École polytechnique fédérale de Lausanne

Semester project Fall 2025

Master in Computational Science and Engineering

Estimating Anisotropic and Multi-state Diffusion Tensors from Single Particle Tracking Data

Author:
Simon Anton

Biomedical Imaging Group

Supervisor:
Daniel Sage

Abstract

Single particle tracking (SPT) is a powerful technique for investigating the motion of particles or molecules. By understanding the molecular dynamics at play, you gain valuable information about the cellular environment. Using SPT data to predict diffusion parameters or characteristics has become a popular task in machine learning in recent years. In this work, we study three cases of molecular diffusion: anisotropic diffusion, two-state binding/unbinding, and two-state isotropic diffusion. We propose transformer based models that outperform the baseline models for all tasks. Due to the lack of experimental data, we have built a customizable simulation package for generating our training and testing sets. The model framework proposed we refer to as the cross-transformer.

1 Introduction

Molecular diffusion is a fundamental process present in cellular processes. Understanding the diffusive behavior of a particle provides important insight into the cellular environment in which it is suspended. Researchers commonly use fluorescence microscopy to track individual particles, then predict the diffusion coefficients via simple methods like mean-squared displacement (MSD). This method has been consistently used for its simplicity. However, with the limitations of imaging resolution, particle localization is an inherently noisy process. Thus, the method suffers greatly due to the noise in the particle trajectories constructed from the imaging data. Additionally, the method is sensitive to trajectory length, and breaks down further for heterogeneous behavior or non-ergodic processes. Due to this, researchers have sought better methods for predicting diffusion properties. Recent works have utilized deep learning models (namely CNNs and LSTMs) to predict or classify diffusion characteristics. These works typically utilize image data or trajectory data. Under the supervision of Dr. Sage, this project was started by another student who showed a proof of concept. In their work, they used a transformer with the frames from a video capturing a particle's trajectory as input. Additionally, they found that the model's performance improved when injecting a set of trajectory-derived features into the encoding used for prediction. However, their project was limited to only isotropic diffusion, standard Brownian motion, in which one diffusion coefficient describes the motion. The work presented here seeks to expand the previous work to more complex cases of molecular diffusion, including anisotropic and two-state diffusion. Anisotropic diffusion, directed Brownian motion, can be described using a diffusion tensor, while two-state diffusion describes the process in which a particle transitions from one form of motion to another. In this work, we present three different models for three different cases of molecular diffusion: anisotropic diffusion, two-state diffusion with binding or unbinding, and two-state isotropic diffusion.

2 Related Work

The Anomalous Diffusion (AnDi) challenge was created as an open competition to test new methods and inspire further work towards the characterization of anomalous diffusion, where particle diffusion has a non-linear relationship between the MSD and time. The challenge's results showed that machine learning based approaches achieved superior performance for all tasks [4].

Park et al. introduced Pix2D, a convolutional neural network (CNN) model to predict the diffusion coefficient directly from a stack of single molecule fluorescence images [5]. Due to the stochastic nature of the diffusion, a single localized image is not sufficient to predict the respective diffusion coefficient D, which describes the physics of the motion. Thus, they used a stack

of uncorrelated, localized single particle images from simulations using the same value D as input for predicting D. Notably, this work does not make use of trajectory data and instead relies fully on the motion blur present in the images to infer the diffusion coefficient.

Other works such as WADNet [3] or that of Granik et al. [1] use the constructed trajectories as input to a model for regression or classification tasks. Requena et al. proposed STEP which takes a trajectory as input and outputs a pointwise prediction of the diffusion coefficient at every time step [6]. The architecture proposed in their work combines convolutional and transformer mechanisms.

Building upon these approaches, our project aims to integrate image and trajectory-based approaches into one framework while also providing support for analysis of more complex diffusion scenarios than simple Brownian motion.

3 Method

3.1 Physics Model

To characterize the diffusion parameters of a molecule, a model is required to represent the underlying physics of the motion and the imaging process.

3.1.1 Anisotropic Diffusion

Anisotropic diffusion in 2D can be modeled as a rank-2 diffusion tensor given by $\mathbf{D} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$. For isotropic diffusion, the diffusion tensor is simply $D\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Instead of representing the diffusion tensor in this form, we utilize the fact that this tensor is a real, symmetric matrix. Knowing this, we can use an eigenvalue decomposition to represent the tensor as

$$\mathbf{D} = \mathbf{R}\Lambda\mathbf{R}^T \tag{1}$$

where $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is the rotation matrix with the eigenvectors of $\mathbf{D}$ along each column, and $\Lambda = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{D}$. θ is an angle with respect to the x-axis, representing the orientation of the axis of the largest eigenvalue D_1. We choose to represent anisotropic diffusion by the rotation matrix $\mathbf{R}$ and the diagonal entries of Λ, $[D_1, D_2]$.

3.1.2 Simulation

Due to the lack of experimental data, it is necessary to simulate particle motion to create data for training a deep learning model. Brownian motion is described by a variance in position of $\sigma^2 = 2Ddt$ for 1D motion. In the case of directed 2D diffusion, the variance in each principal axis is described by $\sigma_1^2 = 2D_1dt$ and $\sigma_2^2 = 2D_2dt$, respectively. The displacement in one time step in one direction can be modeled as $\mathcal{N}(0,1)\sqrt{2Ddt}$, where $\mathcal{N}(0,1)$ is the normal distribution with zero mean and unit variance. However, this displacement must be corrected for the orientation of the motion with respect to the standard x-y axes. Thus, we must apply a rotation matrix to the displacement vector $\begin{bmatrix} \Delta p_1 \\ \Delta p_2 \end{bmatrix}$. The general update in a trajectory simulation becomes

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \mathbf{R}(\boldsymbol{\mathcal{N}} \odot \sqrt{2\mathbf{D}dt}) \tag{2}$$

where $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, $\boldsymbol{\mathcal{N}} = \begin{bmatrix} \mathcal{N}(0,1) \\ \mathcal{N}(0,1) \end{bmatrix}$, $\mathbf{D} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}$, and $\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$.

For the general case of anisotropic diffusion, each trajectory is generated by first uniformly sampling the first diffusion coefficient from the set $\{0.05, 0.10, ..., 6.0\}$ $\frac{\mu m^2}{s}$. For the second diffusion coefficient, we sample the ratio D_2/D_1 uniformly from $[0.1, 1)$. The angle between the x-axis and the direction of the first diffusion coefficient is sampled uniformly from $[0, \pi)$. The sim-

ulation is conducted with a 100 Hz frame rate and a timestep of 1 ms.

Once trajectory data is generated, it is necessary to simulate the image acquisition process to mimic fluorescence microscopy. First, we had to decide how to partition the trajectory to create frames. Due to the frame rate and time step we selected, 10 particle positions are captured within a single frame. For each frame, we center the sub-trajectory, up-sample to fit a 55x55 frame, convolve each sub-position with the microscope's point spread function (PSF), and aggregate the convolutions into one frame. Finally, we down-sample to get the final image size of 11x11. Gaussian noise is added to represent background fluctuations, and Poisson noise is applied to simulate the photon statistics of the acquisition process. Fig. 1 provides an example of a generated image along with the version without noise.

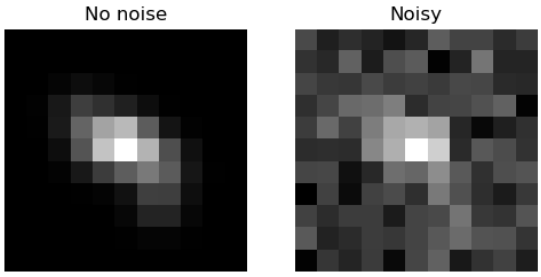

Figure 1: Simulation Sample

In addition to the images, we also compute the centroid of each frame with respect to pixel intensity, then recover the position of this centroid within the original trajectory plane by adding the already computed sub-trajectory mean. We use this data to compute the displacement between each adjacent pair of frames, to maintain a spatial relationship between them. In the case of experimental data, the localization of particle positions during tracking for frame generation will allow users to easily obtain the same notion of position data as in the simulation.

The simulation pipeline is highly configurable such that noise, resolution, number of frames, etc. can be adjusted for different setups. We generate a constant 30 frames per trajectory and keep all imaging settings constant for all simulated data.

3.1.3 Alterations for Two-state Scenarios

Binding describes the scenario where a particle or molecule is confined in a spatial region for a period of time. During this period, the molecule's motion is limited significantly. To simulate this process, we represent the confined state as isotropic diffusion with a diffusion coefficient uniformly sampled from the range $[0.005, 0.01]$ $\frac{\mu m^2}{s}$. We randomly select a timestep in the simulation for the change of state to occur. For any generated dataset, we evenly represent the transition from bound to unbound and unbound to bound. We model the unbound state as a highly diffusive anisotropic state with the primary diffusion coefficient being sampled from the range $[3, 6]$ $\frac{\mu m^2}{s}$ and the second diffusion coefficient being sampled as a fraction of the former, as in the standard simulation process.

The isotropic case is modeled as two-state diffusion, where each state is isotropic diffusion. For each state, we sample the diffusion coefficient uniformly from the set $\{0.05, 0.10, ..., 6.0\}$ $\frac{\mu m^2}{s}$. Again, we randomly select a timestep in the simulation for the change of state to occur.

3.2 Model

With two modes of data, images and displacements, it becomes necessary to find a way to effectively utilize both for prediction. Xu et al. presented a multi-modal learning with transformers survey that included six fundamental methods to combine modalities [8]. Fig. 2 shows the backbone of each of these architectures. We started by implementing each of these models, and after early testing, the cross attention to concatenation method showed to be most promising. The method we propose in this paper

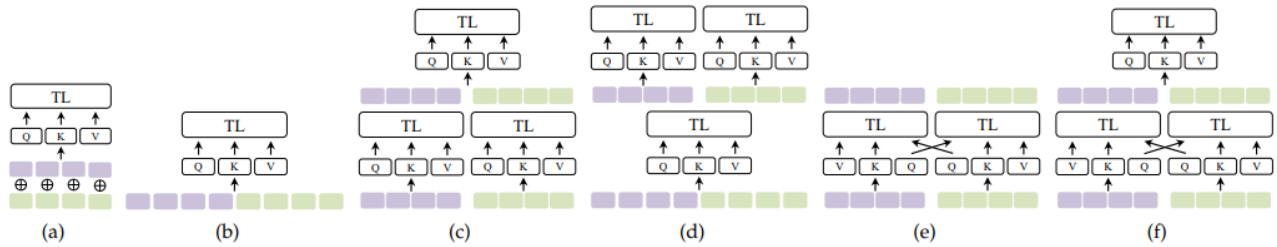

Figure 2: Transformer-based cross-modal interactions: (a) Early summation, (b) Early Concatenation, (c) Hierarchical Attention (multi-stream to one-stream), (d) Hierarchical Attention (one-stream to multi-stream), (e) Cross-Attention, and (f) Cross-Attention to Concatenation. "Q": Query embedding; "K": Key embedding; "V": Value embedding. "TL": Transformer Layer. [8]

is built upon a transformer backbone utilizing a cross attention to concatenation setup. We will refer to this as a cross-transformer.

3.2.1 Cross-transformer

An overview of the model is depicted in Fig. 3. The cross-transformer receives as input two 1D sequences of token embeddings: one for the image tokens and another for the displacement tokens. Traditional vision transformers convert an input image into a sequence of smaller patches, but, since we are working with low resolution images, we are already able to capture local context sufficiently with a single encoding. Thus, we generate one image token per frame in a video. We generate one displacement token per frame in a video, where the token for frame i corresponds to the displacement vector between frames i and $i-1$. For the first frame in a sequence, there is no natural previous frame to calculate the displacement, so we assign a default displacement corresponding to the mean of the training set $[\mu_x, \mu_y]$. With normalized displacements in each direction, this default entry is $[0, 0]$.

Our input image sequence is $\mathbf{z}_A \in \mathbb{R}^{F\times W\times W}$, where F is the number of frames and W is the output size in pixels for the width and height. Our displacement sequence is $\mathbf{z}_B \in \mathbb{R}^{F\times 2}$. Images are encoded using a ResNet-based architecture as proposed in [2], while we use a small 2-layer MLP for encoding our displacement data. Additionally, a learned embedding is added to every token indicating whether it belongs to the image modality A or displacement modality B. Our input to the cross-transformer is two sequences $\mathbf{Z}_{A,B} \in \mathbb{R}^{F\times d}$, where d is the embedding dimension.

The cross attention to concatenation method consists of two levels: a mutual cross attention section with two transformers and a global context transformer. The first level consists of two transformers where each modality learns its relationship to the tokens of the other modality through their own respective transformer. The second level concatenates the output of each transformer in the first level and passes the new sequence into another transformer. This transformer learns the global context by attending to all tokens within its own modality and those of the other modality. Equation 3 shows the outputs of the main parts of the model. *MHSA* represents multi-head self attention used inside a transformer layer. The key aspect is that the query tokens $\mathbf{Q}$ are from the other modality. *Tf* represents a transformer, and C represents the concatenation of the tokens from modalities $\mathbf{A}$ and $\mathbf{B}$.

$$\begin{cases} \mathbf{Z}_A^0 \leftarrow MHSA(\mathbf{Q}_B, \mathbf{K}_A, \mathbf{V}_A) \\ \mathbf{Z}_B^0 \leftarrow MHSA(\mathbf{Q}_A, \mathbf{K}_B, \mathbf{V}_B) \\ \mathbf{Z} \leftarrow Tf(C(\mathbf{Z}_A, \mathbf{Z}_B)) \end{cases} \tag{3}$$

For making a sequence-wise prediction, most

transformer architectures prepend a learnable embedding to the input sequence, typically referred to as the `[class]` token. Once the sequence is processed by the transformer, we use this special token for the prediction task. Typically, this involves passing this embedding through a small MLP before classification or regression. In our case, we have three separate transformers. We prepend a `[class]` token to each of the modality-specific sequences in the first layer. Then, as input to the transformer in the second layer, we learn a linear combination of the class tokens outputted and set that as the class token for the input sequence.

$$\mathbf{z}_0^C = \alpha \mathbf{z}_0^A + (1-\alpha)\mathbf{z}_0^B \tag{4}$$

In Equation 4, alpha is constrained to be between 0 and 1 and we denote C as the sequence created from concatenating the outputs from the first level of transformers. We use the encoding of $\mathbf{z}_0^C$ output from the final transformer for our regression task. We pass it through a small 2-layer MLP for our prediction of the decomposed diffusion tensor: the two eigenvalues and the orientation θ. Instead of directly predicting θ, we predict $\sin 2\theta$ and $\cos 2\theta$ for stability and periodicity independence then reconstruct θ as needed.

3.2.2 Multi-state Modification

It is common for molecules to transition from one form of diffusion to another. This can occur when two molecules bind to each other or when there is a change in environment. Predicting a sequence-wise diffusion tensor will not work when there is a transition point from one state to another. To capture the change in state, we elected to use pointwise predictions. Pointwise predictions are well suited for transformers since each token attends to the global context. Instead of prepending class tokens to the input sequences and using those to build a class representation for prediction, we use the encoding of each token for separate predictions. As output of our final transformer, we have a sequence

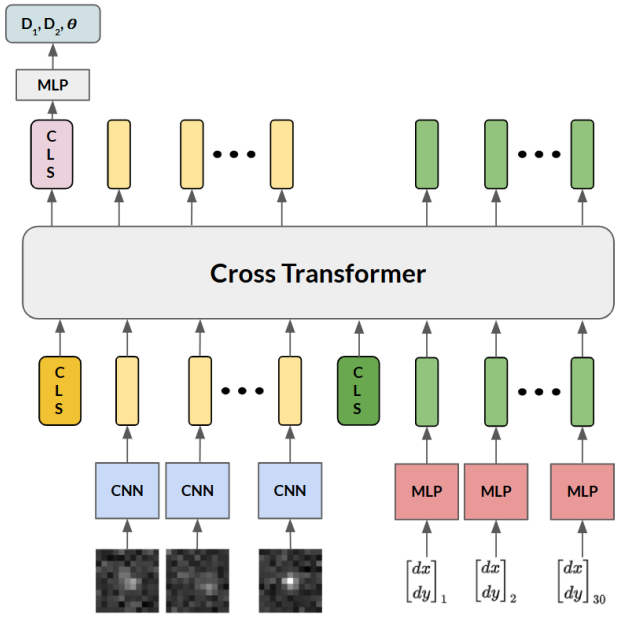

Figure 3: Model Overview

composed of N image tokens and N displacement tokens, each $\mathbf{x} \in \mathbb{R}^d$. We concatenate each (image, displacement) token pair to form a sequence $\mathbf{Z} \in \mathbb{R}^{N \times 2d}$. Each token in this sequence is passed to the MLP for prediction. Our multi-state model now makes a prediction for every frame in the input video.

When there is only one diffusion state present, the ordering of the frames relative to one another is largely irrelevant due to the stochastic nature of the underlying diffusion process. Any given frame is equally likely to occur in a different position in the image sequence. This assumption no longer holds when a video sequence contains multiple states. Frames should be interpreted in the context of their relative temporal positions because their meaning depends on surrounding frames from potentially different states. Incorporating positional information allows a token to weight or condition on neighboring tokens appropriately when constructing its representation within transformer layers. Thus, we have elected to use RoPE embeddings to encode absolute position through a rotation matrix and explicit relative position dependency in the self-attention

formulation [7]. RoPE is used only for the two transformers in the first level of the model.

4 Experiments

4.1 Training

All models were trained using AdamW as an optimizer and a cosine annealing scheduler. We used a MSE loss for our predictions. Since we predict four values, D_1, D_2, $\sin 2\theta$, and $\cos 2\theta$, the loss is the sum of mean squared errors for each prediction. A learning rate of 1e-4 was used. Models were trained with datasets consisting of 100k - 300k entries. 10-30 epochs were trained. An embedding dimension of 64 with a hidden dimension of 128 performed best, along with 6 transformer layers and 4 attention heads. The resulting architecture contains roughly 900k parameters, including encoders.

4.2 Baselines

We have created a set a baseline models to test our model against. We first evaluated on the traditional mean-squared displacement (MSD) method which relies on accurate particle localization. To simulate the localization process, we took the centroid of each frame generated from our simulator and set this point as the particle localization. Testing from the previous work showed that MSD performs most reliably when only fitting the first 10% of the MSD curve, so this is the setup we test on. In addition to our multi-modal cross-transformer, we also trained a version using only image data. A cross-transformer is not used anymore, and instead a standard transformer is used. The resulting model contains around 500k trainable parameters. Also, we created a unidirectional LSTM based model of comparable size to our transformer based model with 800k trainable parameters. Finally, we replicated the architecture of Pix2D described in [5] which contains roughly 260k trainable parameters. This model encodes each frame, then the average encoding across all frames is used for prediction. For testing on the multi-state scenarios, we alter the LSTM and Pix2D architectures to make frame-wise predictions. MSD cannot be applied to multi-state data without knowledge of the transition point between states.

4.3 PSF Variation

The point-spread function (PSF) describes the response of a focused optical imaging system to an idealized point source of light. Altering the PSF results in altering the level of blurriness of a captured source of light in the image. Fig. 4 displays a sample generated with varying PSF sizes. The figure shows that directionality and localization precision are hindered as the PSF size grows. For the case of anisotropic diffusion, the motion blur captured in an image is crucial to inferring the parameters of the diffusion tensor.

Our simulation and training are conducted using a fixed PSF size. The PSF is a function of the imaging system, so it is necessary to study the effect of this parameter on model performance. Using a fixed model architecture and training setup with one epoch, we trained a model for each PSF size studied. Fig. 5 shows the results of this experiment. As expected, when blurring is increased, model performance worsens. This doesn't necessarily signify model's trained on larger PSF sizes are unable to match the performance of those trained on smaller PSF sizes. It may be possible to match performance with more training data and epochs of training. The experiment does however show that the task is more difficult with larger PSF sizes.

4.4 Background Noise Variation

Our data is simulated with Gaussian background noise. We used experimental data to set the particle pixel intensity and background noise pixel

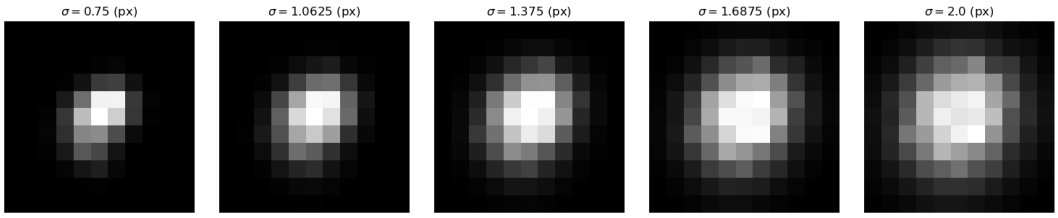

Figure 4: Sample generated with varying PSF sizes

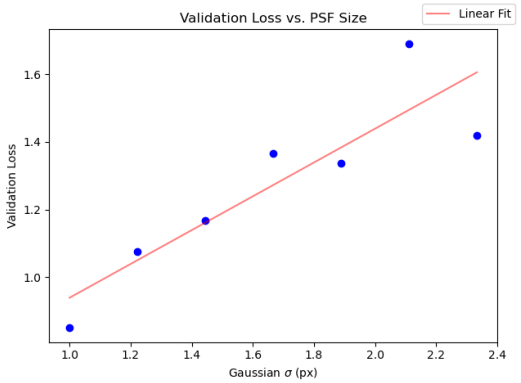

Figure 5: Performance with varying PSF sizes

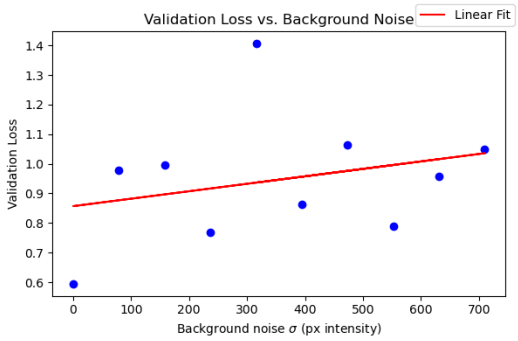

Figure 6: Performance with varying background noise

intensity. By altering the background noise intensity, we can study the model's robustness to noise. Fig. 7 shows a sample generated with different mean background noise pixel intensities. Our simulation and training are conducted using a fixed background mean intensity. To study the effects of different background noise, we trained on different noise intensities using a fixed model architecture and training setup with one epoch. One model is generated for each noise value studied. Fig. 6 shows the result of this experiment. With increasing background noise, model performance tends to worsen. The validation loss appears to saturate as background noise increases, with no clear linear trend.

The experiments show that increasing the PSF size or background noise leads to worse performance for fixed training settings. Learning is still possible, but will likely require more training data and/or model complexity. Fig. 8 provides a useful visualization of a sample generated under varying PSF sizes and background noise.

5 Results

We present three transformer-based models for predicting the components of a rank-2 diffusion tensor:

1. Cross-transformer for Anisotropic Diffusion
2. Cross-transformer for Two-state Binding/Unbinding
3. Cross-transformer for Two-state Isotropic Diffusion

These models were trained on simulated data matching real experimental conditions. Several baselines were used to compare performance, and our models outperform each. There are relatively few publications for directly predicting the components of diffusion tensors for the scenarios we are simulating. For example, Pix2D was created for and applied to isotropic diffusion data. We simply adapted and applied their architecture to our diffusion scenarios as a baseline.

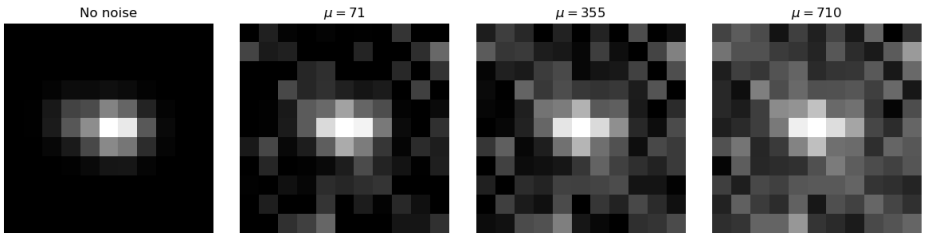

Figure 7: Sample generated with varying background noise

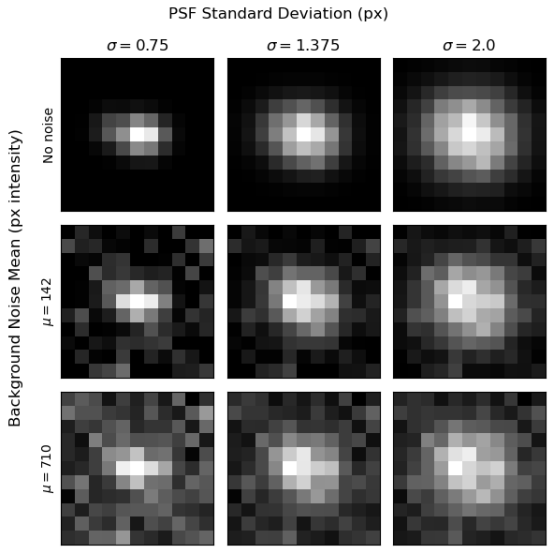

Figure 8: Sample generated with varying PSF sizes and background noise

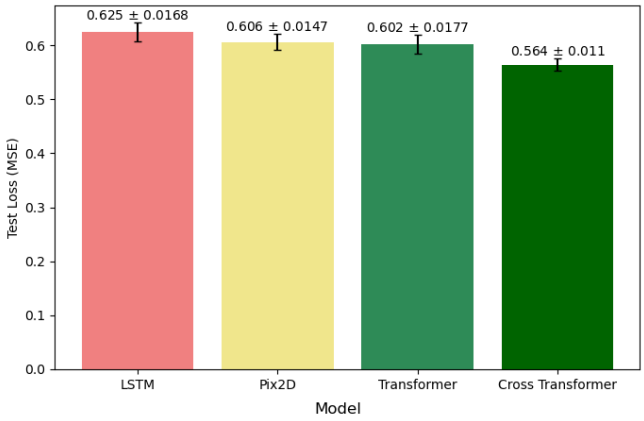

Figure 9: Anisotropic Case Test Losses

5.1 Anisotropic Case

For general anisotropic diffusion, we have a multi-modal cross-transformer and an image-only transformer. Both models outperform the baselines: MSD, LSTM, and Pix2D. MSD results in a MSE loss of 18.18 which is over 30 times worse than the proposed models. Fig. 9 displays the test results for each ML-based model. Using displacement features results in a statistically significant difference between the transformer-based models. Pix2D and the base transformer have similar results, but there is slight improvement with the transformer. The LSTM model performs the worst for this case from a MSE loss perspective.

This representation of model performance can be incomplete, instead, Table 1 shows the mod-

	MAE $\mathbf{D}_1$ $(\frac{\mu m^2}{s})$	**MAE** $\mathbf{D}_2$ $(\frac{\mu m^2}{s})$	**Angle Similarity**
Pix2D	0.47	0.353	0.906
LSTM	0.37	0.261	0.902
Transformer	0.358	0.249	0.905
Cross Transformer	0.361	0.245	0.914

Table 1: Anisotropic Diffusion Results

	MAE D_1 ($\frac{\mu m^2}{s}$)	MAE D_2 ($\frac{\mu m^2}{s}$)	Angle Similarity
Pix2D	0.578	0.556	0.87
LSTM	0.375	0.339	0.92
Cross Transformer	0.279	0.228	0.945

Table 2: Two-state Binding Results

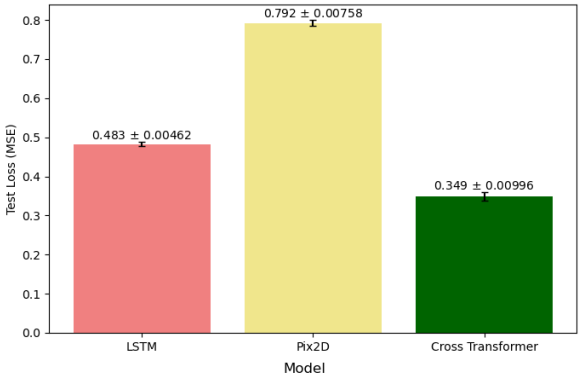

Figure 10: Two-state Binding Case Test Losses

els' performances for each parameter predicted of the decomposed diffusion tensor. For diffusion coefficients, we present the error in terms of mean absolute error (MAE), in $\frac{\mu m^2}{s}$. For angle predictions, we present the similarity between the predicted and ground truth angle. We determine this value by calculating the dot product between each respective unit vector, $[\cos\theta_{pred}, \sin\theta_{pred}] \cdot [\cos\theta_{GT}, \sin\theta_{GT}]$. The MAE for each diffusion coefficient is the largest for the Pix2D model. All other models achieve MAEs of around 0.1 $\frac{\mu m^2}{s}$ less. However, the angle similarity is slightly higher for Pix2D versus the LSTM model's. Considering the MSE loss of the LSTM model was largest, there appears to be a discrepancy between the results presented in Fig. 9 and Table 1. We will discuss the reasoning for this in 6.1. The base transformer and the cross-transformer achieve very similar performance for the diffusion coefficient predictions. However, the cross-transformer is able to better capture the orientation of the diffusive state.

5.2 Binding Case

The alterations for the two-state scenarios means Pix2D simply encodes each image in the sequence and makes a prediction for that encoding. No sense of temporal information is retained for that model. Fig. 10 displays the test results for each model. The cross-transformer performs significantly better for this case. The LSTM model performs second best with 1.4x worse performance. Pix2D achieves 2.26x worse performance compared to our model. Table 2 provides a more in depth view of the performance.

The cross-transformer performs best across all predictions. Notably, the angle similarity is significantly larger than in the anisotropic case. This is due to us setting the angle for the bound state to 0 degrees. This leads to a significant number of frames labeled with the angle being 0 degrees, allowing our model to learn to accurately predict this value. The MAE values are worse for Pix2D than in the anisotropic case, while they are slightly worse for the LSTM model. The LSTM model is able to improve for angle predictions, likely for the same reasons as the cross-transformer.

Another important aspect of making two-state predictions involves accurately predicting the point in time where the transition between states occurs. A model could be trained to output this value as a sequence wise prediction, but it would be easier to utilize our pointwise predictions to automatically detect this transition point. We detect the changepoint by choosing the frame number that minimizes a cost function. Formally, $\arg\min_i \mathcal{L}_i$, where Eq. 5 shows the cost function with n being the number of frames. The equation computes the sum of squared deviations for each region corresponding to a change of state between frames i and $i+1$. The change of state can occur within a frame itself, but we assume otherwise for simplicity.

	Changepoint MAE (Frames)
Pix2D	0.674
LSTM	0.713
Cross Transformer	0.412

Table 3: Changepoint Errors for the Binding Case

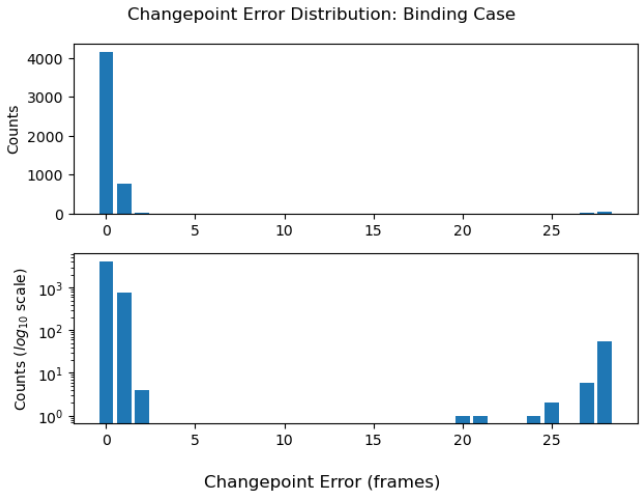

Figure 11: Changepoint Error Distribution for the Binding Case

$$\mathcal{L}_i = \sum_{j=1}^{i}\sum_{k=1}^{2}(x_{j,k} - \bar{x}_{:j,k})^2 + \sum_{j=i+1}^{n}\sum_{k=1}^{2}(x_{j,k} - \bar{x}_{j:,k})^2 \tag{5}$$

We optimize the cost function for our predictions and labels then compute the mean absolute difference between the changepoint predictions. Table 3 shows the result for each model. This result shows that the cross-transformer is able to accurately capture when the transition between states occurs. Pix2D performs second best, while LSTM performs worst. All models predict the changepoint on average within one frame. Fig. 11 shows the distribution of changepoint errors for a test set of 5000 datapoints using the cross-transformer model for inference. Over 80% of predictions are correct, and roughly 10% of predictions are off by one frame. The plot displaying counts on the log scale show that we sometimes make predictions errors of over 20 frames.

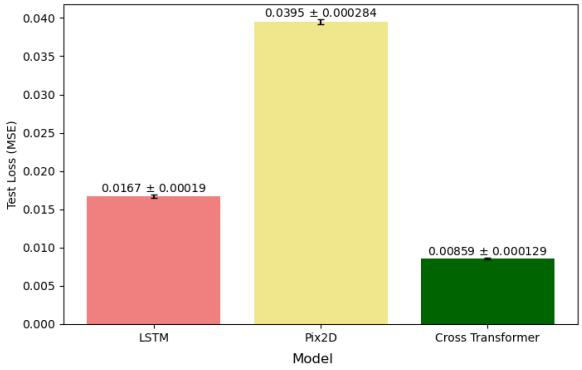

Figure 12: Two-state Isotropic Case Test Losses

	MAE $\mathbf{D}_1$ ($\frac{\mu m^2}{s}$)	MAE $\mathbf{D}_2$ ($\frac{\mu m^2}{s}$)
Pix2D	0.97	0.97
LSTM	0.559	0.559
Cross-transformer	0.394	0.394

Table 4: Two-state Isotropic Results

5.3 Isotropic Case

Fig. 12 displays the test results for this case. Again, the cross-transformer performs best, with a 1.94x better performance than the LSTM model and 4.6x better performance than the Pix2D model. Here, the test loss is noticeably an order of magnitude lower than the previous two cases. This is because we only predict the diffusion coefficients for the isotropic case, leaving out the angle loss term.

Table 4 shows the component-wise results for each model. This task proves to be more difficult than the binding case as both baseline models perform worse relative to our model. Importantly, the model predicts the same diffusion coefficient for both D_1 and D_2. This is the expected result for isotropic diffusion.

Table 5 shows the average changepoint errors for each model. The LSTM model performs worst and Pix2D is slightly better. These are significantly larger than those for the binding case. For simulations composed of 30 frames, our predictions with the cross-transformer model are 1/6 of the total frames away from the true transition point on average. Fig. 13 shows the distribution of changepoint errors for a test set

	Changepoint MAE (Frames)
Pix2D	5.802
LSTM	6.393
Cross Transformer	5.02

Table 5: Changepoint Errors for the Isotropic Case

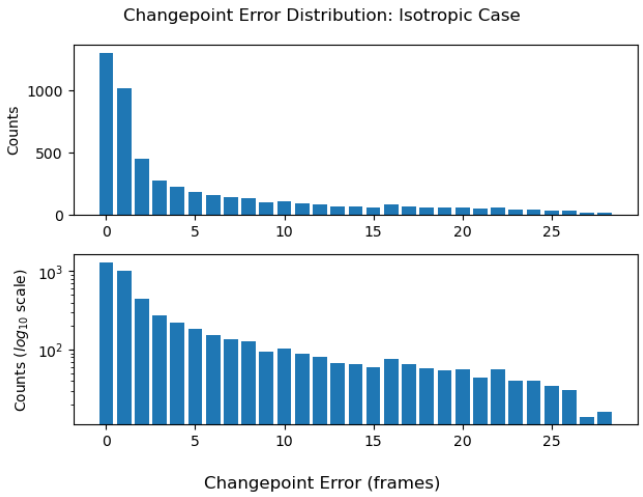

Figure 13: Changepoint Error Distribution for the Isotropic Case

of 5000 datapoints using the cross-transformer model for inference. Contrary to the binding case, the count distribution has a clear right skew. Nearly 50% of the prediction errors fall between zero and one frames. Appendices C and D display similar error distributions for the baseline models.

6 Discussion

We saw for the anisotropic case that the base transformer and the cross-transformer achieved very similar performance for the diffusion coefficient predictions, but the cross-transformer was better at predicting the orientation of the diffusive state. This shows that the displacement features likely allow the model to improve its angle predictions. Notably, each of these transformer-based models were trained on roughly 300k inputs, while the cross-transformer has over 400k more parameters. Due to limitations of computing resources, we were limited to a little over 325k inputs in memory. If training was altered to recycle memory or a different computing system was used, it is likely that the cross-transformer would have been able to improve further for both the diffusion coefficient predictions and the angle prediction. It is also possible that the improved angle prediction is simply due to the difference in parameters.

6.1 Loss Term Discrepancy

We saw in 5.1 that the MSE loss was worse for the LSTM model than the Pix2D model, but the MAE for the diffusion coefficients was much better. However, the angle prediction was slightly better for Pix2D. This shows that the angle terms in the loss function are much larger than the diffusion coefficient terms. The corresponding gradients of the angle loss terms are likely to dominate the optimization process. As a result, parameter updates will primarily favor improvements in angle prediction, potentially hurting the diffusion coefficients' accuracy. If accurate estimation of the diffusion coefficients is more important, then it may be necessary to rebalance the loss terms via explicit loss weighting or normalization of the angle errors. For example, we could alter the loss function such that $\mathcal{L} = \mathcal{L}_{\mathbf{D}} + \lambda\mathcal{L}_{\theta}$ with $\lambda < 1$ to scale down the loss term for the angle.

6.2 Changepoint Error

As was shown in Fig. 11, the binding case performed strongly, mainly predicting correctly or one frame off. However, there were some predictions with a changepoint error of over 20 frames. This behavior seemed odd considering there are no mistakes in the middle region, between five and 20 frames. Upon manual inspection, there is a clear reason why the distribution looks this way. Fig. 14 displays a typical example of a prediction that is off by 20 or more frames. These large errors are found only under two scenarios: when there is an early or late transition of states.

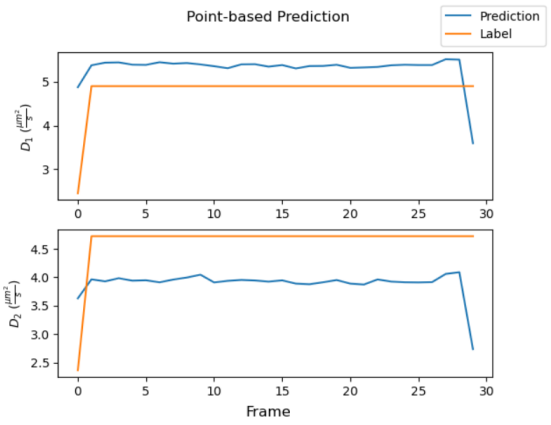

Figure 14: Binding Example of Large Changepoint Error

The example shown displays an early change of state. The model seems to capture a slight jump in the diffusion coefficients at the beginning, but at the end of the sequence a large drop is detected. This points to possible overfitting because the model seemingly learned that a drop must occur if there wasn't a prior jump. The prediction creates an artificial binding state that causes the changepoint prediction to be large. This in itself is an edge case because the model typically captures early/drop changes well, as shown in Fig. 15. Further visual examples of binding predictions and poor changepoint predictions can be found in App. A.

As was shown in Fig. 13 and the MAE, the isotropic case performed much worse than the binding case for changepoint detection. To understand why, it is helpful to visualize some of the predictions. Fig. 16 displays an example with poor changepoint prediction even though the MAE of diffusion coefficients is slightly above average. The model is not able to capture any notion of a transition between states, which results in nonsensical predictions for the changepoint. This explains why the error distribution has a long tail. Examples like this are quite common in the test set. The common pattern amongst these examples is a small change in magnitude of the diffusion coefficient. The model struggles to differentiate diffusion coefficients of similar magnitude. We did not see cases like this in the binding test set because the change in magnitude was typically over 2 $\frac{\mu m^2}{s}$. Upon manual inspection, the isotropic model typically performs well for changes over 1 $\frac{\mu m^2}{s}$. A common example near this threshold is shown in Fig. 17. The model isn't able to capture the sharp drop of the transition and instead produces a gradual change in state. The cost function for computing the changepoint allows decent performance for examples like this. Further visual examples of isotropic predictions and poor changepoint predictions can be found in App. B.

6.3 Baseline Improvements

The LSTM model likely performs worse than the cross-transformer because it's ability to capture local and global context is not as expressive as the cross attention and self attention mechanisms used in the cross-transformer. A bidirectional LSTM was tested and showed no improvements for the anisotropic case, but using one for the two-state scenarios may have improved results due to the significance of the temporal aspect in the data. Pix2D performs the worst for the two-state scenarios, likely due to the lack of a temporal understanding in the model. By altering the model to convolve over the temporal dimension, we would expect to see slight improvements. Alternatively, we could use the trained single-state anisotropic model and apply a sliding window approach to make predictions. This involves partitioning a video into windows of frames, then using each window as a separate prediction. Pix2D works best with larger sequences due to the stochastic nature of the diffusion process and the averaging approach of the model. For the sliding window method, with smaller windows, you add more noise to your predictions, but with larger windows, you likely include frames from each state in your window. Windows that overlap the two states will result in poorer predictions because of the averaging approach in the model. This method is practi-

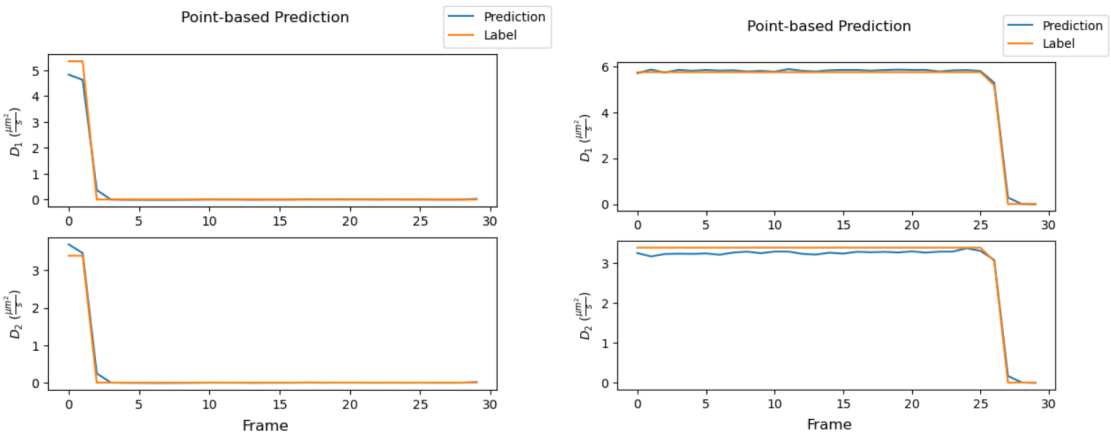

Figure 15: Early and late transitions for binding examples

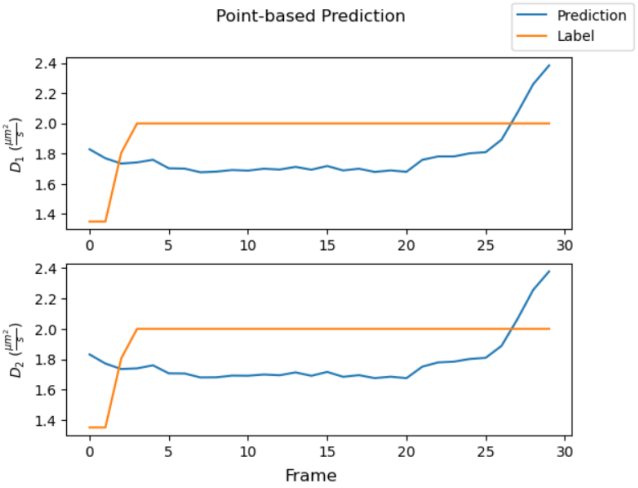

Figure 16: Isotropic Example of Changepoint Error

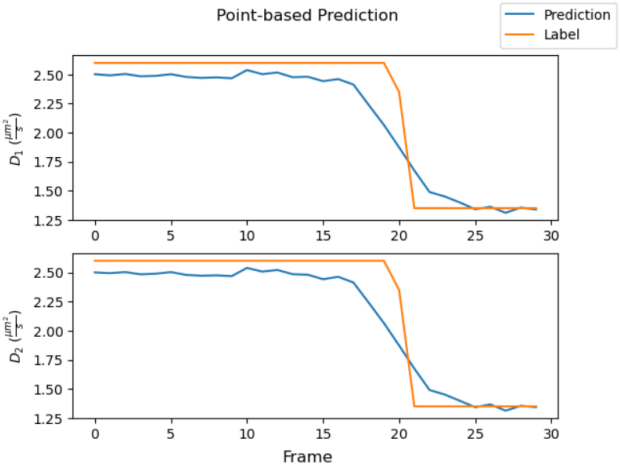

Figure 17: Isotropic Transition Example

cal since no training specifically for the task is required, but you must empirically choose the window size that performs best.

By a quick experiment with a window size of 10 frames (1/3 of the sequence), we achieved better performance in both two-state cases for the MAE of diffusion coefficients, see Tables 6 and 7. However, we lost considerable angle prediction accuracy. This is likely due to setup differences between the normal anisotropic case and the binding case. For binding, we set the angle to zero degrees when the molecule is bound. In the normal simulation, the angle is assigned randomly. Our anisotropic model did not learn to associate small diffusion magnitude with zero degree orientation, so angle prediction performance suffers when we try to apply the pretrained model to the binding case. Applying to the isotropic case, we see boosts in MAE performance, but we lose the notion of isotropic diffusion captured in the original predictions. Even with these improvements, our proposed model still performs significantly better.

	MAE $\mathbf{D_1}$ $(\frac{\mu m^2}{s})$	**MAE** $\mathbf{D_2}$ $(\frac{\mu m^2}{s})$	**Angle Similarity**
Pointwise	0.578	0.556	0.87
Window-based	0.483	0.384	0.753

Table 6: Pix2D Window-based Binding Results

	MAE $\mathbf{D}_1$ ($\frac{\mu m^2}{s}$)	MAE $\mathbf{D}_2$ ($\frac{\mu m^2}{s}$)
Pointwise	0.97	0.97
Window-based	0.624	0.78

Table 7: Pix2D Window-based Isotropic Results

6.4 Future Work

There are many areas of improvement to work towards. A major limitation of this work is the lack of testing on experimental data. It is difficult to tell whether our simulations are accurately modeling real data acquisition. Since our model was trained on fixed imaging settings, any real data obtained would need to match our setup for a proper comparison. Additionally, this shows that our dataset is possibly too restricted. Unless experimental data is captured under the same assumptions as our model, the model will not transfer well to that dataset. For diverse applications, we would need to create a broad dataset representing different imaging setups. This will make the task significantly more difficult, leading to the need for much larger datasets and models.

Each model we trained is limited to a specific type of diffusion. In reality, we would prefer a model that works well for all types of diffusion studied. Unless we have a strong understanding of the diffusive processes expected in the data, we should not apply a model trained only on two-state binding/unbinding data to it. Our two-state simulations were limited to binding/unbinding and isotropic cases. The first step in expanding to broader applications could be to model each state as anisotropic. We started to work on this but encountered difficulties during training. Along the same lines, the models only saw two-state diffusion, when in reality different numbers of states could occur. The simulation code could be expanded to include more states, but this will lead to significantly more complex training.

We saw that the performance of the model has an inverse relationship with background noise. It would likely be beneficial to research image processing techniques to reduce noise in the image data. The same relationship was found with the PSF size, but it would be nearly impossible to process the images such that the PSF of the imaging device appears smaller than it really is. Lastly, the baselines, specifically Pix2D, should be updated to better handle the pointwise prediction tasks.

7 Conclusion

We have introduced transformer-based models that outperform the baseline models for all tasks studied. For the simplest case of anisotropic diffusion, both transformer models outperform the baselines. We showed that including displacement features improved angle prediction accuracy. We think increasing the training set size will allow this larger model to also improve upon diffusion coefficient predictions. For the two versions of two-state diffusion, our model's performance excelled compared to the baselines'. For the binding case, we showed that our model was capable of detecting the transition point between states within less than half a frame on average. However, our models struggle to differentiate small changes in magnitude, as shown in the isotropic case. Future work will further improve the performance and transferability of our framework.

References

[1] Naor Granik, Lucien E. Weiss, Elias Nehme, Maayan Levin, Michael Chein, Eran Perlson, Yael Roichman, and Yoav Shechtman. Single-particle diffusion characterization by deep learning. *Biophysical Journal*, 117(2):185–192, 2019.

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

[3] Dezhong Li, Qiujin Yao, and Zihan Huang. Wavenet-based deep neural networks for the characterization of anomalous diffusion (wadnet). *Journal of Physics A: Mathematical and Theoretical*, 54(40):404003, September 2021.

[4] Gorka Muñoz-Gil, Giovanni Volpe, Miguel Angel Garcia-March, Erez Aghion, Aykut Argun, Chang Beom Hong, Tom Bland, Stefano Bo, J. Alberto Conejero, Nicolás Firbas, Òscar Garibo i Orts, Alessia Gentili, Zihan Huang, Jae-Hyung Jeon, Hélène Kabbech, Yeongjin Kim, Patrycja Kowalek, Diego Krapf, Hanna Loch-Olszewska, Michael A. Lomholt, Jean-Baptiste Masson, Philipp G. Meyer, Seongyu Park, Borja Requena, Ihor Smal, Taegeun Song, Janusz Szwabiński, Samudrajit Thapa, Hippolyte Verdier, Giorgio Volpe, Artur Widera, Maciej Lewenstein, Ralf Metzler, and Carlo Manzo. Objective comparison of methods to decode anomalous diffusion. *Nature Communications*, 12(1), October 2021.

[5] Moon Park, Wang. Machine-learning-powered extraction of molecular diffusivity from single-molecule images for super-resolution mapping. *Communications Biology*, 2023.

[6] Borja Requena, Sergi Masó-Orriols, Joan Bertran, Maciej Lewenstein, Carlo Manzo, and Gorka Muñoz-Gil. Inferring pointwise diffusion properties of single trajectories with deep learning. *Biophysical Journal*, 122(22):4360–4369, 2023.

[7] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.

[8] Peng Xu, Xiatian Zhu, and David A. Clifton. Multimodal learning with transformers: A survey, 2023.

A Binding Case Examples

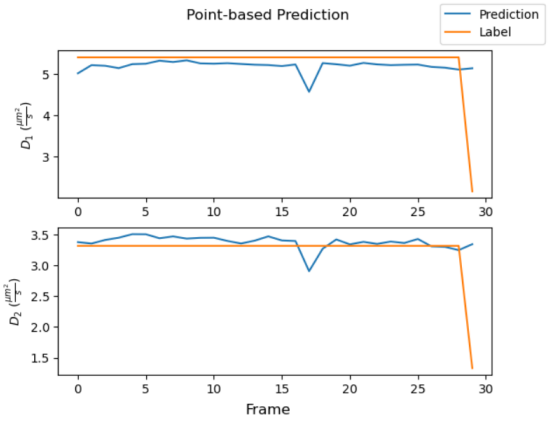

Figure 18: Example not capturing the binding state

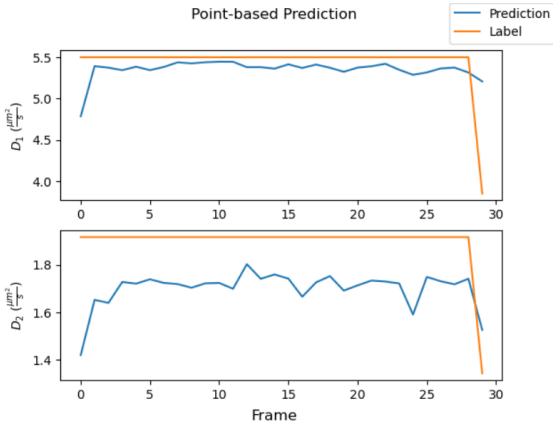

Figure 19: Example of false detection of unbinding at first frame

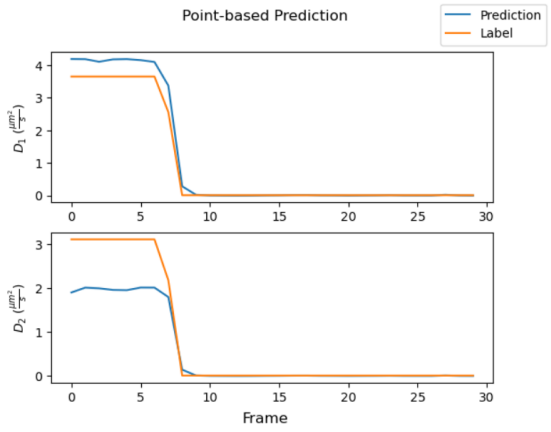

Figure 20: Example of correct changepoint detection but poor diffusion coefficient MAE

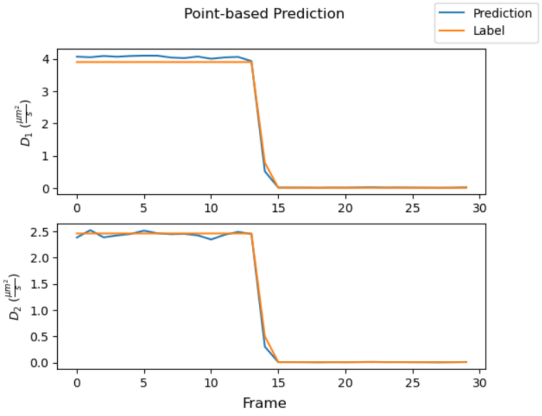

Figure 21: Example of good performance

B Isotropic Case Examples

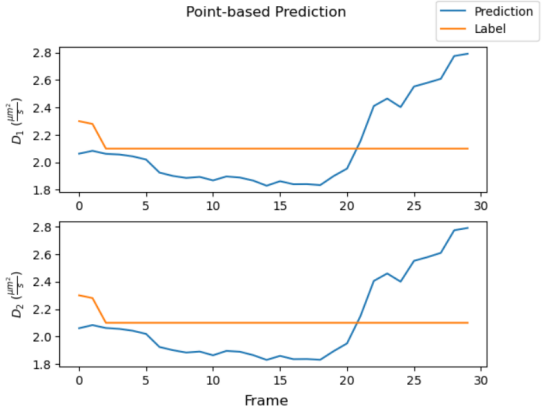

Figure 22: Example of small magnitude change not being captured

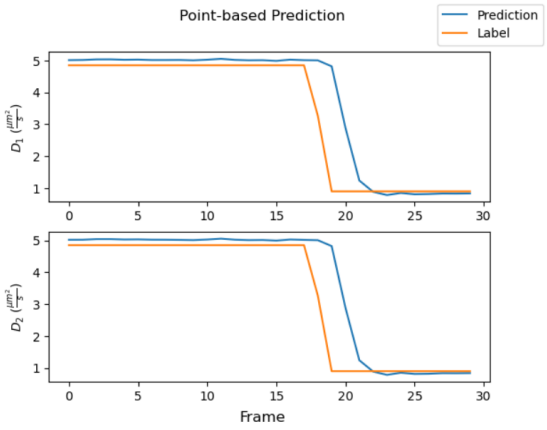

Figure 23: Example of good diffusion coefficient MAE with inaccuracy in changepoint detection

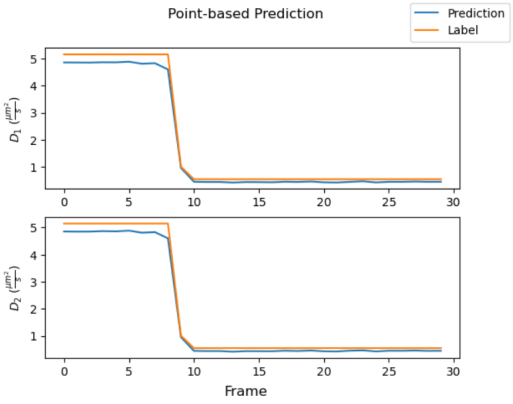

Figure 24: Example of good performance

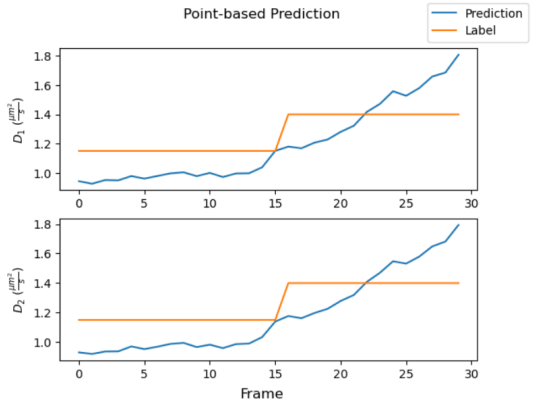

Figure 25: Example of small magnitude change with gradual increase of diffusion coefficient predictions

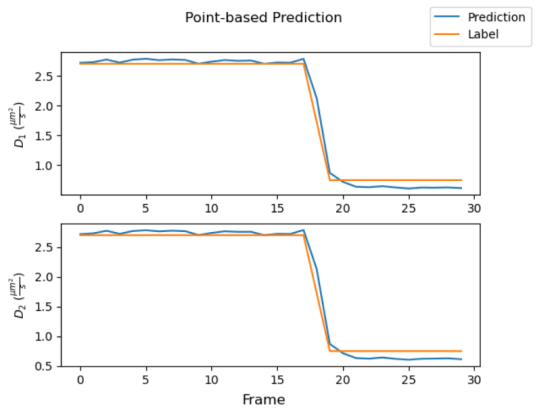

Figure 26: Example of good performance for small magnitude change

C LSTM Figures

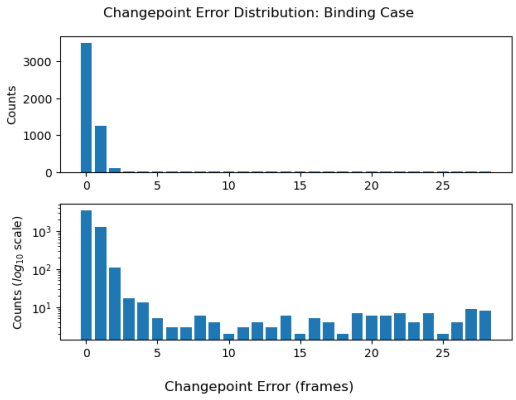

Figure 27: Error distribution for the binding case with LSTM inference

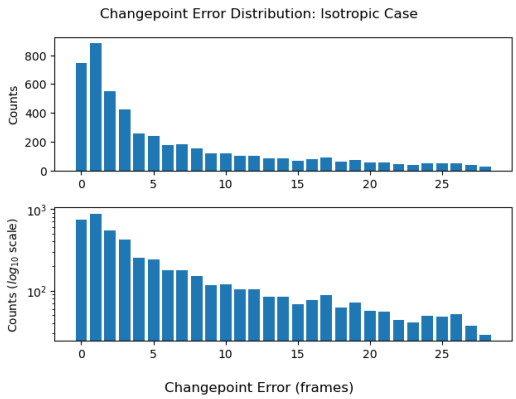

Figure 28: Error distribution for the isotropic case with LSTM inference

D Pix2D Figures

Figure 29: Error distribution for the binding case with Pix2D inference

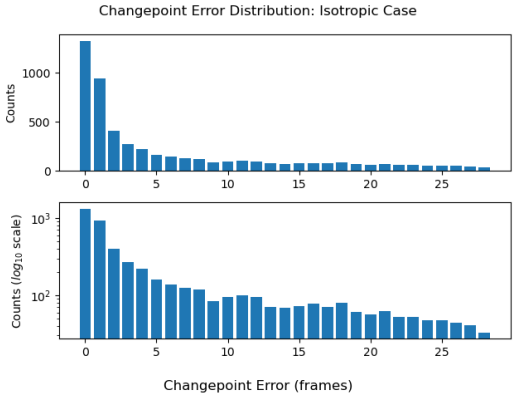

Figure 30: Error distribution for the isotropic case with Pix2D inference