**Technische Universität München**

**Fakultät für Informatik**

**IBBM Group**

# Cell Detection in Lens-free Microscopy Videos

Inter Disciplinary Project (IDP)

Sanjeev Kumar

Supervisor: Prof. Dr. Bjoern Menze

Advisor: Markus Rempfler

Submission Date: April 24, 2017

## Abstract

Deep learning techniques have been successful in analyzing cell growth of traditional microscopy images but has not been studied for Lens-Free Microscopy (LFM) images. In this report, we discuss in detail how deep convolutional neural networks are applied to tackle cell detection problem in LFM images. Further, we will present benchmarking results of the LFM datasets across different network architectures and training methods.

## Contents

## 1 Introduction

### 1.1 Problem Statement

Lens-free Microscopy (LFM) [7, 4] has offered promising alternative to traditional light microscopes in vitro experiments. Unlike traditional microscopy methods, LFM captures the digital hologram of the light wavefront originating from an object. LFM components are extremely cheap and LFM images can distinguish transparent objects like living cells accurately. Thus, LFM has a wide range of applications where a conventional microscope instruments would be too big or expensive, such as the continuous monitoring of growing cell cultures inside standard incubators.

To quantify the clinically relevant information on cell growth and migration from the large amount of images that are acquired in such continuous monitoring, reliable automatic image analysis methods are crucial. Accurate detection is the key ingredient to automatic analysis. It gives immediate access to cell count and density, and statistics on cell motility or growth can be quantified if the localizations are used in combination with an appropriate lineage tracing model.

Deep convolutional neural networks have been successfully applied in traditional microscopy imaging tasks such as [1, 13]. Although these methods cannot directly be used in LFM images as the cell structure is considerably different from traditional microscopy images. In this report, we present methods to solve cell detection problem for LFM image sequences using deep convolutional neural networks.

### 1.2 Contributions

Our main contributions in this work are following

- A fast and robust cell detector for LFM image sequences based on deep convolutional neural networks.
- Probability map and grid based training to adapt deep learning models from natural imaging applications to cell detection in LFM images.
- Comparison of different deep learning networks on LFM datasets for cell detection.

This work has been submitted in MICCAI 2017 [9].

## 2 Methodologies

In this section, we will discuss the different methodologies used for the cell detection in LFM image frames. First, we will describe few key concepts related to convolutional neural networks which influenced our network design choices. Then for each method, we will give an overview of the approach and discuss the corresponding convolutional neural network (CNN) architectures. Finally, we outline the post processing steps to convert network output into cell locations.

## 2.1 Terminology

**Fully Convolutional Networks.** A fully convolutional network (FCN) has all convolutional layers and output of the network preserves spatial relations of the input. Fully convolutional networks have become popular for semantic segmentation tasks [6].

FCNs are translation variant and hence are well suited for detection as well. In LFM images, the global context (background, the arrangement of cells) of an image frame does not impact cell's features, so plain FCNs should be sufficient for accurate detection of cells. Based on theses observations, all network architectures we present are based on fully convolutional features.

**Up-Convolution.** Up-Convolution is convolution applied on strided input image. It has the effect of up-sampling the input, the up-sampling factor can be controlled using stride length and filter size. This is very useful in reconstructing high-resolution output from downsampled feature maps created by max-pooling operation. A detailed explanation of the up-convolution can be found at [2]. It is also sometimes referred as deconvolution or transposed convolution.

**Residual Networks.** Residual networks [5] have outperformed traditional convolutional neural networks in object detection tasks at ImageNet challenge by large margin [11]. In addition to linearly stacked convolutional layers, ResNets have skip connections which can learn identity mappings. The shortcut connections reduce the number of network parameters significantly and network converges to better local minima.

## 2.2 Probability Map Based Detector

Probability map based detectors consist of a fully convolutional network which predicts a dense probability map for each input image. The output probability map has the same resolution as the network input and each value $P(i,j)$ in output response map denotes the probability of pixel $(i,j)$ being the cell center.

In the annotated dataset, we only have cell center annotations. So, for each training image $I_k$, we construct a corresponding cell probability map $P(c=1|I_k)$ by placing a Gaussian kernel $G_\sigma$ with $\sigma = 8$ (cell size $\approx$ 24px) at each annotated center. This implicitly represents the assumption that all cells have about the same extent, which is reasonable for our microscopy data.

The training of the fully convolutional network is done by randomly sampling patches from annotated image region. To make the network robust to brightness changes, the input image patches are locally normalized to zero mean and unit standard deviation. The network is trained using Adam Optimizer with binary cross entropy loss $L_c$ as given by

$$L_c = \sum_{i=1}^{N}(-plog(t) - (1-t)log(1-p)) \tag{1}$$

where $p$ is predicted class probability map and $t$ is ground truth probability map.

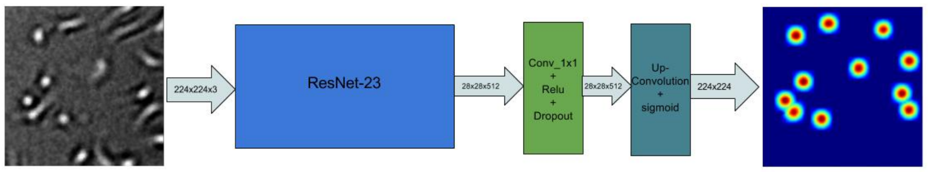


**Figure 1:** The network architecture of ResNet-23. The network uses first 23 layers of ResNet-50. In output response map, red pixels are cell centers with probability of 1.

The predicted probability maps are post processed to get cell centers. The post processing of probability maps is done in following steps.

1. First locations with local maxima are found by apply 1-D maximum filter of size 20 ($\approx$ cell size). This also includes points with constant probabilities in local maxima (i.e. background).
2. Background points are then removed from local maxima locations by discarding points where 1-D minimum filter of size 20 on probability map has zero values.
3. Finally the center of each contiguous local maxima region is marked as cell center.

The different network architecture based on this approach are discussed below.

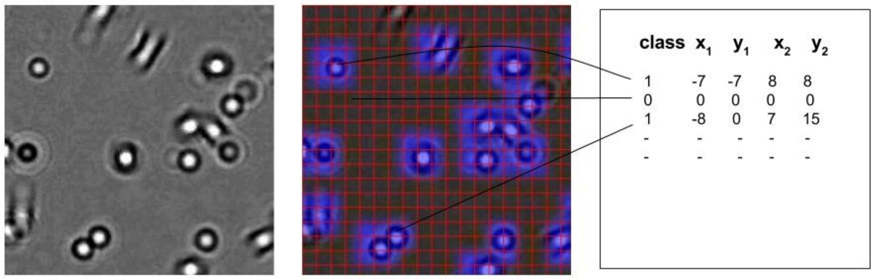
### 2.2.1 ResNet

In this network configuration, we use ResNet-50 [5] as base network architecture. The network is pre-trained for classification task on ImageNet dataset [11]. To predict dense probability maps, we modify the base network by taking output response maps from $k$th layer and add one convolutional layer with filter size $1 \times 1$ followed by a up-convolution layer which makes output response map size equal to the input image. Sigmoid activation is then applied after up-convolution to predict per pixel class probabilities. A drop out layer [12] with probability $p = 0.5$ is also added after convolutional layer to prevent over-fitting. We pick layer k somewhere in the middle of Res-50 such that response maps at $k$th layer have high enough resolution for fine scale reconstruction of output response maps. The network architecture for $k = 23$ is depicted in Figure 1.

During training, the weights of initial $k$ layers are remained fixed and only newly added layers are fine-tuned using Adam optimizer. This network architecture with $k = 23$ achieves best results for the cell detection task.

### 2.2.2 UNet

UNet [10] extends the idea of Fully Convolutional Networks for Semantic Segmentation [6] by splitting the network into contracting and expanding paths. In the contracting path, a series of convolutional and max-pooling layers are applied. In



| class | $x_1$ | $y_1$ | $x_2$ | $y_2$ |
|---|---|---|---|---|
| 1 | -7 | -7 | 8 | 8 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | -8 | 0 | 7 | 15 |
| - | - | - | - | - |
| - | - | - | - | - |

**Figure 2:** DetectNet training data representation. **Left**: Input training image. **Middle**: Ground truth upsampled grid overlayed over input image. **Right**: Ground truth bounding box and class probability values for grid.

the expansion path, the upsampling is done at each step followed by convolutional layers and merging with response maps from contracting path.

We modify this network by replacing upsampling and convolutional layers in expansion path with up-convolutional layers. This is done to reduce the number of network parameters introduced by convolutional layers in expansion path as the network is trained from scratch. Drop layers with probability $p = 0.5$ are also added after each convolution layer in the contracting path to avoid overfitting. The modified network produces output class probability maps of the same size as input.

All weights in the network are initialized using Glorot initialization and the network is trained end to end using Adam optimizer. This network achieves accuracy close to ResNet.

### 2.2.3 Simple CNN

Simple CNN is a shallow network with four convolutional layers. The first three convolutional layers have a filter size of $5 \times 5$ and the last layer has a filter size of $1 \times 1$. Dropout, batch normalization, and max-pooling layers are also added in between convolutional layers. An up-convolution with sigmoid is added in the end to reconstruct output probability map of the same size as input. This network is used as a baseline for comparing performance among different architectures.

## 2.3 DetectNet

This network architecture is inspired from object detector proposed by Redmon et al. [8]. Unlike [8], this network only has fully convolutional layers. For input image of size $h \times w$, the network outputs two grids of size $k \times h/s \times w/s$ for class probabilities, and $4 \times h/s \times w/s$ for bounding box locations. Here, $k$ is the number of output classes and $s$ is the hyperparameter which is usually smaller than the size of the smallest object to detect. Each location $(i, j)$ in output grid corresponds to $s \times s$ region in the input image and represents the class probability and bounding box location of the object in $s \times s$ image region. For cell detection task, we use $k = 1$ since the problem is binary classification and $s = 15$ as the cell size is close to 15 pixels in the image.

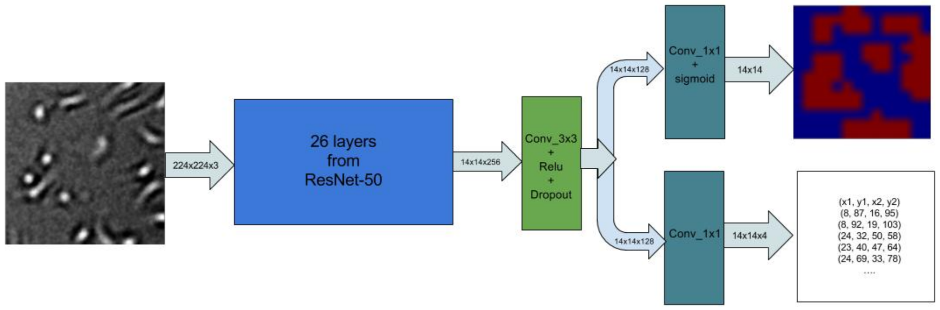


**Figure 3:** The network architecture of DetectNet. The network uses first 26 layers from ResNet-50. The output probability map is upsampled to input image size for better visualization.

The base network for DetectNet is based on first 26 layers of ResNet-50 which is pre-trained on ImageNet classification task. Then, we add one convolutional layer of filter size $3 \times 3$ followed by two parallel convolutional layers, one for bounding box prediction and other for class probabilities. The network architecture is depicted in Figure 3.

The ground truth class score map for a *grid cell* (a point in output response map corresponding to $s \times s$ region of the input image) has value 1 if at least one cell has more than 50 percent overlapping area with corresponding input image region. The bounding box score is relative distance from *grid cell's* center to cell bounding box location. If there are multiple cells corresponding to one *grid cell*, the cell having highest overlapping area is selected. An example ground truth representation is illustrated in Figure 2.

The network uses combined loss function $L = L_b + L_c$, where $L_b$ is $L1$ loss on bounding box predictions and $L_c$ is binary cross entropy loss on class probabilities.

$$L_c = \sum_{i=1}^{N} (-p_i log(t_i) - (1 - t_i) log(1 - p_i)) \tag{2}$$

$$L_b = \sum_{i=1}^{N} (|x_1^t - x_1^p| + |y_1^t - y_1^p| + |x_2^t - x_2^p| + |y_2^t - y_2^p|) \tag{3}$$

$N$ is total number of *grid cells* in output grid, $p_i$ and $t_i$ are the predicted and ground truth probabilities of $i$th *grid cell* containing a cell, $(x_1, y_1)$ and $(x_2, y_2)$ are top-left and bottom-right co-ordinates of cell bounding box relative to *grid cell's* center. The network is trained on combined loss $L_c + L_b$.

## 3 Experiments and Results

### 3.1 Datasets

The annotated dataset consists of one sequence of cell type A549 and two sequences of cell type 3T3. The sequence A549 has 250 images, with each image having a region of interest $1295 \times 972$px. The two 3T3 sequences have 350 and 300 images respectively, with each image having a region of interest $639 \times 511$px. Images were

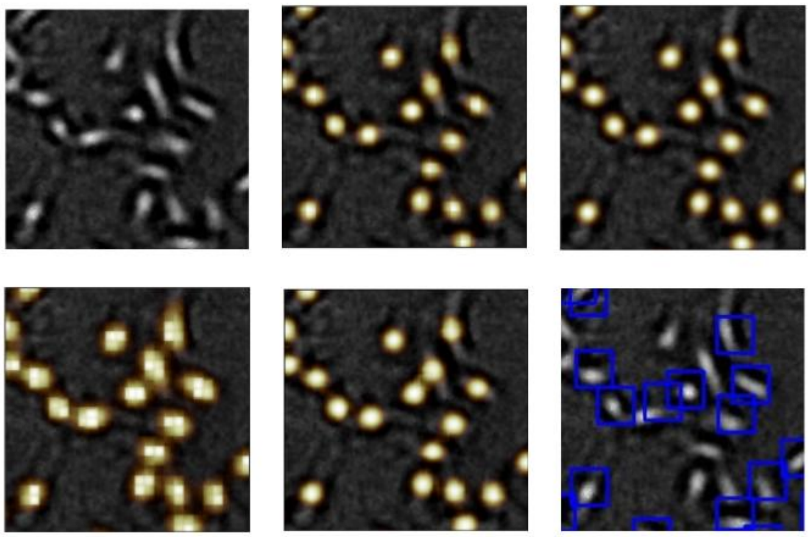


**Figure 4:** Result of applying different detectors on an image patch from 3T3 cell sequence. From top left to right: 1) Raw LFM image, 2) image overlayed with cell probability map generated using ResNet-11, 3) ResNet-23, 4) Simple-CNN, 5) UNet, 6) cell bounding box predictions using DetectNet.

acquired at an interval of 3 min with $1.4\,\mu\text{m} \times 1.4\,\mu\text{m}$ per pixel. For each image frame, the cell center coordinates in the region of interest are available as ground truth.

## 3.2 Training

For all network architectures, we used dataset A549 for training and 3T3 datasets for testing. We trained five different configurations of network architectures presented in last section, namely ResNet-11 (11 layer FCN ResNet), ResNet-23 (23 layer FCN ResNet), UNet, Simple-CNN and DetectNet. All networks converged (i.e. validation loss stopped improving) in less than 60 epoch(s) (1 epoch = 4000 image samples). For training, we divided the dataset (A549) in training (180), validation (20) and testing (50). We used a batch size of 8 with Adam optimizer for training the networks. In each batch, the image patches of network input size were randomly sampled from the training set. UNet and SimpleCNN were trained end to end, only newly added layers were fine-tuned for ResNet and DetectNet.

## 3.3 Results

The results presented here are evaluated on two 3T3 datasets. We are not reporting results of test subset from sequence A549 as the images within one sequence are very similar and such evaluation will not reflect the true performance of the detectors. We match annotated cells to detections within each frame with the Hungarian algorithm and consider only matches closer than 20 px ($\approx$ a cell center region) as a true positive (TP). Unmatched annotations are counted as false negative (FN),

**Table 1:** Evaluation of different architectures. Precision, recall and F1 scores are averaged over two 3T3 datasets.

| Method | Precision | Recall | F1 |
|---|---:|---:|---:|
| ResNet-11 | 89.30 | 97.59 | 93.26 |
| ResNet-23 | **97.01** | 95.61 | **96.30** |
| UNet | 93.29 | **97.14** | 95.17 |
| Simple-CNN | 73.16 | 93.68 | 82.16 |
| DetectNet | 86.89 | 70.74 | 77.99 |

unmatched detections as false positive (FP). The quantitative results of different networks are presented in Table 1. Out of all network architectures, ResNet-23 performs best with F1 score of 96%.

DetectNet performs worse than probability map based detectors because it is not possible to distinguish between false detections and overlapping cells from bounding box locations. Among probability based detectors, ResNet-23 and UNet have comparable performances. This is because these networks are able to produce fine scale output response maps which lead to better separation of overlapping cells. The result of applying different detectors on an image patch is shown in Figure 4.

## 4 Conclusion

We presented different techniques to apply deep convolutional neural networks for cell detection in LFM image sequences. The detector ResNet-23 presented above achieved $\approx$ 96% F1 score for 3T3 image sequences and is suitable for practical purposes. Although, the detector sometimes loses track of detected cells between consecutive frames. This can be improved by taking into account the temporal nature of image sequences.

## References

[1] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *NIPS*, pages 2843–2851, 2012.

[2] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. 2016.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.

[4] A. Greenbaum, W. Luo, T.-W. Su, Z. Göröcs, L. Xue, S. O. Isikman, A. F. Coskun, O. Mudanyali, and A. Ozcan. Imaging without lenses: achievements and remaining challenges of wide-field on-chip microscopy. *Nature methods*, 9(9):889–895, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, Nov. 2015.

[7] O. Mudanyali, D. Tseng, C. Oh, S. O. Isikman, I. Sencan, W. Bishara, C. Oztoprak, S. Seo, B. Khademhosseini, and A. Ozcan. Compact, lightweight and cost-effective microscope based on lensless incoherent holography for telemedicine applications. *Lab on a Chip*, 10(11):1417–1428, 2010.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[9] M. Rempfler, S. Kumar, V. Stierle, P. Paulitschke, B. Andres, and B. H. Menze. Cell lineage tracing in lens-free microscopy videos. In *MICCAI (in review)*, 2017.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Pattern Recognition*, 15:1929–1958, 2014.

[13] D. A. V. Valen, T. Kudo, K. M. Lane, D. N. Macklin, N. T. Quach, M. M. DeFelice, I. Maayan, Y. Tanouchi, E. A. Ashley, and M. W. Covert. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. In *PLoS Comput Biol*, 2016.