Machine Learning ML4Science Project

Building a deep learning model to track molecules diffusion coefficient

Emilien Silly
emilien.silly@epfl.ch

Mathis Solal Krause
mathis.krause@epfl.ch

Anoush Azar-Pey
anoush.azar-pey@epfl.ch

Supervision: EPFL Center for Imaging, Daniel Sage and Thanh-An Pham

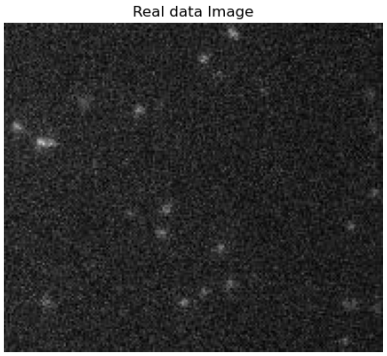
Abstract

Single-Molecule Localization Microscopy (Nobel Prize 2014) is a powerful super-resolution technique for imaging live cell compartments with a resolution of up to 15 nm. In these images, a single moving molecule appears as a bright spot, primarily due to motion blur. By analyzing the shape of this blob, we can gain insights into cellular dynamics, specifically by estimating the diffusion coefficient (D) of the molecule. The goal of this project is to train a deep learning model to predict the diffusion coefficient (D) from a sequence of simulated images. These simulations can faithfully mimic real conditions, thanks to a well-known physical model that incorporates the Brownian motion of the molecule, the microscopy point-spread function, and the noise, allowing us to generate large datasets. We will investigate different neural network architectures to determine which one is most effective for estimating the diffusion coefficient from the sequence of images. Ultimately, we aim to replicate microscopy experimental conditions to apply this model to real images. The actual baseline of the biologists to estimate D, use centroids through the frames to calculate the displacement.

1 Introduction

The diffusion coefficient (D), which quantifies the rate of molecular diffusion, is an important metric in microbiology as it provides insights into molecular dynamics, cellular environments, and the interactions that govern vital biological processes.[2]

Predicting this diffusion coefficient in real-life scenarios is extremely difficult [3] because of the multiple biological factors that affect this coefficient, and the physical limitations in observing the particles at the scale of a few nano-meters. Here is an example of an image containing multiple particles captured by a microscope.



Real images are extremely noisy, due to general Gaussian noise to which is added Poisson noise. In their underlying process, particles follow a brownian motion with a speed dictated by their diffusion coefficient. During their continuous movement, the particles emit light, which is then captured periodically by the microscope depending on its frame rate. Single particles appear brighter if they remain static, or dull if they have a high diffusion coefficient, because the light they emit is spread out around their positions. Finally during the observation, the results are convoluted with the microscope's point spread function (PSF), which creates a blob around the cell.

There are 3 cues to identify a particle's diffusion coefficient.

- The overall trajectory gives a broad estimate of the particles diffusion, and is currently used by microbiologists to deduce the value of D. It is computed by taking the particle's average position over multiple frames, but is known to be imprecise.

- The shape of the blob encodes information about how much the particle moved during a frame, which is linked to the diffusion coefficient. This paper explores the prediction of D with machine learning using only the shape of the particle's blob, without taking into account the overall trajectory. [1]

- The intensity of the peak inside one frame. As stated before, the brightness of a particle is linked to its movement.

As real annotated data is practically absent, in this project we will create simulation data that mimics real data distribution, and apply a machine learning model on 16 frames of size 64 by 64 to it to try to analyze the practicality of using machine learning models that would utilize all 3 information cues.

1.1 Calculating D

Computing the real value of D is only possible given the trajectory of a particle, and is done using the MSD:

The Mean Squared Displacement (MSD) is defined as:

$$\mathrm{MSD}(\Delta t) = \langle |\mathbf{r}(t+\Delta t) - \mathbf{r}(t)|^2 \rangle \tag{1}$$

where:

- $\mathbf{r}(t)$ is the particle's position at time t,
- Δt is the time lag,
- $\langle \cdot \rangle$ denotes averaging over all particles or time steps.

For isotropic 2D diffusion:

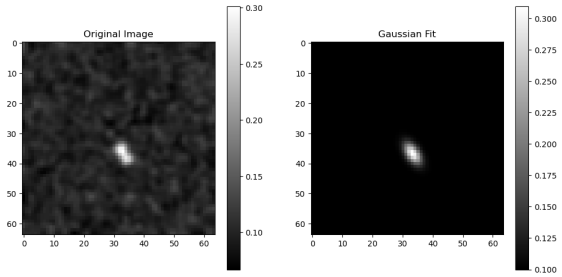
$$\mathrm{MSD}(\Delta t) = 4D\Delta t \tag{2}$$

where D is the diffusion coefficient ($n\mathrm{m}^2/\mathrm{s}$). The diffusion coefficient D can be computed by fitting $\mathrm{MSD}(\Delta t)$ to the linear model:

$$D = \frac{\text{slope of MSD}}{4} \tag{3}$$

1.2 Actual Baseline

To calculate the diffusion coefficient, biologist mainly use an estimation of D based on the centroids to find the trajectory and then to calculate D. We will call this value trajectory-D. For this it is possible to fit a Gaussian function in each frame that will find the coordinates of the center of the frame (brightest in intensity) after filtering the noise. This estimation can be poor depending on the exposure in the case where the estimated centroids do not move so much while the molecule actually does. We believe it is possible to reach this baseline and maybe to exceed it.



2 Simulations

One goal of this project was to artificially generate images of moving molecules that mimic real laboratory-taken photographs. We implemented this task in a python script that, when given some wanted Brownian coefficient D (nm^2/s), generates a trajectory of 10 (x,y) successive positions in nm per frame for each particle using the wanted coefficient `D`. However since this is a random process, the generated trajectories do not always have the expected coefficient D, but an effective value, which we compute for each generated trajectory using the MSD formula 3. Then for each trajectory, we generate 16 images of size 64 by 64 where each pixel represents 200nm. To generate an image where the sub-positions are not clipped to integer multiple of pixels, but rather on the nm scale, we first sample an image in high resolution, before downsampling it to the wanted size. This is an expensive process, and to speed up image generation we added support for multiprocessing creating a 3x speedup.

We then had to determine the optimal Gaussian and Laplace noise amount to add to our generated images. In order to do so, inside the `blur_optimization/` directory, we wrote programs to maximize similarity functions between a real and generated image, using the amounts of these two types of noises.

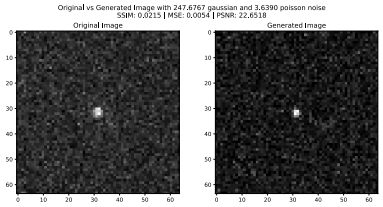


Figure 1: A real image on the left, and a generated image, on its right, given by one of the algorithm. There are, in the title, different ways of computing the difference between the two, and the optimal amount of noises it found

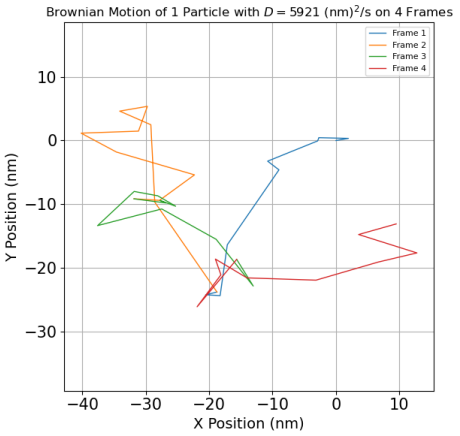


Figure 2: An example of a generated trajectory for 4 frames, with the images associated with each sub-trajectory

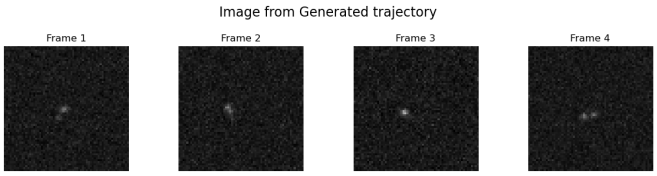


Figure 3: Set of generated images from the trajectory

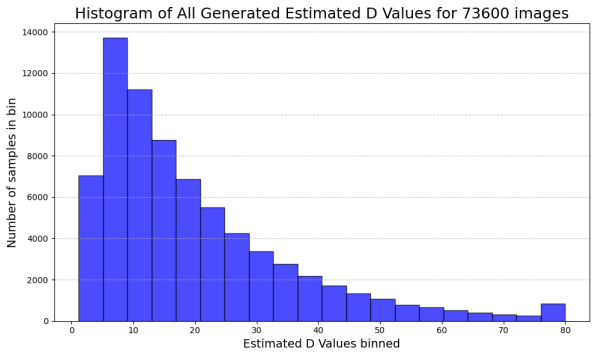
In this set of images, we can see that the trajectory of the third frame is more stationary, which leads to a brighter blob on the third frame, while the other frames are duller and have an elongated shape. Using these trajectories, we get a Centroid-D of 3200 $(nm)^2/s$, which is an underestimation of the real value, notably due to the fact that the centroids for each frame are closer to each other than the real trajectory.

3 Models and Training

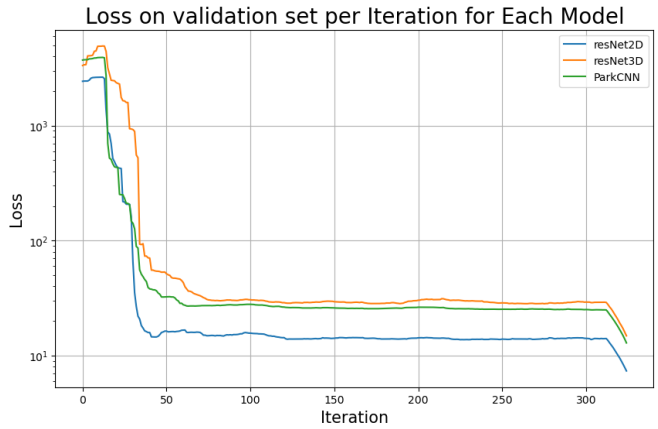
Since the task at hand is a vision task, we had to decide on a model architecture which would best fit our problem. We tried multiple different architectures, notably the one used in the Park et al. paper [1], a CNN with a self-defined Swish activation function, a ResNet doing 2D convolutions and a ResNet doing 3D, or said differently 2D + Time domain, convolutions, both using a ReLu activation function. We wanted to see if a model relying on 2D information would be able to detect the shape of the particles efficiently, or if the 3D approach following the particle over time would lead to better results. We decided to create our own smaller implementations of ResNet, to obtain similar model sizes as the Park et al. paper, all our models number of parameters is in the range $2-5$ millions.

Since we only had a very limited quantity of real data, we trained the model on simulation data only, generating new images for each epoch, because in early trainings trials we let the model train multiple epochs on the same images, which lead to rapid overfitting. Generating new images represents the major part of training time, as we need to feed it a lot of different images for the model to be

able to generalize well enough. In the training pipeline we can specify the number of epochs we want to train for and the number of images for each epoch. At each epoch we generate new images from the wanted parameter D, however since the image generation is a random process, we get images for effective D values over a certain range. Here is an example of the distribution over D values, when querying D=15000 $(nm)^2/s$: To avoid vanishing or exploding gradients, we divided the values of D in $(nm)^2/s$ by 1000, to get values more suitable for training, explaining why the values are in the range [1-80].



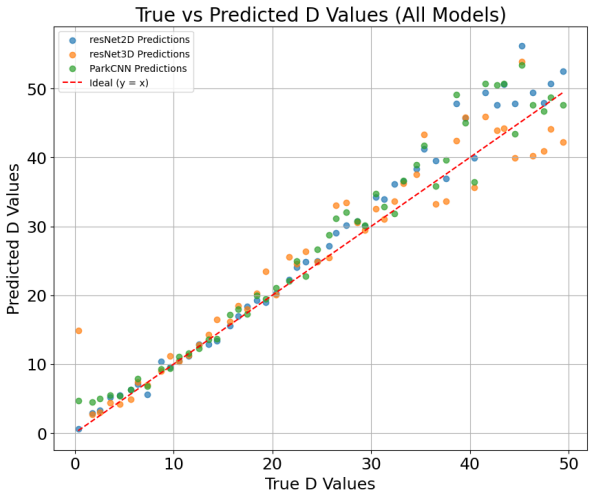
Since using the training loss is very variable, due to the fact that images for each batch can have different distributions of D, we pre-computed 5 images for each value of D in the range of interest [1-50] and after each training step queried the model's prediction on the validation images, to get the general trend of validation loss of the model. We can see that the loss decreases rapidly, and reaches a plateau after around 60 iterations.



All supplementary training seems not to be helping the performance anymore. This could be due to many factors. The primary one is probably the fact that the relatively small architectures, compared to real-life models cannot utilize more information, or the fact that the images are very noisy for the most part, and only contain a small patch of information for each frame. Using small models as we did also induces a much smaller training time compared to bigger and more complex models. In only an hour on modern GPUs, a model could be trained on a dataset generated on the fly mimicking real life conditions.

4 Results

After training of the algorithm, we plot it's predicted values compared to the ground truth of validation images. Here taking 5 real images for each D value in range 1-50, we can see that the algorithm is very good for small values of D, however as the values go higher the models tend to be less accurate, not following perfectly the real values linearly. This is not too surprising as our dataset was composed of more images with smaller D values during training, and is thus more capable of estimating these images accurately.

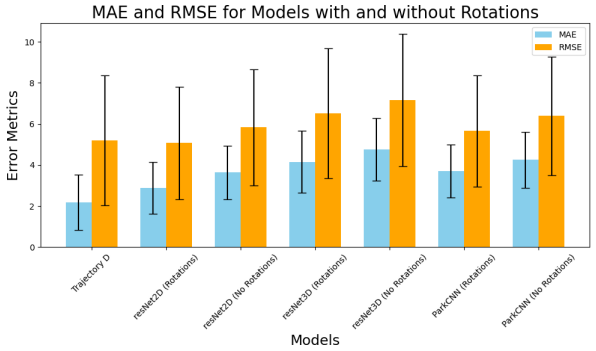


Another point of interest for us, was to compare our models to the methods currently used: the Trajectory-based coefficient. This coefficient only uses the trajectory over multiple frames, to compute a general MSD and Diffusion coefficient D. We hoped our models can also use the shape of the particle in each frame and the intensity of the image peak to outperform the standard method. In the next figure are the results of our models compared to a real approach based on the trajectory-D.

We can see that even though our model is close to the trajectory-D, it does not outperform it. Comparing the RMSE and MAE, we can see that our predictions are comparable depending on the model, but not as accurate as the real trajectory-D.

Finally we tried to understand what part of the in-

put images were learned, notably the 3 discriminative elements of the image: The blob shape, the trajectory and the intensity of the blobs. Instead of feeding only a single image into the algorithm, we queried the model with the same image 4 times, each time rotated by 90°. Using the fact that the problem at hand is symmetric, i.e. a particle moving up or down has the same diffusion coefficient, we then averaged the predictions and noticed a small improvement in accuracy, proving that the model learned to recognize some information about the trajectory.



We also tried using our model to predict on real images of fluorescence microscopy, where the diffusion coefficient is still unknown. However since the images were acquired differently they differ from the simulated images by an unknown normalization factor. Trying out different normalization factors, we noticed that the models predict different diffusion coefficients depending on the max-value of the image. This shows that the peak intensity and global intensity of the input images are also analyzed by our model. However, predictions on real data did not work out as expected, as real data does not follow the same distribution than our simulation data exactly, with slight normalization differences producing different results. Using our models to predict on real data would only be an approximation, which would not be relevant enough for scientific usage.

5 Conclusion and openings

To conclude, we are correctly satisfied with our results on the simulation data, as we were able to approach real predictions in a short time-frame using a small training window. The novelty of our method is to be able to predict the diffusion coefficient without computing the MSD directly, but only feeding the images into a model. This proves that some information about D can be extracted from the images without processing them using the centroids. This gives us good hope that future research in this area might be able to outperform the classical method. Another positive result is the fact that our methods slightly outperforms the method used in this paper[1], which only considers the particle blobs and not the trajectory. However since the predictions did not over perform the Centroid-based computed coefficients, we know we would still need some improvements in our method. Using a bigger model with more parameters could be a solution, as well as trying out different architectures. Vision transformers for example could prove useful in this kind of setting where the number of frames for each particle is variable in real data. We could not apply other computationally expensive models to this problem given the project time-frame, but it would a good way to try to enhance performance.

On the other side, we were not able to reliable predict diffusion coefficients for real data, because of the difference in data distribution. To overcome this problem, using the property that the training time is rather low, it could be possible after analyzing the real data distribution precisely to generate simulation data that matches real data perfectly, in which case the real data could then be applied fed directly into the trained model to get ready-to-use predictions. However, we coded a code pipeline capable of creating new images based on real physical parameters, that are very close to real images. Applying this to real conditions might enable labs to generate synthetic data that resembles their lab conditions.

Finally, We would like to thanks Daniel Sage and Thanh-An Pham from the Biomedical Imaging Group for their implication and help during the whole project. We were able to learn a lot working with them and hope our project will help their laboratory in future research.

6 Ethical risk assessments

In this project, we think that tracking molecular diffusion coefficients using a machine learning model is not generally considered ethically risky because it primarily involves the analysis of physical phenomena rather than personal, sensitive, or identifiable information. Also it is difficult to know if the real data we have comes from human, animal or vegetal samples. Even if it was human, molecular diffusion is a natural process influenced by factors like viscosity, temperature, and molecular size, which are commune and impersonal. Our model trained to predict diffusion coefficients operate on simulated dataset, and do not involve human subjects or private data. As such there is not threat for personal data of subjects as there can be for heath-disease related projects. From an environmental perspective the computational cost of training is very low as everything could be done on our laptop and there is no impact on the environment.

The only thing we could think of is to ensure reproducibility for other scientists that could verify our work or assess our results. We reviewed scientific and ethical guidelines, including those related to data transparency and reproducibility, to confirm compliance and we make everything available on our GitHub.

References

[1] Park, H.H., Wang, B., Moon, S. et al. Machine-learning-powered extraction of molecular diffusivity from single-molecule images for super-resolution mapping. Commun Biol 6, 336 (2023). https://doi.org/10.1038/s42003-023-04729-x

[2] Muñoz-Gil, G., Volpe, G., Garcia-March, M.A. et al. Objective comparison of methods to decode anomalous diffusion. Nat Commun 12, 6253 (2021). https://doi.org/10.1038/s41467-021-26320-w

[3] Rafael L. Schoch, Itay Barel, Frank L. H. Brown, Gilad Haran; Lipid diffusion in the distal and proximal leaflets of supported lipid bilayer membranes studied by single particle tracking. J. Chem. Phys. 28 March 2018; 148 (12): 123333. https://doi.org/10.1063/1.5010341

GitHub repository and code availability:
Our complete repository is available and fully accessible at https://github.com/CS-433/ml-project-2-big-burger.