Segmenting neuronal growth cones using deep convolutional neural networks

Jackson Y Huang
Queensland Brain Institute

The University of Queensland
St Lucia, QLD 4072
Email: jackson_huang@live.com

Nicholas J Hughes
Queensland Brain Institute &
School of Mathematics and Physics
The University of Queensland
St Lucia, QLD 4072
Email: nickjhughes@gmail.com

Geoffrey J Goodhill
Queensland Brain Institute &
School of Mathematics and Physics
The University of Queensland
St Lucia, QLD 4072
Email: g.goodhill@uq.edu.au

***Abstract*—Connections form between neurons during neural development guided by growth cones, highly dynamic structures at the tips of growing axons. Understanding the biological mechanisms underlying this guidance requires understanding the dynamic morphology of growth cones, and this requires the segmentation of growth cone outlines from potentially very long timelapse movies. Previous approaches to this problem have been based either on time-consuming human input, or on algorithms customised for specific datasets. Here we evaluate the effectiveness of deep convolutional neural networks (CNNs) for growth cone segmentation. First, we apply a deep CNN to a standard benchmark cell nuclei segmentation problem, and show that it achieves performance comparable to standard image processing techniques after training with only one image. Second, we show that the deep CNN can achieve good segmentation of a large set of phase-contrast timelapse movies of growth cones after training with only a few frames. Thus, deep CNNs provide a general method for growth cone image segmentation that is broadly applicable and require very little training.**

I. Introduction

Proper brain function depends on precise patterns of wiring between neurons. This wiring forms during neural development via the outgrowth of axons from neurons, which are guided via a variety of molecular cues [1]. A key specialisation at the tip of the axon involved in such guidance is the growth cone [2]. This is a highly dynamic sensory-motile structure that appears to actively sample its environment, with morphological changes occurring on the timescale of one minute or less. Since growth cones often have to traverse hundreds of microns to reach their targets, but move at speeds usually less than 1 micron per minute, a full understanding of their dynamics during growth requires analysis of time-lapse movies consisting of potentially hundreds of frames. Furthermore, since their behaviour shows a large amount of variability, statistical regularities may only become apparent when comparing many tens or hundreds of different growth cones [3]. Such datasets can therefore consist of the order of 10^4 - 10^5 frames. This is beyond the range in which a human can analyse each frame, and thus reliable automated techniques for image analysis are clearly essential.

To analyse morphology, each growth cone image must be segmented from its background. Due partly to the constraints inherent in imaging delicate biological structures for long periods of time, the quality of these images is often not high, making automated segmentation a challenging problem. In previous work Keenan et al. [4] developed algorithms to identify the axon tip in phase-contrast images, without focusing on detailed growth cone morphology. Chitsaz et al. [5] used classical imaging processing methods for fluorescence images of growth cones, relying on human input for setting certain parameter values. Goodhill et al. [3] used similar methods combined with support vector machine learning based on approximately 10 frames per movie, followed by human correction of imperfections, to segment both phase-contrast and fluorescence images. However all these approaches were domain-specific, and required either customized algorithm development, human intervention, or both.

In the past few years, automated image analysis has been revolutionised by the introduction of deep convolutional neural networks (CNNs), which now provide state of the art performance in many domains [6], [7], [8]. It is therefore of great interest to determine how well they perform for the type of biological image segmentation problems described above [9]. We first apply a deep CNN to a benchmark problem in cell nuclei segmentation, and show that it achieves performance comparable to standard image processing techniques with training on only a single image. We then apply deep CNNs to a large dataset of growth cone images we previously generated [3]. The performance of the CNN was slightly lower than that of the customized, semi-automated algorithm developed in [3]. However, the CNN required very little training, and thus this approach is potentially broadly applicable across a wide range of similar segmentation problems.

II. Methods

A. Comparison of deep CNNs with classical algorithms for cell segmentation

1) Image preprocessing: The U2OS and NIH3T3 image datasets [10], including their hand segmentations, were obtained from github.com/luispedro/Coelho2009_ISBI_NuclearSegmentation. To optimise the input data for CNN training, the raw experimental images were normalised by converting pixel values to Z-scores. We re-ran the Python code provided by

978-1-5090-2896-2/16/$31.00 ©2016 IEEE

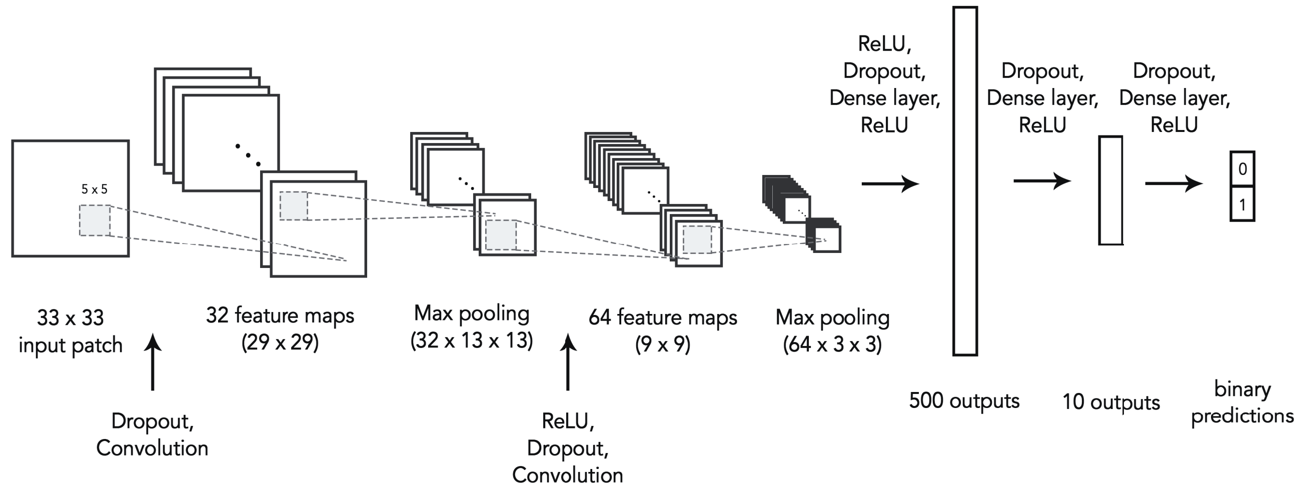
5 x 5
33 x 33 input patch
Dropout, Convolution
32 feature maps (29 x 29)
Max pooling (32 x 13 x 13)
ReLU, Dropout, Convolution
64 feature maps (9 x 9)
Max pooling (64 x 3 x 3)
ReLU, Dropout, Dense layer, ReLU
500 outputs
Dropout, Dense layer, ReLU
10 outputs
Dropout, Dense layer, ReLU
0
1
binary predictions

Fig. 1: **The deep convolutional neural network architecture used.**

[10] to ensure we could reproduce their quantitative analysis of classical image processing algorithms.

2) Deep CNN: We used a deep convolutional neural network to perform image segmentation on 33×33 pixel patches of each image. The output of the network was the probability that the pixel at the centre of the image patch was within the shape being segmented, which we rounded to 0 or 1. For this work we wanted to test the abilities of a fairly 'generic' deep CNN architecture [11], [12], and the network structure was therefore as follows (Fig. 1).

- Input (33×33 pixel image)
- Dropout ($p = 0.1$)
- 2D Convolution (32 5×5 pixel filters)
- 2D Max Pooling (4×4 pixel pool size, 2×2 pixel stride)
- ReLU activation layer
- Dropout ($p = 0.25$)
- 2D Convolution (64 5×5 pixel filters)
- 2D Max Pooling (4×4 pixel pool size, 2×2 pixel stride)
- ReLU activation layer
- Dropout ($p = 0.25$)
- Fully connected linear layer (500 outputs)
- ReLU activation layer
- Dropout ($p = 0.25$)
- Fully connected linear layer (10 outputs)
- ReLU activation layer
- Dropout ($p = 0.25$)
- Fully connected linear layer (2 outputs)
- SoftMax activation layer
- Output (2 probabilities)

All CNN code was implemented in Python 3.4 with the Keras library (github.com/fchollet/keras).

3) Datasets: The U2OS raw images were 1349×1030 pixels, and the NIH3T3 images were 1344×1024 pixels. To construct training/validation/testing datasets, patches of size 33×33 pixels centred on each pixel within the raw image were constructed. All input images were padded with zeros to ensure the outputs were the same size as the inputs. For the training and validation datasets, these collections of patches were balanced such that there was an equal number of examples in each class (i.e., within and not within the shape), by randomly discarding examples from the larger class. Patches were split into training and validation datasets based on which image frame they belonged to (the number of frames used for each dataset is given below). To construct testing datasets, all patches from a single image were used and no balancing was performed.

4) Training: The model was trained on the training dataset (a collection of 33×33 pixel image patches) with batched stochastic gradient descent, with a batch size of 128 patches, a learning rate of 0.01, decay of 10^{-6}, momentum of 0.9, using Nesterov momentum, and with the negative log-likelihood loss function. The model was trained for a maximum of 50 epochs, stopping early if the performance on the validation dataset failed to improve during 5 consecutive epochs, to prevent overfitting. The order of the training examples was shuffled at each epoch.

5) Testing: Testing of trained networks was done by performing predictions on all the patches from a single image (i.e., a testing dataset, as described above), and performance was measured using the metrics described below.

6) Postprocessing: Objects smaller than 2500 pixels in area were removed from all reference and prediction images prior to performance analysis. This was done to allow a fair comparison with [10], who applied the same post-processing to all segmentation results.

Training. Deep CNNs were trained using 1, 2, 4, or 8 training frames plus one additional validation frame. For the U2OS dataset, single frame training, 48 CNNs were trained separately using each of the 48 frames as a training frame. For the NIH3T3 dataset, 49 CNNs were trained using each of the 49 frames as a training frame. In each training instance, the validation frame was randomly selected from the remaining frames in the dataset. The validation frame was used to terminate training when the segmentation accuracy of the validation frame reached saturation. For multi-frame training, 3, 5, or 9 frames were randomly selected from a single dataset. One of the frames was used as the validation frame and the remaining frames were used for training. Unique seeds were used to ensure that random frame combinations would be chosen each time. In multi-frame training approximately 150 CNNs were trained for each of the U2OS and NIH3T3 datasets.

Segmentation. The trained CNNs then segmented all unseen images from the dataset on which they were originally trained.

Performance. Training took 15 min for 1 training / 1 validation frame on an NVIDIA K80 GPU. Inference took 1.5 min. The operating system used for all training, prediction and image analysis was CentOS Linux release 7.2.1511. We used the Python code of [10] to evaluate the segmentation accuracy of the deep CNNs. The following measurements quantify the similarities and differences between corresponding objects in the prediction (P) and reference (R) binary images [10].

Rand and Jaccard indexes. We range over all pairs of corresponding pixels in the P and R binary images to determine the degree of agreement between P and R. A true positive (T_p) is counted if the pixel pair in P agree and the pixel pair in R agree. A true negative (T_n) is counted if the pixel pair in P disagree and the pixel pair in R disagree. A false negative (F_n) is counted if the pair in P disagree and the pair in R agree. A false positive (F_p) is counted if the pair in P agree and the pair in R disagree. The Rand index (RI) is then given by

$$RI = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}.$$

The Jaccard index (JI) is

$$JI = \frac{T_p}{T_p + F_p + F_n}.$$

Hausdorff distance. The minimum distance to the reference border is calculated for each pixel within the border of the predicted object. The Hausdorff distance is defined as the maximum of these distances. A lower Hausdorff distance indicates a smaller error in the border of the predicted object.

Normalised sum of distances. The normalised sum of distances (NSD) captures the segmentation error normalised by the distance of each erroneous pixel to the reference border. Hence, segmentation inaccuracies farther from the border (i.e. closer to the object's centre) incur greater penalty (NSD value closer to 1).

Object separation errors. A split error occurs when a single object is identified as two or more objects. A merge error occurs when multiple objects are identified as a single object. An added (spurious) error occurs when the background is identified as an object. A missing error occurs when an existing object is identified as background pixels.

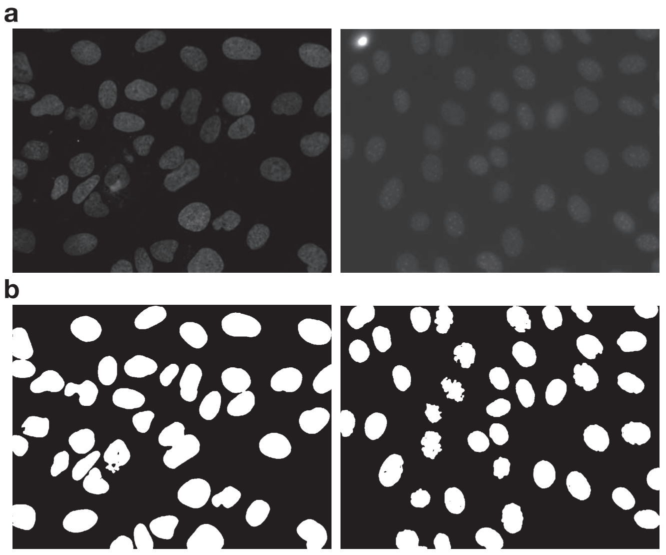

Fig. 2: **Raw and prediction images of cell nuclei.** (a) Raw images from the U2OS (left) and NIH3T3 (right) datasets. (b) Corresponding segmentations produced by the CNN using 1 training frame and 1 validation frame.

B. Comparison of deep CNNs with a semi-automated algorithm for growth cone segmentation

We used the same CNN architecture as described above. Our growth cone image dataset consisted of a total of 600 experimental images of growth cones and axons from a total of 10 movies, which were previously semi-automatically segmented for [3], and we now also segmented manually. Each experimental image consisted of a cropped version of the full movie frame, restricted to a large but local region containing the growth cone (in the range of hundreds by hundreds of pixels in size). Smaller quadrilateral ROIs isolating growth cones from their axons were then drawn by hand for all 600 images.

The semi-automated tracing performed by [3] is described in that paper. Briefly, images within the ROI were filtered using the Matlab image texturing transforms stdfilt and entropyfilt. 5-10 frames were then used to manually optimise the outline, and these outlines were then used to train linear support vector machines. Automated re-segmentation of the remaining frames was then performed, and any remaining imperfections corrected manually. The interaction time required per movie was typically $\lesssim 1$ hour.

As the pixel values in the different movies varied substantially, and were originally 8-bit values, we performed some normalisation procedures and then ZCA whitening to the raw imaging frames to make them more suitable for input to the CNN. There were no restrictions in the CNN which ensure that the final traced shape was connected, and in some cases there were small isolated areas designated as being inside a growth cone. We therefore reduced the results to only the largest 8-connected component. Performance was measured using the same metrics as for cell nuclei data described above, except

Algorithm	Rand ↑	Jaccard ↑	Hausdorff ↓	NSD (×10) ↓	Split ↓	Merged ↓	Spurious ↓	Missing ↓
AS Manual	95%/93%	2.4/3.4	9.8/12.0	0.4/0.7	1.6/1.0	1.0/1.2	0.6/0.0	2.2/3.2
RC Threshold	92%/77%	2.2/2.2	34.6/26.5	1.2/2.6	1.0/1.0	2.5/2.5	0.3/1.9	5.5/22.0
Otsu Threshold	92%/75%	2.2/2.1	34.8/36.7	1.2/3.5	1.0/0.8	2.4/2.1	0.3/1.7	5.5/26.4
Mean Threshold	96%/82%	2.2/1.9	26.6/24.6	1.0/2.4	1.3/1.5	3.4/5.1	0.9/3.1	3.7/4.8
Watershed (direct)	91%/78%	1.9/1.6	35.0/19.3	3.6/3.7	13.9/2.9	1.2/2.4	1.9/11.6	3.0/5.5
Watershed (gradient)	90%/78%	1.8/1.6	34.6/21.7	3.0/3.8	7.7/2.6	2.0/3.0	1.9/11.4	2.9/5.4
Active Masks	87%/72%	2.1/2.0	191.5/98.0	5.6/5.0	10.4/1.9	2.4/1.5	0.4/3.9	10.3/31.1
Merging Algorithm	96%/83%	2.2/1.9	13.3/16.2	0.7/2.5	1.8/1.7	2.2/2.9	1.0/7.1	3.3/5.9

TABLE I: Reproduced version of Table 2 in [10] using their code from GitHub (run using Python 2.7). In each column the first number refers to performance on the U2OS dataset, and the second to the NIH3T3 dataset. 'AS Manual' refers to the degree of agreement between two independent human observers as measured by [10]. The arrows indicate whether higher or lower values of the measure indicate better performance.

Dataset	Algorithm	Rand ↑	Jaccard ↑	Hausdorff ↓	NSD (×10) ↓	Split ↓	Merged ↓	Spurious ↓	Missing ↓
U2OS	Traditional	96%/87%	2.2/1.8	12.9/148.3	0.7/5.5	1.1/13.8	1.2/3.4	0.3/2.0	2.9/10.8
	CNN 1	97%	2.2	20.8	1.0	1.3	3.5	0.7	0.6
	CNN 2	97%	2.2	19.3	0.9	1.3	3.4	0.7	0.4
	CNN 4	97%	2.3	18.3	0.9	1.3	3.4	0.6	0.3
	CNN 8	97%	2.3	17.9	0.9	1.3	3.3	0.6	0.3
NIH3T3	Traditional	83%/72%	2.1/1.6	15.9/98.0	2.5/5.0	0.8/2.9	1.5/5.1	1.7/11.6	4.8/31.1
	CNN 1	87%	2.3	29.2	2.5	1.5	3.8	2.8	5.8
	CNN 2	89%	2.3	27.2	2.2	1.5	3.9	2.9	4.1
	CNN 4	91%	2.4	26.2	2.0	1.6	4.0	2.9	2.1
	CNN 8	92%	2.4	24.1	1.8	1.6	4.1	3.0	1.1

TABLE II: Mean segmentation performances of deep CNNs trained on 1, 2, 4 and 8 frames from the U2OS or NIH3T3 image datasets. The best/worst segmentation results of the traditional algorithms evaluated by [10] (see Table I) are provided for comparison.

that object separation errors were not relevant since there was only one growth cone per image. Training took 15 s for 7 frames, and inference took 5 s per frame.

C. *Artificial growth cone images*

Artificial growth cone images (100×100 pixels) were generated so as to closely match the overall statistics of our experimental images. The images consisted of a noisy background and a growth cone in the foreground. To generate the background, an array of values were randomly sampled from a normal distribution and blurred with a Gaussian filter of width 2.8 pixels. The mean intensity of the background (μ_b) was a random integer between 30 and 120 (informed by our real growth cone dataset). Growth cone outlines were produced by combining semiautomatic tracings of real growth cones used in [3]. The mean intensity of the growth cone (μ_g) was a random integer between 0 and 15 below μ_b. Gaussian noise of variance 10 was added to the growth cone.

To blend the background with the growth cone, the growth cone border was replaced with pixels of intensity between μ_g and $\mu_g + \frac{\mu_b - \mu_g}{1.4}$. The border was either 3 or 4 pixels wide and blurred with a Gaussian filter of width 10 pixels. In addition, to capture the characteristic halo surrounding growth cones obtained by phase contrast imaging, two overlaying strips of noisy, high-intensity pixels (noise variance between 20 and 30; Gaussian filter of width 4 pixels) were added at a distance of 5 pixels from the growth cone border. The inner strip was more intense than the outer strip to mimic the fading of the halo with increasing distance. Finally, a Gaussian filter of width 2 pixels was applied to the entire image. The artificial data was augmented by rotating the images in multiples of 90 degrees to obtain growth cones arranged in all four major orientations.

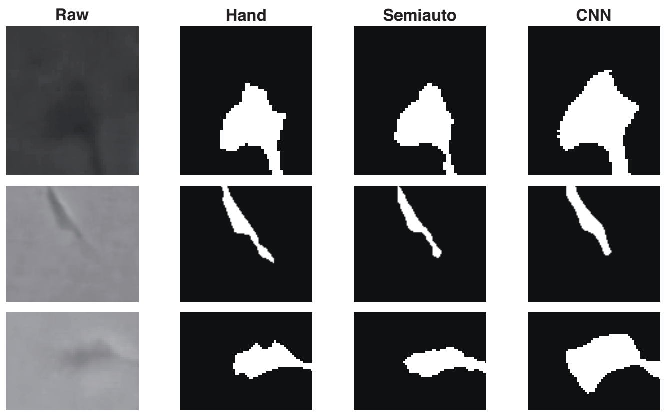

Fig. 3: **Examples of growth cone segmentations.** The CNN was trained with the same seed on a random selection of 7 frames from ten movies. CNN predictions were performed on frames which were not used in training or validation.

III. Results

A. *Application of the deep CNN to a benchmark cell segmentation problem*

In order to provide a general assessment of the abilities of the deep CNN for cell segmentation, we first applied it to a standard benchmark problem. In particular, [10] presented a systematic comparison of the performance of a number of classical imaging processing techniques for the segmentation of two collections of cell images, U2OS and NIH3T3 cells (Fig. 2). These datasets and their original python code are available at github.com/luispedro/Coelho2009_ISBI_NuclearSegmentation. They first manually segmented all the images (about 50 in each dataset, with an average of 40 cells per image), and then used a number of different performance measures to capture several aspects of how well the automated segmentations matched the manual segmentation.

To ensure a fair comparison with the deep CNN we first reran the original code of [10], to confirm we could reproduce their performance values (Table I, compare with Table 2 in [10]). To compare with the deep CNN (see Methods), we tested how its performance varied with the number of images used for training. Remarkably, we found that training with only a single image was sufficient to achieve almost saturating performance (since each image contains about 40 cells) (Figure 2). Table II shows the mean performance of the deep CNN averaged over each image used as the training frame. Despite the extremely limited amount of training, performance was similar to the standard algorithms. Since this performance was already high, we did not attempt a systematic optimisation of the hyperparameters of the deep CNN to determine if it could consistently exceed the other approaches. Thus, a deep CNN requires only very little training to achieve a high level of performance on a standard image processing task.

To determine if a larger number of training frames could improve the segmentation performance we also trained on 2, 4 and 8 frames (Table II). Some metrics were largely unaffected, while others increased slightly with more training. The most noticeable improvements were in the Rand and Missing indices for the NIH3T3 dataset. These images are generally more challenging than U2OS, leaving more room for improvement.

B. Application of the deep CNN to growth cone segmentation

Growth cones generally present harder segmentation challenges than the cell nuclei data discussed above. Firstly they can have fine filopodial structures. Secondly the data often come from timelapse movies of one growth cone at a time, with imaging parameters and quality usually varying between movies, and even over time within a movie.

	Frames	Rand ↑	Jaccard ↑	Hausdorff ↓	NSD (×10) ↓
CNN	7	88%	3.5	13.1	3.5
	14	89%	3.6	10.0	3.0
	28	89%	3.5	10.7	3.0
	56	90%	3.5	9.1	2.7
	100	90%	3.5	9.0	2.6
Semiauto		92%	4.2	6.5	2.1

TABLE III: **CNN trained and tested on growth cone movies.**

Our original dataset [3] consisted of thousands of frames from hundreds of movies, which were segmented by a customized, semi-automated algorithm based on a combination of classical image processing techniques and human judgement ([3], see Methods). In order to compare the performance of

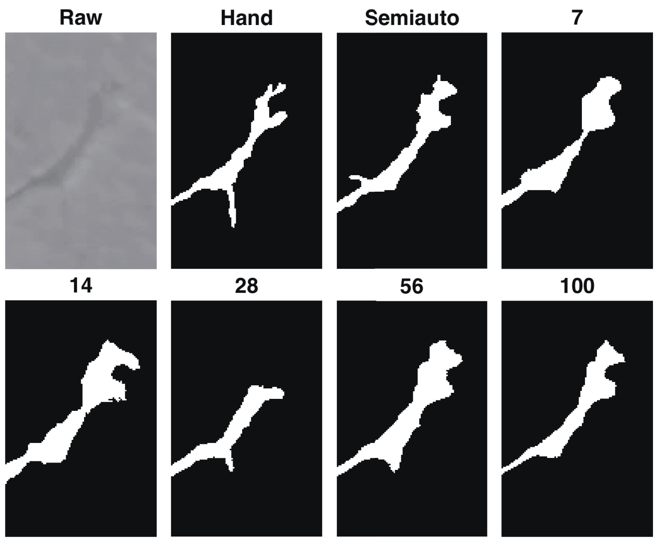

Fig. 4: **Variation of CNN segmentations with number of training frames.**

	Frames	Rand ↑	Jaccard ↑	Hausdorff ↓	NSD (×10) ↓
CNN	7	95%	5.7	4.1	2.1
	14	96%	6.0	3.3	1.6
	28	96%	6.1	3.0	1.6
	56	96%	6.0	2.5	1.4
	100	96%	6.0	2.4	1.4

TABLE IV: **CNN performance on artificial growth cone data.**

the CNN with this algorithm we hand-segmented a representative subset of 10 movies of 60 frames each to provide a ground truth for both algorithms. Since each image contained only a single growth cone we calculated only the Rand, Jaccard, Hausdorff and NSD measures. CNN performance was compared for between 7 and 100 training frames (in each case chosen across several movies) (Figs 3, 4, Table III). As expected performance increased with the amount of training data, though the improvement was only slight. Even for 100 frames it did not quite reach the performance of the semi-automated algorithm. Notably however, performance approaching saturation was achieved for training with only 7 frames.

An intriguing issue is that the quality of some of the growth cone images is so poor that hand tracing may not accurately capture the ground truth. Thus, potentially, the CNN is being evaluated unfairly by comparison only with human judgements. To investigate this issue we generated artificial growth cone images, where we knew the ground truth, with noise characteristics qualitatively matched to the real data (see Methods). The CNN now produced excellent performance, again even with only a very small number of training frames (Figs 5, 6, Table IV).

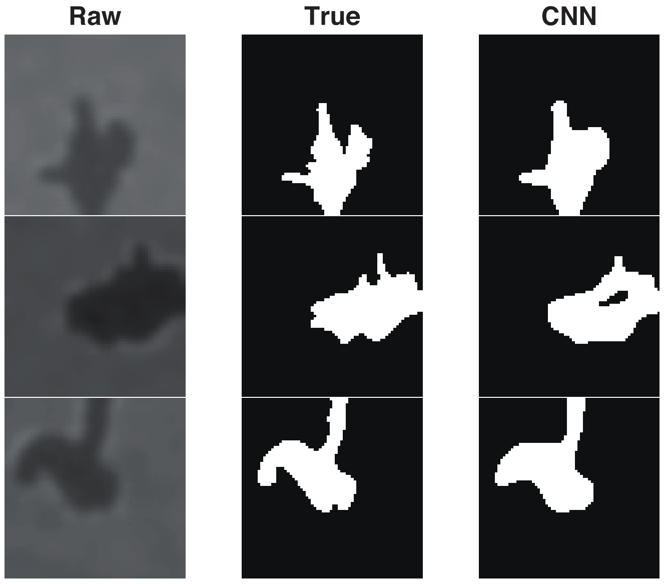

Fig. 5: **Examples of artificial growth cone segmentations.** From left to right: raw image, true outline from which the raw image was generated, and CNN segmentation. CNN was trained with the same seed on a random selection of 7 frames from ten artificial growth cone movies. CNN predictions were performed on frames that were not used for training or validation.

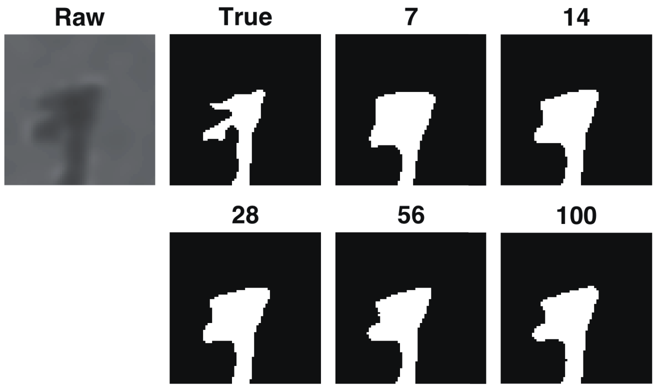

Fig. 6: **Variation of CNN segmentations for artificial growth cones with number of training frames.**

IV. Discussion

Timelapse imaging of growth cones presents particularly difficult image segmentation challenges. Growth cones are very delicate structures, and thus the imaging parameters required to maintain their viability over long periods usually lead to poor-quality images. Furthermore the images can be highly variable between different experiments, and even sometimes within the same movie. Since a general solution to this problem is lacking, we investigated whether a fairly generic deep CNN could provide good segmentation performance for growth cones, first validating our approach on a well-characterised cell nuclei segmentation dataset.

Despite exceeding performance of competing algorithms for many other image-processing problems [8], the CNN did not consistently achieve this for cell nucleus segmentation (Table II). While the CNN was substantially better on some metrics, it was slightly worse on others. The CNN was better on the Rand and Jaccard indexes, indicating its ability to correctly categorize cell and background pixels. However performance was unremarkable for the Hausdorff metric, an approximate measure of the distance separating predicted and reference cell borders. Qualitatively, CNN segmentations were usually more inclusive of border pixels and this may account for the Hausdorff result. The CNN's low NSD values across both the U2OS and NIH3T3 datasets highlight its ability to produce accurate segmentations both at the border and the centre of cells (i.e. less holes inside the cell). The NSD accounts for all pixels in the union of the prediction and reference objects, whereas the Hausdorff metric accentuates an extreme (the maximum of the minima of inter-border distances). While the performance in the split category was mediocre, the performance in the merge category was below most traditional algorithms. A possible explanation for the high merge errors is that many neighbouring cells in the raw images are separated by only 1 or 2 pixels, and the CNN found these small gaps challenging to detect. The CNN was also average at the task of distinguishing between cellular and non-cellular objects. These spurious errors can likely be attributed to the identification of round but non-cellular objects as genuine cells. Interestingly, the CNN achieved excellent performance in the missing category, outperforming all traditional algorithms.

For the growth cone dataset the CNN did not quite achieve the performance of our previous customized semi-automated algorithm on any of the metrics investigated. However the CNN has the major advantage that only small amounts of training data are required. Hand tracing of such images is prohibitively time-consuming for large datasets, and the semi-automated algorithm still requires up to an hour of human intervention per movie. Thus for large datasets the CNN can produce very good segmentations with only a tiny fraction of the human effort required for the semi-automated algorithm.

While the particular CNN architecture we used is fairly standard, recent further refinements of CNNs have produced excellent performance on a more general set of cell segmentation problems [9]. However, whether similar improvements are possible for growth cone segmentation remain to be determined.

Acknowledgment

We thank Jordan Morse for his contribution to this project.

References

[1] Dickson, B.J. (2002). Molecular mechanisms of axon guidance. *Science*, **298**, 1959-1964.

[2] Mortimer, D., Fothergill, T., Pujic, Z., Richards, L.J. & Goodhill, G.J. (2008). Growth cone chemotaxis. *Trends Neurosci.*, **31**, 90-98.

[3] Goodhill, G.J., Faville, R.A., Sutherland, D.J., Bicknell, B.A., Thompson, A.W., Pujic, Z., Sun, B., Kita, E.M. & Scott, E.K. (2015). The dynamics of growth cone morphology. *BMC Biology*, **13**, 10.

[4] Keenan TM, Hooker A, Spilker ME, Li N, Boggy GJ, Vicini P, Folch A (2006). Automated identification of axonal growth cones in time-lapse image sequences. *J Neurosci Methods*, **151**, 232-8.

[5] Chitsaz D, Morales D, Law C, Kania A (2015). An Automated Strategy for Unbiased Morphometric Analyses and Classifications of Growth Cones In Vitro. *PLoS One*, **10**, e0140959.

[6] Krizhevsky, A., Sutskever, I. & Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems*, **25**, 1090-1098.

[7] Ciresan, D., Meier, U. Masci, J. & Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, **32**, 333-338.

[8] LeCun Y, Bengio Y, Hinton G (2015). Deep learning. *Nature*, **521**, 436-444.

[9] Ronneberger O, Fischer P, Brox T (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597

[10] Coelho LP, Shariff A & Murphy RF (2009). Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms. *Proceedings of IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, Boston, 518-521.

[11] Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, **86**, 2278-2324.

[12] Jarrett, K., Kavukcuoglu, K. & Aurelio Ranzato, M. (2009). What is the best multi-stage architecture for object recognition? *IEEE 12th International Conference on Computer Vision*, 2146-2153.