Project 2: Where will Dicty Meet

Xinrong Dong

Approach description

The goal of this project is to predict where Dicty cells will eventually aggregate from only the early frames of a time-lapse movie. Each movie shows migrating cells over time in a single fluorescence channel. We want to know how many consecutive frames we need to predict the future aggregation center. We build three models.

1. Model 1

The previous frames K =4, we will predict the full image at time t+1.

2. Model 2:

The previous frames K =8, we will also predict the full image at time t+1.

3. Model 3:

The previous frames K =8, we will predict a pixel-wise velocity map v = frame(t+1) – frame(t), and reconstruct the next frame as frame(t) + v.

So, by comparing these models in terms of MSE and aggregation center error, we can quantify the model performance to evaluate the assistance of time information and speed modeling for predicting the aggregation location.

Methods (GitHub link: https://github.com/xr0706/5243-Project-2)

1. *Data description*

We worked with the three time-lapse fluorescence movies, and each movie captures a single field of view in which Dictyostelium discoideum cells migrate over time and aggregate into a bright hotspot. The raw data are stored as 5-D Zarr arrays with layout (T, C, Z, H, W), with T = number of time points, *C* = imaging channel, *Z* = the axial index, and *(H, W)* = the spatial dimensions in the imaging plane. To be more specifically, movie A = (34, 1, 32, 256, 256), movie B = (40, 1, 16, 256, 256), movie C = (20, 1, 48, 256, 256). All movies have a single Far-Red fluorescence channel and in-plane size is 256×256 pixels, but they are different in the number of time points and z-slices.

2. *Preprocessing*

All three Dictyostelium movies were stored as Zarr volumes on Google Drive and loaded directly in Colab. For each file we opened the Zarr group and converted it to a NumPy array. To focus on a single imaging plane and keep the prediction task 2-dimensional, we selected one representative z-slice for each movie. All experiments were analyzed using the same fluorescence channel, so the input to the models is a grayscale movie. After choosing a representative z-slice, each Dicty movie contains only the Far-Red

fluorescence channel and can be represented as a 4-D tensor of size (T, 1, 256, 256), where 1 means the single channel. We sometimes denote also this as (T, 256, 256) by omitting the singleton channel dimension, but both notations refer to the same 2D time-series video.

Next, we performed normalization for each movie to make signals from different experiments comparable and stabilize optimization. Each movie was cast to float32 and rescaled to the [0,1] range by using a simple transform $I_{norm} = \frac{I - \min(I)}{\max(I) - \min(I) + 10^{-8}}$. This helps to correct for global brightness differences across acquisitions while preserving the relative contrast and fine structures within each frame. After normalization, the printed summaries confirm that all movies have minimum intensity ≈ 0 and maximum ≈ 1. To avoid blurring subtle cell aggregation patterns, we chose to work directly with the normalized frames rather than applying extra smoothing.

Finally, we converted the continuous movies into supervised learning samples using a sliding temporal window. We slide a window of length K over each movie and, for every starting time t_0, constructs an input target pair with x = $\{I_{t0}, \dots, I_{t0+K-1}\}$ and y = I_{t0+K}. This yields 82 windows for $K = 4$ and 70 windows for $K = 8$, which are then split into train, validation, and test sets with approximately 70%, 15%, and 15%.

3. *Models*

All three models are implemented in PyTorch as compact 3D convolutional neural networks. Each model takes a sliding window of K past frames as input and outputs a single 2D map at time t + 1. To be more specifically, we use TinyCNN, and the encoder applies a single Conv3d layer with kernel size (K, 3, 3), 1 input channel and 8 output channels, followed by a ReLU nonlinearity. This layer integrates information across both time and local space. Its output is squeezed along the temporal dimension to produce an 8-channel 2D feature map.

For Model 1 as a baseline, the task is to predict the full intensity image at time t + 1 from the previous four frames. The network output is interpreted directly as the predicted frame, and we train it with MSE loss between predicted and truth intensities. This formulation treats the problem as per-pixel nonlinear regression and encourages the network to reconstruct detailed spatial patterns of cell fluorescence. We construct 82 sliding windows across the three movies and split them into 57 training, 12 validation, and 13 test samples with batch size 16 and Adam optimizer. Training runs for 30 epochs, and we monitor training and validation MSE to check for overfitting.

Model 2 keeps the same TinyCNN architecture and loss function but doubles the temporal context to eight frames. The goal is to test whether a longer motion history improves the ability to localize where aggregation will occur. Across the three movies, this yields 70 sliding windows, splitting into 49, 10, 11

for train, validation, and test sets. Training also uses MSE loss and Adam with the same hyperparameters, same batch size, and the same data-splitting strategy as Model 1.

Model 3 uses the same TinyCNN but changes the target from the next frame to a *velocity map*. For each window, the label y is defined as y = frame(t+1) – frame(t), so the network predicts per-pixel motion rather than absolute intensity. We train with L1 loss between the predicted and true velocity maps, because velocities are mostly near zero with a few localized changes, and L1 is more robust to these sparse large deviations. At evaluation time we reconstruct the next frame as $I_{t+1} = I_t + v$ and compute both MSE on I_{t+1} and the Euclidean distance between predicted and true aggregation centers. To stabilize training, we add early stopping based on validation L1 with a patience of 5 epochs and a maximum of 40 epochs, restoring the best validation checkpoint before final testing.

Results

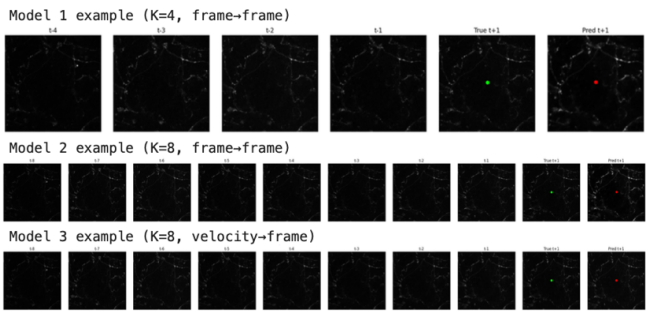
1. Table of metrics

Model	K (frames)	Test MSE on t+1	Test L1 on velocity	Central error (mean ± CI)
Model 1	4	0.000127	-	1.198821 ± 0.389822
Model 2	8	0.000126	-	1.068192 ± 0.448971
Model 3	8	0.000156	0.00646	1.424790 ± 0.554435

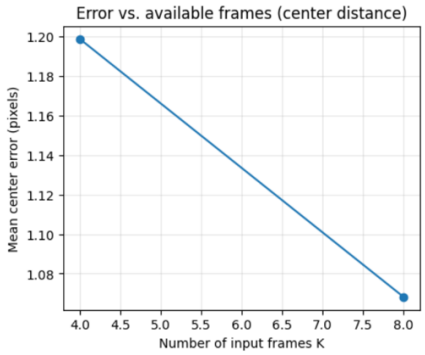
This table summarizes the quantitative performance of our three models on the test set. The Model 1 achieves a test MSE of about 0.000127 and a center error of roughly 1.198821 ± 0.389822. When we increase the temporal window to K = 8 in Model 2, the MSE is similar but the center error is slightly reduced, suggesting that using more history helps the model localize the eventual aggregation spot a bit more accurately. The Model 3 additionally predicts the per-pixel velocity field with a small average velocity L1 error, but its center error is higher than the two direct frame predictors.

In general, we could say that richer motion modeling provides interpretable velocity outputs, whereas the simpler CNN with a longer input window gives the best spatial accuracy on the aggregation center.

2. Figures



Each row shows one test movie. For Model 1 and Model 2, the left panels display the last K input frames, from t−K to t−1. The two right panels show the true frame at t+1 with the truth aggregation center at green dot and the predicted frame with the predicted center at red dot. The bottom row shows Model 3, where the predicted t+1 frame is obtained by adding the predicted velocity to the last input frame. Across all three models, the red and green dots are spatially close, indicating that the models can anticipate the future aggregation hotspot from only a few early frames.



The x-axis shows the number of input frames K, and the y-axis shows the mean center error computed on the test set. We plot one curve using the best frame model for each K, Model 1 for K=4 and Model 2 for K=8. We can see that increasing the temporal window from 4 to 8 frames reduces the average center error, suggesting that having access to longer motion history modestly improves spatial localization of the future aggregation hotspot.

Conclusion

The project shows that even relatively small convolutional models can already capture the aggregation pattern of Dicty and predict the final hotspot with MSE. We found that increasing the temporal window will reduce the center error, suggesting that having more early frames helps the network disambiguate noisy local motion and better infer the eventual meeting spot. The velocity model was able to learn smooth per-pixel motion fields and reconstruct the next frame with competitive MSE, although its center accuracy was slightly worse than the baselines.

However, the models were trained on a relatively small number of movies without strong regularization or data augmentation, so they may not generalize to very different aggregation patterns or imaging conditions. Second, all predictions are point estimates, which means there is no notion of confidence or uncertainty. Third, the current evaluation focuses only on spatial accuracy at a single future time point. We did not explicitly quantify temporal uncertainty.

In the future, we could add uncertainty maps, so that the model can highlight regions where it is unsure and suggest where to acquire more frames. We could also try multi-scale architectures that can better capture both global and local motion cues, and compare them to the current tiny CNN baselines.