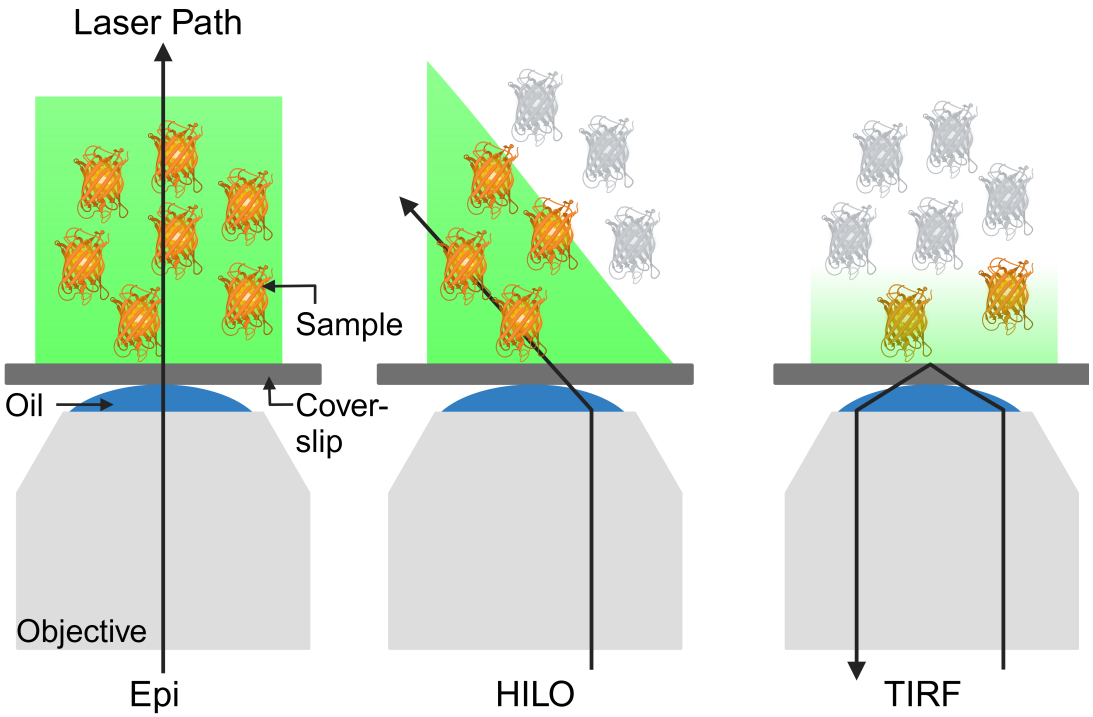


Figure 1.2: PALM imaging modes. Epifluorescence excites all fluorophores in the sample whereas HILO only a subset. TIRF excites fluorophores close to the surface of the sample, within the region of an evanescent wave. Figure created using BioRender.

detected from out-of-focus planes, leading to higher background signal.

HILO illuminates the sample at an angle. As fewer fluorophores are excited, there is a better signal-to-noise ratio than Epi ($\sim 8\times$) [13]. HILO can be used for samples up to 10μm thick and provides a good compromise between Epi and TIRF.

TIRF is achieved once the angle of the incoming laser ray goes beyond the critical angle of the glass coverslip/sample interface. At this interface, the laser light is reflected back (total internal reflection), but also produces an evanescent wave propagating parallel to the interface into the sample. The intensity of the evanescent wave with distance (z) from the surface is given by [14]:

$$I = I_0 e^{-z/d} \tag{1.2}$$

for some constant intensity I_0 and characteristic penetration depth d. As the wave decays exponentially, TIRF excites only fluorophores within a depth of $\sim 200\ nm$ from the coverslip surface, leading to little emitted light from non-focal planes.

This provides the optimal signal-to-noise ratio for single molecule imaging and can be advantageous for studying membrane proteins. The immersion oil and glass have similar refractive indices, minimising light losses due to refraction and reflection at the glass-oil interface.

Each mode has its own drawbacks and advantages. Depending on the goal, sample thickness, and acceptable signal-to-noise ratio, the exact imaging mode will vary. In this work we implement HILO for single molecule tracking in *E. coli* cells.

1.2.2 Fluorophore Photophysics

As depicted in Figure 1.1, fluorophores need to be activated. Subsequent laser excitation allows them to emit light for a brief period of time, until they bleach, and are no longer excitable. The amount of photons a fluorescent protein can emit is fixed and known as its photon budget [12]. High laser intensities produces sharp single molecule images but leads to fast bleaching of the fluorophore as its photon budget quickly depletes. Lower intensities may provide a good compromise between imaging molecules for longer while still attaining sufficiently sharp single molecule peaks. We quantify "sharpness" of single molecule images through the signal-to-noise, defined as the ratio of the power of the signal to the power of the background noise.

Usually in PALM experiments, the laser intensity, exposure time, and fluorescent protein of choice are the most important factors. Depending on the biological process we image, these variables can be tuned. As an example, transcriptional processes of RNA Polymerase can range up to minutes. In this instance, it would require fluorophores with a high photon budget (e.g., Halo tag is very photostable), potentially lower laser intensity depending on its budget, and long exposure times. On the other hand, some processes such as DNA repair mechanisms by DNA Polymerase I only take at most a few seconds [15]. In this case we can afford to use a quickly bleaching fluorophore, but will need to image at shorter exposure times to avoid losing information regarding the process (e.g., PAmCherry is more convenient and affordable) [9, 16]. This is further illustrated in Chapter 2 and Chapter 3.

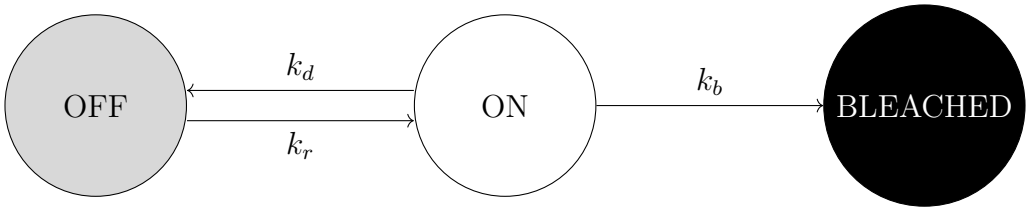


Figure 1.3: State diagram showing transitions between fluorophore states. Transitions between ON and OFF states leads to blinking. Transition rates are given by k.

The photophysics of fluorophores can be modelled using a state transition diagram as shown in Figure 1.3 [17]. This simple diagram encapsulates an ON state, an OFF state (dark but not bleached), and a bleached state. As we have a transition from an OFF state to an ON state and vice versa, the fluorophore can "blink". This can be seen during imaging, as the same fluorophore will briefly turn off for a few frames before turning back on [17]. The concept of blinking complicates matters when determining whether a given fluorophore has bleached or whether it is simply in an OFF state. This can lead to over- or under-counting of the number of proteins that are inside a given cell and hence copy number estimates. We provide a simple fix in the next subsection when tracking proteins.

The transition matrix for the state diagram in Figure 1.3 is given by [17]:

$$T = \begin{pmatrix} e^{-k_d\tau_{\text{exp}}} + e^{-k_b\tau_{\text{exp}}} - 1 & 1 - e^{-k_d\tau_{\text{exp}}} & 1 - e^{-k_b\tau_{\text{exp}}} \\ 1 - e^{-k_r\tau_{\text{exp}}} & e^{-k_r\tau_{\text{exp}}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1.3}$$

where k are the respective rates of the processes and τ_{exp} is the exposure time. At high exposure times, blinking is not visible as it averages out over the time interval (high k_r leads to the OFF state immediately returning to the ON state as can be seen from the transition matrix). Chapter 3 quantifies what is meant by "small/short" and "high/long" exposure times.

1.2.3 Localisation and Tracking

A key part of achieving sub-diffraction limit localisation in single molecule data is the process of fitting Point Spread Functions (PSFs). PSFs describe the resulting light pattern from a point source when imaged through a microscope. Microscopes can have different PSFs depending on the optical setup. In this work we localise single

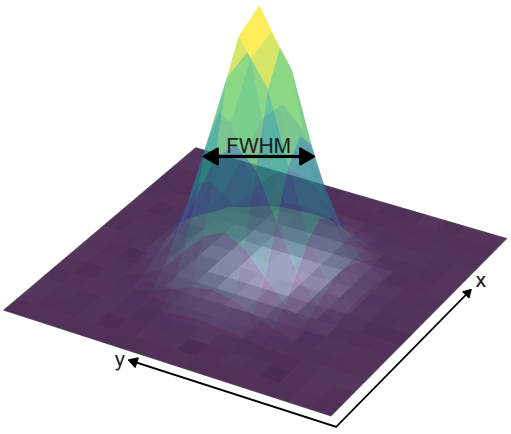


Figure 1.4: A Gaussian PSF is fitted to a noisy single molecule signal. The full-width at half maximum (FWHM) in the $x-$ and $y-$ direction give $2.355\sigma_{x/y}$ (standard deviations/widths of Gaussian). As the molecule can be in a rotated position, typically we use the most general form of a two-dimensional Gaussian fit, which includes a rotation parameter from its major and minor axis. Details are described in Chapter 2. Before fitting, typically image filtering and denoising techniques are applied.

molecules using a Gaussian PSF. This involves finding intensity peaks on single molecule videos and using numerical curve-fitting algorithms to fit two-dimensional Gaussian functions to single molecule spots. Exact details of the fitted function is described in Chapter 2 but Figure 1.4 highlights the fit for a single spot at a given frame. Fitting a PSF to the spot allows the centre of it to be determined with $\sim 20\ nm$ precision, enabling the formation of super-resolution images [8].

Once the process of "localisation" is complete, we essentially "connect the dots" during the tracking process in order to form protein trajectories. There are a few caveats in this process, for instance, blinking. Typically we use a memory parameter to bypass this, which is the maximum time a protein can be in an OFF state for it to be still considered part of the same trajectory. This can depend on the fluorophore used, but for PAmCherry we set this to 1. Another caveat stems from the experimental procedure. Accurate reconstruction of protein tracks requires a minimal activation laser intensity to limit the number of fluorophores excited within a given cell. A high laser intensity renders it difficult to follow a given protein, particularly if multiple proteins are crossing paths, leading to incorrect trajectory assignment. We therefore tune the laser intensity such that at most one or two fluorophores are active in a given cell at any time [12, 15, 18–20].

Tracking PALM is the name given to the process of obtaining PALM images of single molecule movement followed by localisation and tracking [11]. Following successful tracking various physical properties of the protein can be deduced, such

as its diffusion coefficient, copy number, and interactions with other objects in the cell (e.g. DNA binding time in Chapter 3). We apply this in Chapter 2 for developing an understanding of the behaviours of two previously uncharacterised proteins in DNA transcription.

1.2.4 Diffusion

Protein movement in a cell is governed by diffusion and is a constant theme throughout this work. Diffusive processes are driven by random Brownian motion due to molecular collisions. We are particularly interested in how the diffusion coefficient (interchangeably referred to as mobility in this work) links to the mean square displacement (MSD) of protein trajectories. The MSD can be directly calculated from protein tracking coordinates. For simplicity, consider the diffusion equation in one dimension [21]:

$$\frac{\partial p(x,t)}{\partial t} = D\frac{\partial^2 p(x,t)}{\partial x^2} \tag{1.4}$$

where $p(x,t)$ represents the probability of finding the particle in position x at time t and D is the diffusion coefficient.

This has solution (taking $x = 0$ when $t = 0$):

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right) \tag{1.5}$$

The MSD is defined as:

$$\langle x^2(t)\rangle = \int_{-\infty}^{\infty} x^2 p(x,t)dx \tag{1.6}$$

Substituting for $p(x,t)$ and solving yields:

$$MSD = \langle x^2(t)\rangle = 2Dt \tag{1.7}$$

For the n-dimensional case this generalises to $MSD_n = \langle x_n^2\rangle = 2nDt$. As we work with PALM images, which are two-dimensional in nature ($n = 2$), the diffusion coefficient can be measured from tracking coordinates as $D = MSD/4t$. This

equation forms the basis of the work in Chapter 2. In Chapter 4, we consider fractional Brownian motion, in which successive protein steps are correlated during diffusion in complex biological environments.

The exact mechanisms by which proteins diffuse and locate their target site remains poorly understood. It has been suggested that DNA-binding proteins combine one-dimensional sliding with three-dimensional diffusion when searching across the huge landscape of the bacterial chromosome. These approaches require more experimental backing *in vivo* and may vary depending on the mechanisms of the system being studied [22–26].

1.3 Machine Learning

Machine learning (ML) is a branch of artificial intelligence that focuses on developing algorithms (or models) which learn from data and make predictions on unseen cases. ML models generalise well to different tasks, and can learn from high dimensional data such as images. In order to develop an ML model, several steps have to be taken. Data for training the model must be collected and preprocessed/cleaned. As part of this process, features can be extracted from the data that we think are crucial for the model to consider. In the case of single molecule spots, these features may be the intensity and widths of the Gaussian PSF fit to the spot. Depending on the task, the raw data (e.g., image/video) may also be input into the model. Usually features are normalised. This involves constricting the range of the input, which allows ML models to learn patterns more easily, provides numerical stability during training, and places different features on the same scale.

In this work, we focus on supervised learning, in which every input vector $\vec{\mathbf{x}}_i$ has an associated ground truth label y_i, such that our data consists of pairs ($\vec{\mathbf{x}}_i$, y_i) for some maximum dataset size i.

Depending on the task, a suitable model architecture is chosen (see Section 1.3.3 and later chapters). Models have trainable parameters that are tuned during the learning process. There are also a set of hyperparameters that need to be set before learning begins. These involve the learning rate, epochs (number of

times entire dataset is passed through the model), batch size (number of training examples passed through in one iteration), weight decay (to prevent over-fitting, see Section 1.3.2), etc.

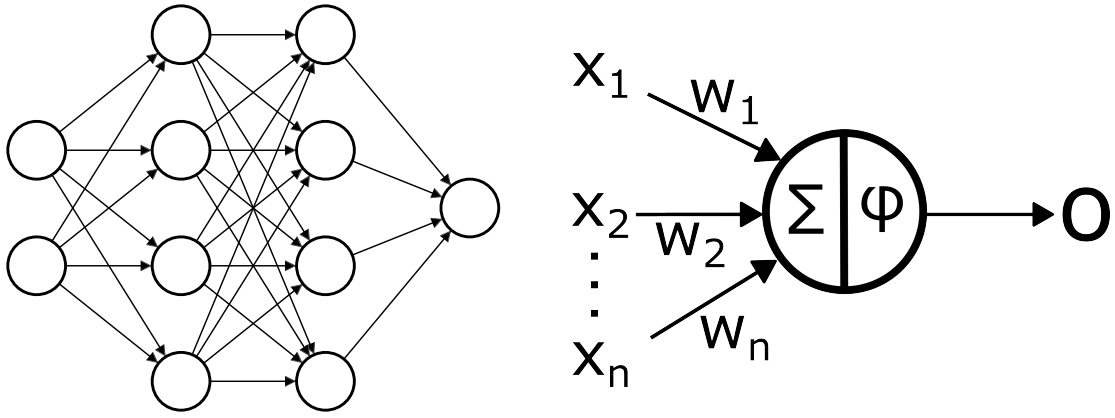
The dataset is split into a training, validation, and a test set. The training set is what the model uses to learn whereas the validation set is used to evaluate the model during training, allowing for detection of under- or over-fitting. The validation set also allows for optimisation of hyperparameters (see Chapter 4). Lastly, the test set consists of data points that the model has not seen, providing an unbiased estimate of model performance. Evaluation metrics will vary on the problem and are shown in Chapter 3 and Chapter 4. ML models are trained up until convergence, at which point learning plateaus and the local minimum (ideally global) of a loss function is reached. The loss function quantifies the difference between the ground truth label (y_i) and the model prediction ($\hat{y}_i$) and is chosen by the ML practitioner.

The process of developing state-of-the-art ML models is lengthy and requires careful evaluation and testing, as they can be "black box" models by nature. However, ML methods can significantly outperform methods based on traditional statistics and learn hidden underlying patterns.

1.3.1 Neural Networks

Neural Networks are mathematical models consisting of groups of neurons arranged in layers. Connections between neurons in different layers allow for information propagation. This brief introduction focuses on the feed-forward neural network where the information is strictly passed forwards. In subsequent chapters, we introduce other types of neural networks.

Figure 1.5a shows an example feed-forward neural network, with an input layer ($\vec{\mathbf{x}} = (x_1, x_2)$), two hidden layers, and a single output. Every neuron performs a set of operations as shown in Figure 1.5b. A summation function ($\sum$) first takes a weighted sum of the inputs multiplied by their respective learnable weights w_i. This is passed into a non-linear activation function φ, returning an output value



(a) **(b)**

Figure 1.5: (a) An example feed-forward neural network with an input $\in \mathbb{R}^2$, two hidden layers $\in \mathbb{R}^4$ and an output $\in \mathbb{R}$. The arrows depict the direction of information flow, with each arrow being assigned a weight. Every circle is a neuron. (b) A single neuron takes in weighted sums of the input and applies a summation function ($\sum$) followed by a non-linear activation function (φ) to produce an output (o) for input to the next neuron.

that serves as the input to the next neuron. Mathematically we can write this process as (for a single neuron):

$$o = \varphi\left(\sum_{i=1}^{n} w_i x_i + b_i\right) \tag{1.8}$$

where b_i is a constant bias term. After every layer of the neural network, we return an output vector that is of the same length as the number of neurons in the layer. Generally, for an input vector $\vec{\mathbf{h}}_{j-1}$, we can obtain the vector for the next hidden layer using:

$$\vec{\mathbf{h}}_j = \varphi(\mathbf{W}_j \vec{\mathbf{h}}_{j-1} + \vec{\mathbf{b}}_j) \tag{1.9}$$

where $\vec{\mathbf{h}}_j \in \mathbb{R}^{n_j}$ is the output vector of layer j, $\mathbf{W}_j \in \mathbb{R}^{n_j \times n_{j-1}}$ is the weight matrix, $\vec{\mathbf{b}}_j \in \mathbb{R}^{n_j}$ is the bias vector, n_j is the number of neurons for layer j, and φ is applied element-wise. For the input layer, $\vec{\mathbf{h}}_0 = \vec{\mathbf{x}}$. $\mathbf{W}_j$ and $\vec{\mathbf{b}}_j$ are learnable parameters.

Note that this transformation would have been linear if we had not applied non-linear activation functions. Without non-linear activations, neural networks

simply become a product of many linear functions, which can be represented by a single linear transformation. The power of neural networks lie in their use of these non-linear functions.

The universal approximation theorem for neural networks states that a network with at least one hidden layer and arbitrary width can approximate certain functions to an arbitrary degree of accuracy (given specific requirements) [27]. However, this does not guarantee that learning algorithms can find this approximate function during training. The optimisation algorithm may not find suitable values for trainable parameters. This may lead to the training procedure learning the "wrong" function and lacking generalisability to unseen cases [28]. We describe the optimisation process in the next section.

1.3.2 Optimisation

Learnable parameters (w_i, b_i) can be initialised through a range of methods. Standard initialisation methods consist of sampling from a uniform distribution or a Gaussian distribution with zero mean and small standard deviation. In order to update parameters, the model makes a prediction for an input $\vec{\mathbf{x}}_i$ and compares the prediction $\hat{y}_i$ to the ground truth y_i using a loss function $\mathcal{L}(\hat{y}_i, y_i)$. As an example, this may be the absolute difference between prediction and ground truth for regression tasks ($\mathcal{L} = |\hat{y}_i - y_i|$) . Note that $\hat{y}_i$ will be a function of the input $\vec{\mathbf{x}}_i$ and all learnable parameters $\boldsymbol{\theta}$ (which includes weights w_i and biases b_i), such that we have $\hat{y}_i = f(\vec{\mathbf{x}}_i; \boldsymbol{\theta})$. The loss function is now given by:

$$\mathcal{L}(\hat{y}_i, y_i) = \mathcal{L}(f(\vec{\mathbf{x}}_i; \boldsymbol{\theta}), y_i) = \mathcal{L}(\vec{\mathbf{x}}_i, y_i; \boldsymbol{\theta}) \tag{1.10}$$

In order to find the optimal values of $\boldsymbol{\theta}$, we need to minimise the loss function with respect to $\boldsymbol{\theta}$ (hence minimising the difference between ground truth and prediction). This is done iteratively through using the gradient descent formula:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \tag{1.11}$$

where η is a tunable learning rate and $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ is the gradient of the loss function with respect to the parameters at iteration t. The gradient calculation and parameter updates are handled by most modern machine learning frameworks, such as PyTorch (using the backpropagation algorithm) [29].

Updating the parameters after every data point is computationally expensive and the gradients computed from individual samples are noisy, providing only a rough approximation of the true gradient over the entire dataset. This can lead to high variance in parameter updates. Hence, training typically processes multiple samples simultaneously in batches of size B, with parameter updates based on the mean of their individual gradients:

$$\overline{\nabla_{\boldsymbol{\theta}}\mathcal{L}} = \frac{1}{B}\sum_{i=1}^{B}\nabla_{\boldsymbol{\theta}}\mathcal{L}_i \tag{1.12}$$

Using the average loss over a small batch in the gradient descent formula is known as mini-batch gradient descent.

In this work, we implement a more complex optimiser known as the Adam optimiser. This is given by the equations [30]:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\epsilon} \tag{1.13}$$

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1-\beta_1^t} \tag{1.14}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1-\beta_2^t} \tag{1.15}$$

$$\mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t) \tag{1.16}$$

$$\mathbf{v}_t = \beta_2\mathbf{v}_{t-1} + (1-\beta_2)[\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)]^2 \tag{1.17}$$

where $\mathbf{m}_t$ and $\mathbf{v}_t$ are estimates of the first and second moments of the gradients respectively, β_1 and β_2 are decay rates for these estimates, ϵ is a small constant to prevent division by zero, and $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are bias-corrected versions of the moment estimates. The loss is still calculated over a small batch size. Adam maintains adaptive learning rates for individual parameters, with frequently updated parameters being adjusted less each update. The use of "momentum" $\mathbf{m}_t$ also allows

the model to escape local minima during training, which it may be stuck in. Generally, Adam has shown to perform well across many tasks and is frequently the optimiser of choice.

Regularisation is normally added to the loss function, in the form of ridge ($\mathcal{L}_{reg} = \mathcal{L} + \lambda \sum w^2$) or lasso ($\mathcal{L}_{reg} = \mathcal{L} + \lambda \sum |w|$), by penalising large weight values (with the summation over all model weights). We also apply other techniques to prevent over-fitting such as dropout.

1.3.3 Model Choice

Various machine learning architectures have been developed to handle different types of data. Inductive bias refers to the architectural assumptions built into a machine learning model that reflect our prior knowledge about the problem. In Chapter 3, we work with image data, hence apply Convolutional Neural Networks, which have built-in assumptions about spatial locality and translation invariance that make them particularly effective at processing visual features. In Chapter 4, we will be introducing Recurrent Neural Networks, which are designed for sequential dependencies, making them well-suited for timeseries data. These architectural choices lead to better performance when the model's inductive biases align with the inherent structure of the data/problem.

1.3.4 Applications to Single Particle Tracking

Machine Learning can be applied to tracking PALM in many ways. We deal with various data types, primarily timeseries, video, and image data. Timeseries data includes the protein position, fluorophore intensity, or other features over the course of its lifetime.

Raw video data is of the single molecule trajectories under laser excitation. Here, we can have machine learning applications in localising or tracking proteins directly from the video. Traditionally, this is done using threshold-based filtering or other techniques that require manual parameter tuning [31].

Localising and tracking proteins involves removing noisy signal. As part of this process, cell masks are produced such that signal outside the cell masks are removed. This is done through the acquisition of a brightfield image. The creation of cell masks offers opportunity for convolutional neural networks to be applied, which has been widely done, not only in tracking PALM, but also for most of biological imaging [32]. This process would be tedious done manually.

We now explore use cases of machine learning for various data formats in tracking PALM while also introducing the data analysis pipeline.

Images

Typically, the start of a tracking PALM experiment begins with the acquisition of a brightfield image. This greyscale image of cells in a medium may be between $400 - 500$ pixels in the x and y dimensions. Various cell segmentation algorithms have been developed that do not implement machine learning approaches. At its simplest, we have threshold-based segmentation. This simply allows the user to set a pixel value, with pixels below the threshold taken as the background and pixels higher taken as the foreground [33]. For more complex edge detection, several advanced techniques can be used: Laplacian filters that convolve the image with specialised kernels [34], clustering algorithms that group similar pixels together [35], and the watershed algorithm, which is guaranteed to produce closed contours [36]. The watershed algorithm treats the image as a surface and traverses along local minima based on pixel intensity. All of these methods are subject to changes in pixel intensities and noise and some may not lead to closed contours. Additionally, they may struggle to find separating boundaries for touching cells. A more successful approach is the use of active contours by software such as MicrobeTracker [37, 38]. This approach tries to minimise a total energy function, which attempts to find a compromise between internal energy (elastic and bending energy controlling the shape and curvature of the contour) and external energy (attracting the contour towards the object boundaries using gradients). One downfall of these approaches is that an initial guess of the contour has to be provided. This is tedious when

imaging many hundreds of cells. All these approaches can also be highly sensitive to the imaging device used and image quality of the brightfield.

To tackle the issues in noise sensitivity, threshold tuning, and initial guess of boundaries, deep learning approaches have been implemented. In this work, we make use of napari-Bacseg [39, 40]. This is a GUI making use of Cellpose, which performs pixel wise semantic segmentation based on convolutional neural networks, particularly a U-Net architecture [32]. Comparable models include Mask-RCNN that work based on region proposal [41]. By training the model on segmented images, the neural network is able to learn which objects in the image correspond to cells. This removes the need for manual segmentation and is able to perform better in the presence of increased noise without the need to tune parameters. However, in this work we do apply some refinement to the resulting masks using active contours as highlighted in Chapter 2. The simple incorporation of deep learning for cell segmentation reduces the data analysis pipeline by hours.

Video

After successful capture and cell segmentation of the brightfield, there is the raw video of protein movement. Localisation is performed in this work using custom in-house software based on bandpass filtering, combined with a threshold for differentiating between background and signal (see Chapter 2). However, with the introduction of machine learning methods in single particle tracking, convolutional neural networks have been implemented to perform localisation from the raw video data [42–44]. For subsequent protein tracking from the video, graph neural networks have been used trace particle motion in time [45].

An issue in single particle tracking using video data and data extracted from it, is the lack of a ground truth. Of particular interest is deducing protein state and behaviour from videos of fluorophore motion (e.g., whether it is confined or bound to the DNA). Given the nature of the experiment, we are not able to know the absolute state of the protein, as we are observing the protein indirectly through the fluorophore. At times, many different biological behaviours are visible in the

raw video but quantifying this without ground truths is difficult (see Chapter 3). Thus, most machine learning models for single molecule use cases are trained on simulations. We shortly discuss the need for simulations.

It can be challenging to evaluate ML models, trained on simulations, on real data. Real experimental data is much more noisy and may not have labels. Sometimes this is done by the ML practitioner for easy-to-label cases. However, this may be subjective and quickly becomes infeasible for complex studies as shown shortly.

Timeseries

After video collection, localisation, and tracking, various timeseries data can be extracted from the Gaussian PSF fit to the protein signal. We obtain data that includes position, brightness (total photon count), background intensity, amplitude, PSF width in the x and y directions, rotation of PSFs, and eccentricity (see Chapter 2). Some of these features may not be useful depending on the task, but all present themself as timeseries over the lifetime of the fluorophore. As mentioned before, knowing the ground truth for these time series is technically challenging as we are observing the fluorophore and not the protein itself. In some cases, the protein's state can be inferred with high confidence. For example, when a DNA-binding protein remains stationary over time, it is likely interacting with the DNA [46]. However, labelling and deducing more complex behaviours such as restricted motion within a larger area can be technically challenging (see Chapter 3 and Chapter 4). Timeseries data obtained from video localisations present a natural use case for ML models. Depending on the task, this may involve the use of any ML architecture, such as recurrent neural networks, feed-forward neural networks, or even more classical techniques involving decision trees and Gaussian mixture models [47, 48]. In Chapter 4, we analyse protein motion through simulations by extracting a set of timeseries features from trajectories, including step size, displacement, and turning angle.

Simulations

The lack of an absolute ground truth does mean simulations, in which the ground truth is known, are a must. While unsupervised learning techniques could bypass the lack of a ground truth, they may prove challenging especially for many labels, and restrict the use of better supervised ML models. Various software has been developed for creating realistic simulations of protein movement and videos [42, 44, 47, 49–51].

The use of simulations for training also addresses the issue of dataset size from real experiments. Due to the curse of dimensionality, larger ML models with more trainable parameters can easily overfit to small datasets, learn noise, and lack generalisability. In order to obtain a generalisable predictive model, dataset size should scale with model size. Obtaining real data from experiments can be very low throughput for particular proteins in tracking PALM, with the added complication of unknown ground truth labels.

As an example, consider RNA Polymerase, a DNA-binding protein. For a healthy *E. coli* cell, the polymerase has a copy number of a few thousand molecules. In a single imaging round ($\sim$ 8 minutes), it may be possible to capture 100 cells within the field of view. Not all of the molecules will be activated per cell. Assume a best case scenario where we obtain roughly 1000 tracks per cell. This allows for a total of roughly $100,000$ protein tracks. For use cases in ML, as done in Chapter 4, we simulate ~ 5 million tracks. To obtain this many tracks, we have to image for another 50 fields of view, taking a total of roughly 7 hours.

Usually this is done over several days and may be performed for different cellular environments to obtain a representative dataset. We also assume everything goes to plan experimentally (e.g., the cells have taken up necessary reagents, are healthy, and the fluorophore works as expected). For the case of RNA Polymerase, this may seem feasible to an experimentalist, but will be quite time-consuming.

However, now consider the case of proteins such as sigma-54 or sigma-70 which are present in the cell with copy numbers $100\times$ and $10\times$ smaller respectively (see Chapter 2). This requires anywhere between 10-$100\times$ more experimentation for the same number of tracks. This gets worse for proteins that are only present

with copy numbers below 10, such the *lac* repressor [52]. Obtaining enough data from experiment for these proteins is not feasible. This simple argument highlights the need for simulations as part of the training process *in vivo*, unless there is a breakthrough in experimental approaches. Without simulations, the dataset size can limit the application of ML for studying the majority of proteins, which have smaller copy numbers. Fine-tuning or evaluation on real data may be possible after initial training on simulations. Note that absolute ground truth labels (e.g. biological state) are not known from experiment, so careful labelling of proteins would also have to be done.

Another issue present in all data are autofluorescent molecules. Shining laser light at cells naturally excite molecules that are not the target fluorophore. We found autofluorescent molecules to only contribute $\sim 10 - 20$ molecules per cell from running control experiments of *E. coli* cells without any fluorophores. For proteins that are naturally abundant in the cell, this addition is not an issue, especially if additional filtering steps are included. However, for very low copy number proteins, it is both difficult and essential to remove these faulty signals, as they can make up the majority of the dataset if not handled carefully. This can lead to artefacts, and incorrect results being reported, alongside the majority of the dataset consisting of false positives.

Figure 1.6 shows the entire tracking PALM data analysis pipeline for reference. Machine learning can be integrated at every stage of the pipeline, but care must be taken as outlined above. This pipeline is followed in Chapter 2 when characterising proteins involved in transcription.

1.4 Overview of Thesis

In this section, we have provided a quick introduction into super-resolution microscopy, tracking PALM, machine learning, and its applications to single particle tracking.

From hereof, this thesis is split into three parts. Chapter 2 applies tracking PALM analysis to obtain ensemble diffusion distribution for two poorly understood

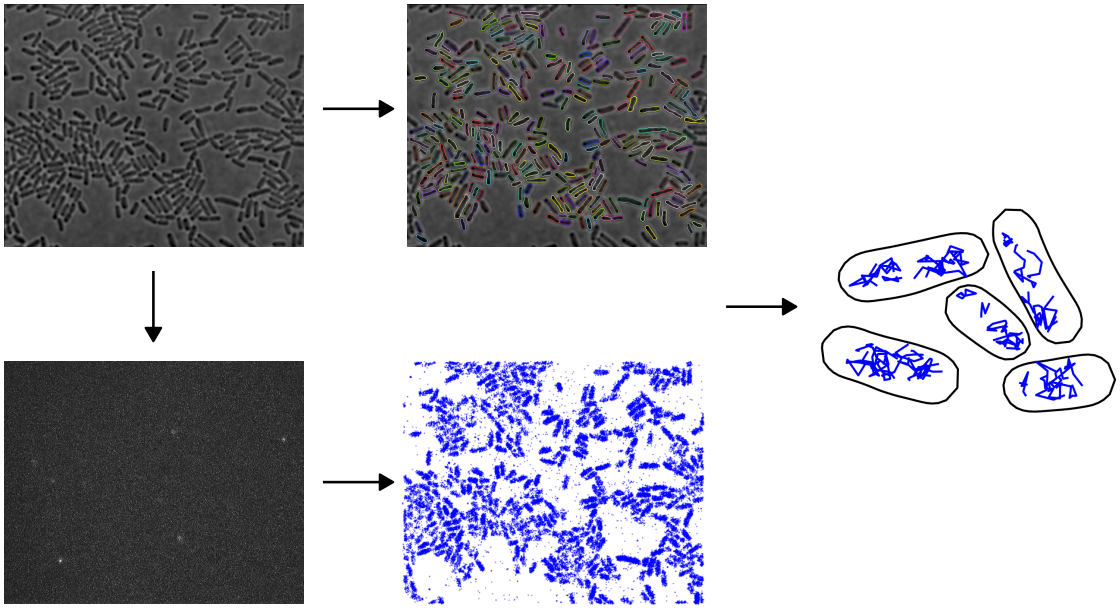


Figure 1.6: Data analysis pipeline from a PALM experiment. We acquire a brightfield and darkfield videos of protein movement under laser excitation. Using a custom localisation algorithm, we detect protein spots. The brightfield also undergoes a cell segmentation algorithm [38, 39]. Using the cell masks and localisations, proteins only inside the cells are tracked. Note there is background noise in places where there are no cells. See Chapter 2 for further details. Machine learning techniques can be incorporated at various stages. In Chapter 3, we deal with video data, whereas in Chapter 4, we deal with time series data (protein coordinates post video localisation).

proteins, sigma-54 and sigma-70. This sheds light on new mechanisms of gene regulation and transcription mechanisms *in vivo* for *E. coli* cells. In Chapter 3, we develop a convolutional neural network for calculating binding times of these proteins, alongside initial verification of our method with DNA Polymerase I. Here, we apply PALM at long exposures and provide a quantitative threshold for distinguishes long and short exposure times. In Chapter 4, we develop another novel ML architecture, based on recurrent neural networks, performing frame level inference at any exposure time for proteins diffusing through complex cellular environments. This chapter is based on proteins moving in fractional Brownian motion, in which consecutive time-steps are no longer independent. We provide a model that performs regression and classification for the determination of diffusion coefficients, anomalous exponents, and biological behaviour of diffusing proteins.

All ML models developed in this thesis are easily generalisable, as they are not

specific to tracking PALM, the protein-fluorophore fusion, or *E. coli* cells. Each chapter is unique in its own right, having its own introduction and methods, which are essential in understanding the work.

2

Photoactivated Localisation Microscopy of Sigma Factors *in vivo*

Contents

2.1 Introduction 25
2.1.1 Brief Overview 25
2.1.2 Background on Transcription 25
2.1.3 Background on Sigma Factors 27
2.2 Methods 29
2.2.1 Bacterial Strains 29
2.2.2 Cell Preparation 29
2.2.3 Microscope 31
2.2.4 Localisation and Tracking 32
2.2.5 Mapping Localisations onto Heatmaps 33
2.2.6 Calculating Diffusion Coefficients 34
2.2.7 Fitting Sub-populations 35
2.3 Results 36
2.3.1 Diffusion Properties in Rich M9 36
2.3.2 Heatmaps for Spatial Distribution in Rich M9 42
2.3.3 Nitrogen Starvation 44
2.3.4 Overnight Stationary 47
2.3.5 Serine Hydroxamate Treatment 49
2.4 Discussion 50

2.1 Introduction

2.1.1 Brief Overview

This chapter dives into single molecule imaging, using tracking PALM, of the sigma factors, sigma-54 and sigma-70. Sigma factors are essential machinery involved in transcription initiation, and we begin with an overview of their roles. Little is understood about their movement *in vivo*, and we present experimental data in the form of diffusion distributions. This provides some initial hypotheses and a foundation into the complex mechanisms governing sigma factors. We particularly focus on cells under stress and compare this to a baseline condition. This is the first time mobilities of sigma-70, and more interestingly sigma-54, have been characterised *in vivo*. We also attempt at an understanding of the interactions between the sigma factors with the core RNA Polymerase. Through the use of ensemble diffusion plots, we identify subpopulations and thereby assign biological behaviours. This chapter also lays the foundation for the following chapters, in which we further dive into computational approaches for diffusion analysis of single particle tracks.

2.1.2 Background on Transcription

The Central Dogma of Molecular Biology describes how genetic information flows within biological systems. DNA is transcribed into RNA, which is then translated into proteins [53, 54]. Transcription, the initial step in this process, involves the reading of genetic code from DNA for downstream protein synthesis and cellular responses.

Transcription involves the recruitment of many proteins inside the cell. Of note, is RNA Polymerase (RNAP), the main protein responsible for reading the DNA strand and catalysing the synthesis of RNA [55–58]. This molecular machine orchestrates several critical steps: it binds to specific DNA sequences known as promoters through the help of sigma factors, unwinds the DNA double helix to access the template strand, and synthesises RNA while maintaining low error rates ($\sim$ 1 nucleotide error for every $10,000$) [59–61]. The promoter is a specific DNA

sequence, usually located between -10 and -35 base pairs upstream of the actual transcription start site [62–66]. It acts as a molecular landing pad for the RNAP to position itself and begin transcription. However, locating the promoter sequence in the vast space of genetic code is not solely performed by RNAP. Proteins known as sigma factors bind to RNAP, forming what is known as the RNAP holoenzyme [23, 60, 61, 67–70]. The RNAP holoenzyme, comprised of the core RNAP and a sigma factor, is then able to locate the promoter site and bind to the DNA [71–75]. The sigma factor essentially acts as a guide, bringing the polymerase to the correct location and preventing it from binding at random to the DNA strand.

The point at which the holoenzyme engages with the promoter is referred to as the closed complex [73, 76–78]. The DNA strand remains base paired in its double helix configuration, but contact is made between the sigma factor of the holoenzyme and the promoter site [79]. The RNAP has not yet engaged with the DNA at this point. Through complex conformational changes, the local DNA strand, roughly 14 nucleotides, is unwound by the holoenzyme [80–82]. This exposes single DNA strands, ready to be transcribed by the polymerase, and leads to the open complex, characterised by a stable transcription bubble. The transcription bubble is maintained by protein-DNA interactions within the holoenzyme, and protects the single DNA strand from nuclease digestion [73, 76–78, 80–84].

Following open complex formation, transcription enters an initial phase marked by abortive initiation. During this phase, RNAP remains firmly attached to the promoter via its sigma factor and repeatedly synthesises and releases short RNA transcripts, typically 2-12 nucleotides in length. These abortive transcripts result from tension created as RNAP attempts to move forward while still tethered to the promoter. This process continues until sufficient energy accumulates to break the promoter contacts, leading to promoter escape. The RNAP has now broken contact with the promoter site, ready to engage into productive transcription [60, 76, 78, 80–83, 85–90].

These structural changes weaken the binding of the sigma factor and RNAP. At this point, the sigma factor dissociates and the core RNAP enters its elongation

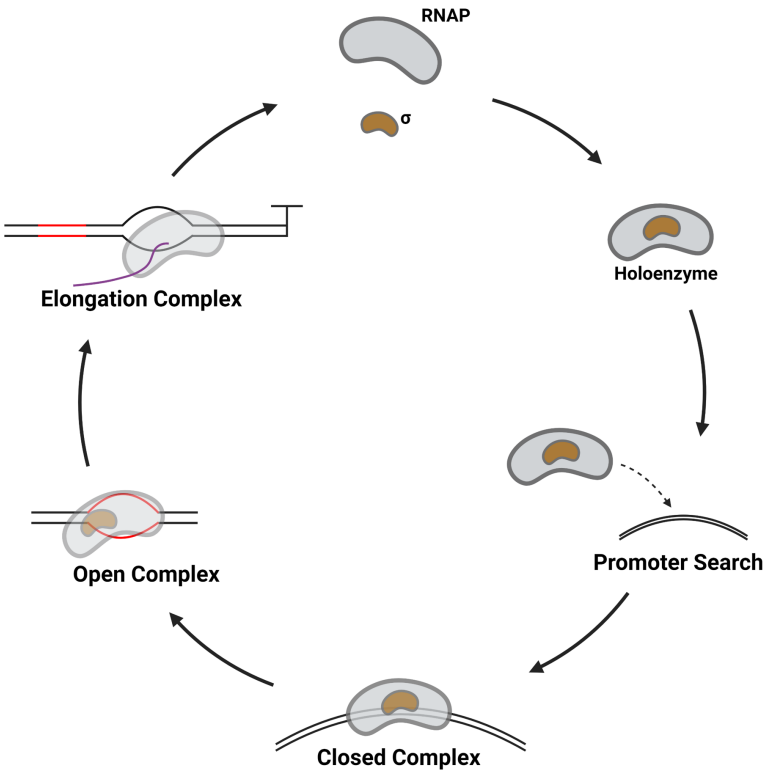


Figure 2.1: The process of transcription. RNAP and a sigma factor (e.g. sigma-70) form the RNAP holoenzyme, with the sigma factor guiding the RNAP to the correct starting site (promoter search). Initial contact with the DNA strand leads to the closed complex. Subsequent conformational changes lead to the open complex, exposing single DNA strands (open promoter site shown in red). Once RNAP enters elongation (RNA shown in purple), the sigma factor disassociates and is recycled. Figure created using BioRender.

phase. The RNAP now successfully moves along the template DNA strand, synthesising RNA, while the released sigma factor is recycled, ready to guide another RNAP molecule to a promoter site [71, 84, 91–98]. This entire process is depicted in Figure 2.1.

2.1.3 Background on Sigma Factors

As highlighted in the previous section, sigma factors play a crucial role in guiding the RNAP to the promoter site. In this work, we focus on sigma factors in *E. coli*, which has a total of seven sigma factors, each playing a different role [99]. Specifically, the two sigma factors of interest are sigma-70 (σ^{70}, also known as RpoD) and sigma-54 (σ^{54}, also known as RpoN). The number associated with

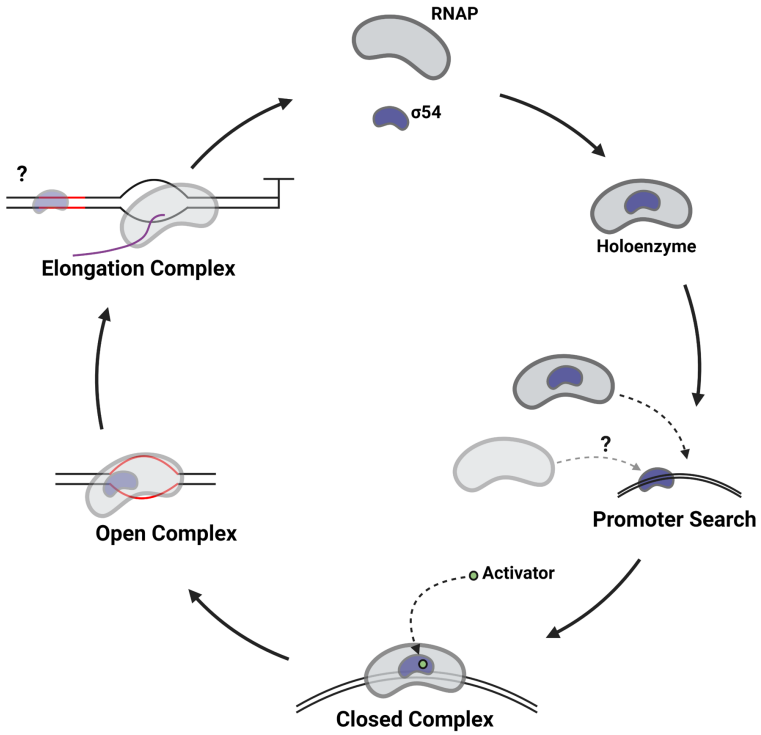


Figure 2.2: The process of transcription initiation for σ^{54}. Unlike the housekeeping σ^{70} which follows the previous schematic, σ^{54} has the ability to bind to DNA independently of RNAP. This raises some mechanistic questions, such as whether RNAP can find the bound σ^{54} during promoter search or if the holoenzyme rests in a ready-to-respond state, and if σ^{54} can stay attached to the promoter site once RNAP enters elongation. σ^{54} requires an activator protein for open complex formation. As before, the RNA is shown in purple in the elongation complex and the promoter site in red. Figure created using BioRender.

each sigma factor is its molecular weight in kDa.

σ^{70} is the primary housekeeping sigma factor, handling essential gene expression for cellular functioning (e.g. protein synthesis machinery) [100, 101]. In contrast, σ^{54} is primarily responsible for cellular stress during nitrogen starvation, guiding the core RNAP to promoter sites responsible for transcribing nitrogen-assimilating genes [102, 103].

σ^{54} exhibits several distinct characteristics that differentiate it from σ^{70}. Unlike σ^{70}, which requires incorporation into the holoenzyme complex to bind DNA, σ^{54} has the unique ability to bind DNA independently or potentially as the holoenzyme in a "ready-to-respond" fashion [99, 104, 105]. The σ^{54} holoenzyme cannot initiate open

complex formation and thus transcription as specific bacterial activator proteins are required, such as NtrC or PspF (in the case of the holoenzyme, σ^{54} would sit at its respective promoter with the core RNAP awaiting activation, with contact to the DNA only through the sigma factor) [102, 103, 106]. This adds an additional layer of regulatory control. The schematic depicted in Figure 2.1 is not entirely correct for σ^{54}, and a different schematic for transcription initiation using σ^{54} is shown on Figure 2.2. Unlike σ^{70}-dependent genes, promoter sites for σ^{54} are also located at the -12 and -24 elements upstream of the transcription site [107, 108]. We also highlight a difference in their cellular abundance (copy number), with σ^{54} at times present with approximately an order of magnitude fewer copies per cell than σ^{70} (see Results).

The observation that σ^{54} can bind to promoter sites independently of RNAP raises some intriguing mechanistic questions. These are whether it first binds to promoter sites and subsequently recruits RNAP, or if it first associates with RNAP while diffusing in the cytoplasm and binds onto sites in a "ready-to-respond" state. These changes are also highlighted in Figure 2.2.

2.2 Methods

2.2.1 Bacterial Strains

PAmCherry fusion strains of σ^{70} and σ^{54} were constructed in *E. coli* MG1655, with σ^{70} using lambda Red recombination and σ^{54} using plasmid mediated genome insertion. Both fusions were expressed from their native promoters.

2.2.2 Cell Preparation

Strains of σ^{70} and σ^{54} were streaked onto lysogeny broth (LB) plates. Single colonies were placed into LB media and grown at 37 °C overnight. Overnight cultures were diluted in Rich M9 (unless otherwise indicated) and grown for >2 h at 37 °C to early exponential phase (OD $<$ 0.2). Cells were centrifuged (8000 revolutions per minute for 3 minutes) and immobilised on 1% low fluorescence agarose pads. The agarose pads contain the same medium in which the cells were diluted in unless

otherwise stated. Glass coverslips (thickness 1.5) were autoclaved in a furnace for 1 hour at 500 °C to remove any fluorescent background particles.

In order to isolate various biological behaviours and further study mobilities of the sigma factors, we put the cells under various environments. We perform the below perturbation experiments.

Rich M9 (Baseline)

After initial overnight growth in LB, a small amount of the solution was diluted in Rich M9 for 3-4 hours before imaging. Rich M9 media contains 10 ml of 5× M9 salts, 0.5 ml of 100× MEM vitamins, 1 ml of 50× MEM Amino Acids, 100 μl of 1 Molar (M) $MgSO_4$, 5 μl of 1 M $CaCl_2$ and 0.5 ml of 20% glucose. The remaining solution is filled to 50 ml using Milli-Q water.

The growth condition in Rich M9 media returns the baseline distribution for both σ^{54} and σ^{70}. Rich M9 provides favourable conditions for the cells, with both sigma factors not affected by stress or a lack of nutrients.

Nitrogen Starvation

Nitrogen Starvation was induced using a modified medium. This medium contains 10 ml of 1× Phosphate Buffered Saline (PBS), 100 μl of 1 Molar (M) $MgSO_4$, 5 μl of 1 M $CaCl_2$ and 0.5 ml of 20% glucose. Again, we fill to 50 ml using Milli-Q water. No M9 salts or amino acids are added, which starves the cells of nitrogen. Cells were grown overnight in LB but diluted in this modified medium.

Under nitrogen-limiting conditions, there is enhanced activation of σ^{54} due its nitrogen assimilating functionality. We expect to induce a change in σ^{54} activity.

Serine Hydroxamate

Serine Hydroxamate (SHX) was diluted in the agarose pad on which the cells were placed on. Cells are grown in LB overnight and diluted in Rich M9, but placed on agarose pads containing 10 mM SHX 30 minutes before imaging. SHX induces amino acid starvation in cells, similar to nitrogen starvation.

Poor M9

Poor M9 contains 10 ml of 5× M9 salts, 100 μl of 1 Molar (M) $MgSO_4$, 5 μl of 1 M $CaCl_2$ and 0.5 ml of 20% glucose. We fill to 50 ml using Milli-Q water. Poor M9 has the same contents as Rich M9, with the exception of amino acids and vitamins. Cells are diluted in Poor M9 but grown in LB overnight as described before.

Stationary Phase

To induce stationary phase, cells are grown in Rich M9 (not LB) overnight. This medium does not contain enough nutrients for overnight growth, such that cells experience heavy stress. The agarose pads are created from the same medium that the cells were grown in. As this medium now contains cells, a small amount is centrifuged and the supernatant is filtered (0.25 μm filter size) to remove any cells. This ensures cells do not uptake any nutrients from new agarose pads during imaging.

2.2.3 Microscope

All single molecule tracking PALM and brightfield images were acquired using a custom-built microscope (Troodos), equipped with a 200 mW 405 nm diode laser and a 200 mW diode-pumped solid-state 561 nm laser. The lasers were modulated using a DAQ system, with the 405 nm modulated directly via analogue voltage commands, and the 561 nm laser using an acousto-optic modulator. A custom-built LabVIEW virtual instrument software was written for laser modulation. The lasers were coupled into single mode optical fibers, collimated, reflected by a multibandpass optical filter, and focused by an achromatic doublet lens onto a single point on the back focal plane of a 100× NA=1.40 oil immersion microscope objective. Exit angle modulation enabled sample illumination in HILO. Brightfield images were illuminated using a white LED light source. Collected light was passed back through the multi-notch filter with transmission windows at 439±15 nm, 521±17 nm, and 605±25 nm, with an achromatic doublet lens forming an image on an EMCCD camera. Image acquisition was performed using the software package Andor SOLIS. Tracking PALM comprised of 561 nm excitation at 340 W/cm^2 and

405 nm excitation at 0-1 W/cm^2. Imaging parameters involved a pre-amplified gain of 1, a memory parameter of 1, and an exposure time of 15.11 ms over 30,000-80,000 frame recordings [109].

2.2.4 Localisation and Tracking

Localisation of proteins from video data is performed using in-house custom software called "StormTracker" (written by Stephan Uphoff, University of Oxford). This operates per frame of the video. Briefly, we apply a bandpass filter by convolving the image with a normalised Gaussian kernel $G(x) = \frac{1}{N}\exp\left(-\frac{x^2}{4}\right)$ where $N = \sum_{x=-5}^{5}\exp\left(-\frac{x^2}{4}\right)$. From this convolved image, we subtract the convolution of the original image with a boxcar kernel of size 7. The Gaussian kernel smooths high frequency noise while matching the PSF shape, and the boxcar subtraction removes low frequency background variations. Subsequently, a manual threshold is chosen to detect protein signal peaks. Each protein localisation is then fit to a 2D elliptical Gaussian Point Spread Function given by:

$$f(x,y) = A\exp\left(-\frac{x'^2}{2\sigma_X^2} - \frac{y'^2}{2\sigma_Y^2}\right) + B \tag{2.1}$$

where A is the amplitude, B is the background offset, and σ_X and σ_Y are the standard deviations along the minor (x') and major (y') axes respectively (we define $\sigma_X < \sigma_Y$). The function is fitted over a 15×15 pixel region, cropped from the entire frame, with the peak position at its centre. The rotated coordinates x' and y' are given by:

$$x' = (x - x_0)\cos\theta - (y - y_0)\sin\theta \tag{2.2}$$

$$y' = (x - x_0)\sin\theta + (y - y_0)\cos\theta \tag{2.3}$$

where (x_0, y_0) is the centre peak position, x and y are the original (non-rotated) coordinates along the cropped image's horizontal and vertical axes respectively, and θ is the angle from the $y-$axis to the major axis of the ellipse, measured counter-clockwise.

Once proteins have been localised, they are tracked using in-house custom software called "LoColi" (written by Mathew Stracy, University of Oxford). This joins protein localisations in consecutive frames within a window size of 7 of its previous localisation. We use a memory parameter of 1, which is the maximum time a protein can be in an OFF (dark) state, while still being considered part of the same track. This process is only performed for tracks inside cells and required additional cell masks, generated using napari-BacSEG, an open-source cell segmentation software [39]. Masks are refined using MicrobeTracker, another tool for cell segmentation [38]. The refinement process led to sharper masks. Protein tracks were only used for downstream analysis if they consisted of a minimum of 4 localisations [15].

2.2.5 Mapping Localisations onto Heatmaps

Heatmap construction is performed using in-house custom software "LoColi". Defining a new coordinate system along the width and height of a cell allows us to overlay varying length cells. For any cell, we define the total length of the cell to be between 0 and 1 (from pole to pole), and the width to vary from -1 to 1. The centre of the cell is then given by (0.5, 0). As all cells now have a comparative coordinate system, we can overlay spatial localisations in the cell to see where the majority of protein localisations lie. However, since the cell has two lines of symmetry, along its width and height, we can split it into four quadrants. A given localisation can then be found in three other positions. To take into account such cases, which can easily occur if the cell has been imaged in a rotated position, we mirror a given localisation in all quadrants. This returns heatmaps of spatial organisation of a given protein across many cells. We can further categorise these heatmaps into diffusive bands, showing the distribution of mobilities for a given cell length. All heatmaps are shown with 15 bins along the width of the cell and 60 bins along the length of the cell. The typical healthy *E. coli* cell has a width of 0.5 μm and a length of 2 μm (roughly 4$\times$ longer than wide).

2.2.6 Calculating Diffusion Coefficients

The mean square displacement (MSD) (one time step) is calculated using:

$$MSD = \frac{1}{N-1}\sum_{i=1}^{N-1}[(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2] \tag{2.4}$$

where x_i and y_i are protein coordinates at frame i and N is the the total number of frames the protein is active [46].

Diffusion coefficients are calculated per protein track. For proteins moving in two dimensions, and using the above MSD, this is given by:

$$D^* = \frac{MSD}{4\tau_{\text{exp}}} \tag{2.5}$$

where D^* is the apparent diffusion coefficient and τ_{exp} the imaging exposure time. D^* will be affected by the localisation error, resulting in a larger observed diffusion coefficient. The correction to the diffusion coefficient due to localisation error (σ_{loc}) is given by $-\sigma_{loc}^2/\tau_{\text{exp}}$.

A given protein trajectory can transition between biological behaviours within a single track (e.g. from free σ^{70} to a holoenzyme). As we are imaging at short exposure (15.11 ms), the trajectories are short-lived in time, such that transitions between behaviours are not likely since the fluorophore bleaches quickly. We also apply trajectory truncation to enforce this, taking only the first eight points of the trajectory for apparent diffusion coefficient calculation. Thus, combined with the short track duration, we ensure that each diffusion coefficient represents the mobility of a single biological behaviour, rather than an average across multiple behaviours, which would affect downstream D^* distributions. Given the imaging parameters, fluorophores bleach on average in $\sim$ 130 ms, much shorter than the time spent in unique biological states (see later).

Note that we use the term "apparent" as the protein can move in all three dimensions yet we observe it in two dimensions. The protein can move into or out of the plane, leading to a smaller observed displacement. The exact diffusion values may be subject to correction, through the use of simulations, although this is not

performed in this work. What is of greater importance is the comparison between the various relative fractions and mobilities of different protein sub-populations.

2.2.7 Fitting Sub-populations

A given protein tagged with a fluorophore can be found in various unique biological states. Each of these states lead to different measured diffusion coefficients. When plotting the distributions of the diffusion coefficients, the total histogram will be a sum of the individual underlying states. Each physical state is modelled with its own characteristic function given by [19]:

$$f_{D^*}(x; D^*, n) = \frac{(n/D^*)^n x^{n-1} e^{-nx/D^*}}{(n-1)!} \tag{2.6}$$

where D^* is the apparent diffusion coefficient and n is the minimum number of frames of a protein trajectory ($n = 4$ in our work). For the case of N underlying states, the total histogram of diffusion coefficients is given by:

$$f_{D^*}(x; D_1^*, ..., D_N^*, A_1, ..., A_N) = \sum_{i=1}^{N} \frac{A_i (n/D_i^*)^n x^{n-1} e^{-nx/D_i^*}}{(n-1)!} \tag{2.7}$$

where A_i is the fraction of sub-population i present in the data with diffusion coefficient D_i. Intuitively, random protein displacements can be modelled as samples from a normal distribution due to Brownian motion. Since D^* distributions are estimated from the MSD, which is proportional to squared displacements, they follow the above gamma distribution.

Experimentally it is very difficult to observe a high number of states ($N \geq 4$). Typically, for PALM experiments, we observe a maximum number of three states. In subsequent analysis, we will make use of Equation 2.7 for the case of $N = 3$, with example fitting for $N = 1$ and $N = 2$.

For a $N = 3$ state system, there are a total of five free parameters. These are A_1, A_2, D_1, D_2 and D_3 with A_3 give by $A_3 = 1 - A_1 - A_2$. Fitting Equation 2.7 with $N = 3$ to diffusion distributions usually leads to physically impossible values of A_i or D_i as the search space is large. To counter this, we fix the search space for one of the D^* values to lie between 0.05 to 1.5 μm^2/s. The mobility of the

DNA-binding proteins which we are studying lie in this range, so we are guaranteed at least one of the states will have a mobility in this range. We iteratively search this space in a grid-like manner and calculate the residual of the data using Equation 2.7. The parameters with the lowest squared residual ($R = \sum(y_{\text{data}} - y_{\text{fit}})^2$) is taken to represent the apparent diffusion coefficients for each of the N states. For error estimation in these parameters, we use the covariance matrix. The covariance matrix is calculated as:

$$\text{Cov} = \frac{R}{n-p} \cdot (J^T J)^{-1} \tag{2.8}$$

where J is the Jacobian matrix of the fit, n is the number of non-zero data points, and p is the number of fitted parameters. The standard errors are then calculated as the square root of the diagonal elements of the covariance matrix. The 95% confidence intervals are obtained by multiplying these standard errors by 1.96, assuming normally distributed errors. For the constrained fraction (A_3), the error is propagated using $\text{error}_{A_3} = \sqrt{(\text{error}_{A_1})^2 + (\text{error} A_2)^2}$.

Curve fitting was performed in MATLAB using non-linear least-squares regression (`lsqcurvefit`). The fitting uses MATLAB's default *trust-region-reflective* algorithm, which iteratively searches for optimal parameters within a dynamically adjusted trust region and aims to minimise the residual sum-of-squares.

2.3 Results

2.3.1 Diffusion Properties in Rich M9

Rich M9 acts as our control against perturbations to the cell. Before calculating diffusion coefficients, we provide some examples of tracks inside live *E. coli*, colour-coded by their mobilities for both σ^{70} and σ^{54}. These are shown in Figure 2.3, obtained through tracking PALM [11]. We further obtain distribution of cell lengths for both sigma factors, as shown in Figure 2.4. The majority of cells have lengths greater than 2 μm, which is to be expected for *E. coli* cells grown in favourable conditions.

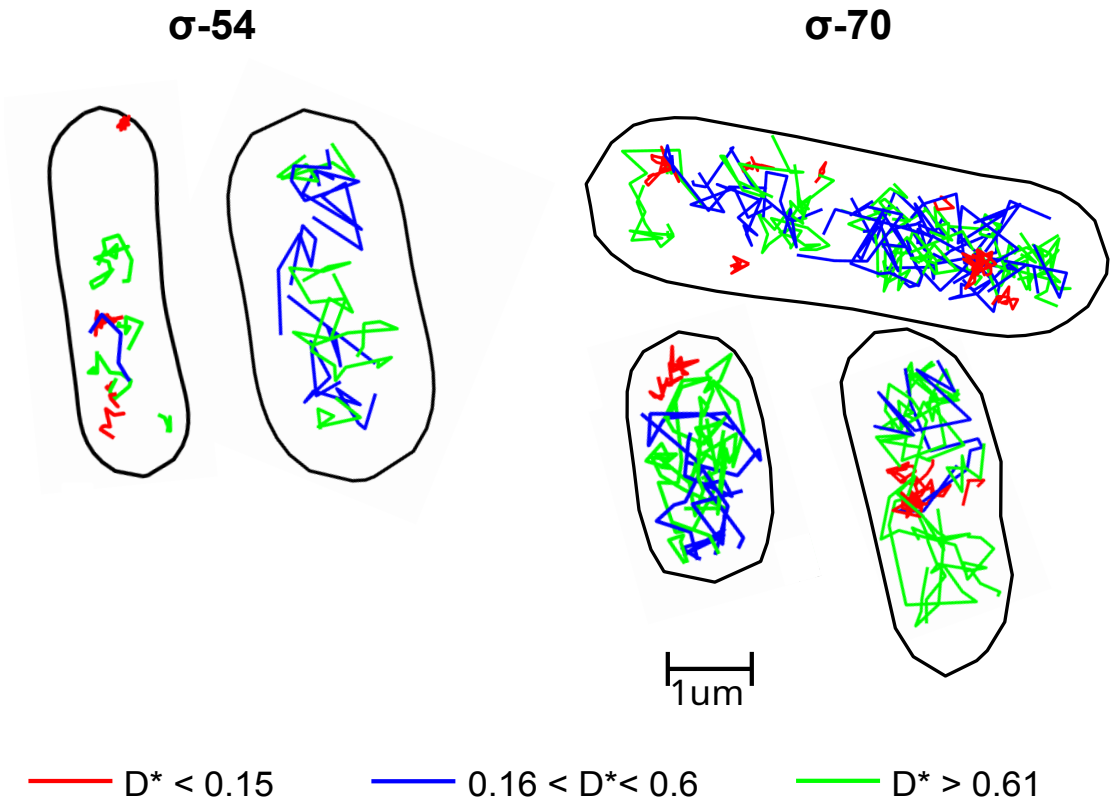


Figure 2.3: Tracks of sigma factors from live *E. coli* cells colour-coded by their mobility. D^* is in $\mu m^2/s$.

From Figure 2.3, we can observe various mobilities of the sigma factors for example cells, with some proteins searching a wider region. Imaging at 15.11 ms exposure time, we collect data for multiple field-of-views (FOVs) of cells. A FOV represents a 512×415 pixels region in which we record videos of protein movement (1 pixel = 0.0957 μm), usually containing $50 - 150$ cells (dependent on growth conditions).

For σ^{70}, we collect a total of 15 FOVs ($\sim 15 \times 8$ mins = 120 mins imaging time across 5 days), giving 2610 cells and 64435 single molecule tracks. Similarly, we collect 17 FOVs across 6 days for σ^{54}, giving 30507 single molecule tracks from 2691 cells. For both sigma factors, the average track length was ~ 9 frames, such that every σ^{54}-PAmCherry and σ^{70}-PAmCherry fusion was active for ~ 135 ms.

Within the imaging time, we plot the frequency histogram of sigma factor single molecule tracks as a measure of copy number. The copy number distributions are shown in Figure 2.5. For σ^{70}, we obtain a mean copy number of 35 with a

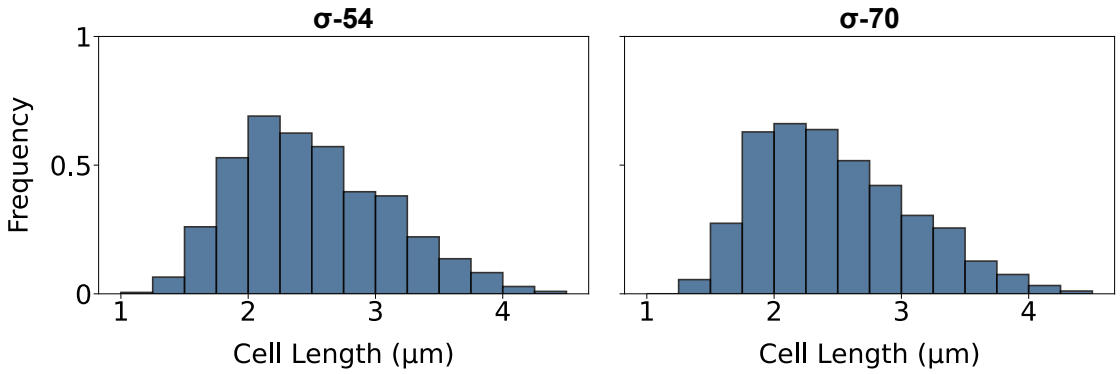


Figure 2.4: Cell length distribution for 2989 and 2622 cells grown in Rich M9, for σ^{54} and σ^{70} respectively. The majority of cells lie above 2 μm, the typical length for *E. coli* cells.

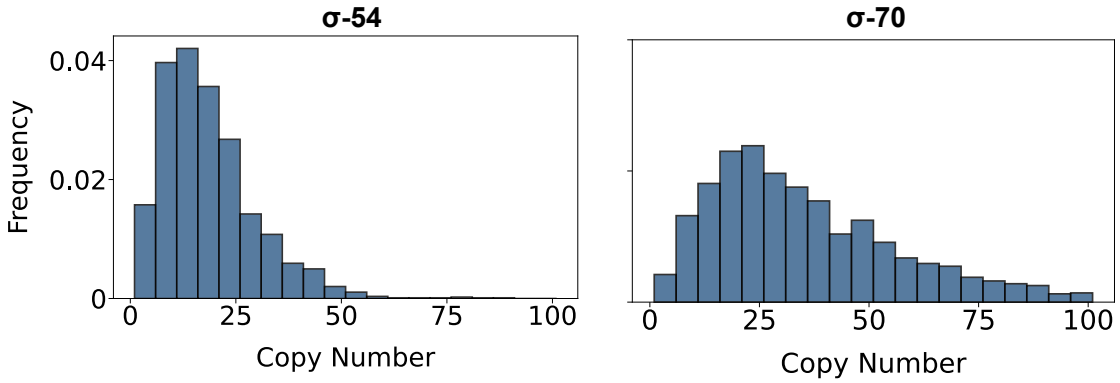


Figure 2.5: Copy numbers distribution for cells grown in Rich M9 with cell lengths between 2-3 μm. σ^{70} copy number distribution shifts to the right as it has many more copies per cell than σ^{54}.

standard deviation of 20 across 1426 cells. For σ^{54}, the mean copy number is 18 with standard deviation of 11 across 1689 cells. Cells with length between 2-3 μm were only included, such that all cells were reasonably healthy. At times, σ^{70} roughly has 10× more copies per cell than σ^{54}, with the maximum number of molecules in a given cell reaching 100 within the imaging time. This also explains the larger number of tracks for σ^{70} despite fewer cells.

Subsequently, for all tracks of each sigma factor, we calculate its diffusion coefficient (following Section 2.2.6) and plot the resulting D^* distribution as normalised frequency histograms, as shown on Figure 2.6. The distributions for σ^{54} and σ^{70} are side-by-side for comparison. The D^* distributions are fitted using Equation 2.7, for $N = 1, 2$, and 3. A one- or two-state fit did not fit the data

very well as shown in the figure. The lowest residual and best fit for the data, evident on the figure, is given by $N = 3$ for both sigma factors. This suggest three biological behaviours underlie the distribution of mobilities for both sigma factors. In reality, each biological behaviour can be sub-divided into further behaviours but these cannot be resolved from diffusion distributions. We now tentatively assign different biological activity to species with different mobilities, similar to population assignment performed by El Sayyed *et al.* for the protein NusG [109].

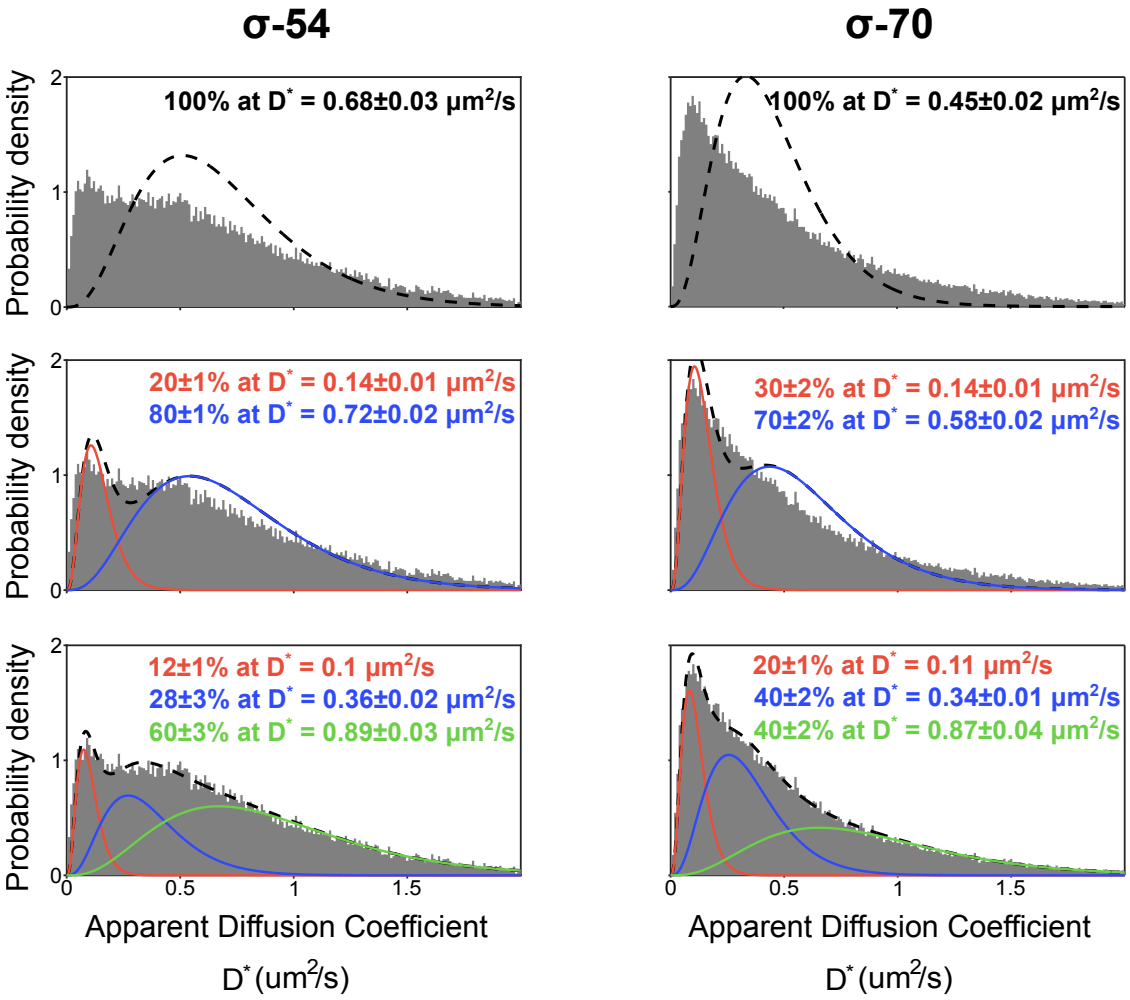


Figure 2.6: Apparent diffusion distributions for σ^{54} (left) and σ^{70} (right) in Rich M9. We test a one-, two-, and three-species fit. The lowest residual and best fit is obtained using a three-species fit representing bound sigma factors (red), holoenzymes (blue) and free sigma factors (green). The sum of squared residuals over all data points from left (σ^{54}) to right (σ^{70}) are as follows: 5.1 and 7.0 for one state fits, 2.0 and 2.2 for two state fits, and 1.2 and 1.0 for three state fits.

We first observe a **very slow** moving population, given by the red curve on

Figure 2.6 for the D^* distribution using $N = 3$. These represent 12% and 20% of the total population sizes for σ^{54} and σ^{70} respectively. The localisation error is 0.04 μm and for an exposure time of 15.11 ms, we obtain $\sigma_{loc}^2/\tau_{\text{exp}} \approx 0.1$ $\mu\text{m}^2/\text{s}$. Thus, the **very slow** moving population is within localisation error and hence essentially static. Note that the movement due to the chromosome for an average time of 135 ms is negligible [110].

As σ^{70} cannot bind to the chromosome without being in the holoenzyme, this population corresponds to the holoenzyme stably bound at the promoter site, before active elongation. On the other hand, σ^{54} is able to bind to the chromosome without the need for a holoenzyme. Thus, the **very slow** moving population for σ^{54} may largely consist of stably bound sigma factors at their respective promoter sites. As also mentioned later, deletion of σ^{54} from uropathogenic *E. coli* cells has been shown to lead to transcription down-regulation of ten nitrogen-related genes [99]. Thus, we may expect a small amount of holoenzymes to form with σ^{54} in Rich M9, as these genes can only be transcribed by σ^{54} (it is important to note this study was performed in pathogenic *E. coli* which may have induced stress). It is unlikely for the σ^{54}-holoenzyme population (transcribing at the promoter site) to represent the majority of its **very slow** population as NtrC (activator protein) is needed for σ^{54} to initiate transcription. This is not readily available in rich media.

Work by Switzer *et al.* also showed deletion of σ^{54} in pathogenic *E. coli* leads to transcription up-regulation of roughly 150 genes [99]. Thus σ^{54} bound at the promoter sites is likely to repress the expression of genes not needed in rich media, which would otherwise be transcribed by the σ^{70}-holoenzyme. Although each sigma factor has its unique promoter elements upstream of the transcription site, σ^{54} may act to repress σ^{70}-dependent genes, acting as a gene regulation mechanism (blocking the σ^{70}-holoenzyme from finding the promoter site) [111]. This **very slow** moving population with $D^* \sim 0.1$ $\mu\text{m}^2/s$, based on the D^* distribution, formulates the first one of our three species.

Next, we observe a **slow** moving population as shown on the blue curve on Figure 2.6, with relative population sizes of 28% and 40% for σ^{54} and σ^{70} respectively.

This species can be explained using our earlier schematics (see Figure 2.2 and Figure 2.1), as representing the mobility of the free holoenzyme. The holoenzyme consists of the core RNAP and a sigma factor, leading to relatively faster mobilities as it searches for the promoter site. The D^* values for this state are roughly the same for both sigma factors ($\sim 0.35\ \mu\text{m}^2/\text{s}$). As the core RNAP is $\sim 500\ kDa$, it is an order of magnitude bigger than both σ^{54} and σ^{70}. Thus, the holoenzyme apparent diffusion coefficient is largely dictated by the free diffusion coefficient of RNAP. This has been measured by Stracy *et al.* as being $\sim 0.36\ \mu\text{m}^2/\text{s}$, in line with the holoenzyme populations on Figure 2.6 [19]. Note that both sigma factors compete for the same position on the core RNAP in order to form the holoenzyme. Since σ^{70} has a copy number an order of magnitude higher, most holoenzymes will likely include core RNAP and σ^{70}. Thus, σ^{54} is outcompeted by σ^{70} during holoenzyme formation.

Lastly, we have the **fast** moving population (green curve on Figure 2.6), representing the free diffusion of the respective sigma factors. These are lone sigma factors diffusing through the cell cytoplasm, in search for the core RNAP. Since the sigma factors do not differ largely in size, the D^* values for the free state is similar. A greater percentage of σ^{54} (60%) is free as compared to σ^{70} (40%), as it is not actively involved in transcription and is outcompeted during any potential holoenzyme formation. Interestingly, the free apparent diffusion value for both proteins lie at $\sim 0.9\ \mu\text{m}^2/\text{s}$. These mobilities are smaller than expected for proteins of their size. Prior work by Stracy *et al.* found that the mobilities of many DNA-binding proteins were independent of their molecular weight [46]. Work by El Sayyed *et al.* found this was also the case for another protein involved in transcription, NusG, which showed low free state mobility for its size [109]. This was explained through transient non-specific binding of the protein to the DNA, and could be the case for both sigma factors. Frequent and quick interactions with the DNA slow down the proteins. It is not clear whether these non-specific interactions also double up as a promoter search mechanism.

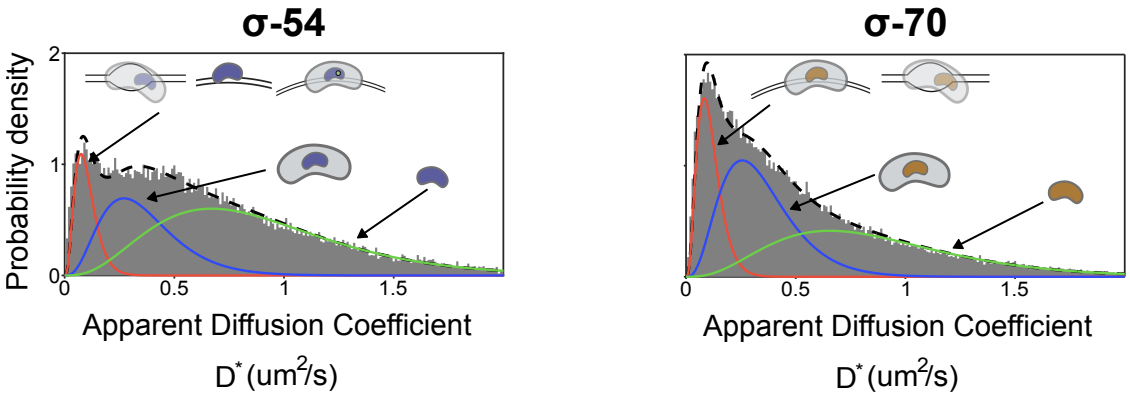


Figure 2.7: Assigning biological behaviour to states observed in the D^* diffusion distributions. σ^{54} and σ^{70} both have a free state, in which they move freely in search for a core RNAP. We also have a holoenzyme state for both sigma factors with slightly lower mobilities. The difference in σ^{54} and σ^{70} lies in their VS population. For the case of σ^{54}, this includes independent binding to the promoter site with or without the core RNAP. We expect the transcribing σ^{54}-holoenzyme population to be minimal (or none) in rich media as activator proteins are needed. Figure partially created using BioRender.

We assign these states tentatively as transcription mechanisms, particularly for σ^{70} and σ^{54}, alongside the interplay with other DNA-binding proteins is quite complex. Other machinery and interactions with the DNA could be at play.

From hereof, we adopt similar notation to El Sayyed *et al.* with the **very slow** population being denoted VS, the **slow** population being S, and the **fast** population being denoted by F. As an example, **very slow** σ^{54} is denoted by VSσ^{54} [109]. We depict all of these states and their biological relevance in Figure 2.7.

2.3.2 Heatmaps for Spatial Distribution in Rich M9

In order to probe the sub-cellular distributions of the sigma factors, we plot normalised heatmaps (see Section 2.2.5). These are shown in Figure 2.8 for diffusive bands that mirror the VS, S, and F population mobilities. These are $D^* < 0.15$ μm^2/s (VS population), 0.16 μm^2/s $< D^* < 0.61$ μm^2/s (S), and $D^* > 0.61$ μm^2/s (F). The heatmaps are filtered by cell length, such that they only include cells with length between $2-3$ μm.

For the majority of heatmaps, we observe a dumbbell-like shape, with two peaks either side of the midline. At this length, the cells are on the verge of dividing, hence they predominately contain two nucleoids.

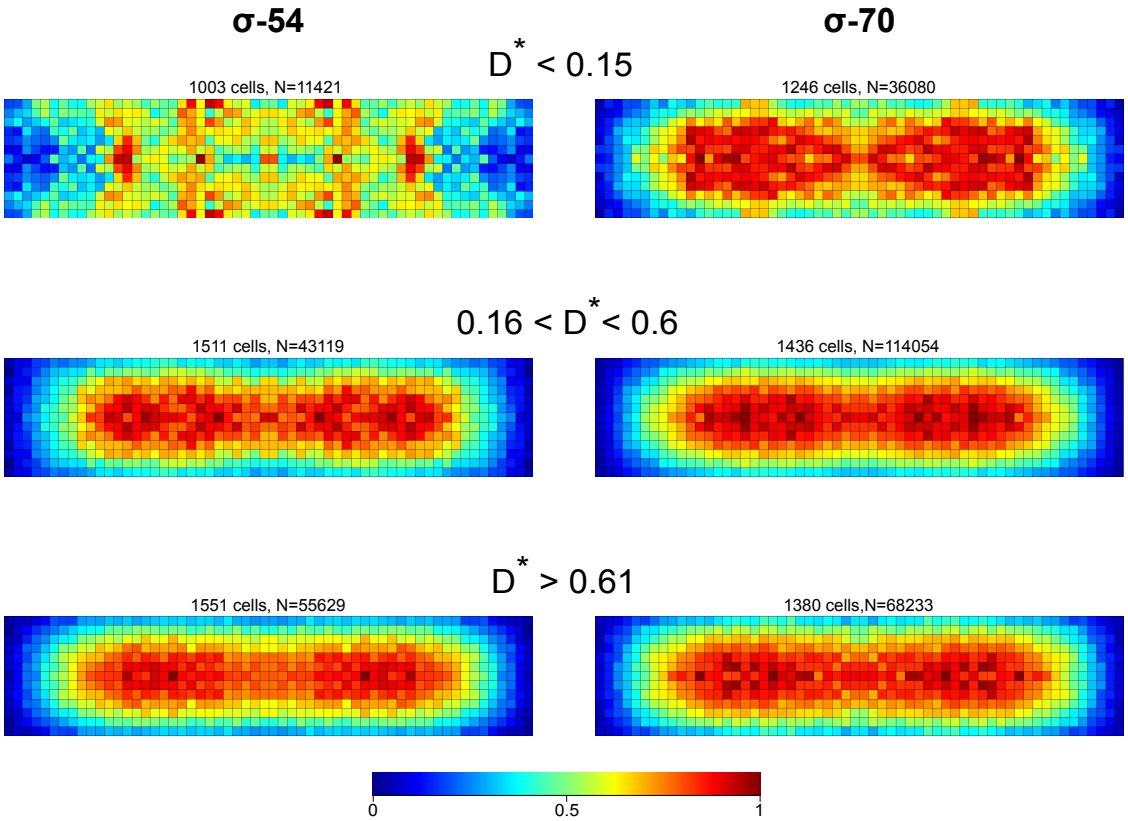


Figure 2.8: Heatmap of localisations for σ^{54} (left) and σ^{70} (right) from live *E. coli* cells. Only cells with length between $2-3$ μm were included. σ^{70} is shown to be co-localised with the nucleoid for all diffusive bands. σ^{54} seems to be spatially located outside the nucloid for low diffusion coefficients. In Rich M9, σ^{54}-dependent genes are not activated. N is the number of spatial localisations included in the entire heatmaps. All heatmaps are normalised by the number of localisations in the maximum bin. Cell numbers and number of points for each heatmap are shown above.

For σ^{54} (left panel) and $D^* < 0.15$ μm^2/s, we see a distribution that is fairly spread out and peaks along the outskirts of the nucleoids. This is consistent with the inactivity of σ^{54} in Rich M9 and may suggest stably bound σ^{54} reside on the periphery of the nucleoid, and hence its associated genes as it acts to repress these genes. For the next two diffusive bands, σ^{54} is shown to locate within the central nuceloid region. These will be the Sσ^{54} and Fσ^{54}, with the majority being Fσ^{54} (from the D^* distribution). Both populations seems to freely diffuse around the cell nucleoid.

In contrast, for σ^{70}, we see similar shapes for all heatmaps and a range of diffusive bands. For the VSσ^{70} population, we observe a slightly more defined dumbbell-like shape in Figure 2.8 (top-right). This population closely follows the spatial organisation of the nucleoid, resembling the distribution of transcribing

RNAP [19]. Thus, the VSσ^{70} population is actively involved in transcription initiation in the open/closed complex.

Again, we see much broader spatial organisation for the following diffusive bands as the sigma factor or the holoenzyme freely diffuses. For higher diffusion coefficients, we expect to see heatmap patterns that appear averaged or "washed out", as rapidly diffusing objects tend to appear centrally located within the confinements of the cell.

While heatmaps provide valuable insights into the spatial distribution patterns, it's important to note that they represent 2D projections of 3D cells. This means that proteins appearing centrally located could actually reside on cell membranes (or vice versa). The heatmap data aligns well with our interpretation from the D^* distributions shown above and distributions that follow. However, additional experimental approaches would be essential to fully confirm these spatial arrangements, as we are aware of the inherent limitation and potential artefacts that could arise from strong conclusions based on 2D projections of 3D cells.

2.3.3 Nitrogen Starvation

Next, we subject *E. coli* cells to nitrogen starvation for the σ^{54}-PAmCherry strain. This should induce more transcription of σ^{54}-dependent genes as the cell expresses proteins responsible for nitrogen assimilation [99, 112]. Cells are grown and imaged as outlined previously in Section 2.2.2. Cell length distributions are shown in Figure 2.9. The figure shows much smaller cell lengths as compared to Rich M9, since these conditions are not favourable for the cell. The peak of the distribution lies just under 2 μm as growth is stunted by a lack of nitrogen.

We again study the apparent diffusion distributions for this condition, highlighted on Figure 2.10. For comparison, we have included the D^* distributions for Rich and Poor M9. Poor M9 does not include amino acids but does not exert as much stress as nitrogen starvation. We collect 3 FOVs for Poor M9, giving a total of 673 cells and 13555 σ^{54} single molecule tracks. For nitrogen starvation, we collect 3 FOVs, for a total of 18122 tracks. The mean lifetime of the fusion is roughly ~ 9 frames for all

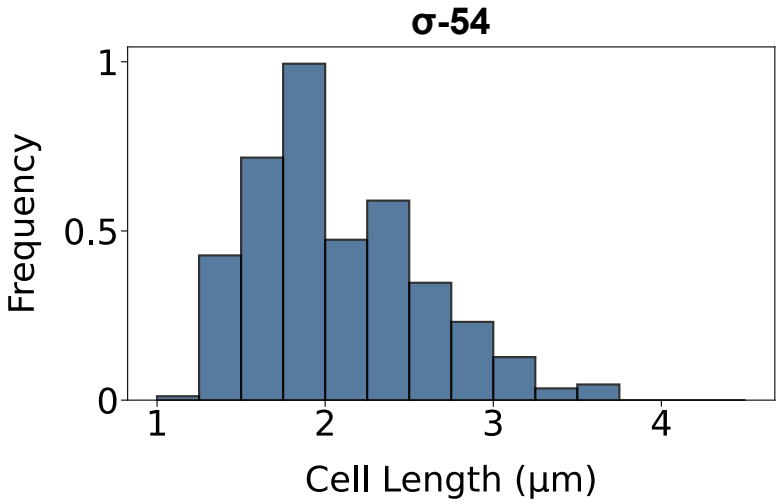


Figure 2.9: Cell length distribution for 347 cells grown during nitrogen starvation. The majority of cells lie below 2 μm as cell growth is stunted.

three conditions, since we do not expect the change of medium to impact fluorophore photophysics (as the pH of all media was experimentally ensured to be $\sim$7).

The added stress from the Poor M9 and a lack of nitrogen can be seen from the reducing number of VSσ^{54}. In favourable conditions, σ^{54} regulons are inactive. In Poor M9 and nitrogen starvation these regulons will increasingly become more active. Specifically, during nitrogen starvation, NtrC (activator for σ^{54}) is phosphorylated creating NtrC-P and this uses ATP hydrolysis to provide energy for the holoenzyme to initiate transcription of nitrogen-regulated genes (through a DNA looping mechanism) [113].

Deletion of σ^{54} has been shown to down-regulate transcription of certain genes such as *glnH*, *glnP*, and *glnQ*, as they can only be detected by this unique sigma factor [99]. This also highlights σ^{54}'s specificity and functionality for nitrogen-assimilating genes. We expect such genes to be transcribed during nitrogen starvation.

The diffusion distributions on Figure 2.10 show a clear dip for the VS population when moving from favourable to harsher conditions, in line with expected behaviour for this protein. This allows the Sσ^{54} (blue) and Fσ^{54} species (green) to dominate and actively take part in transcription processes (e.g., through promoter search and similar recycling mechanisms of σ^{54}).

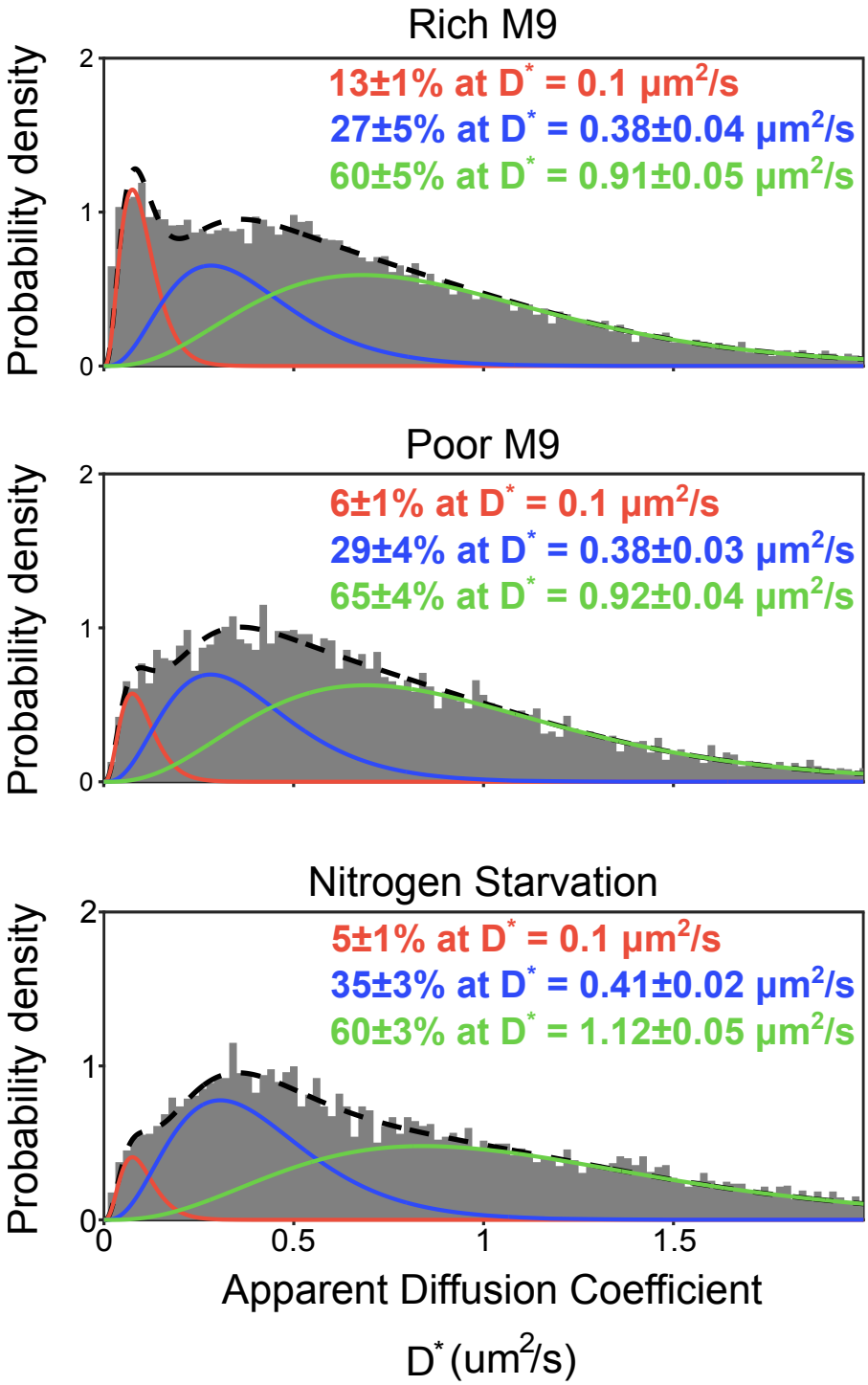


Figure 2.10: *E. coli* cells with σ^{54}−PAmCherry under increasing levels of stress. Poor M9 and nitrogen stress media do not contain amino acids. Rich M9 acts as a control. As conditions become less favourable, VSσ^{54} decreases, indicating less repression of σ^{54}−dependent genes. In rich media, σ^{54} acts as a repressor.

2.3.4 Overnight Stationary

Stationary phase occurs just before cell death, in which nutrients become severely limited. Cell division also stops or significantly slows down as the cell's stress response system is activated. To test the diffusion profiles and response of σ^{70} and σ^{54} in stationary phase, we prepare overnight stationary cultures of *E. coli* cells.

We image as outlined previously for a total of 7 FOVs for σ^{54} (1235 cells and 33530 tracks) and σ^{70} (1936 cells and 89803 tracks). Overnight stationary cultures should put the cells into "survival" mode, and we again see much smaller cell length distribution (see Figure 2.11).

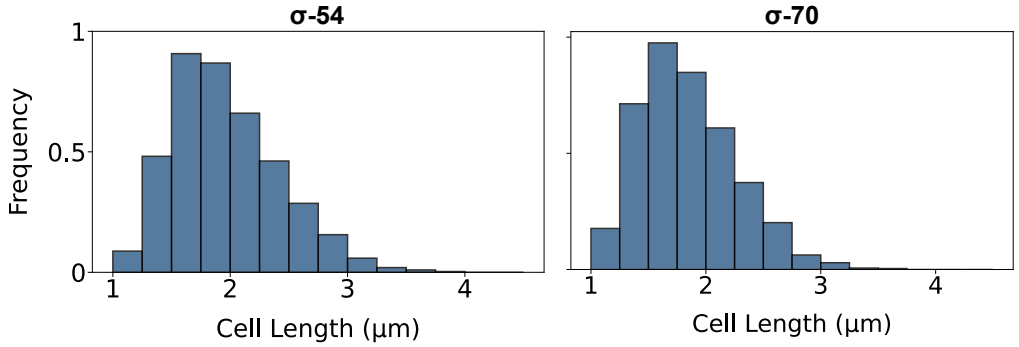


Figure 2.11: Cell length distribution for stationary phase. The distribution shifts towards lower cell lengths, below 2 μm, as cell growth is significantly slowed down.

In Figure 2.12 we present a two- and three-state fit to the D^* distributions, again highlighting lower residuals and better fit achieved using three species. Surprisingly, there are a few key points from the resulting distribution as compared to their Rich M9 counterparts.

Firstly, we see an increase in the VSσ^{54} population and a decrease in the VSσ^{70} population. This also contrasts the nitrogen starvation profile, in which VSσ^{54} decreases. The shift in the distribution can be explained through the activation of sigma-38 (σ^{38}), the major stress response sigma factor in *E. coli*. σ^{38} is active during stationary phase and helps the cell respond to multiple stress sources such as a severe lack of nutrients. σ^{54}'s role differs from σ^{38}, as it specifically handles nitrogen stress and nitrogen-dependent pathways, whereas σ^{38} handles long-term stress response. During stationary phase, the activation of σ^{38} antagonises σ^{54},

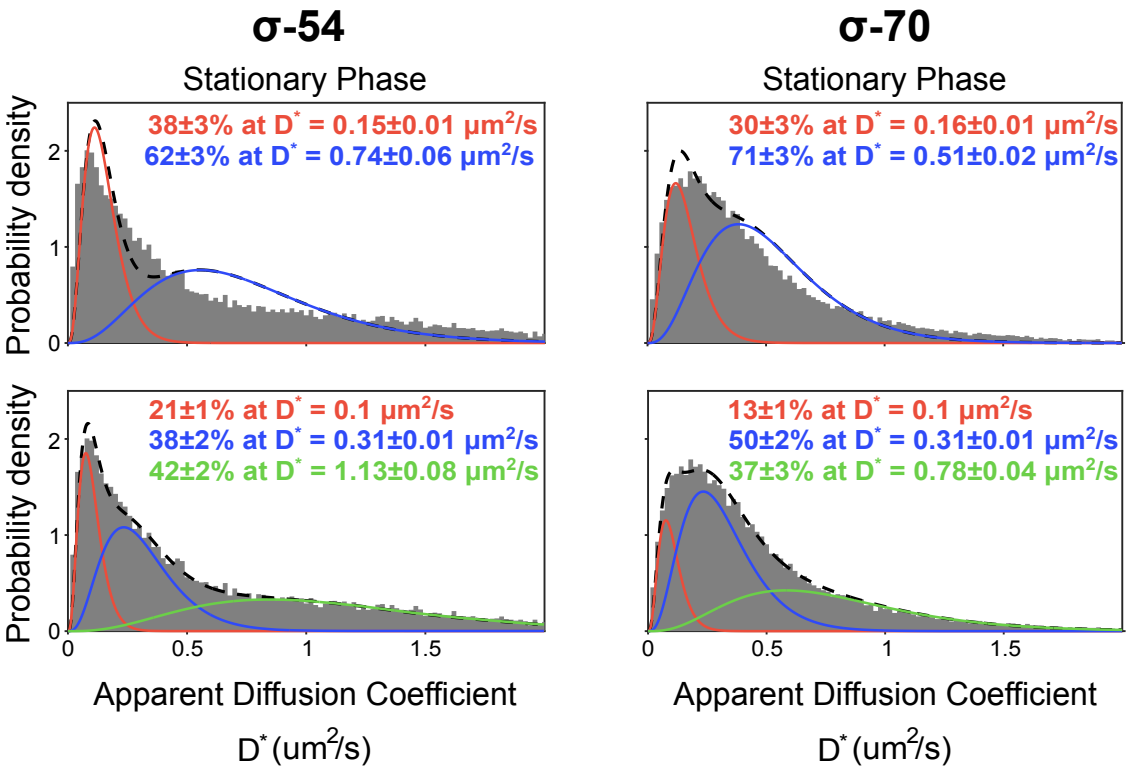


Figure 2.12: *E. coli* cells with σ^{54}−PAmCherry and σ^{70}−PAmCherry grown overnight in Rich M9. We see a decrease in the bound population of σ^{70} due to potential repression by RSD and 6S RNA. The bound population of σ^{54} increases, highlighting increased levels of repressed genes. During stationary phase, the cell enters "survival" mode. We omit a one-state fit as it is clearly not adequate.

leading to the binding of σ^{54} to the chromosome and little involvement in active transcription. Dong *et al.* showed that about 60% of genes controlled by σ^{54} are under reciprocal control by σ^{38} [114]. This binding may lead to the observed increase in VSσ^{54} and hence a decrease in Sσ^{54} and Fσ^{54}. The exact mechanism of this interplay is not well understood.

During stationary phase, *ppGpp*, an alarmone in the cell, signals stress responses across the cell by increasing the transcription of RSD (an anti-sigma factor). RSD binds specifically to σ^{70} and stops it from creating a holoenzyme (and hence downstream transcription of σ^{70}-dependent genes). Another key event in stationary phase, is the presence of 6S RNA. These are small non-coding RNA molecules (not translated into proteins), that mimic the DNA. Thus, 6S RNA essentially tricks the σ^{70} holoenzyme into binding onto it, halting σ^{70} involvement in its normal functioning, as the holoenzyme is now stuck to the RNA molecule [115].

Both RSD and 6S RNA may lead to decreased functionality of σ^{70} and remove competition for the core RNAP. This allows σ^{38} to bind to the core RNAP and leads to the observed decrease in VSσ^{70}. We also note an increase in Sσ^{70} ($\sim 10\%$) as σ^{70}-holoenzymes are now either stuck onto 6S RNA molecules or cannot bind to the holoenzyme at all due to RSD.

2.3.5 Serine Hydroxamate Treatment

Lastly, we add serine hydroxamate (SHX) to cells grown in Rich M9 before imaging. SHX is a chemical compound used to induce amino acid starvation in bacterial cells and added at least 30 minutes before imaging. SHX places the cell under stress, which is signalled to the rest of the cell by the alarmone *ppGpp*. However, unlike stationary phase, we are imaging the cell shortly after stress. We image for a total of 2 FOVs for σ^{54} (11509 tracks, 753 cells) and 3 FOVs for σ^{70} (7187 tracks, 494 cells).

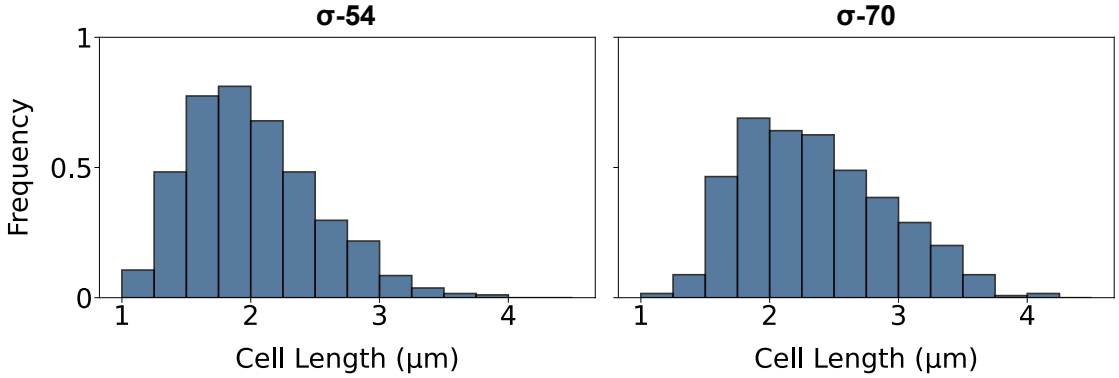


Figure 2.13: Cell length distribution with the addition of SHX to Rich M9.

After SHX treatment, we can see the cell length distribution has shifted slightly to below 2 μm although a relatively broad peak is maintained (see Figure 2.13). Cell stress under SHX treatment is not as severe as compared to stationary phase, in which we note a narrower peak.

The apparent diffusion plots for single molecule tracks of SHX treated cells are shown in Figure 2.14. SHX leads to similar diffusion profiles as compared to nitrogen starvation and Poor M9 for σ^{54}, and σ^{70}'s diffusion profile resembles stationary phase. This suggests that during amino acid starvation, σ^{54}-dependent genes are no longer

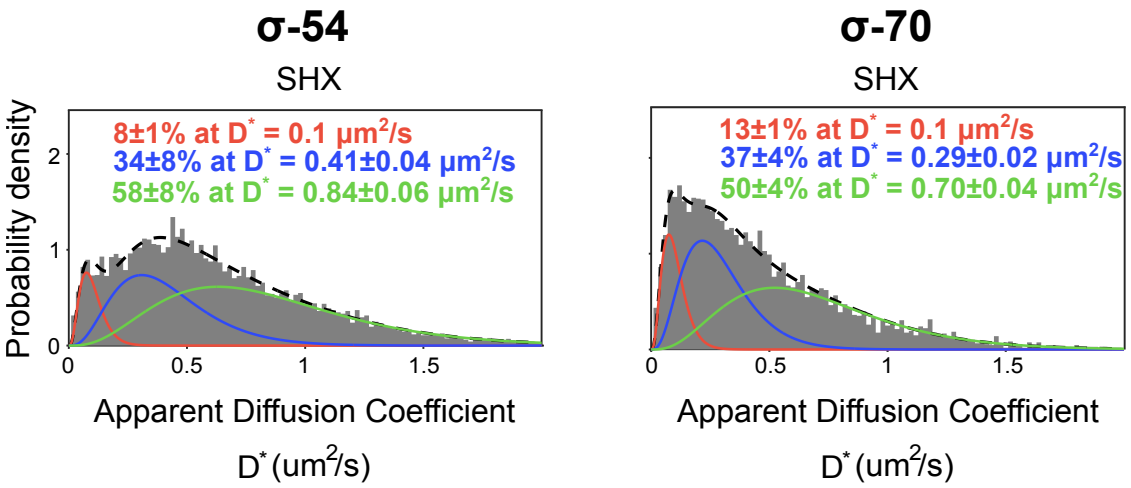


Figure 2.14: *E. coli* cells with σ^{54}-PAmCherry and σ^{70}-PAmCherry with added SHX before imaging. The addition of SHX leads to a decrease in VSσ^{54} as it no longer represses genes, similar to nitrogen starvation profiles. The σ^{70} profile resembles its stationary phase profile due to added stress and potential increased competition for core RNAP.

repressed, allowing for similar downstream biological function as in nitrogen stress. The σ^{70} profile may potentially be explained through similar mechanisms as for stationary phase, where the activation of σ^{54} hinders the functionality of σ^{70} (through repression of σ^{70}-dependent gene and increased competition for core RNAP).

We would like to note, and as evident above, there is a significant amount of interplay between mechanisms that are not fully understood. The interplay between the sigma factors and how they regulate one another is yet to be sufficiently characterised. This work aims to fundamentally characterise the diffusion distributions and biological species of σ^{54} and σ^{70}, while in the process elucidating the potential mechanisms that are at play from perturbation experiments. The interplay between various sigma factors is quite complex, as for instance, the heat shock sigma factor (σ^{32}) has been shown to be transcribed from σ^{54} promoter sites [114, 116]. For rigorous conclusions regarding the mechanisms, more experimentation and extensive mathematical modelling are necessary.

2.4 Discussion

In this chapter, we have for the first time characterised the diffusion profiles of σ^{54} and σ^{70}. They are both essential, yet functionally distinct, sigma factors in *E. coli*.

Prior to this work, little exists on both σ^{70} and σ^{54} mobilities *in vivo*.

This chapter leaves some very important questions open but also establishes a foundation for future work. We started with first successfully characterising the mobilities of the sigma factors in Rich M9. Here, we note a three species fit gave lowest residuals, signifying three main underlying biological behaviours. This is oversimplified for the case of σ^{54}, which has been shown to bind to the DNA without the need for a holoenzyme. Nevertheless, we can assign biological activity with very slow, slow, and fast moving species, as we established in the text. Both sigma factors in the holoenzyme move with the mobility of the core RNAP, which aligns with the much larger size of the core RNAP compared to the sigma factor. However, the diffusion coefficients for proteins in the presence of the chromosome is much smaller than expected. We believe this is the case for both the freely diffusing sigma factor (F population) and the holoenzyme (S population), which are slowed down through transient non-specific interactions with the chromosome. These transient interactions may serve as a promoter search mechanism. In the case of σ^{54}, this search mechanism would also allow its binding to the chromosome without the holoenzyme, leading to the repression of nitrogen assimilating genes. The spatial distribution of slow moving σ^{54}, from its associated heatmaps, suggests these genes may be on the periphery of the nuceloid.

Increasing the level of cellular nitrogen stress has shown to lead to less very slow moving σ^{54} (VS population). This is most likely due to unbinding of the sigma factor from its genes, allowing for their expression. Revisiting our earlier schematic of σ^{54}, these results suggest the removal of σ^{54} from repressed genes. This can occur if the holoenzyme rests on the promoter and is activated, or if the core RNAP can locate the sigma factor (e.g., if bacterial clusters form). Once the core RNAP enters elongation, we expect the free sigma factor to dissociate as normal, ready to bring another core RNAP to the promoter site. This makes sigma factors not only essential in promoter search, but specific to their given role. The release of bound σ^{54} during nitrogen stress leads to more of its holoenzyme and free population, as observed. Previous work has shown deletion of σ^{54}, leads

to the up-regulation of σ^{70}-dependent genes, suggesting that σ^{54} represses other genes, not specifically only nitrogen assimilating ones. However, these genes were found to be functionally unrelated [99].

The stationary phase profiles also unravelled some biological mechanisms involving σ^{38}, the general stress response factor in *E. coli*. This led to more σ^{54} repression and potential complex mechanisms inhibiting the functionality of σ^{70} (which may involve RSD and 6S RNA) [115]. Preliminary results for SHX treatment also show observed diffusion profiles resembling nitrogen starvation for σ^{54} and σ^{70} as expected, and potential repression of σ^{70} genes by σ^{54} (or increased core RNAP competition). This does bring into question exact mechanisms in which the cell responds to stress, both short-term and long-term, and their pathways. During overnight stress we see differing responses as to stress placed shortly before imaging. The interplay between these three sigma factors, and also other sigma factors, remains an open question. We hypothesise duration of stress will impact levels of sigma factors in the cell, but so will the type of stress. The bioavailability of each sigma factor will also play a role in this, possibly making it a competition and numbers game between sigma factors for limited core RNAP binding sites (alongside mechanisms affecting their binding to the core RNAP). We summarise some of the main behaviours in a schematic on Figure 2.15.

We do not expect the fusion protein (σ^{54} or σ^{70} with a PAmCherry tag) to significantly affect the mobility of the sigma factors. Even if the fusion were to cause subtle changes in absolute diffusion values, the main conclusions of this work remain unaffected, as our analysis focused on relative shifts in mobility distributions rather than exact diffusion coefficients. For the very slow population, which represents DNA-bound proteins, the size of the fusion is unlikely to play a significant role, as these proteins are essentially immobile. In the slow population, the sigma factors are part of the RNAP holoenzyme, which has a mass of $\sim 500\ kDa$, much larger than the $\sim 30\ kDa$ PAmCherry tag. We therefore expect the tag to have negligible effect on mobility in these states. The mobility for both sigma factors in the holoenzyme also aligns well with previous observations of free RNAP mobility by Stracy *et al.* [19].

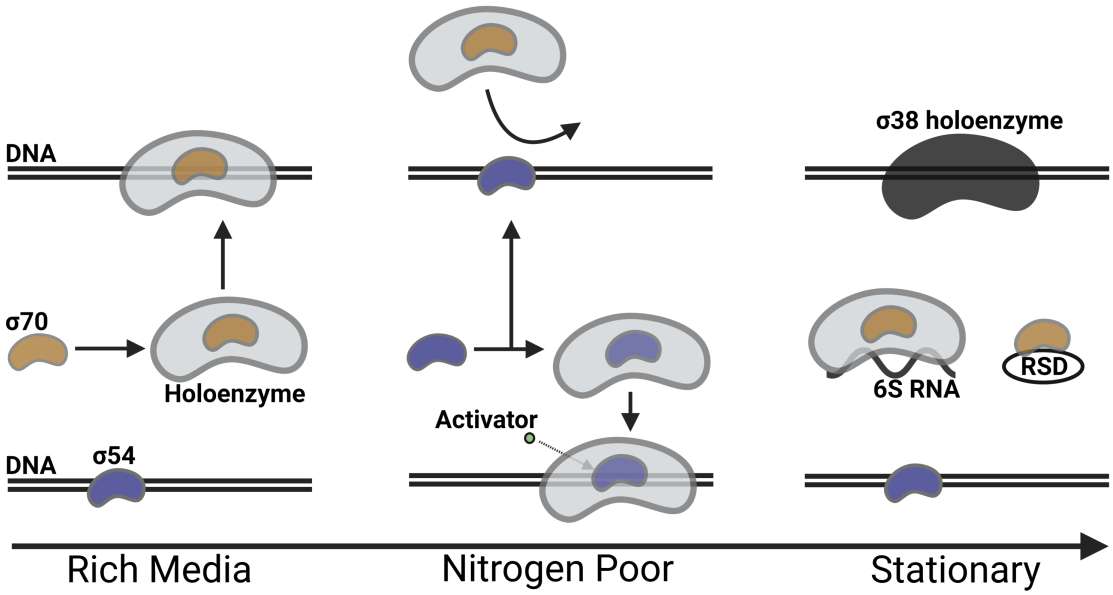


Figure 2.15: Potential mechanisms affecting sigma factor behaviours. In rich media, σ^{54} is bound to the DNA (with or without RNAP), repressing promoter sites. σ^{70} is able to undergo its normal transcription cycle. During nitrogen starvation, σ^{54} no longer represses its promoter sites and the presence of its activator protein allows for it to transcribe nitrogen assimilating genes. σ^{70} holoenzymes may be blocked by σ^{54}, while there is also increased competition for the core RNAP. In severe conditions, the presence of 6S RNA and RSD may prevent normal σ^{70} functioning [115], and σ^{54} stays bound to the DNA. Here, σ^{38} holoenzymes transcribe genes for cell survival. Figure created using BioRender.

This leaves the fast population, in which mobility changes can be more significant due to the relative sizes of the sigma factor and the PAmCherry tag. Prior work by Stracy *et al.* demonstrated that intracellular mobility of DNA-binding proteins is governed primarily by their interactions with DNA rather than their molecular weight or the presence of a fusion tag [46]. This included the HU protein, which is smaller in molecular weight than the sigma factors, yet exhibited similar mobility behaviour consistent with DNA interactions. Thus, we believe DNA interactions mainly drive the fast populations mobility. Stracy *et al.* also compared wild-type proteins to mutants lacking DNA-binding domains, showing that DNA binding interactions, and not the tag, determined *in vivo* behaviour [46]. Although sigma factors were not directly assessed in that study, care was taken in our experimental design to avoid placing the PAmCherry tag near known RNAP- or DNA-binding domains. A flexible linker was also incorporated to minimise steric hindrance. Importantly, σ^{70} is an essential protein and any disruption to its binding activity

caused by the fusion leads to cell death. The fact that we have performed successful experiments and characterisation provides additional support that the fusion is functional and its binding behaviour is not affected by the tag. Nevertheless, we acknowledge that protein binding is complex, and this may be a potential limitation of our work. While we do not expect these factors to alter our key findings, further experimental validation can be pursued to fully rule out any fusion-induced effects.

Future work may also focus on categorising mobilities for these sigma factors under broader cellular stresses, potentially also combination of stresses. This may shed light on pathways the cell uses to handle these stresses and the role of respective sigma factors. σ^{54} diffusion profiles seem to be highly specific to nitrogen-dependent stress, in line with its functionality.

We have established a foundation for future tracking PALM work for σ^{54} and σ^{70} in *E. coli*. This has also shed light but also raised further questions regarding their exact nature. Prior to this work, the σ^{54}-PAmCherry and σ^{70}-PAmCherry fusions had not been developed. We thank Hafez El Sayyed for developing these strains.

Alongside advancements in biological methods, this chapter required various computational methods. These included Gaussian PSF fitting, tracking of protein movement and subsequent calculation of their diffusive coefficients and residuals from curve fitting functions. These are common approaches used in the field of single molecule tracking. Of note, from the "ensemble" apparent diffusion distributions, are the overlapping populations. It is difficult to identify the state of a single track without potential misclassification. In the upcoming chapters, we focus on advancing computational methods for single molecule tracking, addressing issues in diffusion calculation, and state identification for single tracks.

3

Capturing Protein Trajectories at Long Exposures

Contents

3.1 Introduction 56
3.1.1 Brief Overview 56
3.1.2 Protein Binding Times 56
3.2 Methods 59
3.2.1 Convolutional Neural Networks 59
3.2.2 Data Generation 60
3.2.3 Model Architecture 62
3.2.4 Training 64
3.2.5 Inference 66
3.2.6 Bound Time Calculations 68
3.2.7 Cell Preparation 71
3.2.8 Analysis of Classifier Output 72
3.3 Results 72
3.3.1 Simulations 72
3.3.2 Explaining Model Classification 75
3.3.3 Application to *in vivo* PALM 78
3.3.4 Breaking Point 85
3.3.5 Image Quality 87
3.4 Discussion 88

3.1 Introduction

3.1.1 Brief Overview

In the previous chapter, we characterised diffusion profiles for DNA-binding proteins involved in transcription. Protein-DNA binding times are a fundamental part of these characteristics, governing the progression rate of the transcription machinery. As shown in the previous chapter, this is attributed to the very slow moving population, representing immobile proteins. This chapter presents a comprehensive review of established methods for determining protein binding times, followed by the introduction of a machine learning method designed to address current limitations in binding time calculations.

3.1.2 Protein Binding Times

Protein-DNA interactions are essential for cellular function and adaptation to environmental changes. The ability of certain proteins to interact with DNA plays a pivotal role in processes such as gene regulation, transcription, DNA replication, and repair. These DNA-binding proteins explore the cellular environment through diffusion, frequently binding transiently to the DNA and exhibit long-lived binding to specific target promoter sites on the DNA [12, 15, 18–20, 46, 109, 117–119]. This has enabled the direct observation of the different diffusive states of DNA-binding proteins. However, typical PALM experiments, as highlighted in the previous chapter, use short exposure times ($\sim 10 - 15\ ms$), leading to truncated single molecule tracks due to fluorophore photobleaching [48, 120]. Short tracks make changes in diffusive state from mobile (Slow/Fast) to bound (Very Slow) (or vice versa), which typically occur on the order of seconds, unobservable [15]. To resolve this, the imaging time of a single protein track is extended using a variety of approaches, such as stroboscopy or longer exposure times paired with lower laser intensities [121]. At long exposure times ($>\sim 100\ ms$), motion blur leads to diffusing molecules appearing as broad dynamic clouds, whereas bound molecules appear as narrow spots. Stringent criteria for signal intensity and spot width then eliminate

signals from diffusing molecules, allowing for the determination of DNA binding times. Non-specific transient interactions with the DNA are not noticeable and washed out in the diffusing state, leaving the bound states representing long-lived protein-DNA interactions [15, 118, 121]. In the previous chapter, an exposure time of 15.11 ms for an average of 9 frames led to roughly 140 ms of protein observation time. At longer exposures times (e.g., 500 ms) and for the same number of frames (with decreased laser intensity), this can be stretched to a few seconds. The increased observation period allows for determining binding times for certain proteins.

The temporal binding dynamics of proteins provide unique insights into cellular mechanisms by detailing timescales of processes and identifying rate-limiting steps under different physiological conditions [15, 48, 117, 118]. Previous work by Uphoff *et al.* determined the binding time of DNA Polymerase I using PALM, elucidating the kinetics and mechanism of DNA damage repair [15]. A similar approach was employed by Stracy *et al.* to study DNA gyrase [118], both in living *E. coli* cells.

The analysis of bound times has been primarily based on tuning signal and width thresholds [15]. As a result of these hard thresholds, false positives are introduced due to molecules moving out of focus, intensity fluctuations, and truncated tracks. Subsequently, Kapadia *et al.* introduced an alternative approach *Bound2Learn*, based on random forests, for classifying tracks while minimising false positives. This used features from protein trajectories: mean speed, maximum speed, minimum speed, median speed, and maximum quality [48]. *Bound2Learn* allows for lower intensity thresholds to be set during localisation, thus picking up more signals. Additional filtering steps, using Gaussian Mixture Models, are implemented to further remove false positives.

As pointed out by Kapadia *et al.* sensitivity to image quality remains an issue in *Bound2Learn*, with the additional challenge of labour intensive manual curation of training data and tricky parameter tuning, introducing the need for expertise in random forests. Also, this method does not adapt well for tracks undergoing changes in diffusive state.

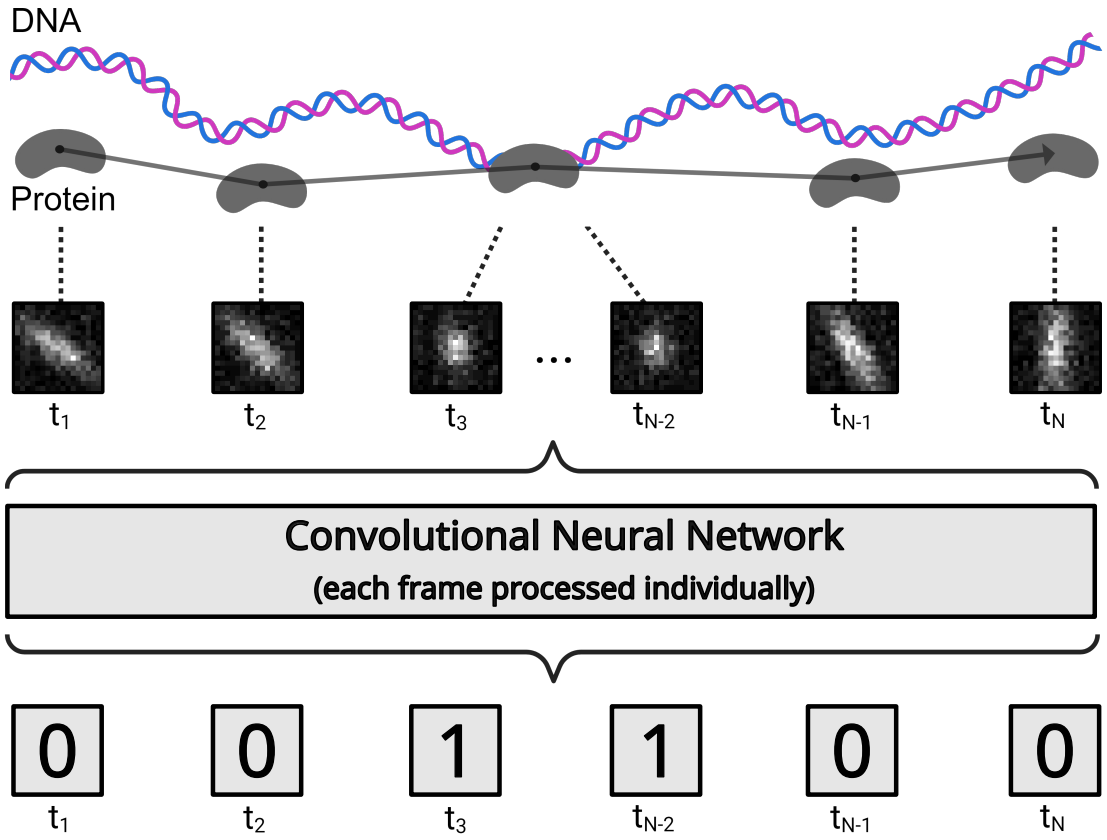


Figure 3.1: A single molecule diffusing and binding to the DNA for a set duration (e.g., DNA Pol I). Image data is cropped to focus on the intensity profile of a single molecule. This is then classified by the CNN for binary classification per-frame. The resulting binary array can be used for downstream bound time analysis. The period that the protein is bound to the DNA is $N - 4$. Figure partially created using BioRender.

As highlighted above, traditionally the estimation of binding times has relied on the mean characteristics of molecular tracks (e.g., mean speed, mean spot width). However, in the context of extended exposure periods, the discernible visual differences between bound and diffusing proteins become increasingly pronounced. In this context, we introduce an approach based on machine learning, facilitating comprehensive information extraction from protein tracks. Specifically, we employ a convolutional neural network to classify every frame of a protein track, allowing for transitions in state to be detected. Augmentations added to the training data ensure the model's resilience to common experimental imperfections, encompassing defocusing and elevated signal-to-noise ratios. After initial testing on simulated data, we apply explainable AI to validate the model's ability to learn correct

distinguishing features. Lastly, the model is used to determine protein-DNA binding times of DNA Polymerase I (DNA Pol I), showing agreement with Uphoff *et al.*, and subsequently RNAP, σ^{70}, and σ^{54} in living *E. coli* cells using PALM. Figure 3.1 depicts a schematic showing the overall structure of the model. Diffusing and bound proteins show differing image intensity profiles caused by their motion. The convolutional neural network processes a cropped image leading to a binary classification and subsequent bound time analysis.

3.2 Methods

3.2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a class of neural networks primarily for any data that has a grid-like topology [28, 122]. This can be 1D time series, or as encountered often, multi-channel image data. As the name suggests, CNNs perform a convolution operation given an input image. A spatially smaller kernel matrix slides over the dimensions of the input image to produce what is referred to as a feature map. The convolution operation is given by [28]:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.1}$$

where K is the kernel matrix, I is the input image and S is the resulting feature map [28].

Usually, multiple kernels are used for every convolutional layer of a CNN. The output of a single convolutional layer would then be multiple feature maps. Every kernel has learnable weights and may detect various properties and features in the image. After the convolution, pooling layers are usually applied. These reduce the dimension of the image by performing some operation on a local region of the image. Examples include max-pooling, which replaces nearby values with their maximum in the local neighbourhood. Pooling allows the CNN to reduce the dimensionality of the image, but also make the CNN invariant to small translations in the input, as the pooled output remains the same. Most CNN architectures

repeat the convolution and pooling operations multiple times and may include connections between layers. Eventually, the final vector representation from the last layer may be flattened and passed into a feed-forward neural network, mapping the vector to a given target shape (e.g., number of target classes).

CNNs have three main advantages over traditional neural networks. Firstly, they share learnable parameters - each kernel reuses the same set of weights as it slides across the image. For example, a 3×3 kernel needs only 9 parameters to process an entire image, while a standard neural network would require a separate weight for every pixel position. Second, CNNs are translation invariant - they can detect features regardless of where they appear in the image. The same kernel can detect the object regardless of its position in the image. Lastly, CNNs use sparse interactions - each point in a feature map only depends on a small region of the input (the receptive field), not the entire image. As the network gets deeper, these receptive fields grow larger, allowing the network to detect increasingly complex patterns. This contrasts with standard neural networks, where every output neuron connects to every input pixel, requiring many more parameters and losing the benefits of spatial relationships in the input image [28, 122].

3.2.2 Data Generation

Before application to real data, we train a model on a simulated dataset as this is readily available. This preliminary step also serves as a proof-of-concept, laying the groundwork for subsequent applications to *in vivo* PALM images. The simulations employed were sourced from the work by Kapadia *et al.* providing TIFF videos depicting simulated protein trajectories [48]. This approach allowed us to also manipulate the following parameters: number of frames, resolution, exposure time, fluorophore intensity, tracks per cell, cell autofluorescence, camera noise, PSF width, binding/unbinding rates and diffusion coefficients of the slow and fast-moving populations. We stick to a two-state system as this is common for long exposure times. Any non-immobile states average out over the exposure time and appear as one state, with narrow bound spots only resulting from bound/immobile states.

Table 3.1: Parameters used for simulating protein trajectories in PALM videos. Any parameters not included are set to their default values [48]. Note we purposefully set the search time of the protein to be very large and the initial bound population to 50%. This is not crucial and can be changed but leads to a roughly equal number of diffusing/bound images for downstream training.

Parameter	**Range**
Exposure time	50-1000 ms
Fluorophore Total Intensity	1000-3000 photons
Camera noise	100-300 photons/pixel
PSF Std. Deviation	8-18 pixels
Image Size	500 × 500 pixels
Bound Population	50%
Search / Binding / Bleach Time	10^7 s / 8 s / 20 s
Diffusion coefficient (slow)	0.01-0.1 $\mu m^2/s$
Diffusion coefficient (fast)	0.2-1 $\mu m^2/s$

Additionally, we are solely interested in the bound state, with potentially other states (including any noise before or after the start of the track) being grouped as one.

We simulate a total of 50 videos, with each video being 50 frames and illustrating the movement of 360 simulated proteins. For every simulation, we uniformly pick from a randomly generated set of parameters. Particularly, the fast population's diffusion coefficient spanned from 0.2 to 1 $\mu m^2/s$, while exposure times ranged from 50 to 1000 ms. The range of values are shown in Table 3.1.

Subsequently, each localization for every protein track was cropped using a window size of 8 from the central spot peak, resulting in the creation of 17 × 17 pixel images. Importantly, these images encode information pertaining to the state of the protein, distinguishing between diffusing and bound states. Every image has an associated ground truth state, either being Bound (1) or Diffuse (0). This process was repeated for all protein tracks, generating 50,000 17 × 17 pixel images for the bound population and an equivalent number for the diffusing population. The dataset was partitioned into training, testing, and validation sets in a ratio of 50:30:20 respectively. The training set is used to train the model and find optimal values for model parameters. The validation set is used to tune training hyperparameters and prevent under- or over-fitting. Lastly, the test set evaluates

the model's final performance on unseen data.

In preparation for real experimental scenarios, *in vivo* PALM images were manually classified on a per-frame basis by inspecting spots from experimental data. We collect data by running experiments using DNA Pol I tagged with the fluorophore PAmCherry (see Section 3.2.7). After data collection, we apply localisation and tracking algorithms, as in Chapter 2. A slightly lower threshold for signal thresholding can be used, ensuring trajectories are not fragmented. Similarly to simulated data, the localisation is cropped around the peak producing 17×17 images. We then classify each image as being 0 or 1. For real PALM data, this process can be tedious and subjective, as we do not know the ground truth. This makes the process of creating training data from real PALM experiments somewhat error prone. Bound proteins and diffuse proteins were manually curated from single spot data that we had high confidence in representing the state of interest. This resulted in 497 images for both bound and diffusing states each.

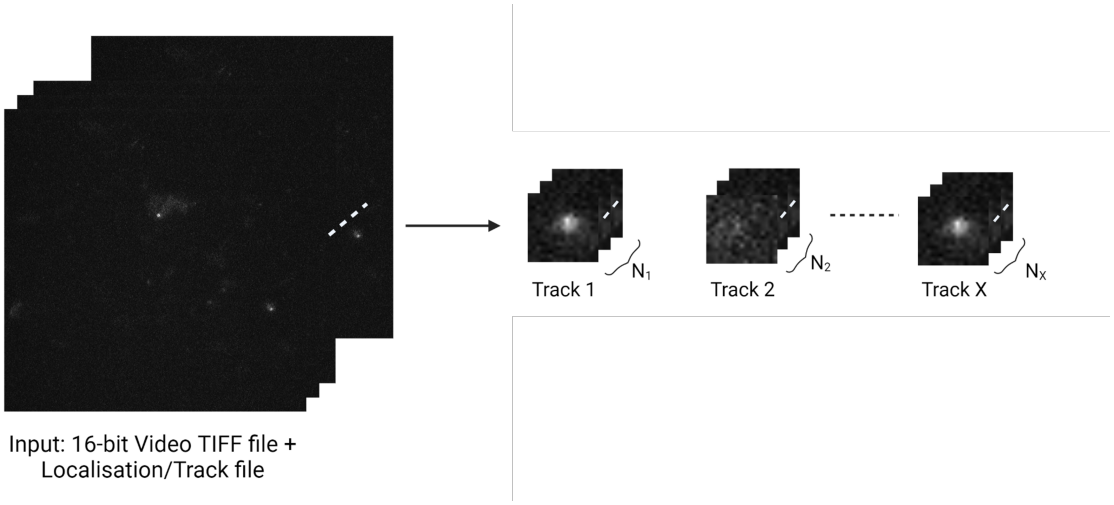
A single protein trajectory then consists of multiple 17×17 images, giving a shape of $17 \times 17 \times T$ where T is the length of the protein trajectory. This process is shown in Figure 3.2a. Dealing with the protein images on a per-frame basis allows us to handle variable length trajectories. An example protein track from a real PALM experiment is also shown in Figure 3.2b.

Our CNN model underwent a two-phase training process, with an initial phase focusing on simulated data exclusively, followed by a second phase featuring additional fine-tuning on experimental PALM data. This latter phase was tailored for the unique nuances of the specific microscope (described in Chapter 2).

3.2.3 Model Architecture

The aim of our model is to distinguish between two classes of input images: Bound (1) and Diffuse (0). Once cropped, images are min-max scaled to produce a consistent input range for the model. This also makes the model more robust to changes in brightness across different experiments or molecules. The resulting normalised

(a)



(b)

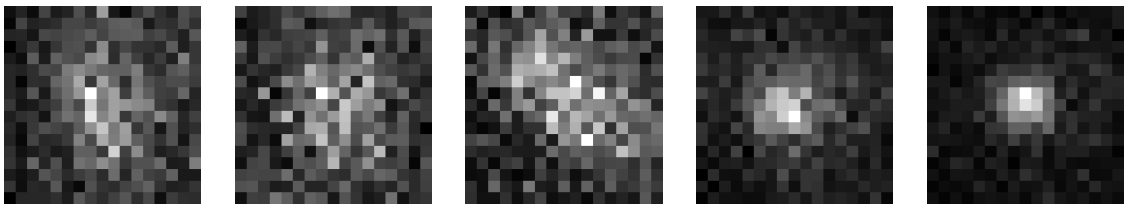


Figure 3.2: (a) Process of producing tracks for classification from video data, either simulated or real experimental data. Figure created using BioRender. (b) A single DNA Pol I molecule undergoing a transition from an initial fast diffusive state to a bound state obtained from real PALM data. Exposure time is 500 ms, so this trajectory lasts 2.5 s. The bound state is truncated by fluorophore bleaching lasting only 1 s. This will be corrected for in binding time calculations. Each image is of size 17×17 pixels ($1\ px = 100\ nm$).

images of a given protein trajectory are passed individually through the model for classification, with the target class either being Bound (1) or Diffuse (0).

The CNN architecture employed in this chapter comprises of three distinct convolutional layers, each equipped with kernels (or filters) of sizes 8, 16, and 32 respectively. Each convolutional layer incorporates zero padding, and is followed by a Rectified Linear (ReLU) activation function and a Max Pooling layer with a stride of 1. The output of the last convolutional layer is flattened and channeled into two fully connected dense layers, consisting of 64 and 16 neurons, respectively. Ultimately, a sigmoid activation function is applied to yield an output within the range of 0 to 1, representing a measure of confidence of the given image being Bound.

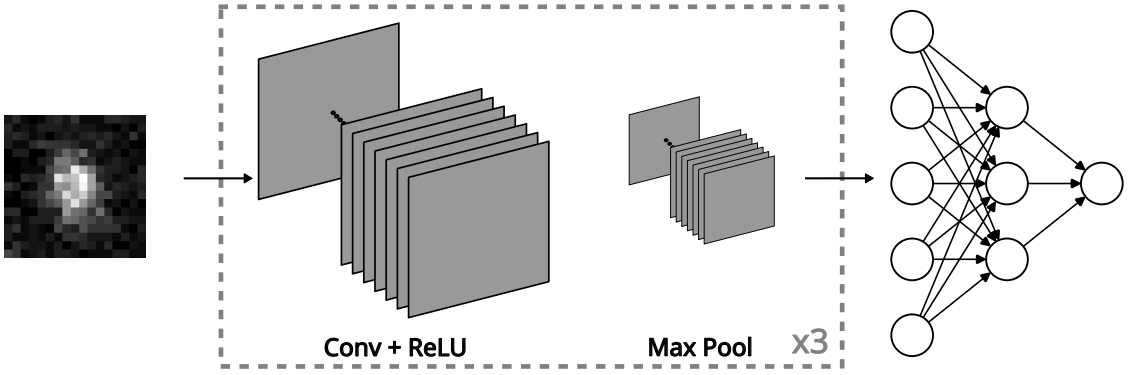


Figure 3.3: CNN architecture. The model consists of three convolutional layers with ReLU activations and Max Pooling Layers in between. Finally the output is flattened and mapped to a binary class. Every frame of a protein trajectory is passed into the model for classification. Dropout is applied after flattening and both dense layer.

A simple threshold of 0.5 then distinguishes between the two classes. During training, dropout is applied with 50% after flattening and both dense layers. The model architecture is depicted in Figure 3.3 (with a total of 408417 trainable parameters).

The model design process started with a single convolutional layer, followed by progressive addition of layers until no substantial improvement was observed. We also performed a grid search (with some trial-and-error) for the optimal kernel size, pooling stride and number of neurons to obtain the optimal architecture. The parameters returning the lowest validation loss were chosen.

3.2.4 Training

Training hyperparameters were chosen via grid search and some trial-and-error. The model is not large making this feasible. In Chapter 4, we use a Bayesian search instead. The lowest validation loss was achieved for a model with a batch size of 32, trained for a total of 90 epochs. The model is trained using the Adam optimiser, with an initial learning rate of 1×10^{-3} [30]. A learning rate scheduler was implemented to ensure training convergence, with the learning rate decreasing by a factor of 10 if no improvement was seen after 10 epochs.

Data augmentation was executed using the Albumentations package [123]. The augmentation process increases the variance of the training dataset, improving the resilience of the model to variations in experimental parameters, such as laser

intensity (affecting brightness), camera noise, and intensity signal quality (e.g., slight blurring/out-of-focus). All augmentation methods added during training are shown in Table 3.2. Due to an oversight, images were augmented with flips at 100% probability, unintentionally excluding the original (non-flipped) images from the training set and reducing dataset diversity. However, this did not negatively impact model training or performance and is not as much of an issue for continuous augmentations (brightness and contrast). Augmentations are only applied to the training dataset.

Training with the above configuration was first performed on the simulated dataset. Then, we repeated the same procedure for fine-tuning on real PALM data, with the exception of limiting the number of epochs to 30.

Table 3.2: Data augmentation techniques applied to the training dataset using the Albumentations package [123]. All augmentations are applied with 50% probability unless otherwise specified.

Augmentation	**Description**	**Amount Applied**
Flips	Mirrors the image horizontally or vertically	100% probability
Zoom	Scales the image	0.5 to 1.5×
Translations	Shifts the image in x and y directions	0 to 5 pixels
Rotations	Rotates the image by an angle	-360 ° to +360 °
Shear	Slants image along an axis	-45 ° to +45 °
Pixel Dropout	Randomly sets pixels to 0	1% probability for every pixel
Gaussian Noise	Adds Gaussian noise to every pixel	$\sigma^2 = 5 \times 10^{-4}$, $\mu = 0$
Brightness Adjustments	Adjusts pixel intensities by sampling between (-0.5, 0.5) where 1 is maximum brightness, 0 is no change, and -1 is minimum brightness	100% probability
Contrast	Adjusts contrast by sampling between (-0.5, 0.5) where 1 is maximum contrast, 0 is no change, and -1 is minimum contrast	100% probability
Multiplicative Noise	Scales pixel values by random factors	0.8-1.2× per pixel
Transpose	Switches x and y coordinates	50% probability
Defocusing	Applies Gaussian blur to simulate out-of-focus	radius: 1-3 pixels blur: 0.1-0.5 pixels

After training, we use the model weights corresponding to the epoch with

the smallest validation error. The loss used for training is the binary cross-entropy given by:

$$\mathcal{L} = -\frac{1}{B}\sum_{i=1}^{B}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.2}$$

where B is the batch size, y_i the true label, and $\hat{y}_i$ the predicted label.

3.2.5 Inference

After successful training, we run inference on protein trajectories by following the same pre-processing steps (localisation and tracking as in Chapter 2, cropping and min-max normalisation). The collective state of the protein trajectory (after model prediction) is then reconstructed by concatenating individual state labels, creating sequences such as 110001110001. These binary arrays encapsulate the protein state but also transitions between bound and diffusing states, on a per-frame basis.

At this point, we implement a custom step-based sliding window heuristic to refine the CNN's classification output and eliminate physically unlikely state transitions. This refinement method uses protein step sizes between frames and examines the step size into and out of a given state. The window considers three states at a time and flips the middle state if the step size into/out of the state is greater or less than the typical physical step size for that state, as depicted in Figure 3.4. As an example, if a diffuse protein (0) is sandwiched between two bound states (as in the figure), but there has been little movement between frames, the state is flipped to 1. The refinement improves the accuracy of the model, as shown later in the Results, and is a simple and flexible way to integrate spatial coordinates into the model.

We exclude sequences of three consecutive identical states (000 or 111) from this correction process since the CNN classifier demonstrated high accuracy in identifying sustained behavioural states. Instead, this refinement focuses on eliminating isolated misclassification, such 010, where a single frame's classification differs from its neighbouring states. Such isolated state changes can represent classification errors rather than genuine behavioural transitions and can artificially fragment bound

times - where a single misclassified frame (0) within a series of bound states (1) would incorrectly split one long binding event into two shorter ones. There are six possible scenarios: 010, 110, 011, 101, 001, 100.

To calculate the typical step sizes in Figure 3.4, we consider the distance moved in a given exposure time (τ_{exp}). The apparent diffusion coefficient (D^*) for a protein moving in two dimensions is given by [46] (see also Equation 2.4 and 2.5):

$$D^* = \frac{MSD}{4\tau_{\text{exp}}} \tag{3.3}$$

In the case of two frames, the MSD is simply the distance between consecutive localisations (d). Plugging in $MSD = d^2$ and rearranging gives us:

$$d = \sqrt{4D^*\tau_{\text{exp}}} \tag{3.4}$$

For a given protein, we are usually aware of its apparent diffusion coefficient for both the mobile (D_d^*) and bound (D_b^*) states from short exposure PALM experiments or literature. If not, we can always measure these from experiment as done in Chapter 2. This allows us to obtain the typical step a protein can take in either state between frames as:

$$\text{Bound Step } (S_B) = \sqrt{4D_b^*\tau_{\text{exp}}} \tag{3.5}$$

$$\text{Diffuse Step } (S_D) = \sqrt{4D_d^*\tau_{\text{exp}}} \tag{3.6}$$



Figure 3.4: Visualisation of steps between states. At any given window, we consider the middle state (red), and check if Step 1 (S_1) and Step 2 (S_2) violate the typical step size (defined by their respective states). If so, the middle state is flipped.

When determining conditions to flip a given state, we take a generous and naive approach. For an example state pattern such as 101, we do not know at which point exactly in time the state switched from $1 \rightarrow 0$ or $0 \rightarrow 1$, since we sample in intervals of τ_{exp}. Let's assume in between the state changes, the protein was moving with the diffusive properties of the middle state. If the protein has been moving with high mobility we expect the displacement to be large. However, if both of these displacements are small ($< S_B$), the middle state is flipped to 1. We apply the same logic to other state patterns. In the case of bound middle states, such as $*1*$, we take the protein to be bound for both state transitions, similar to before. If the displacements are greater than S_B, we flip the state. Additionally, as a second constraint, if either displacement is very large ($> S_D$), we flip the state too (from Bound (1) to Diffuse (0)). These heuristics are summarised in Table 3.3.

Table 3.3: State transition conditions based on jump sizes

State Pattern	Conditions for Middle State Flip
011, 110, 010	($S_1 > S_D$ **or** $S_2 > S_D$) ($S_1 > S_B$ **and** $S_2 > S_B$)
101, 001, 100	($S_1 < S_B$ **and** $S_2 < S_B$)

3.2.6 Bound Time Calculations

Once we obtain a predicted sequence of states (e.g 11100010), we can calculate binding times of the protein to the DNA. Protein binding times are affected by fluorophore bleaching. A given fluorophore can enter a non-emissive/OFF state due to bleaching, truncating the observation of what was a bound protein signal. The observed on-time (number of 1's) in a single molecule experiment is the convolution of the true underlying bound time distribution with the fluorophore bleaching distribution. Correcting for photobleaching is important for determining accurate binding times [120].

The bleaching and on-time distributions can be obtained from experiment or simulations. The bleaching profile is given by the lifetime of the fluorophore, which

is the total track length. The on-time distribution is obtained from the durations of consecutive 1's in a protein track. Using this data, we can generate survival functions by applying a Kaplan-Meier estimator [124]. The survival function returns the likelihood of surviving beyond time t.

Fitting single exponential of the form e^{-kt} to either the on-time or bleaching survival distributions gives the characteristic time for the process as $\tau = 1/k$ [15]. Following the derivation by Yang *et al.* we obtain [120]:

$$\frac{1}{\tau_{\text{on}}} = \frac{1}{\tau_{\text{bound}}} + \frac{1}{\tau_{\text{bleach}}} \tag{3.7}$$

where τ_{on} is the observed binding time, τ_{bound} is the true binding time, and τ_{bleach} is the characteristic fluorophore bleaching time. This gives the true binding times as:

$$\tau_{\text{bound}} = \frac{\tau_{\text{bleach}} \cdot \tau_{\text{on}}}{\tau_{\text{bleach}} - \tau_{\text{on}}} \tag{3.8}$$

τ_{bleach} is the time we obtain from fixed cell data acquired under the same experimental conditions as non-fixed samples (see next Section 3.2.7). For fixed cells, fluorophores are "fixed" in space and hence all signal appears as largely bound, with sharp PSF fits. This makes it easier to obtain accurate protein track lengths as image thresholding is less likely to fragment tracks. Motion blur is also mostly eliminated. A single exponential fit to the distribution of track lengths then gives τ_{bleach}, which accounts for photobleaching. If more than one binding process is observed for a protein, a double exponential fit can be used. This is not the case for the proteins we study as all are only bound to the DNA. Similarly, the on-time distribution gives τ_{on}. Previously, τ_{on} has been inferred from the total track length in long exposure experiments conducted on unfixed cells [15, 118]. Since we have per-frame information on the protein state, we can determine τ_{on} by aggregating consecutive on-times from a given protein track [120]. The above formula correctly accounts for photobleaching and yields τ_{bound}, provided that $\tau_{\text{bleach}} > \tau_{\text{on}}$. On-times that extend beyond the fluorophore bleaching time cannot determine binding times τ_{bound}. In this case, the fluorophore or experimental method needs to be altered.

For comparison, we present three approaches in the Results section for determining binding times:

- The **standard** approach - plotting σ_x and σ_y of Gaussian PSFs for fixed cell data provides two thresholds for filtering bound and diffusing molecules based on PSF width. We perform width-based filtering on the average width of the track. If the average track width is below the thresholds in both axes, the trajectory is considered bound and the total track length is taken as the on-time. This approach is sensitive to not only the width threshold but also the threshold used for localisation. We have attempted to apply this approach per-frame for binary classification but the results were not satisfactory (see Results). Traditionally this is performed for binding time calculations. This approach works well at high exposures, combined with careful averaging and thresholding [15].

- The **CNN-averaging** approach - this takes state sequences as given by the CNN and applies step-based filtering. If the the average state over the track is greater than 0.5, we take the trajectory to be bound and the on-time is the trajectory length. This approach should provide identical results to the standard approach and acts as a sanity check to ensure averaging the CNN outputs replicates the standard (non-CNN) approach.

- The **Spotwise** approach - this takes state sequences as before, but calculates consecutive binding times within the sequence. Thus, we expect this to be lower in value than the previous approaches, as it deals with subsets of the bigger state sequences. This approach is theoretically correct for all exposure times. The above two methods work only at long exposure times.

For all approaches, we consider tracks with a minimum of four localisations to remove noise. We also truncate binding times beyond 8 seconds, in line with previous work and to remove any unphysical false binding times (e.g., autofluorescence or other molecules in the cell/media illuminating for long durations) [15]. Connecting

spots is done using a memory parameter of 1. This can lead to some missing frames during prediction as no frame exists in between protein tracks, but this is rare as long exposure times average out effects of blinking (e.g., protein exists in frame 1, not in frame 2 due to blinking, re-appears in frame 3). Nevertheless, we replace any missing frames with the average of the raw classifier output "probabilities" of surrounding frames. At most, only one frame will be missing.

3.2.7 Cell Preparation

For PALM experiments, overnight bacterial cultures of *E. coli* MG1655 were grown in a rich lysogeny broth medium at a temperature of 37 °C. These cultures harboured an endogenous fusion of PamCherry with DNA Pol I (strains taken as by Uphoff *et al.* [15]). In preparation for PALM microscopy, the bacterial suspensions underwent a 1000× dilution in a minimal M9 medium, followed by incubation for 3 to 4 hours. Subsequently, 1 ml of the resultant solution was subjected to centrifugation at 8000 revolutions per minute for 3 minutes. The supernatant was removed, and a small amount of remaining solution was placed on top of 1% agarose pads. These agarose pads were made using M9 with 100 mM of Methyl Methanesulfonate (MMS) to induce DNA damage, thereby increasing the number of protein binding events (for DNA Pol I only) [15].

To calculate the bleaching lifetime of the fusion protein, the experimental procedure involved cell fixation in a 2.5% formaldehyde solution, combined with phosphate-buffered saline (PBS), for a duration of 45 minutes. Following fixation, the cells were subjected to thorough washing with PBS, repeated four times. The cells were subsequently deposited onto agarose pads for imaging, employing a 500 ms exposure time and utilizing an excitation (561 nm) laser intensity at $\sim 100\ W/cm^2$ and an activation (405 nm) laser intensity at $0 - 0.5\ W/cm^2$ (HILO). For RNAP, σ^{70} and σ^{54}, the steps were identical excluding the addition of MMS. Imaging was performed on the same microscope as described in Chapter 2.

3.2.8 Analysis of Classifier Output

In order to understand the CNN model output, we employ SHAP (Shapley Additive exPlanations) to analyse the output. SHAP leverages game theory principles to provide visual insights into specific regions within an image that influence the model's decision-making process. Briefly, SHAP works by segmenting images into regions. Then, model classification is performed on the image with various regions present or absent, allowing for quantification of important pixels in the image. SHAP serves as an explainable AI algorithm and allows for deeper understanding of the factors governing model classification [125].

3.3 Results

3.3.1 Simulations

First, we train the model on data acquired from video simulations. The CNN underwent training on 50,000 Bound (1) and Diffuse (0) images each. These were sampled from a range of different conditions, each originating from simulations with different initial diffusion coefficients, PSF widths, and background noise level. The data was split into a ratio of train/test/val of 50:30:20. We train for 90 epochs with a batch size of 32, with the training loss curves shown on Figure 3.5.

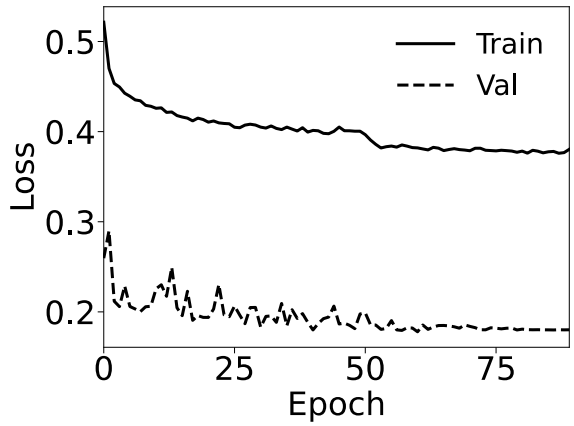


Figure 3.5: Loss curves during training. The training dataset shows a higher loss due to augmentations in the training data making this a harder task. Both losses plateau after an initial drop.

After training is complete, the model achieved an accuracy of 93.6% on the test dataset as shown on the confusion matrix in Figure 3.6. It is worth noting that the accuracy on the training data was lower at 88% as augmentations in the training data made this a more difficult task. Particularly, when multiple augmentations are applied to a specific training sample. However, this makes the classifier more robust to various experimental imperfections (e.g., out-of-focus spot or poor signal-to-noise). We did try to apply the standard approach on a per-frame basis to the test dataset by using width thresholds from Gaussian PSF fits to the training dataset. This returned an accuracy of 70% (and does not take into account any augmentations).

Looking at Figure 3.6, we can see the model predicts more false positives than false negatives, indicating the model more frequently misidentifies diffusing molecules as bound rather than the reverse. This can be explained with diffusing molecules temporarily appearing as bound when their motion is constrained to a small cellular region or if a bound molecule has just transitioned to a diffuse state. Truly bound molecules do not exhibit these characteristics and are more difficult to "appear" diffuse.

Following initial evaluation of the model on the test dataset, we evaluate the effectiveness of the spatial filter using protein displacements between frames. For this particular case, we simulate videos with an exposure time of 500 ms, a fast diffusion coefficient of 1 $\mu m^2/s$, a slow diffusion coefficient of 0.05 $\mu m^2/s$ and 360 protein tracks for a total of 100 frames. Not all proteins persist for all frames. We show the results of this analysis on Figure 3.7, highlighting the confusion matrices side-by-side after model classification with and without imposing the spatial window slider.

The spatial filter decreases the number of false positives by roughly half (1285 to 610). Diffusing molecules that "appeared" as bound molecules to the CNN were picked up by the spatial filter. These states were subsequently overwritten as the displacement between frames violated the conditions highlighted in Table 3.3. We also see a slight improvement in the number of false negatives (670 to 430).

For the same simulation, once we have obtained state sequences from the CNN+spatial filter, we can calculate protein binding times. Given that we have

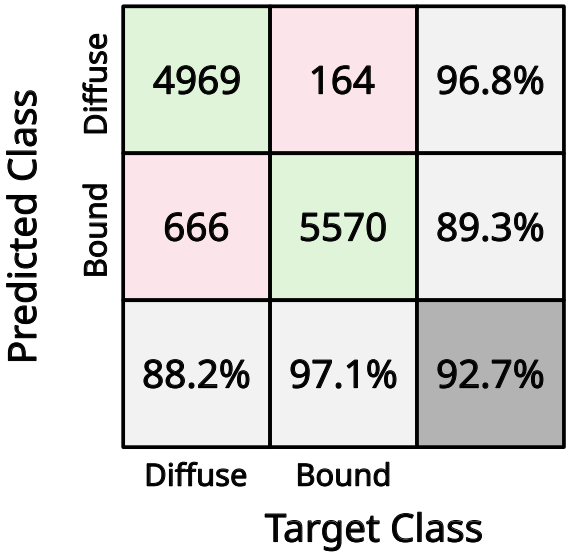


Figure 3.6: Confusion matrix for the test set. Average classifier confidence is 95%.

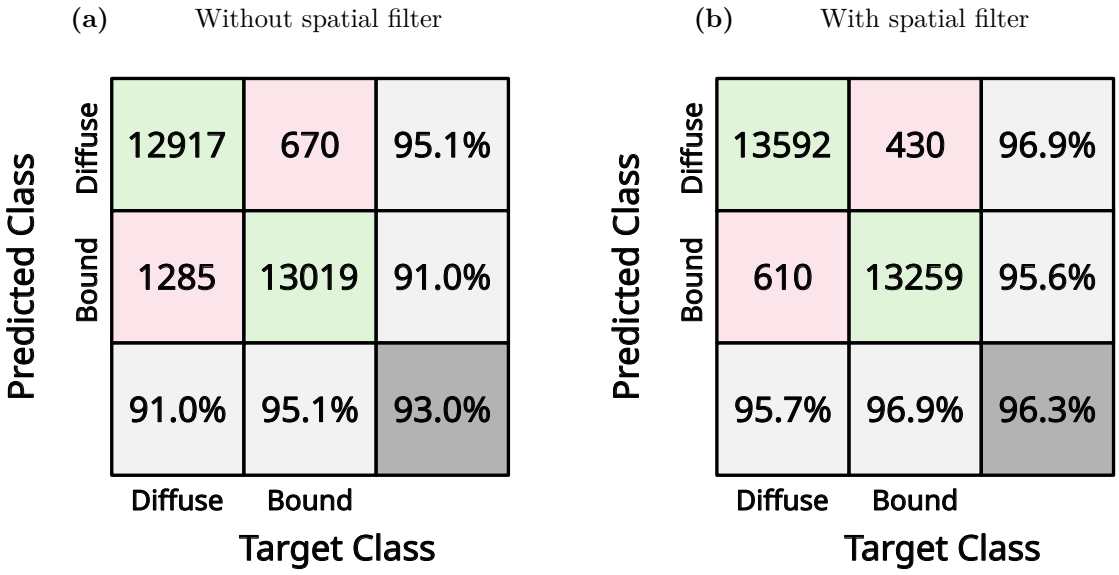


Figure 3.7: Spatial filter using diffusion equations and conditions given by Table 3.3. The filter decreases the number of false positives by roughly half (1285 to 610), improving the overall accuracy by 3.3%.

the ground truth for the simulations, we can directly compare calculated binding times with the ground truth. Figure 3.8 compares the computed bound times from the spotwise approach (CNN+spatial filter) against the ground truth. The plots show good agreement and provides an assessment of our method's ability to recover the underlying kinetic parameters. As shown in Figure 3.8, the spotwise approach

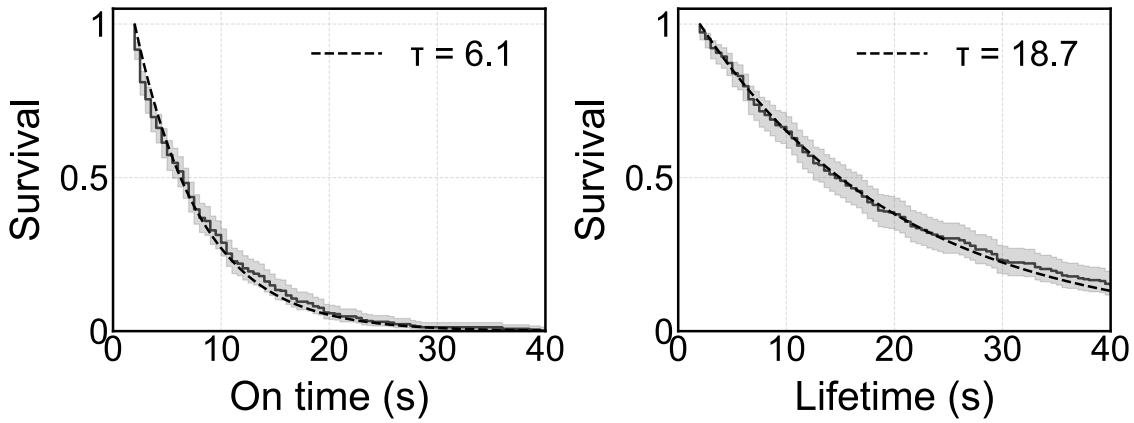


Figure 3.8: Survival distribution for observed on- (calculated using the spotwise approach) and bleaching times (lifetime) for simulated data. These give $\tau_{on} = 6.1\ s$ and $\tau_{bleach} = 18.7\ s$ for a corrected binding time of $\tau_{bound} = 9.1\ s$ using 331 trajectories. A ground truth calculation, in which the correct sequence is known, gives 9.5 s. The true binding time set in the simulation is 10 s. Light grey areas show 95% confidence interval. Dashed lines indicate single exponential fits.

Table 3.4: Bound time analysis for simulated proteins under varying fast diffusion coefficient and underlying binding kinetics. Exposure time is fixed at 500 ms and and the slow diffusion coefficient is fixed at 0.05 $\mu m^2/s$. The model predictions lie reasonably close to the ground truth. The bleaching time is 20 s and 325 protein tracks were used in the calculations.

Fast Diffusion Coefficient $\mu m^2/s$	**Predicted Time**	**Ground Truth**
0.75	$4.7 \pm 0.2\ s$	$4.8 \pm 0.2\ s$
0.5	$9.3 \pm 0.2\ s$	$9.1 \pm 0.2\ s$
0.5	$7.0 \pm 0.3\ s$	$7.8 \pm 0.3\ s$

(+ spatial filter) returns a τ_{bound} of 9.1±0.2 s (using Equation 3.8), with a ground truth calculation (from 331 trajectories) returning 9.5±0.2 s.

We apply the same procedure to determine the binding times of simulated proteins under slower values of fast diffusion coefficients and differing underlying kinetics. The results show agreement between ground truth and predictions from the model. This is highlighted on Table 3.4.

3.3.2 Explaining Model Classification

SHAP analysis reveals which pixels in an image most strongly influence our CNN's classification decisions. When applied to our model, SHAP helps us visualize exactly

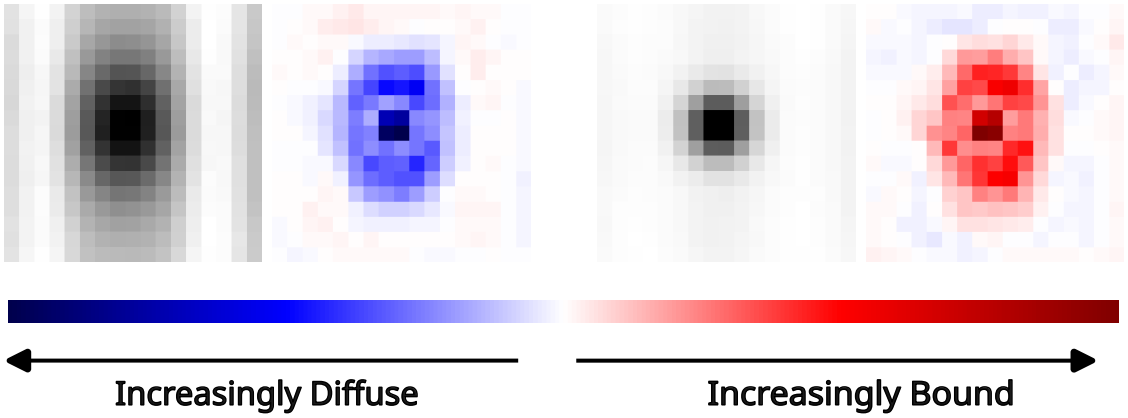


Figure 3.9: SHAP analysis revealing the most influential image regions for CNN classification. The heatmaps show the relative importance of different pixel regions in determining the model's classification decision. For bound states (red), the model identifies a characteristic bright, narrow peak, while diffusing states (blue) are characterized by broader, dimmer distributions of pixel values. This pattern aligns with what we expected to see and confirms the model is not learning noise patterns. SHAP is applied to 100 randomly sampled images of each class from the simulated test dataset.

what features in the image the CNN focuses its attention on. SHAP values, derived from game theory, provide a quantitative measure of each pixel's importance to the classification. Figure 3.9 displays these key regions for both bound and diffusing particle classes. The analysis shows that the CNN primarily focuses on two areas: the central pixels and their immediate surroundings.

For each class we have:

- Bound particles appearing as bright, narrow peaks (high central intensity with rapidly decreasing surrounding values)
- Diffusing particles show dimmer centres with a broader intensity distribution (surrounding pixels have values closer to the centre)

It is important to note that the blue/red regions around the centre suggest the model is detecting the presence or absence of these pixels. These results confirm that our CNN is learning meaningful physical features rather than arbitrary patterns during training. Furthermore, through data augmentation, the model has become robust enough to classify images with various distortions, eliminating the need to manually tune width and intensity threshold parameters for each dataset.

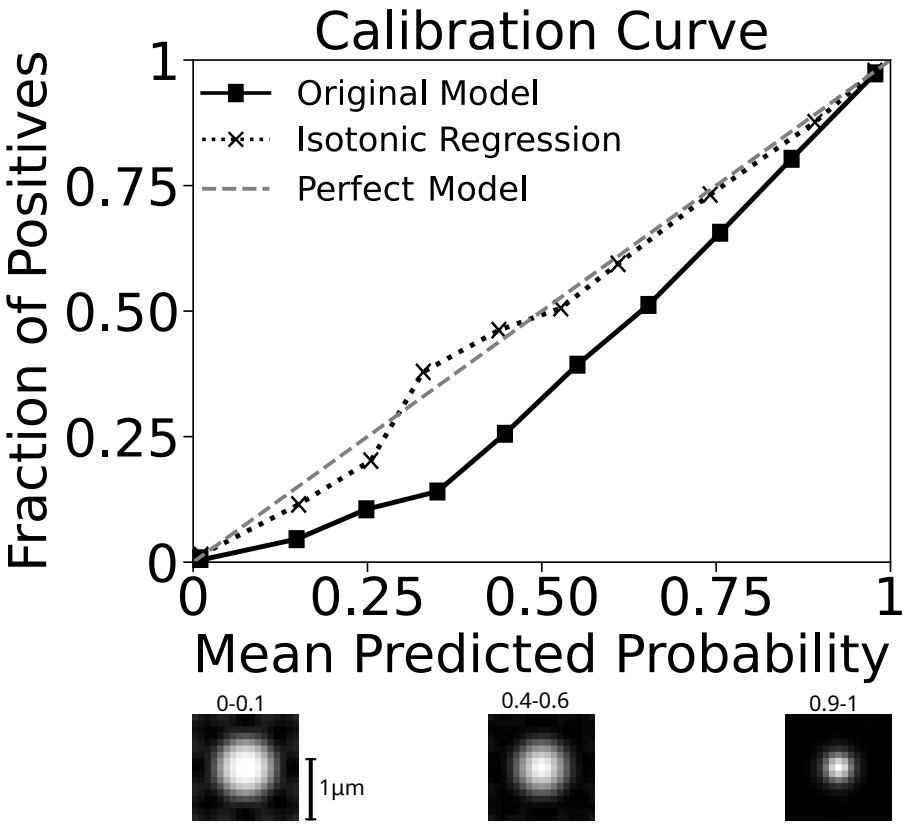


Figure 3.10: Calibration curve on the testing simulation data based on an Isotonic Regressor fitted on the simulated validation dataset. The original model shows output values from the CNN sigmoid function. The changed classification values did not change model accuracy. Simulated images show images (averaged over all rotations) for respective bins based on raw classifier output. If we wish to explore partial/mixed frames, a spot could be assigned a fractional probability of being bound from the classifier output, representing the average time (within the exposure time) that it was in the bound state, but this needs to be explored further as out-of-focus effects may complicate matters. Ground truths are not readily available from PALM experiments so care should be taken (in which case Isotonic Regression may be difficult).

On Figure 3.10, we also show the calibration curve for output values from the CNN classifier. The calibration curve works by binning predicted probability values into intervals and comparing the average predicted probability within each bin to the actual fraction of positive outcomes. As seen in the figure, the classifier tends to be overconfident in its predictions of the bound class — for example, when the model predicts a probability of 0.5, the actual fraction of positives is lower, indicating overestimation of probabilities.

An Isotonic Regressor corrects for this, producing calibrated outputs that better reflect true probabilities. The initial miscalibration observed in Figure 3.10 may

be due to augmentations applied during training. These augmentations are more likely to transform bound (1) images to appear diffuse (0) than the reverse (e.g., through added shear or noise). As a result, the model may learn to associate high output values with augmented images. This can cause the model to assign higher probabilities of being bound to truly diffuse inputs at inference. To test whether augmentations have caused the miscalibration, the model can be retrained without augmentations and a new calibration curve can be plotted. Care was taken when tuning the augmentation parameters to prevent bound images from appearing diffuse. However, the applied transformations may still have been too strong. Certain augmentations (e.g., shear) could be omitted or further adjusted to avoid overly altering the appearance of the images. The calibration curve for various subsets and strengths of augmentations can also be plotted, which can inform us of optimal augmentation parameters (until we observe miscalibration).

3.3.3 Application to *in vivo* PALM

Before application to experimental PALM images, the model was fine-tuned using the same initial hyperparameters and augmentations (with the exception of the number of epochs being set to 30) on a manually curated set of 497 bound and 497 diffusing images specific to the microscope. Data was obtained through manual classification from experimental images that clearly showed bound and diffusing images. After fine-tuning, the model classified 98/100 images correctly on the test set. The f1-score is also 0.97 (0.99 precision and 0.95 recall). We note the small test dataset size given the fact that segmenting real data from PALM experiments is difficult and tedious. However, the model does see a variety of images during training as augmentations artificially increase diversity of the dataset. We will return to this in the Discussion.

Below we highlight the **standard**, **CNN averaging**, and **spotwise** approaches to calculate binding times (to the DNA) of DNA Pol I, RNAP, σ^{70} and σ^{54}. For all samples the photobleaching correction was obtained using fixed cell data. The photobleaching correction can vary if the imaging conditions change, causing

fluorophores to bleach at different rates. Thus, for every protein, we image under normal conditions as described in the Methods but also obtain a complementary fixed cell dataset. This is done using the same imaging conditions and immediately after normal conditions, providing accurate photobleaching times. For use in the step-based filter, DNA Pol I has D^*_{fast} of 1 $\mu m^2/s$ and RNAP has D^*_{fast} of 0.36 $\mu m^2/s$, with all DNA-binding proteins having a D^*_{slow} of 0.1 $\mu m^2/s$ [46]. For the sigma factors, we used 0.4 $\mu m^2/s$ as the fast diffusion coefficient for the step-based filter. This is the mobility of the holoenzyme state during promoter search (as it is a three state system we take the minimum fast diffusion coefficient, see Rich M9 diffusion distribution in Chapter 2).

DNA Pol I

For initial verification of our model *in vivo*, we measured the time that a DNA Pol I PamCherry fusion binds to the DNA in *E. coli*. We first determine the photobleaching correction from fixed cell data (see Section 3.2.7 Cell Preparation). Plotting the survival function for the lifetime of the protein fusion, we obtain Figure 3.11, with an average lifetime of 2.7 ± 0.1 s at an exposure time of 500 ms.

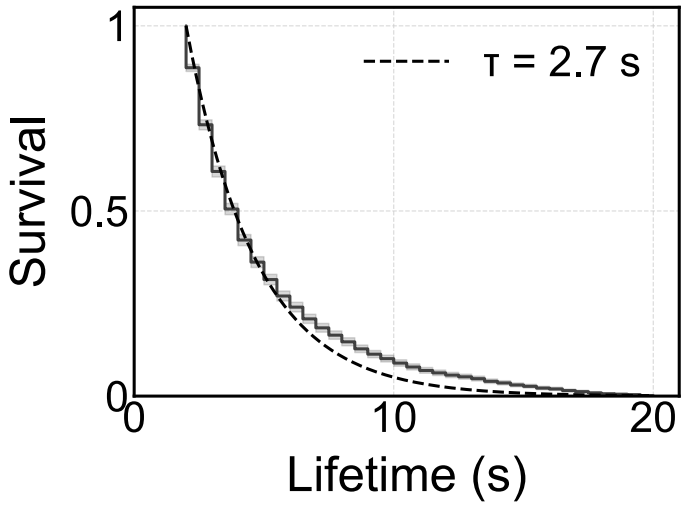


Figure 3.11: Survival distribution for fixed cell lifetimes using DNA Pol I tracks. For every track, we take the total time the protein is fluorescent as its lifetime. The lifetime plot is calculated from 4162 tracks. Light grey areas show 95% confidence interval. Dashed line indicates a single exponential fit.

Standard approach

For this approach, we require histograms of the spot width in the x and y directions from the spot PSF fitting as done in Chapter 2. We show the distributions of σ_x and σ_y from fixed cell data on Figure 3.12. From the Figure, we obtain a threshold of 2.3 pixels for the x (minor axis of Gaussian PSF) width and 3.3 pixels for the y (major axis) width. Any given track, from unfixed cell data, is considered bound if the width in both directions is less than the threshold values.

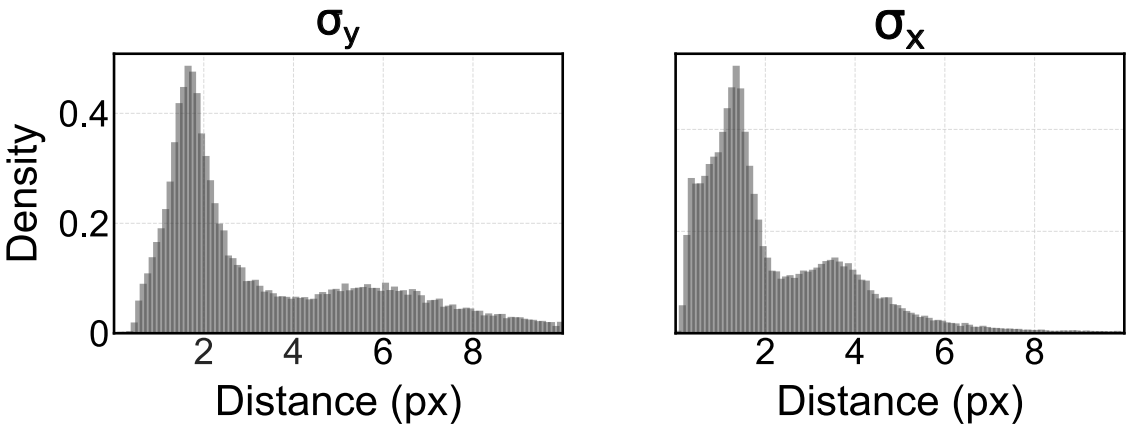


Figure 3.12: Fixed cell data using DNA Pol I localisations. The widths in the minor (x) and major (y) allow us to obtain hard thresholds for distinguishing bound and diffusing proteins. Notice there will be some overlap between the first peak (representing bound molecules with tight PSF fits) and the second peak (representing diffuse molecules with broad PSF fits). In the standard approach, we use hard threshold values of 3.3 and 2.3 pixels ($1\ px = 100\ nm$ separating the two peaks (for σ_y and σ_x). Tracks from unfixed data are considered bound if the average track width falls below both of these values. The histograms show a total of 43158 spots.

We now plot the on-time distribution on Figure 3.13, which is the total track length (unfixed conditions) for tracks with average width less than both thresholds (2.3 and 3.3 pixels). This gives an average on-time of $1.3 \pm 0.1\ s$.

Using the above time constants, we can calculate the binding time from the standard approach as:

$$\tau_{\text{bound}} = \frac{2.7 \cdot 1.3}{2.7 - 1.3} = 2.5 \pm 0.4\ s \tag{3.9}$$

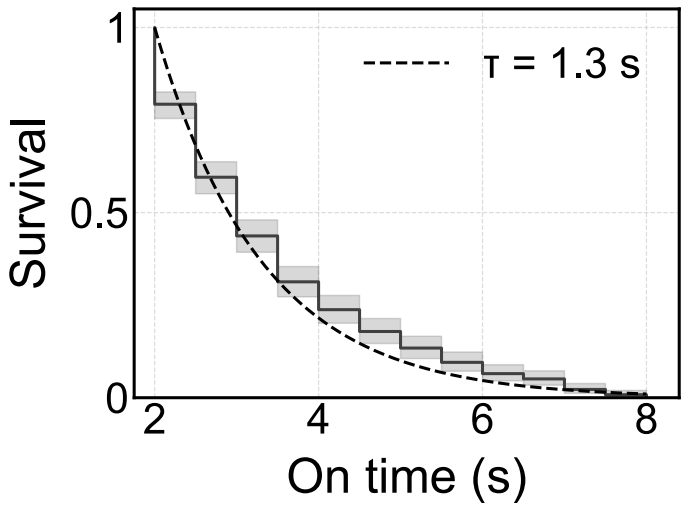


Figure 3.13: Survival distribution for DNA Pol I on-times that satisfy width thresholds. We obtain an average on-time of 1.3 ± 0.1 s. Total tracks: 437. Light grey areas show 95% confidence interval. Dashed line indicates a single exponential fit.

CNN averaging

We can perform the same procedure using the CNN averaging approach. The only change is the calculation of the on-time distribution, which does not use width thresholding. Therefore, we do not need the additional step of determining σ_x and σ_y from fixed cell data. The on-time distribution is the length of the track given it is classified as bound on average over its lifetime. This distribution is shown on Figure 3.14a. We obtain:

$$\tau_{\text{on}} = 1.4 \pm 0.1 \text{ s}$$
$$\implies \tau_{\text{bound}} = 2.7 \pm 0.4 \text{ s}$$

Spotwise Approach

We perform the same procedure using the spotwise approach as described in the Methods. The on-time distribution is shown on Figure 3.14b. Again, we obtain:

$$\tau_{\text{on}} = 1.2 \pm 0.1 \text{ s}$$
$$\implies \tau_{\text{bound}} = 2.2 \pm 0.3 \text{ s}$$

Both standard and CNN averaging approach produce similar values, verifying once frame level predictions are smoothed, classical results are obtained. The

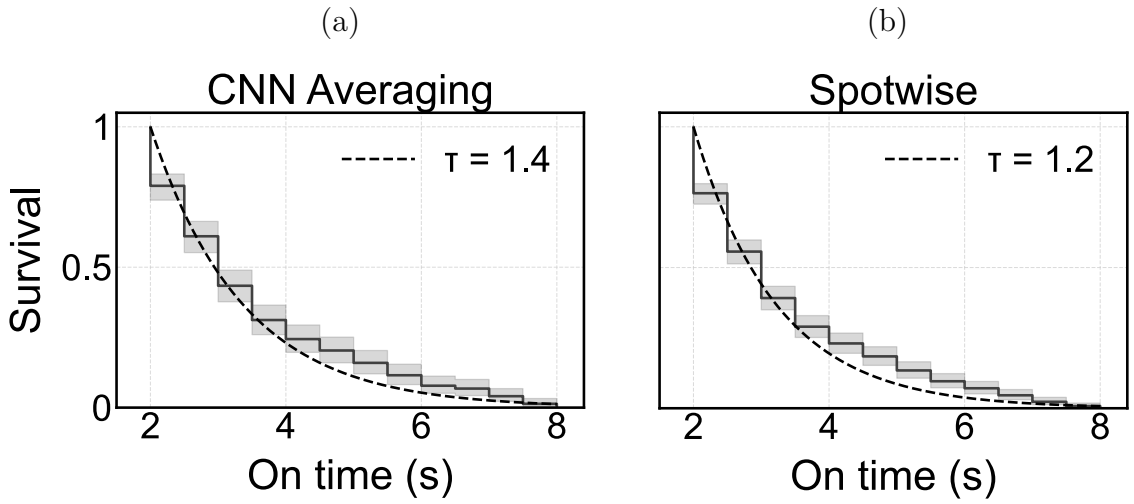


Figure 3.14: Survival on-time distributions for (a) CNN averaging with a total of 295 tracks (b) Spotwise approach with a total of 521 data points (subset of tracks). The spotwise approach gives a lower binding time by nature, as it consists of subsets of the original tracks. Light grey areas show 95% confidence interval. Dashed lines indicate single exponential fits.

spotwise approach gives a smaller binding time by nature of the method. All of the above approaches produce similar values within error.

Out of all tracks, only 55 tracks (out of 295, representing 18%) have a minimum of four localisations in one state and transition to a minimum of four localisations in another state (see Figure 3.15). These proteins have an average length of 19 frames (9.5 s), affecting potential binding at higher times. This leads to the spotwise approach returning similar values. In fact, with PAmCherry, given its fast bleaching and small lifetime, it is hard to observe a large number of transitioning molecules. However, we benefit from the added advantage of frame level state prediction and robustness to experimental imperfections.

All approaches return reasonably close values and are in agreement with the 1.5-2.5 s binding time for DNA Pol I found by Uphoff *et al.* [15].

We repeat the same analysis for RNAP, σ^{70} and σ^{54}. The results are shown in Table 3.5. All protein binding time values are in the range of $\sim 2-3$ seconds.

Given the error range ($\pm 0.3-0.4$ s), these values should not be taken as definite, instead serving as a rough timescale. Of note, is that the binding times of RNAP, σ^{70}, and σ^{54} are quite similar. At the imaging times, and given our fast bleaching

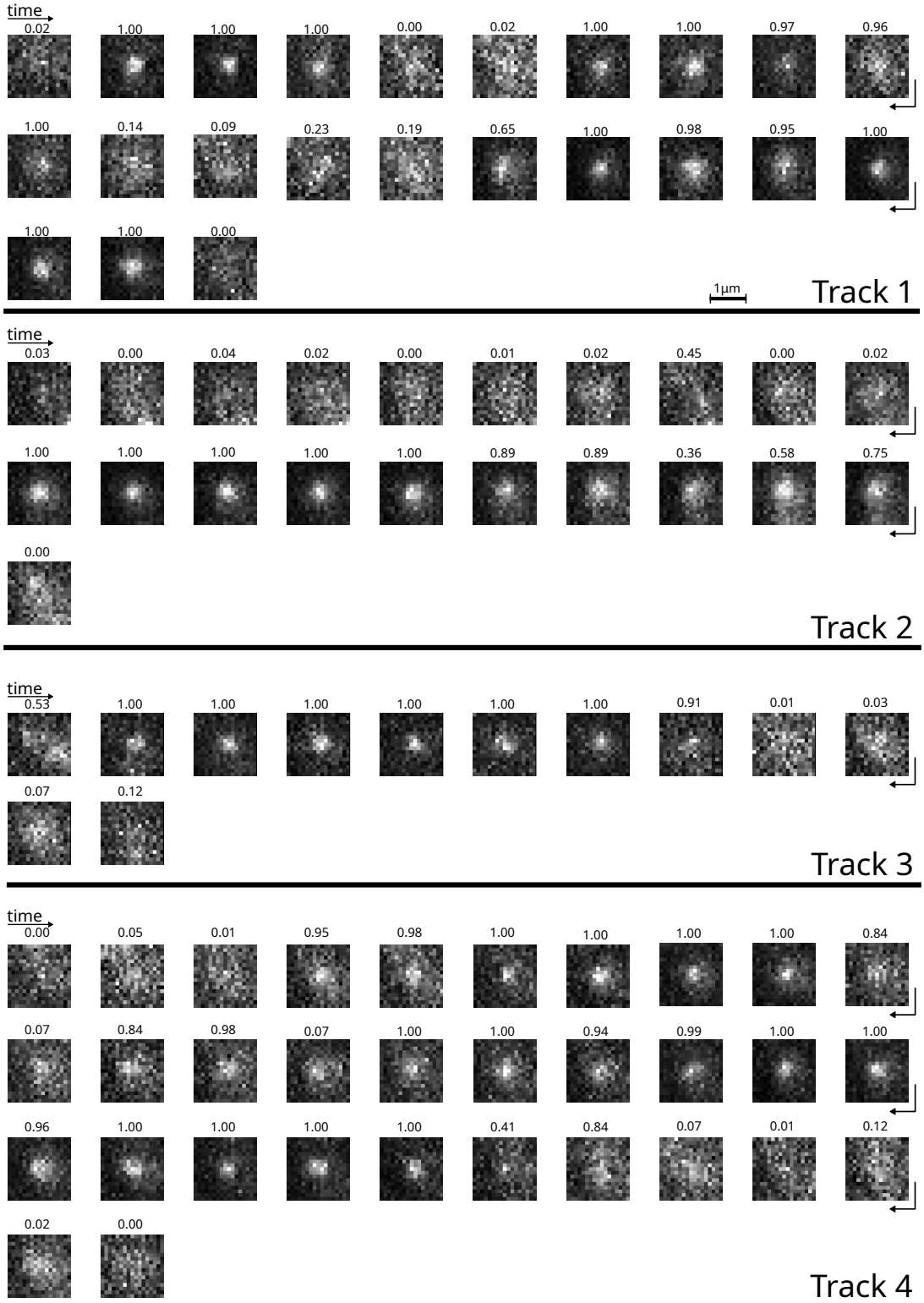


Figure 3.15: DNA Pol I molecules undergoing transitions with CNN outputs above each frame (unseen data from experiment). Images such as Track 1 Frames 10 & 11 highlight some "hard-to-classify" cases. The above tracks can be filtered out from single molecule videos and studied further if needed.

Table 3.5: Comparison of binding time measurements across DNA-binding proteins and analysis approaches. For standard, CNN averaging and Spotwise, we have the following number of tracks for each protein respectively: RNAP (3877, 2119, 3113) with 134 transition molecules, σ^{70}(4282,1482,3074) with 272 transition molecules, σ^{54}(5441, 1104, 2885) with 326 transition molecules. For each protein we followed the procedure as outlined for DNA Pol I. This includes finding the width thresholds for σ_x and σ_y, the total lifetime for photobleaching corrections, and then following the rest of the methods as discussed. Exposure time during PALM imaging is 500 ms.

Protein	**Method**	$\tau_{\text{bleach}}(s)$	$\tau_{\text{on}}(s)$	$\tau_{\text{bound}}(s)$
RNAP	Standard	3.5 ±0.1	1.6 ±0.1	3.0 ±0.3
	CNN Averaging	3.5 ±0.1	1.8 ±0.1	3.7 ±0.4
	Spotwise	3.5 ±0.1	1.6 ±0.1	3.0 ±0.3
σ^{70}	Standard	3.2 ±0.1	1.3 ±0.1	2.2 ±0.3
	CNN Averaging	3.2 ±0.1	1.4 ±0.1	2.5 ±0.3
	Spotwise	3.2 ±0.1	1.2 ±0.1	2.0 ±0.3
σ^{54}	Standard	2.6 ±0.1	1.3 ±0.1	2.6 ±0.4
	CNN Averaging	2.6 ±0.1	1.4 ±0.1	3.0 ±0.5
	Spotwise	2.6 ±0.1	1.2 ±0.1	2.2 ±0.4

fluorophore, we do not expect to observe transcription events by RNAP since these take up to ~ 60 s [19]. We also do not expect to see long-lived gene repression by σ^{54}. It is possible that some of the picked up signal represent fractions of these processes, which may bias the binding times. However, given the fluorophore (PAmCherry) it is not easy to tell the difference between short-lived transient/non-specific interactions and fractions of long-lived interactions within its bleaching time.

For the case of DNA Pol I, its binding time reflects the characteristic timescale at which it repairs damaged DNA, as shown by Uphoff *et al.* [15]. In the case of RNAP and the sigma factors, these binding times could represent transient interactions with the DNA during promoter search (e.g., abortive transcription). However, proving this with certainty requires further experimentation. This may involve a more photostable fluorophore such as Halo tag, with a lifetime up to minutes, making it potentially easier to distinguish between short and long-lived interactions (although an absolute ground truth is unavailable). Note that transitioning molecules make up a small fraction of the total dataset, leading to similar values across all methods. Additionally, molecules with evident transitions between states tend to have longer

lifetimes, leading to changes in the binding time at tail ends of the single exponential graphs, with little effect on the characteristic timescale. This is a limitation of the fluorophore (PAmCherrry) we are using.

3.3.4 Breaking Point

The performance of the classifier (without the addition of step-based filters) is intrinsically linked to the diffusion coefficient and the exposure time. The classifier uses visual differences due to motion blur to classify between bound and diffusing states. For a fixed diffusion coefficient, as the exposure time decreases to smaller values, the effect of motion blur decreases. On the other hand, for a fixed exposure time, as the difference in diffusion coefficient between the two states decreases, the difference due to motion blur of the two states also decreases. Hence, the model can accurately classify states at relatively lower exposure times if the freely diffusing protein state has a high mobility.

The mean square distance moved by a given protein between frames determines the level of motion blur. This is proportional to $\Delta D^* \tau_{exp}$, where ΔD^* is the difference in diffusion coefficient between the two states. Thus, we take $\Delta D^* \tau_{exp}$ as a good metric for estimating the effectiveness of our model for various diffusion coefficients and exposure times [126, 127]. Usually, $D_d^* \gg D_b^*$, such that we can fix D_b^* and vary D_d^*.

By simulating various protein tracks for different combinations of D_d^* and τ_{exp}, and evaluating the model's accuracy for binary classification of the resulting cropped images, we can determine the model's breaking point. This point also serves as a threshold in exposure time at which we consider a protein to be imaged at "long" exposure.

In total, we simulate 110 experiments, with D_d^* ranging from $0.1 - 1$ $\mu\text{m}^2/s$ and the exposure time ranging from $25 - 250$ ms. Per experiment, we obtain 360 protein tracks. We fix the slow diffusion coefficient, D_b^*, at 0.05 $\mu\text{m}^2/\text{s}$. For every set of parameters we determine $\sqrt{\Delta D^* \tau_{exp}}$ (for a direct comparison with a distance

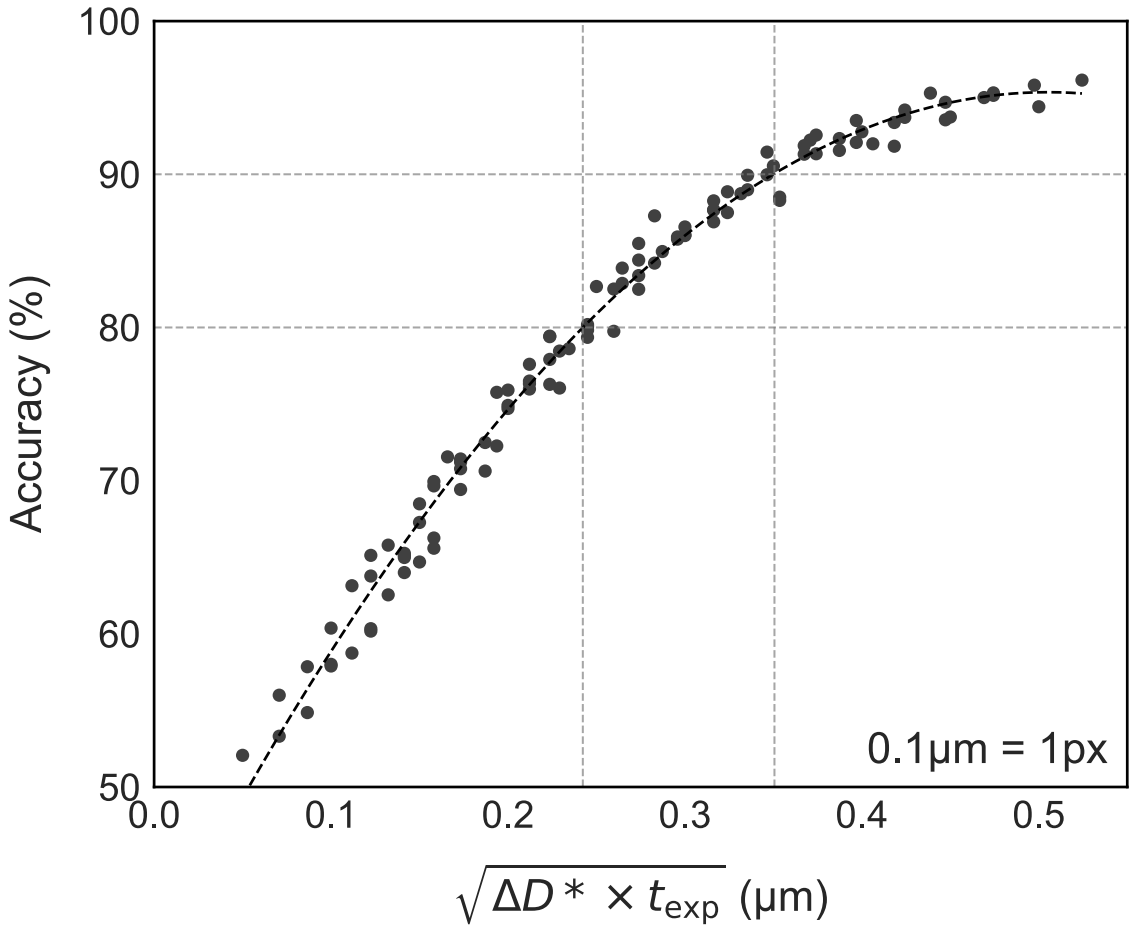


Figure 3.16: Determining the breaking point of the model. At some point bound and diffusing images start to look similar due to decreased motion blur. This varies for proteins depending on the fast diffusion coefficient. The results also quantify the point at which "long" exposure kicks in.

moved, τ_{exp} in seconds) and the model accuracy (given by number correctly classified divided by total number of images). The result is shown in Figure 3.16.

From Figure 3.16, we can see achieving an accuracy of >90% in classifying between different protein states requires $\sqrt{\Delta D^* \tau} \gtrsim 0.35$. This criterion can be adjusted to accept a lower accuracy, with a $\sqrt{\Delta D^* \tau} \gtrsim 0.25$ being adequate for an 80% accuracy rate.

The smallest diffusion coefficient for the fast moving diffusing population of a DNA-binding protein, is given by the HU (Heat Unstable) protein at 0.33 $\mu m^2/s$ [46]. While still maintaining an accuracy rate of 90% or higher, the minimum exposure time, τ_{exp}, is then calculated as $\tau_{\text{exp}} = 0.35^2/0.33 = 370$ ms. Thus, the CNN is able to handle data from this protein at exposure times of 500 ms as done

for the other DNA-binding proteins. In the event of accepting a reduced accuracy rate of 80%, the exposure time can be brought down to $\sim$ 190 ms.

We can also consider DNA Pol I, which, when not attached stably to the DNA, diffuses fairly rapidly at 1 $\mu m^2/s$. For this protein, the exposure time can be as brief as $\sim$ 120 ms, while preserving an accuracy rate exceeding 90%. The point at which "long" exposure is attained thus depends on the nature of the protein itself. Proteins with much smaller free diffusion coefficients need to be imaged at higher exposure times. However, the given protein will also have physical functionality within the cell. This has to be taken into account at longer exposure times, in order to ensure that underlying processes are not rendered unobservable due to longer exposure times (behaviour washes out given the timescale). The breaking point analysis assumed no additional step-based corrections between frames, which of course can be layered onto the CNN to further increase classification accuracy and reduce minimum exposure times for adequate accuracies. We have also not specifically focused training on cases where the distinction in ΔD^* is small, although this may be difficult.

3.3.5 Image Quality

Lastly, we evaluate the CNN and standard approach for classification of simulated Bound (1) images with increasing signal-to-noise ratio (SNR) and defocusing.

We simulate 17×17 matrices with two dimensional Gaussian PSFs at the centre (see Equation 2.1) with $\sigma_x = \sigma_y = 2$ and amplitude (A) equal to 1. We add background noise to this matrix, sampled from a Gaussian distribution with zero mean and standard deviation (σ_N) given by $\sigma_N = A/SNR$. Then, we min-max normalise each matrix. From the definition, we can observe an SNR of 1 leads to a completely noisy image, as the background noise is equivalent to the PSF amplitude.

In order to evaluate defocusing, we make use of the defocusing augmentation provided by the Albumentations Python package. This simulates the effect of an out-of-focus camera by applying a combination of circular disc kernels and Gaussian blur (see also Table 3.2). The defocusing radius (pixels) controls the kernel size, with a larger radius leading to increased defocusing [123]. We perform defocusing for

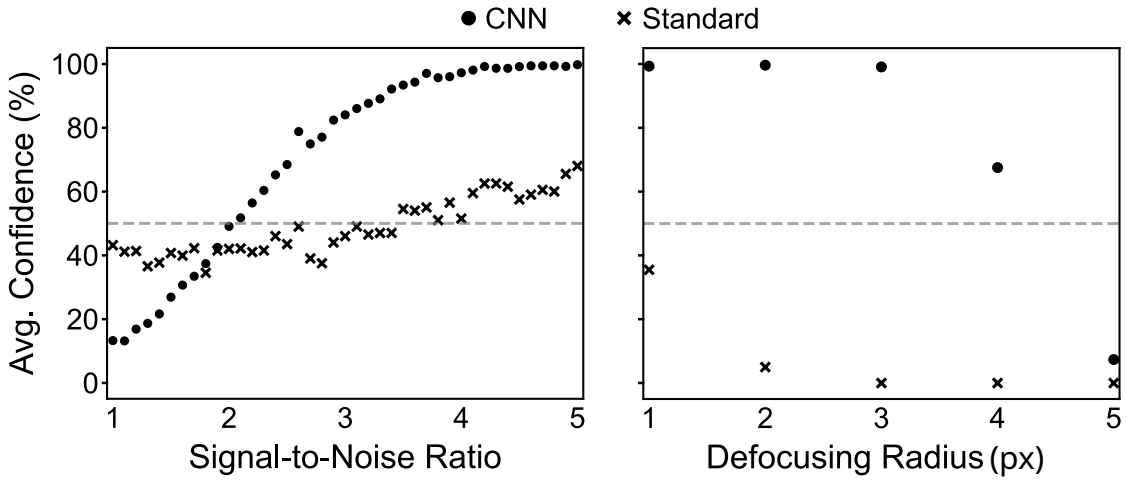


Figure 3.17: Average confidence of being Bound (1) by approach. Each point represents average confidence from predictions on 200 images for the given signal-to-noise ratio or defocusing radius (px = pixels). The grey line represents 50%, the threshold for classification, with points above being classified as Bound (1). Note the CNN performs better, alongside being more confident, for lower signal-to-noise ratios and higher out-of-focus. The augmentations in Table 3.2 used a maximum defocusing radius of 3, but this can be increased if needed.

an image with a Gaussian PSF of $\sigma_x = \sigma_y = 2$, $A = 1$, and $SNR = 4$. The standard approach was evaluated by using width thresholds on σ_x and σ_y from Gaussian PSF fitting to the images after noise has been added. Initially, we performed classification per image using width thresholds of 2 as this was used to simulate the PSFs. However, this led to the standard approach classifying the majority of images as being Diffuse (0). For a fairer and more generous comparison, we thus increased the width thresholds to 2.2. This adjustment does highlight the downfall of this approach, as constant width tuning is needed. We show the results of increased SNR and larger defocusing radii on Figure 3.17.

3.4 Discussion

Probing the temporal dynamics of protein interactions with the DNA is essential for unravelling intricate biological mechanisms. Prior work has utilised single molecule methods such as PALM *in vivo* at long exposure times and mean track features (such as mean spot width and diffusion coefficient) to determine protein binding times to the DNA. While effective under certain conditions, methods based on averaged

quantities sacrifice crucial insights into the trajectory of protein movement. Here, we implemented a CNN and spatial filter for binary classification of every frame within a protein trajectory, achieving greater than 95% accuracy. This methodology offers a higher level of granularity in capturing the diffusive states of a protein, thereby enabling more precise and flexible determination of binding times. Notably, the CNN-based framework is applied to calculate the binding times of DNA Pol I, RNAP, and sigma factors σ^{54} and σ^{70} in living *E. coli* cells. For these proteins, we find binding times in the $2-3$ s timescale. With regards to transcription proteins, this could represent transient interactions with the DNA as in the process of promoter search. At this timescale, we do not expect any specific long-lived interactions. Further experimentation will be needed to verify this, similar to work done by Uphoff *et al.* for DNA Pol I, in which its binding time represented DNA repair timescales [15]. Decreasing exposure times can provide more granularity on the exact binding times and mechanisms, but this would also lead to shorter observation times (fluorophore bleaches quicker due to increased sampling).

The CNN approach returned similar binding times as the standard classical approach. This validates the method as a tool to calculate binding times, while also gaining the added advantage of accurate per-frame classifications. Particularly, the model may be used to accurately detect transitions for a protein exhibiting multiple diffusive states (given sufficient training data), which is not possible with prior approaches. We have not been able to show this extensively due to our fluorophore (PAmCherry) bleaching quickly, but this can serve as future work for more photostable fusions. Again, we thank Hafez El Sayyed for making the RNAP, σ^{54}, and σ^{70} PAmCherry fusions available.

The model starts to break down as exposure times reach ∼100 ms and lower. The limiting cases can be seen for the HU protein and DNA Pol I. In the case of HU, the freely diffusing population has an apparent diffusion coefficient of 0.33 μm^2/s, requiring an exposure time of at least 370 ms (for >90% accuracy). On the other hand, the freely diffusing population of DNA Pol I has a coefficient of 1 μm^2/s [46]. For the same accuracy, we only require a minimum exposure time of 120 ms

(3× smaller). The term "long exposure" varies drastically for different proteins as motion blur starts to take effect at varying exposure times.

Throughout this work, DNA-bound proteins were fused to PamCherry and imaged using PALM. However, the fluorophore, experimental method, and the ability of the protein to bind to the DNA are not specific to the model. The model can accurately classify any protein-fluorophore system that exhibits visually different states regardless of the experimental technique. If the system has greater than two states, the last layer of the CNN can be extended fairly easily to include another class, provided accurate labels. The approach is general, parameter-free, and can be supplemented with additional filtering steps. We implemented a step-size filter using 2D protein coordinates, but this can be omitted or replaced with any other filtering method. The inclusion of specific augmentations reflecting experimental imperfections makes the model less susceptible to factors such as reduced image quality and out-of-focus molecules.

To avoid misclassification, the training data minimised for scenarios in which multiple fluorophores are active in the same cropped window. Experiments using PALM also ensured that a given cell had at most one active fluorophore at any time. The model has not been trained for situations in which multiple fluorophores are active in the same cropping window, which should be considered when applying to custom data. Ideally, this should not be the case as it would also lead to worse protein tracking from localisation data.

Alongside using long exposure times paired with low excitation laser intensities, stroboscopy has been used to image proteins for seconds to minutes. Since typical stroboscopic exposure times range from $10 - 50\ ms$, the application of the model to these timescales will be difficult and is a limitation of the model. The success of accurately classifying states from stroboscopic data would be highly dependent on the protein mobility and exposure time. Improvements could be made to the model architecture, such as using Recurrent CNNs, which would require training on protein tracks in temporal order, or the use of Siamese CNNs, returning a similarity

metric between pairs of images. A threshold can then be applied to distinguish states, although this may be dependent on the specific exposure time used.

Manual curation of training data can at most take ∼1 day of work and this alters the model weights to cater for images from the specific microscope. Labelling of experimental data can potentially be removed through altering the simulation parameters to exactly match the protein images from the microscope (or using histogram matching). This will require simulations accurately encompassing microscope characteristics.

The inherent simplicity and parameter-free nature of the CNN render it user-friendly as expert tuning is no longer needed. The CNN is also very flexible, making it a valuable tool for protein state classification and bound time determination across a wide range of long exposure scenarios. The methods discussed in this chapter lay the groundwork for future binding time calculations from noisy image data.

In this chapter, we focused on the specific case of two states (Diffuse (0) and Bound (1)) on a per-frame basis, while imaging at long exposure times. In the next chapter, we aim to generalise this framework for all exposure times and a range of diffusive behaviours.

4

Diffusive Properties in Heterogeneous Trajectories

Contents

4.1 Introduction 93
4.1.1 Brief Overview 93
4.1.2 Difficulties in Determining Protein Properties 93
4.1.3 Anomalous Diffusion Challenge 96
4.2 Methods 98
4.2.1 Recurrent Neural Networks 98
4.2.2 Model Architecture 101
4.2.3 Feature Selection 103
4.2.4 Generating Training Data 105
4.2.5 Augmentations 107
4.2.6 Training 108
4.2.7 Post-Processing 109
4.2.8 Changepoint Detection 109
4.2.9 Evaluation Metrics 110
4.3 Results 111
4.3.1 Pointwise Inference of α, D, and s 112
4.3.2 Dependence on Trajectory Length 115
4.3.3 Changepoints 115
4.3.4 Dependence on Biological Behaviours 122
4.3.5 Shortcomings 123
4.4 Discussion 125

4.1 Introduction

4.1.1 Brief Overview

The preceding chapters have established foundational methodologies for both protein imaging analysis and machine learning applications in single molecule tracking. Chapter 2 explored conventional ensemble approaches, particularly MSD analysis, for the extraction of diffusion coefficients and state classification. Chapter 3 extended this framework through the implementation of CNNs, specifically optimized for proteins imaged at long exposure times, enabling frame-level classification and downstream binding time calculations.

This chapter aims to generalise these approaches by introducing a novel machine learning method capable of detecting complex diffusive states beyond the scope of traditional MSD analysis, while being applicable across multiple temporal scales. The developed method addresses key limitations of both conventional approaches and provides a comprehensive solution for analysing heterogeneous protein dynamics.

4.1.2 Difficulties in Determining Protein Properties

As highlighted in the previous chapters, protein diffusion is vital to understanding cellular processes and the mechanisms which govern cellular functionality [7, 117, 128, 129]. Advancements in single molecule imaging have allowed for direct experimental observation of protein movement [7, 10, 11, 130–134]. PALM [7, 10] has been implemented *in vivo* to study cellular mechanisms such as DNA repair [15, 20, 119, 135], binding and diffusion of the *lac* repressor [24, 52, 117], and RNAP clustering [121, 136–139]. These studies have made extensive use of spatial-temporal properties of protein trajectories such as MSDs and diffusion coefficients. Subsequent hard thresholding of diffusion coefficients allows for categorisation of proteins into different states with biological relevance as done in Chapter 2 (e.g., an immobile RNAP molecule, with a low diffusive coefficient, may be interacting with the DNA) [19, 109, 119, 140].

Although successful for a small number of states with relatively simple behaviours, these approaches are difficult to extend to complex behaviours such as changes in environment (e.g., viscosity changes), compartmentalisation or co-diffusion [47, 141, 142]. This is further complicated with proteins transitioning between different states, leading to inaccurate diffusion coefficients and overlapping states, causing misclassification [143] (e.g., see diffusion distributions in Chapter 2).

To address these issues, Muñoz-Gil *et al.* introduced the Anomalous Diffusion (AnDi) Challenge in 2020 [47]. This challenge focused on advanced statistical and machine learning approaches to predict anomalous exponents and diffusion type for protein trajectories undergoing at most a single changepoint between states. The anomalous exponent (α) is essential in understanding protein movement and is related to the diffusion coefficient (D) in two dimensions by the equation [144, 145]:

$$\text{MSD}(t) = 4Dt^{\alpha} \tag{4.1}$$

where:

$\boldsymbol{\alpha < 1}$: The protein trajectories have "confined" steps, corresponding to sub-diffusion; MSD $\propto t^{\alpha}$ (with $\alpha = 0$ representing immobile motion).

$\boldsymbol{\alpha = 1}$: The protein moves with no preferred direction in Brownian motion – MSD $\propto$ t. This was used in the previous chapters with $MSD \propto Dt$.

$\boldsymbol{1 < \alpha < 2}$: The protein trajectories have "directed" steps, corresponding to super-diffusion; MSD $\propto t^{\alpha}$ (with $\alpha = 2$ representing ballistic motion).

The above MSD formula can be used to describe fractional Brownian motion (fBm), a generalisation of Brownian motion, in which the increments in position are no longer independent [144–149]. Note that we have changed notation from $D^* \to D$ for convenience. The covariance function for fBm is given by [145]:

$$\text{Cov}(B_H(t), B_H(s)) = \frac{D}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H}) \tag{4.2}$$

where B_H is a stochastic process (in our case protein position with time), H is the Hurst exponent given by $\alpha = 2H$ and t and s are time points [145].

Since fBm is a zero-mean process, we have:

$$\begin{aligned} \mathrm{Cov}(B_H(t), B_H(s)) &= E[B_H(t)B_H(s)] - E[B_H(t)]E[B_H(s)] \\ &= E[B_H(t)B_H(s)] \text{ as } E[B_H(t)] = 0 \end{aligned} \tag{4.3}$$

This can then be written as:

$$E[B_H(t)B_H(s)] = \frac{D}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H}) \tag{4.4}$$

where $E[\cdot]$ is the expectation function.

When $\alpha \neq 1$, different segments of the protein trajectories are negatively or positively correlated, leading to "confined" or "directed" behaviour respectively.

The outcome of AnDi 2020 led to more accurate machine learning techniques compared to traditional MSD methods [47]. Machine learning models, developed both during and after the challenge, have implemented recurrent neural networks, convolutional neural networks, graph neural networks, feed-forward networks with feature engineering, and more recently transformers [150–169]. Models can predict diffusive properties per time step (pointwise), similar to what was done for image data in Chapter 3 using a CNN.

The success of the first challenge led to AnDi 2024 [49], in which the problem focused more specifically on fBm of proteins in different biological states and an arbitrary number of changepoints between these states. Given protein coordinates $x(t)$ and $y(t)$, the task is to infer $D(t)$, $\alpha(t)$, and the protein state, $s(t)$, over the lifetime of the protein. As the protein can transition between biological behaviours, $D(t)$, $\alpha(t)$, and $s(t)$ are piecewise constant functions and can undergo sudden changepoints at arbitrary unknown times. These are to be determined as part of the task. In this chapter, we develop a novel machine learning method to tackle AnDi 2024. The model placed in the Top 5 at the time of challenge closure. Since then, minor improvements have been made to the model architecture (including another normalisation layer) and fixing a training bug, which are reflected in this chapter.

4.1.3 Anomalous Diffusion Challenge

AnDi 2024 considers protein trajectories simulated from five different biological behaviours as shown in Figure 4.1.

A single state model (SSM) does not have any changepoints, providing a control for the sensitivity of the model in the detection of changes. The rest of the models can exhibit an arbitrary number of changepoints either between two or more states given by a transition matrix (parameters described in Methods).

At these changepoints, at least one of D, α, or s will change. The multi-state model (MSM) considers time-dependent changes; the transient-confinement model (TCM) considers proteins entering and exiting compartments; the quenched-trap model (QTM) considers proteins in trapped (immobile) regions; and lastly, the dimerisation model (DIM) considers co-diffusion between interacting proteins. For QTM, $D = \alpha = s = 0$ when the protein is trapped. For the rest of the behaviours, we have $\alpha \in (0, 2)$, $D \in [10^{-12}, 10^{6}]$, and $s \in \{0, 1, 2, 3\}$. The units of D are pixels^2/frame (px^2/frame), with 1 $px^2/frame = 0.1$ μm$^2/s$ (taking the typical 100 nm per pixel for a super resolution microscope). Note there are five underlying biological behaviours but four states. These represent immobile (imm.) motion ($s = 0$ with $\alpha = 0$, only observable in QTM), confinement (conf.) ($s = 1$ with $0 < \alpha < 1.9$, only observable in TCM), free diffusion ($s = 2$ with $0 < \alpha < 1.9$, observable in all models), and directed (dir.) motion ($s = 3$ with $\alpha > 1.9$, observable in all models) [49]. The values of α and D do not necessarily have to change together. As an example, α may change when the protein exits a compartment but the value of D can remain unchanged. This is still a changepoint and a transition from $s = 1$ (conf.) to $s = 2$ (free).

In response to the AnDi 2024 challenge, we developed M3, a machine learning method for the pointwise inference of diffusion coefficients (D), anomalous exponents (α), protein state (s), and subsequent changepoint detection using the Python package ruptures [170]. M3 is based on the Long-Short Term Memory (LSTM) cell, an older architecture in the world of timeseries modelling as compared to transformers. As a brief overview, M3 extracts multivariate timeseries features

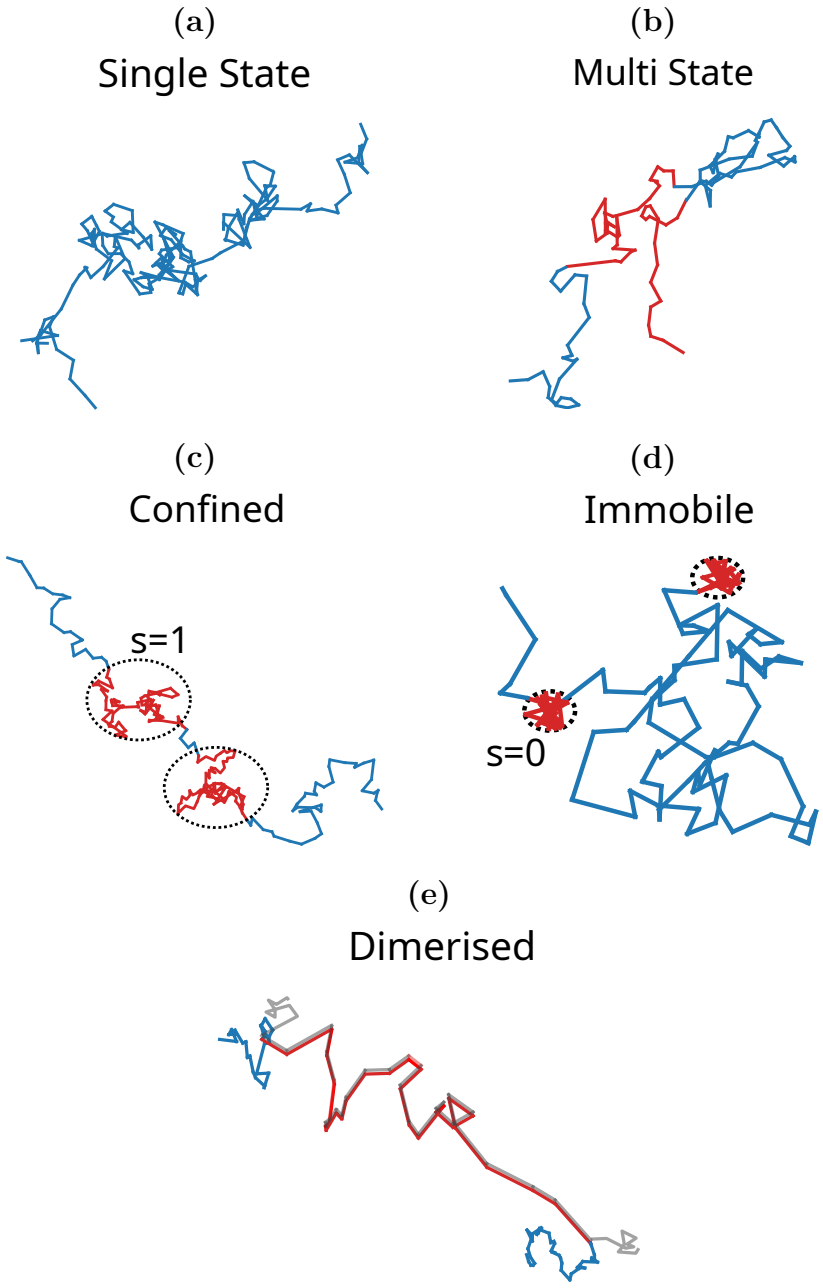


Figure 4.1: Biological behaviours simulated using the AnDi package with examples [50]: (a) Single state model (SSM) with no changepoints (e.g., RNAP freely diffusing in the cell) (b) Multi-state model (MSM) with time-dependent changes (e.g., RNAP switching between non-specific/specific binding) (c) Transient confinement model (TCM) for confined proteins (e.g., RNAP entering/exiting a cluster), (d) Quenched trap model (QTM) for trapped proteins ($D = \alpha = 0$ in the trap, e.g., RNAP entering/exiting transcription) (d) Dimerisation model (DIM) for co-diffusion between interacting (in grey) proteins (e.g., RNAP forming a holoenzyme). Red lines indicate changes in behaviour. When not specified, the protein state is either $s = 2$ or $s = 3$ (for $\alpha \geq 1.9$) depending on the value of α.

from the original coordinates of the protein trajectory and passes these into LSTM cells (with skip connections and dropout), followed by a feed-forward layer. The use of LSTMs with skip connections allows the model to pick up on long-range dependencies while keeping the architecture simple. The model returns D, α, and s for every time point of the protein trajectory. This then allows us to detect sudden transitions in behaviour.

4.2 Methods

4.2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of neural network designed for sequential data (e.g., timeseries). They maintain an internal memory state that is updated at each time step, allowing them to learn dependencies in time. Unlike feed-forward networks, RNNs process inputs sequentially and share parameters across different time steps as shown in Figure 4.2a. RNNs can be built in different ways and use a recurrence relation at each time step t, to update their hidden (memory) state h_t, based on the current input x_t and the previous hidden state h_{t-1}:

$$h_t = g(h_{t-1}, x_t; W) \tag{4.5}$$

where g is a function that maps the state $t-1$ to t and W are learnable weights [28].

RNNs use backpropagation through time (BPTT) and the use of shared weights allows these networks to handle variable length inputs. Figure 4.2b shows an example RNN cell for $g(x) = tanh(x)$.

However, in a sequence of RNN cells, the repeated multiplication of shared weights during backpropagation leads to two potential gradient related issues:

- Vanishing gradients: When weights are less than 1, repeated multiplication causes gradients to decay exponentially towards zero, making it difficult for the network to learn long-term dependencies.
- Exploding gradients: When weights are greater than 1, repeated multiplication results in exponentially growing gradients, causing training instability.

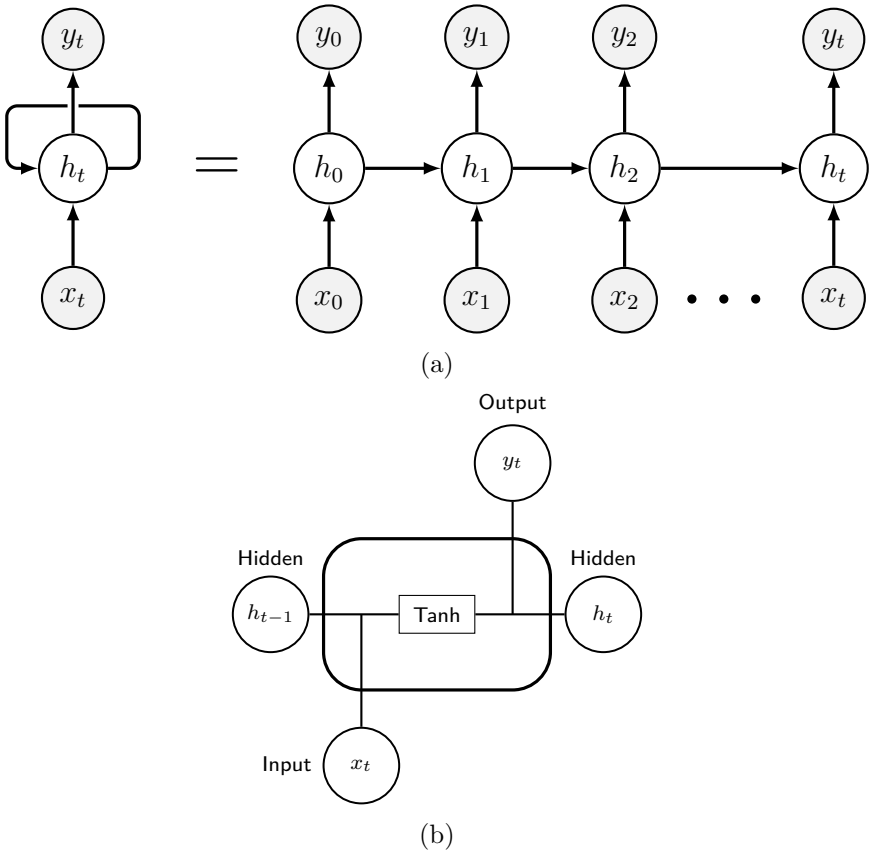


Figure 4.2: (a) RNN architecture showing both folded (left hand side) and unfolded (right hand side) representations. Input nodes (x_t), hidden states (h_t) from the RNN cell, and output nodes (y_t) are shown at each timestep. (b) Simple RNN cell with $g(x) = tanh(x)$.

Feed-forward networks avoid these gradient issues through their layer-wise parameter independence. While one layer might have small weights, subsequent layers can compensate with larger weights, providing better gradient flow. This is not the case for RNNs, where the same weights are repeatedly applied across time steps [28].

To circumvent gradient issues, the LSTM cell was introduced by Hochreiter and Schmidhuber [158]. This cell uses gating mechanisms and maintains a cell state c_t alongside the hidden state as shown on Figure 4.3. The LSTM cell uses three gates that each serve distinct functions in managing information flow. The input gate controls what new information is allowed to enter the cell state. The forget gate determines what information should be removed or discarded from the cell state. Finally, the output gate regulates what information from the cell

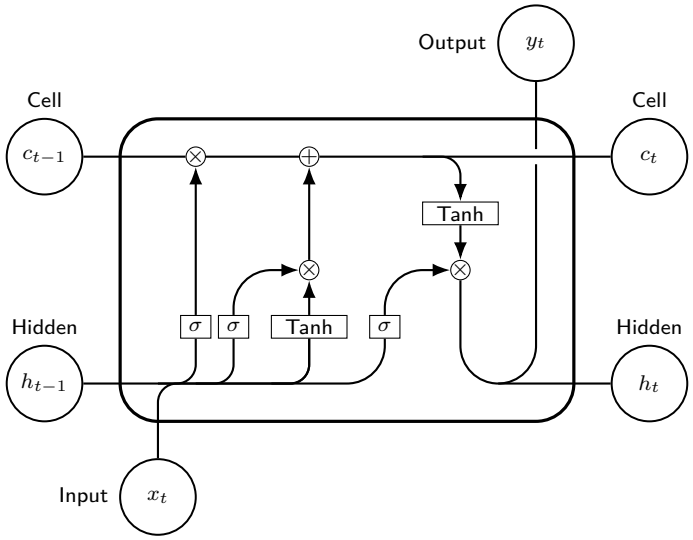


Figure 4.3: The LSTM cell

state is used to produce the output.

These are defined by the equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \text{ (forget gate)} \tag{4.6}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \text{ (input gate)} \tag{4.7}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \text{ (output gate)} \tag{4.8}$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \text{ (candidate cell state)} \tag{4.9}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \text{ (cell state update)} \tag{4.10}$$

$$h_t = o_t \odot \tanh(c_t) \text{ (hidden state)} \tag{4.11}$$

where $\odot$ denotes element-wise multiplication, σ is the sigmoid function, W are weights for the respective gates, b is the bias term, and square brackets denote concatenation.

The cell state provides a direct path for gradient flow, mitigating the vanishing gradient problem. The gating mechanisms allow the network to learn when to preserve or update information, enabling the capture of both long-term and short-term dependencies. This makes LSTMs particularly effective for tasks requiring long-range memory, such as protein trajectory analysis where temporal correlations

can span many time steps. Typically, LSTMs process the inputs from start to end. An extension of this is the bidirectional LSTM (biLSTM), which processes inputs both in a forward and backward direction using two LSTMs. The forward layer processes sequences from $t = 1$ to $t = T$ ($\overrightarrow{h_t}$). The backward layer processes sequences from $t = T$ to $t = 1$ ($\overleftarrow{h_t}$). The output of the biLSTM cell is the concatenation of the forward and backward layers at each timestep, $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. BiLSTMs have the advantage of extracting full context from the input, allowing the model to take advantage of previous ($t < t_{current}$) and subsequent ($t > t_{current}$) information. We make use of biLSTMs in our model architecture described next.

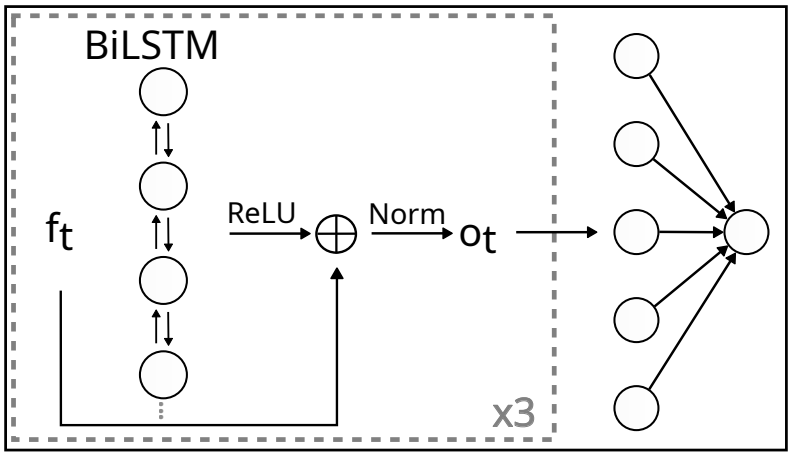
4.2.2 Model Architecture

As we are dealing with timeseries data, we implement a custom sequence-to-sequence model based on biLSTM cells. Notably, we have three stacked biLSTMs with Rectified Linear (ReLU) activation functions, skip connections, and layer normalisations (see Figure 4.4). The first two biLSTM cells have two hidden layers and the last biLSTM has a single layer, with all three cells retuning a vector of length 128 for every time step. The output of the last biLSTM is passed onto a fully connected layer, mapping the result into the desired shape. The input to the model is ten features (shown shortly), in the form of timeseries, extracted from the initial coordinates $x(t)$ and $y(t)$. The model returns a sequence of $D(t)$, $\alpha(t)$ or $s(t)$ for every time step.

For each of the three output variables, we duplicate the model and retrain with a different loss function. It is possible to combine the models into a single model with three output heads, but this is difficult as it combines classification and regression [151]. Since protein state prediction (s) is a classification task, we add a LogSoftmax layer just before the output of the model architecture. The model has a total of roughly 513,000 trainable parameters.

The model architecture resulted from starting with a single biLSTM cell and progressively incrementing the number of cells and layers until no improvement in validation loss was seen. The same procedure was carried out later for other

(a)



(b)

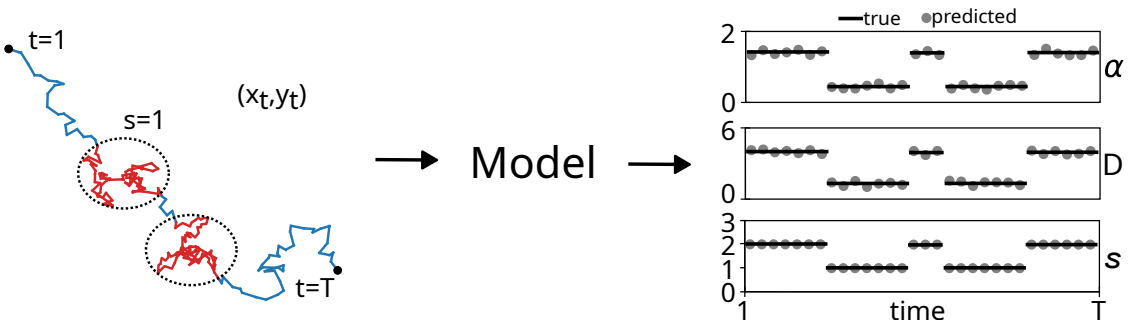


Figure 4.4: Model architecture and pipeline. (a) M3 Model: BiLSTM architecture with skip connections. The model takes in a protein trajectory and extracts features f_t for every time step. The features are processed through a biLSTM layer, a ReLU layer, concatenated with the input, and subsequently layer normalised. This is performed three times with o_t now serving as the input ($f_t = o_t$). The model performs pointwise inference, and is retrained for every target variable. (b) Complete pipeline showing how protein coordinates (x, y) are fed into the model producing predictions at each time step for α, D, and s. Note that the D values are in the $\log(D + 1)$ space, such that the target variable is constrained between 0 and 6, providing a normalised range for model training.

parameters in the model such as the dropout rate, number of fully connected layers, learning rate, and regularisation parameter. To streamline this process, we use Optuna [171], an open-source Python library for Bayesian hyperparameter search. Optuna searches the parameter space, exploring hyperparameter configurations leading to the best validation loss. This is done using a coarse grid of 100 runs, with each run exploring a different set of parameters in a Bayesian fashion. Then, we repeat with another 100 runs in a subset of the initial space. An example of

this is shown later for the case of the learning rate and weight decay.

4.2.3 Feature Selection

For the problem of predicting pointwise D, α, and s, we start by finding features of interest from an extensive literature search, including past AnDi challenge submissions [47, 49, 150–169, 172–180]. We add generic timeseries features (e.g., running means and standard deviations), resulting in a list of roughly 80 features. Using Pearson's correlation coefficient (PCC), we remove highly collinear features ($|\text{PCC}| > 0.95$). Collinear features only add more potential noise to the model, as no extra information is gained. This removed roughly half the features, leaving 40 features. With the remaining features, we perform greedy forward feature selection. This was done using a baseline model consisting of three biLSTM layers at the time. Starting with the $z-$normalised (zero mean and unit variance) $x(t)$ and $y(t)$, we greedily add features to this list. If the added feature does not improve the model, we remove it. This process required domain expertise which ensured important features (e.g., turning angle) were not discarded. Greedy feature selection is necessary as computing the global optimum of 40 features would otherwise require 2^{40} iterations. By symmetry, a feature not improving the model in x was removed alongside its y counterpart and vice versa (e.g., if σ_x was not improving the model then σ_x and σ_y were both removed).

After this process, we were coincidentally left with ten features, each being a timeseries itself, derived from the original coordinates. These are:

$z-$normalised (zero mean and unit variance) coordinates:

$$\hat{x}(t) = \frac{x(t) - \bar{x}}{\sigma_x} \quad \text{and} \quad \hat{y}(t) = \frac{y(t) - \bar{y}}{\sigma_y} \tag{4.12}$$

where σ_x and σ_y are the standard deviations, and $\bar{x}$ and $\bar{y}$ are the respective means of x and y.

Displacement from origin:

$$d(t) = \sqrt{(x(t) - x(0))^2 + (y(t) - y(0))^2} \tag{4.13}$$

$z-$normalised step-size:

$$\text{step}(t) = \sqrt{(\Delta x(t))^2 + (\Delta y(t))^2} \tag{4.14}$$

$$\hat{\text{step}}(t) = \frac{\text{step}(t) - \overline{\text{step}}}{\sigma_{\text{step}}} \tag{4.15}$$

where Δ represents the difference between consecutive elements

Turning angles (between coordinate segments):

$$\vartheta(t) \tag{4.16}$$

Path metrics:

$$\text{str}(t) = \frac{\sqrt{(\hat{x}(t) - \hat{x}(0))^2 + (\hat{y}(t) - \hat{y}(0))^2}}{\sum_{j=1}^{t} \sqrt{(\Delta\hat{x}(j))^2 + (\Delta\hat{y}(j))^2}} \tag{4.17}$$

$$\text{str}_2(t) = \frac{(\hat{x}(t) - \hat{x}(0))^2 + (\hat{y}(t) - \hat{y}(0))^2}{\sum_{j=1}^{t} (\Delta\hat{x}(j))^2 + (\Delta\hat{y}(j))^2} \tag{4.18}$$

Log derivatives:

$$F(k) = \log|\Delta k(t)| \quad \text{for } k \in \{\hat{x}, \hat{y}, \hat{\text{step}}\} \tag{4.19}$$

The features $\text{str}(t)$ and $\text{str}_2(t)$ are measures of trajectory straightness, where a value of 1 represents perfectly directed motion (straight path) and 0 represents a path which returned to its starting position. These features are combined into the input vector:

$$[\hat{x}, \hat{y}, d, \hat{\text{step}}, \vartheta, \text{str}, \text{str}_2, F_{\hat{x}}, F_{\hat{y}}, F_{\hat{\text{step}}}] \tag{4.20}$$

These ten features are calculated from the initial $x(t)$ and $y(t)$ coordinates and return ten timeseries of length equal to the protein trajectory. Any features shorter than the total number of time steps are post-padded with zeros. The final input to the model is of shape $(B, 10, T)$ where T is the maximum number of time steps in the batch and B is the batch size.

4.2.4 Generating Training Data

Training data was simulated using the AnDi Python package [47, 49, 50]. This samples stochastic trajectories based on a set of input variables for each biological behaviour. We simulate protein trajectories for each of the behaviours shown in Figure 4.1. Changes in protein behaviour are determined by transition matrices. Note that the AnDi package simulates trajectories within a large Field-of-View (FOV). From this FOV, a smaller square is placed randomly. Trajectories that lie in this square and have a length between 20 and 200 time steps (inclusive, with 1 time step equal to 0.1 s) are then considered as valid trajectories. Additionally, all models have two states except for the single state model (which has a single state) and the multi-state model, which must have at least two states. Although there are two states for some models, the protein is able to transition multiple times between these states, leading to an arbitrary number of changepoints.

Table 4.1 summaries all parameters for each behaviour and the values used to generate the data. Simulated data is created in two dimensions.

When simulating data, we apply further changes to the parameters to obtain data representative of all possible cases. These are outlined below:

General Parameter Setting

Every simulated FOV contains roughly 100 protein trajectories. Some trajectories are disregarded as they may not meet length requirements. For every model in Table 4.1, we simulate 400,000 tracks, for a total of 2 million tracks. This generates 4000 unique combinations of parameters for each model.

Biased Setting

During training, we noticed the directed ($s = 3$) state is under-represented (as $\alpha \geq 1.9$ for $s = 3$ which is less likely during random sampling). To counter this, we performed the same simulations again as in Table 4.1 with α for states being constrained to $[1.9, 2)$. This generated another 2 million tracks, most of these involving $s = 3$ or transitions into/out-of this state (e.g. $s = 1$ to $s = 3$).

Table 4.1: General parameters used for simulating protein trajectories across different biological models. For all models, $\alpha \in [0, 2)$ and $D \in [10^{-12}, 10^6]$ (px^2/frame). Values are randomly and uniformly drawn from the given range.

Model	**Parameters**	**Sampling Range**
Single-State (SSM)	α	$(0, 2)$
	D	$[10^{-12}, 10^6]$
Multi-State (MSM)	Number of states (N_s)	$\{2, 3\}$
	Transition matrix	$[0, 1]^{N_s \times N_s}$
	α_i for each state	$(0, 2)$
	D_i for each state	$[10^{-12}, 10^6]$
Quenched-Trap (QTM)	Unbinding probability	$[0, 0.1]$
	Binding probability	Fixed at 1
	Trap radius	$[0.5, 2]$
	Number of traps	$[100, 300]$
	Free α, D	$(0, 2), [10^{-12}, 10^6]$
	Trapped α, D	Fixed at $0, 0$
Dimerization (DIM)	Unbinding probability	$[0, 0.1]$
	Binding probability	Fixed at 1
	Interaction radius	$[0.5, 5]$
	Free α, D	$(0, 2), [10^{-12}, 10^6]$
	Dimerised α, D	$(0, 2), [10^{-12}, 10^6]$
Transient-Confinement (TCM)	Transition probability	$[0, 0.3]$
	Number of compartments	$[30, 50]$
	Compartment radius	$[5, 10]$
	Free α, D	$(0, 2), [10^{-12}, 10^6]$
	Confined α, D	$(0, 2), [10^{-12}, 10^6]$

Fixed Setting

Lastly, we generate tracks in which the value of D is fixed between changepoints while α varies and vice versa. As an example, when exiting a compartment, D will be the same for the confined and free state, but α will change, allowing the model to see examples where one variable stays constant across a changepoint. Of course, this is only relevant for models allowing for one of the variables to be fixed (MSM, DIM, TCM). This generated a total of 1.2 million tracks. The parameters are the same as in Table 4.1, except once a variable such as D is sampled, it is the same for both states.

Overall, we simulate a total of 5.2 million tracks from five biological behaviours

spanning a range of values (with four states). The data is split randomly using a train/test/val ratio of 60:20:20 such that there are 3.12 million tracks in the training data.

4.2.5 Augmentations

During training, augmentations are added to improve model learning. The coordinates $x(t)$ and $y(t)$ are augmented using the following transformations:

Gaussian Noise Addition

$$y'(t) = y(t) + \epsilon(t), \quad x'(t) = x(t) + \epsilon(t), \quad \epsilon(t) \sim \mathcal{N}(0, 0.1) \tag{4.21}$$

Rotations

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = M_\vartheta \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \quad M_\vartheta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{4.22}$$

where M_ϑ is the standard rotation matrix with angle $\vartheta \in [0, 2\pi]$.

Mirror Transformations

$$x'(t) = \pm x(t), \quad y'(t) = \pm y(t) \tag{4.23}$$

implementing vertical and horizontal flips along both axes.

Temporal Truncations

$$x(t') = x(t_1 : t_2), \quad y(t') = y(t_1 : t_2) \tag{4.24}$$

where times t_1 and t_2 are randomly selected such that:

$$0 \leq t_1 \leq T - t_{\min} \tag{4.25}$$

$$t_1 + t_{\min} \leq t_2 \leq T \tag{4.26}$$

with minimum segment length $t_{\min} = 20$.

Each augmentation is applied independently with 30% probability of occurring and only to the training data. All subsequent feature are calculated from the augmented coordinates.

4.2.6 Training

We train the model using a batch size of 32, for a maximum of 30 epochs. During training, 10% dropout is applied to the biLSTM cells. Training typically converges before the maximum number of epochs is reached, and the model with the smallest validation loss is saved. A scheduler is implemented, decreasing the learning rate by a factor of ten every five epochs, if no improvement in validation loss is seen. We use the Adam optimiser with a learning rate of 1×10^{-3} and a weight decay (λ) of 2×10^{-6} to prevent over-fitting [30]. These parameters were determined using the Optuna package [171] (see Figure 4.5).

The loss functions for a single output timeseries are given by:

$$\mathcal{L}_\alpha = \frac{1}{T}\sum_{i=1}^{T} |\alpha_{p,i} - \alpha_{g,i}| \tag{4.27}$$

$$\mathcal{L}_D = \frac{1}{T}\sum_{i=1}^{T} |\log(D_{p,i}+1) - \log(D_{g,i}+1)| \tag{4.28}$$

$$\mathcal{L}_s = \frac{1}{T}\sum_{i=1}^{T} -w_{y_i}\log(p_{y_i}) \tag{4.29}$$

where T is the total length of the protein trajectory ($T_{\max} = 200$) and the subscripts p and g denote predicted or ground truth variables. The losses are normalized by trajectory length to account for varying lengths. State classification is imbalanced as the number of $s = 2$ labels outweighs the other states. Hence, we use class weights w_{y_i} for each class, calculated as inverse frequency using the training dataset.

The backpropagated loss is given by the average over the batch size (B) with L2-regularisation:

$$\bar{\mathcal{L}} = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_i + \lambda \sum_{w\in\mathcal{W}} w^2 \tag{4.30}$$

The targets α and s are constrained to suitable ranges ($[0, 2)$ and $\{0, 1, 2, 3\}$ respectively). However, since D varies between 10^{-12} and 10^{6}, the model for D

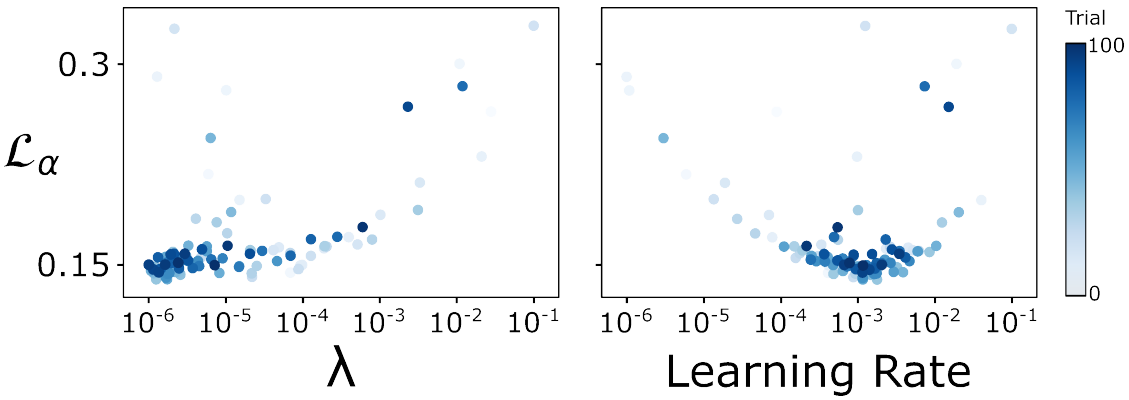


Figure 4.5: Example hyperparameter tuning using the validation dataset for $\mathcal{L}_\alpha$ as given in the text. λ is the L2−regularisation parameter (weight decay).

is trained on the target variable $\log_{10}(D+1)$. This normalises D to be in the range $[0, 6]$ for every time point.

Training was performed on a single NVIDIA RTX A5000 GPU, taking a total of 10 hrs for a single output variable. Since we repeat three times, a total of 30 hrs was spent training. This can be sped up if multiple GPUs are available. We had access to two GPUs, such that training took a total of 20 hrs.

4.2.7 Post-Processing

After model prediction, we smooth $D(t)$ and $\alpha(t)$ by first replacing sections in which the output series varies by less than 0.01 with their mean. This removes any noisy variations in the output sequence and considers changepoints that are above this threshold. After initial smoothing, we apply a median filter with a window size of three. Median filters preserve edges well, providing better downstream changepoint detection. As part of the challenge constraints, the minimum time spent in a state must be at least three frames; hence, the state series $s(t)$ is smoothed such that any state segment lasting fewer than three consecutive frames is deemed invalid and replaced with the value of its preceding state.

4.2.8 Changepoint Detection

Changepoint detection was done using the ruptures library in Python [170]. Ruptures implements offline detection methods using a variety of cost functions. As we do not

know the number of changes beforehand, an additional penalty parameter is applied. In order to obtain an ideal penalty value, we performed a quick grid search over a sensible range using the training set. Specifically, we use the Pruned Exact Linear Time (PELT) algorithm with a Least Squares Deviation (L2) cost function and a penalty of 0.3 [181]. This identifies changepoints in a timeseries by segmenting data to minimise the sum of squared deviations within segments, plus a penalty term dependent on the number of segments. We set the minimum distance of changepoints to be at least three time steps and we min-max normalise the predicted timeseries before changepoint detection. This avoids the need to constantly tune the penalty value. For the state series, changepoint detection is easier as the label changes.

Given changepoints from all three outputs, we combined them by adding changepoints from α and s to the D changepoints and removing duplicates within a window size of 5. This centres changepoints around the D changepoints, where the model generally performed better on. We cannot only take the D changepoints, as it could be constant across a changepoint while α may not be, leading to missing changepoints.

4.2.9 Evaluation Metrics

For α and D, we evaluate the model over all test data using the mean absolute error per time step, given by:

$$\frac{1}{N}\sum_{j=1}^{N}\frac{1}{T_j}\sum_{i=1}^{T_j}|p_{i,j}-g_{i,j}| \tag{4.31}$$

where N is the number of tracks, T_j the number of time steps for the jth track, $p_{i,j}$ is the ith predicted value of α or $log(D+1)$ for the jth track and $g_{i,j}$ the ground truth for the corresponding predicted point. For α, this is simply the mean absolute error (MAE) per time step, whereas for D this is the mean absolute log error (MALE) for every time step.

For the state variable, we evaluate the confusion matrix (row-wise normalised), as the class distributions are imbalanced, and the state loss ($\mathcal{L}_s$) averaged over all trajectories.

Evaluation of changepoint detection is done using the Jaccard index and the root mean square error. The Jaccard index measures the level of overlap between the predicted changepoints and the ground truth changepoints with an allowable error of five timesteps, in-line with the challenge constraints. For two sets A and B, the Jaccard index is defined by:

$$J = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FN + FP} \tag{4.32}$$

where TP, FN, FP are the number of true positives, false negatives, and false positives respectively.

The root mean square error (RMSE) is given by [49]:

$$RMSE = \sqrt{\frac{1}{N_{CP}} \sum_{paired\ CPs} (t_{g,i} - t_{p,j})^2} \tag{4.33}$$

where N_{CP} is the number of changepoints in the trajectory, $t_{g,i}$ is the time point of the ground truth changepoint, and $t_{p,i}$ is the time point of the predicted changepoint for some pair of changepoints (i, j). The RMSE is only calculated for paired segments (< 5 frames of the ground truth).

Matching predicted and ground truth changepoints is performed using the Hungarian algorithm, which solves the assignment problem. The algorithm minimises the total cost of matching predicted changepoints to ground truth changepoints. This is handled by the AnDi Python package [47, 49, 50].

4.3 Results

Once each model is trained, we perform inference on unseen trajectories from the test data set. Training curves are shown in Figure 4.6. Inference is performed once for each model, returning $\alpha(t)$, $D(t)$, and $s(t)$ at every time step of the trajectory. Details of the dataset and simulation are described in Table 4.1. The model, M3, does not directly return changepoints and all ground truth timeseries are piecewise constant. **Note:** From hereof, D is reported in all figures and tables in

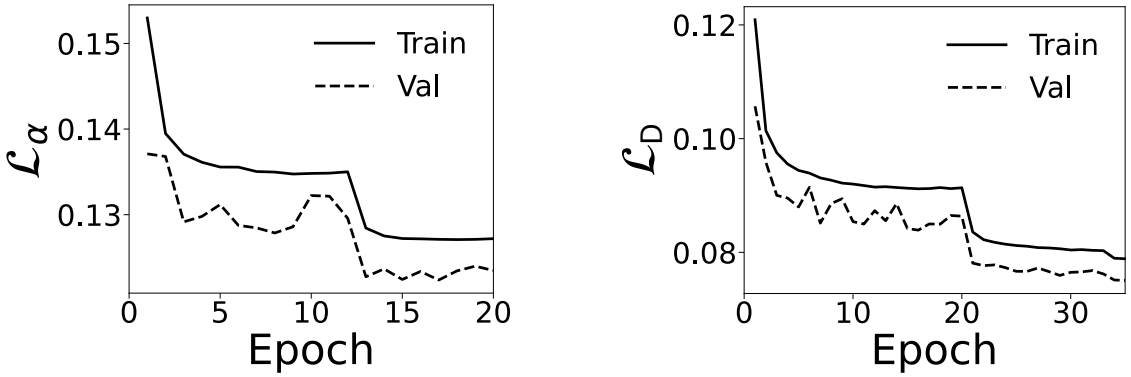


Figure 4.6: Training curves for α and D models. The scheduler kicks in just over halfway, further pushing the losses down.

$log_{10}(D+1)$ space. This is also evident on the scale. We redefine $D \to log_{10}(D+1)$ for convenience as this is the target variable for the D model.

4.3.1 Pointwise Inference of α, D, and s

We first visualise model performance by inspecting model predictions compared to the ground truth for different trajectories from the test set. Figure 4.7 shows some example tracks from different biological behaviours and the resulting model output after post-processing. Generally, it was observed that D predictions were closer to their ground truth than α predictions, and immobile states ($s = 0$) were easily detected by the model, most likely due to their distinct difference from the other states. The ability of the model to perform better on D than α arises from the inherent difficulty of the problem. Varying D directly affects the step size (hence trajectory) of the protein in a noticeable way. However, α can be seen as a measure of "directionality", making it a more difficult task for the model to learn.

Having observed some model predictions, we verify our observations by plotting the mean absolute errors as measures of accuracy for α and D averaged over the test set. This gives a MAE of 0.16 for α and a MALE of 0.14 for D. We also evaluate model predictions on trajectories simulated solely from the single state model for a wide range of D and α values. Heatmaps of predictions against ground truth are shown on Figure 4.8, and allowed for model evaluation while holding the other variable fixed. At an ensemble level, the predictions from both continuous

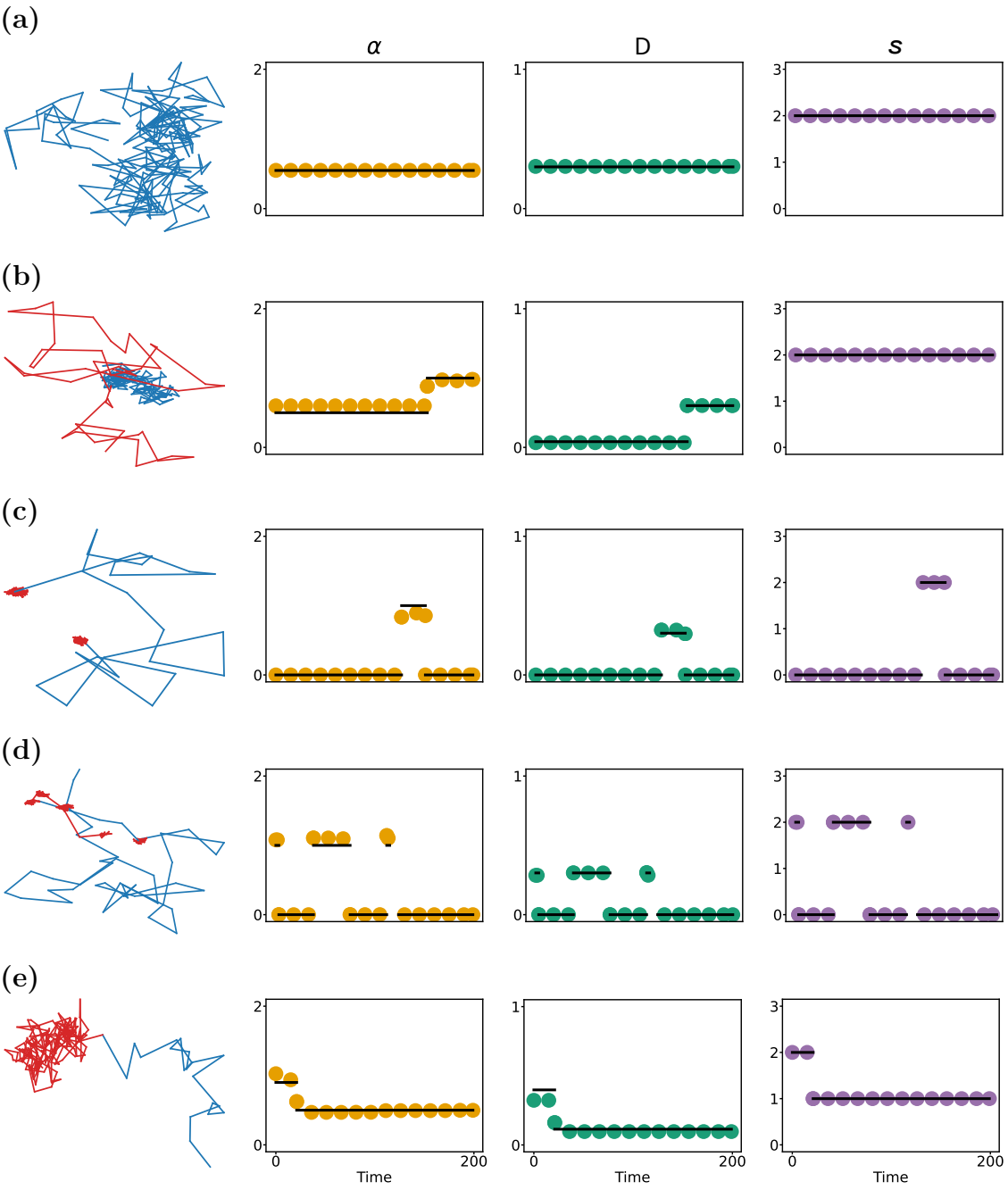


Figure 4.7: Model Predictions for test tracks. For clarity, we plot a subset of point predictions (filled circles) for ground truth segments (black lines). (a) A single state track (SSM). The protein undergoes no changepoints. (b) A multi-state track (MSM). The model correctly predicts the diffusive properties at the changepoint. (c) An immobile track (QTM). The protein is immobile ($s = 0$), escapes briefly ($s = 2$) and enters another trap ($s = 0$). The model is able to pick up this brief transition. (d) An immobile track with more changepoints and shorter segments. (e) A confined track (TCM). The protein diffuses freely ($s = 2$) until it encounters a compartment ($s = 1$).

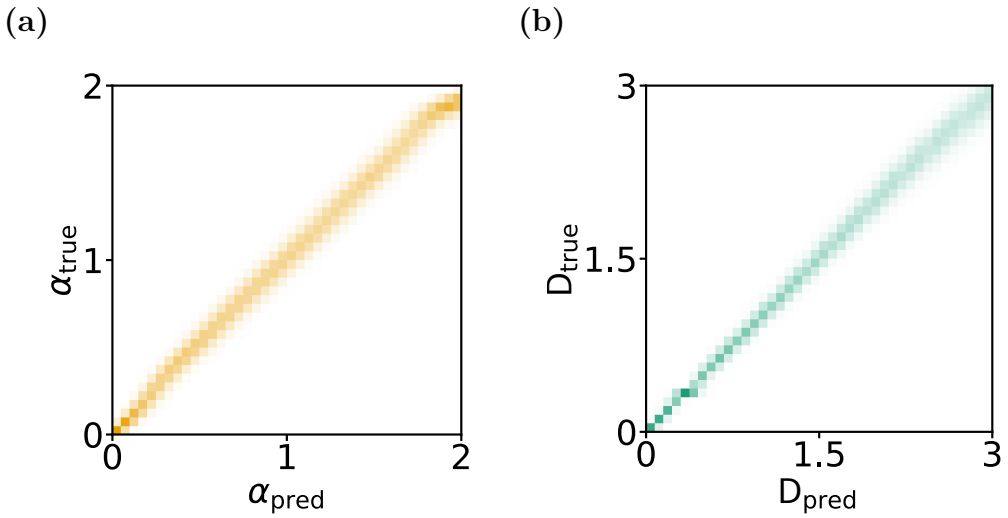


Figure 4.8: Heatmap of predictions against ground truth values. (a) Trajectories were generated using the single state model and D was fixed at one while α varies. For every ground truth bin, 1000 single state protein predictions were averaged. (b) The same plot as (a), except with α fixed at one while D changes.

variables lie close to the ground truth with the exception that D ranges from 0 to 3 in log space. The restricted range of D reflects the fact that the protein leaves the FOV during the simulation for diffusion coefficients larger than this. However, the range includes a wide range of diffusion coefficients ($0 - 100\ \mu m^2/s$) beyond most experimentally observed values [46].

Due to the nature of the challenge and simulation, the test set is imbalanced, with the free state ($s = 2$) observed more frequently. We implement class weights (inverse frequency) during training and evaluate the confusion matrix. This is shown on Figure 4.9, giving an average class accuracy of 92%. The confusion matrix verifies the ability of the model to detect bound states ($s = 0$) very well compared to other states. It is interesting to note the interplay between the states $s = 1, 2$, and 3. For certain trajectories, these three states can be confused for another, not necessarily due to an inherent fault in the model setup, but due to the difficulty of the classification. As an example, a slightly larger compartment ($s = 1$) can be detected as free diffusion ($s = 2$) and vice versa. As shown shortly, these effects can be amplified for shorter tracks, where the model prediction error increases due to less information being available to it. Another potential factor affecting this

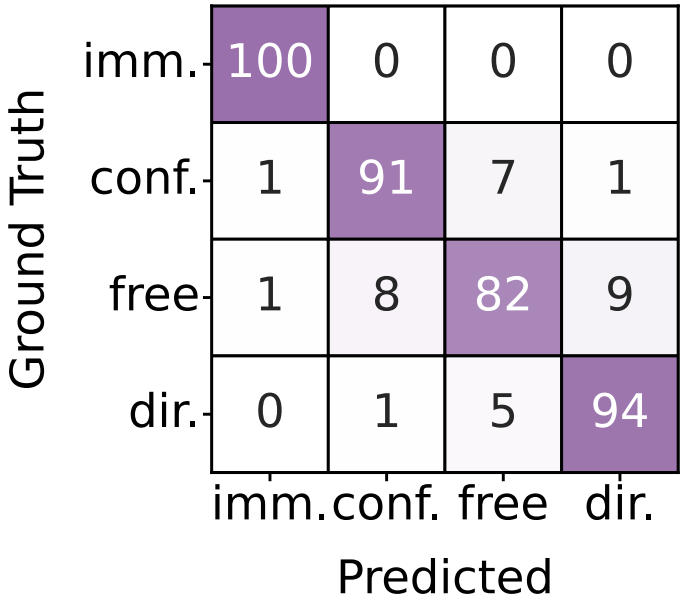


Figure 4.9: Confusion matrix, normalised row wise, showing predicted states against ground truth over the entire test set. The labels correspond to: immobile (imm., $s = 0$), confinement (conf., $s = 1$), free ($s = 2$), directed (dir., $s = 3$). Average class accuracy is 92%.

stems from the use of class weights, which favours minority classes, decreasing the accuracy of the model on majority classes [28].

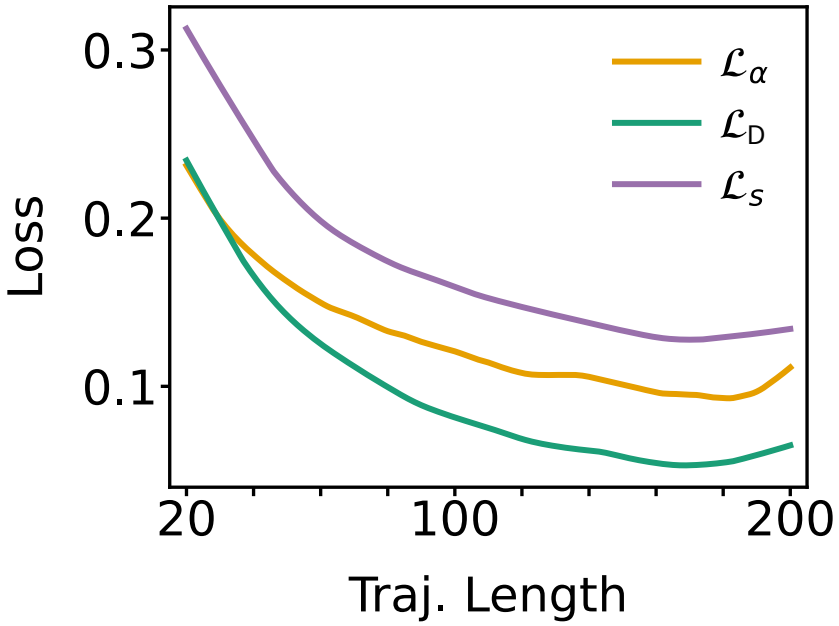
4.3.2 Dependence on Trajectory Length

We further evaluate the model's predictions over the trajectory length, in the range 20 to 200 time steps (inclusive). For this, we plot the model loss and average accuracy per time point for varying lengths across the test set. Consistent with previous work, we see a drop in errors as the trajectory length increases (see Figure 4.10) [47]. As more information is available to the model, better predictions are obtained. Shorter trajectories can originate from a range of possible parameters, leading to higher errors in α, D, and s. Figure 4.10 shows plateaus in both loss and class accuracy for lengths greater than 150.

4.3.3 Changepoints

M3 does not directly return changepoints. Detecting changepoints in the output timeseries is performed using the PELT algorithm [170, 181]. For the state timeseries,

(a)



(b)

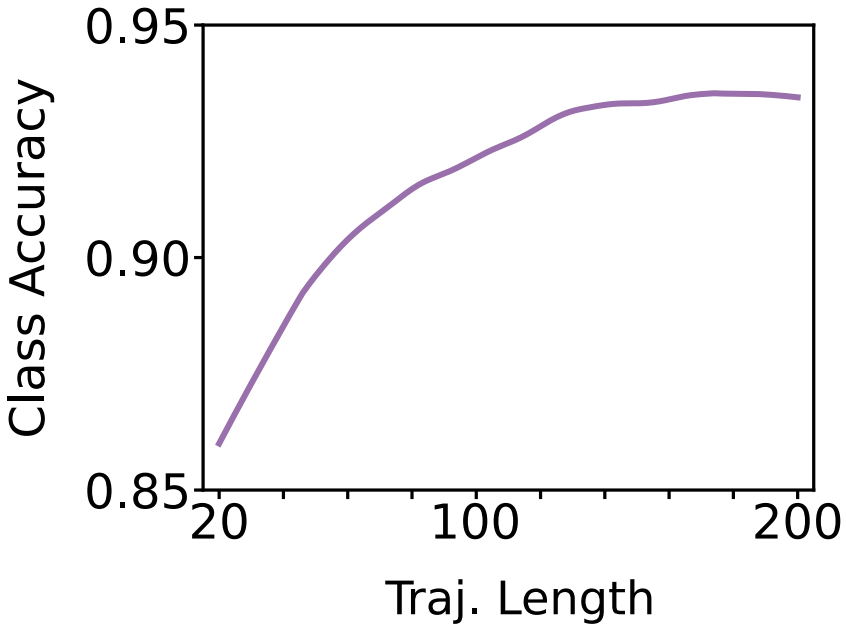


Figure 4.10: Model predictions improve for longer trajectories as expected. (a) The loss for all three variables as trajectory length increases, averaged over the test set. (b) Average class accuracy for trajectory lengths over the test set.

finding changepoints is easier as the value of the label changes. Figure 4.11 shows an example model prediction for D and the detected changepoints.

The model is able to pick up all changes in the track for sufficiently long enough

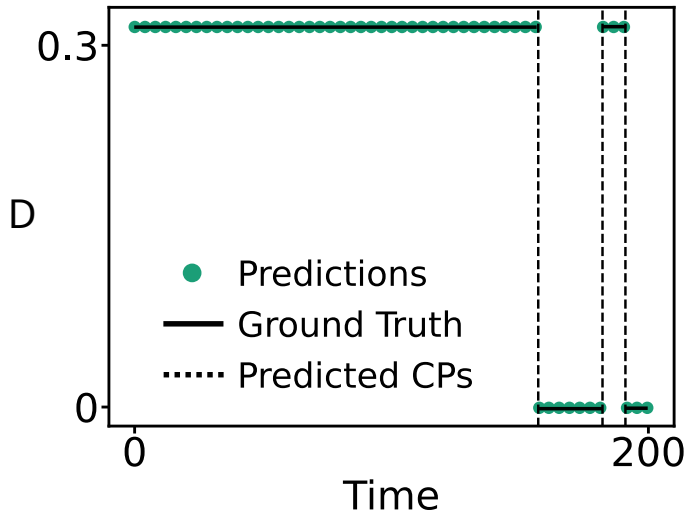


Figure 4.11: Changepoint detection on an example model prediction for D using the PELT algorithm. We are able to pick up all changepoints for sufficiently long tracks.

timeseries. The penalty parameter in ruptures is tuneable, altering the threshold for detection of changes. To avoid constant tuning, all timeseries are min-max scaled between 0 and 1 before changepoint detection, resulting in the same penalty for both D and α. These changepoints are then used for non min-maxed timeseries.

As shown on Figure 4.12, with an increasing number of changepoints (N_{CP}) in the protein trajectory, the MAE for α increases while the MALE for D remains roughly constant. This aligns with the asymptotic behaviour of α, which leads to greater prediction errors for a higher number of changepoints, in which the average segment is shorter (inversely related) [47]. D can be estimated to better accuracy from shorter segments (less information) in a trajectory.

Changepoint detection can be performed on the α, D and s predictions. We first perform this individually, then combine the sets of changepoints into a single set with repeats removed. This merges the changepoints from all the different timeseries and ensures the model is able to pick up on changes from any of the output variables. A metric we use to measure the quality of changepoint detection is the Jaccard index. The Jaccard index quantifies the overlap between two sets, with a value of 1 indicating a perfect match (see Section 4.2.9). In Figure 4.13, we evaluate the Jaccard index for changepoint detection using combinations of α, D, and s.

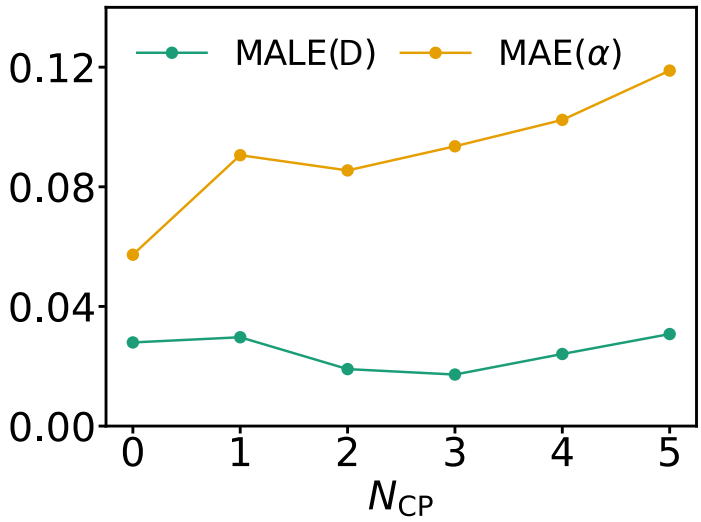


Figure 4.12: The MAE and MALE over the test set for varying number of changepoints (N_{CP}) in the protein trajectory.

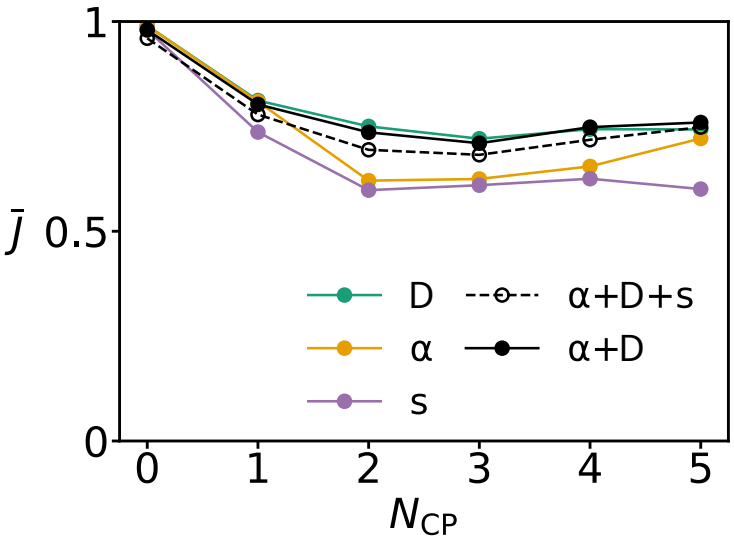


Figure 4.13: Average Jaccard index ($\bar{J}$) for different number of changepoints (N_{CP}) in the trajectories. For each N_{CP}, we average 100 randomly sampled protein trajectories of length 200 from the test set. For every line we use the given set of variables to determine changepoints in test tracks. This plot informs us of combinations of output variables that lead to best downstream changepoint detection.

As expected, the α timeseries alone returns lower Jaccard indices across the range of changepoints as compared to changepoints from D predictions. The state timeseries alone is not enough to detect changepoints, since in multi-state trajectories, the state label remains the same although a changepoint can occur. However, using only D timeseries for changepoints may mean we lose out on

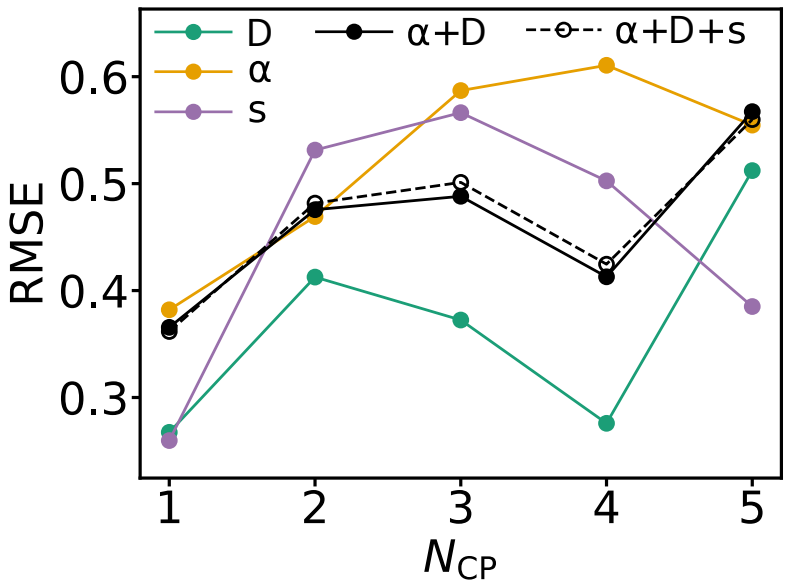


Figure 4.14: RMSE for changepoint detection performed using different combinations of α, D and s over the test set. For every N_{CP} (number of changepoints in trajectory), the RMSE value represents the average of 100 randomly sampled protein trajectories of length 200 from the test set.

changepoints in which D remains constant across the changepoint whereas α may vary. Thus, we settle for using $\alpha + D$ changepoints, while removing duplicates, for producing a set of changepoints for any given trajectory (see Figure 4.13). This may mean we miss changepoints in which both of these variables do not change, while s does, but these trajectories are less likely ($\sim 0.04\%$ of test dataset). Typically, a change in biological behaviour is accompanied by a change in diffusion coefficient and/or anomalous exponent.

We also evaluate the average RMSE values on Figure 4.14 for varying changepoints using the same sets of variables. Again, we see slightly better performance (lower RMSE) across the test set using $\alpha + D$ compared to $\alpha + D + s$. Note that RMSE values are only calculated for paired changepoints. For a higher number of changepoints, any unpaired predicted changepoints are not included in the calculation, such that lower RMSE values do not imply better changepoint detection. As can be seen in Figure 4.13, the Jaccard index decreases for a higher number of changepoints in the trajectory.

We also evaluate the model's ability to detect changepoints at different positions

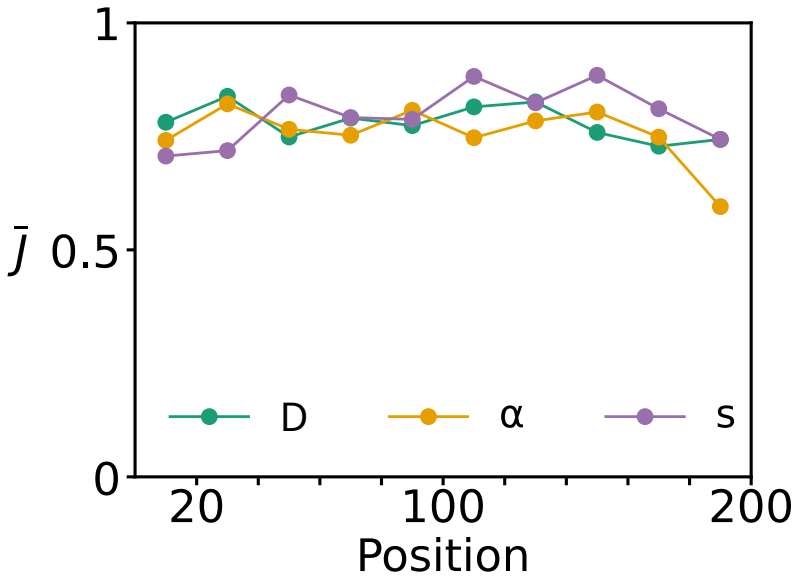


Figure 4.15: Average Jaccard index ($\bar{J}$) for trajectories of length 200 from the test set with a single changepoint at different positions. Each point represents the average over 100 randomly selected trajectories and bins 20 length values together.

in the predicted timeseries. Figure 4.15 shows the Jaccard index for changepoint detection from α, D, and s predicted timeseries for trajectories with a single changepoint at different positions (across the entire test set). The model's bidirectional nature allows for consistent changepoint detection across various changepoint positions in the trajectories.

Changepoint detection is clearly dependent on the difference in α or D at a changepoint. Little or no difference in α or D at a changepoint of a protein trajectory leads to no changepoint being detected. The model may also return a single value for all time steps of the trajectory, although there may have been a changepoint with small change in D (see later).

To evaluate performance as a function of the difference at a changepoint, we simulated two new datasets in which there exists a single changepoint halfway through the trajectory (multi-state model), with all trajectories of length 200 time steps (see Table 4.2). The first dataset fixes α at one for all trajectories, allowing D to change between the two segments, while the second dataset fixes D at one for all trajectories and allows α to vary between the two segments. These datasets allow for robust evaluation of changepoint detection for varying $\Delta\alpha$ and ΔD between the

Table 4.2: Parameters used for simulating protein trajectories for changepoint detection analysis. All trajectories have length $T = 200$ with a single changepoint at $t = 100$. Each dataset represents a controlled scenario where either α or D is fixed between the changepoint and the other variable changes. The protein trajectories have two segments.

Model	Parameters	Sampling Range
Multi-State (MSM)	α_1, α_2	Fixed at $1, 1$
	D_1, D_2	$(0, 6], (0, 6]$
Multi-State (MSM)	α_1, α_2	$(0, 2), (0, 2)$
	D_1, D_2	Fixed at $1, 1$

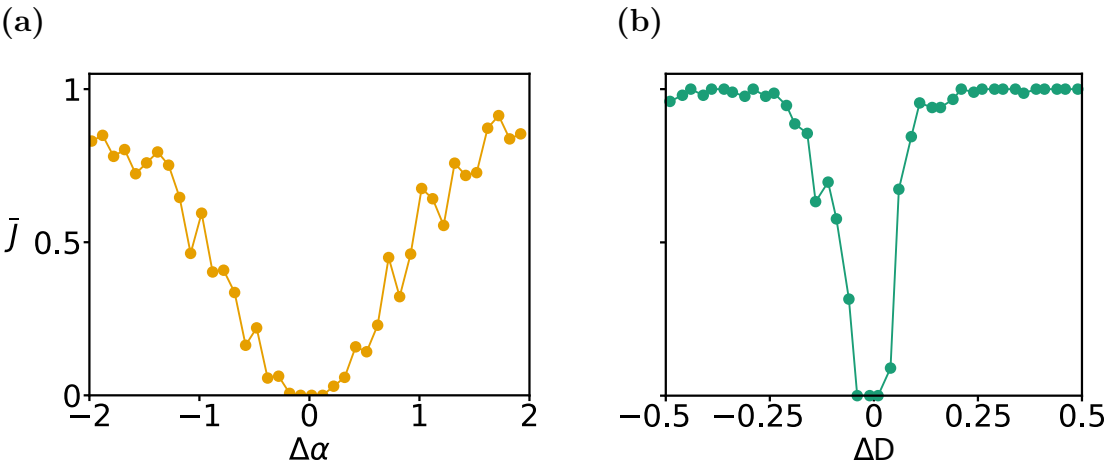


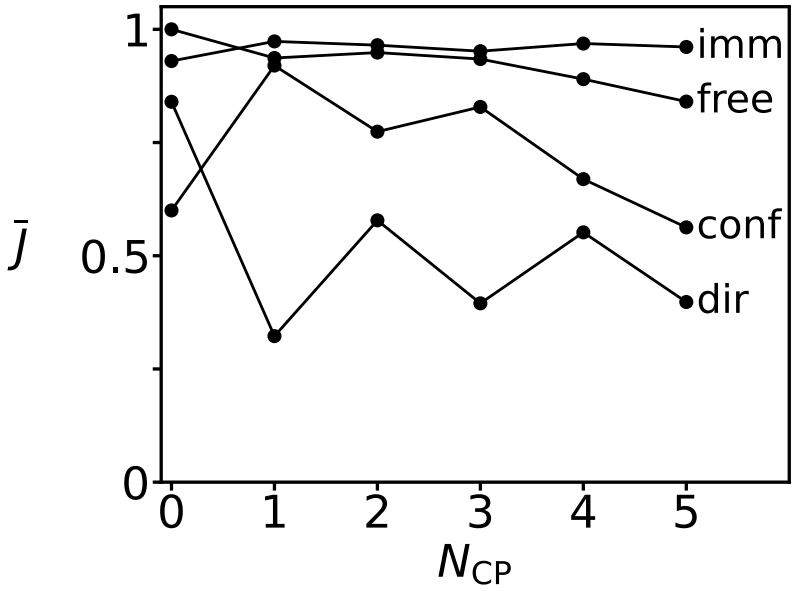
Figure 4.16: (a) Average Jaccard index ($\bar{J}$) for trajectories with a single changepoint (thus having two segments, multi-state model), with $\Delta\alpha$ being the difference in α between the two segments. D is fixed at one for both segments. Each point is the average over roughly 50 trajectories of length 200. (b) The same plot as (a), except for ΔD, with α fixed at one between segments.

segments. We show the Jaccard index for $\Delta\alpha$ between segments on Figure 4.16a. The model performs well for large differences in $\Delta\alpha$ between the segments, with the Jaccard index drastically reducing for smaller differences. As expected, we obtain a Jaccard value of zero when the two segments have identical α values, as the model did not pick up any changepoints. Figure 4.16b repeats the same analysis for ΔD. This shows high values of Jaccard index for a much larger range of values, again reducing for ΔD close to zero. As mentioned before, we believe the increased performance in ΔD is due to a change in D directly affecting the trajectory step sizes. Trajectories with fixed D and increasingly closer values of α are much harder to distinguish as the model needs to spot a change in "directionality" over stochastic consecutive steps.

4.3.4 Dependence on Biological Behaviours

Lastly, we evaluate the Jaccard index using the combined changepoints from $\alpha + D$ timeseries against simulated biological states. As Figure 4.17a highlights, the Jaccard index decreases as the number of changepoints increases for most biological states. The immobile and free states (trajectories containing $s = 0$ and $s = 2$

(a)



(b)

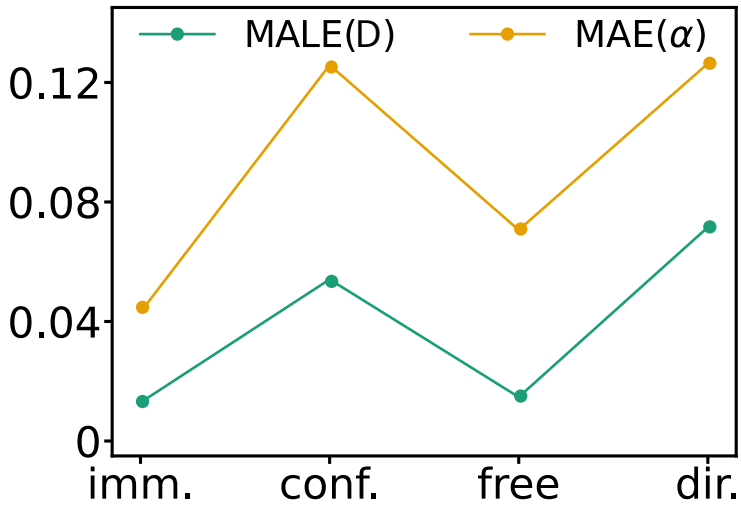


Figure 4.17: (a) Average Jaccard index ($\bar{J}$) for changepoint detection of various biological states and number of changepoints N_{CP}. Each point represents the average over 100 randomly selected protein trajectories of length 200 from the test set. (b) MAE and MALE for α and D predictions respectively from the test set by biological state.

respectively), provide high Jaccard values over a range of changepoints. We see decreased detection of changepoints using the $\alpha + D$ set for the confined and directed models (trajectories containing $s = 1$ and $s = 3$ respectively), particularly for a greater number of changepoints. This aligns with Figure 4.17b, which shows significantly higher values of MAE and MALE for the confined and directed models. The larger error in α and D for these models directly affects subsequent changepoint detection, so we expect to see lower Jaccard indices. The higher error in the confined model may arise from large values of D we simulated. As the protein reflects at a compartment boundary, consecutive time steps can appear close for both high and small values of D. This may impact the model negatively, as it sees higher and lower values of D during training for similarly looking coordinates. A similar issue may be present for directed trajectories, where high values of D can lead to large displacements, potentially creating trajectories similar to lower D values and higher values of α.

4.3.5 Shortcomings

As shown earlier, prediction error increases not only for shorter trajectories, but also for an increasing number of changepoints. Thus, M3 struggles on trajectories that involve both of these aspects. Figure 4.18a shows an example single state protein trajectory, of length 20, in which the model correctly predicts the diffusive properties for all variables. However, we can see on Figure 4.18b and Figure 4.18c, as soon as changepoints are introduced for short trajectories, the model is unable to pick up on changes in diffusive properties. Figure 4.18d highlights another example in which the protein trajectory, of length 200, enters a confined region. The change in α and D at the changepoint is very small and the model is unable to detect confinement. This trajectory, although at maximum length of 200, serves as an example where the model struggles despite the trajectory being long. These shortcomings may be improved upon and should be taken into account if wanting to adopt the model. As an example, we can bypass the short trajectory issue experimentally by designing

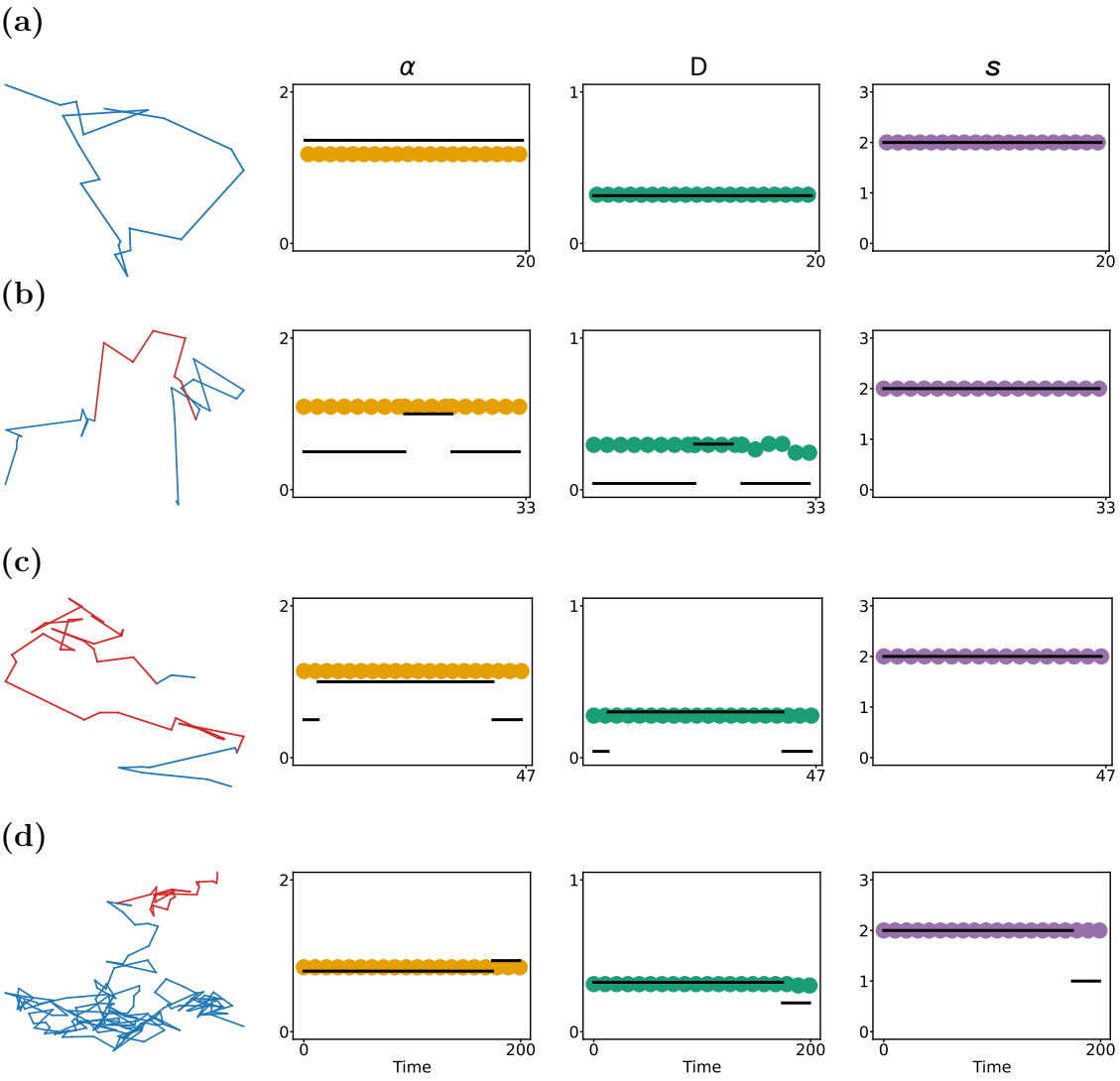


Figure 4.18: Model Shortcomings. (a) An example of model prediction for a short (length 20) single state trajectory, where it correctly predicts diffusive properties. (b) A length 33 multi-state trajectory with two changepoints. The model cannot pick up the changepoints and predicts a single value for both α and D. (c) Another example of the model not being able to pick up changes in diffusive properties, although it is able to correctly predict diffusive values of the majority of the multi state timeseries. (d) A confined trajectory with small changes in α and D between segments. The model also cannot determine the confinement region ($s = 1$) (although this is quite a tough trajectory to segment). For clarity, we plot a representative subset of predictions (circles) for every ground truth segment (black lines).

single molecule experiments that image proteins at lower laser intensities. This depletes the photon budget at a slower rate, thus obtaining longer trajectories.

4.4 Discussion

In this chapter we presented M3, a machine learning model with three stacked biLSTM layers for pointwise inference of diffusion coefficient, anomalous exponent, and biological state of a protein trajectory. Specifically, we show M3 performs well on noisy stochastic trajectories simulated from varying biologically relevant scenarios. Machine learning models provide a new and simple method for detecting complex biological behaviours of proteins in cells. Additionally, the model does not require prior expert knowledge or manual fine-tuning of thresholds, as done with traditional statistical methods relying on mean square displacements. In the Anomalous Diffusion Challenge, at the time of challenge closure, the model placed in the Top 5 on a set of combined metrics including absolute errors, Jaccard indices, and root mean square errors of predicted changepoints. Since then, minor changes have been made to the model, predominantly in the model architecture (addition of another layer normalisation) and the correction of a bug in the training data generation. The model consists of roughly half a million trainable parameters per variable, making it possible to train on a single NVIDIA GPU in 10 hrs. It is possible to combine the individual models into a single model with three output heads, but this is difficult as it combines classification and regression. This can serve as future work, as a more complex loss function with additional parameters would be required. On the contrary, if someone is only interested in one or a subset of the target variables, individual models for each variable, as done here, is easier to adopt.

Improvements can also be made to the architecture, such as the addition of convolutional layers. Convolutional layers may be less flexible with varying length timeseries but can be incorporated in a unified (multi-headed) architecture. The model can also be entirely replaced by transformers, which have shown to outperform LSTMs. We experimented with small transformer-like architectures with self-attention mechanisms and additional convolutional layers, but did not see significant improvements in metrics as compared to our current model [166, 182]. However, this experimentation was brief, not extensive, and can be explored further. Future changes to the model architecture should also focus on short trajectories

(< 150 time steps) as this is where the model performs worst, with a plateau in accuracy for longer trajectories (> 150 time steps, see Figure 4.10).

M3 is easily generalisable to higher dimensional data and can be further trained on real experimental data provided accurate labelling. For current applications to experimental single molecule data, longer trajectories are preferable, which may require running experiments using low laser intensities and photostable fluorophores. For verification with experimental PALM data, ground truth labelling can be subjective or not readily available as discussed in prior chapters. We can combat this with accurate simulations, through software such as Smoldyn [51], which take into account cell boundaries. What can be more challenging is evaluation of the resulting model against a diverse range of experimental conditions. As an initial step, this may consist of using a machine learning model for D to successfully predict and reproduce ensemble diffusion distributions as in Chapter 2.

Lastly, M3 makes use of specific features extracted from the original protein coordinates for this particular task. This may make the model less generalisable for applications to protein trajectories from real data. An architecture relying solely on protein coordinates, without feature extraction, may seem more attractive. Thus, the model can be made larger, with the addition of more learnable parameters, such that the resulting model can learn dependencies itself. Of course, this will require sufficient regularisation to avoid overfitting.

To conclude, machine learning approaches, such as M3, represent a significant advancement over classical MSD methods, offering frame-level analysis of diffusive properties. This granular insight into protein behaviour, combined with improving experimental techniques, opens new avenues for understanding cellular mechanisms. However, as with all black box models, rigorous testing remains essential for reliable scientific interpretation.

5
Conclusion and Future Outlook

Contents

5.1 Summary **127**
5.2 Future Work **129**

5.1 Summary

This thesis has explored novel machine learning and computational methods for analysing protein behaviours in bacterial cells, specifically *E. coli*. In Chapter 2, we apply standard ensemble diffusion analysis *in vivo* to study the mobilities of two previously uncharacterised sigma factors, sigma-54 and sigma-70, involved in transcription initiation. Using tracking PALM, we assigned distinct biological species to the sigma factors, revealing three populations [7, 11, 109]. We identified a very slow moving population, representing sigma factors interacting with the promoter site during transcription initiation (in the holoenzyme or bound independently in the case of sigma-54). Then, we identified the slow moving population, representing the holoenzyme during promoter search. Lastly, we found a fast moving population, representing lone sigma factors transiently binding to the DNA in search for the core RNAP. This chapter also emphasised the importance of different cellular stresses and

effects on the sigma factor sub-populations, with potential mechanisms highlighting the involvement of sigma-38. The stress diffusion distributions demonstrated that sigma-54 acts as a repressor in rich media, with repression less noticeable under increased nitrogen starvation. However, the complex interplay between the sigma factors, various cellular stresses, and pathways for handling stress needs to be further investigated. The findings of this Chapter provide new insights into sigma factor involvement in transcriptional mechanisms for living *E. coli* cells.

Building on Chapter 2, in Chapter 3 we developed a CNN for analysing protein trajectories at long exposure times. This allowed for frame level classifications of protein trajectories, and subsequent determination of protein binding times. Our method removes the need for parameter tuning in current threshold based methods. We also add a spatial filter onto our CNN, highlighting the improvement to the model when taking into account displacements. This filter is optional and easily replaceable with other custom filters (e.g., intensity or location based filters). In this chapter, we achieve similar binding times using both classical approaches and the CNN approach. This is beneficial for two reasons. Firstly, we show agreement with previous results. Secondly, the method returns frame level classifications, not possible with classical threshold based approaches. Additionally, the CNN takes into account defocusing and other experimental imperfections in the data through the use of augmentations in the training data. We have made use of PAmCherry in this work, which photobleaches very quickly (note however PAmCherry is very easy to work with in the wet lab making it a convenient choice). The use of more photostable fluorophores will lead to more transitions being present in the trajectories, leading to better performance of the CNN. The binding times of RNAP, sigma-54, and sigma-70 indicate potential long-lived non-specific interactions with the DNA. We do not expect these binding times to be specific interactions, as transcription timescales are much longer than the PAmCherry lifetime [19]. Lastly, we provided a quantitative framework for determining when "long exposure" effects become significant for various proteins. This is given by the quantity $\sqrt{\Delta D^* \tau_{exp}}$ determining the level of motion blur and hence effectiveness of the CNN in state classification.

Finally, in Chapter 4, we build upon the previous two Chapters using a general machine learning framework (M3) applicable to protein motion in complex cellular environments. We consider the cases of free diffusion with and without changes in diffusive properties, transitions into immobile and confined states, and lastly directed motion. Specifically, we develop a recurrent neural network (LSTM cells) for prediction of protein diffusion coefficient, anomalous exponent, and biological state from its trajectory. This is done pointwise, such that we obtain frame level information for a variety of behaviours. M3 demonstrates high accuracy for state detection and low absolute errors for diffusion coefficient and anomalous exponent.

Throughout this work, we have shown how machine learning can enhance our understanding of protein dynamics beyond traditional statistical methods. The approaches developed in this work are broadly applicable across different experimental setups and proteins, providing new tools for investigating cellular processes at the single molecule level. While tracking PALM provides the foundation for observing protein dynamics, machine learning approaches enable deeper insights from these observations.

5.2 Future Work

Given the findings of this work, there is much more that can be done in the field. As mentioned, Chapter 2 provides a foundation for future sigma factor work. Here, characterising the sigma factor interplay in different stresses will be very interesting. This can shed light on biological pathways for cellular responses to stress. Particularly, the interplay between sigma-54, sigma-38, and sigma-70 is of interest, and how their sub-populations affect each other. Future work can employ more photostable fluorophores such as Halo tag, although this requires more preparation than a simple PAmCherry tag. Halo tag has a lifetime roughly an order of magnitude higher (up to minutes) than PAmCherry, which may potentially show more transitions and reveal intricate dynamics. The use of a more photostable fluorophore will also be helpful in reducing photobleaching effects and determining protein binding times in Chapter 3.

In Chapter 3, the CNN framework can be extended for analysing long exposure trajectories for multiple states and complex behaviours. This will rely on the use of accurate simulations. Alternatively, other architectures can be deployed, such as Recurrent CNNs, taking into account previous and subsequent protein images, or Siamese CNNs, comparing two images at a time (e.g., clustering bound and diffusing images based on similarity). This may help when classifying lower exposure time images. The method may also have to rely on other features such as localisations, instead of images, if applied to stroboscopic data. Alternatively, unsupervised methods can be explored further, removing the need for labelling, a major hurdle in real experimental PALM data.

In a similar light, in Chapter 4, the LSTM architecture can be replaced by a transformer architecture with attention mechanisms. We briefly explored this architecture, alongside integration of convolution layers, but did not find significant improvement over the M3 architecture presented. However, this was only brief and further exploration may reveal improved performance. Attention mechanisms may also reveal changepoints in the trajectories without the need for implementing additional changepoint detection using cost functions (e.g., the model is paying "attention" to changes in diffusive behaviours). The network we developed used feature engineering combined with a small number of parameters ($\sim 500,000$). Larger networks could remove the need for extensive feature engineering. Lastly, to make the model more broadly applicable to tracking PALM data, simulations should take into account boundaries of the cell with reflecting boundary conditions. Accurate simulations, reflecting experimental conditions, are key in the development of accurate machine learning methods. The frameworks in this thesis can be adapted for membrane proteins, protein-protein interactions, or studies performed in other bacterial cells. They can also be integrated with other super resolution imaging techniques.

The development of machine learning models for single molecule imaging can be advanced with publicly accessible datasets and benchmarks. The Anomalous Diffusion Challenge has done great work in doing so, and more initiatives as such are

needed [50]. These datasets will allow for state-of-the-art machine learning models to be developed in an efficient and streamlined fashion, thus greatly enhancing both methodologies and understanding of cellular mechanisms in the field.

References

[1] Antoni Van Leeuwenhoek. "Observations, communicated to the publisher by Mr. Antony van Leewenhoeck, in a dutch letter of the 9th Octob. 1676. here English'd: concerning little animals by him observed in rain-well-sea- and snow water; as also in water wherein pepper had lain infused". In: *Philosophical Transactions of the Royal Society of London* 12.133 (Jan. 1997), pp. 821–831.

[2] Robert Hooke. *Micrographia: or, Some physiological descriptions of minute bodies made by magnifying glasses*. London, 1665.

[3] E. Abbe. "Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung". de. In: *Archiv für Mikroskopische Anatomie* 9.1 (Dec. 1873), pp. 413–468.

[4] J. G. White, W. B. Amos, and M. Fordham. "An evaluation of confocal versus conventional imaging of biological structures by fluorescence light microscopy". eng. In: *The Journal of Cell Biology* 105.1 (July 1987), pp. 41–48.

[5] O. Shimomura, F. H. Johnson, and Y. Saiga. "Extraction, purification and properties of aequorin, a bioluminescent protein from the luminous hydromedusan, Aequorea". eng. In: *Journal of Cellular and Comparative Physiology* 59 (June 1962), pp. 223–239.

[6] George H. Patterson and Jennifer Lippincott-Schwartz. "A Photoactivatable GFP for Selective Photolabeling of Proteins and Cells". In: *Science* 297.5588 (Sept. 2002), pp. 1873–1877.

[7] Eric Betzig et al. "Imaging Intracellular Fluorescent Proteins at Nanometer Resolution". In: *Science* 313.5793 (Sept. 2006), pp. 1642–1645.

[8] Russell E. Thompson, Daniel R. Larson, and Watt W. Webb. "Precise nanometer localization analysis for individual fluorescent probes". eng. In: *Biophysical Journal* 82.5 (May 2002), pp. 2775–2783.

[9] Fedor V. Subach et al. "Photoactivatable mCherry for high-resolution two-color fluorescence microscopy". en. In: *Nature Methods* 6.2 (Feb. 2009), pp. 153–159.

[10] Samuel T. Hess, Thanu P. K. Girirajan, and Michael D. Mason. "Ultra-High Resolution Imaging by Fluorescence Photoactivation Localization Microscopy". In: *Biophysical Journal* 91.11 (Dec. 2006), pp. 4258–4272.

[11] Suliana Manley et al. "High-density mapping of single-molecule trajectories with photoactivated localization microscopy". en. In: *Nature Methods* 5.2 (Feb. 2008), pp. 155–157.

[12] Stephan Uphoff, David J. Sherratt, and Achillefs N. Kapanidis. "Visualizing Protein-DNA Interactions in Live Bacterial Cells Using Photoactivated Single-molecule Tracking". In: *Journal of Visualized Experiments : JoVE* 85 (Mar. 2014), p. 51177.

[13] Makio Tokunaga, Naoko Imamoto, and Kumiko Sakata-Sogawa. “Highly inclined thin illumination enables clear single-molecule imaging in cells”. en. In: *Nature Methods* 5.2 (Feb. 2008), pp. 159–161.

[14] D Axelrod. “Cell-substrate contacts illuminated by total internal reflection fluorescence.” In: *Journal of Cell Biology* 89.1 (Apr. 1981), pp. 141–145.

[15] Stephan Uphoff et al. “Single-molecule DNA repair in live bacteria”. EN. In: *Proceedings of the National Academy of Sciences* 110.20 (May 2013), pp. 8063–8068.

[16] Jonathan B. Grimm et al. “Bright photoactivatable fluorophores for single-molecule imaging”. en. In: *Nature Methods* 13.12 (Dec. 2016), pp. 985–988.

[17] Sang-Hyuk Lee et al. “Counting single photoactivatable fluorescent molecules by photoactivated localization microscopy (PALM)”. eng. In: *Proceedings of the National Academy of Sciences of the United States of America* 109.43 (Oct. 2012), pp. 17436–17441.

[18] Mathew Stracy et al. “Super-Resolution Imaging of Transcription in Live Bacterial Cells”. English. In: *Biophysical Journal* 106.2 (Jan. 2014), 373a.

[19] Mathew Stracy et al. “Live-cell superresolution microscopy reveals the organization of RNA polymerase in the bacterial nucleoid”. EN. In: *Proceedings of the National Academy of Sciences* 112.32 (Aug. 2015), E4390–E4399.

[20] Mathew Stracy et al. “Single-molecule imaging of UvrA and UvrB recruitment to DNA lesions in living Escherichia coli”. en. In: *Nature Communications* 7.1 (Aug. 2016), p. 12568.

[21] Adolf Fick. “Ueber Diffusion”. en. In: *Annalen der Physik* 170.1 (1855), pp. 59–86.

[22] Otto G. Berg, Robert B. Winter, and Peter H. Von Hippel. “Diffusion-driven mechanisms of protein translocation on nucleic acids. 1. Models and theory”. In: *Biochemistry* 20.24 (Nov. 1981), pp. 6929–6948.

[23] Feng Wang et al. “The promoter-search mechanism of Escherichia coli RNA polymerase is dominated by three-dimensional diffusion”. en. In: *Nature Structural & Molecular Biology* 20.2 (Feb. 2013), pp. 174–181.

[24] Petter Hammar et al. “The lac Repressor Displays Facilitated Diffusion in Living Cells”. In: *Science* 336.6088 (June 2012), pp. 1595–1598.

[25] Johanna Weindl et al. “Modeling promoter search by *E. coli* RNA polymerase: One-dimensional diffusion in a sequence-dependent energy landscape”. In: *Journal of Theoretical Biology* 259.3 (Aug. 2009), pp. 628–634.

[26] Michael Slutsky and Leonid A. Mirny. “Kinetics of Protein-DNA Interaction: Facilitated Target Location in Sequence-Dependent Potential”. English. In: *Biophysical Journal* 87.6 (Dec. 2004), pp. 4021–4035.

[27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. 2016.

[29] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[30] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: Jan. 2017.

[31] Daniel Sage et al. “Super-resolution fight club: assessment of 2D and 3D single-molecule localization microscopy software”. en. In: *Nature Methods* 16.5 (May 2019), pp. 387–395.

[32] Carsen Stringer et al. “Cellpose: a generalist algorithm for cellular segmentation”. en. In: *Nature Methods* 18.1 (Jan. 2021), pp. 100–106.

[33] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (Jan. 1979), pp. 62–66.

[34] D. Marr and E. Hildreth. “Theory of edge detection”. eng. In: *Proceedings of the Royal Society of London. Series B, Biological Sciences* 207.1167 (Feb. 1980), pp. 187–217.

[35] Weiling Cai, Songcan Chen, and Daoqiang Zhang. “Fast and robust fuzzy *c*-means clustering algorithms incorporating local information for image segmentation”. In: *Pattern Recognition* 40.3 (Mar. 2007), pp. 825–838.

[36] L. Vincent and P. Soille. “Watersheds in digital spaces: an efficient algorithm based on immersion simulations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.6 (June 1991), pp. 583–598.

[37] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. “Snakes: Active contour models”. en. In: *International Journal of Computer Vision* 1.4 (Jan. 1988), pp. 321–331.

[38] Oleksii Sliusarenko et al. “High-throughput, subpixel precision analysis of bacterial morphogenesis and intracellular spatio-temporal dynamics”. en. In: *Molecular Microbiology* 80.3 (2011), pp. 612–627.

[39] Piers Turner. *napari-bacseg: Bacterial segmentation and analysis platform than can inport/export files in multiple formats. Integrating many tools such as Cellpose, ColiCoords, Oufti/MicrobeTracker.* https://github.com/piedrro/napari-bacseg.

[40] Alison Farrar et al. *Ribosome Phenotypes Enable Rapid Antibiotic Susceptibility Testing in Escherichia coli*. en. https://www.medrxiv.org/content/10.1101/2024.06.18.24309111v2, June 2024.

[41] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[42] Saga Helgadottir, Aykut Argun, and Giovanni Volpe. “Digital video microscopy enhanced by deep learning”. EN. In: *Optica* 6.4 (Apr. 2019), pp. 506–513.

[43] Benjamin Midtvedt et al. “Quantitative digital microscopy with deep learning”. In: *Applied Physics Reviews* 8.1 (Feb. 2021), p. 011310.

[44] Benjamin Midtvedt et al. “Single-shot self-supervised object detection in microscopy”. en. In: *Nature Communications* 13.1 (Dec. 2022), p. 7492.

[45] Jesús Pineda et al. "Geometric deep learning reveals the spatiotemporal features of microscopic motion". en. In: *Nature Machine Intelligence* 5.1 (Jan. 2023), pp. 71–82.

[46] Mathew Stracy et al. "Transient non-specific DNA binding dominates the target search of bacterial DNA-binding proteins". English. In: *Molecular Cell* 81.7 (Apr. 2021), 1499–1514.e6.

[47] Gorka Muñoz-Gil et al. "Objective comparison of methods to decode anomalous diffusion". en. In: *Nature Communications* 12.1 (Oct. 2021), p. 6253.

[48] Nitin Kapadia, Ziad W El-Hajj, and Rodrigo Reyes-Lamothe. "Bound2Learn: a machine learning approach for classification of DNA-bound proteins from single-molecule tracking experiments". In: *Nucleic Acids Research* 49.14 (Aug. 2021), e79.

[49] Gorka Muñoz-Gil et al. "Quantitative evaluation of methods to analyze motion changes in single-particle experiments". In: *Nat. Commun. Registered Report Stage 1 Protocol* (Jan. 2024).

[50] Gorka Muñoz-Gil et al. *AnDiChallenge/ANDI_datasets: Challenge 2020 release.* May 2021.

[51] Martin Robinson, Steven S. Andrews, and Radek Erban. "Multiscale reaction-diffusion simulations with Smoldyn". In: *Bioinformatics* 31.14 (July 2015), pp. 2406–2408.

[52] Federico Garza de Leon et al. "Tracking Low-Copy Transcription Factors in Living Bacteria: The Case of the lac Repressor". English. In: *Biophysical Journal* 112.7 (Apr. 2017), pp. 1316–1327.

[53] Francis Crick. "Central Dogma of Molecular Biology". en. In: *Nature* 227.5258 (Aug. 1970), pp. 561–563.

[54] Matthew Cobb. "60 years ago, Francis Crick changed the logic of biology". en. In: *PLOS Biology* 15.9 (Sept. 2017), e2003243.

[55] Richard R. Burgess. "Separation and Characterization of the Subunits of Ribonucleic Acid Polymerase". In: *Journal of Biological Chemistry* 244.22 (Nov. 1969), pp. 6168–6176.

[56] Xavier Darzacq et al. "In vivo dynamics of RNA polymerase II transcription". en. In: *Nature Structural & Molecular Biology* 14.9 (Sept. 2007), pp. 796–806.

[57] Katsuhiko S. Murakami. "Structural Biology of Bacterial RNA Polymerase". en. In: *Biomolecules* 5.2 (June 2015), pp. 848–864.

[58] Patrick Cramer, David A. Bushnell, and Roger D. Kornberg. "Structural Basis of Transcription: RNA Polymerase II at 2.8 Ångstrom Resolution". In: *Science* 292.5523 (June 2001), pp. 1863–1876.

[59] R. F. Rosenberger and Gillian Foskett. "An estimate of the frequency of in vivo transcriptional errors at a nonsense codon in Escherichia coli". en. In: *Molecular and General Genetics MGG* 183.3 (Nov. 1981), pp. 561–563.

[60] Abhishek Mazumder and Achillefs N. Kapanidis. "Recent Advances in Understanding σ70-Dependent Transcription Initiation Mechanisms". In: *Journal of Molecular Biology* 431.20 (Sept. 2019), pp. 3947–3959.

[61] Emily F. Ruff, M. Thomas Record, and Irina Artsimovitch. “Initial Events in Bacterial Transcription Initiation”. en. In: *Biomolecules* 5.2 (June 2015), pp. 1035–1062.

[62] D Pribnow. “Nucleotide sequence of an RNA polymerase binding site at an early T7 promoter.” In: *Proceedings of the National Academy of Sciences* 72.3 (Mar. 1975), pp. 784–788.

[63] H Schaller, C Gray, and K Herrmann. “Nucleotide sequence of an RNA polymerase binding site from the DNA of bacteriophage fd.” In: *Proceedings of the National Academy of Sciences of the United States of America* 72.2 (Feb. 1975), pp. 737–741.

[64] David Pribnow. “Bacteriophage T7 early promoters: Nucleotide sequences of two RNA polymerase binding sites”. In: *Journal of Molecular Biology* 99.3 (Dec. 1975), pp. 419–443.

[65] Calvin B. Harley and Robert P. Reynolds. “Analysis of E.Coli Pormoter sequences”. In: *Nucleic Acids Research* 15.5 (Mar. 1987), pp. 2343–2361.

[66] R. C. Dickson et al. “Genetic regulation: the Lac control region”. eng. In: *Science (New York, N.Y.)* 187.4171 (Jan. 1975), pp. 27–35.

[67] Kurt Fredrick and John D. Helmann. “RNA polymerase sigma factor determines start-site selection but is not required for upstream promoter element activation on heteroduplex (bubble) templates”. In: *Proceedings of the National Academy of Sciences* 94.10 (May 1997), pp. 4982–4987.

[68] Sergei Borukhov and Evgeny Nudler. “RNA polymerase holoenzyme: structure, function and biological implications”. In: *Current Opinion in Microbiology* 6.2 (Apr. 2003), pp. 93–100.

[69] K.S. Murakami, S. Masuda, and S.A. Darst. “Structural basis of transcription initiation: RNA polymerase holoenzyme at 4 Å resolution”. In: *Science* 296.5571 (2002), pp. 1280–1284.

[70] Andrey Feklistov and Seth A. Darst. “Structural Basis for Promoter -10 Element Recognition by the Bacterial RNA Polymerase σ Subunit”. English. In: *Cell* 147.6 (Dec. 2011), pp. 1257–1269.

[71] Andrew A. Travers and Richard R. Burgess. “Cyclic Re-use of the RNA Polymerase Sigma Factor”. en. In: *Nature* 222.5193 (May 1969), pp. 537–540.

[72] Robert D. Finn et al. “Escherichia coli RNA polymerase core and holoenzyme structures”. In: *The EMBO Journal* 19.24 (Dec. 2000), pp. 6833–6844.

[73] K.S. Murakami and S.A. Darst. “Bacterial RNA polymerases: The wholo story”. In: *Current Opinion in Structural Biology* 13.1 (2003), pp. 31–39.

[74] K.S. Murakami. “X-ray crystal structure of escherichia coli RNA polymerase σ70 holoenzyme”. In: *Journal of Biological Chemistry* 288.13 (2013), pp. 9126–9134.

[75] Sandhya Callaci, Ewa Heyduk, and Tomasz Heyduk. “Conformational Changes of Escherichia coli RNA Polymerase ς70 Factor Induced by Binding to the Core Enzyme *”. English. In: *Journal of Biological Chemistry* 273.49 (Dec. 1998), pp. 32995–33001.

[76] Yu Feng, Yu Zhang, and Richard H. Ebright. “Structural basis of transcription activation”. In: *Science* 352.6291 (June 2016), pp. 1330–1333.

[77] Wayne S. Kontur et al. “Solute Probes of Conformational Changes in Open Complex (RPo) Formation by Escherichia coli RNA Polymerase at the λPR Promoter: Evidence for Unmasking of the Active Site in the Isomerization Step and for Large-Scale Coupled Folding in the Subsequent Conversion to RPo”. In: *Biochemistry* 45.7 (Feb. 2006), pp. 2161–2177.

[78] Ruth M. Saecker, M. Thomas Record, and Pieter L. deHaseth. “Mechanism of Bacterial Transcription Initiation: RNA Polymerase - Promoter Binding, Isomerization to Initiation-Competent Open Complexes, and Initiation of RNA Synthesis”. In: *Journal of Molecular Biology*. Mechanisms of Transcription 412.5 (Oct. 2011), pp. 754–771.

[79] Tomohiro Shimada et al. “The Whole Set of Constitutive Promoters Recognized by RNA Polymerase RpoD Holoenzyme of Escherichia coli”. en. In: *PLOS ONE* 9.3 (Mar. 2014), e90447.

[80] Diego Duchi et al. “Conformational heterogeneity and bubble dynamics in single bacterial transcription initiation complexes”. In: *Nucleic Acids Research* 46.2 (Jan. 2018), pp. 677–688.

[81] Diego Duchi et al. “RNA Polymerase Pausing during Initial Transcription”. English. In: *Molecular Cell* 63.6 (Sept. 2016), pp. 939–950.

[82] Nicole C. Robb et al. “The Transcription Bubble of the RNA Polymerase–Promoter Open Complex Exhibits Conformational Heterogeneity and Millisecond-Scale Dynamics: Implications for Transcription Start-Site Selection”. In: *Journal of Molecular Biology* 425.5 (Mar. 2013), pp. 875–885.

[83] Robert Glyde et al. “Structures of Bacterial RNA Polymerase Complexes Reveal the Mechanism of DNA Loading and Transcription Initiation”. English. In: *Molecular Cell* 70.6 (June 2018), 1111–1120.e3.

[84] Je Ko and Tomasz Heyduk. “Kinetics of promoter escape by bacterial RNA polymerase: effects of promoter contacts and transcription bubble collapse”. In: *Biochemical Journal* 463.1 (Sept. 2014), pp. 135–144.

[85] Yuhong Zuo and Thomas A. Steitz. “Crystal Structures of the E. coli Transcription Initiation Complexes with a Complete Bubble”. English. In: *Molecular Cell* 58.3 (May 2015), pp. 534–540.

[86] Amanda Drennan et al. “Key Roles of the Downstream Mobile Jaw of Escherichia coli RNA Polymerase in Transcription Initiation”. In: *Biochemistry* 51.47 (Nov. 2012), pp. 9447–9459.

[87] Vladimir Mekler et al. “Coupling of Downstream RNA Polymerase–Promoter Interactions with Formation of Catalytically Competent Transcription Initiation Complex”. In: *Journal of Molecular Biology* 426.24 (Dec. 2014), pp. 3973–3984.

[88] Lilian M. Hsu. “Monitoring abortive initiation”. In: *Methods*. Methods Related to Bacterial Transcriptional Control 47.1 (Jan. 2009), pp. 25–36.

[89] Achillefs N. Kapanidis et al. “Initial Transcription by RNA Polymerase Proceeds Through a DNA-Scrunching Mechanism”. In: *Science* 314.5802 (Nov. 2006), pp. 1144–1147.

[90] Andrey Revyakin et al. “Abortive Initiation and Productive Initiation by RNA Polymerase Involve DNA Scrunching”. In: *Science* 314.5802 (Nov. 2006), pp. 1139–1143.

[91] Irina O. Vvedenskaya et al. “Interactions between RNA polymerase and the core recognition element are a determinant of transcription start site selection”. In: *Proceedings of the National Academy of Sciences* 113.21 (May 2016), E2899–E2905.

[92] Nam V. Vo et al. “In Vitro Studies of Transcript Initiation by Escherichia coli RNA Polymerase. 3. Influences of Individual DNA Elements within the Promoter Recognition Region on Abortive Initiation and Promoter Escape”. In: *Biochemistry* 42.13 (Apr. 2003), pp. 3798–3811.

[93] Lilian M. Hsu et al. “Initial Transcribed Sequence Mutations Specifically Affect Promoter Escape Properties”. In: *Biochemistry* 45.29 (July 2006), pp. 8841–8854.

[94] Emmanuel Margeat et al. “Direct Observation of Abortive Initiation and Promoter Escape within Single Immobilized Transcription Complexes”. English. In: *Biophysical Journal* 90.4 (Feb. 2006), pp. 1419–1431.

[95] Sarah A. Perdue and Jeffrey W. Roberts. “σ70-dependent Transcription Pausing in *Escherichia coli*”. In: *Journal of Molecular Biology*. Mechanisms of Transcription 412.5 (Oct. 2011), pp. 782–792.

[96] Timothy T. Harden et al. “Bacterial RNA polymerase can retain σ70 throughout transcription”. In: *Proceedings of the National Academy of Sciences* 113.3 (Jan. 2016), pp. 602–607.

[97] Bin Liu, Yuhong Zuo, and Thomas A. Steitz. “Structures of E. coli σS-transcription initiation complexes provide new insights into polymerase mechanism”. In: *Proceedings of the National Academy of Sciences* 113.15 (Apr. 2016), pp. 4051–4056.

[98] Dmitry G. Vassylyev et al. “Structural basis for transcription elongation by bacterial RNA polymerase”. en. In: *Nature* 448.7150 (July 2007), pp. 157–162.

[99] Amy Switzer et al. “A Role for the RNA Polymerase Gene Specificity Factor σ54 in the Uniform Colony Growth of Uropathogenic Escherichia coli”. In: *Journal of Bacteriology* 204.4 (Mar. 2022), e00031–22.

[100] C. A. Gross et al. “The Functional and Regulatory Roles of Sigma Factors in Transcription”. en. In: *Cold Spring Harbor Symposia on Quantitative Biology* 63 (Jan. 1998), pp. 141–156.

[101] Gil Bar-Nahum and Evgeny Nudler. “Isolation and Characterization of σ70-Retaining Transcription Elongation Complexes from *Escherichia coli*”. In: *Cell* 106.4 (Aug. 2001), pp. 443–451.

[102] Amy E. Danson et al. “Mechanisms of σ54-Dependent Transcription Initiation and Regulation”. In: *Journal of Molecular Biology*. RNA polymerase reaches 60 431.20 (Sept. 2019), pp. 3960–3974.

[103] Matthew Bush and Ray Dixon. “The Role of Bacterial Enhancer Binding Proteins as Specialized Activators of σ54-Dependent Transcription”. In: *Microbiology and Molecular Biology Reviews* 76.3 (Aug. 2012), pp. 497–529.

[104] Martin Buck and Wendy Cannon. “Specific binding of the transcription factor sigma-54 to promoter DNA”. en. In: *Nature* 358.6385 (July 1992), pp. 422–422.

[105] Jordan John et al. “Genome-wide promoter assembly in E. coli measured at single-base resolution”. eng. In: *Genome Research* 32.5 (May 2022), pp. 878–892.

[106] Jonathan T. Wang, Adeela Syed, and Jay D. Gralla. “Multiple pathways to bypass the enhancer requirement of sigma 54 RNA polymerase: Roles for DNA and protein determinants”. In: *Proceedings of the National Academy of Sciences* 94.18 (Sept. 1997), pp. 9538–9543.

[107] Lei Wang and Jay D. Gralla. “Multiple In Vivo Roles for the -12-Region Elements of Sigma 54 Promoters”. In: *Journal of Bacteriology* 180.21 (Nov. 1998), pp. 5626–5631.

[108] Yun Yang et al. “Structures of the RNA polymerase-σ54 reveal new and conserved regulatory strategies”. In: *Science* 349.6250 (Aug. 2015), pp. 882–885.

[109] Hafez El Sayyed et al. “Single-molecule tracking reveals the functional allocation, in vivo interactions, and spatial organization of universal transcription factor NusG”. English. In: *Molecular Cell* 84.5 (Mar. 2024), 926–937.e4.

[110] Steven Elmore et al. “Single-particle tracking of oriC-GFP fluorescent spots during chromosome segregation in Escherichia coli”. eng. In: *Journal of Structural Biology* 151.3 (Sept. 2005), pp. 275–287.

[111] Linda U. M. Johansson et al. “sigma54-RNA polymerase controls sigma70-dependent transcription from a non-overlapping divergent promoter”. eng. In: *Molecular Microbiology* 70.3 (Nov. 2008), pp. 709–723.

[112] Daniel R. Brown et al. “Nitrogen stress response and stringent response are coupled in Escherichia coli”. en. In: *Nature Communications* 5.1 (June 2014), p. 4115.

[113] A. Wedel and S. Kustu. “The bacterial enhancer-binding protein NTRC is a molecular machine: ATP hydrolysis is coupled to transcriptional activation”. eng. In: *Genes & Development* 9.16 (Aug. 1995), pp. 2042–2052.

[114] Tao Dong, Rosemary Yu, and Herb Schellhorn. “Antagonistic regulation of motility and transcriptome expression by RpoN and RpoS in Escherichia coli”. en. In: *Molecular Microbiology* 79.2 (2011), pp. 375–386.

[115] Avantika Lal, Sandeep Krishna, and Aswin Sai Narain Seshasayee. “Regulation of Global Transcription in Escherichia coli by Rsd and 6S RNA”. eng. In: *G3 (Bethesda, Md.)* 8.6 (May 2018), pp. 2079–2089.

[116] Anna Janaszak et al. “A sigma54-dependent promoter in the regulatory region of the Escherichia coli rpoH gene”. eng. In: *Microbiology (Reading, England)* 153.Pt 1 (Jan. 2007), pp. 111–123.

[117] Johan Elf, Gene-Wei Li, and X. Sunney Xie. “Probing Transcription Factor Dynamics at the Single-Molecule Level in a Living Cell”. In: *Science* 316.5828 (May 2007), pp. 1191–1194.

[118] Mathew Stracy et al. “Single-molecule imaging of DNA gyrase activity in living Escherichia coli”. eng. In: *Nucleic Acids Research* 47.1 (Jan. 2019), pp. 210–220.

[119] Mathew Stracy et al. “In vivo single-molecule imaging of bacterial DNA replication, transcription, and repair”. In: *FEBS Letters*. SI: Single molecule techniques - Applications in biology 588.19 (Oct. 2014), pp. 3585–3594.

[120] Haw Yang, Hao Li, and Tao Liu. “Photobleaching statistics in single-molecule on-/off-time distributions”. eng. In: *The Journal of Chemical Physics* 151.17 (Nov. 2019), p. 174101.

[121] Mathew Stracy and Achillefs N. Kapanidis. “Single-molecule and super-resolution imaging of transcription in living bacteria”. In: *Methods*. Transcriptional dynamics 120 (May 2017), pp. 103–114.

[122] Yann LeCun and Yoshua Bengio. “Convolutional networks for images, speech, and time series”. In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[123] Alexander Buslaev et al. “Albumentations: Fast and Flexible Image Augmentations”. In: *Information* 11.2 (2020).

[124] E. L. Kaplan and Paul Meier. “Nonparametric Estimation from Incomplete Observations”. In: *Journal of the American Statistical Association* 53.282 (June 1958), pp. 457–481.

[125] Scott M. Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Red Hook, NY, USA, Dec. 2017, pp. 4768–4777.

[126] Hendrik Deschout, Kristiaan Neyts, and Kevin Braeckmans. “The influence of movement on the localization precision of sub-resolution particles in fluorescence microscopy”. en. In: *Journal of Biophotonics* 5.1 (2012), pp. 97–109.

[127] Martin Lindén et al. “Pointwise error estimates in localization microscopy”. en. In: *Nature Communications* 8.1 (May 2017), p. 15115.

[128] J. Christof M. Gebhardt et al. “Single-molecule imaging of transcription factor binding to DNA in live mammalian cells”. en. In: *Nature Methods* 10.5 (May 2013), pp. 421–426.

[129] Zhe Liu, Luke D. Lavis, and Eric Betzig. “Imaging Live-Cell Dynamics and Structure at the Single-Molecule Level”. English. In: *Molecular Cell* 58.4 (May 2015), pp. 644–659.

[130] Francisco Balzarotti et al. “Nanometer resolution imaging and tracking of fluorescent molecules with minimal photon fluxes”. In: *Science* 355.6325 (Feb. 2017), pp. 606–612.

[131] Mickaël Lelek et al. “Single-molecule localization microscopy”. en. In: *Nature Reviews Methods Primers* 1.1 (June 2021), pp. 1–27.

[132] Mike Heilemann et al. “Subdiffraction-Resolution Fluorescence Imaging with Conventional Fluorescent Probes”. In: *Angewandte Chemie International Edition* 47.33 (2008), pp. 6172–6176.

[133] Alexey Sharonov and Robin M. Hochstrasser. “Wide-field subdiffraction imaging by accumulated binding of diffusing probes”. In: *Proceedings of the National Academy of Sciences* 103.50 (Dec. 2006), pp. 18911–18916.

[134] Michael J. Rust, Mark Bates, and Xiaowei Zhuang. "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM)". en. In: *Nature Methods* 3.10 (Oct. 2006), pp. 793–796.

[135] Stephan Uphoff and Achillefs N. Kapanidis. "Studying the organization of DNA repair by single-cell and single-molecule imaging". In: *DNA Repair*. Single Molecule Approaches: Watching DNA Repair One Molecule at a Time 20 (Aug. 2014), pp. 32–40.

[136] Anne-Marie Ladouceur et al. "Clusters of bacterial RNA polymerase are biomolecular condensates that assemble through liquid–liquid phase separation". In: *Proceedings of the National Academy of Sciences* 117.31 (Aug. 2020), pp. 18540–18549.

[137] Ibrahim I. Cisse et al. "Real-Time Dynamics of RNA Polymerase II Clustering in Live Human Cells". In: *Science* 341.6146 (Aug. 2013), pp. 664–667.

[138] Ulrike Endesfelder et al. "Multiscale Spatial Organization of RNA Polymerase in *Escherichia coli*". In: *Biophysical Journal* 105.1 (July 2013), pp. 172–181.

[139] Jun Fan et al. "RNA polymerase redistribution supports growth in E. coli strains with a minimal number of rRNA operons". In: *Nucleic Acids Research* 51.15 (Aug. 2023), pp. 8085–8101.

[140] Achillefs N. Kapanidis, Stephan Uphoff, and Mathew Stracy. "Understanding Protein Mobility in Bacteria by Tracking Single Molecules". In: *Journal of Molecular Biology*. Plasticity of Multi-Protein Complexes 430.22 (Oct. 2018), pp. 4443–4455.

[141] Michael C. Konopka et al. "Crowding and Confinement Effects on Protein Diffusion In Vivo". In: *Journal of Bacteriology* 188.17 (Sept. 2006), pp. 6115–6123.

[142] Nicolas Chenouard et al. "Objective comparison of particle tracking methods". en. In: *Nature Methods* 11.3 (Mar. 2014), pp. 281–289.

[143] Ulrike Endesfelder and Mike Heilemann. "Art and artifacts in single-molecule localization microscopy: beyond attractive images". en. In: *Nature Methods* 11.3 (Mar. 2014), pp. 235–238.

[144] Ralf Metzler and Joseph Klafter. "The random walk's guide to anomalous diffusion: a fractional dynamics approach". In: *Physics Reports* 339.1 (Dec. 2000), pp. 1–77.

[145] Benoit B. Mandelbrot and John W. Van Ness. "Fractional Brownian Motions, Fractional Noises and Applications". In: *SIAM Review* 10.4 (Oct. 1968), pp. 422–437.

[146] Ralf Metzler et al. "Anomalous diffusion models and their properties: non-stationarity, non-ergodicity, and ageing at the centenary of single particle tracking". en. In: *Physical Chemistry Chemical Physics* 16.44 (Oct. 2014), pp. 24128–24164.

[147] Jae-Hyung Jeon et al. "In Vivo Anomalous Diffusion and Weak Ergodicity Breaking of Lipid Granules". In: *Physical Review Letters* 106.4 (Jan. 2011), p. 048103.

[148] Eli Barkai, Yuval Garini, and Ralf Metzler. "Strange kinetics of single molecules in living cells". In: *Physics Today* 65.8 (Aug. 2012), pp. 29–35.

[149] Felix Höfling and Thomas Franosch. "Anomalous transport in the crowded world of biological cells". en. In: *Reports on Progress in Physics* 76.4 (Mar. 2013), p. 046602.

[150] Alessia Gentili and Giorgio Volpe. "Characterization of anomalous diffusion classical statistics powered by deep learning (CONDOR)". en. In: *Journal of Physics A: Mathematical and Theoretical* 54.31 (July 2021), p. 314003.

[151] Aykut Argun, Giovanni Volpe, and Stefano Bo. "Classification, inference and segmentation of anomalous diffusion with recurrent neural networks". en. In: *Journal of Physics A: Mathematical and Theoretical* 54.29 (June 2021), p. 294003.

[152] Òscar Garibo i Orts, Miguel A. Garcia-March, and J. Alberto Conejero. *Efficient recurrent neural network methods for anomalously diffusing single particle short and noisy trajectories*. en. Aug. 2021.

[153] Marloes Arts et al. "Particle Mobility Analysis Using Deep Learning and the Moment Scaling Spectrum". en. In: *Scientific Reports* 9.1 (Nov. 2019), p. 17160.

[154] Carlo Manzo. "Extreme learning machine for the characterization of anomalous diffusion from single trajectories (AnDi-ELM)". en. In: *Journal of Physics A: Mathematical and Theoretical* 54.33 (July 2021), p. 334002.

[155] Dezhong Li, Qiujin Yao, and Zihan Huang. "WaveNet-based deep neural networks for the characterization of anomalous diffusion (WADNet)". en. In: *Journal of Physics A: Mathematical and Theoretical* 54.40 (Sept. 2021), p. 404003.

[156] Nicolas Firbas et al. "Characterization of anomalous diffusion through convolutional transformers". en. In: *Journal of Physics A: Mathematical and Theoretical* 56.1 (Jan. 2023), p. 014001.

[157] Hippolyte Verdier et al. "Learning physical properties of anomalous random walks using graph neural networks". en. In: *Journal of Physics A: Mathematical and Theoretical* 54.23 (May 2021), p. 234001.

[158] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780.

[159] Gorka Muñoz-Gil et al. "Single trajectory characterization via machine learning". en. In: *New Journal of Physics* 22.1 (Jan. 2020), p. 013010.

[160] Naor Granik et al. "Single-Particle Diffusion Characterization by Deep Learning". eng. In: *Biophysical Journal* 117.2 (July 2019), pp. 185–192.

[161] Patrycja Kowalek, Hanna Loch-Olszewska, and Janusz Szwabiński. "Classification of diffusion modes in single-particle tracking data: Feature-based versus deep-learning approach". eng. In: *Physical Review. E* 100.3-1 (Sept. 2019), p. 032410.

[162] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. *An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling*. Apr. 2018.

[163] Jason Lines, Sarah Taylor, and Anthony Bagnall. "Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles". In: *ACM Trans. Knowl. Discov. Data* 12.5 (July 2018), 52:1–52:35.

[164] Hanna Loch-Olszewska and Janusz Szwabiński. "Impact of Feature Choice on Machine Learning Classification of Fractional Anomalous Diffusion". eng. In: *Entropy (Basel, Switzerland)* 22.12 (Dec. 2020), p. 1436.

[165] Stefano Bo et al. “Measurement of anomalous diffusion using recurrent neural networks”. In: *Physical Review E* 100.1 (July 2019), p. 010102.

[166] Borja Requena et al. “Inferring pointwise diffusion properties of single trajectories with deep learning”. In: *Biophysical Journal* 122.22 (Nov. 2023), pp. 4360–4369.

[167] Ziyuan Chen, Laurent Geffroy, and Julie S. Biteen. “NOBIAS: Analyzing anomalous diffusion in single-molecule tracks with nonparametric Bayesian inference”. eng. In: *Frontiers in Bioinformatics* 1 (2021), p. 742073.

[168] Xiang Qu et al. “Semantic Segmentation of Anomalous Diffusion Using Deep Convolutional Networks”. In: *Physical Review Research* 6.1 (Jan. 2024), p. 013054.

[169] Hélène Kabbech and Ihor Smal. “Identification of Diffusive States in Tracking Applications Using Unsupervised Deep Learning Methods”. In: *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*. Mar. 2022, pp. 1–4.

[170] Charles Truong, Laurent Oudre, and Nicolas Vayatis. “Selective review of offline change point detection methods”. In: *Signal Processing* 167 (Feb. 2020), p. 107299.

[171] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. New York, NY, USA, July 2019, pp. 2623–2631.

[172] Thorsten Wagner et al. “Classification and Segmentation of Nanoparticle Diffusion Trajectories in Cellular Micro Environments”. en. In: *PLOS ONE* 12.1 (Jan. 2017), e0170165.

[173] Satomi Matsuoka, Tatsuo Shibata, and Masahiro Ueda. “Statistical Analysis of Lateral Diffusion and Multistate Kinetics in Single-Molecule Imaging”. English. In: *Biophysical Journal* 97.4 (Aug. 2009), pp. 1115–1124.

[174] Marianne Renner et al. “A Simple and Powerful Analysis of Lateral Subdiffusion Using Single Particle Tracking”. In: *Biophysical Journal* 113.11 (Dec. 2017), pp. 2452–2463.

[175] Claus Metzner et al. “Superstatistical analysis and modelling of heterogeneous random walks”. en. In: *Nature Communications* 6.1 (June 2015), p. 7516.

[176] Yann Lanoiselée and Denis S. Grebenkov. “Unraveling intermittent features in single-particle trajectories by a local convex hull method”. eng. In: *Physical Review. E* 96.2-1 (Aug. 2017), p. 022144.

[177] Anthony R. Vega et al. “Multistep Track Segmentation and Motion Classification for Transient Mobility Analysis”. In: *Biophysical Journal* 114.5 (Mar. 2018), pp. 1018–1025.

[178] Patrice Dosset et al. “Automatic detection of diffusion modes within biological membranes using back-propagation neural network”. In: *BMC Bioinformatics* 17.1 (May 2016), p. 197.

[179] Shuhui Yin, Nancy Song, and Haw Yang. “Detection of Velocity and Diffusion Coefficient Change Points in Single-Particle Trajectories”. In: *Biophysical Journal* 115.2 (July 2018), pp. 217–229.

[180] Jo A. Helmuth et al. “A novel supervised trajectory segmentation algorithm identifies distinct types of human adenovirus motion in host cells”. eng. In: *Journal of Structural Biology* 159.3 (Sept. 2007), pp. 347–358.

[181] R. Killick, P. Fearnhead, and I. A. Eckley. "Optimal Detection of Changepoints With a Linear Computational Cost". In: *Journal of the American Statistical Association* 107.500 (Dec. 2012), pp. 1590–1598.

[182] Ashish Vaswani et al. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA, Dec. 2017, pp. 6000–6010.