JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 37, 827-838 (2021)
DOI: 10.6688/JISE.202107_37(4).0006

Anomaly Chicken Cell Identification Using Deep Learning Techniques

NATINAI JINSAKUL[1,2], CHENG-FA TSAI[2,*] AND CHIA-EN TSAI[3]
[1]*Department of Tropical Agriculture and International Cooperation*
[2]*Department of Management Information Systems*
National Pingtung University of Science and Technology
Pingtung, 912 Taiwan
[3]*Department of Biochemistry and Molecular Biology*
National Cheng Kung University
Tainan, 701 Taiwan
E-mail: {P10522009; cftsai}@mail.npust.edu.tw; {plusntsai}@gmail.com

Chicken cell abnormal identification by manual method that clearly lacks speed and accuracy. However, the success of deep learning techniques from the convolutional neural network (CNN), it may be providing solutions to cell biology laboratory tasks. This paper collected the novel chicken cell microscopic image datasets for training the different kinds of CNN models and optimizers to find promising applications that might be developed. The top model indicates that ResNet34 with Adam optimizer achieved training accuracy of 100%, testing accuracy of 98.14%, and the lower time on the outstanding confusion matrix. In addition, the validation result represented correct identification, guaranteeing by experts. This study shows the potential method to be improved to an application of identification systems in the actual animal and biology laboratories.

Keywords: anomaly identification, chicken cell, microscopic image, artificial intelligence, deep learning, convolutional neural networks, optimizers

1. INTRODUCTION

The success of deep learning indicates the significant potential for utilizing artificial intelligence in the real world [1]. Specifically, it will be possible to apply the convolutional neural network (CNN) to solve problems in cell biology laboratory tasks such as detection [2], tracking [3], screening [4], classification [5] and identification of anomalies of the cell [6]. However, in the animal science there have been no CNN operations, especially in the chicken cell biology laboratory. Chicken meat is one of the main proteins in human consumption [7], but an important disease called chicken anemia virus (CAV) infected chickens in the poultry industry [8]. Therefore, it is possible to use chicken cells to identify that are normal or abnormal cells by the effects of infectious disease. The rapid diagnosis of microscopic cell images is the solution to the identification of anomaly cells. Fortunately, the development of image processing techniques has utilized to the automatic identification [9], with CNN for image classification [10]. The famous CNN development architectures such as AlexNet [10], GoogleNet inception [11], VGGNet [12] and ResNet [13], all of architectures inside consist of several important layers [10, 14].

In the improvement of the CNN accuracy, we can select an optimizer to update the parameters, which is the optimization method [15]. The gradient descent optimization

Received August 27, 2019; revised February 2, 2020; accepted March 26, 2020.
Communicated by Tzung-Pei Hong.

involves simple calculating and is quick with large datasets. Seven types of gradient descent optimizers were compared: Stochastic Gradient Descent (SGD) [15], Adaptive Gradient (Adagrad) [16], Adaptive Delta (Adadelta) [17], Root Mean Square Propagation (RMSProp) [18], Adaptive Momentum (Adam) [19], Adaptive Max Pooling (AdaMax) [19], and Nesterov Adaptive Momentum (Nadam) [20]. We never know which optimizer will outperform the others for all cases of dataset. The theory is not sufficient in order to verify which optimizer is most suitable. The experiment should be performed to investigate the performance which might be appropriate for matching the properties of dataset.

The aim of this study is to apply the well-known CNNs by optimization using seven optimizers to identify anomaly chicken cells. The model accuracy is evaluated by classification evaluation to investigate which CNN architectures and which optimizers harmonize to generate the best results. In the validation phase, we used another unknown class of images to validate the model which might be applied for real operations.

2. LITERATURE REVIEWS

In this section, we provide a brief of the previous studies on CNN techniques for cell identification which have been proposed in recent years, as discussed below.

Automation cervical abnormal cell screening via Pap smear or liquid-based cytology (LBC) has been developed [4], employing the CNN by AlexNet with SGD optimizer, the results representing accuracy of 98.3%. While, the counting of blood cells and molds in the method based on GoogleNet with RMSprop was proposed [6] by achieved a precision of 90.7%. Architecture based on the deep residual network of ResNet to accurate human Epithelial-2 (HEp-2) cell image classification to diagnosis of autoimmune disease [21]. The study achieved identification accuracy of 97.14% with the SGD optimizer. VGG16 model with Adam optimizer were employed in the classification of the abnormal of tumor cells based on actin-labeled fluorescence microscopy images [22]. The results indicated training accuracy of 100.0% and testing accuracy of 97.2%. Feature extraction of AlexNet with the SGD to identify abnormalities of white blood cells which achieved average accuracy of 99% [23]. A comparison of VGG, ResNet with Adadelta optimizer, and GoogleNet with SGD for the diagnosis of cancer cell was presented [24]. The models give a new dimension to cytology studies where achieved an accuracy of 96.25%.

According to various sources in the literature discussing cell classification by applying state-of-the-art deep learning, most approaches employ CNN to obtain solutions in the area of cell biomedicine for humans. There has not yet been an application to solve the target problem in animal cells, since those are dissimilar to the biomedical images, so this study seeks to find the solution by applying CNN with optimizers.

3. THEORETICAL OF CNN ARCHITECTURES AND OPTIMIZERS

3.1 CNN Architectures

Among the modern CNN architectures proposed in the introduction, the three main layers [25] are suggested (shown as in Fig. 1). These are described as follows:

1. The Convolutional layer role play by weight sharing in the same feature map and parameters reduces the total number of parameters, targeting conduct on a small patch of image input for the extracted features. This layer employs a convolution filter over local patches of the input image by calculation in the following function

$$C_j = \sum_i N_i \times K_{ij} + B_j \tag{1}$$

where C_j indicates the output from the convolution process, N_i determines the input to the convolutional layer, K_{ij} is the kernel of convolution, and bias is given by B_j

$$M_j = A(C_j). \tag{2}$$

M_j defines the output feature map of the convolutional layer and $A(C_j)$ is an activation function to compute non-linearity and handling non-linear features of the input.

2. The Pooling layer is computed to reduce the feature map resolutions, and accordingly the number of parameters, to supersede where $D(N_j)$ shows a pooling function in the following

$$C_j = D(N_j) \tag{3}$$

The Pooling operation is picked and remains unchanged during training data. In operations to compute both the maximizing value and averaging value over the feature map, the size of the feature map is reduced.

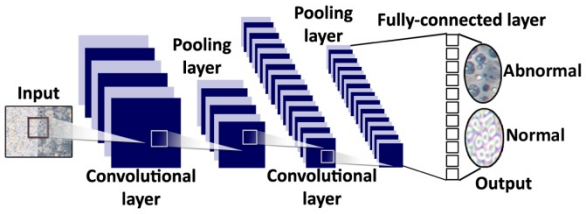


Fig. 1. Basic CNN architecture.

3. The Fully-Connected Layer from the pooling layer is where the feature is extracted and labeled by mapping in a fully-connected layer, with neurons changing to one dimension. Each current layer is linked to previous layers to obtain information on different position patterns with the connected layers.

The basic CNN mentioned above, there are various architecture modifications in order to handle more complex image datasets and problems, which are briefly explained of these modern architectures as follows:

– AlexNet requires input 224×224 pixels 3 color channels, defined as stacked with connected layers. The first convolutional layer has 96 kernel filters of size 11×11×3 and a stride of 4, while the second convolutional layer takes its input as max-pooling layers and utilizes 256 filters of size of 5×5×96. The third convolutional layer uses the size of 3×3×256 with 384 filters connected to the max-pooling layer from the outputs of the

second convolutional layer. The fourth convolutional layer has defining filters of 384, and the fifth convolutional layer creates 256 filters with size of 3×3×384 and linked with max-pooling after the fifth layer. Two fully-connected there are 4,096 and 1,000 units by using the dropout to avoid overfitting and softmax classification with SGD optimizer.

– GoogleNet is improved depth and width inside the network. The inception module comprises a sub-network by parallel operation with convolution filters. The inception reduction, it is determined that 1×1 convolutions are used before the size of 3×3, 5×5 convolutions and after 3×3 max pooling. The output forwards to concatenate the filter and the next modules. The input 224×224 with 3 channels, comprising 22 convolutional layer, 5 pooling layer, a 1024 units fully-connected, a dropout, SGD, and softmax.
– VGGNet was developed based on AlexNet, the main increasing deep with small size 3×3 convolution filters and input of 224×224 with 3 channels. It operates in convolutional layers and pooling by using five max-pooling layers. On 3 fully-connected, the first and second contain 4,096, while the third layer set 1,000 nodes, using SGD for optimization and classification with softmax.
– ResNet adds the residual block that operated on the same dimensions of the input and output, and when the sizes of the input and output increased difference, the residual block shortcut was utilized. ResNet uses a 224×224 with 3 channel colors. The network consists of convolutional layers of 3×3 filters. ResNet also utilizes a batch normalization [26] layer after each convolution layer and before the activation. The final part calculates by the average pooling, and the fully-connected applies the softmax, SGD optimizer.

3.2 Optimizers

The seven different kinds of gradient descent optimization that are widely used [15], in providing the behavioral abilities each algorithm described in this essence.

3.2.1 Stochastic gradient descent (SGD)

Stochastic gradient descent (SGD) [15] computes the gradient of the cost function in θ to update training for faster learning performance. In addition, SGD can generate frequent updates by high variance in fluctuates severely. SGD compared parameters (example such as $x^{(i)}$ and $y^{(i)}$) to update in each training process computation

$$SGD_{update} := \theta = \theta - lr \cdot \nabla_{\theta} lf(\theta; x^{(i)}; y^{(i)}) \tag{4}$$

where the learning rate is defined as *lr* and *lf()* for the loss function. Configurations for the good learning rate of 0.01 and 0.9 of momentum [27] are required for with SGD.

3.2.2 Adagrad, or the Adaptive subgradient

Adagrad, or the Adaptive subgradient [16], is produced for gradient-based optimization. Adjusts the small value and infrequent steps for frequent parameters to improve the SGD and the large-scale training where gf_t as the gradient of the objective, *lr* at each time step assigning t for every θ_i based on the update, that has been calculated for θ_i, the sum is defined as i, for the squares of the gradients to θ_i up to t, while *st* determines the smoothing term while avoiding division by zero. ssq_t comprises the sum of the squares of the gradient

with respect to all parameters θ along its diagonal, it then turns to a vector and can be implemented by operation in the following equation

$$Adagrad_{update} := \theta_{t+1,i} = \theta_{t,i} - \frac{lr}{\sqrt{ssq_{t,ii} + st}} \cdot gf_{t,i} \tag{5}$$

3.2.3 Adadelta resolves the weakness of the Adagrad

Adadelta resolves the weakness of the Adagrad [17] to reduce the decreasing *lr* replaces accumulation by past squared gradients and creates a limitation of accumulated past gradients to a fixed size [15]. The value of $rmse[\Delta\theta]_t$ with updates of the root mean squared error (RMSE) is approximated, and *gf* defines the gradient of the objective, *lr* as 1 and in the $rmse[\Delta\theta]_t$ updates minus *lr*, with rules of update in the following

$$\begin{aligned} &\Delta\theta_t = \frac{rmse\left[\Delta\theta\right]_{t-lr}}{rmse\left[gf\right]_t} gf_t, \\ &Adadelta_{update} := \theta_{t+1} = \theta_t + \Delta\theta_t. \end{aligned} \tag{6}$$

3.2.4 RMSprop or root mean square propagation [18]

RMSprop or Root Mean Square Propagation [18] is developed based on the Adadelta, and it is identical to the first $Adadelta_{update}$, it needs to fix radically subdued learning rates of the Adagrad. RMSprop operates an exponentially decaying average of squared gradients by division [15]. RMSProp computed in the following update the rule

$$\begin{aligned} &run\left[gf^2\right]_t = \gamma run\left[gf^2\right]_{t-1} + 0.1gf_t^2 \\ &RMSprop_{update} := \theta_{t+1} = \theta_t - \frac{lr}{\sqrt{run\left[gf^2\right]_t + st}} gf^t \end{aligned} \tag{7}$$

where $run[gf^2]_t$ is the running average and defines *st* for the smoothing term. The experiment suggests γ of 0.9 and is a default value for the *lr* set to 0.001 [15, 18].

3.2.5 Adaptive momentum

Adaptive Momentum, also known as Adam [19], generates adaptive learning rates by keeps a decaying average exponentially of past gradients. The process computing bias-corrected first as bc_t and second me_t as estimates by beta (β_1, β_2) manually, then update the parameters from Adadelta and RMSprop to update the rule as

$$Adam_{update} := \theta_{t+1} = \theta_t - \frac{lr}{\sqrt{me_t} + st} bc_t. \tag{8}$$

Default of Adam set the β_1 of 0.9, β_2 of 0.999, and 10^{-8} for the smoothing in *st* [15, 19].

3.2.6 AdaMax is an extension of the Adam optimizer [19]

AdaMax is an extension of the Adam optimizer [19]; it generates a rule for an update in weights to the inverse relation to a norm of current and past gradients. It defines the infinity norm-constrained as u_t by relying on the max operation and employing u_t to *AdaMax*$_{update}$ rule in the following calculation

$$\begin{aligned} u_t &= \beta_2^{\infty} pg_{t-1} + (1-\beta_2^{\infty})\left|gf_t\right|^{\infty} \\ &= \max(\beta_2 \cdot pg_{t-1}, \left|gf_t\right|) \\ AdaMax_{update} &:= \theta_{t+1} = \theta_t - \frac{lr}{u_t} bc_t \end{aligned} \tag{9}$$

where past gradients as pg_{t-1}, and current gradient is determined as $|gf_t|^{\infty}$. AdaMax has a suitable with 0.002 of lr, β_1 equal 0.9 and β_2 of 0.999 [15, 19].

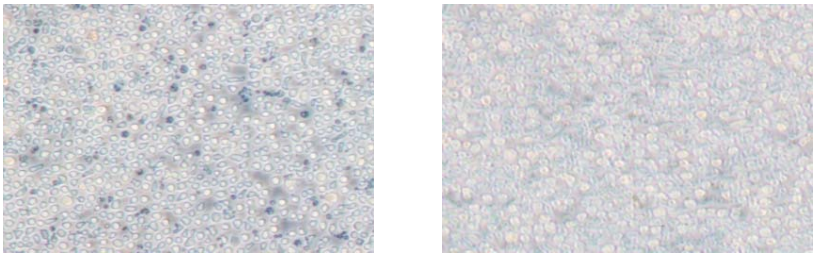
3.2.7 Nesterov-accelerated Adaptive Moment Estimation, or Nadam [20], is an optimizer incorporating two methods of Nesterov-accelerated Gradient (NAG) into Adam through modifying the momentum. NAG can be greater accuracy in the gradient direction by updating the parameters, as shown in the *Nadam*$_{update}$ in the function below

$$Nadam_{update} := \theta_{t+1} = \theta_t - \frac{lr}{\sqrt{me_t + st}} \left(\beta_1 bc_t + \frac{(1-\beta_1) gf_t}{1-\beta_1^t} \right) \tag{10}$$

4. MATERIALS AND METHODS

4.1 Image Dataset Collection and Preparation

The novel chicken cell microscopic image dataset was collected and classified by the expertise at the College of Veterinary Medicine, National Pingtung University of Science and Technology, Taiwan. A microscope device connected to a digital camera was used to capture the chicken cell images (examples of chicken cell images are shown in Fig. 2), by original sizes of 3872 × 2592 pixels. The cell images collected amounted to 2,690 images split into anomaly (1,327) and normal (1,363). We employed the 5-fold cross-validation methods to evaluate the performance by 80% (2,152 images) for training and 20% (538 images) for testing by CNN architecture requirements for input sizes.



(a) Anomaly (b) Normal

Fig. 2. Representation of chicken cell images.

4.2 Training Image Datasets with Different CNN Architectures and Optimizers

This paper used a computer with i7-6700 3.40GHz CPU, 12 GB RAM, and the model is trained on NVIDIA GeForce GTX 1060 GPU with 6 GB of memory. The implementation of this study used Python [28], while the part of the training employed the Keras [29]. All CNN architectures were trained to determine the batch size of 24 for 50 epochs with seven optimizer configurations with default values based on the suggestions of various optimizer paper reviews [15-20].

5. EXPERIMENTAL RESULTS AND DISCUSSIONS

5.1 Accuracy of Anomaly Chicken Cell Identification Results

The model is evaluated in terms of the accuracy. The accuracy requires of true positive, true negative, false positive and false negative, for every class according to the confusion matrix [30]. If the prediction indicates an anomaly cell as abnormal, that is a true positive for the class; then all other true predictions for other are a true negative, and belong to the abnormal cell. When the prediction appears as a normal cell then it is termed as false positive. In another way, if it is the anomaly, then it belongs to the false negative.

After completing the training in the 5-fold cross-validation, the results in the models of GoogleNet and ResNet, by the training accuracy equal to 100%. The training accuracy is indicated in Table 1, while the training time is indicated in Table 2. The testing accuracy results are shown in Table 3, and the drawing of the confusion matrix are illustrated in Fig. 3. The testing image dataset of 538 images, divided into abnormal of 251 images and normal of 287 images. The excellent results were obtained in ResNet34 with Adam optimizer and ResNet101 with Adadelta, similar results were achieved in testing accuracy at 98.14% by identifying 254 images in the abnormal cell with misclassification of 9 images, and 274 images in the normal, while failing only 1 image.

The two available models with a testing accuracy of 98.14% included ResNet34 with Adam and ResNet101 with Adadelta. Although the accuracy indicated equal scores, we can consider the time consumption to determine which model spent the lowest time. ResNet101 with Adadelta which used a time of 49.54 minutes, and the model that employed the lower time of 35.27 minutes represented by ResNet34 with Adam. In the comparison of the training time consumption, we can see that ResNet34 with Adam is the deep learning model which is the most reasonable in our image dataset for this study.

Table 1. Training accuracy results of various CNN architectures and optimizers based on the 5-fold cross-validation method.

CNN architectures	Optimizers and training accuracy (%) results						
	SGD	Adagrad	Adadelta	RMSprop	Adam	AdaMax	Nadam
AlexNet	50.56	49.44	50.56	49.44	50.56	49.44	49.44
GoogleNet	**100**	99.67	73.42	99.44	97.86	**100**	99.02
VGG16	49.86	50.60	49.40	50.60	50.60	50.60	50.60
VGG19	49.40	49.40	49.40	50.60	49.40	49.40	49.40
ResNet34	99.21	**100**	**100**	99.77	**100**	**100**	**100**
ResNet101	75.60	**100**	**100**	99.54	97.82	97.91	97.44

Table 2. Training time consumption based on 5-fold cross-validation method.

CNN architectures	Optimizers in training and testing time (minutes)						
	SGD	Adagrad	Adadelta	RMSprop	Adam	AdaMax	Nadam
AlexNet	6.15	6.39	7.27	6.49	7.26	6.53	7.25
GoogleNet	14.40	15.32	16.33	15.18	15.59	15.21	16.41
VGG16	34.18	34.54	38.20	35.03	38.06	35.13	38.05
VGG19	35.10	36.14	38.12	37.01	40.38	37.14	40.32
ResNet34	34.33	34.47	35.08	35.56	35.27	35.01	35.12
ResNet101	47.04	48.07	49.54	48.35	49.04	48.01	50.11

Table 3. Testing accuracy results for various CNN architectures and optimizers based on the 5-fold cross-validation method.

CNN architectures	Optimizers and testing accuracy (%) results						
	SGD	Adagrad	Adadelta	RMSprop	Adam	AdaMax	Nadam
AlexNet	51.12	48.88	51.12	48.88	51.12	48.88	48.88
GoogleNet	94.24	95.54	77.14	96.65	92.75	97.21	97.58
VGG16	51.12	51.12	48.88	51.12	51.12	51.12	51.12
VGG19	48.88	48.88	48.88	51.12	48.88	48.88	48.88
ResNet34	92.19	97.03	97.96	91.26	**98.14**	96.84	97.58
ResNet101	76.77	94.42	**98.14**	93.68	76.21	86.99	92.57

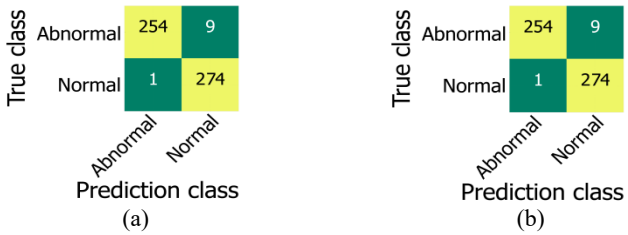


Fig. 3. Test image dataset for the confusion matrix results from the 5-fold cross-validation of (a) ResNet34 with Adam optimizer and (b) ResNet101 with Adadelta.

5.2 Model Validation of Anomaly Chicken Cell Identification Results

In this stage, we collected 20 more microscopic chicken cell images that were of unknown classification for model validation which had been excluded from training and testing procedures. Unknown categories of these images were used for the identification model, and we have shown the result for Fig. 4 to a veterinarian who was an expert to guarantee the accuracy. It appeared that the validation results were verified that all of the images were correct under the model for anomaly chicken cell identification.

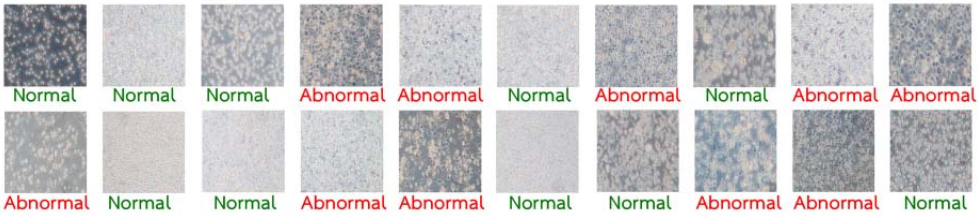


Fig. 4. Model validation of ResNet34 with Adam to identify anomalous chicken cells.

6. CONCLUSIONS

This study, we focused on the best model, which was chosen based on testing accuracy results. The appropriate and excellent probability model in this study appears in ResNet34 with Adam and ResNet101 with Adadelta optimizer from the 5-fold cross-validation method. In addition, for greater clarity clear, we brought the two models to display the classification with the confusion matrix. The results show that both models indicated the achieved result for the lowest misidentification of 10 images and the correct identification achievement was of 528 images from a total set of 538 of test images. Even though the testing accuracy results of each model are equal, the training time consumption can be the judge of which model is most suitable by using the lowest time.

The model that spends the lowest time is ResNet34 with Adam optimizer by took the training time of 35.27 minutes. The comparison of the results in this study can be summarized by concluding that ResNet34 with Adam optimizer is an appropriate deep learning technique in terms of delivering a high level of accuracy and using a suitable period of time for utilization in the real operation. In addition, we conducted the model validation stage with another new and unknown class of 20 chicken cell microscopic images, which were only used for validation. The model validation result represents correct identification predictions on all the chicken cell microscopic images, which guarantees by the experts. All of the results of this study look very promising for the CNN method, which can be improved for a novel application for anomaly chicken cell identification systems in the actual operation of animal science and cell biology laboratories.

ACKNOWLEDGEMENT

The authors would like to appreciate the anonymous reviewers for their useful comments to improving the quality of this paper, and we thank Suttitas Tongkamsai, DVM from the Poultry Health Laboratory-Southern Taiwan Division, College of Veterinary Medicine for providing image dataset. In addition, we thank the Department of Management Information Systems, National Pingtung University of Science and Technology, Taiwan for supporting this research. This research was funded by the Ministry of Science and Technology, Taiwan, grant numbers MOST-107-2637-E-020-006, MOST-108-2637-E-020-003, MOST-108-2321-B-020-003, and MOST-107-2321-B-020-005.

REFERENCES

1. H. Hildmann and B. Hirsch, "Overview of artificial intelligence," *Encyclopedia of Computer Graphics and Games*, in N. Lee, (ed.), Springer International Publishing, NY, 2018, pp. 1-9.
2. M. N. Kashif, S. E. A. Raza, K. Sirinukunwattana, M. Arif, and N. Rajpoot, "Handcrafted features with convolutional neural networks for detection of tumor cells in histology images," in *Proceedings of IEEE 13th International Symposium on Biomedical Imaging*, Vol. 2, 2016, pp. 1029-1032.

3. M. J. Newby, M. A. Schaefer, T. P. Lee, M. G. Forest, and K. S. Lai, "Convolutional neural networks automate detection for tracking of submicron-scale particles in 2D and 3D," in *Proceedings of National Academy of Sciences*, Vol. 115, 2018, pp. 9026-9031.
4. L. Zhang, L. Le, I. Nogues, R. M. Summers, S. Liu, and J. Yao, "DeepPap: Deep convolutional networks for cervical cell classification," *IEEE Journal of Biomedical and Health Informatics*, Vol. 21, 2017, pp. 1633-1643.
5. G. Liang, H. Hong, W. Xie, and L. Zheng, "Combining convolutional neural network with recursive neural network for blood cell image classification," *IEEE Access*, Vol. 6, 2018, pp. 36188-36197.
6. X. Du, L. Liu, X. Wang, G. Ni, J. Zhang, R. Hao, J. Liu, and Y. Liu, "Automatic classification of cells in microscopic fecal images using convolutional neural networks," *Bioscience Reports*, Vol. 39, 2019, No. BSR20182100.
7. M. Henchion, M. McCarthy, C. V. Resconi, and D. Troy, "Meat consumption: Trends and quality matters," *Meat Science*, Vol. 98, 2014, pp. 561-568.
8. P. Castaño, J. Benavides, M. S. Lee, M. Fernández, M. Fuertes, M. Royo, J. M. Fernández, V. Pérez, and M. C. Ferreras, "Tissue tropism of chicken Anaemia virus in naturally infected broiler chickens," *Journal of Comparative Pathology*, Vol. 167, 2019, pp. 32-40.
9. J. Giraldo-Zuluaga, A. Salazar, G. Diez, A. Gomez, T. Martínez, J. F. Vargas, and M. Peñuela, "Automatic identification of Scenedesmus polymorphic microalgae from microscopic images," *Pattern Analysis and Applications*, Vol. 21, 2018, pp. 601-612.
10. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.
11. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.
12. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd IAPR Asian Conference on Pattern Recognition*, 2015, pp. 730-734.
13. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.
14. P. H. B. Diniz, T. L. A. Valente, J. O. B. Diniz, A. C. Silva, M. Gattass, N. Ventura, B. C. Muniz, and E. L. Gasparetto, "Detection of white matter lesion regions in MRI using SLIC0 and convolutional neural network," *Computer Methods and Programs in Biomedicine*, Vol. 167, 2018, pp. 49-63.
15. S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, Vol. abs/1609.04747, 2016, No. 04747.
16. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and Stochastic optimization," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2121-2159.
17. D. M. Zeiler, "ADADELTA: An adaptive learning rate method," *CoRR*, Vol. abs/1212.5701, 2012, No. 5701.

18. G. E. Hinton, "Tutorial on deep learning, in: IPAM graduate summer school: Deep learning," *Feature Learning*, 2012, p. 262.
19. D. Kingma and J. Ba, "Adam: A method for Stochastic optimization," *CoRR*, Vol. abs/1412.6980, 2014, No. 6980.
20. D. Timothy, "Incorporating Nesterov momentum into Adam," in *Proceedings of International Conference on Learning Representations*, 2016, pp. 2013-2016.
21. H. Lei, T. Han, F. Zhou, Z. Yu, J. Qin, A. Elazab, and B. Lei, "A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning," *Pattern Recognition*, Vol. 79, 2018, pp. 290-302.
22. R. Oei, G. Hou, F. Liu, J. Zhong, J. Zhang, Z. An, L. Xu, and Y. Yang, "Convolutional neural network for cell classification using microscope images of intracellular actin networks," *PLOS ONE*, Vol. 14, 2019, No. e0213626.
23. R. B. Hegde, K. Prasad, H. Hebbar, and B. M. K. Singh, "Feature extraction using traditional image processing and convolutional neural network methods to classify white blood cells: a study," *Australasian Physical & Engineering Sciences in Medicine*, Vol. 42, 2019, pp. 627-638.
24. A. R. Saikia, K. Bora, L. B. Mahanta, and A. K. Das, "Comparative assessment of CNN architectures for classification of breast FNAC images," *Tissue and Cell*, Vol. 57, 2019, pp. 8-14.
25. N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *Proceedings of International Conference on Communication and Signal Processing*, 2017, pp. 0588-0592.
26. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Vol. 37, 2015, pp. 448-456.
27. N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, Vol. 12, 1999, pp. 145-151.
28. G. van Rossum, "Python tutorial," Technical Report No. CS-R9526, Centrum voor Wiskunde en Informatica, 1995.
29. F. Chollet, "Keras," https://keras.io, 2015.
30. T. Shanthi and R. S. Sabeenian, "Modified Alexnet architecture for classification of diabetic retinopathy images," *Computers & Electrical Engineering*, Vol. 76, 2019, pp. 56-64.

Natinai Jinsakul (南帝奈) obtained the M.S. degree in Computer Technology from King Mongkut's University of Technology North Bangkok, Thailand. He is currently pursuing the Ph.D. degree in both of Department of Tropical Agriculture and International Cooperation and Department of Management Information Systems, National Pingtung University of Science and Technology, Taiwan. His study interest includes multimedia technology, data science, artificial intelligence, machine learning and deep learning.

Cheng-Fa Tsai (蔡正發) is a Full Professor in the Department of Management Information Systems, National Pingtung University of Science and Technology, Taiwan. He has published over 200 well-known journal papers and conference papers and several books in the MIS. He holds, or has applied for, fifteen U.S. patents and thirty ROC patents in his research areas. Dr. Tsai research interests are in the areas of artificial intelligence, data mining, database systems, mobile communication and intelligent systems, with emphasis on efficient data analysis and deep learning.

Chia-En Tsai (蔡佳恩) received the M.S. degree in Biochemistry and Molecular Biology from National Cheng Kung University, Taiwan. Her research interests include Pathogen-Host interaction and innate immune response.