**Automated Cell counting using Convoluted Neural Networks**

Mateusz Grobelny, Lanxin Shen, Soumya Negi

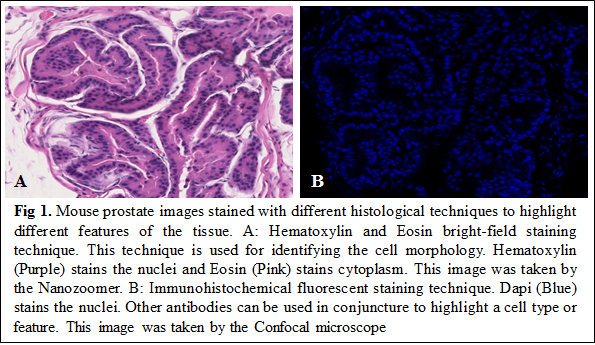
**Abstract:**

This paper describes an automated method of counting cells in microscopy images. Here we use convoluted neural networks

**Introduction:**

Advances in microscopy in the recent years has enabled scientists to understand cellular structure and organization with resolutions reaching up to 600X magnifications. Traditional microscopes have been replaced with automated ones that can generate more than a hundred images a day, like the Nanozoomer instrument by Hamatsu Photonics. Due to large image datasets, there is a need for automated tools that can analyze these images and reduce time spent in image analysis.

Cell counting is necessary in biomedical research to look for differences in the diseased state compared to the wild-type. For example, in cancer there is proliferation of cells due to loss in apoptosis which leads to an increase in cell density and if not treated in time a tumor formation which can spread to other tissues. Cancer is generally detected due to the presence of different biomarkers but these can only be seen once the cancer has matured and is expressing the proteins. For earlier detection, most diagnosticians resort to looking at differences in cell morphology or densities. An automated cell counting algorithm will simplify this process and provide a more unbiased approach to detection of diseased states. These methodologies will reduce time and cost, minimize error, and improve reproducibility of the data.



Several methods have already been developed to perform cell counting. ImageJ has a macro that can be used to count cells in cell culture using the hemocytometer [Grishagin 2015]. This method requires each image to be manually processed to a grayscale using a certain threshold thus taking away most of the automation. In addition to this limitation, tissue histology images taken after different staining processes (like Hematoxylin and Eosin) are more complex and cannot be easily converted to grayscale.

Identifying and annotating cells using algorithms can be challenging due to the diverse variety of cell types and imaging techniques available. Fig 1 illustrates the same tissue region of the mouse prostate stained with different techniques and also imaged with different microscopes. In this paper we use a machine-learning based feed-forward neural network [Hijazi et al., 2015] called Convoluted Neural Networks (CNNs). Convoluted neural networks are used in a variety of areas like image and pattern recognition, speech analysis and video recognition. A neural network is an interconnected system of artificially neurons that relay messages with each other. Each network is classified into different convolutional layers of neurons that respond to different combinations of inputs from the previous layer. A CNN takes an image as an input that has 3 dimensions: width, height, depth. A simple CNN is a sequence of layers and every layer transforms one volume of activation to another through a differentiable function. CNNs transform the original image layer by layer from original pixel to final class scores. Compared to other feed-forward networks, a CNN is easier to train due to fewer connections and parameters.

**The Architecture:**

Our approach draws on recent successes of deep neural networks for cell counting (Xie 2016) and transfer learning. We are using the convolutional neural networks for our image recognition architecture which including end to end training by convolutional layers, max pooling layers and upsampling layers. We aim to **1)** generate and train the primary convolution layers on patch images of cells or cell features which then **2)** would be append to fully convolutional layers to generate the classification of which pixels are apart or not apart of a cell by training on full size cell population images then checking against a matrix of cell location ground truth to adjust network weights and **3)** use the output density map, containing the probability that pixel is apart of a of cell or not find local maximas of high is-a-cell pixel probabilities and then regress those localized probabilities to get the cell counts output. Thus, the main goal for our cell counting algorithm neural net that would predict a density map for an input image with arbitrary size and RGB images and output a cell count. However, given the time limit and high learning curve of this field we were only able to make meaniful progress generate results for the patch detector CNN.

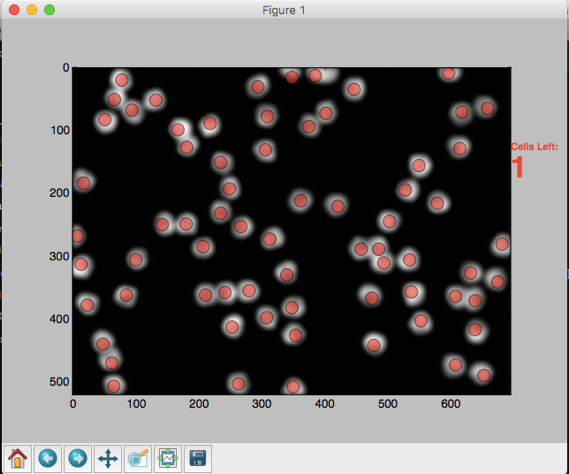
In the convolutional layer, there will be a kernel or filter that multiplies with the input matrix of the layer one by one in the same size and then go through all the matrix, and the value on centered pixel will change at the end. This filter is a small matrix with weights on each location, these weights will be tuned after training by the networks automatically.

Three convolutional layers is used at the beginning of the training architecture for cell object detection though patches by patches. For these layers, we are using the kernel of size 3x3 pixels. Each convolutional layer is followed by a max pooling layer to aggregate spatial information and generate the numbers for feature maps. Max pooling also results in results in a shrinkage of the feature maps for prevent overfitting. After that, a fully connected layer which also can be viewed works as convolution. One 3x3 kernel is used to cover entire input region and make a full convolutional networks for input and output classification mapping (Lempitsky 2010). After the max pooling, we will get a mapping with low sampling rate. Therefore, upsampling layer is performed in our network for end-to-end learning by backpropagation from the pixelwise loss. The upsampling layers using bilinear interpolation are used to reconstruct information for each pixels (Lempitsky 2010). It is a fast and effective way for learning dense prediction. For each upsampling layer, a 3x3 kernel is used as well, the weights can be learnt after training. The size of the output matrix is increased after each upsampling layer, such as from 25x25 to 50x50 and eventually 100x100, the same as the originally input image. The Rectified Linear Units (ReLUs) nonlinearity is applied to the output of every convolutional layers and upsampling layers. which found trains data faster than other methods like tanh units (Krizhevsky 2012).

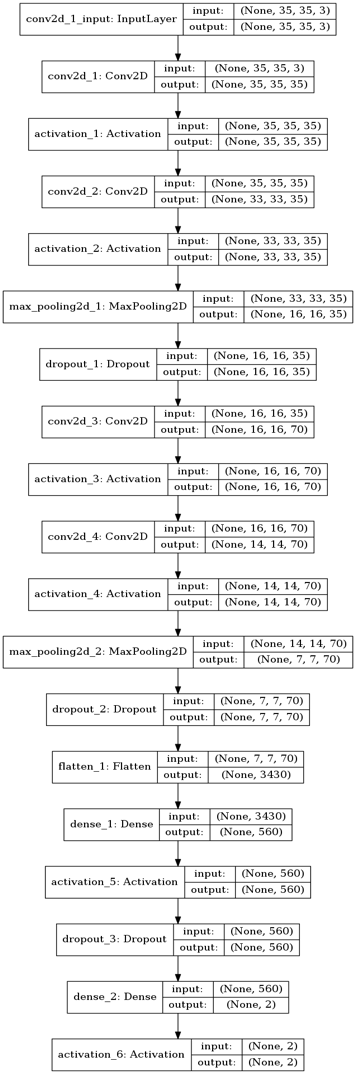
**The Dataset:**

**Fig. 2** – Distribution of cell counts of annotated fluorescent cell population.

The data we used are synthetic images that simulating for fluorescent cell population images by Broad Institute. Each image is 696 x 520 pixels in 8-bit TIF format with random number of cell between 1 and 100. The distribution of cell counts for our image data set is found on Fig. 2 with a 47.8 mean and standard deviation of 28.7. The size of the nuclei and cell areas were made matched to the average nuclei and cell areas. Some amount of focus blur also applied in these images. However, these images are not ready to use in our networks. The area of cell need to be labeled as a heatmap of possibility density distribution for each images. This step is done with python program which as its input takes in an image parses out the ground truth cells counts and presents graphical interface for manual mouse annotation of the locations location of each cell (Fig. 2). The pixel locations of each cell are save as a tuple of x and y with the origin being the top left corner of the image.



**Fig. 3** – Graphic displace for annotation of cell location, red circles indicate cells for which an location point has been registered. Script: click\_label.py

 Following the annotation preprocessing, the images of each nuclei were cropped into patches of size 35x35 pixels in preparation for nuclei detection training of the CNN. In addition, parts of fluorescent cell images which did not contain the nuclei, defined as all regions outside of the total crop areas which the nuclei patches were generated from, were used to randomly patches non-nuclei patches. From the 140 total annotated nuclei images we generated 9127 image patches of both nuclei and non-nuclei patches, which 73% were nuclei and 27% were non-nuclei patches (Fig. 4). To further increase the available data for training of the CNN images of were augmented in training in two ways; random color change in each color channel and the images were flipped vertically and/or horizontally.

**Fig.4** – Example of whole fluorescent cell population, patches of cell nuclei generated post annotation and augmented patches generated per batch during training.

|  |  |  |
| --- | --- | --- |
| **Whole Image** | **Patches** | **Patches Augmented** |
| **../SIMCEPImages_A09_C35_F1_s08_w1.jpg** | |  | | --- | | **../data/Patches_ALL/SIMCEPImages_B21_C87_F4_s22_w1_Patch_7_CX_1.jpg ../data/Patches_ALL/SIMCEPImages_B21_C87_F4_s22_w1_Patch_6_CX_0_.jpg**  **../data/Patches_ALL/SIMCEPImages_B21_C87_F4_s22_w1_Patch_6_CX_1.jpg ../data/Patches_ALL/SIMCEPImages_B21_C87_F4_s22_w1_Patch_78_CX_1.jpg** | | |  | | --- | | **../data/Working_Sets_Patches/Pred_augmented_images/aug__14_9979.jpeg ../data/Working_Sets_Patches/Pred_augmented_images/aug__15_9197.jpeg**  **../data/Working_Sets_Patches/Pred_augmented_images/aug__16_4679.jpeg ../data/Working_Sets_Patches/Pred_augmented_images/aug__8_7596.jpeg** | |
| 140 images | 9127 images | Many |

**Aim 1 – Patch Detection CNN Training:**

To train the detection of nuclei from cell population images we first trained our CNN on nuclei and non-nuclei patches. From the 9127 image patches, 60% of them are used for training and 20% were for validation, and 20% were used for evaluation and 10 images were used for prediction tests. In addition, we also tested our CNNs using mouse prostate images with immunohistochemical fluorescent stained the nuclei for both wild type and mutated type to confirm the CNN can also correctly classify the nuclei and non-nuclei from real world data. Images of mouse prostate images consisted of 512x 512 pixels in 8-bit TIF real images from which 5 patches of nuclei and non-nuclei patches were generated. Script used to generated images patches from cell locations csv was patch\_prep.py.

To generate our patch detector CNN we used the Keras library together with Tensorflow backend. The model patch detector was derived from the Keras Sequential Model example and has been adjusted to takes as input an image 35 x35 pixels with three color channels and a two class output (Fig. 5). The CNN uses categorical crossentropy as a loss function and stochastic gradient dissent as an optimizer. We recorded the accuracy and the loss as metrics to assess how the model is performing. Model was trained with image batch size = 30, and tested with training using 4 and 10 epochs. Validation of the model is preformed after each epoch to confirm the CNN is not over fitting. The final layer outputs a 1 by 2 matrix which is the classification probabilities for the patch a given patch either nuclei or non-nuclei (values 0 to 1). Script which prepared input data for CNN, sets up network architecture, collect results and generate figure is patch\_CNN.py.

**Fig. 5** - Patch detector architecture.

**Results -** Patch Detector Training, Validation and Evaluation

The patch detector was successfully trained and produced high accuracy classification. High classification accuracy did not take many epochs of training as each epoch had 5467 patch images. We tested trained the model for 4 and 10 epoch to insure the model had sufficient data to train on and we found that follow 1 epoch the model had very high accuracy >90% and after 4 epoch the model evaluated to 99% accuracy (Fig. 5 *Top Left*). Allowing the model to train for an additional 6 epochs did not appear to provide any benefits as the model started to exhibiting overfitting behaviors with 2% decreased in accuracy and 0.07 increase in the loss metric (Fig. 5 *Top Right & Bottom Left and Right*).

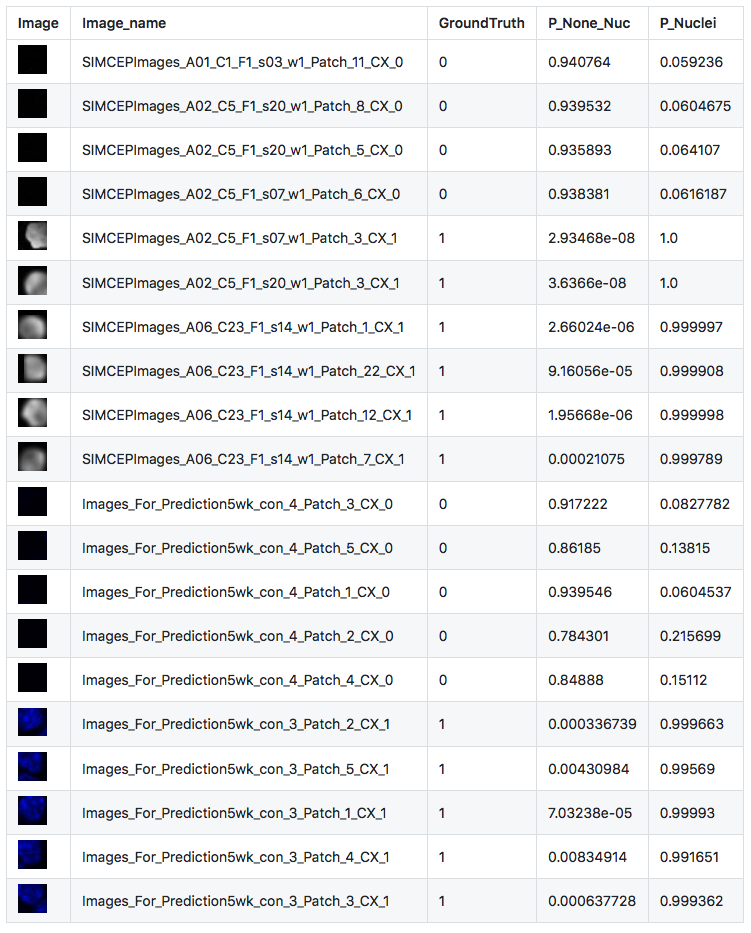
|  |  |  |
| --- | --- | --- |
| **Training** | ../cnn_models/patches_models_Full_Arc_test/patch_images_CNN_Metric.jpg | ../cnn_models/patches_models_Full_Arc_test/patchcnn_Full_arch_Metric_Patch_CNN.jpg |
| **Evaluation** | 1816 images belonging to 2 classes.  loss --- 0.0189  acc --- 0.9966 | 1816 images belonging to 2 classes.  loss --- 0.0816  acc --- 0.9766 |

**Fig. 5** – (Top Left) Training of patch detector with Accuracy and Loss on the left and right axis for 4 epoch (Top right) and 10 epochs (right). Sold blue and red colored line plots the Accuracy metric metrics for Training data and Validation data, respectively. While the thinner blue and red dashed line, plot the categorical crossentropy loss metric for the training and validation datasets, respectively. (Bottom Left & Right) Model evaluation metrics for a set of patch images not present in the training or validation stages. Following 10 training epochs the model exhibited some over fitting with 2% decreased accuracy and 0.07 increase in the loss metric.

**Fig 6.** – Example mouse prostate images stained for nuclei.

**Results -** Patch Detector Real Data Classfication

Following the successful training of the patch detector, we tested the patch classification capabilities on real world data namely, mouse prostate images. For prediction testing, 5 non-nuclei and 5 nuclei patches were prepared and passed into the model for classification. We found that out of the 20 randomly prepared patches each was classified correctly (Fig. 7). Furthermore, the probabilities of each correct class were typically >0.90 with the lowest classification probabilities found for the P\_None\_Nuc classification of a ground truth non-nuclei images from the mouse prostate images with immunohistochemical fluorescent stained the nuclei (Fig 7.). This however may be consistent with the bias of the complete patch image dataset as only 27% of the patch images are non-nuclei and no steps were taken to balance this bias during training.



**Fig. 7** – Patch detector CNN test using both synthetic (SIMCEP images) and real world data mouse prostate images with immunohistochemical fluorescent stained the nuclei (Images\_for\_Prediction\_5wk\*). Each patch was correctly classified as nuclei or non-nuclei, using the softmax output on the two classes which provides the probability for the image belonging to either of the two categories.

**Discussion:**

How accurate the network is.

Future directions: Our method identifies the cells and determines the number of cells, and density of cells in images. Our method in the current state is limited to cell identification and counting. This method can be adapted for a more comprehensive analysis of the images. The diversity in cell morphology requires a more intensive approach where the algorithm is trained to recognize the cell shape and size. For example in Benign prostatic hyperplasia the epithelial cells lose their   
Future directions to this methodology: Moving beyond just counting cells and looking for differences in cell structure, differences in cellular organization. Potential applications.

**Data Resources:**

Simulated cell images: <https://data.broadinstitute.org/bbbc/BBBC005/>

**Author Contributions:**

MG, LS and SN initiated and conceived the idea. MG, LS and SN helped with the training data. MG wrote the code, LS provided the architecture and SN provided the biological significance and real world data. MG, LS and SN wrote the paper. All authors reviewed and approved the final manuscript.

**References**

Xie, W., Noble, J. A., & Zisserman, A. (2016). Microscopy cell counting and detection with fully convolutional regression networks. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 1-10. doi:10.1080/21681163.2016.1149104

Krizhevsky A., I. Sutskever, and G. E. Hinton, ImageNet classi\_cation with deep convolutional neural networks, in NIPS, 2012, pp. 1097-1105.

Lempitsky V. and A. Zisserman, Learning to count objects in images, in NIPS,

2010, pp. 1324{1332.