CS230

Quantifying the power of light-controllable molecular motors: Tracking velocities of individual actin filaments using NN facilitated polarity labeling

Cooper Galvin
Department of Biophysics
Stanford University
coopgalv@stanford.edu

Nina Hooper
Department of Aeronautics and Astronautics
Stanford University
nhooper@stanford.edu

Abstract

Automated segmentation and tracking of polar, single molecules of biological matter is an ongoing challenge in biology. The application of deep learning techniques to these types of problems can facilitate higher throughput and better quantification of visual data, making it extremely valuable to the field. We attempt to build a system that will track and calculate the signed velocity of actin filaments in microscopy data. We trained an existing implementation of the YOLOv2 CNN to identify filaments of interest and to provide signed feature data, with a view to then build an RNN architecture for feature tracking. Despite inconclusive results, we are still hopeful that our approach can improve upon current options and our research has suggested paths for further improvement of the methodology.

1 Introduction

Cytoskeletal motors perform the functions of force generation and transport throughout eukaryotic cell species and types, from the beating of cardiac myocytes to the long-range transport of neurotransmitters or nutrients in plants. The Bryant lab at Stanford has designed myosin motors that can be optically controlled in order to test our understanding of protein structure-function relationships and in order to develop new tools for controlling cellular processes in vivo. Ultra-fast motors with deep modulation depth (quickly moving forward and then backward) upon exposure to blue light molecular motors (myosins) have been engineered, but accurately quantifying their speeds in lit and dark states has proved difficult [1][2].

We were requested by Paul Ruijgrok, a postdoctoral scholar in the Bryant lab, to determine whether deep learning could provide an easy solution to calculating the signed velocity of actin filaments propelled by these motors. This directional velocity is called the signed velocity [3]. To establish directionality, actin filaments were polarity labeled. The input to our algorithm is a collection of still frames from the gliding data videos prepared by Paul Ruijgrok. We aimed to implement a YOLO CNN algorithm to identify and output polar filaments' position and orientation.

2 Related work

There are a variety of strategies for identifying and tracking generic objects using a combination of convolutional and recurrent neural networks [4][5][6]. One such approach was proposed in [4] that is based on YOLO for object detection with convolutional neural networks. This method

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

extends the neural network analysis into the spatio-temporal domain and harnesses the power of LSTMs for efficient visual object tracking. It appears to have been tested only on single object tracking so far. Other earlier approaches used tracking-by-detection methods [7][8]. The advantage of tracking-by-detection methods over the former recurrent neural network approach lies in only training a single neural network.

Offering higher throughput and better quantification of visual data, the application of deep learning techniques is becoming popular in the detection and tracking of microscopic objects in the field of biology. Work analogous to this project has been done in cell-motility studies, where accurate segmentation and tracking of cells in microscopic imaging is required. CellTrack [9] is an open-source software package for detecting and tracking cells. Until now, research on the behavior and velocities of actin filaments has been largely conducted by hand [2].

3 Dataset and Features

We worked with a postdoctoral scholar in the Bryant lab, Paul Ruijgrok, to collect videos of the myosin motors sliding actin filaments in the lit and dark states and converted those videos into 8-bit RGB jpeg files. We sampled 25 frames from each lit and dark frames from 2 videos, totaling 100 frames. One of the videos has a "fast" frame rate, at one frame per 5ms, and the other has a "slow" frame rate, at one frame per 100ms. The frames each 471 by 208 pixels. We also performed data augmentation which added 200 additional up-close photos of the filaments. The photos were created by cropping the video frames and rotating them to add more data and variety to the dataset. Based on online tutorials that describe training the YOLO9000 to recognize a new class of object, we felt confident that this was a sufficient amount of data to achieve basic detection.

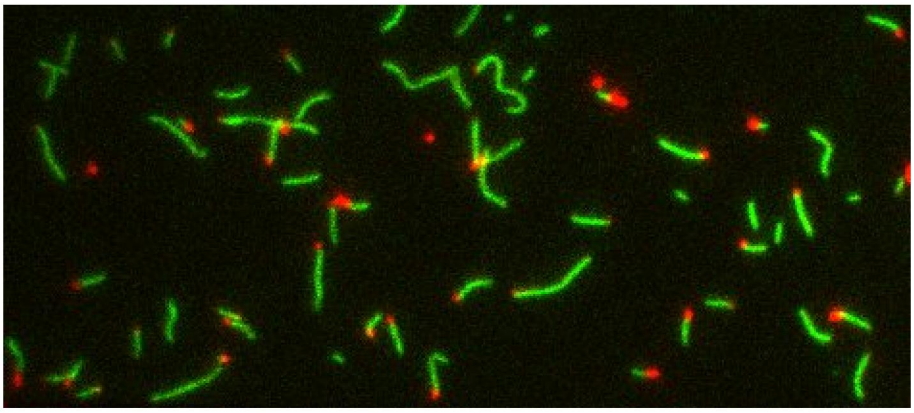

Figure 1: An example dark RGB frame from a myosin motors sliding actin filaments video.

We used LabelBox, a free online data tagging application, to label our images. Because the direction that the filaments are facing is of interest for this application, we created 8 label categories $(up, down, left, right, up_left, up_right, down_left, down_right)$ to describe the direction of the head of the filament. Figure 3 As the filaments move over the course of the video, their label will change to reflect the direction they are pointing in. We excluded filaments that did not have heads (red point) and lone heads from our tagging.

We split our data 90-10 into training and test data, selecting the full frames for test data. We did not have an independent dev set, although we would like to include this in future work. Before training on the data, we had to perform preprocessing that converted the labels from the coordinate system output by LabelBox to the format required for YOLO9000.

The LabelBox output is a single JSON file for all input images and it provides coordinates for all four corners of each of the segmentation boxes. In LabelBox, the bottom left corner is the origin. We created a python script that generated .txt files which correspond to each of the training and test images. These text files contain the class of object detected followed by the coordinates of

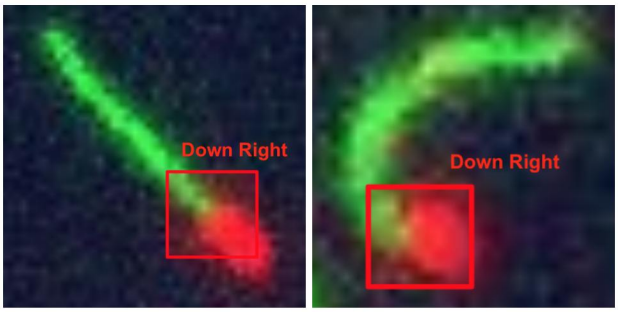

Figure 2: Example labels on individual filaments. Notice that the label is determined only by the direction that the head is facing.

the object. We utilized a dictionary with numbers for each of the labeling classes (referring to the orientation of the filaments). The training algorithm takes in the x and y coordinates of the center of the detection rectangle and the width and height of the detection rectangle. These values are then scaled to represent a fraction of the total width and height of the input image. We also later wrote a script that would convert all the labels to "filament", omitting the direction information, with the hope that learning only one new category (instead of 8) would improve the results.

4 Methods

We chose to apply the YOLO9000 algorithm [5] to our image segmentation portion of the project. YOLO9000, so named because it can detect 9000 classes of objects, is a variant of YOLOv2 that we learned in CS230 and is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. YOLO9000 uses the YOLOv2 loss function. We selected this implementation because it has been shown to be effective in cases where limited labeled pictures are available for training [11]. YOLO requires a single forward propagation pass through the network to make predictions, enabling real time results. Running YOLO9000 on an input image, the output is the same image overlaid with bounding boxes around recognized objects. YOLO bounding boxes are output only if over a defined confidence threshold. Bounding boxes are also passed through non-max suppression to eliminate duplicate bounding boxes on a single object.

5 Experiments/Results/Discussion

We conducted two main experiments: one in which the images were labeled with heading direction (8 classes) and one in which labels only reflected the existence of a filament (1 class). We also varied our learning rate according to a log scale, beginning at 0.00001 and continuing to 0.001 and determined that we had the most stable descent at 0.0001. We also varied our batch size from 5-30 images.

Figure 3 shows the training error per training iteration from our first experiment using a batch size of 5 and learning rate of 0.0001 with heading direction labels. Despite running 700 iterations, the training error did not reach an acceptable level. There was also very strange behavior at 100 iterations and again after 200 iterations where the error spiked up to 2000-3000. We assume this is a result of the video data being fed into YOLO sequentially and thus the algorithm over-fitting to one video before being given another one that is significantly different. However, this answer is a bit unsatisfying because we don't know how to explain why it doesn't happen earlier in training, given our relatively small amount of data and batch size. Loading these weights and outputting predictions, we initially found that no boxes were being draws, indicating that the algorithm either detects nothing or has very low confidence about its detections. Lowering the confidence threshold to zero, an abundance of detection boxes appeared as shown in Figure 4. With a slightly higher confidence threshold (0.01) but still well under the default value (0.2), we see fewer detected objects, yet none of the detections represent accurate filament orientations as shown in Figure 5.

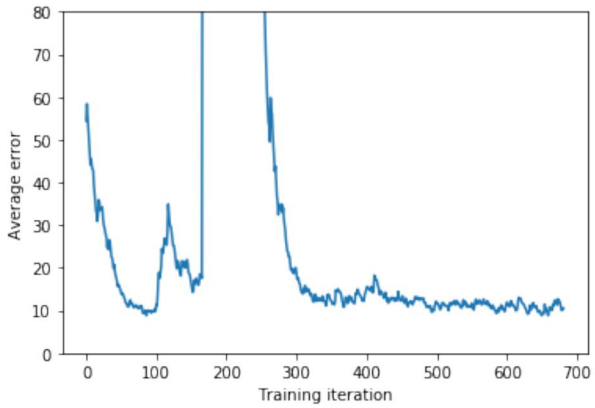

Figure 3: Training error for experiment with labels that reflect heading direction.

Figure 4: Predictions when the confidence threshold is lowered to zero.

Our second experiment used a batch size of 30 and data labels were all converted to "filaments", only detecting a single class. Figure 6 shows a much smoother training curve and reaches an average error of 3.194115. While this curve looks much more promising, the connection to AWS terminated unexpectedly at 96 iterations and weight files for testing are only saved each 100 iterations. Re-running this model will be part of our future work.

6 Conclusion/Future Work

The goal of this project was to both identify and track individual filaments as they moved throughout microscopy videos. Despite using a state of the art algorithm for object detection, we struggled to get the neural network to predict correct bounding boxes and categories. We believe this is a result of the labels changing as the filaments wiggle. The images are grainy and the direction labels were done by eye. we believe there may have been enough inconsistency in tagging that the neural network wasn't properly able to learn what a label like "up left" meant, for example. We hoped that reducing the categories to a single class would improve the results and, based on the training error curve, it seems like this is a more promising direction. However, training was disrupted and further exploration of the results has not yet been conducted. The functions we wrote and training outputs can be found at: `https://drive.google.com/open?id=1VdPApDezS8hQLDHYWRQzYvjv5cwRjuHQ`

Unfortunately, due to a family medical emergency, time to work on this project was cut short and resulted in a pivot from our original proposal "MentorBot". We believe that the results of this project should not discourage the Bryant Lab at Stanford from pursuing deep learning approaches to filament tracking, but recognize that our approach needs significantly more time and refinement

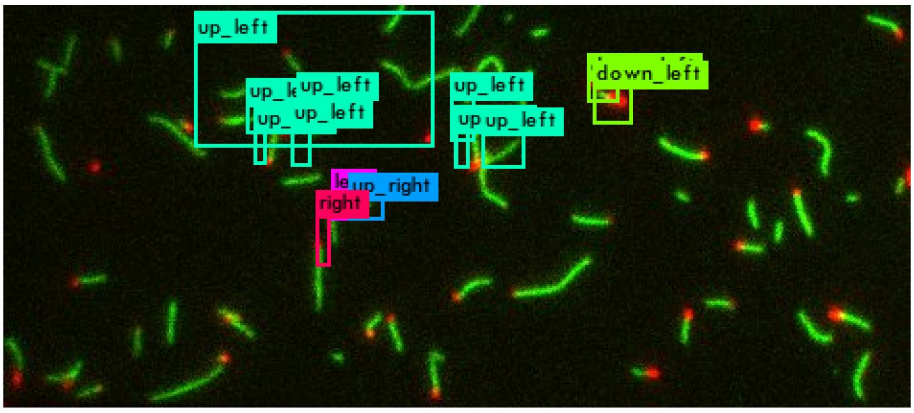

Figure 5: Predictions when the confidence threshold is 0.01.

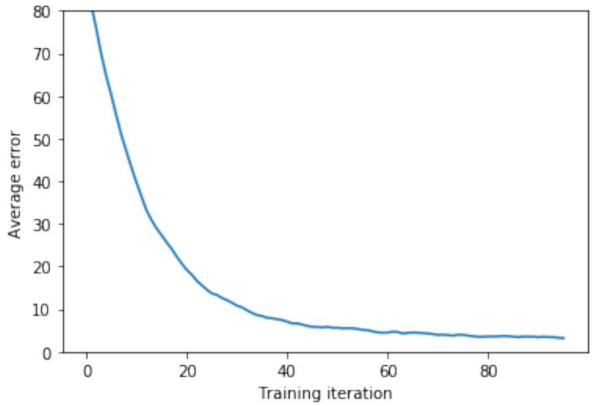

Figure 6: Predictions when the confidence threshold is 0.01.

to achieve useful results. So a simple implementation of the YOLO algorithm to the problem of polarity labeling was not attainable. Either more data needs to be labeled or we need to further tune hyperparameters until it can give reasonable predictions on the test set. Also, we may want to experiment with stopping training earlier as apparent over-training to the training set was apparent as training error became low (under 10% error, which is around the error between human labelers), but the performance was both quantitatively and qualitatively suboptimal on test data. Before reporting the final data to Dr. Ruijgrok, we will also try some regularization techniques to see if they can preclude overfitting.

Directions for future work include changing the labeling scheme to capture the whole filament, as opposed to just the filament-to-head (green-to-red) transition. We believe this may be more successful at at least putting bounding boxes around the filaments. We would still like to pursue the ultimate goal of feeding the YOLO output into an LSTM that can then output the velocity and orientation of filaments.

7 Contributions

Cooper and Nina worked on this project together in person for the majority of time spent on the project. All code was discussed and planned together. Cooper implemented the making_helper_python_function.ipynb and plotting_progress.ipynb functions while Nina was labeling the data. Cooper also labeled some data and Nina made modifications to Cooper's functions. The first test was run on Cooper's lab desktop computer and after getting poor results, Nina changed the

labels of the training data and ran the second test on AWS. Both Nina and Cooper contributed to the writing of this report.

References

[1] Nakamura, M et al. Remote control of myosin and kinesin motors using light-activated gearshifting, 2014, Nature Nanotechnology,, 9, 693-697

[2] Chen, L. Engineering controllable bidirectional molecular motors based on myosin, Nat Nanotechnol. 2012 Apr; 7(4): 252–256

[3] Aksel, T. Ensemble force changes that result from human cardiac myosin mutations and a small-molecule effector. Cell Rep. 2015 May 12;11(6):910-920

[4] Ning, G. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking, arXiv:1607.05781 [cs.CV]

[5] Daniel, G., Farhadi, A., and Fox, D. "Re3: Real-Time Recurrent Regression Networks for Visual Tracking of Generic Objects." IEEE Robotics and Automation Letters. May 17, 2017.

[6] Ondruska, P. and Posner, I. "Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks." AAAI Conference on Artificial Intelligence. February 2, 2016.

[7] Wang, L., Ouyang, W., Wang, X., and Lu, H. Stct: Sequentially training convolutional networks for visual tracking. CVPR, 2016.

[8] Hall, D. and Perona, P. From categories to individuals in real time–a unified boosting approach. In CVPR, pages 176–183, 2014.

[9] Sacan, A., Ferhatosmanoglu, H., Coskun, H. CellTrack: an open-source software for cell tracking and motility analysis. Bioinformatics, Vol 24, Issue 14. July 2008. 1647-1649.

[10] Redmon, J., Farhadi, A. YOLO9000: Better, Faster, Stronger, arXiv:1612.08242 [cs.CV]

[11] Tijtgat, N. "How to Train YOLOv2 to Detect Custom Objects." Timebutt.io, May 16, 2017. http://my-ghost-blog.com/how-to-train-yolov2-to-detect-custom-objects/.