Quantifying Mammalian Learning: Large-Scale Detection of Dendritic Spines

Ivaylo Bahtchevanov
Stanford University
450 Serra Mall, Stanford, CA
ivaylogb@stanford.edu

Seth Hildick-Smith
Stanford University
450 Serra Mall, Stanford, CA
sethjhs@cs.stanford.edu

John Lambert
Stanford University
450 Serra Mall, Stanford, CA
johnwl@stanford.edu

Abstract

We demonstrate that by training a CNN on a sliding window and by altering the distribution of classes we can predict the location of dendritic spines in microscopies of stained mouse neurons. We show that, for our specific task of detecting the location of spines on a dendrite, transfer learning is not well-suited to the task, despite the success achieved by the YOLO[14] and Faster-RCNN[15] algorithms on the PASCAL VOC 2007, MS COCO, and ILSVRC datasets. Our task is complicated by incomplete gold standard labellings; to make progress without editing the ground truth labeled by neroscientists in [4] we simplified our problem into a sliding window classification on down-sampled data. Our simple three-layer CNN shows strides towards replacing laborious hand-annotation of dendritic spine structures with automatic detection by a computer. This classifier achieves the best results, and even succeeds in finding spines not originally labelled by humans. Our goal is to build out a model that can be highly scalable and useful for researchers in the future.

1. Introduction

1.1. Receivers for Synaptic Connections

In mammals, most excitatory synapses are indicated by structures extending from dendritic shafts called dendritic spines. As a brain develops, variations in the density of dendritic spines and their morphology indicate the creation, alteration and destruction of neuron-to-neuron synapses. This plasiticy of dendritic spines allows connectivity within neuronal circuits to evolve as an animal learns. [13] "Dendritic spines may be tiny in volume, but are of major importance for neuroscience." Present in mammals ranging from mice to humans, "they are the main receivers for excitatory synaptic connections, and their constant changes in number and in shape reflect the dynamic connectivity of the brain." [2] Not only are the density of these structures positively correlated with neural plasticity[5], they are negatively correlated with the incidence of neuropsychiatric disorders, including autism spectrum disorders, schizophrenia and Alzheimers disease [13][9].

1.2. A Shift from Hand-Annotation to Scalable CNN-Annotation

In close collaboration with Stanford's BIO-X Research Program, our work aims to advance the use of computer vision into dendritic spine research. Dr. Maja Djurisic, a neuroscientist affiliated with the program, and our primary contact, has made substantial progress using the density of dendritic spines on mouse neurons to understand the cognitive development of mammals. Her current research, an on-going work of 6 years, seeks to better understand the role of dendritic spine density in cognitive ability/learning. We were inspired by an opportunity to apply computer vision to aid her research's potential to contribute to the field of neuroscience, especially in a subject linked to the understanding of neuro-degenerative diseases in humans. Currently, her work has been hampered by the slow process of hand annotating the microscopies. Further difficulties arise in actually determining the location of the spines, as the human eye is not well-equipped to detect these inconspicuous structures within the images. We aim to use Dr. Djurisic's dataset to give her work the last push she needs to complete her research: we have developed a model for detecting and labelling dendritic spines, which can further be used to analyze the density.

1.3. Problem Structure

The input to our algorithm is a Z slice from a confocal microscopy in the form of a 1024x1024 RGB image. For different approaches, we used different data pre-processing. We then use a CNN to output the four coordinates of a bounding box circumscribing each dendritic spine.

We utilize two evaluation techniques. First, we test our mAP as a quantitative result. However, fundamentally our work aims to augment the work of Dr. Djurisic. Her approval of our work as useful for her research will be the final evaluator of our work.

Given sufficient time, this model could be extended to provide dendritic spine density, the crucial metric in Dr. Djurisic's work.

2. Related Work

2.1. Dendritic Spine Detection

2.1.1 Hand Annotation

All the dendritic spine labelings used in work by the lab we have partnered with have been made by hand. Some attempts have been made by the Shatz Lab to experiment with automated detection algorithms but none have been successful enough to be applied.

2.1.2 Previous Automated Neuron Mircoscopy Analysis

In 2007, researchers from Harvard and Tsinghua University published a paper aptly named, "Automatic Dendritic Spine Analysis in Two-Photon Laser Scanning Microscopy Images."[1] They used an unsharp mask filter to regularize image intensity. The spines are detected based on their "skeleton image" and segmented by type of spine according to width-based criteria. The width criteria are derived from a common morphological feature of the spine. They use blind de-convolution to correct for heavy blurred images. Their research achieves recall of 88.06% and precision of 86.47%. Although highly successful, we believe that further application of deep learning will be enable higher levels of both precision and recall at the optimal threshold.

2.1.3 "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images"

In [3], the authors create a model to drive automatic segmentation of neuronal structures depicted in stacks of electron microscopy images. Their work is part of a larger process of efficiently mapping 3D brain structure and connectivity necessary for understanding how these images translate to biological processes. The network effectively computes the probability of a pixel being a membrane, using as input the image intensities in a square window centered on the pixel itself. Each image is then segmented by classifying all of its pixels as a deep neural network trains on a different stack with similar characteristics (the membranes here were manually annotated).

Their DNN performed 4 stages of convolutional and max-pooling layers of several fully connected. The output layer is always a fully connected layer with one neuron per class. The last layer uses a softmax activation to guarantee each neuron outputs a probability of a particular image belonging to a class. Because each class is equally represented in the training set but not in the testing data, the network outputs cannot be directly interpreted as probability values; instead, they tend to severely overestimate the membrane probability. To fix this issue, they apply a polynomial function post-processor to the network outputs.

2.2. Detection Algorithms

2.2.1 "Object Detection and Localization Using Local and Global Features"

In [12], Kevin Murphy and Antonio Torralba propose a methodology for localization and object detection by examining both local and global features, demonstrating how this combination can lead to substantially improved detection rates. The most common tool for localization is to slide a window across the image (possibly at multiple scales), and to classify each such local window as containing the target or background. While this strategy is proved to be relatively successful, problems arise when the target is very small or highly occluded. In these circumstances, the model can gain substantial information by examining the context around the image (gist-based priming) and use object detection (determining the number of instances of an object in an image).

They apply a adaptive boost ("Adaboost") algorithm[6] for boosting as follows: The training data is computed by creating a set of features for each labeled image and sampling the resulting filters at different locations (Once the filter is near the center of the object and 20 random locations to create one positive and 20 negative examples). These feature vectors are then passed to the classifier, and they perform approximately 50 rounds of boosting. Here, the majority of the computation time is spent creating the feature vectors and normalizing through cross correlation. Each classifier is trained independently, and it is applied to a new image at various scales to find the location of the strongest response. The output of the boosted classifier is a score for each patch.

2.2.2 "You Only Look Once": YOLO Detection

The "You Only Look Once" model [14] considers bounding boxes and can achieve a mean average precision (mAP) of 63.4% on the PASCAL VOC 2007 dataset. The model frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since YOLO's entire detection pipeline operates as a single network, it can be optimized end-to-end directly on detection performance. The framework was attractive for our problem because it uses a global approach (rather than sliding window and regional analysis) to encode contextual information about classes when making predictions.

The YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1

convolutional layers reduce the features space from preceding layers

It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities. At test time, the conditional class probabilities and the individual box confidence predictions are multiplied together as follows:

$$Pr(Spine|Object) \times Pr(Object) \times IOU_{gt} =$$
$$Pr(Dendrite) \times IOU_{gt}$$

Equation 1

Notably, YOLO, like work done in [18], has real time frame rate at test time.

2.2.3 "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"

As noted in [10], the current state-of-the-art detection algorithm is the Faster RCNN method, submitted 4 Jun 2015. The Faster R-CNN VGG-16 Network, developed by [15], can achieve a mAP of 73.2%. In [15], the authors build upon and accelerate the learning pipeline of their previous Fast-RCNN model. Their goal is detection: propose a region of interest, and then classify that region according to the classes enumerated in the PASCAL VOC 2007, 2012, and MS COCO datasets. To cite the authors, their "Region Proposal Network," "shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals." This enables the acceleration of the learning of the ConvNet. The previous Fast-RCNN model is used for detection of these region proposals. The authors rely upon so-called "anchors" and "Intersection-over-Union" to facilitate and evaluate their region proposals.

3. Methods

3.1. Application of Faster-RCNN

We applied Ross Girshick's implementation of Faster-RCNN in Python, with hopes of achieving a similar State-of-the-Art mAP (72% on VOC PASCAL 2007) on our task. Faster RCNN presented a number of challenges: notably, the necessity of creating an XML file for each annotated image in our dataset, and the lack of a suitable GPU for training Girshick's best model, the VGG-16Net. Amazon's AWS instances offer up to a maximum of 4 GB per graphics processing unit (the NVIDIA GRID K520 GPU), and Girshick's standard mini-batches require up to 8 GB of GPU memory. Instead of using the customary 20 classes from the PASCAL VOC dataset, we chose two classes – dendritic spine, and background.

We began with transfer learning, since our dataset is limited in size. We hoped to use the lowest layers of a Zeiler-Fergus Net [17] (ZFNet, trained at MSRA) to accurately detect edges and blobs, cut out the last convolutional, pooling and fully-connected and layers. We planned to load pre-trained weights from open-source .caffemodel files for our weight blobs. [7]

As a point of reference and a baseline, we first trained a model from scratch and tested on 160 of our images.

In order to encode a preference for some weights W over all other possible weights, and in order to discourage large weights, we extend the cross-entropy data loss to include an L2 regularization penalty:

$$L = \underbrace{\frac{1}{N}\sum_i L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

The data loss is the average loss L_i over all examples. $R(W)$ is defined as the sum of the squared elements of W:

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

Note that in the expression above, the regularization function is not a function of the data, it is only based on the weights [11].

The cross-entropy loss per training example n can be written as:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

or, equivalently, as:

$$L_i = -f_{y_i} + \log\sum_j e^{f_j}$$

We use the notation f_j to mean the j-th element of the vector of class scores f [11].

$$ConvolutionalPadding = \frac{(ConvFilterSize - 1)}{2}$$

3.2. Application of YOLO

We applied the work of [14] to our problem. Dendritic spines are incredibly difficult to notice on their own, and an understanding of the surrounding stack is crucial to their detection. We assumed that YOLO would be an effective algorithm for our detection problem.

However, once we applied and implemented their framework, we understood this was not a great approach to the problem.

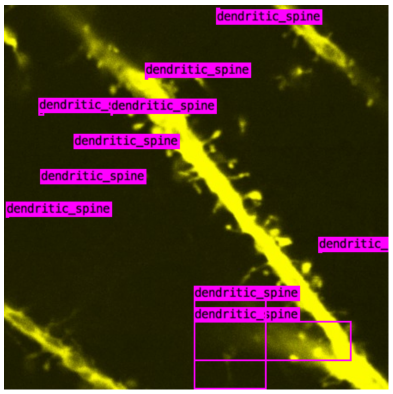

Figure 1. The YOLO algorithm failed to accurately place bounding boxes around dendritic spine structures

3.3. Our Custom ConvNet Architecture

Our two most promising results came from smaller networks. We built two conv-nets with the architecture shown below:

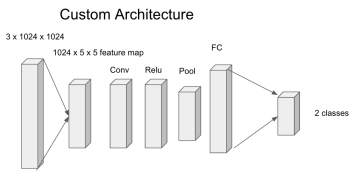

Figure 2. DOWNSAMPLE-CONV-RELU-POOL-FC-SCORES-SOFTMAX.

3.3.1 Sliding Window with 3-Layer and 7-Layer CNN

To solve the problem of arbitrary detection of dendritic spines we developed a sliding window detector. We first reduced the size of our images to 512×512 and splintered the microscopies into 256 16×16 sub-images. We labeled those images as either containing the center point of a dendritic spine or not and then trained two models on this classification problem. Our first architecture was a three layer shown in Figure 2. In hopes of improving results, we also applied this classification task to a 7 layer architecture: (conv - batch_norm - relu - pool)×3 - (affine - batch_norm - relu)×4 - softmax.

At test time we apply the best model to a sliding window test across the tested image and placed bounding box around all sub-images which were predicted to contain dendritic spines, see Fig. 14.

3.3.2 Classification as Localization

The large majority of slices of the microscopies were labeled with a single dendritic spine (see Fig. 8). As such the majority of our detection problem can be simplified to a regression problem: locating the dendritic spine in the image. We therefore removed the softmax classification loss from the head of the 3-layer architecture (Fig. 2) and replace it with a euclidean loss layer. We feed in the class scores into the L2 Norm as follows:

$$E = \frac{1}{2N}\sum_{n=1}^{N} ||\hat{y}_n - y_n||_2^2$$

We compute the derivative with respect to scores and backpropagate through the network.

4. Dataset and Features

4.1. Available Dendrite Microscopy Images

117 images from June 2 2011, 566 images from June 3 2010, 103 images from June 3 2011, 80 images from June 9 2011, 471 images from June 10 2010, 368 images from June 11 2010. Altogether, we have 1705 microscopy images in .tif format. These images represent slices along the z-axis (which we will refer to as the “z-stack”).

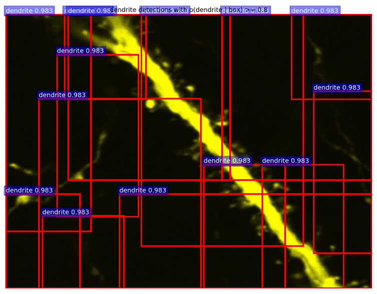

Figure 3. Using Girshick’s alternating optimization algorithm alternates the training of the Region Proposal Network and Fast RCNN. Attempt to overfit 20 images was unsuccessful, although loss dropped to 0.3-0.4 from an original value of 1.5. Note that all boxes have same predicted confidence, and that boxes are far too large for the task of predicting small dendritic spines

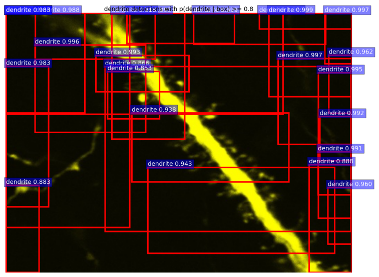

Figure 4. Use of Girshick’s end-to-end training optimization algorithm was unsuccessful. Only improvement is that bounding boxes now have different predicted confidences

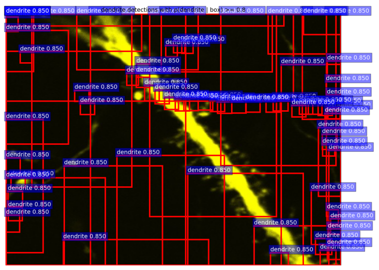

Figure 5. Attempt to fine-tune ZF Net [17] by decreasing learning rate to 10% of original. Note that a smaller learning rate generated smaller, more refined boxes at some points. Attempt to overfit 20 images was not successful

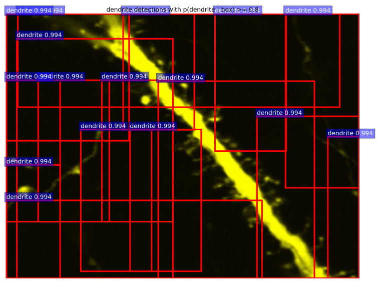

Figure 6. Use of a training set of 171 images, as opposed to 20 images, did not improve accuracy on the training set

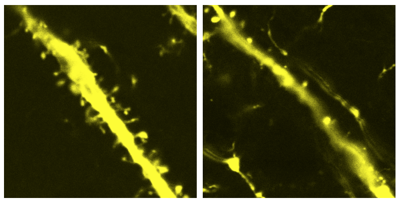

Figure 7. Microscopy of Unlabeled Mouse Dendrites [4]

4.2. Hand-Annotation

The amount of fully-annotated (i.e. labelled) images is a small fraction of the total images. In September 2015, Dr. Djurisic, along with her colleagues, annotated the locations of all of the dendrites in several of the microscopy images.

4.3. Ground Truth

Our ground truths represent x-y coordinates for the center of mass of the spines. Images range from having zero spines to 18 within one frame. We have labels for the locations of the indicator structures. Note that the labeled image is a amalgamation of an entire z-stack.

4.4. Preprocessing

The preprocessing of Dr. Djurisic's data was a significant task. We were given a Dropbox directory containing dozens of subdirectories. We chose the directories that contained .xls-format spreadsheets, where each row contained the XY-Coordinates of one dendritic spine in one particular image. We parsed the subdirectories in Python, choose the csv files that contained files pertaining to dendritic spines, mapped each row to its corresponding image, and inserted these labeling structures for our various algorithms. The labels had to be parsed into an XML-annotation-tree format that was compatible with the Faster RCNN framework as well as a .txt file annotation unique to the YOLO code base. The Faster RCNN source code relies on a dataset structured according to the PASCAL VOC 2007 dataset. Furthermore, we converted our image files from .TIF format to .JPG format. Originally, we believed that we had 879 labelled images (94 of which were annotated by Dr. Djurisic herself). These corresponded with about 20 full Z-Stacks of 50 images each. However, the number of annotated images was far fewer. For the Faster RCNN training set, we discovered 171 unique, annotated images, containing from 1 to 18 dendritic spines each. See Figure 8.

4.5. Distribution

Our dataset contains just two classes: background and dendritic spine.

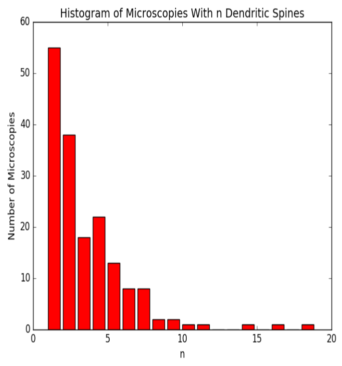

Figure 8. The distribution of dendritic spines per image in our dataset has a heavy left skew.

We had to wrestle with the problem of sparse labeling. Some of the images, which appear to contain up to a dozen dendritic spines, have a gold standard annotation with only one dendritic spine. For instance, the microscopy predicted on in Figure 14. Although it would help, we specifically chose not to alter the data set by adding many additional XY coordinate-labels per image. We felt that such an approach does not reflect integrity to the labels Dr. Djurisic's team generated.

4.6. Conversion of Units on Labels to Pixels

Microscopies performed on equipment on Leica Microsystems are measured in microns. Thus, we converted microns to pixels according to the equation:

$$\frac{pixels}{1micron} = \frac{1meter}{1.00 \times 10^6 microns} \frac{1024pixels}{2.976 \times 10^{-5} meters}$$

$$\frac{pixels}{1micron} = 34.4086 \frac{pixels}{micron}$$

We assume that each dendritic spine is roughly equivalent in shape and size. Assuming that our input image has been downsampled from 1024x1024x3 to 128x128x3, we demonstrate that a window size of 8x8 pixels can capture the shape and intensity of a dendritic spine.

5. Experiments, Results, and Discussion

Analysis of Faster-RCNN Results The Faster-RCNN model was ill-suited to our task. The model was able to identify objects and people extremely well in images that we fed in to its demo model. These preliminary images were pictures of our family, friends, and automobiles – very similar to the contents and classes of the 2007 VOC PASCAL dataset. The train.prototxt of the Faster RCNN pipeline contains 28 Caffe layers – excessively complex for our purpose. Yet when the task of over-fitting even one image with randomly initialized weights, the model delivered inaccurate results. See Figures 3, 4, 5 and 6 for images of Transfer Learning.

The decreasing loss shows that the model learned to reduce it's loss function, but the resultant predictions show that it's learning did not aid our predictions. (see Figure 9).

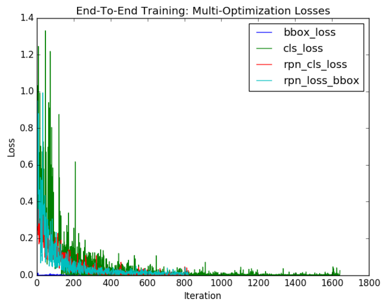

Figure 9. Faster RCNN Losses Produced While Training via End-To-End Optimization.

5.1. Training the ConvNet With a Euclidean Loss Layer

We implemented a model with a regression head to localize a single spine in the microscopy (see Figure 10). After 30,000 iterations the regression model was able to achieve a RMSE of 9.623552. After 10200 iterations, the sliding window classifier achieved 99.96% accuracy by always predicting the "background" class.

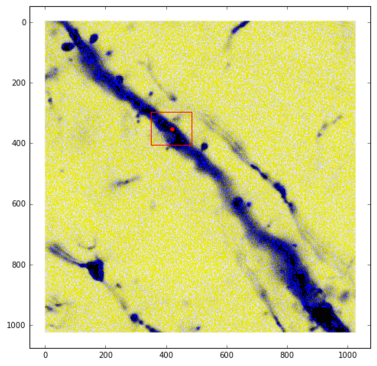

Figure 10. Spine predictions with Regression model

5.2. Training the Sliding Window CNN

We used the Adam [8] update for our gradient descent algorithm (Adam and RMSProp seemed to work best with sliding window). Only when we lowered the learning rate to about 1.0×10^{-6} were we able to stabilize the loss and not see divergence to infinity immediately. We trained for 100 epochs; however, the learning occurred in the first 5 epochs on average. We set batch size equal to 10. Because we used Cython im2col-optimized layers for convolutions, we down-sampled our images from dimensions of 1024x1024x3 to 1/8th of the resolution: 128x128x3. Computation with full-resolution pictures on our CPUs was infeasible.

We used a three layer ConvNet, initializing the weights with a weight scale of 1.0×10^{-3}. Our hidden dimension in the fully-connected layer was 100 neurons wide. We used a regularization constant of 1.0×10^{-3} in order to prevent over-fitting.

Our images, cut down to the size of the sliding window, were (3 x 8 x 8) in dimension. We used 32 filters in the convolutional layer, with a filter size of 5 x 5. Furthermore, we used a stride of 1 and a padding of 2.

$$ConvolutionalPadding = \frac{(ConvFilterSize - 1)}{2}$$

Input and Output Dimensions are preserved during the convolutional transformation when the padding is computed above.

In our pooling layer, we use a pooling filter size of 2x2 and a stride of 2 in order to down-sample the width and height of the input by half. Our Softmax classifier performs Maximum-Likelihood-Estimation (MLE) over two classes.

5.3. Rebalancing Class Distribution

We found it necessary to alter the class distribution. After 10,200 iterations of stochastic descent, our sliding win-

	Dendritic Spine	Background
Without Sampling	256	1
With Sampling	2	1

Table 1. Class Distribution: Number of Labels Per Class, Per Image

dow classifier achieved 99.96% accuracy by learning to always predict the “background” class. Since we break each image (of dimension 1024x1024x3) into a 16x16 grid of smaller images, we end up with 256 window patches per original image. On average, Dr. Djurisic’s team annotated 1-3 dendritic spines per image, leaving us with a skew of up to 1 dendritic spine. By sampling uniformly from the background class with probability $\frac{2}{256}$ and by keeping all dendritic spine-labeled images, we dramatically improved our CNN model and reduced the class bias skew.

5.4. 3-Layer CNN Result

Learning occurs within 5 epochs, or about 200 iterations (almost instantaneously), after which the training set accuracy plateaus indefinitely.Down-sampled, then only use 20 images. We used this training set of 20 images for our validation set and test set.

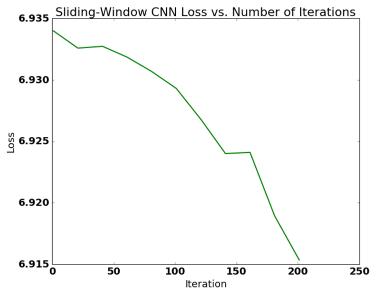

Figure 11. Loss Decreases As a Function of Iterations While Training 3-Layer Conv-Net

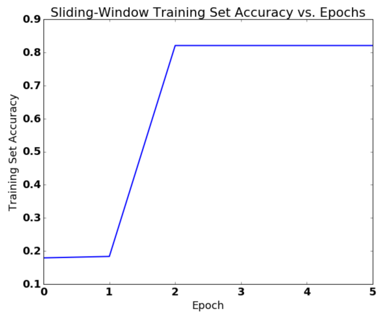

Figure 12. Training Set Accuracy Increases At Inflection Point, Then Plateaus

This CNN model is the first one to detect spines at a elementary level of proficiency.

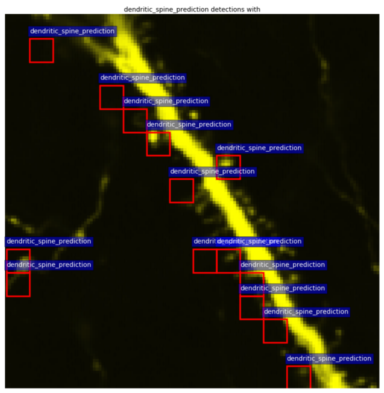
dendritic_spine_prediction detections with
dendritic_spine_prediction

Figure 13. The sliding window predicts with 100% recall for this image and reasonable accuracy when accounting for missing labels. Only one of dendritic spine is labeled in the ground truth for this image

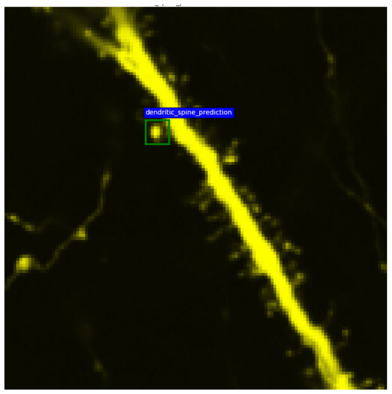

Figure 14. Ground Truth Label for Image, Assuming Constant 8x8 pixel Window Size centered around Dr. Djurisic’s XY-Coordinate Labels

Interestingly enough, adding in a stride and requiring the XY coordinate to sit in the middle 1/2 of the image in the x and in the y dimensions did not improve the model. In general, we found a larger stride to be more beneficial for speed but will hurt accuracy.

At test time, our sliding window achieved a 15.73% mAP. The relatively low precision was counter balanced by nearly perfect recall. We settled on these results as successful due to the fact that our labeling (as we have previously discussed) was very sparse and we attempted to capture spines that were not necessarily labeled in the gold standard labels.

At test time, our classification as regression problem achieves a rMSE of 9.62. This indicates that on average we are within about 10 pixels of the correct XY coordinate of the center of mass of the spine.

For both classification and regression tasks, the 7-layer conv-net that we applied to the same tasks achieved poorer results than the 3-layer conv-net. We believe this was a function of the very primitive nature of our images. Further complexities in our model did not better fit a “simple” problem.

5.5. CONV and FC Weight Visualizations

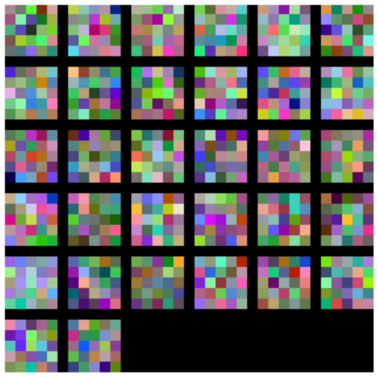

Figure 15. Visualization of the Learned Weights of the Sliding-Window Convolutional Layer

6. Conclusions and Future Work

6.1. Analysis of Learning Capability

It is worth noting that the Sliding-Window not only learns the annotations but also understands when the spines are not present at all. On the training set, it will learn the existing labels and find additional labels that humans were not able to label (or even, in some cases, perceive). This aspect makes the approach highly scalable and effective as a tool for neuroscientists. In Figure 14, the CNN is trained on 1 label for this image, but predicts a total of 7 correct dendritic spines, out of 14 predicted bounding boxes, thus having 50% accuracy.

6.2. Further Work: Expansion of Training Data Set

We used a fraction of the annotated images we had extracted, for our model. Given more time, we would have trained our model on all images, which we anticipate would improve our model’s ability to generalize on novel data. Another approach would have been to fully annotate a segment of the data-set in order to increase the number of fully labeled images as well as produce more generalizable representations for spines.

6.3. Further work: Combining Outputs of Regression and Classification Channels, as per OverFeat

We believe we could improve our sliding window algorithm by implementing the OverFeat CNN architecture in 16 [16]

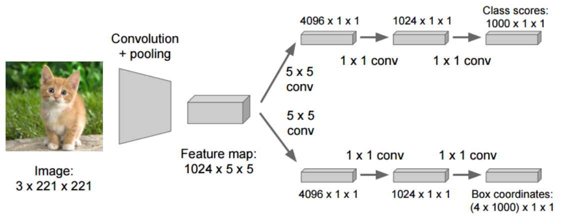

Figure 16. OverFeat [10] converts all fully-connected layers into convolutions

Since the sliding-window approach computes an entire pipeline for each window of the input one at a time, ConvNets become very efficient when applied in a sliding fashion since they inherently share computations common to overlapping regions. Ideally, we want to extend the output of each layer to produce a map of output class predictions, with one spatial location for each window of input. In the interest of time, we did not merge our two approaches into one predictor. We would perform the following: 1) Assign the set of classes (2 in our case) 2) Assign set of bounding boxes predicted by the regressor network for the spine class across all spacial locations 3) Merge the bounding boxes 4) Compute the match score using the sum of the distances between centers of the two bounding boxes and the intersection of the boxes. This box merge would compute the average of the bounding boxes coordinates. 5) The final prediction results from taking the merged bounding boxes with maximum class scores. This result is calculated by cumulatively adding the detection class outputs associated with the input windows that corresponds to the specific bounding box prediction.

.

References

[1] W. Bai, Z. Zhou, L. Ji, J. Cheng, and S. T. Wong. Automatic dendritic spine analysis in two-photon laser scanning microscopy images. *Journal of the International Society for Advancement of Cytometry*, 2007. 2

[2] C. Blumer, C. Vivien, C. Genoud, A. Perez-Alvarez, J. S. Wiegert, T. Vetter, and T. G. Oertner. Automated analysis of spine dynamics on live ca1 pyramidal cells. *Medical Image Analysis*, 19(1):87–97, 2015. 1

[3] D. Ciresan, A. Giusti, and L. Gambardella. Deep neural networks segment neuronal membranes in electron microscopy images. 2013. 2

[4] M. Djurisic. Manual annotations of dendritic spines, 2016. 1, 5

[5] M. Djurisic, G. S. Vidal, M. Mann, A. Aharon, T. Kim, A. F. Santos, Y. Zuo, M. Hbener, and C. J. Shatz. Pirb regulates a structural substrate for cortical plasticitys. In *Proceedings of the National Academy of Sciences of the United States of America: vol. 110 no. 51*, page 2077120776. 1

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(SS971504):119–139, 1996. 2

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 3

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[9] M. Knobloch and I. M. Mansuy. Dendritic spine loss and synaptic alterations in alzheimers disease. *Molecular Neurobiology*, 37(1):73–82, 2008. 1

[10] F.-F. Li, A. Karpathy, and J. Johnson. Lecture 8: Spatial localization and detection, 2016. 3, 8

[11] F.-F. Li, A. Karpathy, and J. Johnson. Linear classification: Support vector machine, softmax, 2016. 3

[12] K. Murphy and A. Torralba. Object detection and localization using local and global features. 2015. 2

[13] P. Penzes, M. E. Cahill, K. A. Jones, J.-E. VanLeeuwen, and K. M. Woolfrey. Dendritic spine pathology in neuropsychiatric disorders. *Nature Neuroscience*, 14(3):285–293, 2011. 1

[14] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 1, 2, 3

[15] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1, 3

[16] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 8

[17] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. 3, 5

[18] S. Zickler and M. Veloso. Detection and localization of multiple objects. 2015. 3