ELMGAN: A GAN-based Efficient Lightweight Multi-scale-feature-fusion Multi-task Model

Lijia Deng, Shui-Hua Wang, Yu-Dong Zhang*
1 School of Computing and Mathematical Sciences, University of Leicester, Leicester, UK
E-mails: ld232@le.ac.uk, shuihuawang@ieee.org, yudongzhang@ieee.org.
* Correspondence should be addressed to Yu-Dong Zhang

ABSTRACT

Cell segmentation and counting is a time-consuming and important experimental step in traditional biomedical research. Many current counting methods are Point-based methods which require exact cell locations. However, there are few such cell datasets with detailed object coordinates. Most existing cell datasets only have the total number of cells and a global segmentation annotation. To effectively use existing datasets, we divide the cell counting task into the cell's number prediction and cell segmentation. We propose a GAN-based efficient lightweight multi-scale-feature-fusion multi-task model (ELMGAN). To coordinate the learning of these two tasks, we propose a Norm-Combined Hybrid loss function (NH loss) and use the method of the generative adversarial network to train our networks. We propose a new Fold Beyond-nearest Upsampling method (FBU) in our lightweight and fast multi-scale-feature-fusion multi-task generator (LFMMG), which is twice as fast as the traditional interpolation upsampling method. We use multi-scale feature fusion technology to improve the quality of segmentation images. LFMMG reduces the number of parameters by nearly 50% compared with U-Net and gets better performance on cell segmentation. Compared with the traditional GAN model, our method improves the speed of image processing by nearly ten times. In addition, we also propose a Coordinated Multitasking Training Discriminator (CMTD) to refine the accuracy of the details of the features. Our method achieves non-Point-based counting that no longer needs to annotate the exact position of each cell in the image during the training and achieves excellent results in cell counting and segmentation.

KEYWORDS

Convolutional neural network, Generative adversarial networks, Cell segmentation, Cell counting

1 Introduction

Cell segmentation and counting have a very vital significance in medicine. The number of cells in a sample is an important measure of medical testing. For example, the disease can be verified by counting cells in a sample [1].

The cell counting plate−based counting method is the traditional cell counting method. Medical workers count the number of cells in each label repeated under the microscope to estimate the cell density of the whole sample. However, this method is manual work and has many shortcomings: At first, this is a labour-intensive job, which costs much time on monotonous physical work; Secondly, because of the use of manual counting and high intensity of repetitive labour, manually identifying cells may lead to uncontrollable subjective errors in statistics; Thirdly, the manual counting method lacks the quantization of the error range; Finally, although there are several devices available for rapid cell counting tasks, these devices are expensive and often lack the ability to locate images of the cells being counted.

In the past decade, deep convolutional networks have shown excellent performance in many visual recognition tasks [2-5]. It has become a trend to use computer technology for cell image recognition and statistics: He et al. designed a concatenated fully-convolutional-regression network to regress the density map of cells and count the number of cells in microscopy images [1]; Xie proposed a synthetic-data-trained two fully convolutional density regression networks for cell counting [5]; Graham et al. leveraged the instance-rich information encoded within the vertical and horizontal distances of nuclear pixels to their centres of mass to segment the nuclear [6]; Dan, Alessandro and et al. developed a deep max-pooling convolutional neural network to detect breast histology images [7]; Carlos et al. first used the Extremal Region Trees to detection of overlapping objects in microscopy [8]; Liu and Yang proposed general cell detection algorithm based on a deep convolutional neural network [9].

The training process of these methods is based on the accurate coordinate of each target on the image. Depending on these labelled points, these kinds of neural network models learn the characteristics of the labelled points and their vicinity. Therefore, these methods are point-based, trained on datasets with detailed cell coordinates annotation. And their datasets are required detailed coordinate labelling. However, the work of labelling is time-consuming and laborious. Such datasets need to be mathematically converted into density maps when

used. Such mathematical transformations are error-prone, resulting in an insufficient accuracy of the ground truth values. At present, most existing cell datasets only have segmentation annotation and the total cell count. To make use of these datasets, we propose the non-Point-based method, which can segment and count at the same time. However, in the process of training such a multi-task convolutional neural network, more manual design is usually needed to get better performance. Fortunately, the generative adversarial network (GAN) allows us to give a high-level target task simply and then automatically learn a loss function suitable for that task [10]. GANs can judge the authenticity of the generated image and try to make the generated image less fuzzy. Thus, GANs learn losses based on data situations, so they can be applied to tasks that require complex loss functions [11].

Therefore, we propose a GAN-based efficient lightweight multi-scale-feature-fusion multi-task model (ELMGAN) to segment and count cell images simultaneously. Furthermore, we design a lightweight and fast multi-scale-feature-fusion multi-task generator (LFMMG). Compared with the backbone model, this generator has obvious simplification in memory consumption and realizes the simultaneous generation of multiple prediction targets. We also take the discriminator as an important part of multi-tasking training. We design a Coordinated Multitasking Training Discriminator (CMTD) as the discriminator to improve the performance of the generator and independently coordinate the weight of multi-task learning. Training of discriminator can reduce the difficulty of our loss function design and make us use simple loss function to complete the coordination of multi-task training. To coordinate the learning of these two tasks, we propose a Norm-Combined Hybrid loss function (NH loss). Our ELMGAN achieves non-Point-based counting that overcomes the limitation of traditional Point-based counting methods, which required datasets with detailed cell coordinates annotation. We use two cell microscope image datasets without the exact position annotation of each cell to validate the model. According to the experimental results, compared with other methods that just can complete only one task, our method can complete multiple tasks, and our method achieves excellent results in both counting and segmentation tasks. In short, our contributions are:

- We propose ELMGAN to segment and count cell images.
- We propose the NH loss.
- We propose a new upsampling method: FBU.
- Our ELMGAN overcomes the limitation of traditional Point-based counting methods.
- The result of ELMGAN is better than state-of-the-art.

2 Background

Recently, Deep Convolutional Neural Networks (DCNNs) have shown great success in both the natural and medical image domains [2, 3, 5, 8]. The fully convolutional networks (FCN) [3] method is one of the most widely used segmentation networks of convolutional neural networks in computer vision. FCN uses convolutional layers instead of fully connected layers and uses skip layers to retain advanced image features. Many medical image segmentation networks use FCN for medical image segmentation [12-15]. Schmitz et al. used multi-scale fully convolutional neural networks for histopathology image segmentation [15]; Vigueras et al. obtained that a convolutional network is superior to a sliding window network in cell segmentation through experimental comparison [16].

In previous studies, most networks used the encoder-decoder model [17]. Input data are gradually feature extracted through several successive downsampling units. After the sampling passes through the final bottleneck layer, highly abstract features are continuously re-stored to their original dimensions. However, the downsampling process can cause the feature to lose detailed information. In order to preserve the details of advanced features, in ResNets [18], researchers used residual layers to convey information about advanced features downward. Nevertheless, there is a lack of direct communication between each bottleneck layer of ResNets.

Based on FCN and ResNets, U-Net has shown excellent performance in medical imaging [2]. U-net uses skip layers to transfer the encoder features to the decoder and carries out multi-scale feature fusion, which performs well in medical image segmentation. Many image segmentation networks are inspired by U-Net [19-21].

In addition, image upsampling is an important part of the decoder. Researchers often use upsampling methods to restore the feature vector to the desired image. At present, there are two commonly used upsampling methods: interpolation upsampling methods and non-interpolation upsampling methods. The non-interpolation upsampling method usually uses dilated convolution to enlarge the image. However, the dilated convolution usually produces a Checkerboard pattern, blurring the generated image [22]. Although the image size is enlarged, the quality of the image is reduced. Expanding the image size in large steps can reduce the quality of the image to an unbearable level. The computer can only process and save discrete data, and the results of many image-processing are continuous. In order to obtain the values of discrete points, interpolation is necessary.

Therefore, researchers prefer to use interpolation methods to expand the images. At present, the most commonly used interpolation upsampling methods are the nearest upsampling method and the bilinear upsampling method. The logic of the

nearest interpolation method is very simple. It only compares the distance between the interpolated pixel and the original pixel. The interpolated pixels come from the surrounding original pixels without generating any new pixel values. However, some lines and edges in the image have an obvious sawtooth effect. The bilinear interpolation method requires cubic linear interpolation from two directions. Although its calculation process is more complex than the nearest upsampling method, the new pixel values can usually fit the surrounding pixels, making the image's colour transition softer. Although the current semantic segmentation models restore the image in size by applying various interpolation methods for the output of the last layer of the decoder, these methods have no learning parameters, which may lead to a suboptimal prediction result. [23].

On the other hand, most cell counting methods are divided into two categories: Point-based counting and non-Point-based counting. The Point-based methods are trained on datasets with detailed cell coordinates annotation and these methods could learn the characteristics of the annotated points. The two most common Point-based counting methods are the detection-based counting method and the density-map-based counting method. The detection-based counting method marks each cell by box and counts the boxes to get the number of cells [1, 24, 25]. Although the detection-based counting method has the highest accuracy, this method is severely limited by the complexity of images and requires a complex ground truth to frame each cell. The density-map-based counting method is to expand the annotation point of each cell into a spot with a similar cell size by Gaussian filter [1, 5]. In the image, these spots are similar to real cells, which reflect the location and density of cells well. Additionally, the total pixel value of the image is the number of cells. This method combines the two functions of counting and locating. However, like the detection-based counting method, the density-map-based counting method also needs detailed object coordinates. Non-Point-based counting does not require datasets to contain detailed object coordinates. The early non-Point-based counting method, the direct counting method, usually uses foreground feature and edge feature extraction to construct the mapping between features and the number of objects, such as standard background subtraction techniques to get foreground features [26]. But direct counting lacks the position information of the target.

Currently, few datasets with detailed cell coordinate labels are available specifically for cell counting because accurate labelling of high-density cells is a time-consuming and labour-intensive task. Most datasets still only have the classification, segmentation information, and the total number of cells. It may be a convenient and efficient method to count cells by predicting cell numbers and semantic segmentation to present cell location [6, 27, 28]. Although it is a relatively simple task to predict the numerical value with the help of neural networks, it may be difficult for the network to correctly learn effective knowledge if it wants to complete the generation of segmentation image and prediction of cell counting at the same time using traditional training methods. Because image segmentation and cell count prediction are two different jobs. Image segmentation is similar to the answer question, which has high requirements for the detailed restoration of the generated image. Another is closer to the True or False, which needs to judge the potential number of cells from the image features and give the possible prediction value.

The generation of segmented images from cell images can be regarded as image-to-image translation. At present, generative adversarial networks perform well in this area [11, 29]. Andreini et al. successfully used GAN in the segmentation of colony images [30]. In some papers, GAN is working for image-to-image translation. With the using of conditional constraints, GAN has shown impressive performance in style transfer and so on [31-33]. Therefore, we use conditional generative adversarial networks further to enhance the semantic segmentation capability of the model.

3 Methods

We propose a GAN-based efficient lightweight multi-scale-feature-fusion multi-task model (ELMGAN) that can segment and count cells simultaneously. In the whole GAN structure, We propose a lightweight and fast multi-scale-feature-fusion multi-task generator (LFMMG) and a Coordinated Multitasking Training Discriminator (CMTD). In order to optimize multi-tasking, we propose a Norm-Combined Hybrid loss function (NH loss).

3.1 Lightweight Fast Multi-scale-feature-fusion Multi-task Generator

We expect to build a lightweight and fast multi-scale-feature-fusion multi-task generator (LFMMG). Although the multi-scale feature fusion based on encoder-decoder segmentation models can well generate segmentation images with good low-level features, such architectures are complicated, and the model occupies a large memory, which does not meet our goal of lightweight. However, without the Encoder-Decoder architecture, the model does not perform as well as we need. Fortunately, we noted that the feature extraction networks like VGG [34] already have good feature extraction capability and have a lot of redundancy in network structure.

Therefore, in our encoder, we optimize the structure design of the model. As Figure 1 shows, we use six downsampling units,

which is only 36% the size of VGG16. This leaves a lot of memory for further addition of decoders. The convolutional layer with a step size of two is used in the downsampling unit, which reduces the feature map size while extracting features and avoids feature loss caused by using the pooling layer. Therefore, each of our Down-Sampling Units could reduce the size of the feature map by half, and our encoder could reduce the image by 64 times in total.

For upsampling, we use the Fold Beyond-nearest Upsampling (FBU) to expand the size of our feature map. After studying and comparing a variety of upsampling methods, we design FBU, which will be introduced in the next section. Compared with the traditional nearest interpolation method, FBU has a simpler calculation process, which speeds the calculation of the model. Besides, FBU not only enlarges the feature size, but also reduces the involvement of external errors from the interpolate upsampling layers. Moreover, compared with the traditional no learning upsampling methods such as nearest neighbour upsampling and bilinear interpolation upsampling, FBU has learnability. We add learnable parameters to make the FBU better enlarge the boundary changes in the image.

In the Decoder, we first magnified the image 36 times through five FBU blocks in total, and each FBU block can enlarge the image twice. Each FBU block contains an FBU layer, an Instance Normalization layer, and a Leaky ReLU activation layer. However, it is easy to blur the resulting image using the upsampling method alone, and the deep-seated feature map may lose many low-level features associated with image outlines and textures because it is highly abstract.

According to the successful experience of U-Net, we use multi-scale feature fusion to alleviate this problem. The feature map is fused with the feature map of the same size in the encoder, after it was enlarged by the FBU block.

Finally, we design another independent output layer to predict cell numbers. Based on the idea of FCN, we use the 1×1 convolutional layers. At the same time, we set up the global average pooling layer so that the network could accommodate different sizes of input images. This direct prediction method overcomes the limitation of Point-based counting on datasets.

In conclusion, we propose a lightweight and fast multi-scale-feature-fusion multi-task generator, which achieves non-Point-based counting, and gives the location of the target. Furthermore, our LFMMG reduces the number of parameters by 50% compared with U-Net, and its feature extraction encoder is 36% the size of VGG16. By reducing the number of convolutional layers and using FBU, our method significantly reduces the amount of computation and memory consumption.

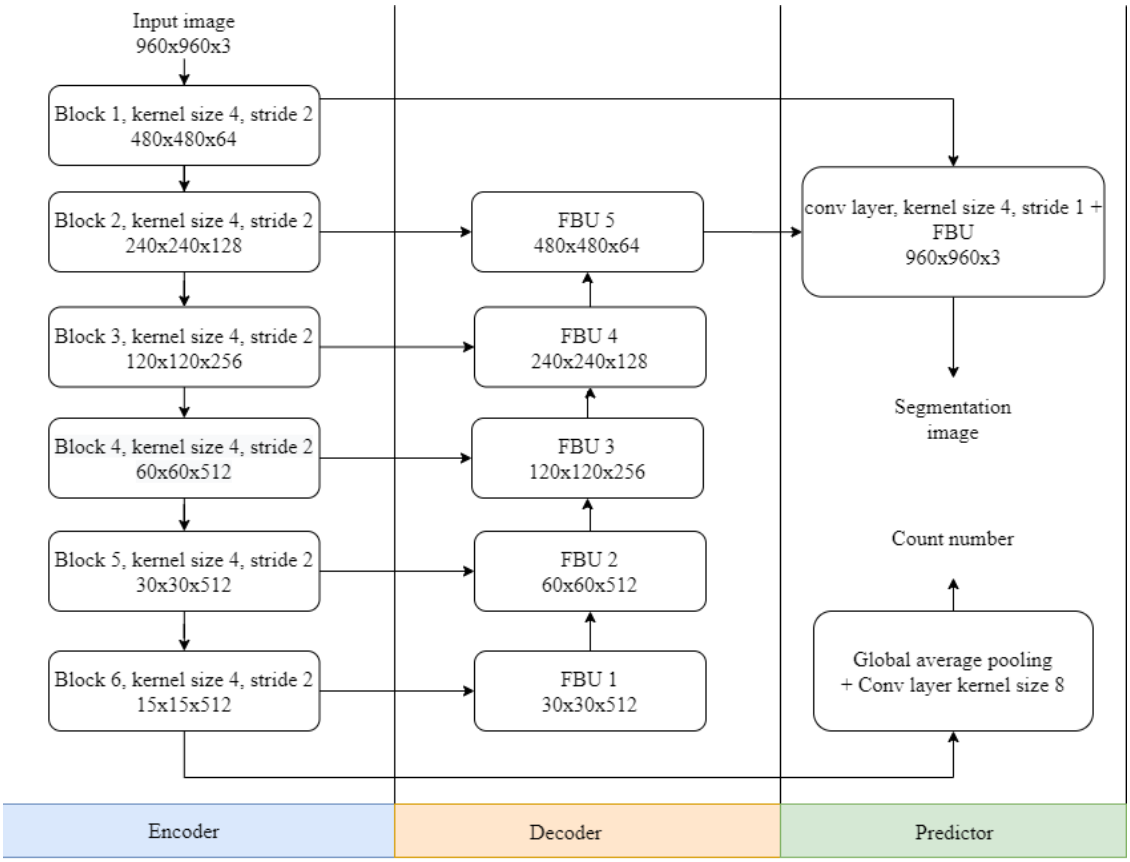

Figure 1 The structure of Lightweight Fast Multi-scale-feature-fusion Multi-task Generator (LFMMG); The LFMMG is the generator of ELMGAN, which is used for cell counting and cell segmentation.

3.2 Fold Beyond-nearest Upsampling Method

In the decoder, researchers often use upsampling methods to restore the feature vector to the desired image. However, some upsampling methods, like dilated convolutional layer, produce a Checkerboard pattern and blurry the generated images [22]. Although the image size is enlarged, the quality of the image is reduced. Expanding the image size in large steps reduces the quality of the image to an unbearable level. Therefore, researchers prefer to use interpolation methods to get larger images. Although the current semantic segmentation models restore the image in size by applying various interpolation methods in the last few layers of the decoder, these methods have no learning parameters, which may lead to a suboptimal prediction result. [23]. The classical nearest or bilinear interpolate upsampling method has a complex calculation process. Calculating a new pixel P_{x_n,y_n} requires the dot product of four scalar numbers W and the pixel values P of four vertices:

$$P_{x_n,y_n} = W_{x,y}P_{x,y} + W_{x+1,y}P_{x+1,y} + W_{x,y+1}P_{x,y+1} + W_{x-1,y}P_{x-1,y} + W_{x,y-1}P_{x,y-1}. \quad (1)$$

In order to reduce the loss introduced by this method, the out stride of interpolating upsampling layers should be set small when restoring the image pixel by pixel. Because of that, the interpolate upsampling method often has to be used multiple times, and it causes a significant increase in computational complexity and memory requirements. For restoring an image $I \in R^{w,h,c}$ to a new image $I_n \in R^{\hat{w},\hat{h},\hat{c}}$, through N Interpolate upsampling layers, the traditional interpolate upsampling method needs $N \times 4 \times \hat{w} \times \hat{h} \times \hat{c}$ times multiplication.

Although this kind of interpolation method avoids the chessboard effect caused by expansion convolution, these methods lack learnable parameters. Sometimes the result of interpolation may not be the best. The Beyond-bilinear Data-dependent Upsampling (BDU) [23] uses convolutional layers to restore the feature map. It uses the learnable parameters brought by the convolutional layer to calibrate the accuracy of interpolation. For preset out stride d, BDU divided the image into $d \times d$ small patches with the size of $\frac{w}{d} \times \frac{h}{d} \times c$. Then the small patch arranges to a vector $v \in [0,1]^{d \times d \times c}$. The v is compressed to $u \in R^{\hat{c}}$ by a convolutional layer, then u are mapped vertically to $\hat{v}$ by another convolutional layer as the new image I_n.

In formal language, that is:

$$u = Hv;\ \hat{v} = Vu, \tag{2}$$

where H and V are two convolutional kernels, respectively.

The total number of multiplications has been reduced to $w \times h \times c + \hat{w} \times \hat{h} \times \hat{c}$. This method successfully applies a learnable convolutional layer to the interpolation sampling method. And, this method avoids the problem of the Checkerboard pattern when sampling on dilated convolution.

In addition, the spatial encoder of SFCN effectively improves the continuity of the feature structure by folding and segmenting the image [35].

This method inspired us. We use the convolutional layer to generate multiple weighted new values of a pixel. We note that the 3×3 convolutional kernel can maximize the reference to the characteristics of the surrounding elements, to enhance the continuity of the image structure. While a 1×1 convolutional kernel also can do such tasks well and reduce the number of parameters greatly. Therefore, we use a 1×1 convolutional kernel for pixel value-added by default. Each new pixel is computed directly from the original pixels of the image. And, the image is scattered into pixels and then reorganized according to the sequence, which makes each point multi-pixel mixed and avoids the Checkerboard pattern.

This process is much like the repeated folding of dough when making pastry. Therefore, we call this method as Fold Beyond-nearest Upsampling (FBU). In our FBU, the channels of feature map x is expanded from c to $\hat{c} \times d^2$. Then the expanded feature map $\hat{x}$ is arranged into a one-dimensional vector v in the horizontal direction, and then we rebuild the vector v to a matrix $\hat{v} \in R^{w,hd,\hat{c}d}$. Similarly, the matrix $\hat{v}$ is arranged into a one-dimensional vector u in the vertical direction, and then we rebuild the vector u to a matrix $\hat{u} \in R^{wd,hd,\hat{c}}$.

In formal language, that is:

$$\hat{x} = wx + b;\ \hat{x} \to v_{w,h,c};\ \ \hat{v} = v_{h \leftrightarrow c};\ \hat{u} = u_{w \leftrightarrow c}, \tag{3}$$

where w is the weight of the convolutional layer. The process is shown in Figure 2.

Only one convolution calculation is carried out in the whole upsampling process, and the total number of multiplications has been reduced to $w \times h \times \hat{c}$. Compared with the traditional upsampling methods, the FBU reduces the number of multiplication calculations greatly. The calculation process is so simple that can speed up the calculation of the model. In addition, compared with the traditional upsampling methods (such as nearest upsampling and bilinear interpolation upsampling), FBU has learnability. In the traditional upsampling method, the new pixel is generated by calculating the original pixel. New pixels in FBU are generated through the convolutional layer. By training the learnable parameters of the convolutional layer in FBU, FBU can generate new pixel values more consistent with the feature change trend and better enlarge the boundary changes in the image.

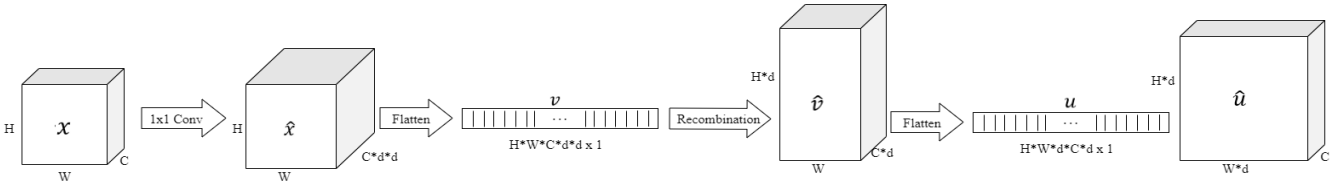

Figure 2. The process of Fold Beyond-nearest Upsampling

3.3 Coordinated Multitasking Training Discriminator

Because the FBU is trainable, some new changes might be introduced when the feature map is expanded. As the result, sometimes the generated image has little visual difference, but the feature map of the generated image may have different features. Therefore, although the overall loss is decreasing during the training, only one task might be trained. The conditional GAN [36] can specify the generated range by entering specific additional information into the discriminator. Similarly, we want the model to remember the features needed for counting when generating segmented images. So, we train the discriminator with the original image that contains the features needed for counting. Besides, the segmented image is usually similar to the original image in the contour structure. The original picture with complete information is regarded as structural attention.

Based on that thinking, we propose the Coordinated Multitasking Training Discriminator (CMTD) to coordinate the learning weights of the two tasks and improve the quality of the generated images. In CMTD, the original image x_i fuses the generated segmentation image G_i on the channel as the input of the discriminator to get the discriminant matrix D_i:

$$D_i = \mathrm{CMTD}\left(\begin{bmatrix} x_i \\ G_i \end{bmatrix} \in R^{w,h}\right), \tag{4}$$

where i represents the serial number of the image and h, w represent the height and width of the image, respectively.

We hope this can make the difference between the contour structure of the segmented image and the original image more prominent. By training the CMTD, the generator can predict cell structure profiles more accurately and maintain attention to the features required for counting.

As the left part of Figure 3 shows, the original image is fused into the generated synthetic image as a part of the input. The fused image is fed into four consecutive convolutional units.

Each convolutional unit contains a downsampling convolutional layer with stride two, a Leak-ReLU layer with a negative slope of 0.2, and an Instance Normalization Layer [37]. The image features were condensed 16 times in the process. Finally, we get a feature map that can reflect the difference in image structure. To amplify these structural differences, we choose L2 loss as the loss function of the discriminator. This discriminator identifies pixel errors between the two, making the synthetic image structurally closer to the original.

In Section 4.3, we verified that CMTD could effectively improve the performance of the multi-task generator.

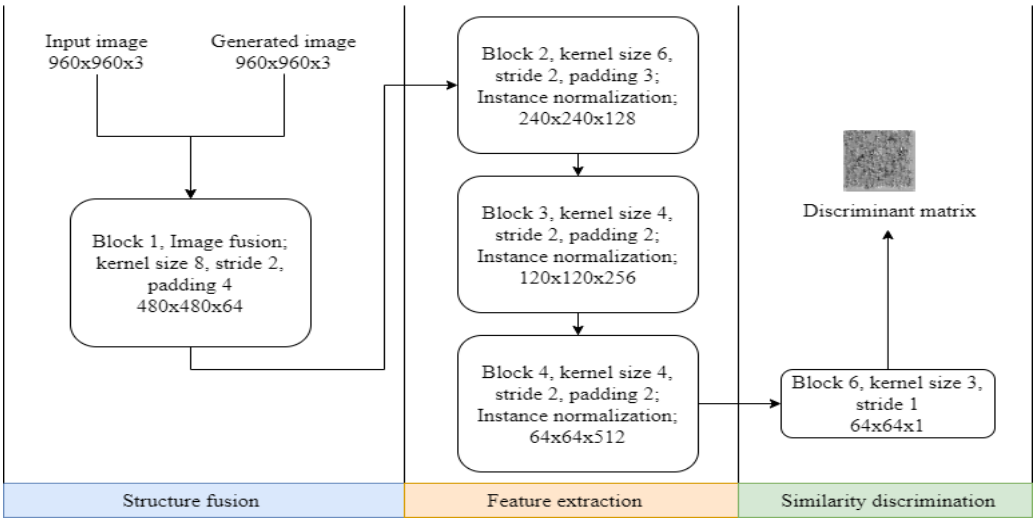

Figure 3. The structure of Coordinated Multitasking Training Discriminator (CMTD); The CMTD is the discriminator of ELMGAN, which is used to assist in training LFMMG.

3.4 ELMGAN Model with NH Loss

In order to enhance the performance of our model and strengthen the connection between counting and segmentation, we propose the Efficient Lightweight Multi-scale-feature-fusion Multi-task GAN model (ELMGAN) to train our LFMMG and CMTD. As shown in Figure 4, in the ELMGAN, the LFMMG generates the segmentation images and cell counts, and the CMTD estimates the probability that the input image comes from the real data rather than a generated sample by constructing a Markov random field.

For ELMGAN training, LFMMG and CMTD are trained together under real data x, and ground truth y. The best model is obtained through the minimax two-player game: Fixed the CMTD and adjusted the parameters of LFMMG to minimize the expectation of $\log[1-\mathrm{CMTD}(x,\mathrm{LFMMG}(x))]$ and fixed the LFMMG and adjusted the parameters of CMTD to maximize the expectation $\log \mathrm{CMTD}(x,y)$. In formal language, that is:

$$\min_{LFMG}\max_{SFD}\mathrm{ELMGAN}(\mathrm{LFMMG},\mathrm{CMTD}) = E_{x,y}[\log \mathrm{CMTD}(x,y)] + E_{x,y}\log[1-\mathrm{CMTD}(x,\mathrm{LFMMG}(x))]. \quad (5)$$

We use our Norm-Combined Hybrid loss (NH loss) for Generators to learn multi-tasks. Using loss function matching tasks can effectively improve the learning efficiency of the model. L2 loss is the most commonly used loss function in counting tasks, so we use L2 loss to evaluate the counting task. For the generation of segmentation images, we pay more attention to the structural similarity of images. Because L1 loss can capture the low frequency correctly, the L1 loss is employed to adjust the structural error of the image at the pixel level. In addition, in GANs, the original loss of the generator is calculated by the discriminator's loss, so our NH loss also includes a part of discriminator loss. In a word, our NH loss includes three parts: counting loss, pixel loss, and a part of discriminator loss.

For counting loss, for R images, the i-th image with c_{gt} cells are predicted by the generator to obtain c_{pred} cells, we use

$$L_{count} = \frac{1}{R}\sum_{i=1}^{R}\left(c_{gt}^{i} - c_{pred}^{i}\right)^2 \quad (6)$$

as the loss function of the counting task.

The pixel loss is calculated by the generated image G_i and segmentation ground truth S_{gt_i}:

$$L_{pixel} = \frac{1}{R}\sum_{i=1}^{R}\left|(S_{gt} - G_i)\right|. \quad (7)$$

Considering the discriminator's influence on the generator's task attention allocation, when training the generator, we need to calculate the discriminant loss of the generator by assuming that the generated segmentation image is completely reliable.

In the last part of NH loss, we use the generated synthetic image G_i and the original image to generate the discriminant matrix D_i through the discriminator and calculate an L2 loss by using the discriminant matrix D_i and the valid matrix $valid$, which was full by one:

$$Loss\_D_{real} = \frac{1}{R}\sum_{i=1}^{R}(valid_i - D_i)^2. \quad (8)$$

Then Finally, these three losses are combined to build our NH loss:

$$NH\ Loss\ =\ Loss\_D_{real}\ +\ aL_{pixel} + bL_{count}, \quad (9)$$

where a is the weight of pixel loss, and b is the weight of counting loss.

In LFMMG, the number of cells is predicted directly through the features extracted by the decoder. Loss count is employed to learn cell number prediction, which can directly affect the preference of LFMMG in feature extraction. The synthetic image is constructed based on these extracted features. Like the butterfly effect, changes in the extracted features can create a huge disturbance to the prediction of the synthetic image. Therefore, we recommend using a larger a and smaller b to balance the training of the two predictors. In Section 4.4, we experimentally verified this inference. So here, the preset value of a is 100 and that of b is 0.1. This preset value certainly does not represent an optimal parameter. Based on the experiment results, it is recommended to fine-tune the value of a in ten units depending on the complexity of the data. However, turning up the value of b need to be as precise as possible.

When training the discriminator, in the first step, the Discriminant matrix D_i calculates an L2 loss with the valid matrix, which is the same with $Loss\_D_{real}$.

Then the discriminant matrix is calculated with the fake matrix $fake$, which was full by zero:

$$Loss\_D_{fake}\ =\ \frac{1}{R}\sum_{i=1}^{R}(fake_i - D_i)^2. \quad (10)$$

In the end, the total training loss of the discriminator is the average value of these two losses:

$$Loss\_D\ =\ avg(Loss\_D_{real} + Loss\_D_{fake}). \quad (11)$$

Algorithm 1 ELMGAN training process

```
Input   Data: image, segmentation ground truth, counting ground truth
1.      function Main: {
2.          for i in epoch: {
3.              function training: {
4.                  for j in range(data): {
5.                      Construct a valid matrix with all pixel values of 1 in the same size as
                        the input image;
6.                      Construct a fake matrix with all pixel values of 0 in the same size as
                        the input image;
7.                      input images are input into LFMMG to predict the cell's number and
                        segmentation;
8.                      Calculate the loss of the generator {
9.                          Calculate the structure loss between the input image and the
                            generated segmentation image;
10.                         Calculate the counting loss between the counting ground truth
                            and predict the cell's number;
11.                         Calculate the generator Hybrid loss based on structure loss and
                            counting loss;
12.                         }
13.                     Obtain the discrimination matrix by the generated segmentation
                        images, and the input images are input into the CMTD;
14.                     Calculate the loss of the discriminator {
15.                         Calculate the generator discrimination loss between the
                            discrimination matrix and the valid matrix;
16.                         Calculate the Norm-combined Hybrid loss based on
                            discrimination loss and generator Hybrid loss;
17.                         }
18.                     Norm-combined Hybrid loss backward;
19.                     Train CMTD {
20.                         Obtain the valid discrimination matrix by segmentation ground
                            truth and input image;
21.                         Calculate the valid CMTD loss of discrimination matrix and valid
                            matrix and fake matrix, respectively;
22.                         Obtain the valid discrimination matrix by the generated
                            segmentation images and input images;
23.                         Calculate the fake CMTD loss of discrimination matrix and valid
                            matrix and fake matrix, respectively;
24.                         Calculate the total CMTD loss by valid CMTD loss and fake CMTD
                            loss;
25.                 }
26.             Evaluate every few epochs: {
27.                 Evaluate LFMMG;
28.                 Print evaluation results;
29.                 }
30.             Save the model;
31.         }
32.     }
```

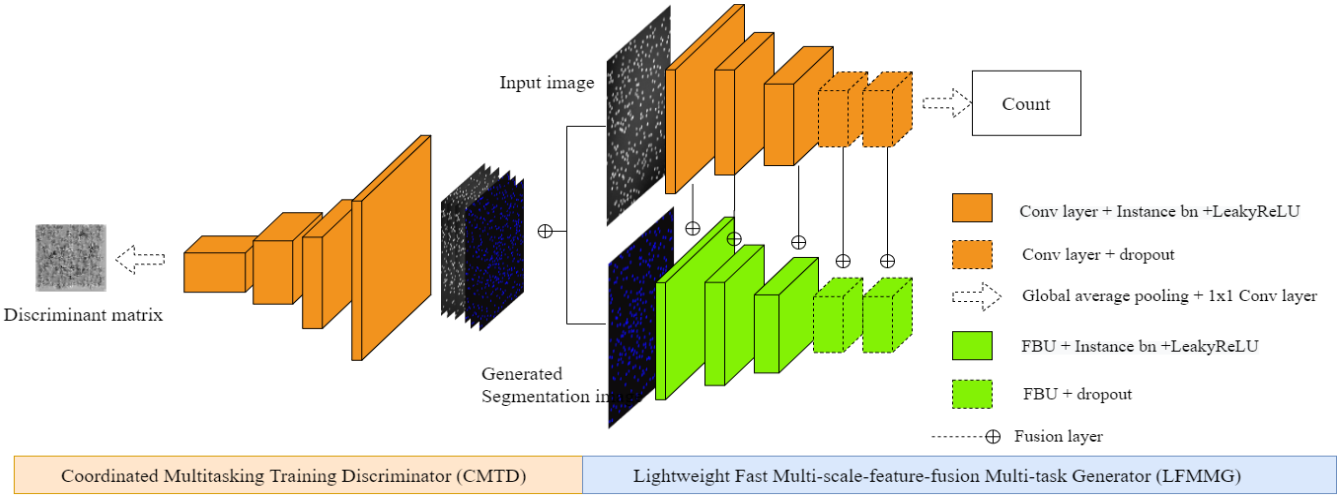

Figure 4. The structure of ELMGAN; ELMGAN is used for training our LFMMG and CMTD so that we can get the best performance cell number and location prediction generator.

4 Experiments and Results

We performed segmentation and counting experiments on cell images with different degrees of overlap. The images were divided into five different levels of overlap. We use PSNR and SSIM to evaluate the segmentation of the generator. Besides, we use MCE and RMSE to evaluate the counting ability.

4.1 Datasets and Experimental Environment

We use the datasets from the Broad Bioimage Benchmark Collection (BBBC) [38]. The image samples are shown in Figure 5. The BBBC04 dataset has five subsets of 20 images per group provided. Each image contains 300 objects, but objects overlap and aggregate with different probabilities in five subsets. Images were generated using SIMCEP simulating platform for fluorescent cell population images [39, 40]. The dataset contains images with a size of 950 by 950. To facilitate transformation in the neural network, we added a border of 0-pixel value to the image and processed the image to a size of 960×960. According to the records of BBBC on the official website of the dataset, we are the first team to use this data for segmentation counting multi-task deep convolutional neural network learning.

Another dataset we used is the BBBC05 dataset. The dataset is a collection of simulated microscope images of 9600 stained cells. These images were generated using the SIMCEP simulation platform. The clustering probability of cells in this data set is 25%, and the cell area matches the average cell area of human U2OS cells [33, 34]. The focal blur is simulated by applying a variable Gaussian filter to the image. Each image is encoded in 8-bit TIFF format and has a size of 696×520. The model has been trained with 1,200 labelled images.

In our experiments, the normalization process of subtracting the mean and dividing by the variance is carried out. Training images are cropped to a 4-point scale and placed in the same batch. Because no default testing subset segmentation is available, the model was evaluated by the use of 5-fold cross-validation on both datasets.

Our experiments are based on a Python environment. Batch size is set to 1, Adam is the optimizer, and 50 epochs are learned at a fixed learning rate of 2e-4. For training the generator, the beta of the optimizer is set to 0.5. For training the discriminator, the beta of the optimizer is set to 0.999. We use an RTX 2070 graphics card for the experiments.

For the convenience of calculation, for images whose size does not meet the multiple of 128, we have zero padding on the left and bottom to make the image size a multiple of 128.

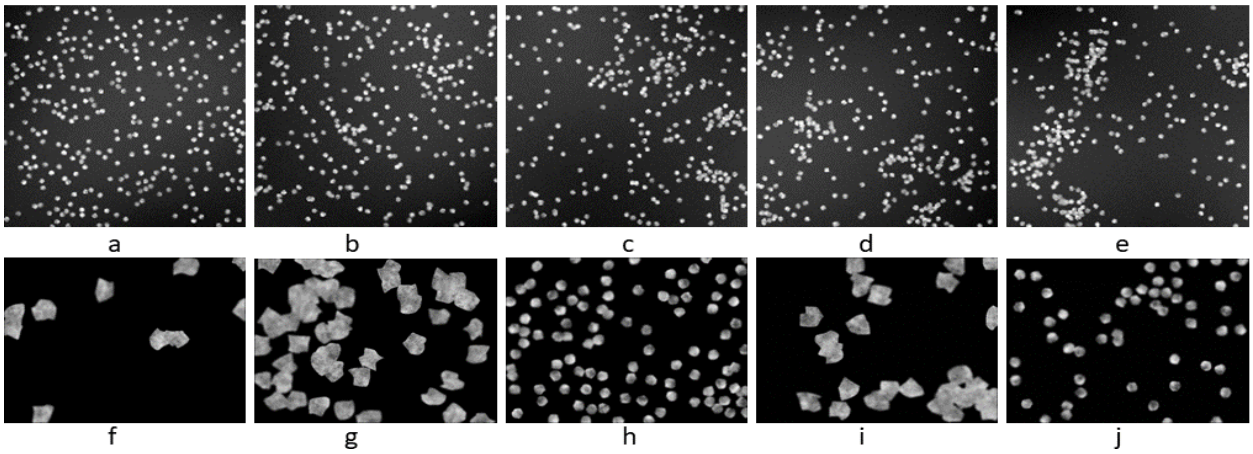

Figure 5. The first row shows the cell images from the BBBC04 dataset; a. b, c, d, and e represent the images of cell overlap from 0%, 15%, 30%, 45%, and 60%, respectively. In the second row, f, g, h, i, and j are image samples from the BBBC05 dataset

4.2 Evaluation methods

Our model performs two tasks simultaneously: one is to segment the cell image, and the other is to count the number of cells. For cell segmentation, the Peak signal-to-noise ratio (PSNR) and Structure Similarity Index Measure (SSIM) are used as the evaluation indexes between the generated segmentation map and the truth map.

PSNR is an engineering term representing the ratio of the maximum possible signal power to the destructive noise power affecting its accuracy. Since many signals have a wide dynamic range, the peak signal-to-noise ratio (PSNR) is usually expressed in decibels. Researchers usually use PSNR to evaluate the quality of an image compared with the original image. The higher the PSNR, the smaller the distortion.

SSIM can compare the structural distortion of the new image and reference images' structural distortion more instantly. Therefore, we also use SSIM to evaluate the quality of the

generated image. SSIM is generally between 0 and 1. The larger its value, the better the image quality.

We use two indexes to evaluate the prediction results to predict cell numbers. The mean counting error (MCE), which measures instance counting accuracy for R images:

$$MCE = \frac{1}{R}\sum_{i=1}^{R}\left|C_{gt}^{i} - C_{pred}^{i}\right|, \quad (12)$$

where C_{gt}^{i} represents the ground truth number of cells in the i-th picture, and C_{pred}^{i} is the cell number predicted by the generator. MCE can numerically represent the average number of false identifications per image. A smaller value of MCE means a smaller amount of counting error per image on average

The root mean squared error (RMSE) can reflect the deviation value of the prediction error of each instance:

$$RMSE = \sqrt{\frac{1}{R}\sum_{i=1}^{R}\left(C_{gt}^{i} - C_{pred}^{i}\right)^{2}}, \quad (13)$$

where R is the number of responses, C_{gt}^{i}represents the ground truth number of cells in the i-th picture, and C_{pred}^{i} is the cell number predicted by the generator. RMSE is a commonly used model performance evaluation metric in the field of object counting. A smaller value of RMSE means better accuracy of the model prediction.

In addition, we use floating-point operations (FLOPs) to measure the complexity of the model. For a convolutional layer, FLOPs are calculated as:

$$FLOPs = 2HW(C_{in}K^{2} + 1)C_{out}, \quad (14)$$

where H and W represent the height and width of the output feature map, respectively; K is the kernel size of the convolutional layer, C_{in} is the channel of the input feature map, and C_{out} is the channel of the output feature map.

4.3 Validation of LFMMG and CMTD

Based on the thinking of control variates, we first constructed a generator with VGG16-bn as the backbone and combined it with our CMTD to form a VGG+CMTD GAN. At the same time, we constructed an LFMMG GAN model with LFMMG and a general discriminator without structure fusion. We experimentally compared our ELMGAN with these two GAN models on the BBBC04 dataset. The experiment shows that our LFMMG and CMTD-based ELMGAN effectively could improve the accuracy of the image segmentation and the count prediction.

Table 3 shows that ELMGAN, based on LFMMG and CMTD, achieves the best counting accuracy and image segmentation accuracy. Compared with the generator without multi-scale feature fusion based on VGG16, our LFMMG can significantly improve the accuracy of cell counting and image generation. Additionally, Figure 6 shows that images generated by LFMMG are smoother at the edges than images generated by VGG16 without multi-scale feature fusion. Thus, using large stride upsampling to restore the image might lose more image details than using multi-scale feature fusion. Besides, from the comparison of count results, compared with the convolutional layer with a larger stride, the pooling layer for downsampling indeed causes information loss and change.

On the other hand, in Table 3, according to the SSIM results of LFMMG+CMTD and LFMMG, the CMTD plays a good role in optimizing the structural accuracy of the generated images. Moreover, CMTD is helpful to feature extraction of the generator. The addition of the CMTD not only gives the model a better result in the segmentation of images, but also gives a surprising performance in predicting cell numbers.

In addition, from Figure 7, our ELMGAN, based on LFMMG and CMTD, shows a great segmentation performance on every overlap level. The values of MCE and RMSE show that our method has few counting errors on the BBBC04 dataset. The values of PSNR and SSIM show that the generated segmentation images can be almost identical to the ground truth. Figure 8 shows that our method also handles the details very well: Narrow gaps and voids were successfully segmented.

Furthermore, we compare the difference in model complexity between LFMMG with feature fusion and the VGG-based model without feature fusion based on images with the size of 256x256x3.

The value of FLOPs shows the total number of floating-point calculations required for running the statistical model. The speed at which the generator processes a single picture is calculated by recording the total time of processing 100 pictures. The memory usage shows the requirement of inference memory. The parameters show the total parameters of this model. FLOPs and speed can represent the time complexity of the model and the operation speed of the model. The memory usage and the parameters represent the spatial complexity of the model.

From Table 1, the feature fusion method complicates the model and reduces the processing speed. However, although the decoder-encoder-based feature fusion method increases the complexity of the model, this method could improve the accuracy of cell contour segmentation and improve the performance of the model greatly.

Table 1. The Network Complexity Analysis

Feature fusion	FLOPs	Speed	Memory usage	Parameters
Yes	5.12GFLOPs	89ms/p	56.34MB	15.83MB
No	2.14GFLOPs	57ms/p	32.32MB	10.72MB

4.4 Validation of Proposed NH Loss

The model using our NH loss and the models using other loss functions (L1 loss, L2 loss, and Hybrid loss) are compared based on BBBC04 dataset. Hybrid loss (H loss) is our NH loss

without pixel loss. As shown in Table 4, our NH loss-based method achieves the best counting accuracy and image segmentation accuracy.

The model trained with L1 loss gets a better image accuracy. The L2 loss training model gets a better accuracy in predicting the number of cells. Therefore, we should use both L1 loss and L2 loss in the training process. L2 loss is suitable for counting tasks, and L1 loss is suitable for segmentation tasks. H loss integrates the characteristics of both L1 loss and L2 loss, which could balance the learning of the two tasks.

However, if no pixel loss is calculated and only using the H loss to train the model, the segmented image generated by the model will be significantly worse than that generated by the model trained with our NH loss function. Moreover, the experiment results deteriorate as the level of cell overlap increases. Adding pixel loss can effectively control this trend. Besides, when training the model without combined loss, the counting ability decreases as the training time increases.

In NH loss, we use two parameters to adjust the learning concerns of the model for the counting task and the segmentation task. In Table 2, As the ratio of the pixel loss weight a to the counting loss weight b increases, the quality of the generated image has been improved. However, the increase in the counting loss weight has no significant effect on the change in count results. Therefore, the default value of a is 100 and b is 0.1. This default value does not represent an optimal parameter. We can enlarge the value of b moderately when the counting task is more difficult, and we can enlarge the value of a moderately when the image segmentation is more complicated.

Table 2. Comparison of different loss weights in NH loss on the BBBC05 dataset. a is the weight of pixel loss, and b is the weight of counting loss.

Pixel loss weight (a)	Counting loss weight (b)	MCE	RMSE	PSNR	SSIM
1	10	1.01	1.3	63.07	0.465
1	1	0.816	1.024	62.58	0.524
10	1	0.966	1.302	65.41	0.569
100	1	0.714	0.929	65.44	0.650
100 (Default)	0.1 (Default)	0.803	1.023	64.33	0.983

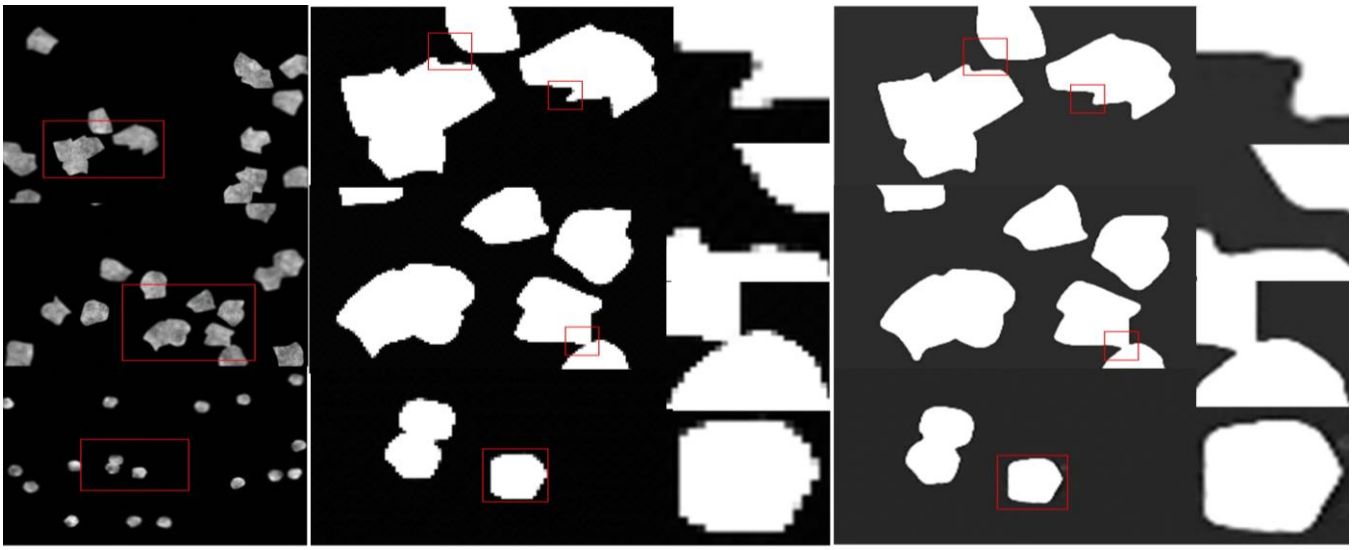

Figure 6. Segmentation comparison with (LFMMG) and without (VGG) feature fusion. The image is magnified in the red box.

Table 3. Comparison of different network structures. Red means the best result.

	Models	Overlap levels	0%	15%	30%	45%	60%	Avg.
Counting	LFMMG + CMTD	MCE	0.27	0.18	0.33	0.29	0.64	0.34
		RMSE	0.34	0.25	0.4	0.33	0.7	0.40
	VGG + CMTD	MCE	0.48	0.7	0.66	0.84	0.79	0.694
		RMSE	0.59	0.96	0.87	1.06	0.86	0.869
	LFMMG GAN	MCE	0.43	0.33	0.27	0.63	0.83	0.50
		RMSE	0.52	0.43	0.31	0.87	0.92	0.61

Segmentation	LFMMG + CMTD	PSNR	64.9	66.6	65.1	66.5	66.9	65.9
		SSIM	0.98	0.96	0.95	0.96	0.95	0.96
	VGG + CMTD	PSNR	62.8	63.1	63.1	63.2	63.4	63.1
		SSIM	0.92	0.93	0.93	0.94	0.94	0.937
	LFMMG GAN	PSNR	62.96	64.4	63.15	63.39	63.39	63.45
		SSIM	0.95	0.95	0.96	0.94	0.96	0.95

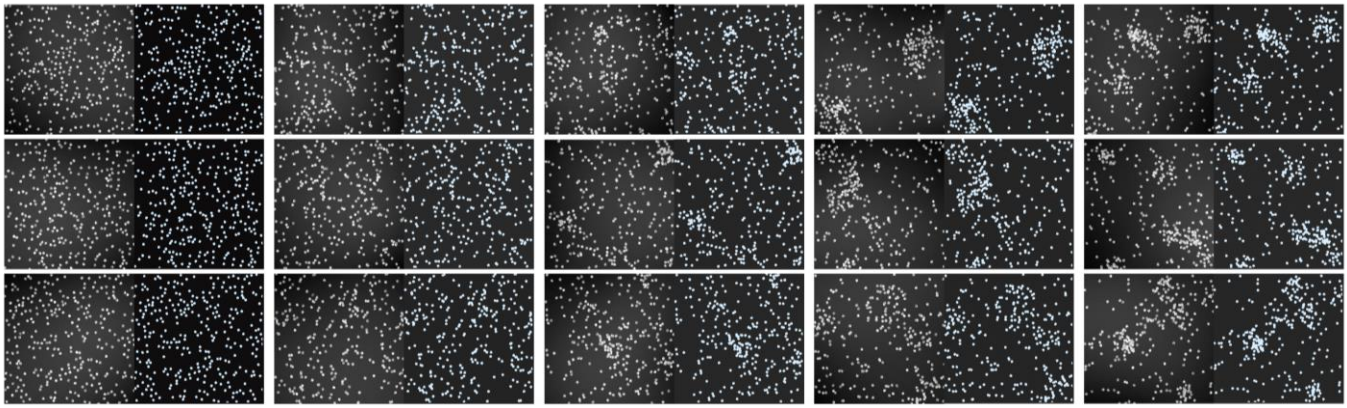

Figure 7. The segmentation results of our ELMGAN were on the BBBC04 dataset. White dots are the cells in input images, and the blue dots are the sells in the segmentation images; From the left to right are the images of cell overlap from 0%, 15%, 30%, 45%, and 60%, respectively.

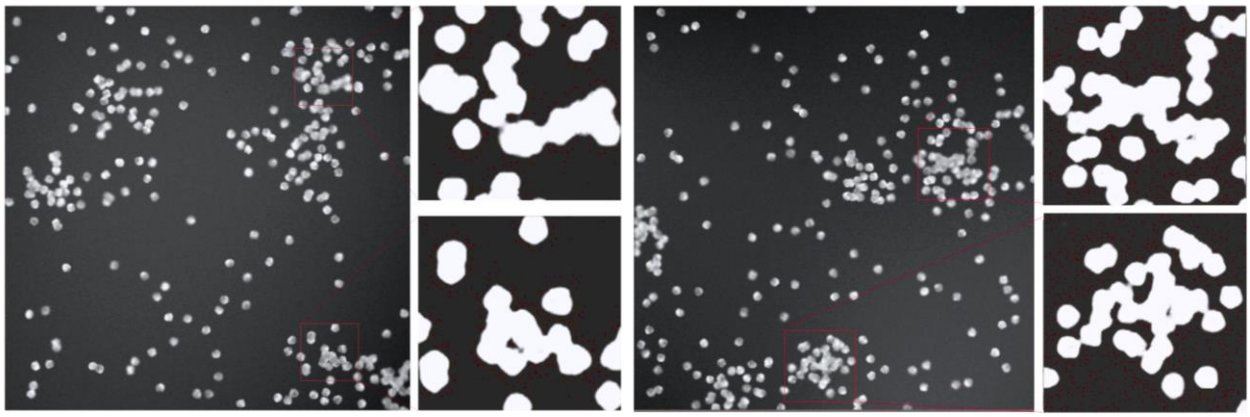

Figure 8. The detailed segmentation results of our ELMGAN on the BBBC04 dataset

Table 4. Comparison of different loss functions. Red means the best result.

	Models	Overlap levels	0%	15%	30%	45%	60%	Avg.
Counting	NH loss (ours)	MCE	0.27	0.18	0.33	0.29	0.64	0.342
		RMSE	0.34	0.25	0.4	0.33	0.7	0.404
	L1 loss	MCE	0.507	0.457	0.308	0.54	0.359	0.434
		RMSE	0.581	0.528	0.41	0.671	0.46	0.53
	L2 loss	MCE	0.505	0.412	0.23	0.315	0.347	0.361
		RMSE	0.694	0.477	0.3	0.401	0.461	0.466
	H loss	MCE	0.542	0.542	0.348	0.481	0.565	0.495
		RMSE	0.573	0.598	0.373	0.491	0.651	0.537
Segmentation	NH loss (ours)	PSNR	64.96	66.55	65.07	66.47	66.9	65.99
		SSIM	0.979	0.963	0.949	0.958	0.954	0.961
	L1 loss	PSNR	63.02	63.33	63.22	63.54	64.05	63.43
		SSIM	0.956	0.926	0.975	0.962	0.955	0.954

L2 loss	PSNR	62.85	64.09	63.14	63.22	63.39	63.33
	SSIM	0.859	0.89	0.802	0.782	0.785	0.823
H loss	PSNR	63.2	63.47	63.27	63.47	66.12	63.90
	SSIM	0.725	0.218	0.27	0.306	0.592	0.422

4.5 Comparison with FBU and other Upsampling Methods

To verify the performance of FBU in the decoder of the model of LFMMG, we try FBU, the nearest interpolation sampling method, and the bilinear interpolation sampling method to expand the size of the image, respectively. Experiments are performed on the BBBC04 and the BBBC05 datasets.

From Table 5, we can see that our FBU is much better than the other two traditional interpolation sampling methods in the upsampling performance. From the results of SSIM, using the FBU method to get the segmented image can restore the edge structure of cells to the maximum extent. Additionally, as predicted by mathematical analysis, the FBU method is much faster in processing a single picture than the traditional interpolation sampling method because it reduces the amount of computation. The average speed at which the generator processes a single picture is calculated by recording the total time of processing 100 pictures. Furthermore, the unit of the image processing speed is the average number of milliseconds per picture. From Table 5, with the same model structure, simply changing the traditional sampling method to FBU increases the processing speed by about 30%. The FBU improves the computing speed of the network, dramatically enhances the efficiency of the network, and improves the network experience.

Table 5. Comparison of different upsampling methods on two datasets. Red means the best result.

Dataset	Models	PSNR↑	SSIM↑	Speed
BBBC04	FBU (ours)	65.99	0.960	1.9ms
	Nearest	63.278	0.824	2.9ms
	Bilinear	63.75	0.831	2.9ms
BBBC05	FBU (ours)	64.33	0.983	1.9ms
	Nearest	75.442	0.83	2.9ms
	Bilinear	76.64	0.876	2.9ms

4.6 Comparison with other Methods

We also compare our method with other studies on the BBBC04 and the BBBC05 datasets. The results are shown in Table 6. Our method has the best performance on both segmentation and counting tasks. Our method has the greatest advantage over other methods in that it can simultaneously perform the cell count and segment tasks.

In the counting task, our method has a higher accuracy for predicting cell numbers than classical CNN models, such as VGG16. Some classical Point-based counting models cannot directly use these datasets without the coordinate information of each cell because they cannot generate the ground truth like density map and bounding box coordinates.

Our model outperforms classical segmentation models, such as the U-Net, in the segmentation task. U-Net segmentation results are better than VGG-GAN, which illustrates the importance of multi-scale feature fusion for image segmentation. GAN and cGAN may not complete convergence due to slow training and the epoch limitation.

Table 6. The experimental results of the models on the BBBC04 and the BBBC05. '↑' shows that the larger the indicator, the better; '↓' shows that the smaller the indicator, the better. '/shows that the test was not carried out in the original paper; '×' shows that the model cannot do this task directly; Red means the best result, and Blue means the second-best result.

Dataset	Models	MCE↓	RMSE↓	PSNR↑	SSIM↑
BBBC04	ELMGAN-NH loss (ours)	0.342	0.404	65.99	0.961
	ELMGAN-H loss (ours)	0.495	0.537	63.90	0.422
	ELMGAN-L1 loss (ours)	0.434	0.53	63.43	0.954
	ELMGAN-L2 loss (ours)	0.361	0.466	63.33	0.823
	LFMMG-NH loss (ours)	0.501	0.612	63.46	0.955
	VGG+CMTD	0.694	0.869	63.14	0.937
	U-Net [2]	×	×	62.312	0.951
	cGAN [36]	×	×	65.53	0.181
	GAN [10]	×	×	65.08	0.046
	FPNet [8]	/	/	/	/
	VGG16 [34]	2.192	2.748	×	×
BBBC05	ELMGAN-NH loss (ours)	0.803	1.023	64.33	0.983
	ELMGAN-H loss (ours)	0.867	1.083	62.82	0.393

ELMGAN-L1 loss (ours)	0.928	1.157	63.09	0.988
ELMGAN-L2 loss (ours)	0.868	1.08	59.44	0.878
LFMMG-NH loss (ours)	0.828	1.038	74.56	0.851
VGG+CMTD	2.35	3.47	65.17	0.8629
U-Net [2]	×	×	61.0	0.874
cGAN [36]	×	×	56.6	0.194
GAN [10]	×	×	56.3	0.034
FPNet [8]	2.4	3.34	/	/
VGG16 [34]	2.87	3.55	×	×

We also record the file size and average processing speed of each image of each method. We use the 'torch.save' function to save the main parameters of the generator models and compare their file sizes. The unit of file size is the Megabyte. In addition, we record how fast the generator was processing the pictures. The image processing speed is statistical from the time that the image is sent to the generator until the result is recorded. The average speed is calculated by recording the total time of processing 100 pictures. And the unit of the image processing speed is milliseconds per picture.

In Table 7, our ELMGAN is a more lightweight model. MemR+W shows the sum of the size read from the memory and the size written into the memory while the network is running. At run time, our method consumes the least memory. And it only takes up half the disk space of the U-Net. The computational speed is extremely fast because of the simplification of the model. Our ELMGAN processing is ten times faster than GAN and twice the speed of the classical image processing models VGG16 and U-Net. The experimental results verify that our method achieves an excellent network light-weighting result and successfully improves the processing efficiency of the model.

Table 7. Comparison of file size and process speed of models; Red is the best result.

Models	File Size	Parameters	Speed	MemR+W
ELMGAN (ours)	69.2MB	21.04M	1.9ms/p	138.89MB
cGAN [36]	903MB	252.48M	3.6ms/p	965.66M
GAN [10]	894MB	227.383M	12.6ms/p	868.91M
U-Net [2]	131MB	32.96M	2.9ms/p	1.11GB
VGG16 [34]	93MB	138M	2.2ms/p	499.09MB

5 Conclusion

Cell segmentation and counting are important tasks. We propose a GAN-based efficient lightweight multi-scale-feature-fusion multi-task model (ELMGAN) for cell segmentation and counting simultaneously. Point-based counting methods require datasets with detailed annotations of cell locations. This may lead to scarce datasets, tedious labelling processes and complex models. Our method achieves non-Point-based counting that overcomes the limitation of traditional Point-based counting methods and can make wider use of existing datasets. Our NH loss function coordinates the training attention of the counting and segmentation tasks and helps ELMGAN better to train the multi-task model. Our CMTD makes ELMGAN to achieve higher segmentation accuracy and prediction ability. Experiments show that our method can generate high-quality segmentation images with excellent counting accuracy. In addition, the experimental results verify that our method achieves an excellent network lightweight result and successfully improves the processing efficiency of the model.

Cell counting and segmentation have a strong basic demonstration significance in the whole field of object counting and segmentation. This non-Point-based counting method could be extended to the whole field of object counting, just like its predecessor density map-based counting method. Additionally, for a wider range of unlabeled datasets, models based on weakly-supervised or self-supervised learning may further strengthen the significance of non-Point-based counting in object counting. In the future, we plan to continue to expand the application of this counting method in other object counting tasks.

ACKNOWLEDGMENTS

The paper was partially supported by: Medical Research Council Confidence in Concept Award, UK (MC_PC_17171); Royal Society International Exchanges Cost Share Award, UK (RP202G0230); British Heart Foundation Accelerator Award, UK (AA/18/3/34220); Hope Foundation for Cancer Research, UK (RM60G0680); Global Challenges Research Fund (GCRF), UK (P202PF11); Sino-UK Industrial Fund, UK (RP202G0289); LIAS Pioneering Partnerships award, UK (P202ED10); Data Science Enhancement Fund, UK (P202RE237).

REFERENCES

[1] He, S., Minn, K. T., Solnica-Krezel, L., Anastasio, M. A. and Li, H. Deeply-supervised density regression for automatic cell

counting in microscopy images. *Medical Image Analysis*, 68 (2021/02/01/ 2021), 101892.
[2] Ronneberger, O., Fischer, P. and Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints* (2015), arXiv:1505.04597.
[3] Shelhamer, E., Long, J. and Darrell, T. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 4 (2017), 640-651.
[4] Yu, F. and Koltun, V. J. C. Multi-Scale Context Aggregation by Dilated Convolutions, abs/1511.07122 (2016).
[5] Xie, W., Noble, J. A. and Zisserman, A. Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6, 3 (2018/05/04 2018), 283-292.
[6] Graham, S., Vu, Q. D., Raza, S. E. A., Azam, A., Tsang, Y. W., Kwak, J. T. and Rajpoot, N. Hover-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, 58 (2019/12/01/ 2019), 101563.
[7] Cireşan, D. C., Giusti, A., Gambardella, L. M. and Schmidhuber, J. *Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks*. Springer Berlin Heidelberg, City, 2013.
[8] Hernández, C. X., Sultan, M. M. and Pande, V. S. Using Deep Learning for Segmentation and Counting within Microscopy Data. *arXiv e-prints* (2018), arXiv:1802.10548.
[9] Liu, F. and Yang, L. *A Novel Cell Detection Method Using Deep Convolutional Neural Network and Maximum-Weight Independent Set*. Springer International Publishing, City, 2017.
[10] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative Adversarial Networks. *arXiv e-prints* (2014), arXiv:1406.2661.
[11] Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A. A. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv e-prints* (2016), arXiv:1611.07004.
[12] Wang, C., Anisuzzaman, D. M., Williamson, V., Dhar, M. K., Rostami, B., Niezgoda, J., Gopalakrishnan, S. and Yu, Z. Fully automatic wound segmentation with deep convolutional neural networks. *Sci Rep*, 10, 1 (2020), 21897-21897.
[13] Zhang, J., Liu, M., Wang, L., Chen, S., Yuan, P., Li, J., Shen, S. G.-F., Tang, Z., Chen, K.-C., Xia, J. J. and Shen, D. Context-guided fully convolutional networks for joint craniomaxillofacial bone segmentation and landmark digitization. *Medical image analysis*, 60 (2020), 101621-101621.
[14] Gerard, S. E., Herrmann, J., Kaczka, D. W., Musch, G., Fernandez-Bustamante, A. and Reinhardt, J. M. Multi-resolution convolutional neural networks for fully automated segmentation of acutely injured lungs in multiple species. *Medical image analysis*, 60 (2020), 101592-101592.
[15] Schmitz, R., Madesta, F., Nielsen, M., Krause, J., Steurer, S., Werner, R. and Rösch, T. Multi-scale fully convolutional neural networks for histopathology image segmentation: From nuclear aberrations to the global tissue architecture. *Medical Image Analysis*, 70 (2021/05/01/ 2021), 101996.
[16] Vigueras-Guillén, J. P., Sari, B., Goes, S. F., Lemij, H. G., van Rooij, J., Vermeer, K. A. and van Vliet, L. J. Fully convolutional architecture vs sliding-window CNN for corneal endothelium cell segmentation. *BMC Biomed Eng*, 1 (2019), 4-4.
[17] Hinton, G. E. and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313, 5786 (2006/07/28 2006), 504-507.
[18] He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. *arXiv e-prints* (2015), arXiv:1512.03385.
[19] Xu, Y., Zhou, Z., Li, X., Zhang, N., Zhang, M. and Wei, P. FFU-Net: Feature Fusion U-Net for Lesion Segmentation of Diabetic Retinopathy. *Biomed Res Int*, 2021 (2021), 6644071-6644071.
[20] Smith, A. G., Petersen, J., Selvan, R. and Rasmussen, C. R. Segmentation of roots in soil with U-Net. *Plant Methods*, 16, 1 (2020/02/08 2020), 13.
[21] Long, F. Microscopy cell nuclei segmentation with enhanced U-Net. *BMC Bioinformatics*, 21, 1 (2020), 8-8.
[22] Gauthier, J. *Conditional generative adversarial nets for convolutional face generation*. City, 2015.
[23] Tian, Z., He, T., Shen, C. and Yan, Y. Decoders Matter for Semantic Segmentation: Data-Dependent Decoding Enables Flexible Feature Aggregation. *arXiv e-prints* (2019), arXiv:1903.02120.
[24] Tofighi, M., Guo, T., Vanamala, J. K. P. and Monga, V. Prior Information Guided Regularized Deep Learning for Cell Nucleus Detection. *IEEE Transactions on Medical Imaging*, 38, 9 (2019), 2047-2058.
[25] Xie, Y., Xing, F., Shi, X., Kong, X., Su, H. and Yang, L. Efficient and robust cell detection: A structured regression approach. *Medical Image Analysis*, 44 (2018/02/01/ 2018), 245-254.
[26] Zheng, Y., Chen, Z., Zuo, Y., Guan, X., Wang, Z. and Mu, X. Manifold-Regularized Regression Network: A Novel End-to-End Method for Cell Counting and Localization. In *Proceedings of the Proceedings of the 2020 the 4th International Conference on Innovation in Artificial Intelligence* (Xiamen, China, 2020). Association for Computing Machinery, [insert City of Publication],[insert 2020 of Publication].
[27] Hussain, E., Mahanta, L. B., Das, C. R., Choudhury, M. and Chowdhury, M. A shape context fully convolutional neural network for segmentation and classification of cervical nuclei in Pap smear images. *Artificial Intelligence in Medicine*, 107 (2020/07/01/ 2020), 101897.
[28] Naylor, P., Laé, M., Reyal, F. and Walter, T. Segmentation of Nuclei in Histopathology Images by Deep Regression of the Distance Map. *IEEE Transactions on Medical Imaging*, 38, 2 (2019), 448-459.
[29] Emami, H., Aliabadi, M. M., Dong, M. and Chinnam, R. B. SPA-GAN: Spatial Attention GAN for Image-to-Image Translation. *IEEE Transactions on Multimedia*, 23 (2021), 391-401.
[30] Andreini, P., Bonechi, S., Bianchini, M., Mecocci, A. and Scarselli, F. Image generation by GAN and style transfer for agar plate image segmentation. *Computer Methods and Programs in Biomedicine*, 184 (2020/02/01/ 2020), 105268.
[31] Li, C. and Wand, M. *Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks*. Springer International Publishing, City, 2016.
[32] Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S. and Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *arXiv e-prints* (2018), arXiv:1803.10892.

[33] Zhou, Y. and Berg, T. L. *Learning Temporal Transformations from Time-Lapse Videos*. Springer International Publishing, City, 2016.
[34] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints* (2014), arXiv:1409.1556.
[35] Wang, Q., Gao, J., Lin, W. and Yuan, Y. J. a. e.-p. *Learning from Synthetic Data for Crowd Counting in the Wild*. City, 2019.
[36] Mirza, M. and Osindero, S. Conditional Generative Adversarial Nets. *arXiv e-prints* (2014), arXiv:1411.1784.
[37] Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints* (2015), arXiv:1502.03167.
[38] Ljosa, V., Sokolnicki, K. L. and Carpenter, A. E. Annotated high-throughput microscopy image sets for validation. *Nat Methods*, 9, 7 (2012), 637-637.
[39] Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H. and Yli-Harja, O. Computational Framework for Simulating Fluorescence Microscope Images With Cell Populations. *IEEE Transactions on Medical Imaging*, 26, 7 (2007), 1010-1016.
[40] Lehmussola, A., Ruusuvuori, P., Selinummi, J., Rajala, T. and Yli-Harja, O. Synthetic Images of High-Throughput Microscopy for Validation of Image Analysis Methods. *Proceedings of the IEEE*, 96, 8 (2008), 1348-1360.