Rapidly Adaptive Cell Detection using Transfer Learning with a Global Parameter

Nhat H. Nguyen[1], Eric Norris[2], Mark G. Clemens[2], and Min C. Shin[1]

[1]Department of Computer Science, University of North Carolina, Charlotte
[2]Department of Biology, University of North Carolina, Charlotte
{nhnguye1,enorris9,mgclemen,mcshin}@uncc.edu

Abstract. Recent advances in biomedical imaging have enabled the analysis of many different cell types. Learning-based cell detectors tend to be specific to a particular imaging protocol and cell type. For a new dataset, a tedious re-training process is required. In this paper, we present a novel method of training a cell detector on new datasets with minimal effort. First, we combine the classification rules extracted from existing data with the training samples of new data using transfer learning. Second, a global parameter is incorporated to refine the ranking of the classification rules. We demonstrate that our method achieves the same performance as previous approaches with only 10% of the training effort.

1 Introduction

Dramatic advances in biological imaging over the past decade, including both the development of more powerful imaging hardware and novel fluorescent probes, have revolutionized many areas of biological research [9]. The manual approach of cell analysis is too tedious and error-prone; thus it is not feasible for handling large datasets. Automated cell detection is a growing field of interest with a wide range of applications which permits statistical analysis of various cell parameters such as apoptosis, adherence, morphology and motility [1]. Thus, it has the potential to identify even subtle effects of many physiological stimuli on many cell types [4].

Recently, a number of automated cell detection methods, which are based on machine learning algorithms, have been proposed [2, 4, 8]. To our knowledge, these cell detection methods are not effective on different cell types since they required a large number of training samples from each cell type. For each new cell dataset with different appearance (e.g., size, shape, color), the users often need to re-train the algorithm by collecting training samples. This is a tedious and time-consuming process. Thus, there is a great need for a method that can be rapidly trained on new datasets with minimal training effort.

In this paper, we propose a novel cell detection method that can be rapidly trained to new datasets. The goal is to minimize the number of samples required to train the detection method which should translate to the reduction of human effort. We use a transfer learning algorithm [10] to leverage the classification

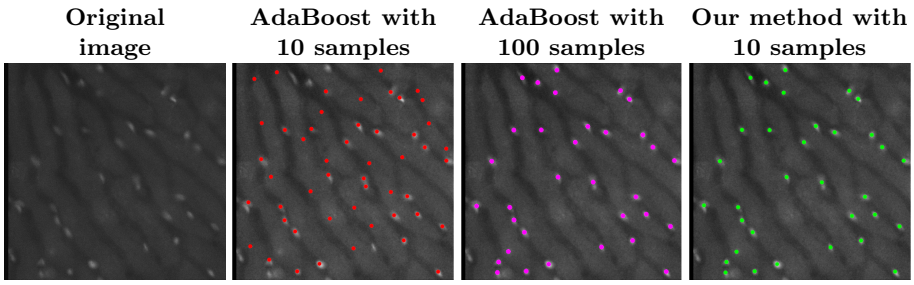

Fig. 1. Sample detection results from AdaBoost and our method. Note the poor performance of AdaBoost with a small training effort (10 samples). The proposed method with only 10 training samples is able to achieve equal performance of AdaBoost with 100 training samples.

rules gathered from existing data (source classes) to improve detection on the new data (target class). By incorporating the cell size distribution as a global parameter on new dataset, we further increase the accuracy of the detection algorithm using only a minimal number of training samples. The cell size distribution is determined during the training step and does not need to be re-trained for each individual image. We refer to our method as the GlobalTrAdaBoost, a boosting-based method that integrates a global parameter into the transfer learning framework. The evaluation on five cell types with 50 real images (2660 cells) demonstrates that the GlobalTrAdaBoost is able to achieve equal performance of a typical pattern recognition algorithm with only 10% of the training effort required. Sample detection results are shown in Figure 1.

2 Method

We first describe a baseline cell detection using Adaptive Boosting. Then, we explain the proposed method of using transfer learning, followed by the inclusion of the global parameter. The overview of the method is shown in Figure 2.

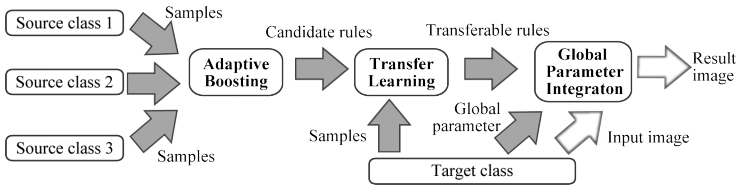

Fig. 2. The overview of the proposed method. The gray arrow indicates the training step while the white arrow indicates the testing step.

2.1 Adaptive Boosting (AdaBoost)

Boosting is an iterative method of constructing an accurate classifier by combining many weak classifiers, each of which only needs to be reasonably accurate [3]. One of the most popular boosting methods, Adaptive Boosting (also known as AdaBoost), can be used to train the cell detector.

The AdaBoost algorithm weights each weak classifier based on its prediction accuracy. In the feature space $\mathcal{X}$, and the label space $\mathcal{Y} = \{-1, +1\}$ denoting a pixel as background or cell sample, the detection task is to estimate a classifier function $f : \mathcal{X} \rightarrow \mathcal{Y}$ given training data $D = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, 1 \leq i \leq n\}$ where $\mathbf{x}_i$ and y_i are respectively the feature vector and the label of training sample i, and n is the number of training samples. Initially, AdaBoost constructs a distribution of weights $\mathbf{w} = \{w_i | w_i = \frac{1}{n}, 1 \leq i \leq n\}$ over the training data. For each iteration $t = 1$ to T, AdaBoost selects a weak classifier that gives the least classification error as $h_t(\mathbf{x}_i) = \arg\min(\epsilon_t)$ where $\epsilon_t = \sum_i w_i[y_i \neq h_t(\mathbf{x}_i)]$. In the next iteration, the weights associated with the samples misclassified by the selected weak classifier are increased as

$$w_i \leftarrow w_i e^{-\alpha_t y_i h_t(\mathbf{x}_i)} \tag{1}$$

where $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$. Finally, the strong classifier $\hat{f}$ is computed as the signum function of the weighted linear combination of T weak classifiers

$$\hat{f} = \text{sign}(\sum_t \alpha_t h_t(\mathbf{x}_i)). \tag{2}$$

2.2 Transfer Learning with AdaBoost (TaskTrAdaBoost)

Since AdaBoost requires a large number of training samples to be effective, training with only few samples often leads to poor performance. Transfer learning re-uses the classification rules from source classes with the target class to minimize the required amount of target training samples. We use a recent transfer learning algorithm, TaskTrAdaBoost, to conduct the training process [10].

First, we would like to gather the classification rules (weak classifiers) from the source classes. Let source class S_k ($1 \leq k \leq K$ where K is the number of source classes) contain the source training data $D^{S_k} = \{(\mathbf{x}_j, y_j) | 1 \leq j \leq n^{S_k}\}$ where n^{S_k} is the number of source training samples. We use AdaBoost from Section 2.1 to train the classifier function $\hat{f}^{S_k}$ based on D^{S_k}. Then, we collect all candidate weak classifier h_c from $\hat{f}^{S_k}$ of K source classes into set $\mathcal{H}_c = \{h_c(\mathbf{x}_j) | h_c(\mathbf{x}_j) \in \hat{f}^{S_k}, 1 \leq c \leq C\}$ where $C = T \times K$.

Second, we use set $\mathcal{H}_c$ to build the strong classifier $\hat{f}^{\tau}$ for the target class $\mathcal{T}$ given training data $D^{\tau} = \{(\mathbf{x}_l, y_l) | 1 \leq l \leq n^{\tau}\}$ where n^{τ} is the number of target training samples (note that $n^{\tau} \ll n^{S_k}$). The target weight distribution is initialized as $\mathbf{w}^{\tau} = \{w_l | w_l = \frac{1}{n^{\tau}}, 1 \leq l \leq n^{\tau}\}$. For each iteration, we find a transferable weak classifier h_β that minimizes the error over target data D^{τ} as

$$h_\beta(\mathbf{x}_l) = \arg \min_{h_\beta \in \mathcal{H}_c} (\epsilon_l) \tag{3}$$

where the classification error is computed as: $\epsilon_l = \sum_l w_l[y_l \neq h_\beta(\mathbf{x}_l)]$.

With few training samples n^τ, it is possible that more than one h_β yield the same minimal error ϵ_l over D^τ. Since the TaskTrAdaBoost uses only one h_β, it could select the h_β which over-fits the training samples and thus reduces performance on the target class. In the next section, we propose to use a global parameter, distribution of cell sizes, to refine the ranking of weak classifiers to regularize the problem and alleviate the risk of over-fitting.

2.3 Global Parameter Integration (GlobalTrAdaBoost)

We select the transferable weak classifier h_β to conform with a global parameter, the cell size distribution, in addition to minimize the target training error. We measure the conformity of h_β to the cell size distribution using the Kullback-Leibler divergence, a non-symmetric measure of the difference between a true distribution and an approximated model [5]. Under *in vivo* microscopy, the florescent intensity within the cell population differs depending on the location of a cell with respect to the microscope focal lens resulting in different cell sizes [6]. Thus, we model the cell sizes using a Gaussian distribution.

Let us define the cell size distribution $\mathcal{P} \sim \mathcal{N}(\mu_m, \sigma_m^2)$ with $\mu_m = \frac{1}{M}\sum_m(r_m)$, $\sigma_m = \sqrt{\frac{1}{M}\sum_m(r_m - \mu_m)^2}$ where r_m is the cell size and $1 \leq m \leq M$ is the number of size samples. In other words, the cell size distribution $\mathcal{P}$ can be estimated using set $\mathcal{R} = \{r_m | 1 \leq m \leq M\}$. In the target training image $\mathcal{I}_\tau$, a user can measure r_m by one additional mouse click on the cell boundary after getting the cell location. Thus, acquiring M samples of cell sizes requires only additional M mouse clicks. In Section 3.3, we show that the estimation of $\mathcal{P}$ is robust enough to maintain stable performance with $M = 6$.

After acquiring $\mathcal{P}$, we collect all h_β into set $\mathcal{H}_\beta = \{h_\beta(\mathbf{x}_l) | 1 \leq \beta \leq B\}$ where B is the number of weak classifiers that satisfy (3). Then, we employ each $h_\beta \in \mathcal{H}_\beta$ to classify the target training image $\mathcal{I}_\tau$ to obtain a binary classification image containing cell and background pixels. Using the connected component labeling procedure, we group cell pixels into cell regions and construct set $\mathcal{R}_\beta = \{r_u | 1 \leq u \leq U\}$ where r_u is the radius of a cell region and U is the number of detected regions. From set $\mathcal{R}_\beta$, we can compute the detected cell size distribution $\mathcal{Q}_\beta \sim \mathcal{N}(\mu_u, \sigma_u^2)$. We select the transferable weak classifier h'_β which minimizes the Kullback-Leibler divergence between $\mathcal{P}$ and $\mathcal{Q}_\beta$:

$$h'_\beta(\mathbf{x}_l) = \arg\min_{h_\beta \in \mathcal{H}_\beta} (\mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}_\beta)) \tag{4}$$

where $\mathcal{D}_{KL}(\mathcal{P}||\mathcal{Q}_b) = \sum_p \mathcal{P} \log \frac{\mathcal{P}(p)}{\mathcal{Q}_\beta(p)}$ and p is the bin containing the range of cell size values [5]. As the result, the weak classifier h'_β conforms with the global parameter $\mathcal{P}$ besides minimizing the target training error. The procedures to update $\mathbf{w}^\tau$ for each iteration and to construct the strong classifier $\hat{f}^\tau$ are similar to (1) and (2), respectively.

3 Experiments

3.1 Evaluation Procedure

Data Description Five different cell types (white blood cells, natural killer T-cells, HT29 colon cancer, red blood cells, and drosophila) are acquired using 2 imaging protocols (*in vivo* epi-fluorescence and isolated fluorescently labeled) and 3 magnification levels (10X, 20X, and 40X). Sample images of each cell types are shown in Figure 3. We evaluate the performance of the detection algorithms on a total of 50 real images (10 images from each cell type). Each image contains from 25 to 100 cells (total of 2660 cells). We divide the images from each cell type into two halves for training and evaluating. A biology technician has manually determined the center and the radius of the cells in 50 images.

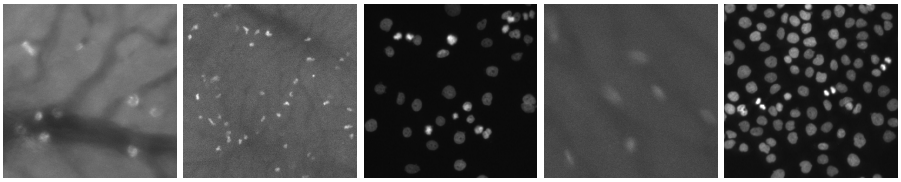

Fig. 3. Representative images of five different cell types (from left to right): white blood cells, natural killer T-cells, HT29 colon cancer, red blood cells, and drosophila.

Local Features The feature space $\mathcal{X}$ is defined by a 10-D local feature vector $\mathbf{x}_i$ from the training sample i. The description of each feature is explained as follow. First, we compute the normalized radial mean response of sample i as the ratio between the mean intensity of the inner circle to the outer circle surrounding the pixel location of i [7]. We apply six different scales for the inner and outer circle diameters to accommodate a variety of cell sizes. Second, we calculate the mean of gradient magnitude within a square region around sample i. The edge length of the region is same as the largest circle diameter from above. Third, we collect the filtering responses at sample i from circular averaging, low-pass Gaussian and isotropic Laplacian of Gaussian kernels. The response values are normalized by dividing with the maximum within the entire image. Note that additional features can be added for potential improvement.

Performance Metric We measure the performance according to the manually marked ground truth data. Each cell detection by an algorithm is determined as true positive if there is a corresponding ground truth cell within the mean value of cell radii r_m (estimated by the user, as discussed in Section 2.3). We compute the number of true positives (TP), false positives (FP) and false negatives

(FN). Recall and precision are computed as $\frac{TP}{TP+FN}$ and $\frac{TP}{TP+FP}$, respectively. For measuring the overall performance, we use F-measure, a harmonic mean of precision and recall, as $F = 2 \times \frac{Precision \times Recall}{Precision + Recall}$.

3.2 Detection Accuracy

To evaluate the detection accuracy, each cell type is chosen as a target class, and the remaining cell types are used as source classes. We measure the training effort E_τ as the number of target training samples and the number of size samples required to obtain the global parameter ($E_\tau = n_\tau + M$). Note that $M = 6$ for GlobalTrAdaBoost and $M = 0$ for AdaBoost and TaskTrAdaBoost. Training is conducted with E_τ varying from 10 to 100. For each value of E_τ, we conduct 30 executions of training and testing on each of 5 target classes. In each execution, the target training samples D_τ are randomly selected. The performances (in terms of F-measures) are shown in Table 1.

Table 1. F-Measures (Mean ± Standard Error of the Mean) of boosting-based cell detection methods for different training efforts. A performance number is highlighted in bold if it is significantly better than other methods based on a paired t-test at $p = 0.05$. $Global_{under}$ and $Global_{over}$ (as discussed in Section 3.3) are two versions of GlobalTrAdaboost with fluctuated values of the cell size distribution.

E_τ	AdaBoost	TaskTrAdaBoost	GlobalTrAdaBoost	$Global_{under}$	$Global_{over}$
10	0.57 ± 0.024	0.69 ± 0.018	**0.81 ± 0.014**	0.78 ± 0.032	0.79 ± 0.038
20	0.68 ± 0.019	0.72 ± 0.018	**0.82 ± 0.011**	0.79 ± 0.039	0.80 ± 0.042
30	0.69 ± 0.018	0.75 ± 0.015	**0.81 ± 0.011**	0.79 ± 0.031	0.80 ± 0.012
40	0.76 ± 0.013	0.78 ± 0.012	**0.81 ± 0.008**	0.81 ± 0.034	0.81 ± 0.008
50	0.76 ± 0.012	0.77 ± 0.012	**0.82 ± 0.006**	0.82 ± 0.023	0.82 ± 0.023
60	0.79 ± 0.010	0.80 ± 0.010	0.83 ± 0.008	0.82 ± 0.008	0.81 ± 0.043
70	0.80 ± 0.009	0.82 ± 0.007	0.83 ± 0.004	0.82 ± 0.007	0.82 ± 0.014
80	0.80 ± 0.012	0.81 ± 0.010	0.83 ± 0.006	0.83 ± 0.008	0.82 ± 0.011
90	0.80 ± 0.012	0.81 ± 0.009	**0.83 ± 0.004**	0.83 ± 0.009	0.83 ± 0.010
100	0.82 ± 0.009	0.83 ± 0.007	0.84 ± 0.004	0.83 ± 0.008	0.83 ± 0.010

When trained with a large number of samples ($E_\tau = 100$), AdaBoost, TaskTrAdaBoost, and GlobalTrAdaBoost reach similar maximum performance (F-measures equal to 0.82, 0.83, and 0.84, respectively). However, GlobalTrAdaBoost shows significant improvement to other methods with small training efforts. First, at training effort $E_\tau = 10, 20$, and 30, the average improvement of the proposed method over TaskTrAdaBoost is 17%, 14%, and 8% with standard error reduced by 22%, 39%, and 27%, respectively (see Table 1). Second, GlobalTrAdaBoost only needs 10% of the training effort ($E_\tau = 10$ versus 100) to achieve the same performance ($p = 0.87$) as AdaBoost. Third, with just 10 training samples, GlobalTrAdaBoost's F-measure is already 0.81, which is only 4% lower than the maximum performance. The sample results comparing AdaBoost and the proposed method are shown in Figure 4.

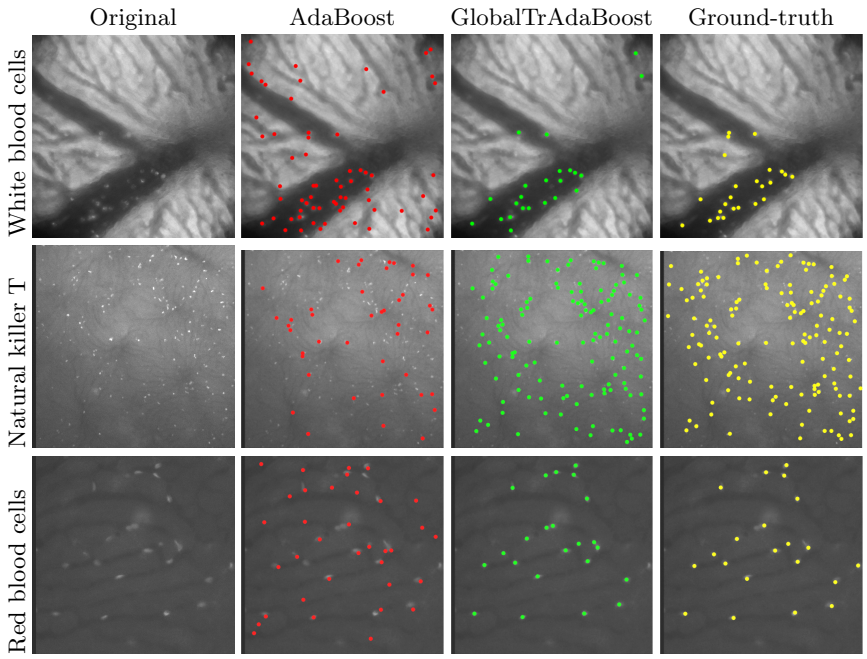

Fig. 4. Sample results comparing AdaBoost and the proposed method (GlobalTrAdaBoost) using 10 training samples.

3.3 Sensitivity of Global Parameter

In this section, we investigate the sensitivity of the algorithm's performance with respect to the accuracy of the global parameter estimation. To fully estimate the range of values that a global parameter can take for a dataset, we randomly select sets of M size samples. For each set of M samples, we compute the mean and standard deviation of cell sizes. We repeat this randomly sampling process 30 times in each dataset. We observe that the mean of the standard deviations of multiple sets start converging when $M \geq 6$. For each cell type, we examine multiple sets of 6 size samples and compute the mean μ_Δ and standard deviation σ_Δ. If the GlobalTrAdaBoost performance is still higher than other methods when the global parameter estimation varies by the value of σ_Δ, then 6 cell size samples ($M = 6$) are sufficient to estimate the cell size distribution.

Consequently, we integrate the GlobalTrAdaBoost with two fluctuated values of the global parameter $\mathcal{P}_{under} \sim \mathcal{N}(\mu_\Delta - \sigma_\Delta, \sigma_\Delta^2)$, and $\mathcal{P}_{over} \sim \mathcal{N}(\mu_\Delta + \sigma_\Delta, \sigma_\Delta^2)$. The corresponding methods $Global_{under}$ and $Global_{over}$ are executed 30 times in the same procedure described in Section 3.2. We show the F-measures in conjunction with other detection methods in Table 1. The performance of both $Global_{under}$ and $Global_{over}$ at each E_τ are significantly better ($p < 0.05$) than both AdaBoost and TaskTrAdaBoost up to $E_\tau = 60$.

4 Conclusion

In this paper, we integrate a global parameter into the transfer learning algorithm to reduce the amount of training effort required for cell detection. Our method is able to achieve the performance of previous boosting-based algorithms with only 10% of the training effort. To further improve our algorithm, we plan to incorporate additional global features to handle overlapping cells and investigate in the automated estimation of the global parameter. We believe that these results demonstrate the potential of the proposed method for greater applicability in cell detection by reducing the amount of manual effort.

References

[1] Balagopalan, L., Sherman, E., Barr, V., Samelson, L.: Imaging techniques for assaying lymphocyte activation in action. Nat Rev Immunol 11(1), 21–33 (2011)

[2] Carpenter, A., Jones, T., Lamprecht, M., Clarke, C., Kang, I., Friman, O., Guertin, D., Moffat, J., et al.: Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biology 7(10), R100 (2006)

[3] Freund, Y., Schapire, R.: A short introduction to boosting. Japonese Society for Artificial Intelligence 14(5), 771–780 (1999)

[4] Hodneland, E., Bukoreshtliev, N., Eichler, T., Tai, X.C., Gurke, S., Lundervold, A., Gerdes, H.H.: A unified framework for automated 3-d segmentation of surface-stained living cells and a comprehensive segmentation evaluation. Medical Imaging, IEEE Transactions on 28(5), 720–738 (May 2009)

[5] Kullback, S.: The kullback-leibler distance. The American Statistician 41(4), 340–341 (1987)

[6] Mukherjee, D., Ray, N., Acton, S.: Level set analysis for leukocyte detection and tracking. Image Processing, IEEE Transactions on 13(4), 562–572 (April 2004)

[7] Nguyen, N., Keller, S., Huynh, T., Shin, M.: Tracking colliding cells. In: IEEE Workshop in Applications of Computer Vision (2009)

[8] Pan, J., Kanade, T., Chen, M.: Heterogeneous conditional random field: Realizing joint detection and segmentation of cell regions in microscopic images. Computer Vision and Pattern Recognition (CVPR), 2010 on pp. 2940–2947 (2010)

[9] Toomre, D., Bewersdorf, J.: A new wave of cellular imaging. Annual Review of Cell and Developmental Biology (Jan 2010)

[10] Yao, Y., Doretto, G.: Boosting for transfer learning with multiple sources. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 1855 –1862 (jun 2010)