applied sciences

MDPI

Article

A Cell Counting Framework Based on Random Forest and Density Map

Ni Jiang and Feihong Yu *

Department of Optical Engineering, Zhejiang University, No. 38, Zheda Road, Hangzhou 310027, China; jiangni@zju.edu.cn

* Correspondence: feihong@zju.edu.cn

Received: 4 November 2020; Accepted: 22 November 2020; Published: 24 November 2020

Abstract: Cell counting is a fundamental part of biomedical and pathological research. Predicting a density map is the mainstream method to count cells. As an easy-trained and well-generalized model, the random forest is often used to learn the cell images and predict the density maps. However, it cannot predict the data that are beyond the training data, which may result in underestimation. To overcome this problem, we propose a cell counting framework to predict the density map by detecting cells. The cell counting framework contains two parts: the training data preparation and the detection framework. The former makes sure that the cells can be detected even when overlapping, and the latter makes sure the count result accurate and robust. The proposed method uses multiple random forests to predict various probability maps where the cells can be detected by Hessian matrix. Take all the detection results into consideration to get the density map and achieve better performance. We conducted experiments on three public cell datasets. Experimental results showed that the proposed model performs better than the traditional random forest (RF) in terms of accuracy and robustness, and even superior to some state-of-the-art deep learning models. Especially when the training data are small, which is the usual case in cell counting, the count errors on VGG cells, and MBM cells were decreased from 3.4 to 2.9, from 11.3 to 9.3, respectively. The proposed model can obtain the lowest count error and achieves state-of-the-art.

Keywords: cell count; random forest; detection; density map

1. Introduction

In biomedicine and pathology, the number of cells is a significant indicator of cell analysis. Initially, the counting task is executed by naked eyes and the result is sensitive to many objective factors, such as the parameters of the microscope, the image contrast, non-uniform illumination, various shapes and sizes, or the overlap of cells. In these circumstances, the counting task becomes more difficult and time-consuming. Furthermore, the counting result may vary from person to person. To automate the process of counting and make the task easier to be implemented, many algorithms have been proposed. With the prior information about the appearances of cells, some morphology-based methods extract cells from the background [1–3]. They can only handle well the images with separated cells. To count the overlapped cells, the density-based model is proposed [4]. The density-based model learns the relationship between the input cell image and density map which can be integrated to obtain the number of cells. Considering the powerful feature representation, convolutional neural networks (CNNs) are often used to learn the mapping from the source domain to the target domain. The non-linear mapping makes it possible to fit any function. However, CNN has too many parameters to set and it is not friendly to the non-professionals. More importantly, the backpropagation requires a large number of floating point operations, which means it is almost impractical to train a network on central processing unit (CPU) and graphics processing unit (GPU) support is necessary. By contrast,

random forest (RF) has fewer parameters to set and it can be trained on CPU efficiently. In terms of this, RF is more likely to be applied in practice. There have been some models and applications based on RF [5,6]. Reviewing the algorithm of RF, it predicts by collecting the values of inputs that arrive at the same leaf node. Namely, it is the statistics of the subset of inputs. When the cells in the testing set are more crowded or less crowded than the cells in the training set, the density value is out of the input scope. In this circumstance, the prediction will always be lower or higher than the ground truth. Therefore, it is not proper to directly predict the density map via RF. To solve this problem, we proposed a cell counting framework (CCF) based on RF to predict the density map by detecting cells. First, we defined a probability map as the ground truth to make sure that the centroids of cells can be detected even when cells are overlapped. With the well-prepared training data, cells can be detected robustly by Hessian matrix on the output of RF. The proposed CCF contains multiple RFs that have various decisions about the cell detection result. To decrease the risk of false detection, we averaged all detection results and turn the average into the density map by Gaussian function. In this way, the accuracy can be improved. Compared to the traditional counting methods based on detection [7] or density map [5], our proposed CCF has the following advantages. (I) Cell detection is sensitive to the output of RF. The proposed method combines various detection results to decrease the count error and improves the robustness. (II) With the same number of decision trees (DTs), using multiple small RFs can avoid overfitting better than using a large RF. (III) Traditional density map prediction aims to recover the pixel-wise density value, so the pixel-wise error is aggregated which is related to the total count error. Using RFs to predict the probability map to detect cells has a lower accuracy requirement than the density map prediction. Besides that, we also extracted the feature patch with a stride to reduce the computational overhead.

In summary, the contributions of this paper are as follows:

1. We defined a probability map to describe the cell locations. Compared to the maps generated by the Gaussian function [4,8] or distance function [7], training data are more balanced and the proposed probability map can better distinguish the cell centroids.
2. A dilated feature extraction was proposed to reduce the spatial redundancy and mitigate computation overhead.
3. We proposed a CCF that contains multiple RFs. Different RFs will generate different probability maps. Taking advantage of the proposed probability map, we suggested to locating the cells by the eigenvalue of the Hessian matrix. After combining all the detection results, the overall count error was decreased.
4. We validated the proposed model on three different kinds of cell datasets and the results prove that our model is competitive to the CNN-based models and even better. In addition, the comparison results between the proposed CCF and the individual RF prove that the CCF has lower bias and variance than the individual RF.

2. Related Work

Since Lempitsky and Zisserman creatively put forward a learning framework for object counting [1], the counting task has made great progress. They cast the counting task as the prediction of the objects' distributions and placed a dot near the center of an object to represent the object. With the discrete and sparse labels, it was tough to predict objects. To solve the problem, they defined a density function to smooth the isolated labels. The ultimate task was to predict the pixel-wise density value and the sum over the images was equal to the predicted count. In contrast, Fiaschi et al. proposed to learn the mapping between the feature vector and density by a regression forest with structured labels [2]. This model is very efficient and is an open-source project in an interactive toolkit [3]. Oman et al. attempted to count cells using two RFs [4]. One judged if there is any cell in the superpixel and the other one predicted the number of cells in each superpixel.

The CNNs are also popular in the counting task. Xie et al. used a fully convolutional regression network (FCRN) to estimate the density map and it performed well in both synthetic images and

real images [5]. Cohen et al. proposed to count cells by redundant counting [6] and improved the performance further. On the predicted density map, cells can be located by finding the local maxima or applying the non-maxima suppression algorithm [7–9]. Zhu et al. took the fully convolutional network (FCN) as the backbone and found the local maxima beyond the threshold on predicted density maps, the detection result was regarded as the counting results [10]. Rad et al. combined the U-Net [11] with the residual network [12] and multi-scale dilated network [13] to enlarge the receptive fields and located the centroids of cells on predicted maps [14]. Xie et al. presented a CNN to regress structured patches and the detections get more robust [15]. Another way to count cells is by detecting cells one by one. Ma et al. detected objects by integer programming on the predicted density map, which worked for small instances [16]. Akram et al. adopted two CNNs to segment cells, whose by-product was the number of cells. The first CNN predicted cell proposals with bounding boxes and the second CNN predicted the masks for segmented cells [17]. Xue leveraged the sparsity of labels to encode the positions of cells by compressive sensing and recovered the positions by decoding the prediction [18]. Recently, the attention mechanism [19,20] has been widely used to improve the performances of CNN-based models. U-Net with a self-attention module (named SAU-Net) [21] incorporates a self-attention module to explore the long-range dependencies between pixels by a huge attention matrix.

Excluded the learning frameworks, some approaches based on traditional image processing techniques can also count cells automatically. Maitra et al. used Hough transform to detect red blood cells [22], but it was highly dependent on the shape of cells. Faustino et al. leveraged the luminance information of fluorescence cells and treated the appearance as a topological surface, then analyzed the gray histogram and divided the image into different connected components [23]. The count result was obtained by selecting the target components. It is obvious that the method requires the particular appearance of cells and thus it cannot handle the complex images. To count the cell clusters, researchers tried to take advantage of the concavity where cells overlap to analyze how many cells the cluster contains. Kothari et al. implemented the counting task in two steps [24]. The first was to extract the cluster's edge and the second was to separate the single cells by detecting concavities. The premise of this work is the clear boundaries. The defocused cells or debris will degrade the detection. The distance transform algorithm is a common way to segment cell clusters. However, it may fail when cells overlap too much. Zhang et al. calculated the weights by curvature rather than distance and the cell cluster can be segmented well [25]. The disadvantage is the edges of cell regions have to be detected first which greatly influences the calculation of curvature weights. Jung et al. first applied distance transform and viewed the result as a mixture of Gaussians, then adopted linear discriminant analysis (LDA) to separate cells [26].

3. Method

Figure 1 shows the overview of the proposed CCF. First, the training data should be well prepared. Based on the training pipeline, we constructed the probability map as ground truth that can help detect cells. To improve computational efficiency, we proposed to extract features with a stride, as the dilated convolution [27] does. The labeled data and the extracted feature vectors compose the training data in pairs. Then, these individual RFs are trained with random and fixed training data subsets and predict the probability map where the cells can be detected by Hessian matrix. All the detection maps are averaged to generate the final density map. Both training data preparation and the detection framework contribute to the high performance.

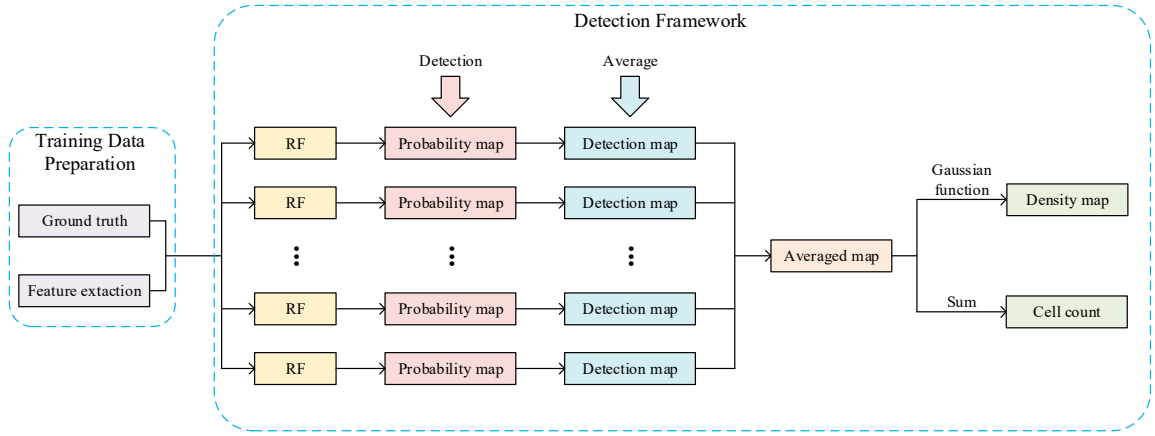

Figure 1. The overview of the proposed cell counting framework (CCF). After the training data preparation, the random forests (RFs) can be trained. The probability maps are predicted first where the centroids of cells are easier to be detected. Take the decisions of all RFs into consideration, the final count result is decided by multiple detection maps rather than a detection map, which can improve accuracy and robustness.

This section is divided into two parts to respectively introduce how the training data are prepared, how we detect cells on the predicted probability map and why the proposed CCF can improve accuracy and robustness.

3.1. Training Data Preparation

The training data consists of the label data and the feature vector. In this subsection, we will introduce the probability map, namely the ground truth, where the label data are sampled, and how the feature vector is generated.

3.1.1. Ground Truth Generation

As we have analyzed in Section 1, the RF has two drawbacks when predicting the density map. The one is that when training data are imbalanced the output will be inclined to majority data, and the other one is that the output is always limited in the range of input label data. Let's take the most used Gaussian function as an example to explain it. Each cell is labeled by a dot and the ground truth is the convolution result of Gaussian function and the labeled dot map. Figure 2 is the ground truth and the histogram distribution of the labeled region. It can be observed that the data with high density values are in minority, which may result in underestimation. However, if the DT gets deeper to learn the minority data, there is a risk of overfitting. In addition, even the data are balanced, RF still cannot predict the data that are beyond the scope of the training data. In terms of accuracy, it is improper for RF to predict density map directly.

To mitigate the above problems, we proposed a novel way to generate the ground truth which is defined as the probability map.

First, we used a blob to label a cell rather than a dot. This can make sure that the distribution of data is more identical after convolution. Second, the limited output of RF impedes the accuracy of density map. Therefore, we chose to count cells by detecting the centroids of cells, rather than by predicting density map directly. The convolution filter is designed as:

$$f(x,\sigma) = G(x,\sigma) + \delta(x - x_c) * G(x,\sigma) \tag{1}$$

where, G is a Gaussian function, δ is a pulse function, σ controls the spread, and x_c is the center of the filter. Note that there is a $\delta(x)$ function in the convolution filter, which makes it different from the Gaussian function.

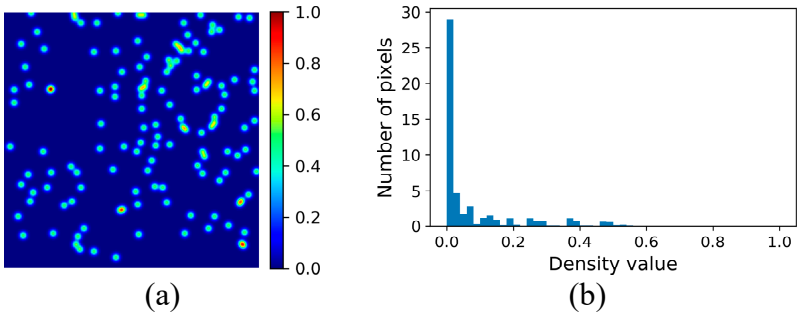

Figure 2. The ground truth that is generated by the Gaussian function. (**a**) The density map. (**b**) The histogram.

Figure 3 is the generated probability map. Compare the histograms in Figures 2b and 3b, it can be observed that data in Figure 3a are more equally distributed. In this way, the training data get more balanced without any extra operations. Figure 4a is the probability map generated by Gaussian function and Figure 4b is the probability map we defined. It can be noticed that when cells overlap the centroids of cells in Figure 4a cannot be distinguished anymore but the centroids of cells in Figure 4b still are highlighted which is helpful to locate cells.

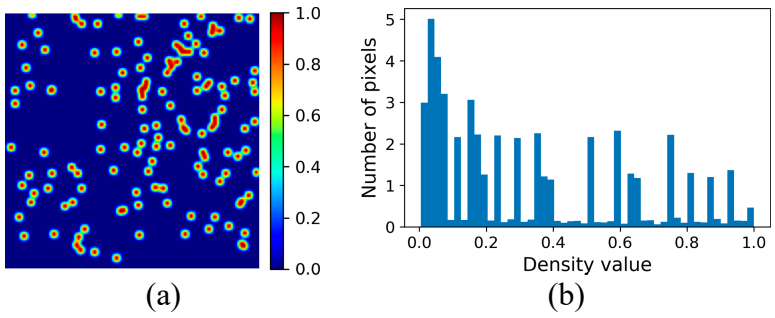

Figure 3. The probability map. (**a**) The proposed probability map. (**b**) The histogram.

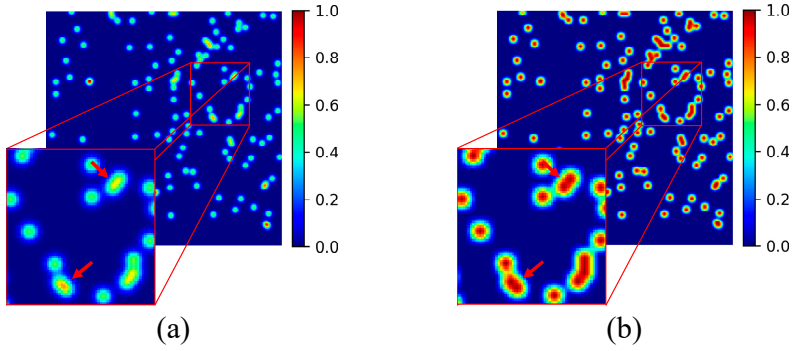

Figure 4. The differences between the two probability maps. (**a**) The probability map that is generated by the Gaussian function. (**b**) The proposed probability map.

3.1.2. Dilated Feature Extraction

Following the previous RF-based work [2], we generated the feature maps by calculating Laplacian of Gaussian, Gaussian gradient magnitude, and two eigenvalues of the structure tensor at scales 0.8, 1.6, 3.2, and the original image is also as a feature map. Therefore, there are 13 feature maps in all. To take the pixels in a local vicinity into consideration, we used all pixels in an $N \times N$ patch to describe the current pixel and the feature vector is $N \times N \times 13$ dimensions.

To learn context information as much as possible, the patch size should be large. However, limited by the memory size, the RF is unsuitable to process the data with a high dimension. There is a

conflict between the quantity of context information and the feature dimension. Inspired by dilated convolution [27], we extracted features with a pixel stride. Assuming we extracted a 7×7 patch as shown in Figure 5a, the left branch is the full-size patch and the feature vector has 49 dimensions. The right branch is the dilated extraction and the feature vector size is reduced to 25 dimensions. The computation overhead is cut off by 50%. Actually, the features between the neighboring pixels are similar, so the dilated feature extraction can not only improve computational efficiency but also reduce redundancy. For a group of feature maps, the extracted positions in adjacent feature maps are stagger, which makes the context and location in complementary and helps to leverage all the pixels in the patch.

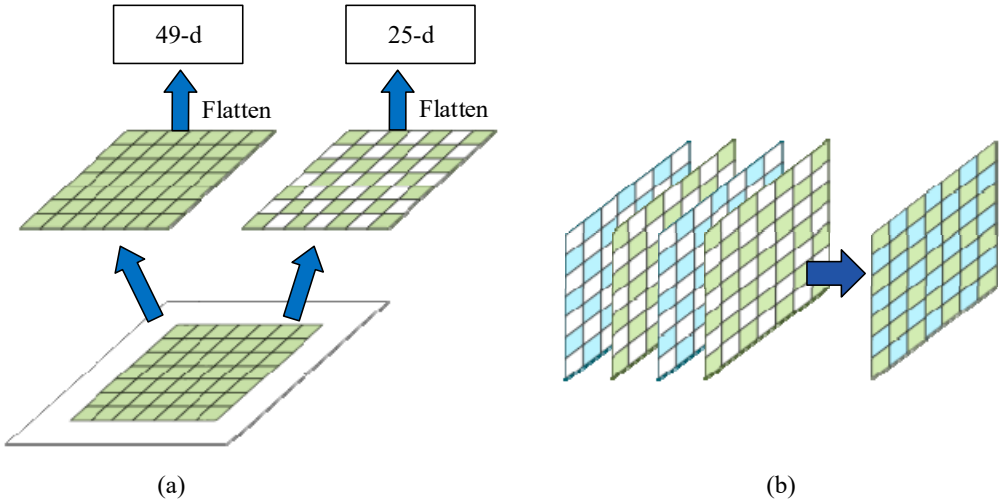

Figure 5. Illustration of dilated feature extraction. (**a**) The comparison between full-size extraction and dilated extraction. (**b**) The stacked feature maps contain both context information and position information.

Different kinds of cells usually are various in size, it is impossible to use the same patch size to extract feature vector for different kinds of cells. It is better to adjust the patch size, or the pixel interval, or the image resolution to fit different cells' scales.

3.2. Detection Framework

The detection framework shown in Figure 6 is the core work of our proposed method. It has three steps. (1) Probability map prediction. Train RFs and make the predicted probability close to the ground truth. (2) Detection map prediction. Take advantage of the defined probability map, cells are detected by the eigenvalue of Hessian matrix. (3) Density map prediction. Based on the previous two steps, there are multiple different detection maps and the proposed CCF considers each detection map equally to generate the final density map.

The proposed model is an ensemble model whose base learner is the RF [2] with a robust detection. Equipped with the robust detection, cells can be located on the predicted probability map. Each RF plays as an expert and has its own decision about the detection result. Finally, the predicted density map, whose sum is the number of cells, is the averaging of the all detection result.

3.2.1. Probability Map Prediction

For each base learner, the training data are sampled with replacement and then train the base learner with all sampled data. Due to the different training data and random feature split, each RF has a unique structure. Hence, they will output different probability maps even though the inputs are the same for each RF. The probability maps reflect the coarse locations of cells and the centroids of cells are supposed to be the local maxima. The probability map is used to detect the cell centroids to count cells

in the next step, and it is easier than the methods [1,2] that counting cells by directly predicting the density map.

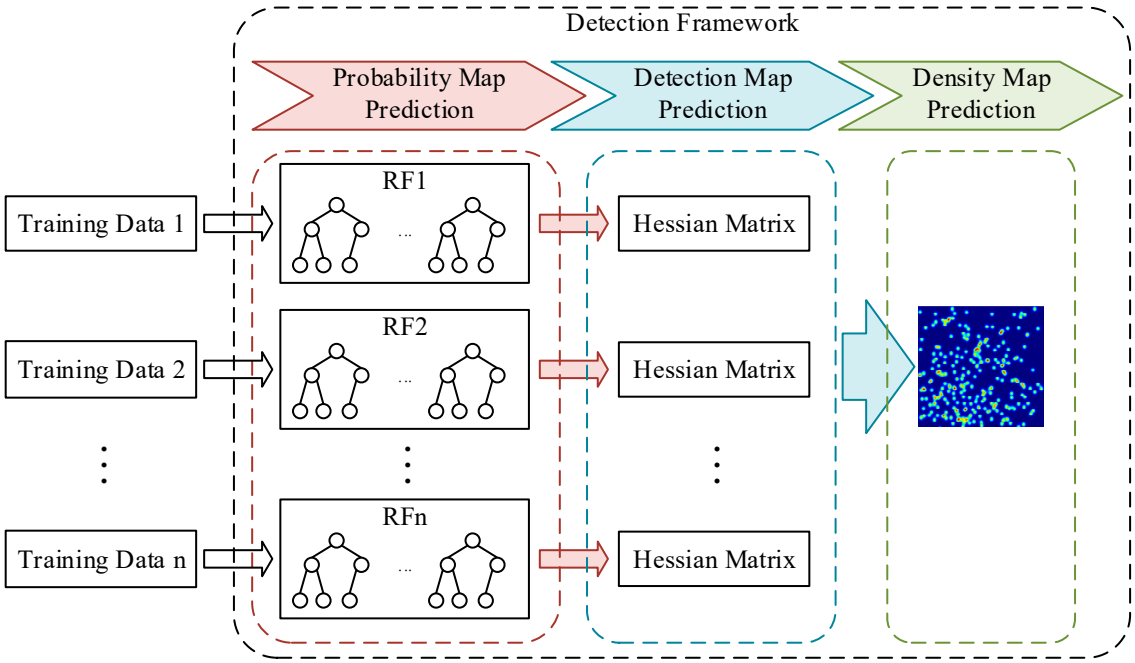

Figure 6. The detection framework.

3.2.2. Detection Map Generation

Many works try to detect cells by finding the local maxima. However, we found it is not robust enough sometimes. Figure 7a shows the ground truth of a probability map we defined. In practice, the predicted density map could not recover the distribution perfectly. There may be some noisy points that are the local maxima but not the true centroids, which results in a false detection. To detect cells more robustly, we presented a curvature-based detection algorithm. Supporting by the ground truth, the centroids of cells are at peaks and values decrease along the radial, which can be well-captured by the eigenvalues of Hessian matrix. This is the basis we detect cells. For a two-dimensional image I, the Hessian matrix is given as:

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix} \tag{2}$$

where I_{xx}, I_{xy}, I_{yx}, I_{yy} are the second partial derivatives of an image I. The eigenvalues of Hessian matrix indicate the curvature and they are defined as

$$\lambda_1 = \frac{1}{2}\left(I_{xx} + I_{yy} + \sqrt{\left(I_{xx} - I_{yy}\right)^2 + 4I_{xy}^2}\right) \tag{3}$$

$$\lambda_2 = \frac{1}{2}\left(I_{xx} + I_{yy} - \sqrt{\left(I_{xx} - I_{yy}\right)^2 + 4I_{xy}^2}\right) \tag{4}$$

From Equations (3) and (4), it can be inferred that the λ_1 and λ_2 reflect the changes of gradient. We use the signs of eigenvalues to ensure the cell regions and the values of eigenvalues to detect the centroids of cells. At different points of the image I, the differences between λ_1 and λ_2 are shown in Figure 7b,c.

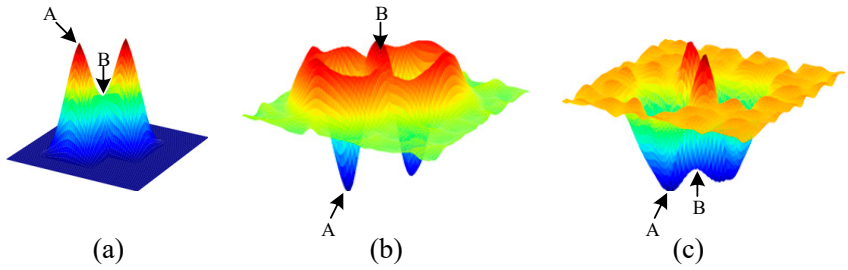

Figure 7. The eigenvalues of the Hessian matrix. (**a**) The ground truth, (**b**) The λ_1 of Hessian matrix, (**c**) The λ_2 of the Hessian matrix.

Figure 8a explains the process of cell detection. First, we needed to set a threshold to segment the cell region proposals from the probability map. Next, we would detect the centroids of cells in the cell region proposals by curvature. The eigenvalues of the Hessian matrix reflect the curvature. Figure 7b,c are the two eigenvalues λ_1 and λ_2 of Figure 7a. It can be observed that both λ_1, and λ_2 of A are negative and λ_1 and λ_2 of B are opposite in sign which means it is not the centroid of cells. Hence we use λ_1 to further select the cell region proposals where λ_1 should be negative, and the saddle points like B are abandoned. Finally, the centroids of cells can be detected by local minima of λ_1. The two cell region proposals in Figure 8a indicates that $\lambda_1 < 0$ can help to narrow the cell detection region further, which makes the detection results more robust.

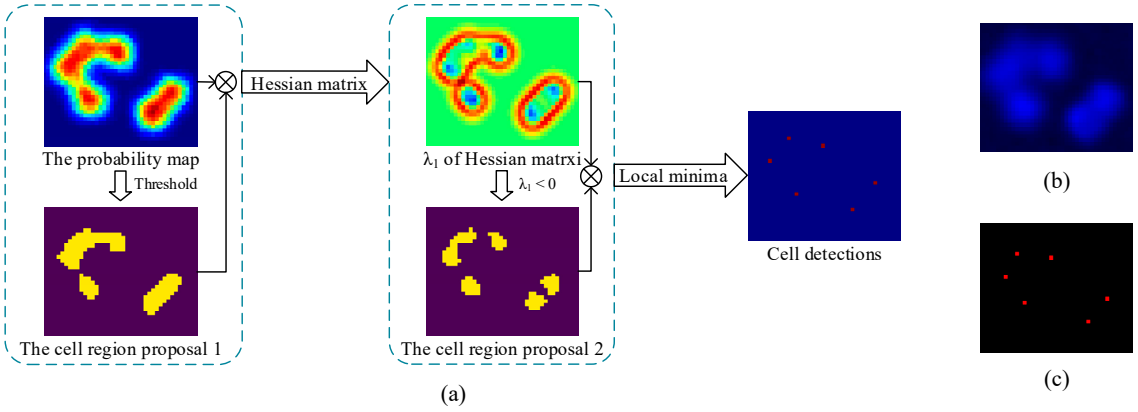

Figure 8. The illustration of the detection map generation. (**a**) The detection map generation. (**b**) Cell image. (**c**) Cell annotations.

3.2.3. Density Map Generation

The last step of the detection framework is averaging all detection maps predicted in the previous step. This step is very key to refine the count result because it benefits from the overall CCF including the well-prepared training data, the ensemble of RFs, and the detection based on the probability map.

Next, we will explain the reason why the average can refine the count. Four cases are listed for discussion in Figure 9. The red circle represents the true cell region, the box represents the local region belonging to different detection maps and the colorful patch is the corresponding detected cell. The left of Figure 9a–d is the stack of detection maps, and the right is the averaged detection result which is denoted as the lighter colors. Assuming there are 4 detection maps, and each detected cell in the averaged detection map denotes 0.25 cells. (a) All the detection maps have a detected cell in the cell region, so the result is that there is a cell. (b) Three detection maps have a detected cell in the cell region and the result is 0.75 cells. (d) Two detection maps have a detected cell in the cell region and the result is 0.5 cells. (c) Only one probability map detected a cell in the region, which means the prediction is disapproved by most probability maps. Therefore, there are only 0.25 cells and the detection is suppressed.

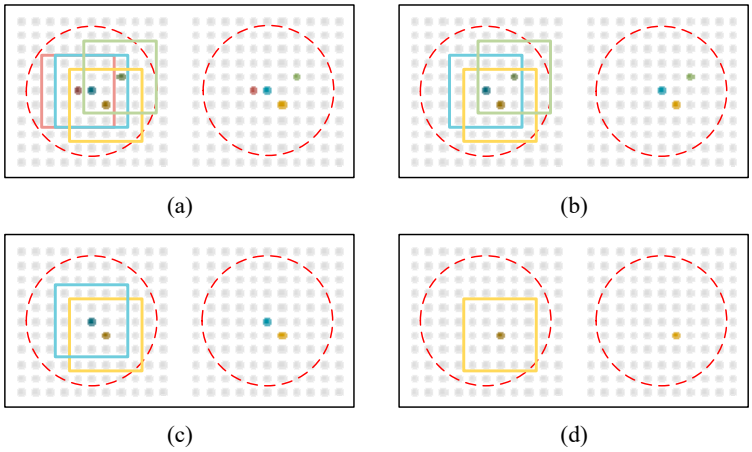

Figure 9. The diagram of detection is preserved or suppressed. (**a**) A cell is detected by 4 RFs. (**b**) A cell is detected by 3 RFs. (**c**) A cell is detected by 2 RFs. (**d**) A cell is detected by 1 RF.

In summary, the density map generation process can be described as:

$$D_{avg} = \frac{1}{N}\sum_{i=1}^{N} D_i \tag{5}$$

$$density\ map = G(\sigma) \otimes D_{avg} \tag{6}$$

where, N represents how many RFs the proposed CCF contains, D_i denotes the detection result of the i-th RF and D_{avg} denotes the averaged detection result, and they all maintain the same resolution as the input image. G is the Gaussian function and σ controls the degree of smoothness. $\otimes$ means the convolution operation.

The training and testing of RFs are independent. Therefore, the proposed model can be running in parallel to improve computational efficiency.

4. Experiments and Results

4.1. Datasets

In this paper, we validated our proposed CCF on three public cell datasets. Three sample images are shown in Figure 10. The VGG cells dataset [1] is a synthesis dataset that simulates the occlusion, out-of-focus blur, non-uniform luminance, and various density in real images. The modified bone marrow (MBM) cells dataset [6,7] is introduced by Kainz et al. [7] and it is about the bone marrow (BM) from eight patients. Cohen et al. updated the annotations and divided the dataset from 11 images to 44 images by cropping [6]. The images are filled with staining fragments that challenge to count. The Adipocyte cells dataset [6,28] comes from the Genotype Tissue Expression Consortium [28] and following Cohen's setting [6], images are down-sampled to 150×150. The Adipocyte cells are adherent severely and varies in size and shape drastically.

The three datasets are summarized in Table 1, where the average means the average number of cells per image, the N_{train} and the N_{test} denote the split of the dataset for training and testing respectively. To reduce the computational overhead, we resized the images of MBM cells to 300×300. In this paper, we assigned 10 RFs to the proposed CCF and each RF consists of 20 DTs. For VGG cells and Adipocyte cells, the max depth of DTs is limited in 20. Because of the small size of MBM cells, the max depth of DTs is limited in 15.

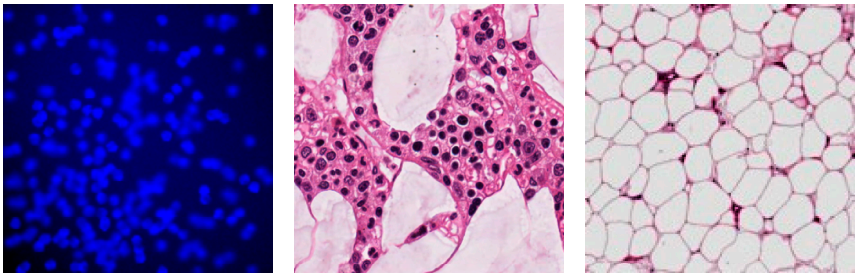

Figure 10. Sample images from three datasets. (**a**) VGG cells. (**b**) MBM cells. (**c**) Adipocyte cells.

Table 1. The summary of three datasets.

Dataset	Resolution	Average	N_{train}	N_{test}
VGG cells [1]	256×256	174 ± 64	100	100
MBM cells [6,7]	600×600	126 ± 33	30	14
Adipocyte cells [28]	150×150	165 ± 44	100	100

4.2. Evaluation Metric

For each dataset, the training set and the validation set are sampled from the N_{train} split training images randomly and equally. After the proposed model is trained, the testing images are used to evaluate the performance of the proposed model. Same as the previous works [5,6,21,29], the performance of the result was measured by mean absolute error (MAE), which is formulated as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{y}_i - y_i\right| \tag{7}$$

where, N is the number of test images, y_i represents the true count and $\hat{y}_i$ represents the predicted count of the i-th image.

For each group training images, the experiment is repeated for 10 times. The mean MAE and the standard deviation will be shown next.

4.3. Results and Discussion

In this subsection, we displayed the comparison results between the proposed CCF and state-of-the-art methods on three datasets. In addition, we provided the primary density maps that are generated by all RFs to show the effectiveness of CCF. The bias and variance on the whole test sets are also compared to prove that CCF performs better than RF.

4.3.1. VGG Cells

Table 2 exhibits the comparison results on the VGG cells when $N = 8$, $N = 16$, $N = 32$, and $N = 50$, where the training set and the validation set are sampled randomly from the split training images, respectively. The methods from the first row to the third row are based on machine learning models. It can be observed that CCF performs much better than them and even better than the CNN-based models which are listed in the fourth row and fifth row. When the training data increases, the count-ception model outperforms CCF and becomes the best model. Even so, the proposed model is competitive to state-of-the-art approaches, and the superiority is highlighted when training data are scarce. Besides, the count-ception model improved performance with the compromise of location information and the predicted map is poorly visible. In contrast, our predicted density map is more meaningful, which is shown in Figure 11. For the test image in Figure 11, the ground truth of cell number is 306 and CCF predicted 306.4 cells. It reveals that CCF performs well both on the global

region and local regions. When N increases from 32 to 50, the improvement is almost invariable. The main reason is the dense cells are in minority and there is less context information to be learned than the sparse cells. Unless the information about the dense cells is supplemented, the increment of training data will not be helpful for the training model.

Table 2. The comparison of mean absolute error (MAE) on the VGG cells. The testing set is fixed and 10 times training with different training data and different network initializations are used to calculate the mean and standard deviation.

Method	N = 8	N = 16	N = 32	N = 50
Lempitsky et al. [1]	4.9 ± 0.7	3.8 ± 0.2	3.5 ± 0.2	N/A [1]
Fiaschi et al. [2]	3.4 ± 0.1	N/A	3.2 ± 0.1	N/A
Arteta et al. [30]	4.5 ± 0.6	3.8 ± 0.3	3.5 ± 0.1	N/A
FCRN-A [5]	3.9 ± 0.5	3.4 ± 0.2	2.9 ± 0.2	2.9 ± 0.2 [2]
SAU-Net [21]	N/A	N/A	N/A	2.6 ± 0.4 [2]
Count-ception [6]	3.9 ± 0.4	2.9 ± 0.5	2.4 ± 0.4	2.3 ± 0.4
CCF	2.9 ± 0.2	2.8 ± 0.1	2.6 ± 0.1	2.6 ± 0.1

[1] The references don't provide the data. [2] These models are trained on 64 images.

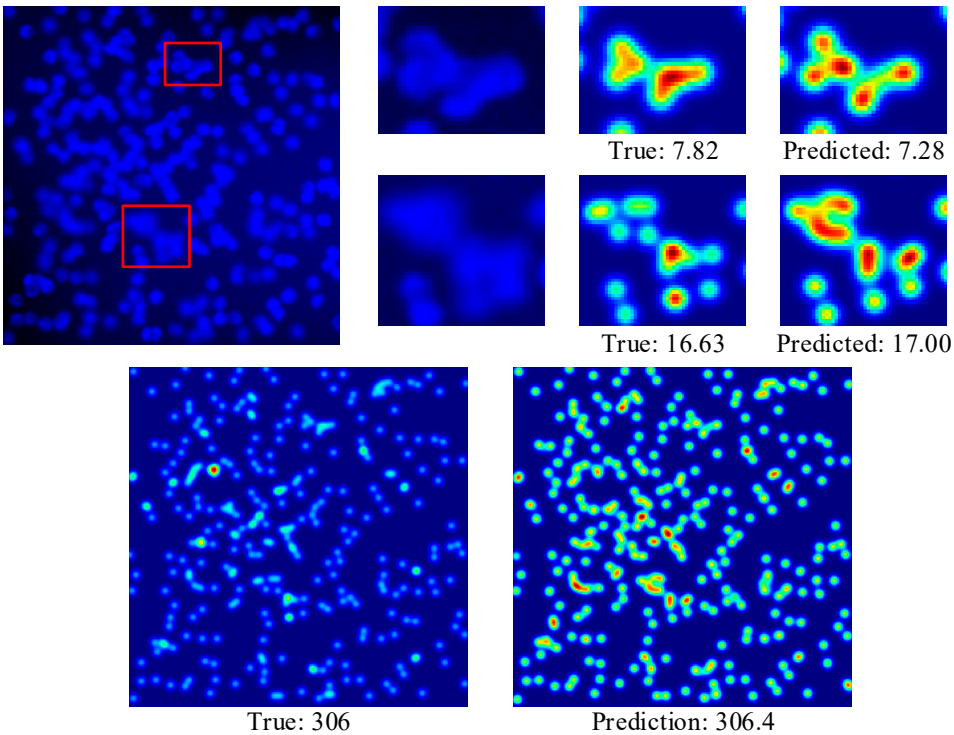

Figure 11. A test image of VGG cells. The two boxes highlighted in the input image contains high density overlapped cells that are difficult for human to count correctly. But the proposed CCF could predict the number of cells with acceptable errors. The second row displays the ground truth of the density map and the predicted density map.

4.3.2. MBM Cells

The greatest challenge of the MBM cells is the pink tissue and the fake particles. With the dilated extracted features, the patch in feature space is enlarged with the dimension reduced. The first two compared methods are based on CNNs and both of them count cells by predicting a density map. The comparison results listed in Table 3 indicate that CCF has the lowest MAE, which means it is more suitable to handle the counting problem without enough training data. Marsden et al. [31] designed a network that can be adaptive to various visual domains and it only outputs the number of objects.

Similar to CCF, Cell-Net [29] also counted cells by locating the centroids of cells. However, Cell-Net has many parameters to be learned and it is not suitable for the small datasets. In Figure 12, the predictions of two local patches show that there is no noise in the background and CCF can detect the separated cells well. In the full-size density maps, it can be observed that there are several uncertain locations which are not approved by all RFs. It is exactly the function that averaging all detection maps plays. The number of ground truth is 174 cells and the prediction is 168.2 cells.

Table 3. The comparison MAE on the MBM cells. The testing set is fixed and 10 times training with different training data and different network initializations are used to calculate the mean and standard deviation.

Method	N = 5	N = 10	N = 15
FCRN-A [5]	28.9 ± 22.6	22.2 ± 11.6	21.3 ± 9.4
Count-ception [10]	12.6 ± 3.0	10.7 ± 2.5	8.8 ± 2.3
Marsden et al. [31]	23.6 ± 4.6	21.5 ± 4.2	20.5 ± 3.5
Cell-Net [29]	11.3 ± 4.8	9.8 ± 3.2	N/A
CCF	9.3 ± 1.4	8.9 ± 0.9	8.6 ± 0.3

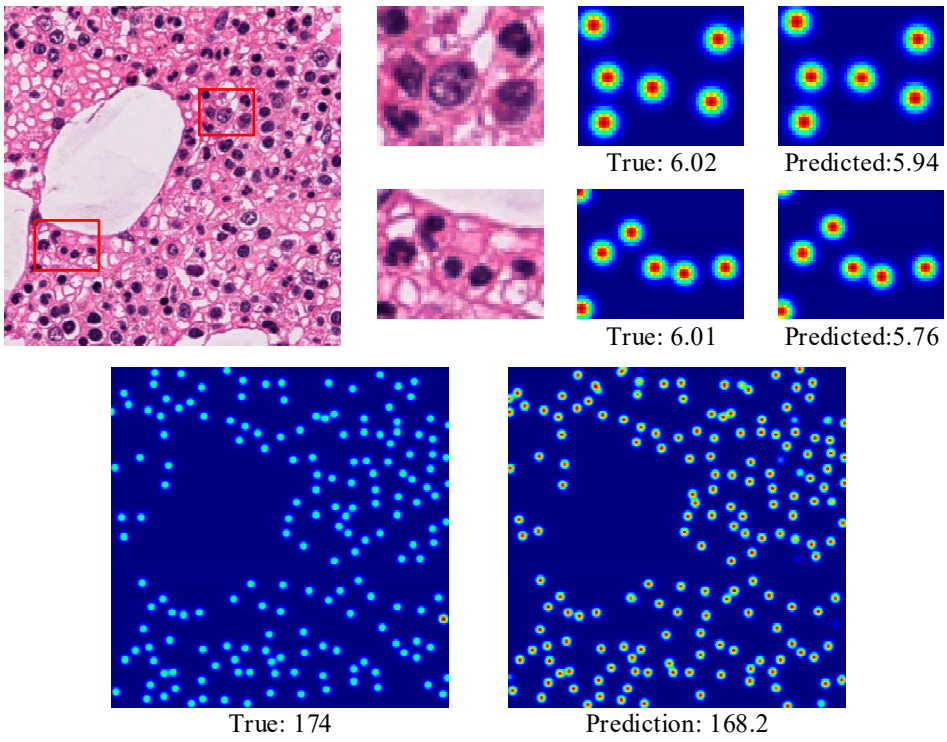

Figure 12. A test image of MBM cells. The cells in two close-ups differ in size and texture. The local predicted density maps show that each cell is detected well and there is no noise in the background which can avoid the accumulated error of background. The true density map and the predicted density map are shown on the second row.

4.3.3. Adipocyte Cells

In Adipocyte cells, there some empty regions that are similar to the adipocyte cells, and some small cells are crowded in large cells or occluded by tissues. The variety in size and shape makes the Adipocyte cells harder to count. In the experiment, CCF is compared to two open-source software [32,33], the count-ception model [6], and SAU-Net [21]. From the results listed in Table 4, it can be observed that CCF performs similarly to the Adiposoft [33]. Even more, the MAE is improved from 19.4 to 14.5 over the count-ception model and the proposed model achieves a 25% relative improvement. SAU-Net [21] is a U-net with a spatial attention block. With the spatial attention block,

SAU-Net can capture more context which is helpful to train the model. But it will cost much memory to calculate the attention matrix. Figure 13 shows two local patches that contain both small cells and large cells and it seems that CCF performs well. This image contains 129 cells and CCF predicted it contains 126.3 cells.

Table 4. The comparison MAE on the Adipocyte cells. The testing set is fixed and 10 times training with different training data and different network initializations are used to calculate the mean and standard deviation.

Method	N = 10	N = 25	N = 50
CellProfiler [32]	----------	25.06 ± 2.6	----------
Adiposoft [33]	----------	14.8 ± 13.6	----------
Count-ception [6]	25.1 ± 2.9	21.9 ± 2.8	19.4 ± 2.2
SAU-Net [21]	N/A	N/A	14.2 ± 1.6
CCF	16.9 ± 1.9	14.5 ± 0.4	14.5 ± 0.4

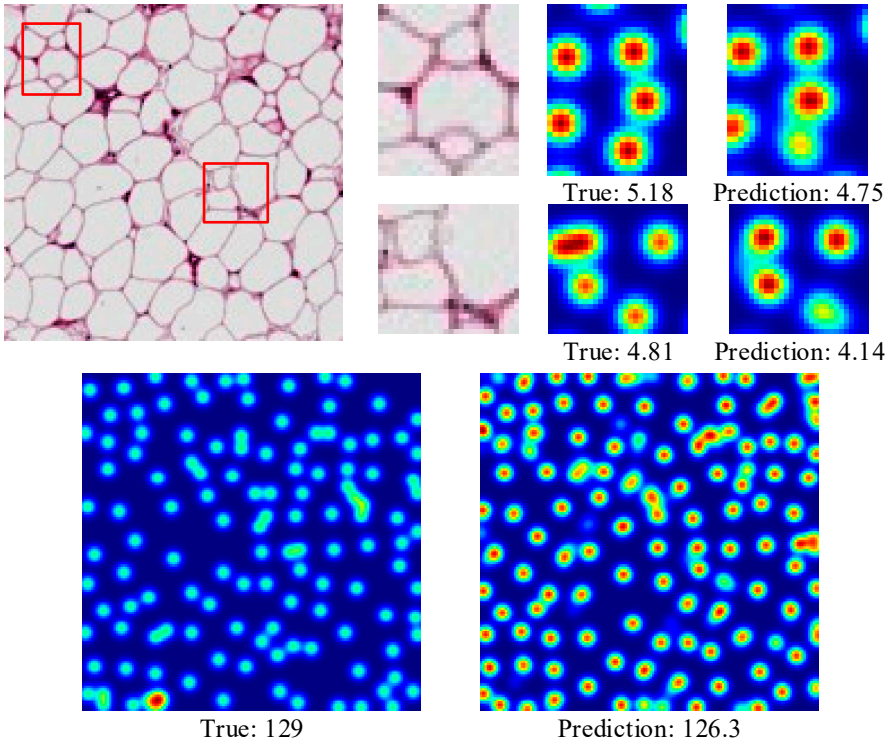

Figure 13. A test image of Adipocyte cells. The adipocyte cells are very various in shape and size. Some small cells are tightly surrounded by large cells. The results show that the proposed CCF can detect both small cells and large cells. The ground truth of the density map and the predicted density map are also listed.

4.3.4. The Effectiveness of CCF

To demonstrate the effectiveness of suppressing the wrong detections by average, the detection maps of 10 RFs have been converted to density maps, as shown in Figure 14. Figure 14a,b are the true density map and the final density map, respectively. Figure 14c–l are the predicted density maps of 10 RFs, which are the smooth versions of detection maps. If paying attention to the regions marked with red boxes, we could see that the cells detected by several RFs in the largest box and smallest box have lower densities in the final density map, which are shown as the color is close to blue. There are four cases to be discussed which are similar to Figure 9a–d. (a) The cells that are detected by all RFs are also well reserved in the final density map. (b) The medium box where most RFs detected a cell has the highest error in the final density map. (c) The result of the large box is supported by

several RFs, whose bias is lower. (d) In the smallest box, there is only an RF detected a cell so the detection contributes 0.1 cells to the final result. This indicates that detection with low confidence can be suppressed differently depending on how many detection maps contain it. In this way, the final predicted result gets more robust.

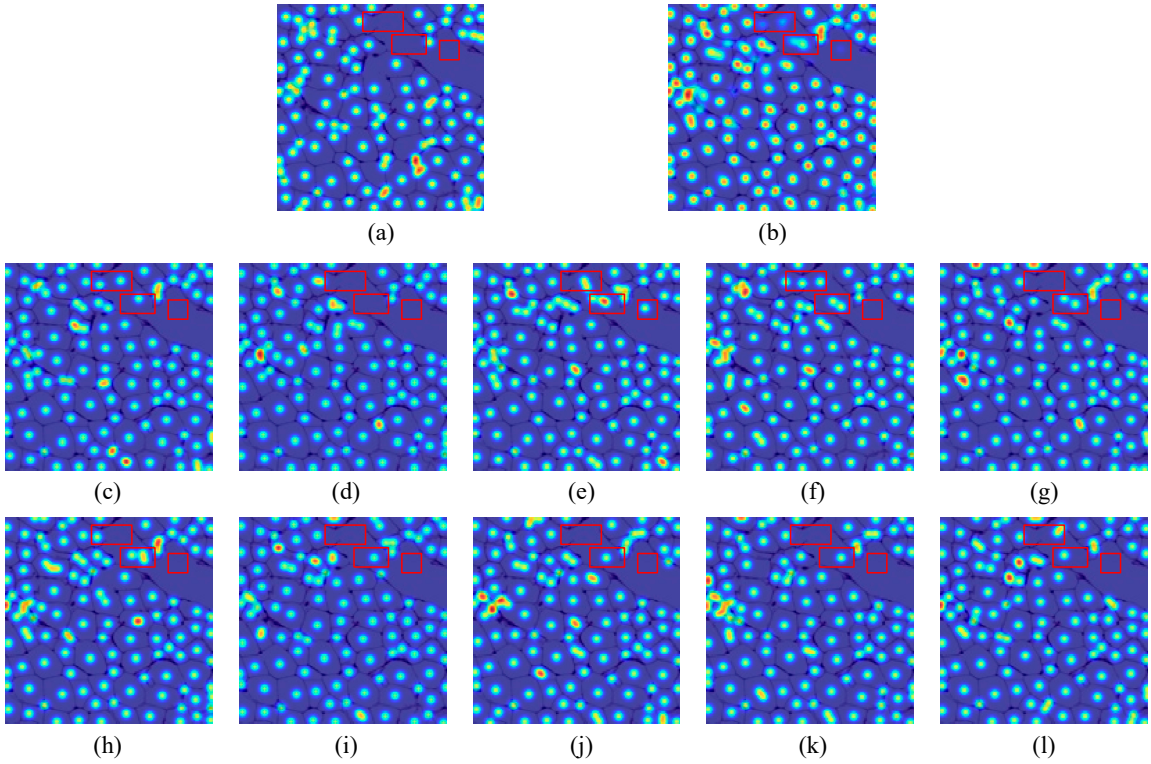

Figure 14. The illustration of the suppression effect by average: (**a**) the true density map, (**b**) the final predicted density map, and (**c**~l) are the predicted density maps for 10 RFs.

Someone may be curious that what happens if we use an RF that has as many DTs as CCF. Except for the number of DTs, we kept all parameters the same. However, we found the result gets worse. There are two reasons: (1) Too many trees cause overfitting. (2) The count result is decided by only one probability map and it is susceptible to noise, which is the motivation we proposed the CCF.

Besides, we also displayed the strength of CCF intuitively by curves. To compare the bias and variance between CCF and RF, we took MAE and mean square error (MSE) as the metric to evaluate performance. MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{8}$$

where, N is the number of images, $\hat{y}_i$ and y_i are the predicted count and the ground truth of the i-th image.

Figure 15 plots MAEs and MSEs on three datasets. The solid lines represent the results for the proposed CCF that consists of 10 RFs in this paper and the dot-dash lines represent the base RF. It can be observed that the MAEs of CCF are lower than the averaged MAE of 10 RFs and MSEs of CCF are also lower than the averaged MSE of 10 RFs, which means both the bias and variance are getting lower.

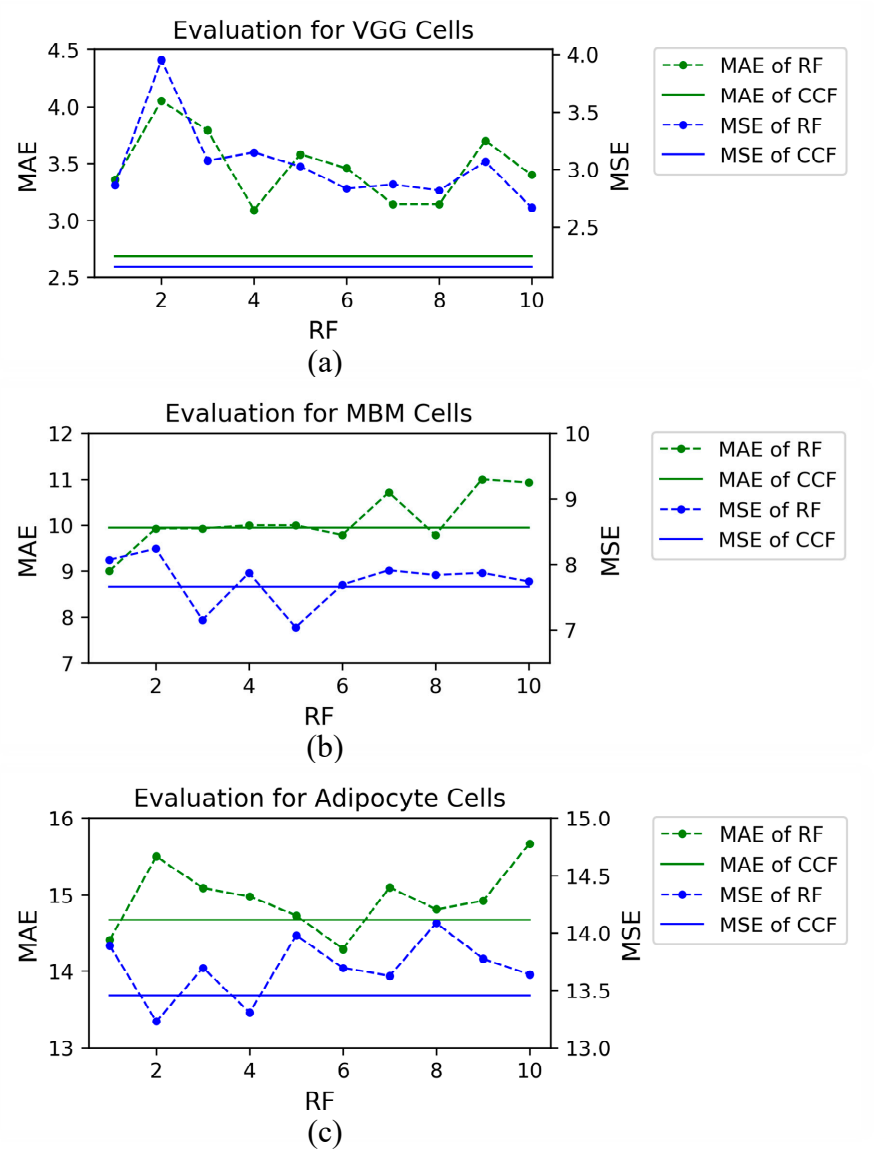

Figure 15. The Evaluation for three datasets. (**a**) The evaluation for VGG cells. (**b**) The evaluation for MBM cells. (**c**) The evaluation for Adipocyte cells.

5. Conclusions

In cell counting, the cell occlusions and scarce training data make the counting task difficult. To this end, we proposed a cell counting framework based on random forest (RF) and density map. In the proposed cell counting framework (CCF), the training data preparation and the detection framework are combined to provide an accurate and robust counting result. The training data preparation aims to make the cells can be better detected even when overlapping and reduce the computational overhead. The detection framework contains multiple RFs and each RF can predict a probability map where the cells can be detected robustly by Hessian matrix. To decrease the overall count error, we averaged all detection maps such that the false detection disapproved by most RFs can be suppressed. In this paper, we assigned 10 base RFs to the proposed CCF and proved that CCF outperforms the base RF in terms of MAE and MSE. The comparisons between CCF and some state-of-the-art methods show that CCF can achieve higher performance, especially when the training data are small. However, the cell images with annotations are usually difficult to collect. Therefore, the proposed CCF can serve as an automatic counting tool in cell image analysis. From training to testing, it does not require extra high hardware resources.

In this paper, the CCF is trained by handcrafted features that are low-level. Further, the receptive field of the extracted feature vector is also small. These factors limit the performance improvement. In future studies, we will try to use some classic pre-trained CNN models as the feature extractor to enhance the feature representation. At the last step of CCF, the final detection result is the average of all detection results, which means each base RF contributes equally to the final count result. To make the count result more accurate, we will explore how to re-weight base RFs to perform a better count. Furthermore, we will also focus on speeding up the training and testing processes by parallelization.

Author Contributions: Conceptualization, N.J. and F.Y.; methodology, investigation and resources, N.J.; writing—original draft preparation and software, N.J.; formal analysis, N.J., writing—review and editing, N.J. and F.Y.; validation, data curation and visualization, N.J.; supervision, project administration, F.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lempitsky, V.; Zisserman, A. Learning to count objects in images. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 6–9 December 2010; pp. 1324–1332.
2. Fiaschi, L.; Köthe, U.; Nair, R.; Hamprecht, F.A. Learning to count with regression forest and structured labels. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba, Japan, 11–15 November 2012; pp. 2685–2688.
3. Sommer, C.; Straehle, C.; Koethe, U.; Hamprecht, F.A. Ilastik: Interactive learning and segmentation toolkit. In Proceedings of the 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Chicago, IL, USA, 30 March–2 April 2011; pp. 230–233.
4. Magana-Tellez, O.; Vrigkas, M.; Nikou, C.; Kakadiaris, I.A. SPICE: Superpixel Classification for Cell Detection and Counting. In Proceedings of the 13th International Conference on Computer Vision Theory and Applications, Funchal, Madeira, Portugal, 27–29 January 2018; pp. 485–490.
5. Xie, W.; Noble, J.A.; Zisserman, A. Microscopy cell counting and detection with fully convolutional regression networks. *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.* **2018**, *6*, 283–292. [CrossRef]
6. Paul Cohen, J.; Boucher, G.; Glastonbury, C.A.; Lo, H.Z.; Bengio, Y. Count-ception: Counting by fully convolutional redundant counting. In Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 22–29 October 2017; pp. 18–26.
7. Kainz, P.; Urschler, M.; Schulter, S.; Wohlhart, P.; Lepetit, V. You should use regression to detect cells. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; pp. 276–283.
8. Pan, X.; Yang, D.; Li, L.; Liu, Z.; Yang, H.; Cao, Z.; He, Y.; Ma, Z.; Chen, Y. Cell detection in pathology and microscopy images with multi-scale fully convolutional neural networks. *World Wide Web* **2018**, *21*, 1721–1743. [CrossRef]
9. Liang, H.; Naik, A.; Williams, C.L.; Kapur, J.; Weller, D.S. Enhanced center coding for cell detection with convolutional neural networks. *arXiv* **2019**, arXiv:1904.08864.
10. Zhu, R.; Sui, D.; Qin, H.; Hao, A. An extended type cell detection and counting method based on FCN. In Proceedings of the 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), Washington, DC, USA, 23–25 October 2017; pp. 51–56.
11. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; pp. 234–241.
12. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
13. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2015**, arXiv:1511.07122.

14. Rad, R.M.; Saeedi, P.; Au, J.; Havelock, J. Blastomere cell counting and centroid localization in microscopic images of human embryo. In Proceedings of the 2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP), Vancouver, BC, Canada, 29–31 August 2018; pp. 1–6.
15. Xie, Y.; Xing, F.; Kong, X.; Su, H.; Yang, L. Beyond classification: Structured regression for robust cell detection using convolutional neural network. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; pp. 358–365.
16. Ma, Z.; Yu, L.; Chan, A.B. Small instance detection by integer programming on object density maps. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3689–3697.
17. Akram, S.U.; Kannala, J.; Eklund, L.; Heikkilä, J. Cell segmentation proposal network for microscopy image analysis. In Proceedings of the Deep Learning and Data Labeling for Medical Applications, Athens, Greece, 21 October 2016; pp. 21–29.
18. Xue, Y.; Ray, N. Cell Detection in microscopy images with deep convolutional neural network and compressed sensing. *arXiv* **2017**, arXiv:1708.03307.
19. Wang, X.; Girshick, R.; Gupta, A.; He, K. Non-local neural networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7794–7803.
20. Zhang, H.; Goodfellow, I.; Metaxas, D.; Odena, A. Self-attention generative adversarial networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7354–7363.
21. Guo, Y.; Stein, J.; Wu, G.; Krishnamurthy, A. SAU-Net: A Universal Deep Network for Cell Counting. In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA, 7–10 September 2019; pp. 299–306.
22. Maitra, M.; Gupta, R.K.; Mukherjee, M. Detection and counting of red blood cells in blood cell images using Hough transform. *Int. J. Comput. Appl.* **2012**, *53*, 13–17. [CrossRef]
23. Faustino, G.M.; Gattass, M.; Rehen, S.; de Lucena, C.J. Automatic embryonic stem cells detection and counting method in fluorescence microscopy images. In Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Boston, MA, USA, 28 June–1 July 2009; pp. 799–802.
24. Kothari, S.; Chaudry, Q.; Wang, M.D. Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques. In Proceedings of the 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Boston, MA, USA, 28 June–1 July 2009; pp. 795–798.
25. Zhang, C.; Sun, C.; Su, R.; Pham, T.D. Segmentation of clustered nuclei based on curvature weighting. In Proceedings of the 27th Conference on Image and Vision Computing, Dunedin, New Zealand, 26–28 November 2012; pp. 49–54.
26. Jung, C.; Kim, C.; Chae, S.W.; Oh, S. Unsupervised segmentation of overlapped nuclei using Bayesian classification. *IEEE Trans. Biomed. Eng.* **2010**, *57*, 2825–2832. [CrossRef] [PubMed]
27. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848. [CrossRef] [PubMed]
28. Lonsdale, J.; Thomas, J.; Salvatore, M.; Phillips, R.; Lo, E.; Shad, S.; Hasz, R.; Walters, G.; Garcia, F.; Young, N. The genotype-tissue expression (GTEx) project. *Nat. Genet.* **2013**, *45*, 580–585. [CrossRef] [PubMed]
29. Rad, R.M.; Saeedi, P.; Au, J.; Havelock, J. Cell-net: Embryonic cell counting and centroid localization via residual incremental atrous pyramid and progressive upsampling convolution. *IEEE Access* **2019**, *7*, 81945–81955.
30. Arteta, C.; Lempitsky, V.; Noble, J.A.; Zisserman, A. Interactive object counting. In Proceedings of the Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014; pp. 504–518.
31. Marsden, M.; McGuinness, K.; Little, S.; Keogh, C.E.; O'Connor, N.E. People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8070–8079.

32. Carpenter, A.E.; Jones, T.R.; Lamprecht, M.R.; Clarke, C.; Kang, I.H.; Friman, O.; Guertin, D.A.; Chang, J.H.; Lindquist, R.A.; Moffat, J. CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. *Genome Biol.* **2006**, *7*, R100. [CrossRef] [PubMed]
33. Galarraga, M.; Campión, J.; Muñoz-Barrutia, A.; Boqué, N.; Moreno, H.; Martínez, J.A.; Milagro, F.; Ortiz-de-Solórzano, C. Adiposoft: Automated software for the analysis of white adipose tissue cellularity in histological sections. *J. Lipid Res.* **2012**, *53*, 2791–2796. [CrossRef] [PubMed]