Engineering Journal issn: 0125-8281

Article

Smart Microscopy Camera Kit: Automatic Counting of Blood Cells in Peripheral Blood Smear Images Using RetinaNet on Raspberry Pi CM3+

Natthakorn Kasamsumran[1,a], **Amornthep Phunsin**[2,b], **Suree Pumrin**[1,c,*], **and Wanchalerm Pora**[1,d]

1 Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand
2 Q-Wave Systems Co., Ltd., Bangkok 10140, Thailand
E-mail: [a]natthakorn.engr@gmail.com, [b]amornthep@qwavesys.com, [c,*]suree.p@chula.ac.th (corresponding author), [d]wanchalerm.p@chula.ac.th

Abstract. Microscopic examination of peripheral blood smear images for blood cell counting remains a critical yet labor-intensive task in clinical diagnostics. This research presents MicrosisDCN, an intelligent microscopy camera system designed to automate blood cell detection and counting, powered by a compact embedded platform based on the Raspberry Pi Compute Module 3+. The system incorporates a 5-megapixel image sensor and a versatile eyepiece fitting that is compatible with the most compound microscopes, providing a portable, cost-effective, and user-friendly solution. Calibration procedures ensure alignment with traditional high-power field (HPF) standards, allowing cell counts to be reported in standard mitotic count units. To detect red blood cells, white blood cells, and platelets in real-time, the system uses a special version of a deep learning model called RetinaNet, which has been improved with a technique called auto-anchor parameterization. MicrosisDCN achieves a mean Average Precision (mAP) of 86.81% in detecting a few types of blood cells with minimal errors: 1.06% for red blood cells, 0.06% for white blood cells, and 4.23% for platelets. The results indicate that MicrosisDCN, which combines traditional microscopy with advanced vision technologies, serves as an efficient, practical, and scalable solution for clinical and medical laboratory applications.

Keywords: Microscopy imaging, peripheral blood smear, object detection, Raspberry Pi.

ENGINEERING JOURNAL Volume 29 Issue 6
Received 10 June 2024
Accepted 26 May 2025
Published 30 June 2025
Online at https://engj.org/
DOI:10.4186/ej.2025.29.6.43

DOI:10.4186/ej.2025.29.6.43

1. Introduction

The diagnosis of infectious diseases in both humans and animals often relies on the analysis of biological specimens such as blood, urine, stool, and sputum. Among these, blood analysis—specifically the complete blood count (CBC)—is one of the most commonly utilized methods in clinical laboratories. The CBC provides essential quantitative data on various blood components, including red blood cells (RBCs), white blood cells (WBCs), and platelets (thrombocytes). This diagnostic tool plays a crucial role in identifying infections, monitoring immune responses, and assessing overall health status.

A substantial proportion of infection diagnoses—estimated at up to 70% of all laboratory investigations [1]—relies on the examination of peripheral blood smears under a microscope. This technique enables detailed visualization of individual cells and abnormalities in their morphology or count, providing critical insights for clinical assessment. The microscope, a cornerstone of medical diagnostics, utilizes combinations of eyepiece and objective lenses, offering magnifications typically ranging from 40× to 1000× [2], thereby facilitating the observation of microscopic structures with high resolution and clarity.

Although widely used, microscopic examination is inherently labor-intensive and time-consuming, often requiring technicians to manually observe blood smears for extended periods. This repetitive and tedious process can lead to visual fatigue and increase the likelihood of human error, ultimately affecting diagnostic accuracy. To overcome these limitations, this research introduces the development of an automated smart camera system, MicrosisDCN, designed to detect and count blood cells autonomously. By minimizing the need for prolonged manual inspection, this innovation aims to significantly reduce analysis time while enhancing diagnostic efficiency.

MicrosisDCN (see Fig. 1 and Fig. 2) is an intelligent camera system designed to interface seamlessly with conventional microscope eyepieces. It leverages a deep convolutional neural network (DCNN) to perform automatic classification and counting of blood cells. The system is built around the EagleEYE Smart Camera (EY-PRO-32), which features an embedded Raspberry Pi Compute Module 3+ (CM3+) developed by Q-Wave Systems. The accompanying development kit includes custom circuit boards with an integrated power supply and a 5-megapixel image sensor, optimized for efficient image acquisition and real-time neural network processing.

To ensure seamless integration with a wide range of microscopes, the camera is designed to be compatible with standard eyepiece tube diameters of 23.2 mm, 30 mm, and 30.5 mm. Tailored for machine vision applications, the device includes all essential electronic circuits required for research-grade performance. Its robust design provides operational stability, making it suitable for extended laboratory use and precise biological analyses.

The software component of the system integrates image processing and classification algorithms implemented in Python. The OpenCV library [3] is utilized for handling blood cell images, while TensorFlow [4] and Keras [5] are employed for developing and training the neural networks. To enhance processing capabilities, the system leverages multi-GPU processing. A mean Average Precision (mAP) of 80% is set as the minimum performance benchmark to ensure the system's reliability and accuracy in real-world diagnostic applications.

Fig. 1. 3D CAD design concept of the MicrosisDCN smart camera kit for integration with compound microscopes.

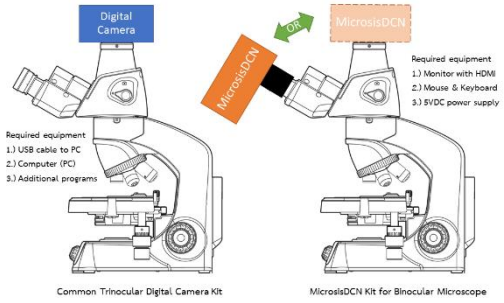

Fig. 2. Comparison between a standard trinocular microscope with a digital camera attachment (left) and the MicrosisDCN smart camera kit, which supports both binocular and trinocular microscope attachments (right).

2. Methodology and Methods

2.1. Convolution Neural Network

This section describes the types of neural networks utilized in this research. A convolutional neural network (CNN) [6] simulates human vision by dividing the visual space into small, manageable parts. These areas are analyzed and combined to determine what is visible in the scene. The network extracts features from these subspaces, such as color borders of images that contrast between objects. Just as humans recognize contrasting colors by focusing on specific regions and their surrounding context, CNNs identify objects by applying mathematical operations, including spatial convolution and image processing, to the photographs.

The process begins with the configuration of a kernel designed to extract object-recognition features. This involves applying filters to the first pixel of an image. The filter is then moved across the entire image, where it is applied to every pixel. Max pooling is then used to identify the maximum value within the area covered by the filter. Essentially, the filter scans the image, applies the selected operation, and the highest value in each area becomes the result. This method is repeated throughout the image, with the filter moving in steps, also known as strides, across the image grid. Each stride reduces the size of the feature map. The process ultimately helps train the neural network by adjusting weights over several iterations to improve accuracy and reduce loss.

Feature extraction occurs in sub-areas of the image, such as contours that intersect. Filters help classify and extract features based on differences in pixel values, allowing the CNN to discern relevant characteristics. To increase the breadth of features, multiple filters can be employed, and this process is performed on large datasets to ensure a comprehensive feature set. Image filters are typically two-dimensional, depending on the area being analyzed. For instance, to detect diagonal black lines in an image, a 3×3 filter can be applied. The center of the filter represents the aggregation of data from the image pixels, known as anchors. When applied to a 3×3 pixel section, the filter combines the pixel values and computes the resulting feature map. The filter then slides over the entire image, creating a feature map based on the combined data from all covered pixels.

The sliding process is referred to as the stride, determining the number of steps the convolution filter moves. Typically, strides are set to one, but if the stride is increased to two, the feature map size will shrink. Padding is sometimes added to the image's edges, usually with zero values or other constants, to maintain consistent feature map dimensions. This ensures that convolution operations can cover the entire image without dimension loss. In practice, the convolution operates in three dimensions, considering not only the height and width of the image but also its depth, which accounts for color channels like red, green, and blue. The filter size—whether 3×3 or 5×5—also influences the convolution operation's design.

Once the convolution is complete, dimensionality reduction is necessary to decrease the number of variables. This step reduces the model's training time and helps prevent overfitting. It also allows the neural network to function with fewer layers by reducing the size of each layer while maintaining the same depth. Pooling methods, such as max pooling and mean pooling, are commonly used for dimensionality reduction. Max pooling involves identifying the highest value within a filter's area, using this maximum value as the feature map's result. As the filter moves over the image, the pooling operation continually extracts relevant features, reducing the image's spatial dimensions. For example, with a 2×2 max pooling filter, the feature map will be reduced from 4×4 to 2×2, preserving only the most significant values, with the depth unchanged.

2.2. Object Detection Using CNNs

Neural networks used for object recognition enable computers to identify, process, and interpret objects from images, closely mimicking the human brain's visual processing capabilities, also known as computer vision. In computer vision, the task is divided into four key areas:

- *Object classification* [7], which involves categorizing an image based on its content, typically identifying the overall object class.
- *Classification and localization* [8], which involves categorizing the objects in the image and pinpointing their exact locations with the help of bounding boxes.
- *Object detection* [9], which identifies multiple object classes within an image and specifies the locations of each class, utilizing bounding boxes to mark each object's position.
- *Image segmentation* [10], which identifies various classes of objects and accurately locates them using either polygonal or curve-boundaries.

The Large Scale Visual Recognition Challenge 2015 (ILSVRC2015) [11] served as a critical test platform for neural network models, where researchers competed using a dataset of over 1.4 million images, divided into 1,000 object classes. This competition fostered advancements in object detection and recognition, with models like VGGNet [12], R-CNN [13], Fast-RCNN [14], Faster-RCNN [15], GoogleNet (Inception) [16], ResNet [17], and RetinaNet [18] making significant contributions.

In the landmark study on deep residual learning for image recognition [17], the authors addressed the vanishing gradient problem, which occurs when the gradient of the loss function approaches zero, hindering further training. This problem often arises in very deep neural networks. To combat this, a shortcut connection was introduced within the network architecture, allowing the gradients to propagate more effectively. By replacing traditional activation functions like Sigmoid with ReLU, residual networks (ResNet) were able to maintain performance even with greater depth.

Further advancements were made with the introduction of focal loss in dense object detection [18], which enhanced the ResNet model by addressing class imbalance issues, a common problem in object detection tasks. Focal loss modifies the cross-entropy loss function to focus on hard-to-detect examples, thereby improving the accuracy of predictions for underrepresented classes. Researchers combined ResNet with feature pyramid networks (FPN) and focal loss, creating a more effective object detection framework.

The Keras RetinaNet [23] library, released by researchers, allows developers to train and improve RetinaNet models in Keras using Jupyter Notebooks [19], an interactive Python environment for development. The

DOI:10.4186/ej.2025.29.6.43

Table 1. The comparison is based on the average precision (AP) of the Faster R-CNN, Faster R-CNN with FPN, Faster R-CNN with TDM, YOLOv2, SSD 513, and RetinaNet models [18].

Model	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN with FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN with TDM	Inception-ResNet	36.8	55.7	39.2	16.2	39.8	52.1
YOLOv2	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
RetinaNet	ResNet-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

ability to train these models in Jupyter makes it easier to experiment with various configurations and fine-tune the models for specific applications.

When evaluating the performance of object detection models, the intersection over union (IoU) metric plays a crucial role. IoU measures the overlap between the predicted bounding box and the ground truth box. The average precision (AP) metric, which is calculated at various IoU thresholds (e.g., 0.50, 0.75, and the range between 0.50 and 0.95), provides an overall measure of a model's accuracy [20]. The PASCAL VOC metric [21] focuses on specific thresholds like IoU = 0.50 (AP50) and IoU = 0.75 (AP75), which are commonly used to evaluate the effectiveness of object detection algorithms. Furthermore, the AP values for small, medium, and large objects (APS, APM, and APL) provide insights into the model's ability to detect objects of various sizes.

Based on the research findings, RetinaNet [18] outperformed Faster R-CNN [14] in terms of accuracy, which led to its selection for further work. The researcher chose RetinaNet as the backbone for developing a neural network model within the MicrosisDCN camera system. Using transfer learning, the model is trained on a new, untrained image dataset to accurately classify blood cell images captured by a microscope. The trained model will be embedded into the EagleEYE Smart Camera, which will leverage libraries such as OpenCV [3], TensorFlow [4], and PiCamera [22] for image processing and camera integration.

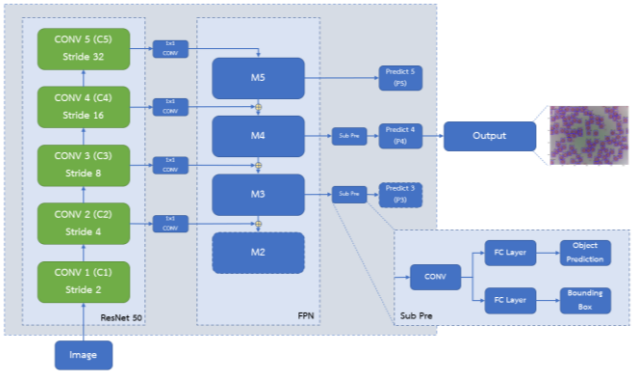

Fig. 3. Architecture of RetinaNet with a Feature Pyramid Network (FPN) using a ResNet-50 backbone [18].

2.3. RetinaNet Architecture

Facebook AI Research introduced RetinaNet, a one-stage object detection model utilizing focal loss to address the issue of class imbalance and the overwhelming presence of easy negative samples during training [18]. This design improves prediction accuracy, particularly in scenarios with numerous background objects. RetinaNet employs ResNet and Feature Pyramid Network (FPN) as its core architectural components for feature extraction, alongside two task-specific subnetworks for classification and bounding box regression. It outperforms two-stage detectors such as Fast R-CNN [14] and Faster R-CNN [15] in terms of accuracy and efficiency.

The backbone network of RetinaNet (Fig. 3) is typically a pre-trained Convolutional Neural Network (CNN) from the ResNet family, such as ResNet50 or ResNet101. Traditional CNNs pass outputs sequentially from one layer to the next. In contrast, ResNet introduces shortcut (residual) connections, enabling information to bypass intermediate layers. These residual links improve gradient flow during training, making deeper networks easier to optimize and less prone to accuracy degradation. The ResNet backbone comprises multiple stages, each generating feature maps at different spatial resolutions—critical for detecting objects at various scales.

To enhance multi-scale feature representation, RetinaNet integrates a Feature Pyramid Network (FPN) into the backbone. The FPN constructs a rich, multi-resolution feature pyramid by merging semantically strong but low-resolution features from deeper layers with high-resolution features from earlier layers. This fusion enables robust object detection across different object sizes. At each level of the FPN, RetinaNet generates a set of anchors—predefined bounding boxes with various aspect ratios and scales—that tile the image. These anchors serve as reference boxes for predicting object locations.

Each level of the FPN feeds into two separate subnetworks, also known as heads: one for classification and another for bounding box regression.

- The classification subnet estimates the likelihood of object presence for each of the A anchors and K object classes at every spatial location. It applies four 3×3 convolutional layers with 256 filters each, followed by ReLU activations. A final 3×3 convolutional layer with K×A filters outputs the class scores. A sigmoid function is used to produce binary predictions per class. The focal loss is employed here to reduce the impact of well-classified examples and focus learning on hard, misclassified

ones, effectively mitigating the class imbalance problem.

- The regression subnet is architecturally identical to the classification subnet but outputs bounding box coordinates. Specifically, it predicts four values per anchor, corresponding to the adjustments needed to refine the anchor box into a tightly fitting bounding box around the detected object.

These subnetworks are shared across all levels of the FPN, enhancing efficiency and consistency. By combining deep residual learning, multi-scale feature aggregation, and focal loss, RetinaNet achieves state-of-the-art results in one-stage object detection.

We implemented RetinaNet using a Keras-based framework [23], incorporating anchor optimization techniques [24]. For our experiments, we selected ResNet50 as the backbone due to its strong performance and suitability for transfer learning. An FPN was constructed atop the backbone, and during training, we used pretrained weights while freezing the backbone layers to retain general-purpose features and accelerate convergence.

2.4. Metrics Evaluation

The success of this study is defined by the final model achieving a minimum mean Average Precision (mAP) of 80% across the validation datasets. To thoroughly evaluate the performance of the proposed model, we computed a comprehensive set of key metrics, including precision, recall, mean Average Precision (mAP), mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R^2 or R-square). These metrics were chosen to evaluate both the classification performance and the quantitative prediction accuracy of the model.

The following sections provide detailed explanations of each metric, including their calculation and interpretation.

Precision, also referred to as positive predictive value (PPV), measures the model's ability to correctly identify instances of a particular class. It quantifies how many of the predicted positive instances are actually correct. Precision is calculated using Eq. (1) as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{PP}} = \frac{\text{TP}}{\text{TP+FP}} \quad (1)$$

where TP (true positives) and FP (false positives) represent the number of objects correctly and incorrectly recognized by the model as belonging to the target class, respectively. The sum of TP and FP, denoted as PP (Predicted Positives), reflects the total number of objects predicted to belong to that class.

Recall, also known as the true positive rate (TPR) or sensitivity, measures the model's ability to correctly identify all relevant instances of a particular class. In other words, it quantifies how many actual positives were successfully detected by the model. Recall is computed using Eq. (2) as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{P}} = \frac{\text{TP}}{\text{TP+FN}} \quad (2)$$

where P indicates the number of items in an interested class, and FN represents the number of objects that the model does not recognize as belonging to that class.

The average precision of class X, AP^x, is calculated by averaging precision, P^x, over the recall domain, R^x, with $R_x \in [0,1]$. Its value is computed using the following formula from (3):

$$AP^x = \int_0^1 P^x(R^x) dR^x \quad (3)$$

For class X, if we test the model only N times, we will have a set of N pairs of $(R_n^x, P_n^x), n = 0, \ldots, N-1$ and we can assume that $R_n^x \geq R_{n-1}^x$. Using the following method, we can estimate the average precision of class X:

$$AP^x \approx \sum_{n=1}^{N-1} (R_n^x - R_{n-1}^x) P_n^x \quad (4)$$

Assume we have M classes of items, including the one(s) in which we are not interested. The metric known as mean average precision (mAP) can be determined using the formula in (5) as follows:

$$mAP = \frac{1}{M} \sum_{x=0}^{M-1} AP^x \quad (5)$$

mAP provides a single numerical representation that captures the model's overall performance across several classes and instances of objects.

The mAP at IoU=0.5 (or 0.95) represents the mean average precision calculated at a given threshold. The IoU is a quantitative measure of object detector accuracy. A correct detection covers the ground truth bounding box by at least 50% (95%). The study enabled an IoU of 0.5 to accurately predict a certain cell type.

This is due to comparing the image prediction results using our model with the solutions counted cell by cell, called the test set. We count the total number of RBCs, WBCs, and platelets in each image, determining the actual count for the test set images as a_i, and the predicted count for the predicted ones as p_i. We will explain how we calculated the following metrics:

Mean Absolute Error (MAE) is a statistical measure that calculates the average magnitude of the difference between the actual and projected values in a dataset. It calculates the mean of the differences between the observed values and the predicted values in the dataset.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |p_i - a_i| \quad (6)$$

Root Mean Squared Error (RMSE) is the square root of the Mean Squared Error (MSE). It calculates the standard deviation of the differences between observed values and predicted values. The average of the squared

differences between the original and projected values in a given data collection calculates the MSE. The metric quantifies the residual dispersion.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2 \quad (7)$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2} \quad (8)$$

Additionally, we can use the coefficient of determination (R-square) to determine the accuracy of the proportion between the variance of the enumerated data and the variance of the total data. The R-square value is close to 1, indicating that the neural network model has high accuracy and a low error value.

$$R^2 = \frac{(\sum_{i=1}^{n}(p_i \cdot a_i))^2}{(\sum_{i=1}^{n} a_i^2)\cdot(\sum_{i=1}^{n} p_i^2)} \quad (9)$$

3. Camera Kit Implementation

The core idea of this study was to integrate imaging and control systems into a unified camera unit. To achieve this, we leveraged the Raspberry Pi—a compact computer board equipped with a CPU, GPU, and RAM, capable of connecting to a display screen, keyboard, and mouse—as an embedded system suitable for development and neural network execution. For our specific needs, we selected and customized the EagleEYE Smart Camera (EY-PRO-32) by Q-Wave Systems Co., Ltd., a commercial vision solution designed for industrial applications.

The EagleEYE Smart Camera (EY-PRO-32) (Fig. 4 and 5) is an industrial-grade machine vision camera that operates on a Linux Real-Time (RT) operating system. It contains an embedded Raspberry Pi Compute Module 3+ with a supporting circuit board, a stable power supply, and an OV5647 image sensor offering a resolution of 5 megapixels.

Fig. 4. Specifications of the EagleEYE Smart Camera (EY-PRO-32).

We chose this camera setup because it complemented our existing C-mount lens system (0.5X magnification), which supports eyepiece sizes of 23.2 mm, 30 mm, and 30.5 mm. The camera unit includes essential hardware interfaces for research and has been modified for compatibility with compound microscopes. Switching from a 32GB SD card to a 32GB eMMC significantly improved system stability during extended operation.

To enable the system to support neural network inference, we customized the Raspberry Pi OS (32-bit), also known as Raspbian, with a desktop environment and necessary packages such as OpenCV, TensorFlow, and PiCamera. This customized image—MicrosisDCN V.1.0e—was tailored to work on Raspberry Pi Model B 3, Model B 3+, and Raspberry Pi 4 (with at least 1 GB RAM). Internet connectivity is achieved through the built-in LAN port, and USB ports were configured to support Wi-Fi adapters and allow control via wireless mouse and keyboard on most models.

Assembly of the MicrosisDCN smart camera kit into the compound microscope follows a straightforward process, as illustrated in Figs. 6, 7, 8 and 9.

Fig. 5. The EagleEYE Smart Camera is modified to become the MicrosisDCN smart camera and installing an operating system and our main program.

Fig. 6. Assembly process: First, prepare the MicrosisDCN smart camera kit by inspecting component conditions and removing the dust protection cap from the eyepiece lens. If the eyepiece tube diameter is 23.2 mm, the kit can be attached directly.

Fig. 7. Using a screwdriver, loosen the screw securing the eyepiece and remove the eyepiece from the microscope.

Fig. 8. Securely attach the MicrosisDCN smart camera kit to the microscope. Then, use a screwdriver to tighten the screws sufficiently to prevent any movement of the attached camera kit.

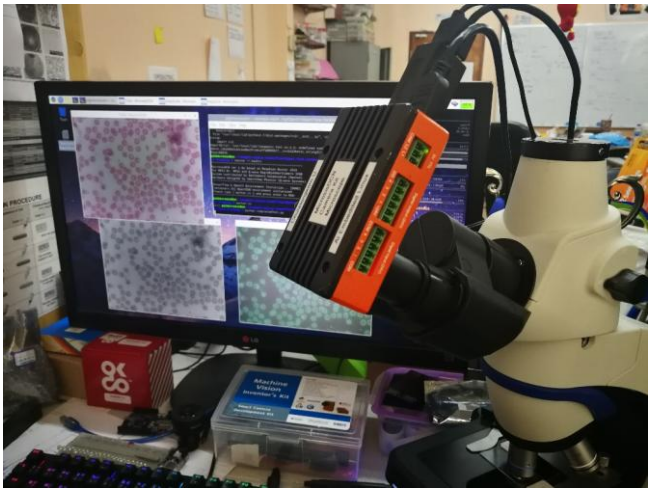

Fig. 9. Connect the HDMI cable, wireless keyboard and mouse set, and LAN cable or Wi-Fi card to establish an internet connection. The system will automatically boot up upon completion of these connections.

Once the neural network model is trained on a host computer, it is deployed onto the MicrosisDCN camera unit. The unit attaches directly to the eyepiece tube of the microscope. Users can operate the model via terminal commands within the pre-configured virtual environment. For instance, running the Python script for inference can be done using a command like

python blood_detection.py

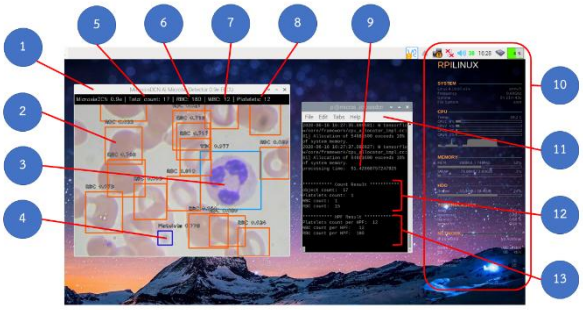

Fig. 10. Python GUI interface of the MicrosisDCN smart camera kit, enabling access to the deep learning model for counting small cells, including RBCs, WBCs, and platelets.

The visualization of the inference results is shown in Fig. 10, which illustrates the integrated display interface of the MicrosisDCN smart camera system. The screen presents both real-time detection outcomes and system performance metrics. The components displayed are as follows:

1. *Display Window*—Shows the original microscope image with localized, countable cells using the OpenCV library.
2. *Red Blood Cell Detection*—The neural network detects and localizes red blood cells (RBCs), showing their class name and corresponding mAP value.
3. *White Blood Cell Detection*—White blood cells (WBCs) are similarly localized with class labels and their respective mAP scores.
4. *Platelet Detection*—Platelets are detected, annotated, and presented with their class name and mAP value.
5. *Total Cell Count*—Displays the total number of microscopic cells detected and counted in the full image frame.
6. *RBC Count per HPF*—Indicates the number of red blood cells counted within the standard field of view (HPF) of the camera.
7. *WBC Count per HPF*—Shows the number of white blood cells counted in the same field of view.
8. *Platelet Count per HPF*—Displays the platelet count within the camera unit's standard HPF area.
9. *Processing Time*—Reports the total inference and processing duration from model execution to final cell count, in seconds.
10. *System Resource Toolbar*—Shows the camera unit's system status, including CPU usage, available RAM, and ongoing processes.
11. *Terminal Window*—Displays the Raspbian OS terminal running within the virtual environment, where the neural network model is executed.
12. *Actual Cell Count (Ground Truth)*—Presents the manually validated number of cells in the image, separated by class (RBC, WBC, Platelet).

13. *Predicted Cell Count per HPF*—Reports the number of cells predicted by the system within the camera's standard HPF area, classified into RBCs, WBCs, and platelets, following the method in [25, 26].

This comprehensive output allows for easy visual verification of detection performance and supports quantitative comparisons between predicted and actual cell counts. By combining image annotation, neural network inference, and system monitoring into a single interface, the MicrosisDCN kit offers a complete and efficient tool for microscopic blood analysis.

4. High-Power Fields in Digital Microscopy for Mitotic Count

Mitotic count is a widely used method for assessing cell proliferation and identifying abnormal cell populations in histopathology. The count is typically reported as the number of mitotic figures per high-power field (HPF), where an HPF refers to the field of view (FoV) under a microscope at a standardized total magnification of 400×—usually achieved by combining a 10× eyepiece with a 40× objective lens. The concept of HPF plays a crucial role in ensuring consistency across observers and imaging systems.

4.1. Total Magnification

In optical microscopy, total magnification, denoted m_T, is calculated as the product of the magnifications of the objective lens m_o and the eyepiece lens m_e:

$$m_T = m_o \times m_e \tag{10}$$

For instance, using a 40× objective and a 10× eyepiece yields m_T = 400×, which satisfies the high-power field condition commonly used in biological studies.

In digital microscopy, magnification does not stop at the eyepiece or objective. Instead, the image formed by the objective is relayed through additional optics—such as a projection lens or adapter—before reaching the camera sensor. This additional scaling is represented by the adapter magnification m_a, leading to the total camera magnification:

$$m_C = m_o \times m_a \tag{11}$$

For example, using a 40× objective with a 0.5× adapter results in m_C = 20×.

Unlike optical microscopy, where the human eye perceives magnification, digital microscopy relies on sensor-captured image dimensions. Therefore, accurate calibration of the digital field of view and understanding of magnification scaling are essential for quantitative tasks such as mitotic index estimation.

4.2. Field of View

The field of view (FoV) represents the observable area through a microscope. In optical microscopy, the FoV is typically circular and depends on the field number (FN) of the eyepiece and the magnification of the objective lens:

$$D_{FV} = \frac{FN}{m_o} \tag{11}$$

$$A_{FV} = \pi \left(\frac{D_{FV}}{2}\right)^2 = \pi \left(\frac{FN}{2m_o}\right)^2 \tag{12}$$

Table 2. The diameter of the field of view (FoV).

Magnification	FN18	FN20	FN22
4×	4.50 mm	5.00 mm	5.50 mm
10×	1.80 mm	2.00 mm	2.20 mm
40×	0.45 mm	0.50 mm	0.55 mm
100×	0.18 mm	0.20 mm	0.22 mm

Table 3. The area of the field of view (FoV).

Magnification	FN18	FN20	FN22
4×	15.89625 mm²	19.62500 mm²	23.74625 mm²
10×	2.54340 mm²	3.14000 mm²	3.79940 mm²
40×	0.15896 mm²	0.19625 mm²	0.23746 mm²
100×	0.02543 mm²	0.03140 mm²	0.03799 mm²

Table 2 provides the diameter of the FN, and Table 3 provides examples A_{FV} for different combinations of FN and objective magnification. For instance, at 40× magnification with FN = 20, the field area is approximately 0.19625 mm².

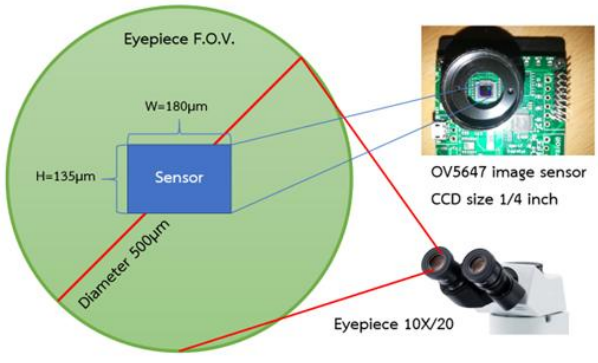

Fig. 11. Comparison between the circular field of view (FoV) observed through the eyepiece and the rectangular FoV captured by the digital image sensor. The eyepiece FoV is defined by the field number (FN) and objective magnification, while the digital sensor captures a rectangular area whose real-world dimensions should be calibrated using a stage micrometer.

Figure 11 illustrates the contrast between a circular FoV from optical microscopy and a rectangular FoV from digital imaging. In digital microscopy, the FoV is rectangular, determined by the physical dimensions of the image sensor and the total camera magnification. The real-world field area is:

$$A_{real} = W_{real} \cdot H_{real} \tag{13}$$

where W_{real} and H_{real} are the calibrated width and height of the image, measured in micrometers using a stage micrometer (see Figs. 12 and 13).

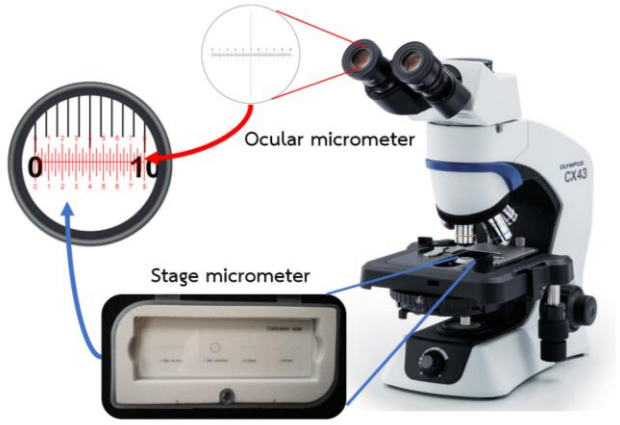

Fig. 12. Calibration of the microscope using both an ocular micrometer and a stage micrometer. The stage micrometer provides a known scale (typically 0.01 mm per division), allowing for accurate determination of the field dimensions as seen through the eyepiece or captured by the digital sensor.

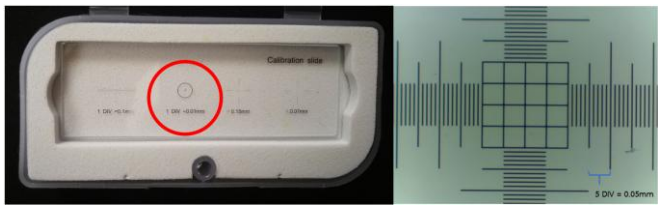

Fig. 13. Captured image showing a calibration grid with squares of known dimensions, used to accurately determine the field of view (FoV) of the digital camera in microscopy applications.

4.3. Mitotic Count and HPF Normalization

A high-power field (HPF) corresponds to the area observed at 400× magnification, and mitotic figures are commonly reported per 10 or 50 HPFs to enhance statistical reliability.

However, the physical size of an HPF—A_{HPF}—varies depending on the microscope's FN and objective lens. This variability introduces inconsistency between laboratories and observers. To address this, a standardized HPF area of $A_{std} = 0.23746\ mm^2$ has been proposed, corresponding to the area of an HPF when FN = 22 at 40×.

To convert an observed mitotic count over 10 HPFs (n_{obs}) into a normalized count based on the standard area, use:

$$n_{10std} = \frac{A_{10std}}{A_{10HPF}} \times n_{obs} \tag{14}$$

This normalized count n_{10std} can then be compared directly with diagnostic guidelines, without needing to account for variations in HPF size across microscopes.

In digital microscopy, normalization proceeds by computing the ratio of the digital field area A_{real} to both A_{HPF} and A_{std}. For example, if one digital image frame captures an area of 0.180 × 0.135 = 0.0243 mm² (as in Section 4.4), then 10 such images equate to 0.243 mm² which less than 1/9.77 of the standard area. Thus, the mitotic count over 10 digital frames must be scaled down by this ratio to yield the count per 10 standardized HPFs. This ensures equivalence between digital and conventional methods and enables consistent thresholding across modalities.

4.4. Calibration of Digital Microscopy

Calibration is essential in digital microscopy to determine the real-world dimensions represented by each pixel in a captured image. This step is necessary because, unlike optical microscopy, digital magnification depends on the entire optical path and image acquisition system, including the objective lens, intermediate optics (e.g., adapters), and the camera sensor.

Calibration is typically performed using a stage micrometer or test patterns with known physical dimensions (Fig. 12 and 13). By comparing the known length on the micrometer to the number of pixels it spans in the image, one can determine the physical size per pixel and, subsequently, the field of view (FoV).

For example, using a Raspberry Pi Camera Module (with an OV5647 sensor) coupled with a 0.5× optical adapter, an image with a resolution of 640 × 480 pixels may initially correspond to an approximate physical area of 0.180 mm × 0.135 mm, yielding:

$$A_{real} = 0.180 \times 0.135 = 0.0243\ mm^2 \tag{15}$$

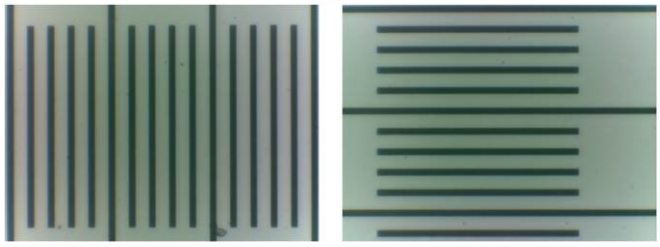

Fig. 14. Captured images showing the apparent sensor area during calibration with a stage micrometer: (left) horizontal calibration to determine the apparent width of the sensor field, and (right) vertical calibration to determine the apparent height. These measurements provide the actual field of view (FoV) dimensions used in quantitative digital microscopy.

However, this theoretical estimate based on optical specifications often deviates from actual measurements due to variations in lens alignment, sensor cropping, and projection geometry. Thus, empirical calibration is needed.

In Fig. 14, calibration results using a stage micrometer show that the actual calibrated field dimensions are smaller:

- Width: $W_{real} = 0.150\ mm$
- Height: $H_{real} = 0.110\ mm$

This gives a corrected FoV:

$$A_{real} = 0.150 \times 0.110 = 0.0165\ mm^2 \quad (16)$$

To normalize mitotic counts from digital images to a standard high-power field (HPF) area of $A_{std} = 0.23746\ mm^2$, we compute the ratio:

$$\frac{A_{std}}{A_{real}} = \frac{0.23746}{0.01650} \approx 14.39 \approx 14.4 \quad (17)$$

Hence, to obtain a mitotic count equivalent to that over 10 standardized HPFs, one must multiply the total count from 10 digital images by this factor:

$$n_{10std} = 14.4 \times n_{obs} \quad (18)$$

This normalization step ensures compatibility with standard histopathology guidelines, even when using compact or low-cost digital microscopy systems.

5. Results of RBC, WBC, and Platelet Detection

The RetinaNet neural network model, composed of a ResNet50 backbone and a Feature Pyramid Network (FPN), was implemented to enable high-performance object detection within the MicrosisDCN camera set. Utilizing transfer learning, the ResNet50 architecture was pre-trained on the publicly available Blood Cell Dataset to initialize its convolutional layers. This allows the model to extract meaningful spatial features using learned filters, which are then passed through the FPN for multi-scale object classification and localization.

To enhance detection performance, the anchor boxes were modified to better match the typical bounding box sizes of red blood cells (RBCs), white blood cells (WBCs), and platelets in microscopic images. This process enabled RetinaNet to generate high-quality feature maps for accurately localizing and classifying small cellular structures.

Training was categorized into three approaches:

1. Non-parameterized training—The model was trained directly using default settings and anchor parameters without further customization.
2. First parameterized training—Anchor parameters and other hyperparameters were adjusted to optimize detection for blood cell sizes and spatial distributions.
3. Second parameterized training—Additional tuning was performed based on insights from the first model, including anchor scaling and anchor ratio refinement.

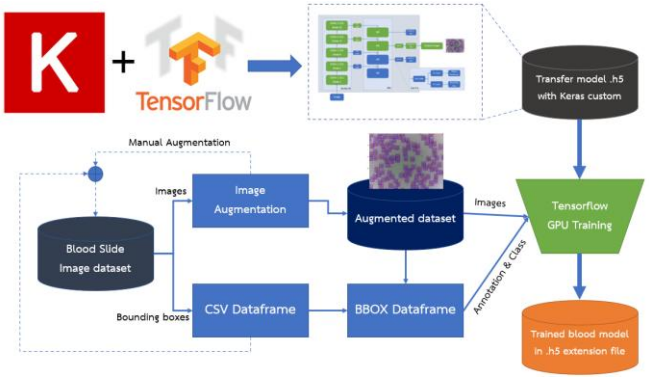

Fig. 16. Overview of data preparation, model training process, and evaluation of cell detection capability.

The expanded training dataset consisted of microscope images with a resolution of 1280×960 pixels. Within this dataset, subregions were extracted for use in feature learning. A filter, or kernel, was applied to these image regions to extract features relevant for classification and localization.

Key aspects of the RetinaNet anchor configuration include:

- Strides: Represent the step size of the convolutional filter as it slides across the image. Larger strides yield coarser but faster feature maps.
- Ratios: Define the aspect ratio (height to width) of anchors. These are used to model objects of varying shapes.
- Scales: Determine the scale of anchor boxes relative to the input image size.
- Anchors: Computed using the combination of scales and ratios. For instance, if a feature map has dimensions of 128×128 pixels, the number of anchors is determined by the product of ratio and scale combinations applied at each location.

In RetinaNet, five levels of feature maps are generated from the ResNet50 backbone, each handling different object sizes due to downsampling. As an example, a feature map of 512×512 allows for fine-grained object localization, while lower-resolution feature maps (e.g., 128×128) focus on larger-scale patterns.

This anchor-based detection strategy enabled robust and scalable cell detection performance across varying cell types and sizes in microscope images. Detection outcomes using each training approach are evaluated and discussed in Section 6, including mAP scores for each cell class.

Table 4. The RetinaNet default parameters.

Sizes	32	64	128	256	512
Strides	8	16	32	64	128
Ratios	0.5	1	2	3	
Scales	1	1.2	1.6		

Table 5. The model with the first adjusting parameter.

Sizes	16	32	64	128	256
Strides	8	16	32	64	128
Ratios	0.5	1	2	3	
Scales	1	1.2	1.6		

Table 6. The model with the second adjusting parameter.

Sizes	8	16	32	64	128
Strides	4	8	16	32	64
Ratios	0.5	1	2	3	
Scales	1	1.2	1.6		

To train the three neural network models, we used a standardized command-line interface. The base command for training was

python train.py --weights ./snapshots/blood_model.h5 --config config.ini --compute-val-loss --tensorboard-dir ./tensorboard_log --multi-gpu 2 --multi-gpu-force --batch-size 2 --epochs 50 --steps 1000 csv annotations.csv classes.csv --val-annotations val_annotations.csv.

For the non-parameterized model (i.e., the default training configuration), the --config config.ini argument was omitted.

After training, each model was converted into an inference model using the command:

python convert_model.py –config config.ini
./snapshots/blood_model_csv_XX.h5
./snapshots/blood_model_csv_XX_convert.h5,

where XX represents the number of the training epoch (ranging from 01 to 50).

To evaluate the inference model's performance using the validation set, we calculated the mean Average Precision (mAP) for all three classes—RBC, WBC, and platelets—using the command:

python evaluate.py csv class.csv
val_annotations.csv./snapshots/blood_model_csv_XX_convert.h5.

This procedure enabled consistent model evaluation and comparison across different parameter configurations and training strategies.

5.1. The Model Without Adjusting Parameters

We trained the neural network model without parameter adjustments for 50 epochs. The mean Average Precision (mAP) values for the three target classes—red blood cells (RBC), white blood cells (WBC), and platelets—are summarized in Table 5. The highest recorded mAP value is 0.7356, while the lowest is 0.6794. All three classes yielded mAP values below 0.80 (80%), indicating moderate overall performance.

To further assess the model's predictive accuracy, we compared its cell counts against ground-truth annotations derived from human visual inspection over a set of 100 identical microscope images. Evaluation metrics included the mean absolute error (MAE), root mean squared error (RMSE), and the coefficient of determination (R-squared). The MAE and RMSE values for RBC and WBC detections were relatively low, suggesting that the model performs well in identifying these two cell types. Conversely, platelet detection exhibited significantly higher MAE and RMSE values, indicating suboptimal performance.

In terms of R^2, RBC and WBC predictions showed values close to one, reflecting strong correlation with ground-truth counts. However, the platelet class exhibited an R-squared value of only 0.1564, implying weak predictive capability. The low score can be attributed to insufficient overlap between the predicted and actual bounding boxes, which hampers accurate class identification. Representative results are illustrated in Fig. 17 for RBCs, and Fig. 18 for WBCs and platelets. While the neural network effectively detects RBCs and WBCs, it consistently fails to identify platelets.

Table 7. The result of trained model without adjusting parameter.

Class Name	**RBC**	**WBC**	**Platelets**
MAE	6.09	0.12	58.91
RMSE	8.4977	0.4899	96.7841
R-Squared	0.9985	0.9730	0.1564
mAP	0.7356		

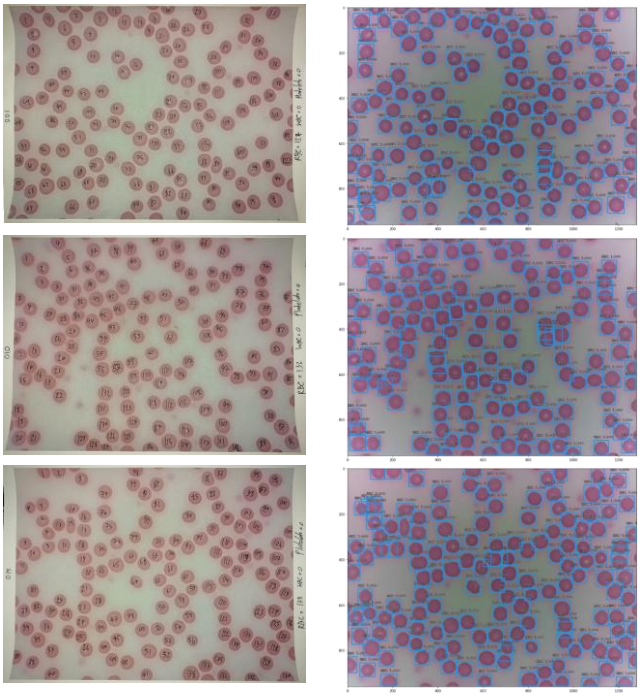

Fig. 17. The actual (left) and predicted (right) red blood cells (RBCs) count results of the trained model without adjusting parameters.

DOI:10.4186/ej.2025.29.6.43

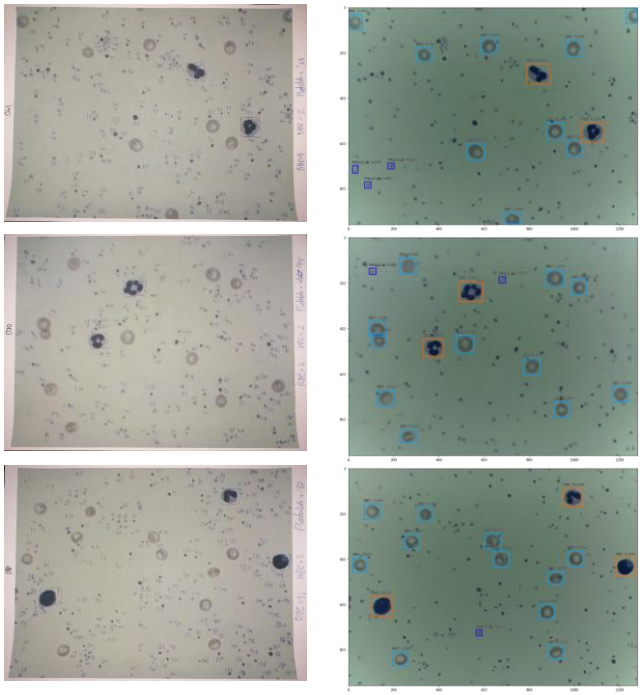

Fig. 18. The actual (left) and predicted (right) white blood cells (WBCs) and platelets count results of the trained model without adjusting parameters.

5.2. The Model with the First Adjusting Parameters

In the initial model, where parameters were not adjusted, the prediction accuracy for all three classes—red blood cells (RBC), white blood cells (WBC), and platelets—remained below 0.8000 (80%). This suggests that the neural network could effectively detect RBCs and WBCs but failed to accurately identify platelets. The mean absolute error (MAE) for RBCs and WBCs was 6.09 and 0.12, respectively, whereas the MAE for platelets reached 58.91, underscoring the model's inadequacy in detecting platelets.

To improve the model's performance across all three cell types, we modified the anchor settings to better align with the actual object sizes in the annotated images, thereby producing feature maps more capable of accurate classification. Table 4 presents the default anchor parameter values, where the sizes parameter defines the image point area used for feature separation, based on the number of feature layers. The smallest value, 32, corresponds to an area of 32×32 pixels. Objects smaller than this threshold are mapped inadequately and therefore remain undetected.

Following the anchor optimization method proposed by Zlocha et al. [24], which tailored anchors for small object detection in CT scan images, we reconfigured the anchor parameters (referred to as type 1) and retrained the model. Table 8 summarizes the training process over 50 epochs. The resulting mAP values for the three classes ranged from 0.7305 to 0.8681, representing a substantial improvement, with all values now exceeding the 0.8000 (80%) threshold. These results suggest that the refined model performs better in detecting all three cell types.

To validate the model, we compared its predictions against ground-truth counts performed by human observers using 100 identical microscope images. The revised model produced low MAE and RMSE values for all classes, and the R-squared values were close to 1.00, indicating high correlation and low variance between predicted and actual counts. Fig. 19 and 20 present sample predictions of RBCs, WBCs, and platelets. The results confirm that the improved neural network model is capable of reliably detecting and counting all three cell types.

Table 8. The result of trained model with the first adjusting parameter.

Class Name	RBC	WBC	Platelets
MAE	1.06	0.06	4.23
RMSE	1.6432	0.2449	9.1011
R-Squared	0.9998	0.9933	0.9960
mAP	0.8681		

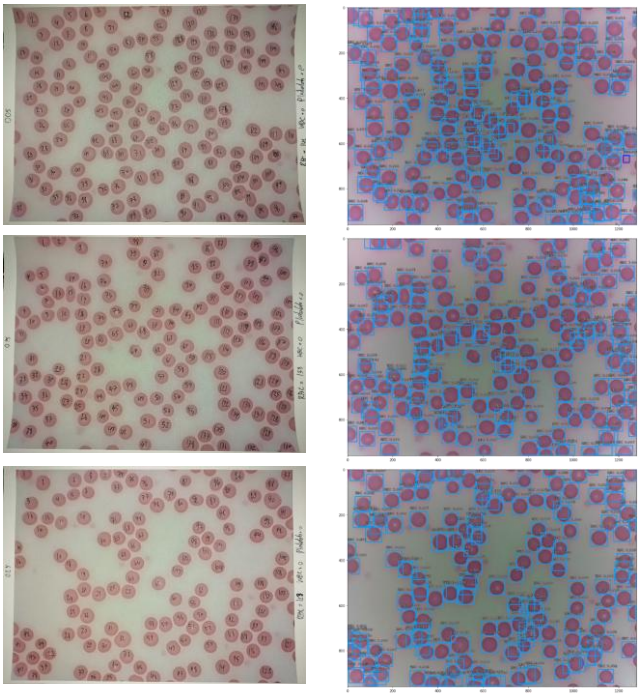

Fig. 19. The actual (left) and predicted (right) red blood cells (RBCs) count results of the trained model with the first adjusting parameter.

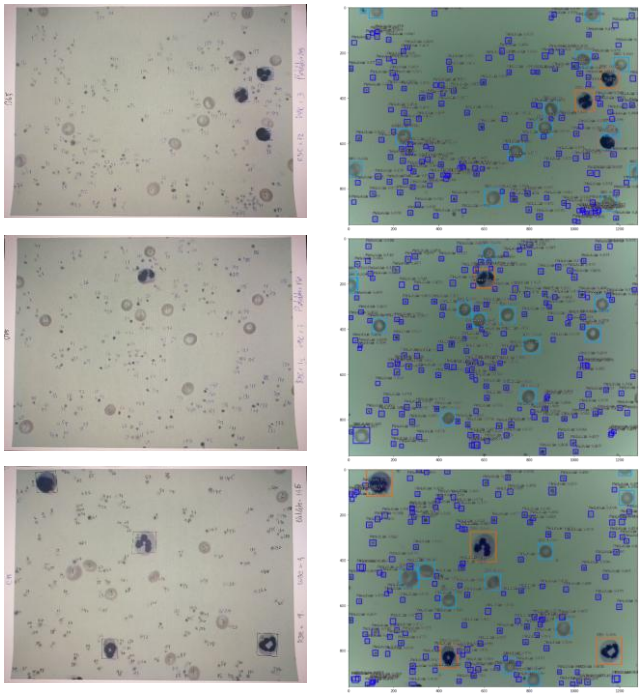

Fig. 20. The actual (left) and predicted (right) white blood cells (WBCs) and platelets count results of the trained model with the first adjusting parameter.

5.3. The Model with the Second Adjusting Parameters

In the second parameter adjustment, the prediction accuracy for all three classes declined significantly. This configuration employed anchor optimization to reduce anchor sizes further, enabling the convolutional (CONV) layers to extract more localized features—an essential aspect of accurate feature extraction. As shown in Table 16, the Sizes parameter represents the pixel area assigned for each anchor, which must be appropriately scaled to allow smaller feature layers to detect fine details. In this configuration, the smallest anchor was set to 32×32 pixels, as in earlier models.

For this experiment, the anchor parameter was adjusted to type 2 for retraining the neural network. Table 9 presents training progress across 50 epochs. The resulting mean average precision (mAP) values for the three classes ranged from a maximum of 0.2293 to a minimum of 0.0037. These values indicate substantially lower prediction accuracy compared to both the baseline model and the first parameter-adjusted model, with all mAP values falling below the 0.8000 (80%) threshold.

The model was further evaluated by comparing its predictions on a test set of 100 identical microscope images to human-counted ground truth values. The resulting mean absolute error (MAE) and root mean squared error (RMSE) values were high for all three cell types, confirming the model's poor performance. While the R-squared value for red blood cells approached 1.0, suggesting some correlation between predicted and actual values, the white blood cell count was low, and platelet detection failed entirely—with no correct predictions recorded.

Table 9. The result of trained model with the second adjusting parameter.

Class Name	RBC	WBC	Platelets
MAE	48.9	1.36	59.36
RMSE	61.0210	2.5287	97.3676
R-Squared	0.9710	0.3294	Negative
mAP	0.0037 to 0.2239		

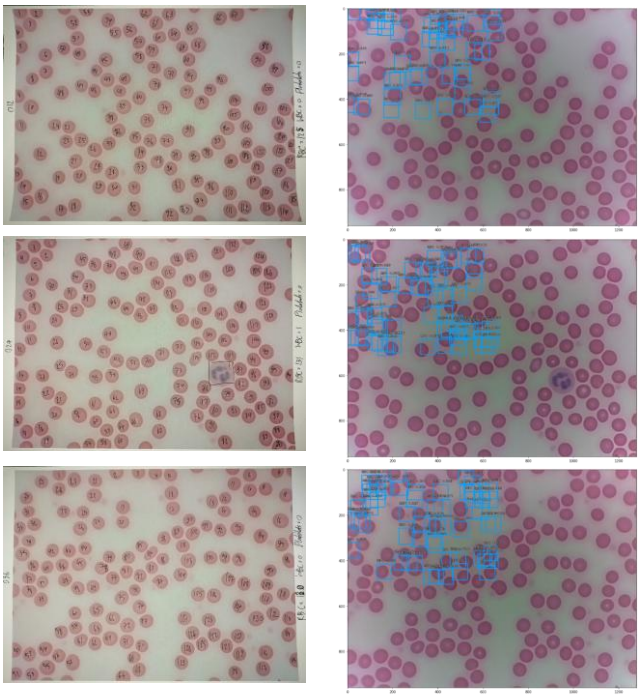

Fig. 21. The actual (left) and predicted (right) red blood cells (RBCs) count results of the trained model with the second adjusting parameter.

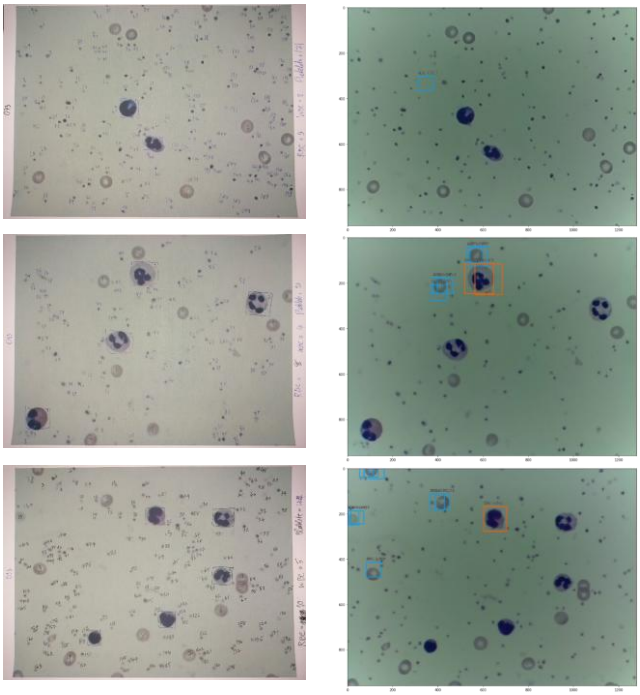

Fig. 22. The actual (left) and predicted (right) white blood cells (WBCs) and platelets count results of the trained model with the second adjusting parameter.

DOI:10.4186/ej.2025.29.6.43

6. Conclusion and Discussion

This study presents the development of an intelligent camera kit capable of automatically detecting and counting blood cells from peripheral blood smear (PBS) images. The system integrates RetinaNet with auto-anchor optimization to enhance detection performance and accelerate the blood cell counting process for diagnostic applications. The resulting device, named MicrosisDCN, is an embedded smart camera that operates alongside a compound microscope to classify and quantify blood cells in real time using a neural network model.

The MicrosisDCN camera attaches directly to the eyepiece lens tube of a microscope. By multiplying the standard FoV area by a factor of 14.4 40× "field images" to equal standard area, the system can display blood cell counts in a standardized format—matching the mitotic count unit observed under a 40X objective lens. This approach effectively emulates the conventional high-power field (HPF) count but automates the detection process, reducing manual effort and interpretation variability.

Unlike commercial microscope cameras that serve only as imaging interfaces and require specific trinocular models and proprietary software, MicrosisDCN is designed for versatility and independence. It supports eyepiece diameters of 23.2 mm, 30.0 mm, and 30.2 mm, and comes equipped with a fully functional embedded system that includes screen, keyboard, and mouse connectivity, eliminating the need for an external computer.

The device runs a pre-installed neural network model but can also be reprogrammed for custom detection tasks. Users can retrain or fine-tune the detection algorithm for specific applications—such as identifying parasites, protozoa, or novel pathogens—by transferring the model to a Linux or Windows computer, retraining with new image datasets, and then redeploying to the MicrosisDCN device.

To support this adaptability, the researcher created an expanded image dataset by capturing and annotating blood cell images under the microscope. A custom script was developed to streamline the annotation process and convert bounding box data into a scalable format that is more efficient than traditional XML as the dataset grew. This dataset was then used to train neural networks using TensorFlow, forming the basis for the models embedded in the MicrosisDCN system.

Three types of model training strategies were tested:

- Non-parameterized model training achieved the highest mAP of 0.7356, successfully detecting red and white blood cells but failing to identify platelets.
- Parameterized model 1 training reached the highest performance with a mAP of 0.8681, accurately detecting red blood cells (RBCs), white blood cells (WBCs), and platelets.
- Parameterized model 2 training resulted in a significantly lower mAP of 0.2239, failing to reliably locate small cells.

Based on these findings, parameterized model 1 is recommended for deployment on the MicrosisDCN system. Users can initiate model inference or training directly on the embedded Raspberry Pi Compute Module 3+ using the OV5647 image sensor (5 MP resolution).

In conclusion, AI-powered tools have revolutionized blood cell counting, enhancing accuracy and time efficiency. Utilizing techniques like CNNs, image segmentation algorithms, and deep learning classification models, these tools reduce manual labor and detect blood cell types from microscopic images. However, these solutions are often costly and require specialized hardware, making them less accessible in low-resource settings or smaller clinics. Our proposed method offers affordability, simplicity, and adaptability, making it suitable for broader clinical implementation in under-resourced healthcare environments. MicrosisDCN offers a compact, cost-effective, and highly adaptable solution for automated blood cell analysis in diagnostic and research settings. Its flexibility and self-contained design make it ideal for fieldwork, education, and laboratory automation—bridging the gap between traditional microscopy and intelligent vision systems.

Fig. 23. MicrosisDCN received a gold medal at the 48th International Exhibition of Inventions Geneva, the Swiss Confederation.

Acknowledgement

This research was supported as part of the Ph.D. dissertation titled "Development of Abnormal Leukocyte Counting System via Smartphone using Convolution Neural Networks [26, 27]," which obtained the 100th Anniversary Chulalongkorn University Fund for Doctoral Scholarship, and was subsequently extended as part of the M.Eng. thesis titled "The application of neural network to detect multiple cells via compound microscope with image sensor [28]," funded by Chulalongkorn University: CU_GI_62_18_21_03.

Finally, we are pleased to recognize the following awards received for this research: Gold medal in class M: Medicine, Surgery, Orthopedics, Material for the Disabled at the 48th International Exhibition of Inventions Geneva, the Swiss Confederation; the Higher Education Innovation Award: Outstanding Award and Gold Medal for the Artificial Intelligence and Smart Devices Topic; and an Outstanding Research Proposal Award from the National Research Council of Thailand (NRCT), Ministry of Higher Education, Science, Research and Innovation at the Thailand Research Expo 2020, Thailand.

References

[1] A. S. Adewoyin and B. Nwogoh, "Peripheral blood film—A review," *Annals of Ibadan Postgraduate Medicine*, vol. 12, no. 2, pp. 71–79, Dec. 2014.

[2] D. Payne, "Use and limitations of light microscopy for diagnosing malaria at the primary health care level," *Bulletin of the World Health Organization*, vol. 66, no. 5, pp. 621–626, 1988.

[3] G. Bradski, "The opencv library," Dr. Dobb's Journal: Software Tools for the Professional Programmer, vol. 25, no. 11, pp. 120–123, Nov. 2000.

[4] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, Jun. 2020.

[5] N. K. Manaswi and N. K. Manaswi, "Understanding and working with Keras," in *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition with TensorFlow and Keras*, 1st ed. Berkeley, CA: Apress, 2018, pp. 31–43.

[6] T. Kattenborn, et al., "Review on convolutional neural networks (CNN) in vegetation remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 173, pp. 24–49, Mar. 2021.

[7] B. Zhao, et al., "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119–135, Apr. 2017.

[8] Y. Wu, et al., "Rethinking classification and localization for object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp. 10186–10195.

[9] Z. Zou, et al., "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.

[10] H.-D. Cheng, et al., "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2259–2281, Dec. 2001.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, Sep. 2014.

[13] R. Girshick, et al., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587.

[14] R. Girshick, "Fast R-CNN," in *Proc. IEEE International Conference on Computer Vision*, Dec. 2015, pp. 1440–1448.

[15] S. Ren, et al., "Faster R-CNN: towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, Dec. 2015, pp. 91–99.

[16] C. Szegedy, et al., "Going deeper with convolutions," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 1–9.

[17] K. He, et al., "Deep residual learning for image recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 770–778.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE International Conference on Computer Vision*, Oct.–Nov. 2017, Venice, Italy, pp. 2999–3007.

[19] B. M. Randles, et al., "Using the Jupyter notebook as a tool for open science: An empirical study," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries*, Jun. 2017, pp. 489–490.

[20] T.-Y. Lin, et al., "Microsoft COCO: Common objects in context," in *Computer Vision – ECCV 2014: 13th European Conf. Zurich, Switzerland*, Sept. 6–12, 2014, Proc., Part V, Cham, Switzerland: Springer, 2014, pp. 740–755.

[21] M. Everingham, et al., "The PASCAL visual object classes challenge: a retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[22] A. F. Symon, et al., "Design and development of a smart baby monitoring system based on Raspberry Pi and Pi camera," in *Proc. 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Sep. 2017, pp. 130–135.

[23] H. Gaiser, M. de Vries, and V. Lacatusu, Keras RetinaNet. 2019. Fizyr. Accessed: Jun. 1, 2024. [Software]. Available: https://github.com/fizyr/keras-retinanet/tree/0.5.1

[24] M. Zlocha, Q. Dou, and B. Glocker, "Improving RetinaNet for CT lesion detection with dense masks from weak RECIST labels," in *Med. Image Comput. Comput. Assist. Interv. – MICCAI 2019: 22nd Int. Conf.*, Shenzhen, China, Oct. 13–17, 2019, Proc., Part VI, Cham, Switzerland: Springer, 2019, pp. 402–410.

[25] D. Kim, L. Pantanowitz, P. Schüffler, D. V. K. Yarlagadda, O. Ardon, V. E. Reuter, M. Hameed, D. S. Klimstra, and M. G. Hanna, "(Re)defining the high-power field for digital pathology," *Journal of Pathology Informatics*, vol. 11, p. 33, 2020. [Online]. Available: https://doi.org/10.4103/jpi.jpi_48_20

[26] D. J. Meuten, F. M. Moore, and J. W. George, "Mitotic count and the field of view area: time to standardize," *Veterinary Pathology*, vol. 53, no. 1, pp. 7–9, Jan. 2016. doi: 10.1177/0300985815593349. :contentReference [oaicite:1]{index=1}

[27] N. Kasamsumran, W. Pora, and E. Karoopongse, "Development of abnormal leukocyte counting system via smartphone using convolutional neural network," Chulalongkorn Univ. Theses Dissertations (Chula ETD). [Online]. Available: https://digital.car.chula.ac.th/chulaetd/11536. Accessed: Jul. 04, 2025.

[28] N. Kasamsumran, P. Ittichaiwong, C. Chinudomporn, K. Veerakanjana, E. Karoopongse, and W. Pora, "AI-assisted web application for leukocyte abnormality counting with YOLOv11 and smartphone microscopy," *IEEE Access*, vol. 13, pp. 89079–89107, 2025. doi: 10.1109/ACCESS.2025.3569767.

[29] N. Kasamsumran and S. Pumrin, "The application of neural network to detect multiple cells via compound microscope with image sensor," Chulalongkorn Univ. Theses Dissertations (Chula ETD), [Online]. Available: https://digital.car.chula.ac.th/chulaetd/9614

Natthakorn Kasamsumran received a B.Eng. degree in electrical engineering from Burapha University, Thailand and an M.Eng. degree in electrical engineering from Chulalongkorn University, Thailand, where he is currently a Ph.D. candidate in electrical engineering with the 100th Anniversary Chulalongkorn University Fund for Doctoral Scholarship in the department of electrical engineering, faculty of engineering, Chulalongkorn University, Thailand. His current research interests include embedded systems, the internet of things (IoT), artificial intelligence in medicine, microscopy imaging, computer vision, machine learning, mechatronics, and invention creation and development.

Amornthep Phunsin holds a B.Sc. degree in industrial education with a major in telecommunication engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand. With nearly two decades of experience in the engineering field, he specializes in test and measurement systems, embedded systems design, and hardware development. He founded Q-Wave Systems Co., Ltd., where he leads the company's strategic direction in engineering services, research and development, and the innovation of embedded hardware solutions for industrial applications and tech start-ups.

Suree Pumrin received her B.Eng. degree in electronics engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand, in 1990, followed by M.S.E. degree in electrical engineering from Arizona State University, USA., in 1992, and Ph.D. degree in electrical engineering from University of Washington, USA., in 2002. She is currently an assistant professor at the Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, Thailand. Her research interests include image processing, computer vision, and embedded systems.

Wanchalerm Pora received his B.Eng. and M.Eng. degrees in electrical engineering from Chulalongkorn University, Thailand in 1992 and 1995, respectively. In 2000, he went to the United Kingdom and earned a Ph.D. from Imperial College in London. He is currently the Head of Embedded Systems and IC Design Research Laboratory, Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University. His special interests include smart devices, IoT systems, and artificial intelligence.