electronics

Article

CellNet: A Lightweight Model towards Accurate LOC-Based High-Speed Cell Detection

Xianlei Long [1,2], Idaku Ishii [3] and Qingyi Gu [1,*]

[1] Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; longxianlei2017@ia.ac.cn
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[3] Smart Robotics Laboratory, Graduate School of Advanced Science and Engineering, Hiroshima University, Hiroshima 739-8527, Japan; iishii@robotics.hiroshima-u.ac.jp
* Correspondence: qingyi.gu@ia.ac.cn

Abstract: Label-free cell separation and sorting in a microfluidic system, an essential technique for modern cancer diagnosis, resulted in high-throughput single-cell analysis becoming a reality. However, designing an efficient cell detection model is challenging. Traditional cell detection methods are subject to occlusion boundaries and weak textures, resulting in poor performance. Modern detection models based on convolutional neural networks (CNNs) have achieved promising results at the cost of a large number of both parameters and floating point operations (FLOPs). In this work, we present a lightweight, yet powerful cell detection model named CellNet, which includes two efficient modules, CellConv blocks and the h-swish nonlinearity function. CellConv is proposed as an effective feature extractor as a substitute to computationally expensive convolutional layers, whereas the h-swish function is introduced to increase the nonlinearity of the compact model. To boost the prediction and localization ability of the detection model, we re-designed the model's multi-task loss function. In comparison with other efficient object detection methods, our approach achieved state-of-the-art 98.70% mean average precision (mAP) on our custom sea urchin embryos dataset with only 0.08 M parameters and 0.10 B FLOPs, reducing the size of the model by 39.5× and the computational cost by 4.6×. We deployed CellNet on different platforms to verify its efficiency. The inference speed on a graphics processing unit (GPU) was 500.0 fps compared with 87.7 fps on a CPU. Additionally, CellNet is 769.5-times smaller and 420 fps faster than YOLOv3. Extensive experimental results demonstrate that CellNet can achieve an excellent efficiency/accuracy trade-off on resource-constrained platforms.

Keywords: cell detection; high-speed vision; convolutional neural network (CNN); efficient convolutional block; medical image analysis

Citation: Long, X.; Ishii, I.; Gu, Q. CellNet: A Lightweight Model towards Accurate LOC-Based High-Speed Cell Detection. *Electronics* **2022**, *11*, 1407. https://doi.org/10.3390/electronics11091407

Academic Editors: Nantheera Anantrasirichai and Oktay Karakuş

Received: 30 March 2022
Accepted: 26 April 2022
Published: 28 April 2022

1. Introduction

There are increasing demands for deep learning aids for biomedical research and clinical diagnosis, as these aids would enable superhuman performance for many challenging medical tasks. The ability to accurately detect certain types of cells in flow cytometry is of significant importance to the performance of cell counting, analysis, and sorting [1–5]. Numerous precision and highly integrated cell analysis platforms have been used to assist biomedical scientists, who previously had to inspect cells massively, while working under pressure with a high workload, and who were consequently prone to make fatigue-induced errors. However, these off-line processing systems cannot perform high-throughput cell detection in a fast-flowing microchannel in real-time [6]. Apart from this, some methods need fluorescent labeling, which negatively affects cell migration [7]. Imaging flow cytometry is an emerging technology that combines flow cytometry with microscopy to enable multiparametric analysis of single cells at a high throughput. However, this technology

acquires images at a high spatial resolution, yet it relies on simple and low-level image processing algorithms, which greatly increases the cost with little performance gain [6,7].

Recent progress in vision-based cell detection models has made it possible to record cells in microfluidic lab-on-a-chip (LOC) devices. Although many of them have achieved rapid cell sorting and morphology analysis [8], they are limited by background noise, occlusion, and weak boundaries. If the complex features of cells could be simultaneously detected and located by efficient detection models combined with a microfluidic LOC, the accuracy and efficiency of automated intensity cell analysis could be remarkably improved.

Cell detection needs to resolve two fundamental tasks: recognition and localization. When inputting an image, the detection system needs to output the corresponding specific categories of the detected cells in this image and their locations. Traditional methods, such as texture, color, and HOG, are based on the design of effective handcrafted features [9–12]. However, in most cases, they are limited by rotation and scale changes. The rapid development of convolutional neural networks (CNNs) in the computer vision community has led to the recent proposal of many CNN-based object detection models. For example, region-based CNN (R-CNN) [13], Fast-RCNN [14], and Faster-RCNN [15] are a series of classical two-stage detection models. However, these two-stage detectors are computationally intensive, which limits their usage in real-time cell detection systems. As a result, to alleviate problems related to the inference time, one-stage object detection frameworks have drawn researchers' attention. In this regard, YOLO [16] and SSD [17] are two popular unified models that combine the prediction of class probabilities and the localization of bounding boxes into one deep CNN model. These models are more straightforward to train by using an end-to-end optimization scheme. Since then, YOLOv2 [18] and YOLOv3 [19] were proposed successively to achieve an excellent trade-off between speed and accuracy.

However, the above models improve the accuracy at the cost of computational efficiency and memory requirements, which means that existing detection approaches are nowhere near achieving the processing speed required for real-time cell detection. These limitations prompted the construction of a small-sized model with low latency by employing depthwise convolution and group convolution, two modules that enable the model parameters to be compressed and the computational cost to be reduced. In practice, this led to the development of successive versions of the MobileNet series, which are customized for resource-constrained platforms and mobile devices [20,21]. Nevertheless, simply integrating MobileNet blocks into a high-speed cell detection system as a feature extraction backbone is unlikely to accomplish a promising gain in accuracy and efficiency, mainly owing to the redundancy of building blocks.

The cell detection model we introduce in this paper, CellNet, advances state-of-the-art technology by significantly decreasing the memory requirements and expensive FLOPs, yet it yields promising results. We demonstrate the combination of a one-stage detection header with novel CellConv blocks to build a lightweight, real-time, high-speed cell detection model.

The main contributions of this paper are as follows:

(1) A lightweight, yet powerful cell detection model: We propose a robust and fast end-to-end cell detection system, CellNet, which is small in size and highly efficient. The novel CNN architecture is specifically tailored for resource-constrained environments.
(2) A novel CellConv block to reduce the number of parameters and FLOPs: The proposed CellConv block, which is an effective convolutional block to reduce the parameters and FLOPs of the conventional convolutional layer, is widely used as a building block in the feature extraction backbone of CellNet.
(3) CellNet is designed for easy deployment on mobile devices: The compact and regular architecture of the proposed CellNet facilitates its deployment on graphics processing units (GPUs), CPUs, field-programmable gate arrays (FPGAs), and in other resource-limited environments to build a real-time cell detection system that accelerates online LOC-based fast-flowing cell analysis.

2. Related Work

In recent years, high-throughput cell detection in microscopy images has emerged as a powerful method to analyze circulating tumor cells (CTCs) in an unprecedentedly accurate and efficient manner [6]. Several researchers focused on cell detection by using traditional image processing techniques, such as intensity thresholding, region growing, k-means, morphological filtering, and graph cuts. The Laplacian-of-Gaussian (LoG) operator based on the Euclidean distance map has been applied to blob detection and automatic nuclei localization. LBP features are enriching many interesting textures [22], which have been used for cell detection tasks [23]. Several kernel-based radial voting methods were introduced to iteratively localize cells. Cosatto et al. employed the difference of Gaussian (DoG) followed by a Hough transformation to detect radially symmetric shapes on pathology images [24]. Similarly, Veta et al. applied the direction of the gradient to identify the central nuclei [4]. However, these methods may fail to detect irregularly shaped cells. Ali et al. proposed an active contour-based shape model to detect and segment overlapping cells and obtained promising results on their cell datasets [25]. Gu et al. developed an FPGA-based cell-labeling algorithm to extract multiple cells in real-time at 2000 frames per second (fps) [26]. The algorithm was applied to a microfluidic-based fast-flowing analysis system. The detection results of the cell-labeling method were found to be limited when two cells or more were stuck together. Vink et al. employed an AdaBoost classifier to train two detectors, one based on Haar-like features and the other using pixel-based features, after which the outputs of the two detectors were merged together using an active contour algorithm to detect the nuclei in breast tissue images [27]. Other traditional detection methods employed the histogram of oriented gradients (HOG) features [11], scale-invariant feature transform (SIFT) features [28], etc., to improve the accuracy. Despite the success of the aforementioned methods, their performance relies extensively on the design of handcrafted features, such as gradients, histograms, and region shapes. However, because of their lack of prior knowledge about the target cells and other cells, it is difficult for researchers to select suitable features. Besides, most traditional handcrafted features contain many empirical hyperparameters that are crucial for the overall results.

Recent studies have demonstrated that deep-neural-network (DNN)-based methods are more robust and superior to traditional features. CNN-based models have a remarkable ability to extract task-specific feature representations, which helped the models achieve state-of-the-art performance on several benchmarks, including biomedical image analysis datasets. Heo et al. proposed an R-MOD cell detector by using an FCRN structure to segment images [6]. The segmented images were then input into a "Flattening" algorithm to output a probability density map, thereby enabling cells to be localized according to the means and variances. They achieved 500 fps and 93.3% mAP. Nitta et al. designed an intelligent image-activated cell sorting (IACS) system [5], which integrates high-throughput cell microscopy, focusing, and sorting, enabling real-time automated operation for image acquisition, processing, decision-making, and actuation. The IACS system was conducted on CPUs, GPUs, and FPGAs, which employed a six-layer CNN model for cell classification. The system achieved 99.0% specificity and 82.0% sensitivity with a latency of only 32 ms. A stacked sparse auto-encoder (SSAE), which was employed to learn features from a sliding window patch following which these features were fed into a softmax classifier to output the probabilities of nuclear or non-nuclear samples, was reported by Xu et al. in [29]. The SSAE achieved a 78.83% mAP and an 84.49% F-measure and outperformed nine popular detection methods. Ciresan et al. proposed a CNN model for mitotic detection in breast cancer images [30], where the CNN model was trained to regress the probability of either belonging to a mitotic or the background for each pixel. Xie et al. proposed a novel structured regression model based on a fully residual CNN for efficient cell detection [31], which produces a dense proximity map that shows higher responses at locations near cell centers. A framework combined with CNN and compressed sensing was used for cell detection and output space encoding by Xue et al. [32] and showed comparable performance for microscopy images. Dong et al. applied a supervised max-pooling CNN to detect zebrafish

larval regions that were preselected by an SVM classifier [33]. The spatially constrained (SC) CNN has also been utilized to perform nucleus detection [34]. SC-CNN first uses regression to determine the likelihood of a pixel being the center of a nucleus, then couples a neighboring ensemble predictor to more accurately predict the class probabilities of detected cells. Most of these systems focus on specific medical images, and their performance is greatly compromised when migrating to images of other types of cells or tasks.

In the field of general object detection, a region-based CNN (RCNN) was proposed by Girshick et al. [13]. The RCNN first generates 2000 region proposals by using selective search and a CNN-based feature extractor, then feeds these features into a classifier and a bounding box regression module to obtain the final results. Girshick et al. additionally designed Fast RCNN based on SPPNet and R-CNN to further improve the detection performance [14]. However, their inference speed was relatively slow for the time-consuming region proposals stages. Attempts to address this problem led to the proposal of the region proposal network (RPN), which uses the convolutional features of an image as candidates with a negligible processing time and achieved a processing rate of 7 fps [15]. A sequence of improvements has enabled these two-stage detection frameworks to attain state-of-the-art detection performance on various challenging benchmarks, such as the COCO [35] and VOC [36] datasets.

At the same time, one-stage detectors demonstrated promising results and were faster than the best existing two-stage ConvNet. A unified architecture named YOLO, which was proposed by Redmon et al. [16], operated at 45 fps and combined the prediction of class probabilities and the localization of bounding boxes into one deep CNN. Liu et al. proposed the single-shot multibox detector (SSD) to directly predict each object category and the adjustments to each bounding box on multiple resolutions [17]. Subsequently, to more optimally balance accuracy and latency, YOLOv2 [18], RetinaNet [37], and YOLOv3 [19] were proposed in succession.

Despite their accuracy, these approaches are too slow for real-time high-speed applications [12,38–41]; even equipment with high-end hardware is not sufficiently efficient because of the computationally expensive convolutional operation and large memory requirements. The ubiquity of resource-constrained mobile equipment and robotics has made it important to design efficient mobile-sized CNNs. In this regard, many simple, but powerful neural network architectures have emerged recently, such as SqueezeNet [42], MobileNets [20,21], and ShuffleNets [43]. These networks focus on optimizing the latency, but also have a small model size. Depthwise convolution and pointwise convolution are always combined to reduce the parameters while decreasing the number of FLOPs. However, for task-specific applications, such a mobile-sized model would need architecture-level optimization to meet the requirement of real-time high-speed cell detection.

3. Designing Efficient Building Blocks

Traditional CNN blocks are inefficient in terms of convolutional weights and FLOPs. In this section, we first review previous progress in the design of efficient convolutional blocks and then discuss the new feature extraction convolutional blocks, named CellConv blocks, which were used to build our CellNet.

3.1. Depthwise Separable Convolutions

MobileNetV1 introduced the use of depthwise separable convolutions as a computationally efficient block to replace traditional convolutional layers [20]. The basic idea is to factorize a convolutional layer into a spatial filtering layer and a feature generation layer. The first layer, which employs depthwise convolution, applies a single convolutional kernel per input channel to perform both lightweight and efficient spatial filtering. The second layer performs pointwise convolution and uses 1×1 convolution to create a linear combination of the output of the depthwise layer.

Traditional convolution takes an $H_i \times W_i \times C_i$ input tensor F_i and applies a $K \times K \times C_i \times C_o$ convolutional kernel to produce an $H_i \times W_i \times C_o$ output tensor F_o (we assume that

the output tensor has the same width and height as the input tensor, respectively, and there are no biases in convolutional layers). The total computational cost is:

$$H_i \times W_i \times C_i \times C_o \times K \times K. \tag{1}$$

The computational cost of depthwise separable convolutions includes two parts. For depthwise convolutions, both the input and output tensor are $H_i \times W_i \times C_i$, and the convolutional kernel is $K \times K \times C_i$. Therefore, the cost is $H_i \times W_i \times C_i \times K \times K$, while the pointwise convolution only costs $H_i \times W_i \times C_i \times C_o \times 1 \times 1$ computations by using a $1 \times 1 \times C_i \times C_o$ computational kernel. A substantial literature shows they can work as well as regular convolutions with an extremely efficient cost:

$$H_i \times W_i \times C_i(K^2 + C_o), \tag{2}$$

which is the sum of depthwise convolutions and 1×1 pointwise convolutions. In most cases, the kernel size K is set to three; thus, the computational cost is 8–9-times smaller than the standard convolutions according to Equation (3):

$$\frac{K^2 + C_o}{K^2 \times C_o} = \frac{1}{C_o} + \frac{1}{K^2}. \tag{3}$$

3.2. Inverted Residual and Linear Bottlenecks

Linear bottlenecks assume the manifold of interest to be embedded into a low-dimensional subspace of the higher-dimensional activation space [21], to ensure that the input information is preserved by linear transformations. As nonlinearities are likely to destroy too much information, it is of crucial importance not to jeopardize the performance by inserting linear bottlenecks into convolutional blocks. This structure is built by inserting a 1×1 expansion layer followed by a depthwise convolutional layer and a 1×1 projection layer, which projects the higher-dimensional information to a compact representation. If the input and output feature maps have the same number of channels (i.e., $C_i = C_o$), a shortcut connection is used to connect the input and output features. The expansion factor t is assigned a fixed value of 6, such that the structure is narrow at both ends and wide in the intermediate layer. With reference to classical residual blocks, these blocks are informally known as inverted residual blocks.

3.3. Efficient CellConv Blocks

Inspired by the efficient convolutional blocks of the MobileNet series, we designed a more lightweight building block, named CellConv block. The details of the implementation are provided in Table 1. The CellConv block aims to optimize the trade-off between computational cost and detection speed by transforming the input information to a high-dimensional space by a relatively small expansion factor $t = 3$.

Table 1. The construction of CellConv block.

Input Tensor	Operations	Output Tensor
$H_i \times W_i \times C_i$	1×1 PW Conv, H-Swish	$H_i \times W_i \times (tC_i)$
$H_i \times W_i \times (tC_i)$	3×3 DW Conv, H-Swish	$H_i \times W_i \times (tC_i)$
$H_i \times W_i \times (tC_i)$	1×1 PW Conv, Linear	$H_i \times W_i \times C_o$

Thus, the total computational cost is:

$$H_i \times W_i \times tC_i(C_i + K^2 + C_o). \tag{4}$$

Compared with Equation (2), the CellConv block contains an additional pointwise convolution term, but the expressiveness of the network is stronger than that of simple

depthwise separable convolutions. Our CellConv block is 2× more computationally efficient and 2× smaller in terms of the number of parameters than the aforementioned inverted residual and linear bottlenecks. Thus, our block mitigates the effect of excessive depthwise convolution on the reduction of the inference speed. Analogous to linear bottlenecks, the last projection layer does not include any nonlinear terms; instead, computation occurs via a linear function. Figure 1 shows the diagram of the CellConv block.

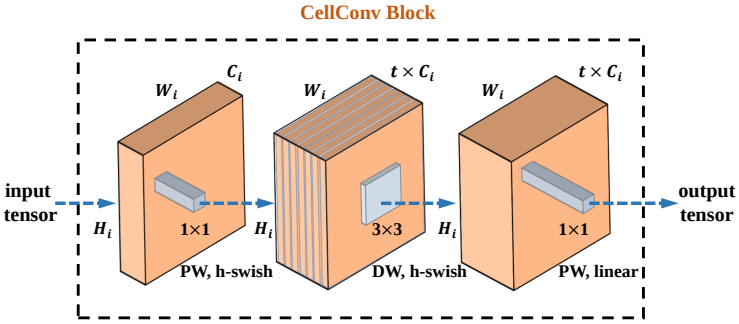

Figure 1. The diagram of the CellConv block. It first employs the 1×1 pointwise convolutional layer, then uses the 3×3 depthwise convolutional layer, and again, employs the 1×1 pointwise convolutional layer, where PW represents pointwise convolution and DW represents depthwise convolution.

To further improve the detection accuracy, we used the h-swish function, which is a modified version of the swish function, to strengthen the nonlinearities of the CellConv blocks. *Swish* nonlinearity was introduced previously in [44] as a replacement for ReLU and is defined as Equation (5):

$$swish(x) = x \cdot \sigma(x), \tag{5}$$

where $\sigma(x)$ represents the sigmoid function. The computational cost associated with the *swish* function on resource-constrained platforms is non-negligible. This prompted us to introduce the *h-swish* function to construct a nonlinear function that enables faster computing and is easier to deploy. The *h-swish* function is as simple as follows:

$$h\text{-}swish(x) = x \cdot \frac{ReLU6(x+3)}{6}, \tag{6}$$

where *ReLU6* is describe as:

$$ReLU6(x) = \begin{cases} 0 & (x < 0); \\ x & (0 \leq x < 6); \\ 6 & (x \geq 6). \end{cases} \tag{7}$$

A comparison of the hard and soft versions of the swish function can be seen in Figure 2. This function is a good approximation of the smooth one. In our experiments, we note that the h-swish nonlinearity achieved a small improvement on the mAP with no discernible difference in processing speed, but with multiple advantages when deployed. First, as the ReLU6 nonlinearity is easy to calculate, h-swish is available in almost all software and hardware environments. Second, h-swish is a general method to compress the detection model by using low-bit quantization when deploying models on FPGA or mobile devices. The h-swish function is quantization friendly, which could eliminate the potential numerical precision loss caused by other approximations of sigmoid functions [45]. Additionally, as we demonstrate in Section 5, replacing ReLU6 by h-swish increases the accuracy by 0.79% with only 2.82 M additional FLOPs, indicating the positive impact of these optimizations on the accuracy and detection speed.

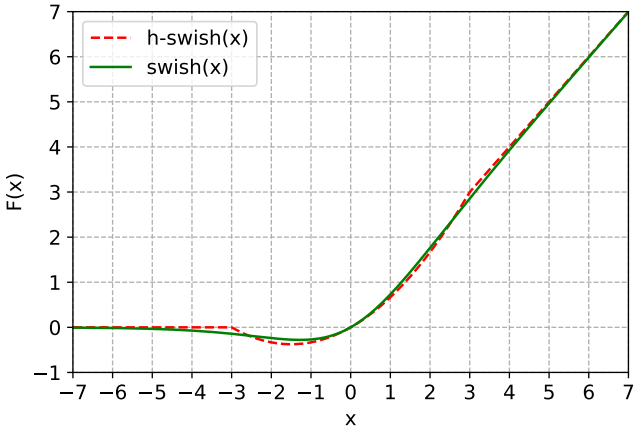

Figure 2. Comparison of swish nonlinearity and the modified hardware-oriented h-swish nonlinearity. We can see that the swish function is well approximated by h-swish.

4. The Proposed CellNet for Cell Detection

In this section, we first introduce CellNet, a new CNN architecture for detecting cells in a fast-flowing microchannel at real-time speed. Then, to further improve the recall rates of hard examples, we used predefined optimization anchor boxes, which were placed on the input image. In addition, a newly designed multi-task loss function was introduced for joint optimization of CellNet in the training phase.

4.1. CellNet Architecture

As shown in Figure 3, CellNet is a simple and unified one-stage detection network composed of a backbone network and a detection layer. The backbone, which is responsible for extracting convolutional features over the entire input image, is a dedicated and efficient CNN. The backbone is used in combination with the YOLOv3 detection header as our detection layer to meet the high-speed requirement of a cell analysis system. Thus, we obtained an extremely efficient trade-off between accuracy and speed.

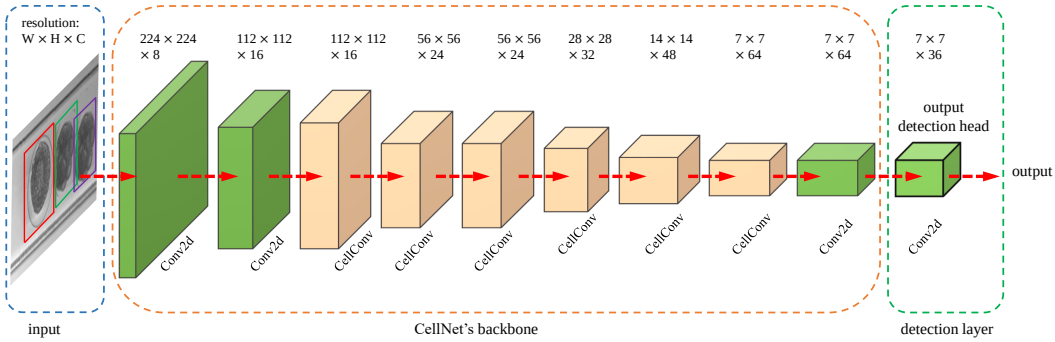

Figure 3. The architecture of CellNet, which includes 3 parts: input, backbone, and detection layer. The green block represents the basic 2D convolutional block, while the yellow block represents the CellConv block.

4.1.1. CellConv Network Backbone

Despite the availability of many possible detection components, they are neither compact nor efficient. Most designs in particular target generic object detection datasets (e.g., COCO and VOC). However, the gain in efficiency diminishes when these larger CNN models are utilized as backbone networks for cell detection. In terms of accuracy, the existing Faster R-CNN and FPN are more accurate at the cost of computational efficiency and memory consumption. The original feature extractor in YOLOV3 is Darknet-53, which is a hybrid approach between Darknet-19 and novel residual network components. However, it contains 53 convolutional layers, resulting in 40.59 M parameters, and requires

a tremendous amount of computation with 7.14 B FLOPs, which prevents the model from be deployed and achieving high-speed processing. The need to address the imbalance between the processing accuracy and speed motivated us to use the CellConv block as the basic convolutional block to construct the backbone network in our detector.

The detailed structure of CellNet is shown in Table 2. As many efficient networks always employ a conventional convolutional layer in the first and last few stages, we adopted the same design scheme to build the backbone. The beginning two layers are basic convolutional layers (Conv2d), named Conv1 and Conv2, which extract low-level features of cells with 8 and 16 filters, respectively. The kernel size is 3×3 with different strides (i.e., $3 \times 3 \times 8$-stride1 for Conv1 and $3 \times 3 \times 16$-stride2 for Conv2). Each convolutional layer consists of a batch normalization layer and a LeakyReLU (LK) nonlinearity. Next are five successive CellConv layers, i.e., CellConv3 to CellConv8, where the implementation details of CellConv are described in Table 1. We fixed the expansion factor t to 3 in this paper, so the channels of the expansion layer (exp size) were 3-times the input channels. With the CellNet going deeper, the number of output channels (#out) of each CellConv block was gradually increased from 16 to 64. As discussed previously, the CellConv blocks utilize h-swish (HS) nonlinearity to improve the backbone's performance. The stride for different CellConv stages is different by considering halving the resolution of feature maps to save computational resources. After the base feature extractor, we added a transition layer called Conv9, which is also a conventional convolutional layer (Conv2d). Then, we fed these features to the detection layer for further processing. The structure of the backbone is presented in Figure 3.

Table 2. Architecture of CellNet.

Stage	Input	Operator	Exp Size	#Out	NL	Stride
Conv1	$224^2 \times 3$	Conv2d, 3×3	-	8	LK	1
Conv2	$224^2 \times 8$	Conv2d, 3×3	-	16	LK	2
CellConv3	$112^2 \times 16$	CellConv, 3×3	48	16	HS	1
CellConv4	$112^2 \times 16$	CellConv, 3×3	48	24	HS	2
CellConv5	$56^2 \times 24$	CellConv, 3×3	72	24	HS	1
CellConv6	$56^2 \times 24$	CellConv, 3×3	72	32	HS	2
CellConv7	$28^2 \times 32$	CellConv, 3×3	96	48	HS	2
CellConv8	$14^2 \times 48$	CellConv, 3×3	144	64	HS	2
Conv9	$7^2 \times 64$	Conv2d, 3×3	-	64	LK	1
Det10	$7^2 \times 64$	Conv2d, 1×1	-	36	-	1

Note: exp size denotes the channel number of the expansion layer in the CellConv blocks. #out denotes the output channel of the conv blocks. NL denotes the type of nonlinearity function used. Here, LK denotes LeakyReLU and HS denotes h-swish.

4.1.2. Detection Layer

The detection layer in our network, named Det10, is inspired by YOLOv3, which enables the prediction of convolutional objects and the localization of bounding boxes to be performed in a single network. However, we only used one-scale feature maps to perform detection, and we only predicted seven classes, while YOLOv3 detected several objects at multiple resolutions, which was more complicated. Det10 uses 1×1 pointwise convolution to integrate input feature maps into a fixed-size 3D grid cell (i.e., $1 \times 1 \times 36$-stride1). The output tensor is an encoding of the objectness, class predictions, and bounding boxes. In our task, as the cell size does not differ hugely, we simply used a single scale to predict all three boxes, thereby preventing redundant cell detection from occurring in a feature pyramid regime. Following the CellConv backbone, the last feature extraction layer has an output stride of 32, indicating the output of the feature map is 7×7 in width and height. Thus, at the detection layer, the final prediction tensor is $7 \times 7 \times [3 \times (4 + 1 + 7)]$ for three bounding boxes. This tensor contains four bounding box offsets, one objectness prediction, and seven predictions of cell classes. The construction of our detection layer is illustrated in Figure 4.

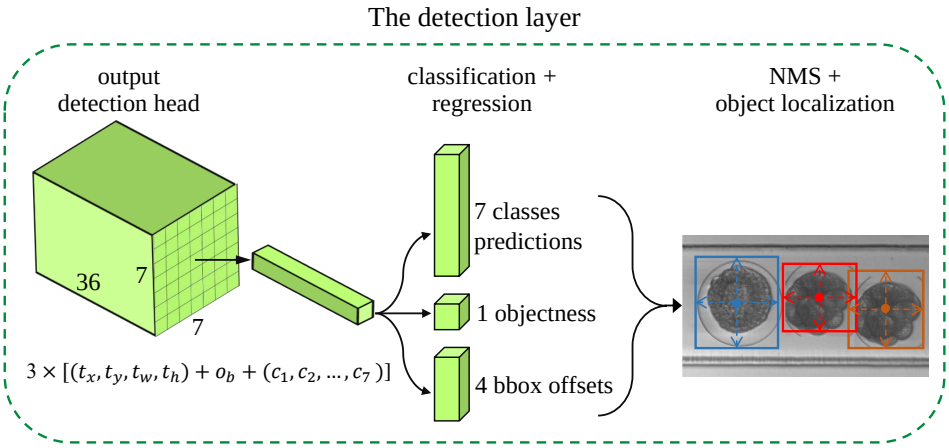

Figure 4. Illustration of the detection layer. It first decomposes the output tensor of CellNet's backbone into three parts, including four bounding box offsets, one objectness prediction, and seven class predictions. Then, it employs non-maximum suppression (NMS) to refine the detection and, finally, localizes the detected objects.

The four bounding box offsets represent the location coordinates relative to the corresponding grid cell, t_x, t_y, t_w, and t_h. Specifically, for each prediction center, (t_x, t_y), we used a logistic activation function to constrain the CellNet prediction to fall in a bound between 0 and 1. If the cell has an offset (c_x, c_y) relative to the top left corner of the image and the anchor box prior has width p_w and height p_h, then the bounding box prediction corresponds to:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x, \; b_y = \sigma(t_y) + c_y, \\ b_w &= p_w e^{t_w}, \qquad b_h = p_h e^{t_h}. \end{aligned} \tag{8}$$

Our modified detection header predicts on 7×7 high-level semantic feature maps, benefiting from fine-grained features for the localization of large objects. Our analysis indicated that the size of different stages of sea urchin embryo cells ranges from 50 to 80 pixels in a 224×224 full image, which is a relatively small variance. Consequently, these images are a natural choice to detect cells at one scale without building a feature pyramid structure. Additionally, our CellNet can detect cells of various sizes with only a minimal change by fusing multi-scale features via the FPN architecture.

4.2. Refinement of Anchor Scales

We assigned a specific scale of anchors to the detection layer to improve the precision. Instead of selecting the anchor priors by hand, a process that was dominated by empirical selection, we ran k-means clustering on the *training* dataset. The output anchors differ from the existing settings suggested for the size and aspect ratios of the anchors (i.e., 64×64, 1:1, 1:$\sqrt{2}$, $\sqrt{2}$:1). In this study, for simplicity, we assigned a fixed value of 3 to k. As described previously in [18], the Euclidean distance is expected to generate more error boxes. Therefore, we chose the Jaccard distance to calculate the distance from the samples to the clustering centers, which is independent of the bounding box size. Thus, the distance metric is:

$$d_J(box, centeriod) = 1 - J(box, centriod), \tag{9}$$

where $J(box, centriod)$ is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{10}$$

Intuitively, d_J ranges from 0 to 1. For our cell dataset, the three clusters are: (65×41), (60×59), and (52×63). We employed the following anchor assignment rules: anchors

were assigned to ground-truth object boxes by using the best intersection over union (IoU) that exceeds the IoU threshold of 0.5. As each anchor was assigned to a top-one IoU object box, and the corresponding entry was set to 1; otherwise, if an anchor is unassigned, it was ignored during the training phase.

4.3. Training of CellNet

Given an input image x and a ground-truth label y, we define our multi-task loss function to jointly optimize CellNet's parameters as follows:

$$Loss(x, y) = \lambda_{cls} L_{cls} + \lambda_{reg} L_{reg} + \lambda_{conf} L_{conf}, \tag{11}$$

where L_{cls} is the classification loss, L_{reg} is the represented bounding box regression loss, and L_{conf} is the objectness confidence loss. Further, λ_{cls}, λ_{reg}, and λ_{conf} are the hyper-parameters for balancing three tasks.

For the classification counterpart L_{cls}, we use the sigmoid-activation-processed binary cross-entropy (BCE) loss as the training criterion:

$$\begin{aligned} L_{cls} = \sum_{i=1}^{M \times N} \sum_{j=1}^{A} \mathbb{1}_{ij}^{obj} [- \sum_{k=1}^{K} [& C_k log(\sigma(\hat{C}_k)) \\ & + (1 - C_k) log(1 - \sigma(\hat{C}_k))]], \end{aligned} \tag{12}$$

where C_k is the ground-truth class label for each prediction $\hat{C}_k$. $\mathbb{1}_{ij}^{obj}$ is the indicator function, when cell i has a ground-truth box j, $\mathbb{1}_{ij}^{obj} = 1$, and vice versa. M, N denote the width and height of the grid cells in the output of the detection header, and A is the number of bounding boxes for one grid cell ($A = 3$ in our study). As CE with softmax activation emphasizes the assumption that there is only one exact category for each bounding box, the corresponding prediction is enhanced, whereas others are suppressed, which is often not the case in practice. In contrast, our experimental environment equipped with the microchannel can flow through multiple cells at a time in parallel; thus, a bounding box that may contain several highly overlapping cells frequently occurs. Therefore, the BCE loss with a sigmoid function for multi-label classification is more suitable in our design scenario.

The bounding box regression loss L_{reg} is simple: we used the mean of the squared error (MSE) loss to train the bounding box prediction as follows:

$$\begin{aligned} L_{reg} &= L_{bbox}^{xy} + L_{bbox}^{wh} \\ &= \sum_{i=0}^{M \times N} \sum_{j=0}^{A} \mathbb{1}_{ij}^{obj} \frac{1}{N_{reg}} [(t_x - \sigma(\hat{t}_x))^2 \\ &\quad + (t_y - \sigma(\hat{t}_y))^2 + (t_w - \hat{t}_w)^2 + (t_h - \hat{t}_h)^2]. \end{aligned} \tag{13}$$

Here, as mentioned above in Section 4.1.2, we used the sigmoid-processed $\hat{t}_x$ and $\hat{t}_y$ as the offset predictions of the box center, which are related to the corresponding grid cell i if anchor j is the responsible bounding box. N_{reg} is the number of ground-truth boxes.

The last term is the BCE loss L_{conf}, which aims to measure how confident we are that every grid cell contains objects, i.e.,

$$\begin{aligned} L_{conf} = \sum_{i=0}^{M \times N} \sum_{j=0}^{A} [& -O_b log(\sigma(\hat{O}_b)) \\ & - (1 - O_b) log(1 - \sigma(\hat{O}_b))]. \end{aligned} \tag{14}$$

where O_b denotes the objectness of each grid cell, 1 if the grid cell contains the center of objects, otherwise 0. In this way, the network concentrates more on the corresponding grid cells by using logistic regression. Therefore, these grid cells have predictions $\sigma(\hat{O}_b)$ close to 1.

5. Experiments and Results

This section first describes our dataset and experimental setup. Then, we analyze the experimental results on classification tasks to demonstrate the effectiveness of the proposed CellNet. Finally, we present the systematic evaluation and comparison of the performance of CellNet against YOLOv3, SSD, and other object detection networks based on MobileNetV2 on our custom cell dataset.

5.1. Datasets and Experimental Setup

5.1.1. Datasets

The detection dataset was constructed based on our previous works [8], which aimed to design a real-time microchannel-based high-throughput cell morphology analysis system. We verified the performance of the hardware-implemented vision platform by conducting several experiments to analyze sea urchin egg cells in a straight microchannel [26,46]. The system consisted of a dual high-speed camera, a microscope, a metal–halide light source, an electric syringe pump, and microfluidic chips with microchannels (6 cm $\times$ 200 µm $\times$ 100 µm; length $\times$ width $\times$ depth). Apart from feature extraction, the platform also recorded 512 $\times$ 256 original images successively at 4000 fps in PC memory under 10$\times$ magnification through a microscope. The cell dataset was composed of the portion of captured images. Figure 5 shows snapshots of sea urchin embryos at different phases after fertilization. In practice, each image may contain 1, 2, or 3 cells. Based on cytology, cell development is divided into seven different stages, i.e., fertilized (one-cell stage), two-cell stage, four-cell stage, anaphase, blastula, gastrula, and pluteus (larval). According to these criteria for cell classification, our custom cell dataset consisted of 11,839 annotated cell bounding boxes in 6589 images with variations in rotation, occlusion, and illumination. We split the dataset into three subsets: *training*, *validation*, and *test*, which included 4575, 1100, and 1014 images, respectively. Following common practice, we trained our model on the *training* set, then analyzed our method by performing ablation studies on the *validation* set. We report the results in terms of the mean average precision (mAP) on the *test* set in comparison with state-of-the-art methods.

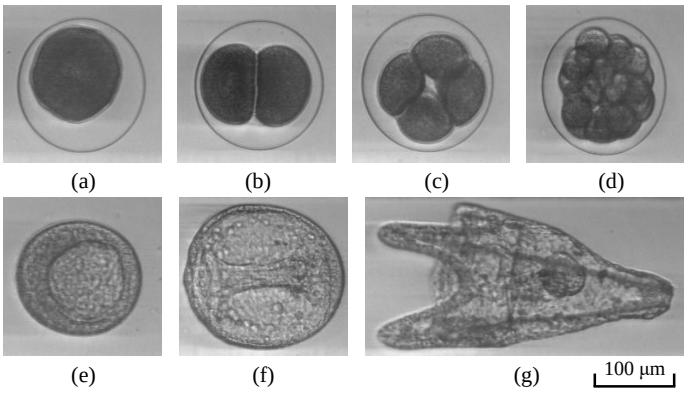

Figure 5. Seven snapshots of sea urchin embryos at different phases in the cell cycle after fertilization. (**a**) Fertilized. (**b**) Two-cell stage. (**c**) Four-cell stage. (**d**) Anaphase. (**e**) Blastula. (**f**) Gastrula. (**g**) Pluteus.

5.1.2. Experimental Setup

To enable a fair comparison, all methods were run on a PC with an Intel Xeon E5-2680 v4 2.4 GHz CPU, 256 GB memory, and a single GTX 1080 Ti GPU.

Initialization. The initialization helps stabilize the learning of CellNet in early iterations by preventing gradients from vanishing or exploding. Considering there is no off-the-shelf pre-trained backbone network for our newly designed model, we trained CellNet from scratch on our sea urchin egg dataset. The base convolutional layers in Conv2d and CellConv were initialized by the "Xavier" method with biased $b = 0$.

Optimization. We used the stochastic gradient descent (SGD) algorithm to optimize the multi-task loss $Loss(x, y)$ on one GPU with 0.9 momentum, 0.0005 weight decay, and batch

size 128. The trade-off parameters λ_{cls}, λ_{reg}, and λ_{conf} were set to 4, 1, and 32, respectively. The model was trained for 35k iterations with an initial learning rate of 0.015, which was multiplied by 0.97 every seven epochs. To construct a robust model and prevent overfitting, we used some data augmentation strategies to augment our dataset. In addition to the standard horizontal image flipping, each training image was randomly affine transformed, and we also randomly selected some photometric distortions algorithms to change the hue, saturation, and value of the training images, respectively.

Inference. At the inference phase, CellNet outputs a large number of detection boxes. We first filtered out most of the boxes with a confidence threshold of 0.35, then non-maximum suppression (NMS) with a threshold of 0.45 was applied for each class separately to generate the final high-confidence detections. There are more advanced ways to perform post-processing, such as test-time image augmentations or optimized threshold. To fairly compare against the baseline models, we used the same size of the input image and the same thresholds.

5.2. Cell Dataset Classification Performance

First, to shed light on the efficiency of the backbone we designed for CellNet, we evaluated the cell classification capability of the proposed method and compared the accuracy by using various measures of resource usage such as the number of parameters and FLOPs. The classification images were randomly cropped from the detection dataset described above. We compared our networks with four state-of-the-art models: ResNet-18 [47], DarkNet-53, MobileNetV2, and MobileNetV3. Then, we studied the efficiency by comparing CellNet with the four representative models. Notably, Darknet-53 is the backbone of YOLOv3; the MobileNet series is widely recognized as being efficient and targets resource-constrained platforms. Table 3 presents the performance results. As can be seen in Table 3 and Figure 6, our CellNet uses an order of magnitude fewer parameters and 2.5× fewer FLOPs than other models with similar accuracy.

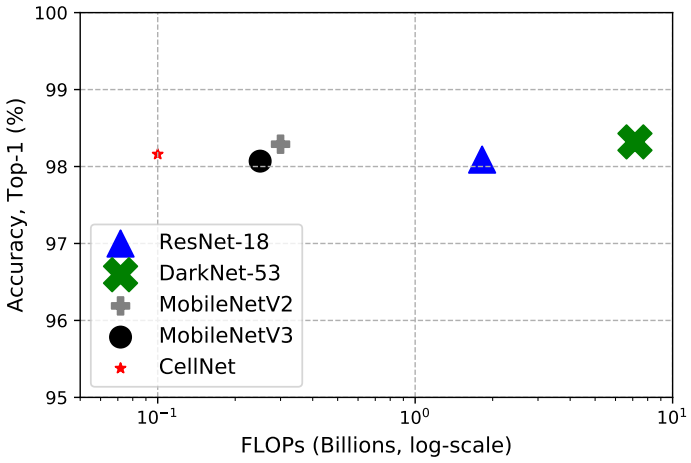

Figure 6. FLOPs vs. model size vs. accuracy. All models are run on the cell classification dataset. The relative sizes of each model are drawn in the figure by the corresponding markers. Our CellNet significantly outperforms the other four efficient models in terms of FLOPs and model size. In particular, CellNet is 2.5× more efficient than MobileNetV3 and is 27.9× smaller than the 2nd smallest model (i.e., MobileNetV2). To facilitate the comparison, FLOPs are drawn in a log-scale. Details are in Table 3.

Compared with Darknet-53, CellNet achieved 98.16% accuracy with only 0.08 million parameters and 0.10 billion FLOPs, considering that CellNet is 507.4× smaller in size and 71.4× computationally more efficient. In particular, CellNet is more accurate than MobileNetV3, but utilizes 47.5× fewer parameters and 2.5× fewer FLOPs. These experimental results show that CellNet is able to capture rich feature representation to perform classification tasks with a constrained small size and much reduced computational cost.

Table 3. Performance results on cell classification.

Network	Top-1 Acc	#Params	#FLOPs
ResNet-18	98.09%	11.18 M [1]	1.82 B [2]
DarkNet-53	**98.32%**	40.59 M	7.14 B
MobileNetV2	98.29%	2.23 M	0.30 B
MobileNetV3	98.07%	3.80 M	0.25 B
CellNet (Ours)	98.16%	**0.08 M**	**0.10 B**

[1] M represents millions; [2] B represents billions.

5.3. Analysis of Cell Detection Results

This subsection presents our evaluation of the performance in several respects of the proposed CellNet on the custom cell *test* set. Selected cell detection results are also presented in Figure 7. Because our focus is on real-time efficient models, rather than comparing the performance with other architectures such as Faster R-CNN and FPN, we only compared our model with representative fast detection models and mobile-sized models, e.g., the SSD, SSDLite, and YOLOv3 models. All the lightweight models are listed in Table 4. The first model is SSD300, which is the original version of SSD that accepts a 300 × 300 input image. The next three models are combinations of the MobileNet backbone and an SSD detector, where MobileNetV1 and MobileNetV2 are used as feature extractors for cell detection with a modified resource-efficient version of SSD, named SSDLite [21]. Following that are YOLOv3 and its compressed variant YOLOv3-tiny. We also re-implemented a model that uses MobileNetV2 as a backbone followed by a YOLOv3 detection header. To further validate the feasibility and superiority of our model, we conducted comprehensive experiments in different deployment environments, i.e., a single GPU and a CPU, respectively. We statistically compared the execution times of these detection models.

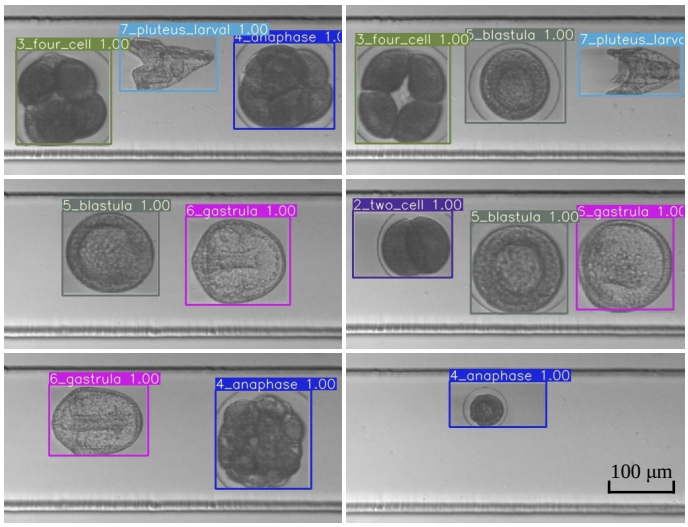

Figure 7. Some qualitative detection results on the self-collected sea urchin embryo dataset.

Table 4 lists the performance of the eight models described above in terms of their mAP, number of parameters, FLOPs, and the inference time t_g on the GPU and t_c on the CPU, respectively. The running times t_g and t_c were measured with batch size one on a single GPU and a single core of the Intel Xeon CPU, respectively. Because different models have different post-processing methods, we only recorded the forward propagation time for one model without additional cost. Subsequently, the two different processing speeds were derived by determining t_g and t_c.

Table 4. Performance results on cell detection.

Network	mAP	#Params	#FLOPs	Inference t_g [1]	fps (GPU)	Inference t_c [2]	fps (CPU)
SSD300	97.26%	24.55 M	30.79 B	5.9 ms	169.5	179.1 ms	5.6
MobileNetV1-SSD	98.41%	7.49 M	1.34 B	7.6 ms	131.6	59.1 ms	16.9
MobileNetV1-SSDLite	96.05%	4.09 M	1.14 B	8.2 ms	122.0	70.3 ms	14.2
MobileNetV2-SSDLite	97.21%	3.16 M	0.67 B	12.6 ms	79.4	92.9 ms	10.8
YOLOv3	98.62%	61.56 M	9.49 B	12.5 ms	80.0	113.3 ms	8.8
YOLOv3-tiny	98.41%	8.64 M	0.80 B	3.0 ms	333.3	26.2 ms	38.2
MobileNetV2-YOLOv3	98.07%	4.78 M	0.46 B	7.4 ms	135.1	52.3 ms	19.1
CellNet (Ours)	**98.70%**	**0.08 M**	**0.10 B**	**2.0 ms**	**500.0**	**11.4 ms**	**87.7**

[1] t_g is the inference time with batch size 1 on a single GPU. [2] t_c is the inference time with batch size 1 on a single core of the Intel Xeon CPU.

5.3.1. Evaluation Criteria

Notably, the mAP over classes with an IoU threshold of 0.5 is an evaluation criterion that was employed to measure the performance of the detection model. This criterion was introduced in the PASCAL VOC Challenge [36]. It first calculates the precision and recall values for each class. Then, the model plots the precision–recall curve and computes the area under the P-R curve, which is formally known as the average precision (AP) value of one category. Finally, we obtain the mAP of all classes across the total *test* dataset. The definitions of precision, recall, AP, and mAP are described as follows:

$$\begin{aligned} precision &= \frac{TP}{TP+FP}, \\ recall &= \frac{TP}{TP+FN}, \\ AP &= \sum_{r=0}^{1}(r_{n+1}-r_n)P_{interp}(r_{n+1}), \\ mAP &= \frac{1}{N}\sum_{i}^{N}AP_i. \end{aligned} \tag{15}$$

When calculating AP, the precision P_{interp} at each recall value r_{n+1} is interpolated by taking the maximum precision measured for a method for which the corresponding recall exceeds r_{n+1}:

$$P_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r}\geq r_{n+1}} p(\tilde{r}), \tag{16}$$

where $p(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

5.3.2. Comprehensive Analysis Compared with SSD Detection Architectures

The results in Table 4 indicate that our CellNet achieved state-of-the-art cell detection performance with 98.7% mAP, even though it only needed 0.08 M parameters and 0.10 B FLOPs. Furthermore, when deployed on a GPU, it operated at 500.0 fps, and on a CPU, it ran at 87.7 fps, thereby demonstrating the effectiveness of CellNet in real-time high-speed cell detection. Compared with SSD300, our CellNet achieved 1.44% better accuracy with 306.9× fewer parameters and was 307.9× more efficient on the FLOPs. It is noteworthy that, to improve the AP of objects with both small and large sizes, SSD employs multi-scale detection strategies to fuse feature maps with multiple resolutions. The inference speed on the GPU was improved by 330.5 fps, from 169.5 fps to 500.0 fps, whereas it was 15.7× faster than SSD300 on the CPU. The inference latencies on different platforms are presented in Figure 8, which shows CellNet on the top left of the two figures. This demonstrates the effectiveness of CellNet with respect to increasing the detection speed.

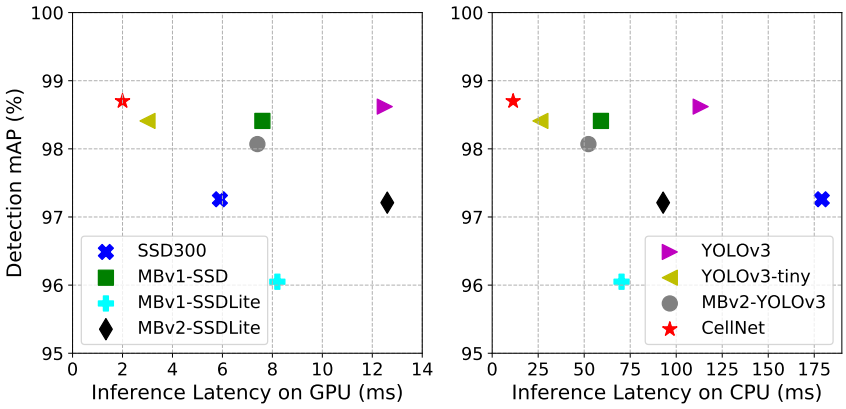

Figure 8. The trade-off between processing speed and detection mAP. This allows comparing models that target different resource-constrained software and hardware platforms. Best viewed in color.

We used the same settings to investigate the differences between the convolutional blocks of the MobileNet series and our efficient CellConv block. These results appear in Table 4, which shows that CellNet improved the mAP by 0.29%, 2.56%, and 1.49%, respectively. Considering the model size and FLOPs, our CellNet significantly surpassed the three MobileNet-SSD architecture-like models, has at least 39.5× fewer parameters (i.e., 3.16 M vs. 0.08 M), and requires 6.7× less computation (i.e., 0.67 B vs. 0.10 B) than the other three models. When measuring the inference speed on different platforms, our CellNet also outperformed these models by 368.4 fps, 378.0 fps, and 420.6 fps on the GPU and 70.8 fps, 73.5 fps, and 76.9 fps on the CPU, respectively. The results suggest that the CellConv block has superior detection performance as a result of its compressed model size and constrained computational cost.

5.3.3. Comprehensive Analysis Compared with YOLOv3 Detection Architectures

As described in Section 4.1.2, similar to YOLOv3, we simply attached a detection header to the final convolutional feature maps. CellNet offers considerable improvements in that it is 769.5× smaller and 94.9× computationally more efficient than YOLOv3. As can be seen in Figure 8, regardless as to whether CellNet ran on the GPU or CPU, it outperformed YOLOv3 by 6.3-times and 10.0-times, respectively. CellNet also achieved a more optimal accuracy/efficiency trade-off than YOLOv3-tiny (98.41% mAP at 333.3 fps on the GPU and 38.2 fps on the CPU). When we replaced the backbone of YOLOv3 with MobileNetV2 (i.e., MobileNetV2-YOLOv3 in Table 4), despite the reduction in the number of parameters and FLOPs, the mAP and processing speed were still lower than those of CellNet by a large margin. Figure 9 summarizes the comparisons between our CellNet and the other efficient state-of-the-art models in terms of FLOPs and model size. To facilitate the comparisons, the different models are drawn in the figure to indicate their relative sizes. As shown in Figure 9, CellNet achieved a higher mAP with an order of magnitude fewer parameters and 4.6× fewer FLOPs than existing efficient models. In general, the given results indicate that CellNet is capable of extracting fine-grained features and providing the location of objects with precision by using less memory and increased processing speed.

5.4. Ablation Study

To demonstrate the effectiveness of the new components of CellNet, we report the results of the CellConv block, combined with other effective modules such as h-swish nonlinearity and residual blocks in Table 5. We also present Figures 10 and 11 for an intuitive comparisons of the impact of the different components on the accuracy–efficiency curve. All ablation models were trained on the *training* set of the cell dataset and tested on the *validation* set. With plain convolutional blocks without other modules (Table 5, first row), the model only achieved 97.41% mAP. In contrast, with CellConv blocks, we obtained a comparable result using 0.05 M fewer parameters and 73.56 M fewer FLOPs, resulting in

an increase of 1.92× in inference speed (Table 5 2nd row). As shown in Figures 10 and 11, the mAP–FLOPs curve and the mAP–latency curve shifted in the negative direction of the x-axis by a large margin when CellConv modules were added, suggesting our CellNet is indeed fast on real hardware. This again proves the efficiency of the CellConv block.

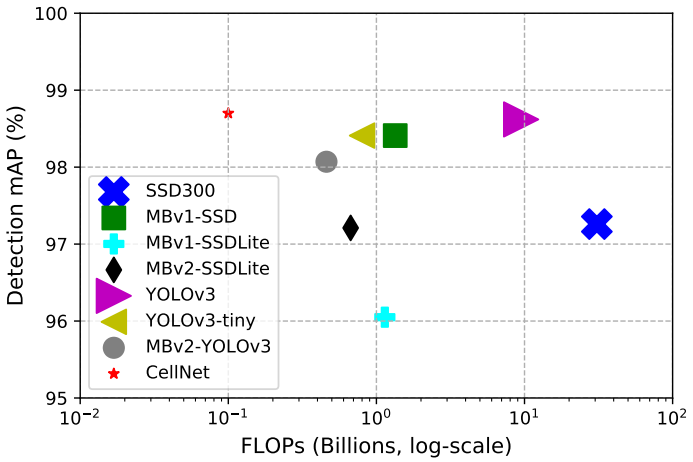

Figure 9. The comparison of FLOPs and detection mAP for different models. The size of different markers reflects the relative size of the corresponding models. We can notice that CellNet is on the top-left of the figure. This demonstrates the effectiveness of CellNet with respect to FLOPs and model size.

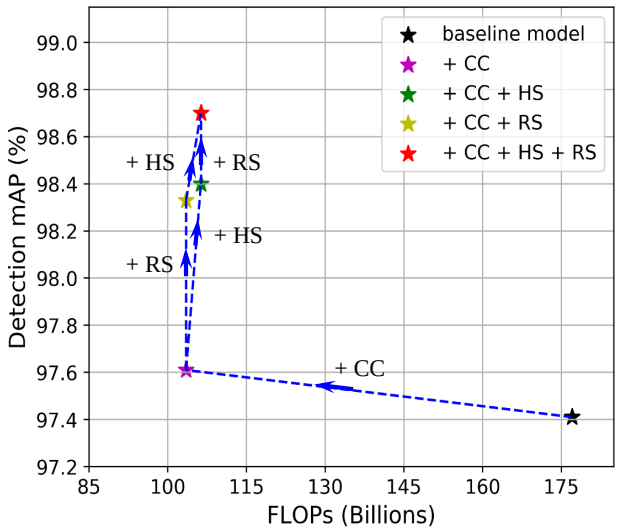

Figure 10. The impact of adding different components to CellNet on the detection mAPs and FLOPs, where CC, RS, and HS denote CellConv blocks, residual connection, and h-swish nonlinearity, respectively.

Table 5. Ablation study on CellNet and its variants.

	CellConv	h-Swish	Residual	mAP	#Params	#FLOPs	t_g [1]
1				97.41%	0.13 M	177.07 M	2.86 ms
2	✓			97.61%	**0.08 M**	**103.51 M**	**1.49 ms**
3	✓		✓	98.33%	**0.08 M**	**103.51 M**	1.56 ms
4	✓	✓		98.40%	**0.08 M**	106.33 M	1.93 ms
5	✓	✓	✓	**98.70%**	**0.08 M**	106.33 M	1.98 ms

[1] t_g is the inference latency with batch size 1 on a single GPU.

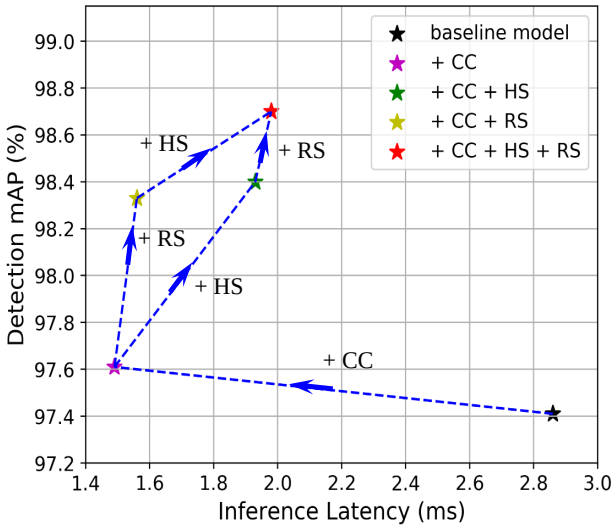

Figure 11. The impact of the different components on the mAP–latency curve. Since our focus is on dedicated parallel computing equipment, we only report the processing time on the GPU. CC, RS, and HS denote CellConv blocks, residual connection, and h-swish nonlinearity, respectively.

Deeper models are known to easily fall into gradient vanishing or exploding, which makes it difficult for the model to reach convergence. In this situation, our 22-layer deep model showed inferior performance on the detection mAP. However, upon the addition of the residual connection we built, as shown in Figure 10 and the third row of Table 5, we achieved an improvement of 0.72% (97.61% vs. 98.33%). This indicates that the residual block is critical to improving the performance when the model goes deeper, which is not surprising.

As mentioned in Section 3.3, h-swish was introduced to increase the nonlinearity of the CellConv block, and subsequently improve the performance of our model, without increasing the FLOPs significantly. We assessed the impact by comparing h-swish with the original ReLU function. The experimental results in Table 5 (fourth row) show that h-swish significantly improves CellNet by 0.79% mAP with an increase in the computational cost of only 2.82 M, which demonstrates the impact of h-swish nonlinearity. However, as shown in Figure 11, an additional 0.44 ms consumption in the inference phase reduces the detection speed when the target scenario is ultra-high-speed object detection, and these delays need to be taken into consideration. It is noteworthy that residual blocks typically need more on-chip RAM for caching intermediate feature maps; however, h-swish does not need to buffer previous features. Thus, the h-swish function can be employed to construct a pipeline computational architecture without consuming expensive on-chip resources.

Finally, the fusion of h-swish with the residual block into one model constitutes our final version of CellNet. As illustrated in Figures 10 and 11, CellNet continuously achieves a state-of-the-art balance between accuracy and efficiency, surpassing the plain model by 1.29% mAP with a 1.44× speedup on the GPU. In addition, slight improvements in terms of the mAP are also obtained: 0.37% for the model that employs h-swish nonlinearity (third row) and 0.30% (fourth row) for the equipment with residual blocks, respectively. These results highlight the importance of the nonlinear function and residual blocks to obtain promising results. Moreover, these two figures show that the two components work in a complementary way to mutually reinforce the overall cell detection performance.

Table 5 combined with these two figures allows us to choose the most appropriate detection model that most optimally balances the trade-off between accuracy and efficiency, especially for resource-constrained platforms with low-capacity memories, a smaller model size, and limited computational elements, yet with strong requirements with respect to low-latency and high-speed image processing (e.g., FPGAs, mobile phones, and self-driving cars).

6. Conclusions

Fast and high-throughput label-free single-cell analysis technology is critical for modern cancer screening. This paper presented a novel real-time high-speed cell detection model for LOC-based cell analysis. The proposed CellNet has a lightweight, yet powerful network architecture, which achieved an excellent accuracy/efficiency trade-off. To reduce the number of parameters and FLOPs, the efficient CellConv block was proposed as an effective alternative to conventionally expensive convolutional layers. In addition, the h-swish nonlinearity function was introduced to further improve the cell detection performance. Finally, we defined a precise multi-task loss to boost the prediction and localization ability of the detection layer. The experiments conducted on our custom sea urchin embryo dataset demonstrated that our CellNet achieved state-of-the-art performance with an order of magnitude fewer parameters and 4.6× fewer FLOPs than existing efficient mobile-sized models. We verified the efficiency of CellNet by evaluating the cell detection speed on different platforms (500.0 fps on a single GPU and 87.7 fps on a CPU).

In the future, we would like to improve the current study in two aspects. (1) As our CellNet is a compact model with a regular structure, we plan to actually deploy CellNet on FPGAs and other resource-constrained platforms. This would enable our detection model, in combination with an LOC-based microchannel, to realize real-time fast-flowing cell analysis and sorting without requiring computationally expensive GPUs. Thus, automatic vision-guided cell manipulation could be realized in an environment with a more complex background. (2) We aim to continue extending our work to high-speed general object detection, face detection, and other applications.

Author Contributions: Conceptualization, Q.G. and I.I.; methodology, Q.G. and X.L.; software, X.L.; validation, Q.G. and X.L.; formal analysis, X.L.; investigation, X.L.; resources, I.I. and Q.G.; data curation, Q.G.; writing—original draft preparation, X.L.; writing—review and editing, Q.G. and I.I.; visualization, X.L.; supervision, Q.G. and I.I.; project administration, Q.G.; funding acquisition, Q.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Scientific Instrument Developing Project of the Chinese Academy of Sciences under Grant YJKYYQ20200045.

Data Availability Statement: Data is available on request to the corresponding author.

Acknowledgments: We thank the anonymous Reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

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

References

1. Ferlin, M.A.; Grochowski, M.; Kwasigroch, A.; Mikołajczyk, A.; Szurowska, E.; Grzywińska, M.; Sabisz, A. A Comprehensive Analysis of Deep Neural-Based Cerebral Microbleeds Detection System. *Electronics* **2021**, *10*, 2208. [CrossRef]
2. Mahum, R.; Rehman, S.U.; Okon, O.D.; Alabrah, A.; Meraj, T.; Rauf, H.T. A Novel Hybrid Approach Based on Deep CNN to Detect Glaucoma Using Fundus Imaging. *Electronics* **2022**, *11*, 26. [CrossRef]
3. Guan, Z.; Zhao, P.; Wang, X.; Wang, G. Modeling Radio-Frequency Devices Based on Deep Learning Technique. *Electronics* **2021**, *10*, 1710. [CrossRef]
4. Veta, M.; Pluim, J.P.W.; van Diest, P.J.; Viergever, M.A. Breast cancer histopathology image analysis: A review. *IEEE Trans. Biomed. Eng.* **2014**, *61*, 1400–1411. [CrossRef] [PubMed]
5. Nitta, N.; Sugimura, T.; Isozaki, A.; Mikami, H.; Hiraki, K.; Sakuma, S.; Iino, T.; Arai, F.; Endo, T.; Fujiwaki, Y.; et al. Intelligent image-activated cell sorting. *Cell* **2018**, *175*, 266–276. [CrossRef] [PubMed]
6. Heo, Y.J.; Lee, D.; Kang, J.; Lee, K.; Chung, W.K. Real-time image processing for microscopy-based label-free imaging flow cytometry in a microfluidic chip. *Sci. Rep.* **2017**, *7*, 11651. [CrossRef] [PubMed]
7. Nolte, M.A.; Kraal, G.; Mebius, R.E. Effects of fluorescent and nonfluorescent tracing methods on lymphocyte migration in vivo. *J. Int. Soc. Anal. Cytol.* **2004**, *61*, 35–44. [CrossRef]
8. Gu, Q.; Kawahara, T.; Aoyama, T.; Takaki, T.; Ishii, I.; Takemoto, A.; Sakamoto, N. LOC-Based high-throughput cell morphology analysis system. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 1346–1356. [CrossRef]
9. Lee, K.-M.; Li, Q.; Daley, W. Effects of classification methods on color-based feature detection with food processing applications. *IEEE Trans. Autom. Sci. Eng.* **2007**, *4*, 40–51. [CrossRef]

10. Yang, H.; Zheng, S.; Lu, J.; Yin, Z. Polygon-invariant generalized Hough transform for high-speed vision-based positioning. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1367–1384. [CrossRef]
11. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–25 June 2005; pp. 886–893.
12. Li, Z.; Ma, L.; Long, X.; Chen, Y.; Deng, H.; Yan, F.; Gu, Q. Hardware-Oriented Algorithm for High-Speed Laser Centerline Extraction Based on Hessian Matrix. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–14. [CrossRef]
13. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
14. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
16. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
17. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2016; pp. 21–37.
18. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
19. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. 2018. Available online: https://arxiv.gg363.site/abs/1804.02767 (accessed on 19 March 2020).
20. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. Available online: https://arxiv.gg363.site/abs/1704.04861 (accessed on 24 March 2020).
21. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
22. Ojala, T.; Pietikinen, M.; Harwood, D. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognit.* **1996**, *29*, 51–59. [CrossRef]
23. Al-Kofahi, Y.; Lassoued, W.; Lee, W.; Roysam, B. Improved automatic detection and segmentation of cell nuclei in histopathology images. *IEEE Trans. Biomed. Eng.* **2010**, *57*, 741–852. [CrossRef] [PubMed]
24. Cosatto, E.; Miller, M.; Graf, H.P.; Meyer, J.S. Grading Nuclear Pleomorphism on Histological Micrographs. In Proceedings of the IEEE International Conference on Pattern Recognition (ICPR), Tampa, FL, USA, 8–11 December 2008; pp. 1–4.
25. Ali, R.; Gooding, M.; Szilágyi, T.; Vojnovic, B.; Christlieb, M.; Brady, M. Automatic segmentation of adherent biological cell boundaries and nuclei from brightfield microscopy images. *Mach. Vis. Appl.* **2012**, *23*, 607–621. [CrossRef]
26. Gu, Q.; Aoyama, T.; Takaki, T.; Ishii, I. Simultaneous vision-based shape and motion analysis of cells fast-flowing in a microchannel. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 204–215. [CrossRef]
27. Vink, J.P.; Van Leeuwen, M.B.; Van Deurzen, C.H.M.; De Haan, G. Efficient nucleus detector in histopathology images. *J. Microsc.* **2013**, *249*, 124–135. [CrossRef]
28. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
29. Xu, J.; Xiang, L.; Liu, Q.; Gilmore, H.; Wu, J.; Tang, J.; Madabhushi, A. Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images. *IEEE Trans. Med. Imaging* **2016**, *35*, 119–130. [CrossRef]
30. Cireşan, D.; Giusti, A.; Gambardella, L.M.; Schmidhuber, J. Mitosis detection in breast cancer histology images with deep neural networks. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), Granada, Spain, 16–20 September 2013; pp. 411–418.
31. Xie, Y.; Xing, F.; Shi, X.; Kong, X.; Su, H.; Yang, L. Efficient and robust cell detection: A structured regression approach. *Med. Imaging Anal.* **2018**, *44*, 245–254. [CrossRef]
32. Xue, Y.; Ray, N. Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing. 2017. Available online: https://arxiv.org/abs/1708.03307 (accessed on 5 April 2020).
33. Dong, B.; Shao, L.; Costa, M.D.; Bandmann, O.; Frangi, A.F. Deep learning for automatic cell detection in wide-field microscopy zebrafish images. In Proceedings of the IEEE 12th International Symposium on Biomedical Imaging, Brooklyn, NY, USA, 16–19 April 2015; pp. 772–776.
34. Sirinukunwattana, K.; Raza, S.; Tsang, Y.; Snead, D.; Cree, I.; Rajpoot, N. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Trans. Med. Imaging* **2016**, *35*, 1196–1206. [CrossRef]
35. Lin, T.-Y.; Lin, T.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; et al. Microsoft COCO: Common Objects in Context. 2015. Available online: https://arxiv.org/abs/1405.0312 (accessed on 12 April 2020).
36. Visual Object Classes Challenge 2012 (VOC2012). 23 October 2016. Available online: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/ (accessed on 19 April 2020).

37. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal loss for dense object detection. In Proceedings of the Computer Vision and Pattern Recognition (ICCV), Venice, Italy, 30 October–1 November 2017; pp. 2980–2988.
38. Yang, H.; Chen, Y.; Song, K.; Yin, Z. Multiscale feature-clustering-based fully convolutional autoencoder for fast accurate visual inspection of texture surface defects. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1450–1467. [CrossRef]
39. Long, X.; Hu, S.; Hu, Y.; Gu, Q.; Ishii, I. An FPGA-Based Ultra-High-Speed Object Detection Algorithm with Multi-Frame Information Fusion. *Sensors* **2019**, *19*, 3707. [CrossRef] [PubMed]
40. Li, J.; Long, X.; Xu, D.; Gu, Q.; Ishii, I. An Ultrahigh-Speed Object Detection Method with Projection-Based Position Compensation. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 4796–4806. [CrossRef]
41. Xie, Q.; Li, D.; Xu, J.; Yu, Z.; Wang, J. Automatic detection and classification of sewer defects via hierarchical deep learning. *IEEE Trans. Autom. Sci. Eng.* **2019**, *16*, 1836–1847. [CrossRef]
42. Iandola, F.; Han, S.; Moskewicz, M.; Ashraf, K.; Dally, W.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and <0.5 MB Model Size. 2016. Available online: https://arxiv.org/pdf/1602.07360v3.pdf (accessed on 1 May 2020).
43. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
44. Ramachandran, P.; Zoph, B.; Le, Q.V. Swish: A Self-Gated Activation Function. 2017. Available online: https://arxiv.org/abs/1710.05941v1 (accessed on 3 May 2020).
45. Howard, A.; Sandler, M.; Chu, G.; Chen, L.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for Mobilenetv3. May 2019. Available online: https://arxiv.gg363.site/abs/1905.02244 (accessed on 9 May 2020).
46. Gu, Q.; Aoyama, T.; Takaki, T.; Ishii, I. Rapid vision-based shape and motion analysis system for fast-flowing cells in a microchannel. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 5848–5853.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.