Deep Learning For Dendritic Spines Detection

Ilyas Aroui
Sorbonne University
4 Place Jussieu, 75005 Paris
Ilyas.aroui@etu.upmc.fr

Abstract

Dendritic spines are tiny proteins found in different shapes and densities on neuronal dendrite, they indicate individual's memory encoding capabilities. Detecting them automatically and in real time is essential to supervise human brain changes throughout different experiences. Several approaches relied on hand crafted features and heavy mathematical modeling for automatic detection [13, 12, 8]. Here we demonstrate that adapting state of the art deep learning models for object detection can localize dendritic spines on fluorescence microscopic images with high accuracy and speed. We show that YOLOv2 [11] and Faster-RCNN [15] work well and in real time for our task and achieve F1-scores > 0.96 outperforming [8] who used the same dataset [4] and reported F1-score = 0.95. We also demonstrate that using shape priors from experts to guide the predictions of a Fully Convolutional Neural Network [7] don't generalize well to dendritic spines detection. Code is available at `https://github.com/ilyasAr/Deep-Learning-for-Dendritic-Spines Detection`

1. Introduction

Dendritic spines are the main receptors in electrical and chemical transmissions from a pre-synaptic neuron's axon to a postsynaptic neuron's dendrite. They extend from dendritic shaft of most excitatory neurons in the mammalian brain. Modern imaging showed that, as brain develops, variation in density, and thus synaptic connectivity is positively correlated with healthy functionality [17]. Furthermore, diseases caused by a disorder in the nervous system, such as Alzheimer or Parkinson, have been found to share the same symptoms of loss and decrease in dendritic spines density [16]. With the huge amount of high-resolution microscopy images generated, it becomes tedious to inspect these structures manually. Even if categorizing them to – mushroom, thin, or stubby might require human supervision, automatic and end-to-end learnable models are a huge support to the community of this field.

Many techniques relied on studying the structure of the dendritic shaft and spines and try localizing the latter in a through pipeline based on different mathematical measurement approaches.

Smirnov *et al*. [8] proposed an open-source tool for automatic dendritic spines detection. They start by cleaning the fluorescent images using a custom thresholding and binarization, then proceed with backbone extraction. From there they create a 221-features vector for each input image, such that features encode intensities and distances from spine and dendrite backbones to different image's segments. Finally, the 221-feature vector is used as input to train a single hidden layer with 20 nodes neural network, where the output classifies each feature as *spine* or *non-spine*.

Su *et al.* [12] followed the same approach of extracting the backbone as a primary step. From a 2D maximum intensity projection (MIP) images they apply an iterative algorithm that refines the extraction of the backbone based on morphological filtering. Then, dendrite boundaries are obtained with respect to the backbone using shorted path techniques, and finally spines are segmented if found outside the boundary of the dendrite.

Mukai *et al.* [13] developed a software called Spiso-3D they worked instead on confocal laser microscopy images. The idea was to make localization of the spines less dependent on brightness using instead eigenvalues, calculated from the Hessian matrix of the image, as an indicator.

These methods need manual adjustment of different parameters throughout the pipeline and tend to be bias toward the dataset they were mostly tested on.

Recently deep learning and especially convolution neural networks (CNN) [18] showed great performance in different visual tasks [6, 19]. Including Medical image analysis research [2, 9, 27].

Deep learning models take raw inputs and do not require hand crafted feature vectors as traditional machine learning algorithms. Therefore, all the mathematical modeling can be learned end-to-end through labeled data instead of being hardcoded. A sub-field of CNN that have been largely used lately, and that assures scale invariance by omitting the dense layers at the end of the CNN is Fully Convolution

neural networks (FCN). Xiao *et al.* [9] achieved promising F1 score >0.80 on large dataset. Using a model inspired from GoogleNet [10] where the down-sampling encode the raw microscopic images to a set of feature and up-sampling map this vector to a probability map with the same size as the input image. To our knowledge, this is the only paper that used FCN to tackle dendritic spines detection.

Here we show that state of the art for general object detection can be adapted well to our task. YOLOv2 [11] is an improvement of the original paper YOLO [5] by improving recall and localization. Their approach relied on grid voting on the feature map achieving, at 67 FPS, 76.8 mean average precision (mAP) on VOC 2007 and 78.6 mAP at 40 FPS.

Faster-RCNN [15], with a different approach from YOLO, proceeds by region proposal networks (RPN) before doing classification and bounding box regression on filtered RPNs. Faster-RCNN is also an improvement to accelerate the learning time of RCNN and Fast-RCNN [20, 14].

Using transfer learning of weights trained on ImageNet [3] or MS-COCO dataset [21] we could achieve great result with few training datasets.

We also demonstrate that SP-CNN [7] failed to perform well on our task as it did on nucleus detection. This matter will be further discussed in section 4.

These approaches are stable as they do not require any parameter tuning during test phase. They also can be generalized to never-seen dataset as it will be shown in section 3.

2. Dataset and preprocessing:

The dataset we used in this project [4] has been created by Smirnov *et al* and used in their paper [8]. They indicated that the microscopic images were collected from a variety of genotypes to ensure an algorithm that works on a variety of morphologies. The dataset is a collection of fluorescent spines and their labels. Each image comes with the coordinates of spine's center measured in pixels with a scale of *15.36 pixels per µm* in both X and Y directions. A sample image and its corresponding ground truth label is shown in Figure. 2.

The main metric we used throughout the comparison was F1-score which is a function of true positives (TP), false positives (FP) and false negatives (FN). Using TP, we can easily calculate FN and FP as follows:

$$FP = \#predicted\ spines - TP$$
$$FN = \#groundtruth\ spines - TP \quad \textbf{(1)}$$

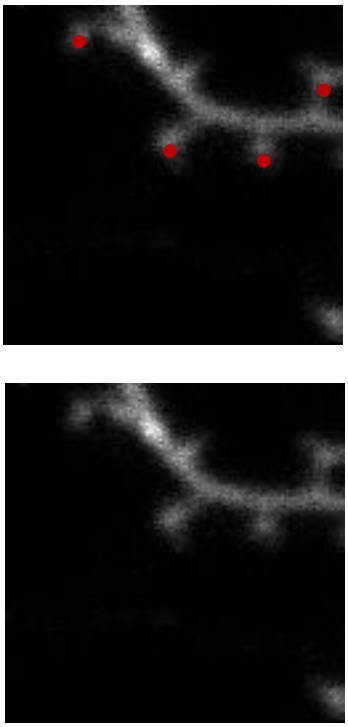

Figure 1. Bottom: A sample microscopic image from the dataset. Top: the ground truth of spine's center indicated in red.

Thus, defining TP rate is a crucial step to guarantee a good metric for comparison in the task of dendritic spines detection. In nucleus detection [7], golden distance of *6 pixels* is taken as the minimum distance between a successful prediction and a ground truth label. Whereas, in most object detection task intersection over union (IOU) is used, and the reported scores depends on the IOU threshold taken to accept or refuse a detection.

As the dataset did not come with the bounding box width and height, the authors suggested that a square of length *15 pixels* can be drawn around the center of a detection and be regarded as a bounding box. Using this suggestion, and testing both TP definitions, we found that IOU and golden distance give the same F1 scores with the YOLOv2 or Faster-RCNN.

We also note that the dimensions of the microscopic images are not unique. Most of the images were saved in rectangular shape of height ranging from *128* to *244 pixels* and width ranging from *128* to *212 pixels*.

We split our data to train test sets ensuring that the test set follow the same distribution of the train set to guarantee the stability of our models' prediction to a new image. From a total of *1836* microscopic images, each contains from 1 to 8 dendritic spines, we split *90%* to train the models and *10%* to test it. We see in Figure 2. that the two partition follow the same distribution as the whole dataset.

These partitions were kept using on the three models we testes to ensure fair score comparisons.

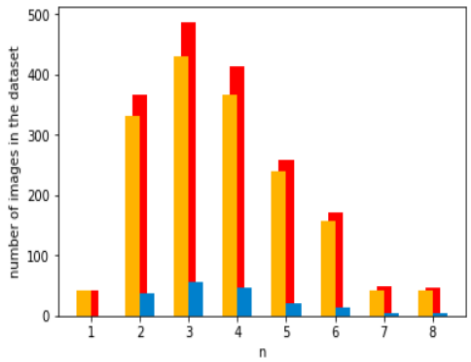

Figure 2. Histogram of the images in the dataset with *n* dendritic spines. Red: the whole dataset. Orange: training set. Blue: test set.

Large CNNs models have over 100 million parameters and require huge amount of data to train from scratch. But with the availability of pre-trained models transfer learning is used. However, to avoid underfitting that may occur from lack of data, we performed different data augmentation on the training set. The techniques consisted of random horizontal flipping, rotation and a slight brightness contrast variation to add some sort of noise to the data. Figure 3.

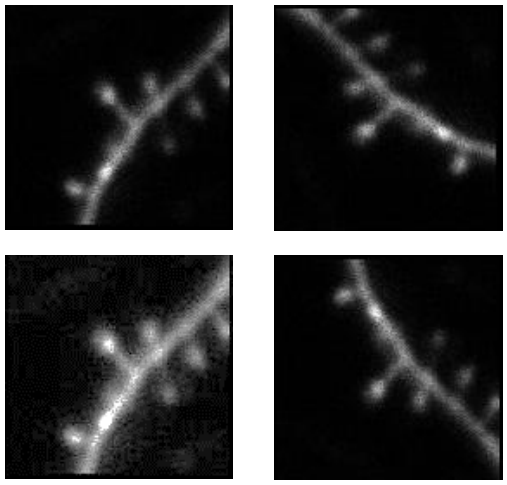

Figure 3. Different data augmentation technics applied randomly to the training set. Top-left: original image. Top-right: +90˚ rotation. Bottom-left: increase of brightness. Bottom right: horizontal flip.

Since CNNs graph once trained is fixed, we had to resize all the images in the dataset to a unique size. It was different for the two models we implemented because they have different down-sampling ratios. This will be discussed separately in the next section.

3. Methods

3.1. Faster-RCNN:

Faster-RCNN [15] is one of the state-of-the-art algorithms in the field of object detection. In this paper, the authors introduce Region Proposal Networks (RPN) to accelerate the training and inference times of their previous work Fast-RCNN [14]. In previous version of this model [20], region proposals were not learned end-to-end. The idea was to share the feature mapper VGG-16 [22] layers' weights during training between the detection and the RPN pipeline. With 300 proposals per image Faster-RCNN achieved *73.2% mAP* on PASCAL VOC 2007.

As the authors indicate, the training is proceeded in four phases:

1- Train RPN model from scratch using IOU between ground-truth bounding box and anchors as a filtering metric at each feature map cell of VGG-16.
2- Initialize the detection pipeline with weights of VGG-16 trained on ImageNet [3] and used the proposed regions from 1 to train the whole Faster-RCNN.
3- Fix VGG-16 weights obtained from phase 2 and retrain the layers that are uniquely related to RPN.
4- Using the proposed layer from phase 3 retrain the whole Faster-RCNN model.

Since the whole network has 3 output vectors during training three loss functions are optimize simultaneously. The detection pipeline has a multi-classification output which uses *categorical cross-entropy* ($Loss_{cls}$) on the predicted objects. The second output is a 4-element vector containing the two coordinated of the object center (x, y), and the height and width (h, w) of the bounding box. This vector is optimized using a modified version of *mean-square-error (MSE):*

$$Loss_{boundingBox} = \sum_{i \in \{x,y,w,h\}} smooth_{L1}(i_{pred} - i_{true}) \quad \textbf{(2)}$$

In which

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

So

$$Loss_{total} = Loss_{cls} + \alpha\, Loss_{boundingBox}$$

The RPN network followed the same concept of multi-class loss in **(2)** treating the proposed regions as the predicted bounding boxes, except for the fact that RPN perform a binary classification with a proposed region being object or non-object. The whole architecture is resumed in Figure 3.

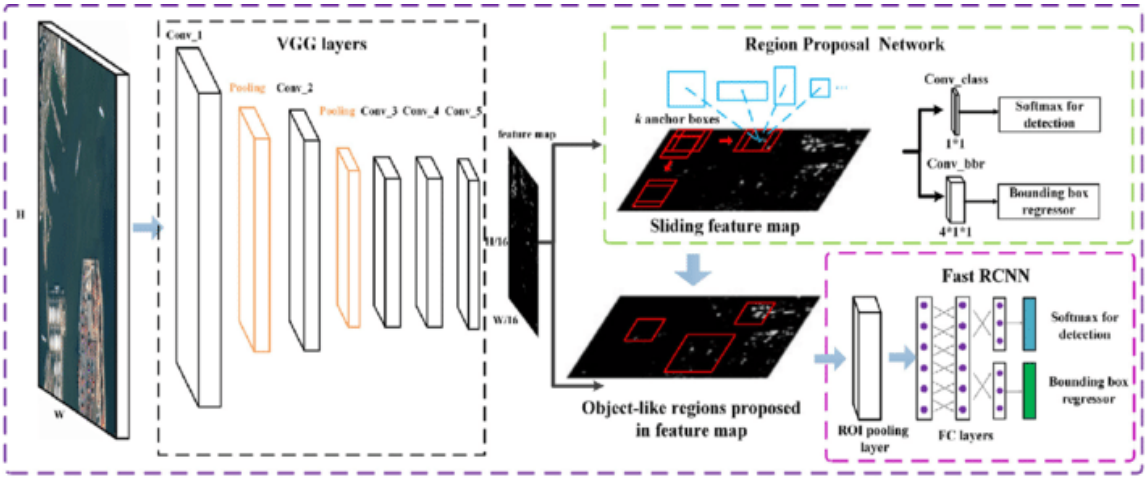

Figure 3. Faster-RCNN architecture using VGG-16 feature mapper. *Source: researchgate.net/ Zhipeng Deng*

Since many other architectures of feature extraction backbone have been published after Faster-RCNN, improving its predictions scores. we used InceptionV2 architecture [23]. The model was initialized first by weights trained on MS-COCO [21] and then the layers of RPN and detection pipeline have been retrained to adapt to our dataset distribution (*transfer learning*). As the ground-truth bounding boxes of the dendritic spines are always squares (*15pixel by 15pixels*) we used one anchor size ratio of unity instead of three different ones as proposed by the paper.

Only horizontal flipping was used for data augmentation during training, with a input image size of *448px by 448px* and a fixed number of proposals of 300.

TensorFlow framework [24] was used to train the model with an INVIDIA GeForce® GTX 1080 GPU of 6G memory. Choosing a learning rate of 0.00002 and a momentum Adam optimizer [25] the training took about ten hours for 9000 steps.

Figure 4. shows that our model is adapting well to the fluorescent microscopic images and the total loss is below 0.05 at around 8900 steps.
After training, the 183 images of the testset were used for predictions and score report.

Figure 5. shows the results obtained from the trained model. We used circles to differentiate some exact overlapped predictions on the true labels.
The predicted boxes are perfect squares because we fixed the anchor size ratios to unity, which pushed the model to learn only square detections.
We see that the model learned well the properties of a dendritic spine from raw images without hand coding them. And this is because the Inception-v2 model knows already low-level features such as: edges, skeleton, intensities gradients…etc.

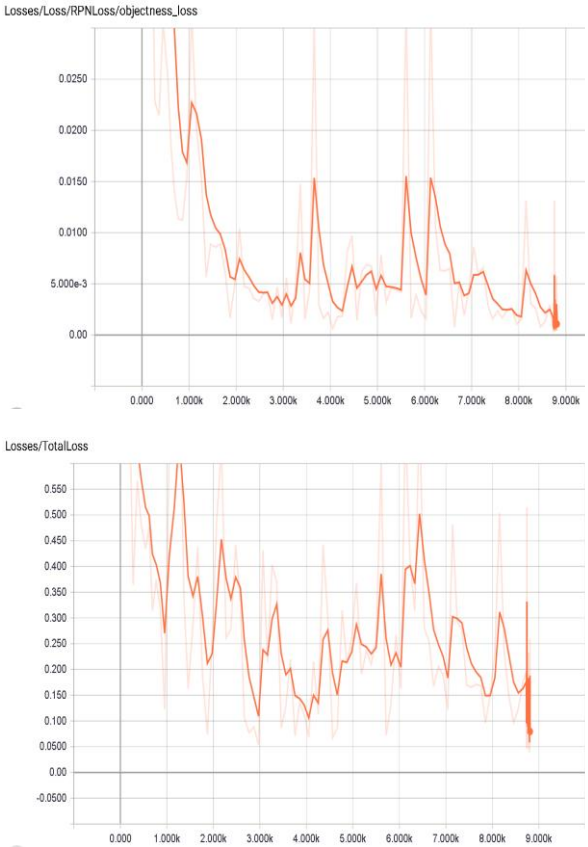

Figure 4. Faster-RCNN losses during training obtained with the supervision of Tensorboard. Top: RPN classification loss. Bottom: total loss.

Thresholding at 0.4 IOU with ground-truth boxes we achieved an F1-score = 0.96 with precision and recall of 0.96 and 0.98 respectively.
Scores and comparisons will be discussed in section 4.

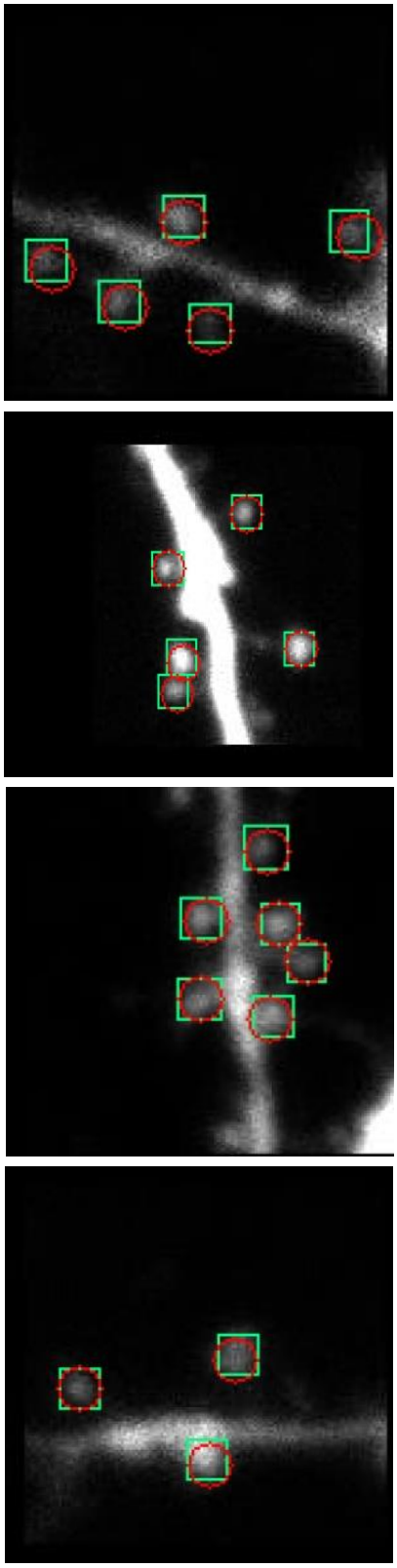

Figure 5. Prediction results of Faster-RCNN on fluorescent microscopic images from the testset. Red: Ground truth. Green: predicted bounding box.

3.2. YOLOv2:

YOLOv2 [11] is another state-of-the-art algorithm with a very high speed. at 67 FPS, the authors reported 76.8 mAP on VOC 2007 and 78.6 mAP at 40 FPS. YOLO follow a different approach from Faster-RCNN, instead of doing region proposals, it divides the feature map to grid cells, such us each cell votes whether it has a center of an object or not.

Instead of using an already existing feature extractor architecture like Resnet or Inception [23, 26], a model called Darknet was proposed with 24 convolutional layers.

Given an image with a list of true bounding boxes, the Darknet model take it as input and down-sample it to $k \times k \times n$ tensor output. Such as:

k: number of grid cells.
n: number of anchors per cell × (5 + number of classes).

Whereas the number 5 correspond to the 4 coordinates of a bounding box and its confidence. So, in all, the output tensor contains, for each grid cell, and for each anchor the confidence of a bounding box, its coordinate, and the conditional probabilities for each class in the dataset.

Just like Faster-RCNN, to find the optimal weights, an *MSE* loss function is applied to predict the bounding box, and categorical cross-entropy loss to predict the probability of a class being inside a bounding box.

Sine our task consists only of one class and to avoid overfitting, we used the tiny-YOLOv2 with 9 convolutional layers instead of 24. The architecture of darknet is inspired from VGG-16 being fully convolutional network. Also, Batch-normalization was included in this version as a form of regularization to tackle recall problem faced in YOLO [5].

The overall YOLOv2 network is shown in Figure 6.

The size of the grid was chosen carefully to adapt to our task. Basically, as the true size of a dendritic spine is given by *15px* and the highest dimension of the images in the dataset is *214.*

$$grid\ size = \lfloor 214/15 \rceil_{odd} = 13$$

We adapted the training dataset to YOLOv2 by making fractions of spines center's coordinates with respect to the image size as well as height and width. Which means despite having different sizes of images rescaling will not hurt the performance as the network is trained on multi-scale inputs at different phases. We did transfer learning by initializing the weights with a darknet model's weight trained on ImageNet [3] and retrained the whole network.

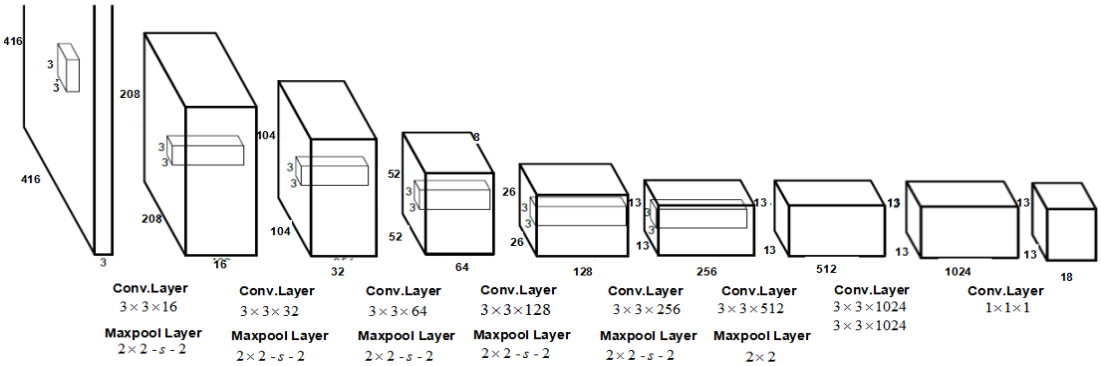

Figure 6. tiny-YOLOv2 architecture with 3 anchors and 1 class. The network is down-sampling the input by 32 using 5 max-pooling layers.

With a learning rate starting from 0.001 and a decay of 0.0005 we trained the model with Adam optimizer for 10k epochs for 2 days using a GeForce® GTX 650 GPU and

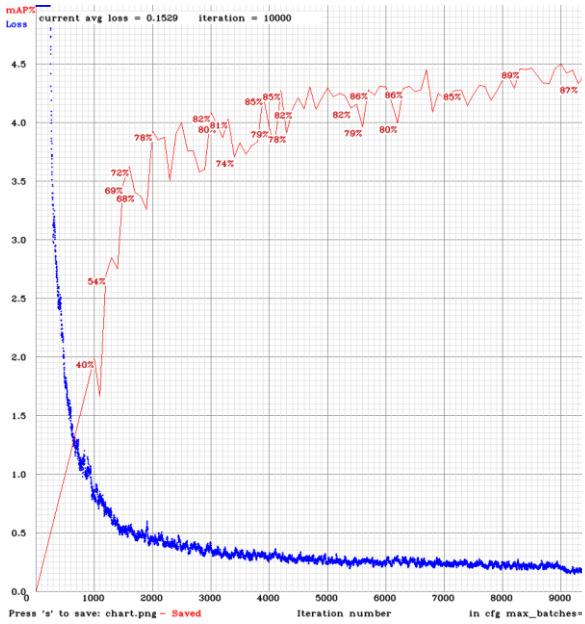

Figure 7. Training loss curve and mAP of YOLOv2 on our dataset. Blue: loss curve. Red: mAP at 0.5 threshold.

CUDA 10.0 Figure 7.

Using the trained model, we inference on the test data Figure 8. We see how almost perfect the predictions at 27ms per image are. We achieved a mAP of 89.9% at 0.5 threshold, it is bigger than the authors score because we are averaging on 1 class only.

We also note that unlike Faster-RCNN the bounding boxes of YOLOv2 are an absolute prediction of a regression, which results in rectangular boxes instead of squares.

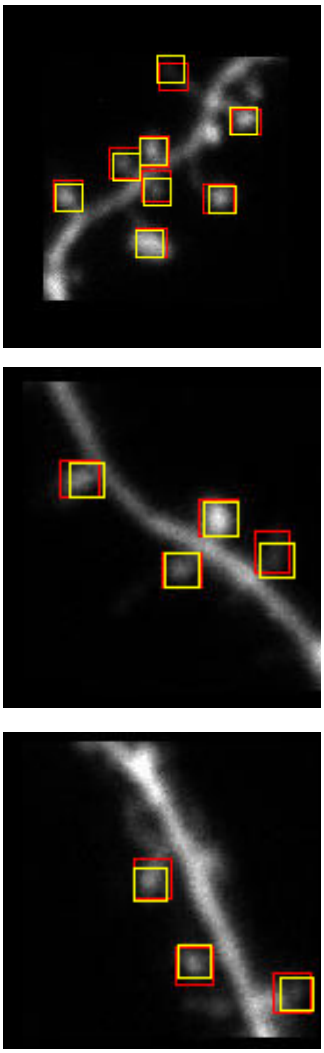

Figure 8. Prediction results of YOLOv2 on fluorescent microscopic images from the testset. Yellow: Ground truth. Red: predicted bounding box.

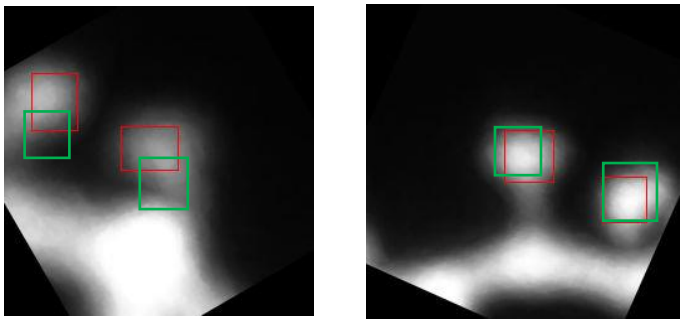

Figure 9. Results of prediction using our trained models on never seen data. Red: YOLO. Green: Faster-RCNN.

Model	Precision	Recall	Image per (ms)	mAP@0.5	F1 score
YOLOv2	96%	95%	29.016	89.9%	95%
Faster-RCNN	96%	98%	58	87.3%	97%
Smirnov *et al.*	94.6%	94.5%	unknown	unknown	94.5%

Table 1 Comparison of different scores from models we trained and a paper that worked on the same dataset.

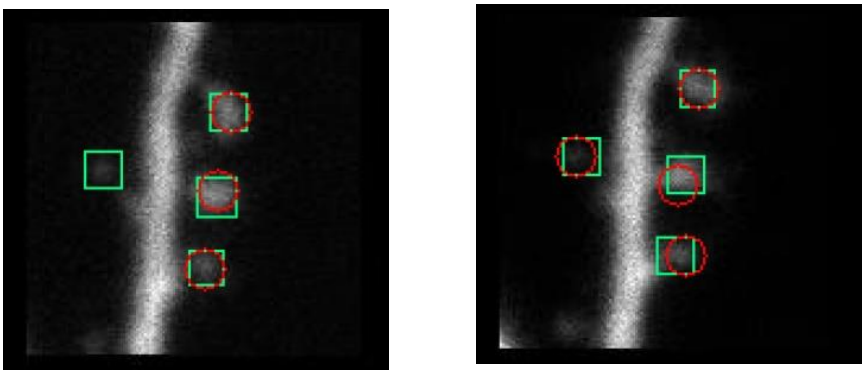

Figure 10 special cases from Faster-RCNN predictions. Green: true label. Red: predictions.

4. Results comparison and discussion:

We see in Table 1. how the two models outperformed Smirnov *et al.* on the same dataset. These models were trained end-to-end with no feature engineering.

We note that the data is very similar, such that the same side of the 3D volume is snapped at different intensities to generate different fluorescents microscopic images. We see however, how the models performed well on a never-seen data. This data is much brighter than our dataset, and we were cautious that the models may consider a spine as if it was a dendrite. But after confirming with my supervisor, the predictions are correct and any expert in the field would have given the same.

Figure 10. shows a False negative given by Faster-RCNN. The reason for that is the low brightness of the spine in that image. As the dataset may contain same images with different pixel intensities, the model predicts the existence of a spine in the same position but on a totally different image in the testset. Left image Figure 10.

In addition to time and hardware need, such deep learning models may require a lot of parameter tuning. In fact, the result shown in this project represent the best we achieved after tuning several times the model.

5. Shape Priors:

5.1. **Application:**

Tofighi *et al* [7] used shape priors to train a CNN model for nucleus localization. The idea was to propose a regularization term, in function of the output, that is targeted at penalizing false positive, while pushing the model to predict inside cell nucleus boundary.

In short, they trained a 6 layer fully convolutional neural network using *MSE* loss between the true labels and the predictions, and a regularization term that is define as follows:

$$R = \sum_{i=1}^{64} \left\| (g_p(\hat{y}) \circ \hat{x}) * S_i \right\|_2^2 \qquad \textbf{(3)}$$

such that:

$\hat{x}$: is the edge of the input image. Figure 11.
S_i: are the shape priors defined by experts. Figure 12

So, the algorithm is as follows, we threshold the output $\hat{y}$, apply on it g_p a max-pooling operation with wind of size p and then we perform an element-wise multiplication with $\hat{x}$ and convolve the result with the all the shape priors S_i.
Which means we are trying to maximize the correlation with the expected shape, so the regularization term is subtracted from the *MSE* instead of being added.
Finding a good-looking edge was a difficult task as the fluorescent images were noisy. After doing grid search on different parameters and applying bilateral filtering we archived Figure 11.
After implementing the algorithm in TensorFlow, with defining this custom loss and Adam optimizer. We were stuck in a singleton loss value and the model seem not to converge and further.

Which resulted in zero predictions, and the network didn't learn the mapping. Figure 13.

Figure 11. edges of a dendrite using bilateral filtering and Canny-edge.

Figure 12. a shape of a dendritic spine taken from the edges of the training set for test the algorithm.

5.2. Addressing issues:

There are two possible explanation for these results:

1- The idea is not well adapted to our case as Tofighi *et al* were trying to detect the nucleus inside the cell, which means the edges here play the role of a boundary. Whereas in our case the edges for the spine, which is what we want to localize.

2- Their CNN model implementation is very poor. No max-pooling, dropout or batch-normalization were used. Which makes the network heavily dependent on the correlation term to avoid overfitting. Unfoundedly in our case the regularization term was not doing his job, so the model was not regularized properly.

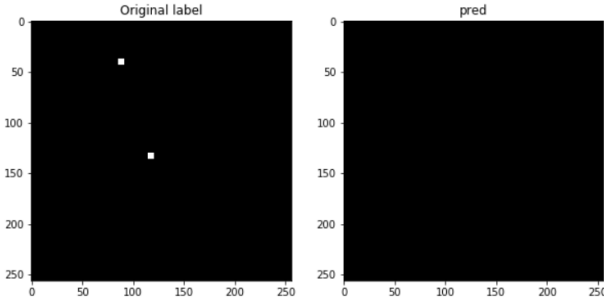

Figure 13 results of SP-CNN on our dataset. Left: original label. Right: prediction.

6. Acknowledgement:

I would like to thank my supervisor Dr. Suvadip MUKHERJEE, a researcher in Institut Pasteur, for guiding me through this project, answering my questions, and pushing me forward to get exposed to a lot of research in the field. I did an intensive literature survey and paper reading through this journey, in which I gained valuable knowledge.

7. Conclusion and future work:

Deep-learning models, with less labor can perform well on many visual recognition problems. Medical image analysis is gaining attention these recent years and we may see a lot of hand crafter features and mathematical modeling replaced by a feature mapper model. However, we can still combine the two to make our model learn either better or quickly as the work by Saha et al. [27] for mitosis detection, where they trained a neural network by concatenating features from auto-encoder and handcrafter features.

This project did not consider the fact that there are three types of dendritic spines: mushroom, thin and stubby.
So, for a future work, we are planning on extending the detection to 4 classes instead of just one. We are also thinking about using Mask-RCNN or PFPNet [6, 1] to segment the spine from the dendrite for further analysis.

Moreover, we would like to gather more data from Pasteur institute to test our models, and thus have a less biased predictions. In case the performance is stable with a the new data, why not using the models in the labs as a preprocessing tool?

References

[1] S.-W. Kim, H.-K. Kook, J.-Y. Sun, M.-C. Kang, and S.-J. Ko, "Parallel Feature Pyramid Network for Object Detection," in Computer Vision – ECCV 2018, 2018, pp. 239–256.

[2] I. Suleymanova et al., "A deep convolutional neural network approach for astrocyte detection," Scientific Reports, vol. 8, no. 1, p. 12878, Aug. 2018.

[3] J. Deng, W. Dong, R. Socher, L. Li, and and, "ImageNet: A large-scale hierarchical image database," in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[4] M. Smirnov and T. Garrett, "Labeled Dendritic Spines - Training Data." 17-Apr-2018 https://figshare.com/articles/Labeled_Dendritic_Spines_-_Training_Data/6149207

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," arXiv:1506.02640 [cs], Jun. 2015.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," arXiv:1703.06870 [cs], Mar. 2017.

[7] M. Tofighi, T. Guo, J. K. P. Vanamala, and V. Monga, "Deep Networks with Shape Priors for Nucleus Detection," in 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 719–723.

[8] M. S. Smirnov, T. R. Garrett, and R. Yasuda, "An open-source tool for analysis and automatic identification of dendritic spines using machine learning," PLoS ONE, vol. 13, no. 7, p. e0199589, 2018.

[9] X. Xiao, M. Djurisic, A. Hoogi, R. W. Sapp, C. J. Shatz, and D. L. Rubin, "Automated dendritic spine detection using convolutional neural networks on maximum intensity projected microscopic volumes," J. Neurosci. Methods, vol. 309, pp. 25–34, Nov. 2018.

[10] C. Szegedy et al., "Going Deeper with Convolutions," arXiv:1409.4842 [cs], Sep. 2014.

[11] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," arXiv:1612.08242 [cs], Dec. 2016.

[12] R. Su, C. Sun, C. Zhang, and T. D. Pham, "A novel method for dendritic spines detection based on directional morphological filter and shortest path," Comput Med Imaging Graph, vol. 38, no. 8, pp. 793–802, Dec. 2014.

[13] H. Mukai et al., "Automated Analysis of Spines from Confocal Laser Microscopy Images: Application to the Discrimination of Androgen and Estrogen Effects on Spinogenesis," Cereb Cortex, vol. 21, no. 12, pp. 2704–2711, Dec. 2011.

[14] R. Girshick, "Fast R-CNN," arXiv:1504.08083 [cs], Apr. 2015.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," arXiv:1506.01497 [cs], Jun. 2015.

[16] M. Knobloch and I. M. Mansuy, "Dendritic spine loss and synaptic alterations in Alzheimer's disease," Mol. Neurobiol., vol. 37, no. 1, pp. 73–82, Feb. 2008.

[17] P. Penzes, M. E. Cahill, K. A. Jones, J.-E. VanLeeuwen, and K. M. Woolfrey, "Dendritic spine pathology in neuropsychiatric disorders," Nat. Neurosci., vol. 14, no. 3, pp. 285–293, Mar. 2011.

[18] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6.

[19] S. Hong, T. You, S. Kwak, and B. Han, “Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network,” arXiv:1502.06796 [cs], Feb. 2015.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” arXiv:1311.2524 [cs], Nov. 2013.

[21] T.-Y. Lin et al., “Microsoft COCO: Common Objects in Context,” arXiv:1405.0312 [cs], May 2014.

[22] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv:1409.1556 [cs], Sep. 2014.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” arXiv:1512.00567 [cs], Dec. 2015.

[24] https://www.tensorflow.org/

[25] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980 [cs], Dec. 2014.

[26] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv:1512.03385 [cs], Dec. 2015.

[27] M. Saha, C. Chakraborty, and D. Racoceanu, “Efficient deep learning model for mitosis detection using breast histopathology images,” Computerized Medical Imaging and Graphics, vol. 64, pp. 29–40, Mar. 2018.