CS904 Assignment 2

Callum Marvell
u1606521

February 19, 2020

1 Introduction & Overview

This brief report outlines the results obtained by the code written for the CS904 assignment, focused on modelling single cell dynamics.

2 Task 1

The objective of task 1 was focused on channel gating to produce simulation runs of a calcium ion channel opening, closing and having ions pass through it.

The final code implemented here produces a visualisation run for changes within a time interval of roughly 100 seconds, and a simulation run for an interval of 100,000 seconds for analysis purposes.

The visualisation is as follows

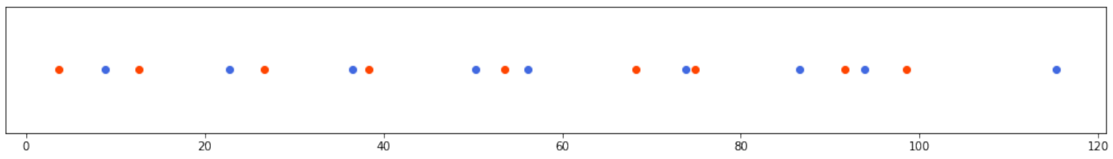
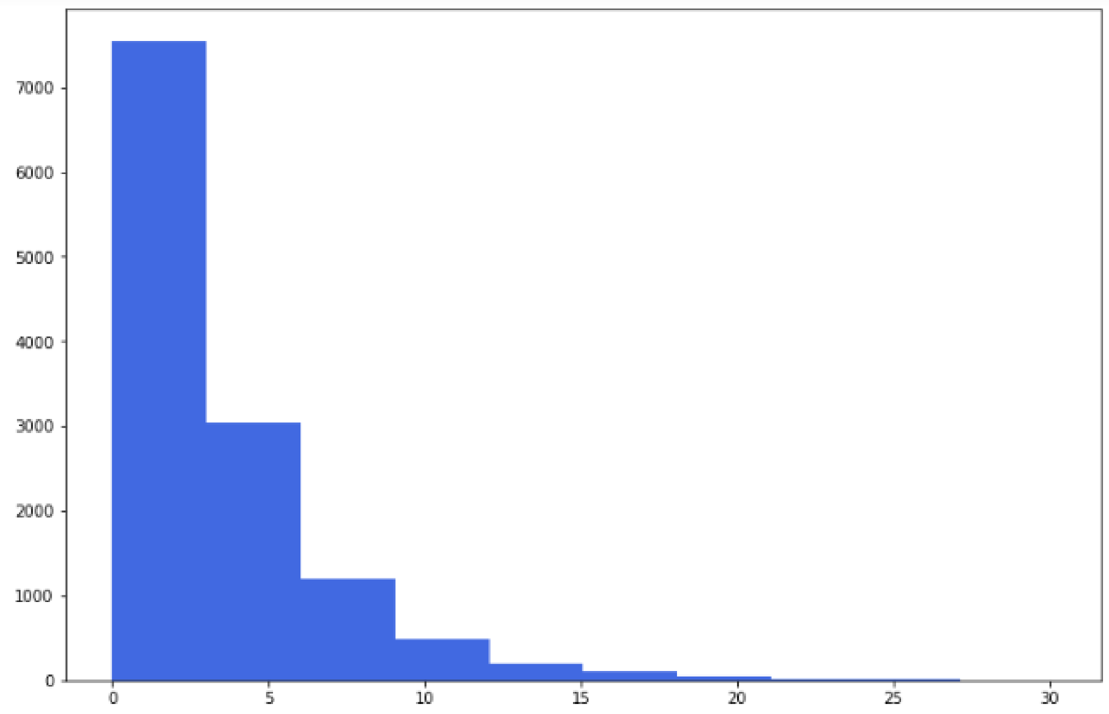


Figure 1: Visualisation of Calcium Ion Channel Behaviour

The average open and closed times for the channel over the course of the 100,000 time length interval are 3.31 and 9.93 seconds respectively. These are reasonably good/accurate results well within reasonable bounds - since expected results according to the model should be 3.3333 and 10 respectively

The following histogram further details these open and closed times:



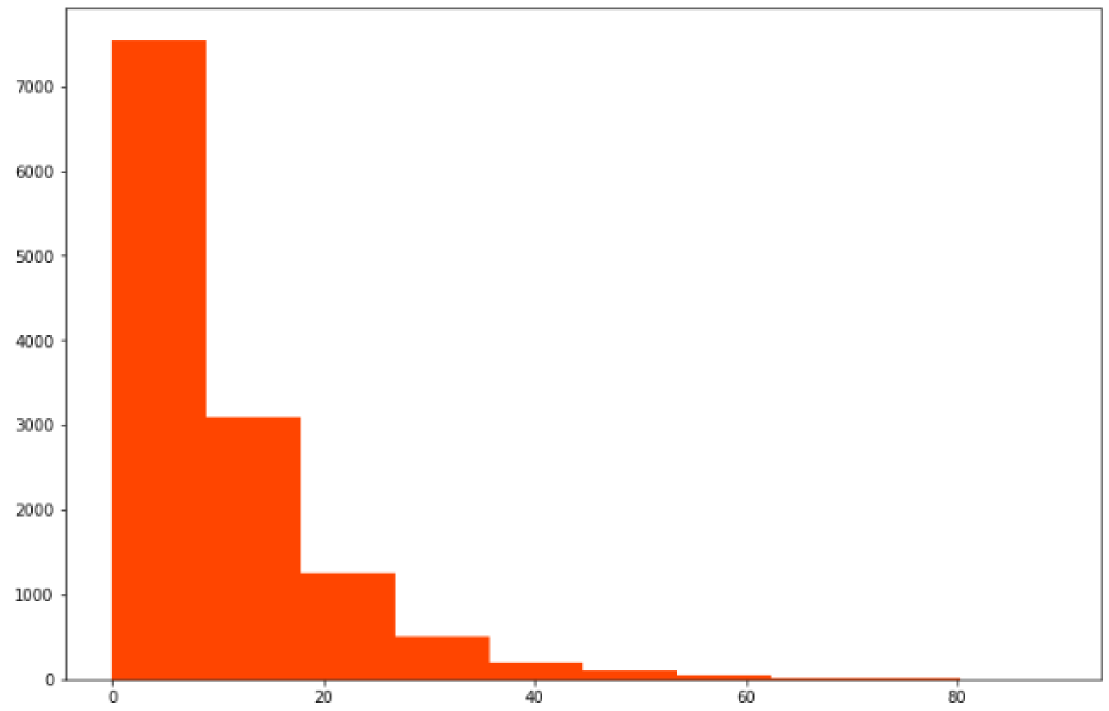


Figure 2: Histograms showing Open (blue) and Closed (orange) times

These differ to the predicted histograms for each, which would be representative of every open session being $1/k^-$ timesteps long, and every closed session being $1/k^+$ timesteps long. Under this cumulative histogram model, that would result in straight lines at 100% of datapoints (7555 in this case) from 0-3.333 and 0-10 for open/closed respectively.

3 Task 2

This task follows directly on from the previous one, focusing on visualising how the number of ions within the channel will increase over time. A simulation run was created where as much time as was necessary to increase the number of ions moved across the channel was used, and a line plot was created showing the times at which the channel was open/passing ions/closed respectively. The image below details this, with red line segments indicating periods where the channel is closed, blue segments indicating where the channel is open and green segments indicating periods in which the channel is open and ions are actively passing through it.

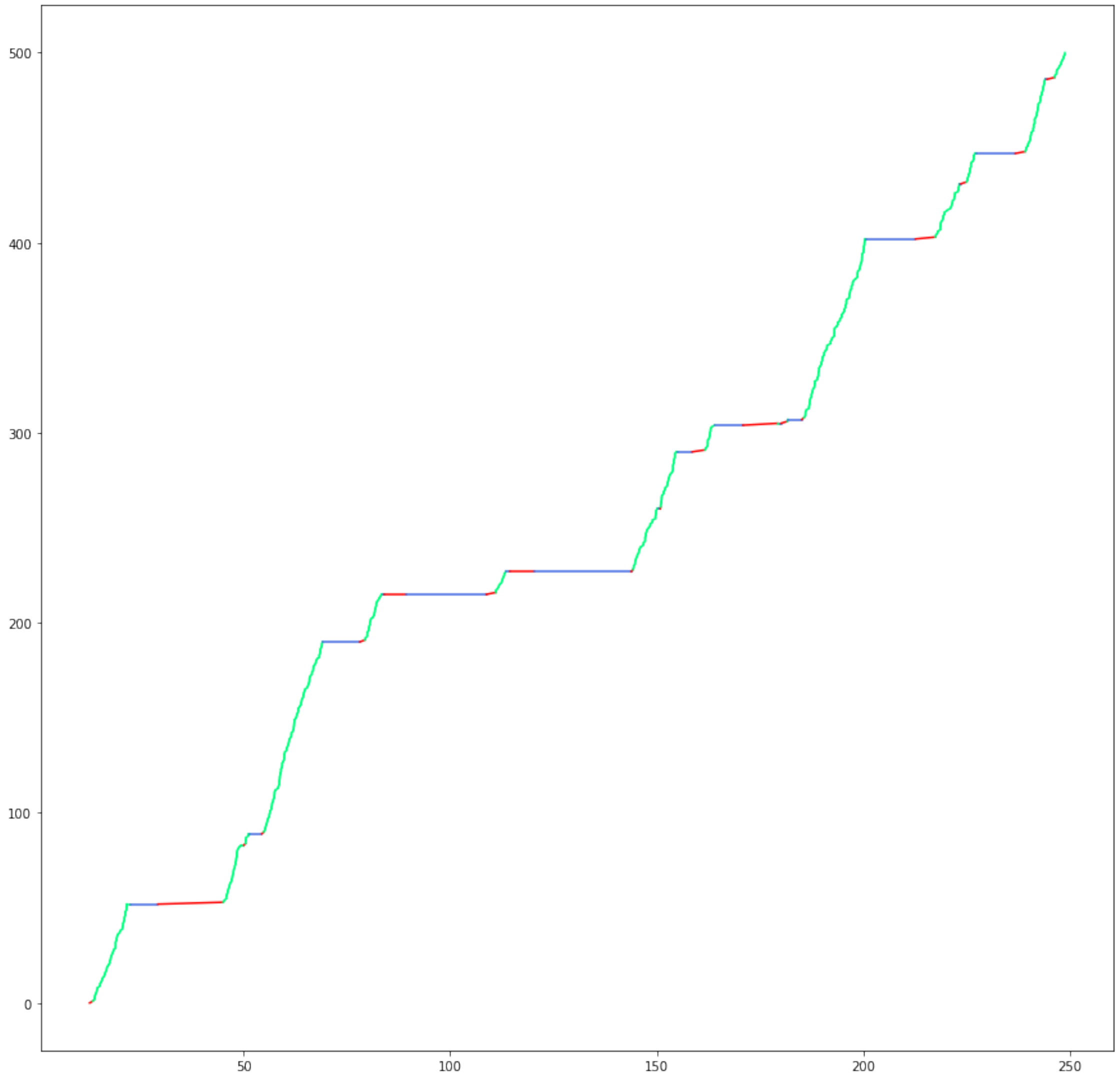


Figure 3: Plot showing change in number of intracellular ions

4 Task 3

This task follows on from the previous two, this time focusing on showing the way in which the intracellular ions move/diffuse after having been released. Initially, this was simply kept to an infinite halfplane diffusion, seen in the below image:

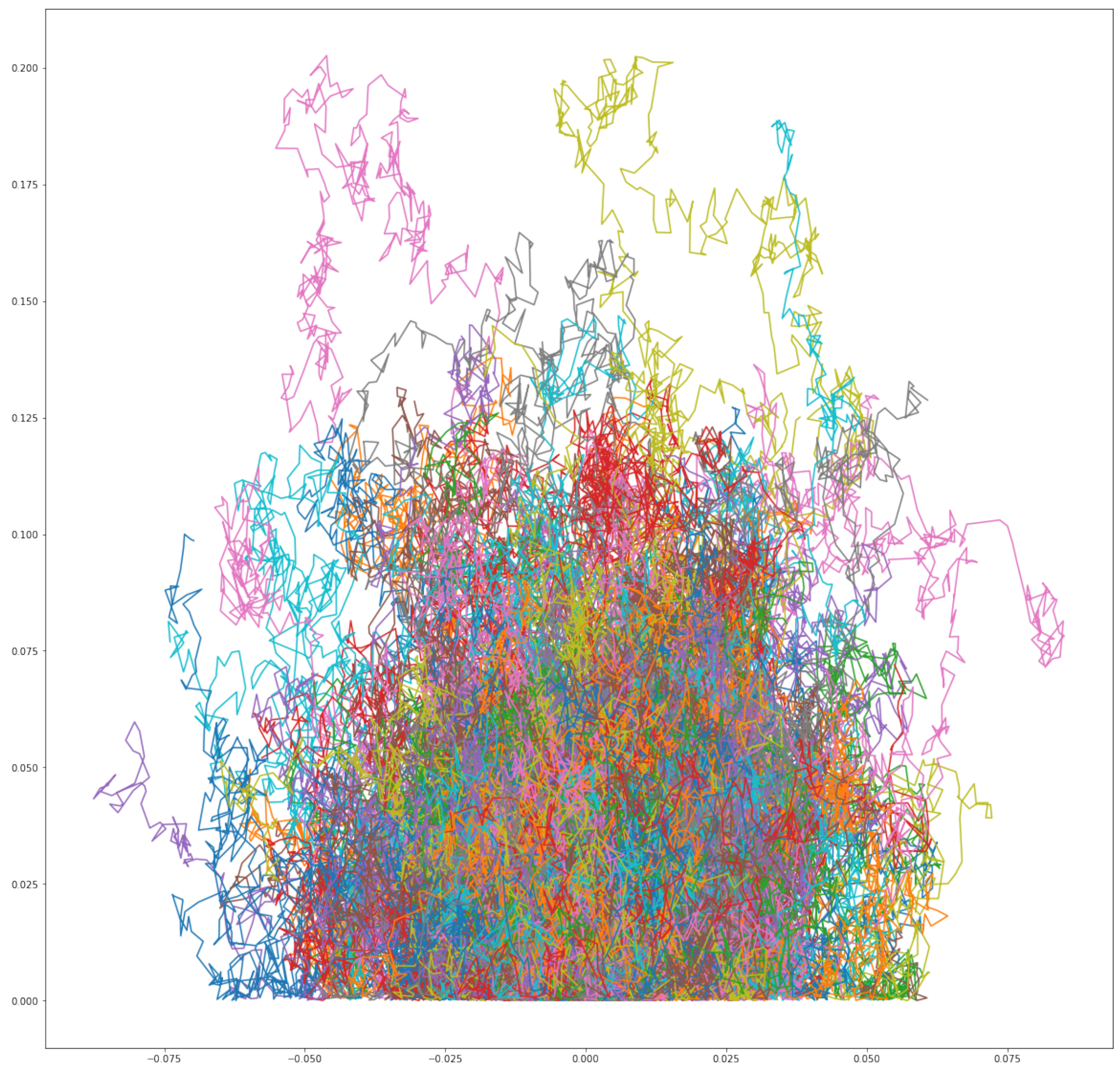


Figure 4: Halfplane diffusion pattern for 500 intracellular ions released

Following this, the situation was changed to keep ion motion/random walks bounded within a small box - thus modelling the subdiffusion process. The values provided in the lab sheet were too small, so value ranges 5x larger for both axes were used in their place, to seemingly good effect. Any ions moving into one of the walls simply reflects off - thus keeping the motion contained to within the specified bounding box:

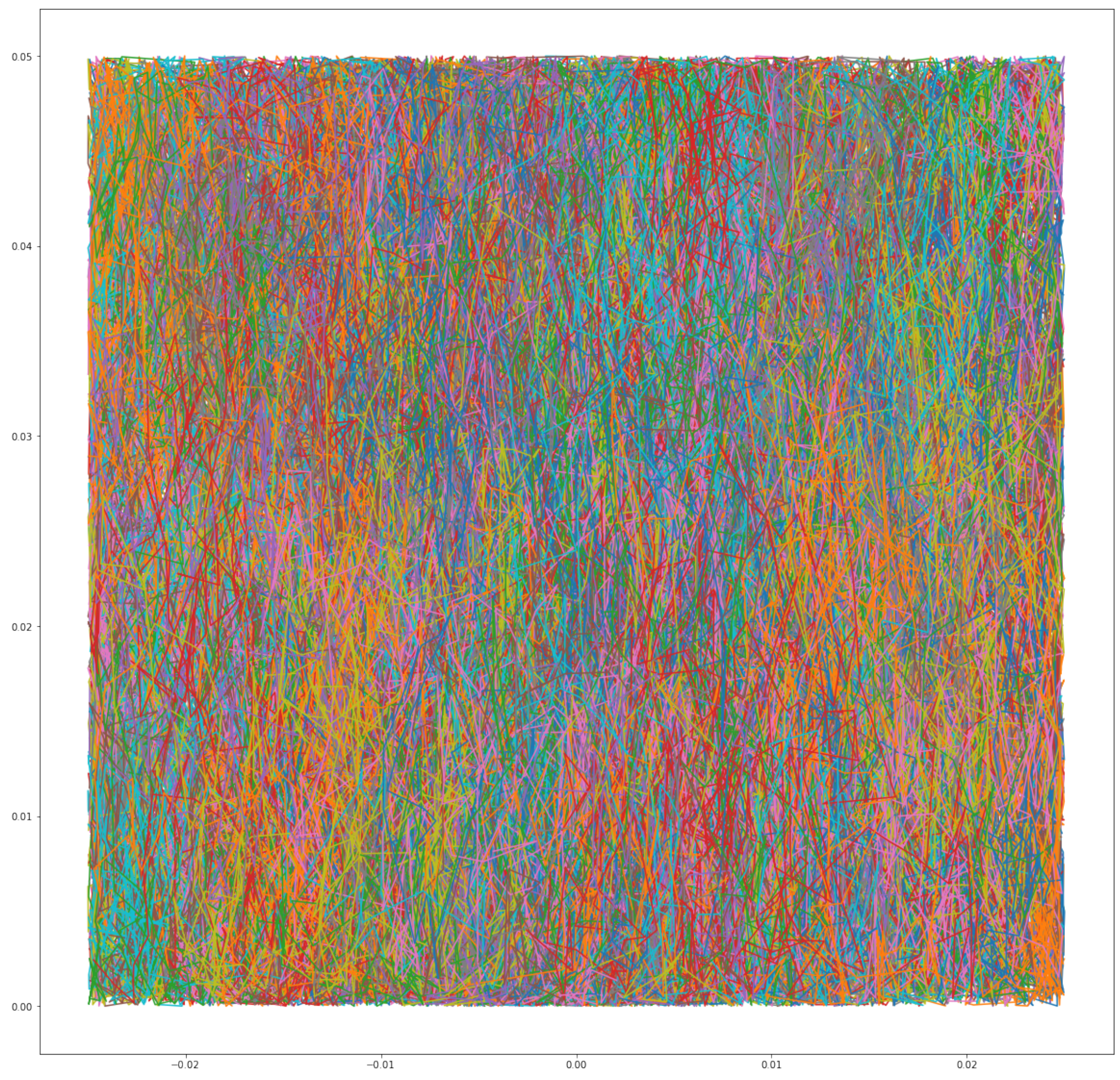


Figure 5: Bounding box diffusion pattern for 500 intracellular ions released

5 Task 4

In task 4 the aim was to build simulation model of a Kai cyanobacterial clock. The code for this can be plainly seen in the jupyter notebooks files submitted, with the end result being a set of values over a given timecourse for T,S,D,U and A.

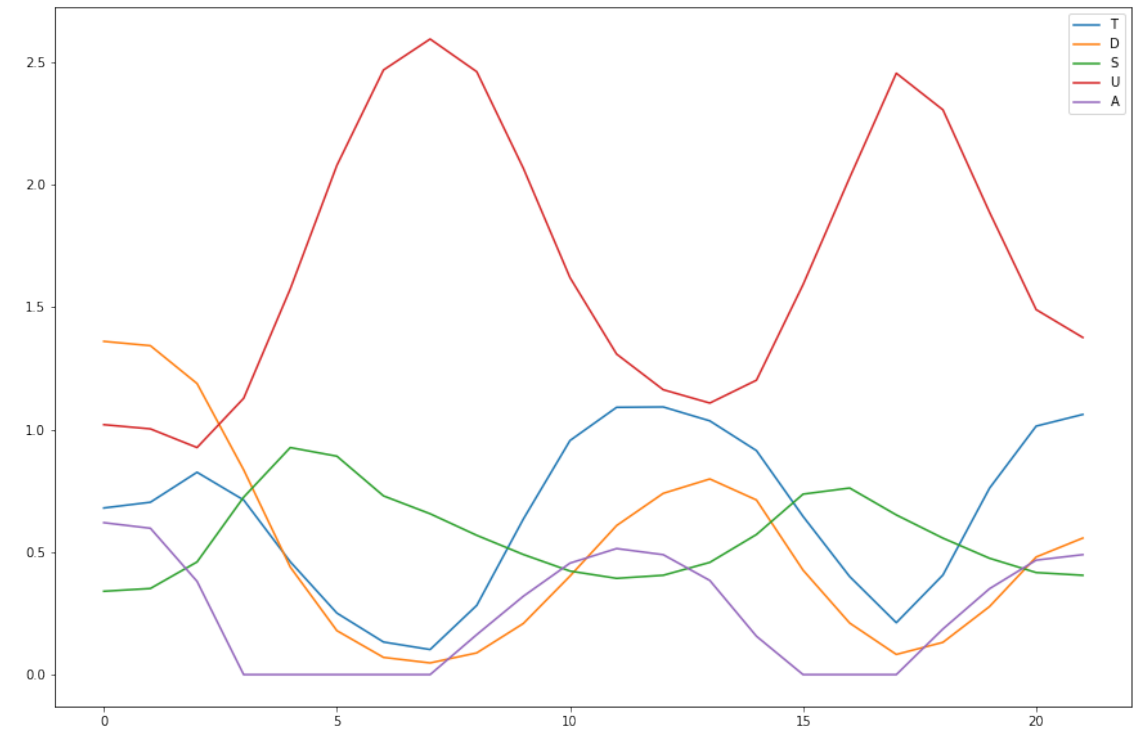


Figure 6: Timecourse data for Kai Clock Simulation

In this particular case, the python solve_ivp function only found solutions at 22 different steps, but the entire model was run for 42 hours. As can be seen from observing the last half of the graph, one complete cycle of any one line in the plot thus takes about 21 hours (or 11 ”interesting steps” according to the ivp solver).

6 Task 5

Task 5 focused on using the Hungarian algorithm (or similar) for matching purposes with regards to cells which have moved a small amount between one image frame and another/the next. The assumption is made that, between these time steps, the location of cells doesn't change much and neither does the intensity of them within the image representation.

The first step was to define a suitable cost function to solve the assignment problem of matching the cells between images. Since we can assume that cells don't move much between timesteps (and thus images) and similarly that their intensity changes little, these are good metrics to use for calculating similarity (thus using as parameters in a distance/cost function). A similarity matrix can be built between each cell in one image and each in the next, using euclidean distance between centroids multiplied by the difference in intensity between the two cells as the metric for similarity. Then it's simply a case of finding a matching between cells that minimises the sum of these values obtained. Ideally, assuming that our assumptions at the beginning regarding cells not moving or changing intensity much hold, it should be a simple case of matching each node in turn to it's closest match in the other image. Worst case a naive approach will be $O(n!)$, so as long as n remains small, the runtime of the matching algorithm will be reasonable.

The next step was to describe a strategy to use in a situation where cells are entering or leaving the field (the above approach cannot adequately cope with such situations). One such strategy (inspired by this paper (1)) that could work would be the following:

- When assigning N(i) objects in frame i to N(i+1) objects in frame i+1, we can introduce shadow objects for each of the objects in the two frames. We then match as normal, with a few caveats.

- If an object in frame i is matched with it's shadow object in frame i+1, then it can die (as it has evidently left the field).
- If an object in frame i+1 is matched with it's shadow object in frame i, then it can be born (as it has evidently entered the field).

By using these rules and regulations for cases where objects enter or exit the field we can ensure behaviour remains consistent and accurate.

7 Conclusion

All of the coursework tasks were completed and this report outlines the results obtained therein. Full codebases for the problems tackled in tasks 1,2 and 3 can be found in Lab4_CellDyn.ipynb, task 4 can be found in Lab5_KaiClock.ipynb and an implementation for the solution discussed in part 5 can be found in Lab6_Hungarian.ipynb. Checkpoints have been included for these notebooks that should still have the correct images, values, etc. loaded, but failing that execution of all cells in order should lead to comparable results.

References

[1] E. Moen, E. Borba, G. Miller, M. Schwartz, D. Bannon, N. Koe, I. Camplisson, D. Kyme, C. Pavelchek, T. Price, T. Kudo, E. Pao, W. Graf, and D. Van Valen, "Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning," *bioRxiv*, 2019.