Translation of article 10.22146/jnteti.v14i1.17162

Comparison of Optimizer Use in White Blood Cell Classification Employing CNN

Dede Kurniadi[1], Rifky Muhammad Shidiq[1], Asri Mulyani[1]

[1] Program Studi Teknik Informatika, Jurusan Ilmu Komputer, Institut Teknologi Garut, Garut, Jawa Barat 4415, Indonesia

[Submitted: 19 November 2024, Revised: 21 January 2025, Accepted: 1 February 2025]
Corresponding Author: Dede Kurniadi (email: dede.kurniadi@itg.ac.id)

ABSTRACT — White blood cells are crucial components of the immune system responsible for combating infections and diseases. The classification and counting of white blood cells are typically performed manually by experienced operators or via automated cell analysis systems. The manual method is inefficient, time-consuming, and labor-intensive, while automated analysis machines are often expensive and require stringent sample preparation. This study aimed to compare the performance of three optimizers—root mean square propagation (RMSProp), stochastic gradient descent (SGD), and adaptive moment estimation (Adam)—in a white blood cell classification model using a convolutional neural network (CNN) algorithm. The dataset consisted of 12,392 images spanning four white blood cell classes: eosinophils, neutrophils, lymphocytes, and monocytes. The results indicate that the Adam optimizer achieved the best performance, with a training accuracy of 98.65% and an evaluation accuracy of 97.73%. Adam also outperformed the other optimizers in key metrics, including recall (97.43%), precision (97.42%), F1-score (97.42%), and specificity (99.11%). The AUC values for all classes exceeded 90%, demonstrating the model's exceptional ability to distinguish between different cell types. The RMSProp optimizer yielded a training accuracy of 98.63%, whereas SGD achieved a lower training accuracy of 83.46%. This study highlights the significant impact of optimizer selection on CNN performance in white blood cell image classification, providing a foundational step toward the development of more accurate medical classification systems.

KEYWORDS — Adam's Optimizer, Convolutional Neural Network, White Blood Cell Classification, RMSProp Optimizer, SGD Optimizer.

I. INTRODUCTION

White blood cells, or leukocytes, are essential components of the immune system responsible for fighting infections and diseases. Several types of white blood cells exist, including neutrophils, lymphocytes, monocytes, eosinophils, and basophils, each playing a specific role in an immune response. Leukocytes are produced in the bone marrow and circulate throughout the body via the bloodstream. The levels and proportions of different white blood cell types in the blood provide important indicators of an individual's health and assist in diagnosing various medical conditions [1]. Each white blood cell type exhibits unique characteristics in terms of color and morphology. Neutrophils appear bluish-red and typically possess three-lobed nuclei with varying shapes. Basophils display a spotted bluish appearance. Eosinophils exhibit reddish spots, monocytes have blue color with an elongated round nucleus, and lymphocytes appear pale blue with limited motility [2].

Traditionally, white blood cell classification and counting have been conducted manually by skilled operators or through fully automated cell analysis. However, manual counting methods are inefficient, time-consuming, and labor-intensive. Although automated cell analysis has been employed for white blood cell classification, these machines often impose high sample requirements and are costly, limiting their widespread adoption in healthcare facilities and hospitals [3]. The advancement of technology has facilitated the classification of white blood cell images through deep learning. Deep learning, a subset of machine learning, involves artificial neural networks composed of multiple layers. The neural network in deep learning can have many layers, enabling the extraction of complex features from input data, and making deep learning applicable in various domains, including medical image analysis such as white blood cell classification [4].

One of the prominent deep learning algorithms is the convolutional neural network (CNN), which utilizes convolutional layers to extract essential image features such as edges, textures, and patterns. These layers integrate the extracted features to reduce data dimensionality, allowing the network to identify more complex patterns and classify images into distinct categories. In addition to convolutional layers, CNNs consist of pooling and fully connected layers. Pooling layers reduce the size of feature representations and prevent overfitting through operations such as max-pooling or average-pooling. Meanwhile, fully connected layers consolidate all extracted features to generate final classification results [5].

In the implementation of CNN algorithms for image classification, optimizer selection plays a crucial role in the model training process. An optimizer is an algorithm that adjusts the model's weight and bias updates to minimize the loss function during training. Commonly used deep learning optimizers include stochastic gradient descent (SGD), adaptive moment estimation (Adam), and root mean square propagation (RMSProp), each employing different approaches to handle gradients during training [6]. SGD is computationally simple and lightweight but often requires more iterations to achieve convergence. Adam and RMSProp, on the other hand, offer advantages in convergence speed through adaptive approaches. This study aims to compare the performance of these three optimizers in white blood cell image classification, focusing on four white blood cell types: eosinophils, neutrophils, lymphocytes, and monocytes. The evaluation employs metrics such as the confusion matrix and receiver operating characteristic-area under the curve (ROC-AUC), including

accuracy, precision, recall, F1 score, and ROC-AUC curves. The dataset used in this study was sourced from Kaggle, an online dataset provider, under the name Blood Cells Images. It consists of 12,392 images representing four classes of white blood cells: eosinophils, neutrophils, lymphocytes, and monocytes [7]. By understanding the impact of each optimizer on accuracy, this research aims to provide insights into the most suitable optimizer for white blood cell classification

II. RELATED WORKS

Several prior studies have developed fully connected CNN models consisting of six convolutional layers, six pooling layers, and two fully connected layers. These models were tested using the Blood Cell Count and Detection (BCCD) dataset obtained from Kaggle, consisting of 285 images, achieving an accuracy of 96.84% [8].

Another study proposed an innovative approach to white blood cell image classification using meta-learning and color constancy methods. The meta-learning method, based on the Xception model, achieved a peak accuracy of 96.47% using the Raabin dataset comprising 10,175 images [9]. Additionally, a study employing a decision tree algorithm for classifying 167 white blood cell images obtained through hospital communication achieved an accuracy of 92.2% [10].

Other studies have applied convolutional features-support vector machines (Con-SVM) to classify white blood cell images using a dataset of 12,442 images downloaded from Kaggle, achieving an accuracy of 85.96% [11]. Meanwhile, the k-nearest neighbors (KNN) algorithm has been used to detect white blood cell abnormalities for early diagnosis of myeloproliferative neoplasm syndrome. Out of 159 test data points, 150 were correctly classified, resulting in an accuracy of 94.3% [12].

A recently conducted study developed a white blood cell classification method employing a deep dilated residual convolutional neural network (DDRNet), achieving an accuracy of 91.98% and an F1 score of 0.96, utilizing a dataset of 16,249 images from Kaggle [13].

Based on Table I, various methods for white blood cell image classification, such as fully connected CNN, meta-learning, decision tree, Con-SVM, and DDRNet, exhibit variations in accuracy and efficiency. Although some methods achieve high accuracies, such as fully connected CNN (96.84%) and Xception-based meta-learning (96.47%), challenges remain, including model complexity, sensitivity to contrast variations, and instability in handling image noise. Methods like decision trees and Con-SVM demonstrate limitations in processing complex features, resulting in lower accuracy.

This gap indicates that despite advancements in classification methods, further exploration is required to enhance computational efficiency and model robustness, particularly in medical image classification, which demands both high accuracy and stability. In this study, optimizers such as Adam, SGD, and RMSProp were selected for white blood cell image classification using CNN, as each optimizer has distinct advantages in addressing challenges during model training. Adam is known for accelerating convergence [14], RMSProp provides stability for fluctuating gradients [15], and SGD remains a common baseline due to its simplicity [16]. By comparing these three optimizers, this study aimed to identify the optimal combination that improves classification performance in terms of both accuracy and computational efficiency.

TABLE I
RESEARCH GAP ANALYSIS IN WHITE BLOOD CELL IMAGE CLASSIFICATION

Research	Method	Advantages	Disadvantages
[8]	Fully connected CNN	High accuracy (96.84%)	Complex architecture increases computational time and cost.
[9]	Meta learning dan color constancy	Accuracy of 96.47% with Meta Learning	Susceptible to contrast changes and color variations.
[10]	Decision tree	Easy to implement, accuracy of 92.2%	Less effective with noisy data or complex features.
[11]	Con-SVM	Improves generalization	Lower accuracy (85.96%) compared to other methods.
[12]	KNN	Simple, accuracy of 94.3%	Sensitive to outliers and requires careful selection of K parameters.
[13]	DDRNet	High accuracy (91.98%) and F1-score (0.96), effective for complex features	High computational demands due to architectural complexity.

Choosing the appropriate optimizer is important as it can address several limitations observed in previous methods, such as model stability in handling data variations, convergence speed, and generalization capability across different datasets. Therefore, this study focuses on optimizing CNN using these three optimizers to achieve a balance between accuracy and computational efficiency in white blood cell image classification.

III. METHODOLOGY

This study employed an experimental study strategy to test the white blood cell image classification model using CNN and to evaluate its performance. The process entailed designing the steps to be applied during the study.

The study workflow began with obtaining a dataset of white blood cell images from an online source, Kaggle. The dataset was separated into three subsets: training, testing, and validation data. This partition ensured that the model had good generalization capabilities for new data that had never been seen before and accurately measured the model's performance.

The preprocessing stage was conducted to prepare the dataset to suit the needs of the CNN model. This included balancing the dataset using the random oversampling (ROS) technique, resizing images for consistency, segmenting images to isolate white blood cells from the background, and normalizing pixel values. Normalization helped standardize pixel value distributions and reduced unwanted variations within the data.

Next, the CNN architecture was developed and trained using the training and validation data. The model was optimized using three different optimizers: Adam, RMSProp,

and SGD. Once training was completed and parameters were optimized, the testing data was used for final model evaluation. This evaluation provided an objective measure of the model's generalization ability and classification performance on new data.

Performance evaluation was conducted using a confusion matrix. Several metrics, including accuracy, precision, recall, F1 score, specificity, and ROC-AUC, were employed to assess the model's capability in accurately distinguishing between positive and negative classes.

The results and analysis section presented findings from CNN implementation by comparing the performance of the three optimizers in white blood cell image classification. The results from each process stage were discussed to provide a comprehensive overview of the model's effectiveness in classification tasks.

A. CONVOLUTIONAL NEURAL NETWORK

CNN is a prominent deep learning algorithm that has gained widespread popularity due to its ability to automatically identify relevant features without human supervision. CNN has been applied in various fields, including computer vision, speech processing, and facial recognition. Its structure is inspired by biological neurons in the human and animal brains, utilizing shared weights and local connections to efficiently process 2D input data such as image signals [17]. CNN remains the primary choice for image classification tasks in computer vision. Recent studies have introduced the Vision Transformer (ViT) architecture, integrating CNN for computer vision tasks. This study suggests that combining CNNs and transformers can improve performance in image classification [18].

Advancements in CNN architecture have significantly improved facial recognition system accuracy. One study demonstrated that CNN effectively addresses challenges such as lighting conditions, viewing angles, and facial expressions, thereby supporting the implementation of facial recognition for security and biometric authentication [19]. Figure 1 illustrates an example of CNN architecture.

As shown in Figure 1, during the feature learning stage, the network receives an input image, processes it through convolutional and pooling layers, and transforms it into numerical feature maps that represent the image. Each layer refines the image representation before passing it to the classification stage. In this stage, multiple fully connected layers receive input from the final feature maps of the previous stage. These layers process and refine the extracted features through a series of hidden layers within the neural network, then ultimately generating classification predictions for each class.

The convolutional layer is an important component of CNN, responsible for performing convolution operations on the output from previous layers. It contains a series of randomly learned filters designed to extract important features from input data. The goal is to obtain a representation of the important features of the input data, especially in images.

The pooling or subsampling layer typically follows the convolutional layer in CNN. It reduces the spatial dimensions (height and width) of the convolutional layer's output. The primary objective of pooling is to decrease the number of learnable parameters in the network, mitigating overfitting while enhancing overall network performance and classification accuracy.

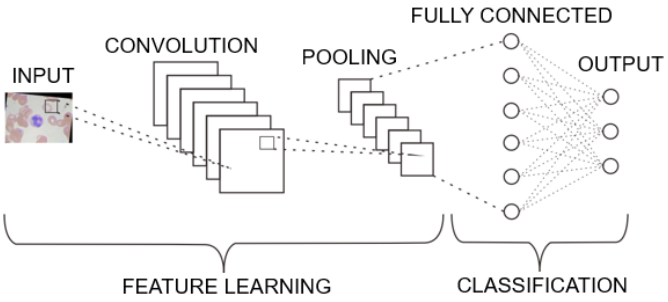


Figure 1. Convolutional neural network.

The rectified linear unit (ReLU) function is the most commonly used nonlinear function in neural networks in the current era. ReLU produces a value of 0 for all negative values of x, while for positive values of x, the value remains unchanged. This function is demonstrated in (1).

$$f\_relu\ (x) = max(0, x). \quad (1)$$

In this function, x denotes the input function, and max(0, x) represents the maximum function that selects the larger value between 0 and x.

The fully connected layer represents a prevalent component in neural networks, a notable distinction being its absence of a convolution operation during the process of generating output. This layer facilitates direct connectivity between neurons within the preceding layer, thereby establishing more intricate connections and enabling the network to discern more abstract relationships between the features manifested by activations. The fully connected layer is positioned in the final two layers of the network and employs a softmax activation function to ascertain the probability of the output based on the provided input [20].

B. ADAPTIVE MOMENT ESTIMATION (ADAM)

Adam's optimizer is one of the widely used optimization algorithms in deep learning model training. It combines the advantages of adaptive gradient (AdaGrad) and RMSProp by dynamically adjusting the exponential rates for the first (mean) and second (variance) moment estimates of the gradient to update the parameters. Adam's optimizer is particularly suitable for large data and nonstationary objective optimization with noisy and sparse gradients. In comparison to alternative optimization algorithms, Adam demonstrates superior performance in terms of rapid convergence and stability during training, particularly when confronted with problems involving numerous parameters and loss functions that may be nonconvex [21]. The utilization of Adam is also employed in the optimizations of deep learning models, particularly in the medical domain. A study demonstrated that Adam expedites convergence while preserving accuracy, even when dealing with substantial and intricate medical datasets [14]. The formulas for calculating the Adam's optimizer are stated in (2) to (6).

Estimation of the first moment (average):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t. \quad (2)$$

Estimation of the second moment (off-center variance):

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2. \quad (3)$$

Bias-corrected first-moment estimate:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}. \quad (4)$$

Bias-corrected second-moment estimate:

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}. \quad (5)$$

Parameter update:

$$\hat{\theta}_t = \theta_{t-1} - a\,\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \quad (6)$$

where a is the learning rate, β_1 and β_2 represent the decay rates for moment estimation, ϵ is a small constant for numerical stability.

C. ROOT MEAN SQUARE PROPAGATION (RMSProp)

RMSProp is an optimization algorithm that has been designed to accelerate the model training process in deep learning. It is a development of the SGD method that aims to overcome the problems of too slow learning rate decline and large fluctuations in parameter updates. RMSProp works by storing the mean square of the gradient that has been calculated at each iteration. By doing so, it can adjust the learning rate for each parameter individually [15]. In recent years, RMSProp has been identified as a leading method in various deep-learning applications. The study has demonstrated its superiority to Adam, particularly in nonconvex optimization tasks, making it well-suited for complex architectures and challenging datasets [22]. The formulas for calculating the RMSProp optimizer are presented in (7) and (8).

Moving average of the squared gradient:

$$v_t = \beta_2 v_{t-1} + (1 - \beta) g_t^2 \quad (7)$$

Parameter update:

$$\theta_t = \theta_{t-1} - a\,\frac{g_t}{\sqrt{v_t}+\epsilon} \quad (8)$$

where α denotes the learning rate, β defines the decay rate for the moving average of the squared gradient, and ϵ serves as a small constant for numerical stability.

D. STOCHASTIC GRADIENT DESCENT (SGD)

SGD is a simple yet highly effective optimizer in machine learning. It updates parameters using a single randomly selected example from the dataset per iteration, significantly reducing computational time and memory usage compared to the batch gradient descent method, which processes the entire dataset before updating parameters. While this approach accelerates training, it also introduces noise into gradient estimation, leading to fluctuations in the convergence path and potential instability. However, these fluctuations can help escape saddle points and discover better solutions [16]. Recent studies highlight that variations of SGD, such as momentum-based SGD and adaptive learning rate techniques, further enhance its effectiveness across deep learning applications, including image recognition and object detection [23]. The mathematical formulation of the SGD optimizer is given in equation (9).

Parameter update:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (9)$$

where α represents the machine learning, and g_t denotes the gradient of the objective function with respect to the parameters at time step t.

E. IMAGE SEGMENTATION

Image segmentation is a fundamental process in computer vision technology, designed to partition an image into distinct regions based on specific characteristics. This process facilitates the identification and extraction of target objects, serving as a crucial bridge between image processing and analysis. Segmentation enables the separation of the object of interest (target) from its background, which has a wide range of applications in fields such as medical image processing, pattern recognition, and artificial intelligence [24]. In medical image processing, deep learning-based segmentation methods have improved detection accuracy in lung cancer diagnosis from computed tomography (CT) images [25]. Moreover, segmentation is also instrumental in applications such as road mapping in satellite imagery, and provides better results in automated navigation systems [26].

F. CONFUSION MATRIX

A confusion matrix is an $N \times N$ table used in classification tasks to assess model performance, where N represents the number of predicted classes. By comparing the model's predictions with the actual values, the confusion matrix provides a detailed overview of classification accuracy and highlights areas requiring improvement [27]. The matrix comprises four key components: true positive (TP), false negative (FN), false positive (FP), and true negative (TN). These values serve as the foundation for calculating various evaluation metrics, including accuracy, precision, recall, and F1 score, which collectively quantify the model's overall performance.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (10)$$

Accuracy is the ratio of the number of correct predictions (both positive and negative) to the total number of predictions.

$$Presision = \frac{TP}{TP+FP}. \quad (11)$$

Precision measures the accuracy of the model in predicting the positive class. The higher the precision, the fewer false positive predictions.

$$Recall = \frac{TP}{TP+FN}. \quad (12)$$

Recall or sensitivity is the ability of the model to capture all positive samples.

$$F1\ Score = \frac{2\ x\ precision\ x\ recall}{precision+recall}. \quad (13)$$

F1 score is the harmonic mean of precision and recall. This metric is used when balancing between precision and recall.

$$Specificity = \frac{TN}{TN+FP}. \quad (14)$$

Specificity measures the ability of the model to correctly predict the negative class.

G. ROC-AUC

The ROC-AUC curve is used to measure the ability of a classification model to differentiate between classes across various threshold scenarios [28]. This evaluation to calculate the AUC value results in (15)

$$AUC = \frac{1}{2}(Sensitivity/Recall + Specificity). \quad (15)$$

AUC is a measure of the model's ability to distinguish between positive and negative classes at various thresholds..

IV. RESULTS AND DISCUSSION

This section presents the study findings from the implementation of a CNN algorithm by comparing three

optimizers for white blood cell image classification. The process flow of each stage is explained as follows.

A. DATA SOURCE

The first step in the data source stage involved collecting datasets from an online dataset provider, namely Kaggle. Figure 2 illustrates four sample images of white blood cells. Based on Figure 2, this dataset comprises a total of 12,392 images, including 3,120 images for the eosinophil class, 3,103 images for the lymphocyte class, 3,098 images for the monocyte class, and 3,123 images for the neutrophil class.

After data collection, the dataset was divided into three subsets: 80% for training data, 10% for validation data, and 10% for testing data. This proportion was selected based on previous study findings, which demonstrated that this split could achieve an accuracy of up to 95.69% [29]. Figure 3 illustrates the distribution of the dataset split.

Figure 3 was generated using a program that partitions the white blood cell image dataset into three main subsets: training data (blue), validation data (orange), and testing data (green). The x-axis represents the dataset classes, while the y-axis represents the total number of data points. For the eosinophil class, there were 2,443 images for training, 303 for validation, and 295 for testing. The lymphocyte class contained 2,467 images for training, 313 for validation, and 321 for testing. The monocyte class comprised 2,324 images for training, 312 for validation, and 304 for testing. Finally, the neutrophil class included 2,482 images for training, 287 for validation, and 295 for testing.

B. PREPROCESSING

In this stage, preprocessing was performed to prepare the dataset before proceeding to the modeling phase. The first step involved balancing the dataset to address class imbalance. Figure 4(a) presents a visualization of the white blood cell image dataset, which initially exhibited class imbalance. To mitigate this issue, a data balancing process was conducted using the random oversampling technique. This technique addresses class imbalance in classification tasks by increasing the number of samples in minority classes through random duplication with replacement [30].

Figure 4 illustrates the dataset condition before and after data balancing. After the balancing process, the results are displayed in Figure 4(b), which shows that the sample count for all classes has been balanced.

However, it is important to note that oversampling techniques such as random oversampling can introduce potential biases into the model. This bias arises because random duplication with replacement may increase the likelihood of overfitting the minority class, resulting in model learning patterns that are less general [31].

Following dataset balancing, image resizing was performed to reduce image dimensions before inputting them into the CNN model. Figure 5 visualizes the resizing process, where images were resized from 320 × 240 to 120 × 120. This step ensures that input images maintain consistent dimensions, thereby reducing computational complexity [32].

The next step involved image segmentation, which was carried out to separate white blood cells from the background using OpenCV. The process began with thresholding to highlight objects based on pixel intensity, followed by dilation to enlarge object areas and erosion to remove noise. A second dilation step was applied to refine object shapes. Subsequently, the Canny edge detection algorithm was employed to detect

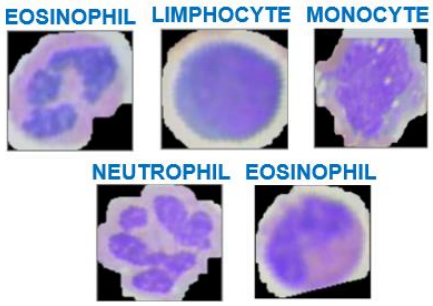


Figure 2. Dataset samples.

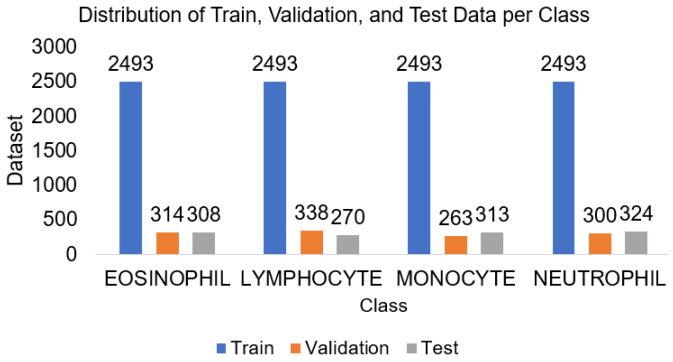


Figure 3. Dataset sharing distribution.

object edges, followed by contour detection to identify object boundaries. Bounding boxes were then created around the contours to highlight detected areas. The final outcome was an extracted white blood cell image, as shown in Figure 6.

Figure 6 illustrates the segmented areas from the previous process, ensuring that the image is more focused on the segmented object, revealing clearer details after the object has been separated from the background. Once all segmentation processes were completed, the final segmentation results were applied to the entire white blood cell image dataset.

Figure 7 presents five examples of successfully segmented and classified white blood cell images. Each image represents different types of white blood cells—lymphocytes, neutrophils, eosinophils, and monocytes—that have undergone segmentation. The segmentation process effectively isolated white blood cells from the background, enhancing the visibility of each cell's shape and structure. This preprocessing step is crucial before proceeding with classification using the CNN model.

After the data balancing, image resizing, and segmentation processes were completed, the next step was pixel value normalization. Normalization is an essential step to ensure that input data maintains uniform values, particularly in terms of consistent pixel sizes. Pixel value normalization was performed on the white blood cell image dataset used for training, validation, and testing. This was achieved by dividing each pixel value in the images by 255.0, thereby converting the pixel value range from [0.255] to [0.1].

C. CNN MODEL

At this stage, the development of a CNN model for white blood cell image classification was conducted. The CNN architecture employed for this classification task is presented in Table II.

As shown in Table II, the CNN architecture designed in this study was specifically structured for white blood cell image classification while considering both efficiency and

Jurnal Nasional Teknik Elektro dan Teknologi Informasi
Volume 14 Number 1 February 2025

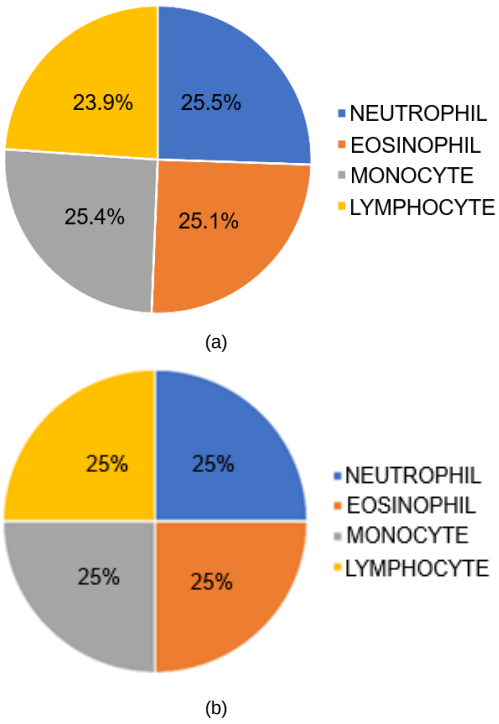


Figure 4. Data distribution comparison, (a) before balancing, (b) after balancing.

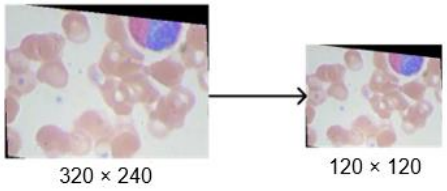


Figure 5. Image resizing process.

effectiveness. The initial layer consists of a Conv2D layer with 16 filters of size 3 × 3, which aims to extract fundamental features such as edges and simple patterns from input images of 120 × 120 pixels. Subsequently, each convolutional layer was followed by max pooling to reduce data dimensionality, thereby decreasing the number of parameters and mitigating the risk of overfitting while enhancing computational efficiency. The separable convolution layer was utilized to capture more complex patterns with fewer parameters, ensuring a balance between accuracy and computational cost. To stabilize training, batch normalization layers were applied after certain convolutional layers, facilitating faster training convergence and gradient stability. Additionally, dropout was implemented before the fully connected layer to prevent overfitting by randomly deactivating neurons during training. The selection of these architectural components reflects the balance between complexity and generalization capability for the white blood cell image classification task.

Following the architectural design, the next step involved training the model using both training and validation data with the previously built model. At this stage, adjustments were made to the use of optimizers on the parameters used. The development of the study method was carried out using Kaggle Notebooks, with the T4 GPU provided by the platform utilized to accelerate the model training process. The software tools utilized included Python 3.8, TensorFlow and Keras 2.8 for CNN model development, and several supporting libraries such as NumPy 1.21, Pandas 1.3, Matplotlib, and Seaborn for data analysis and visualization. The optimizers utilized in this study

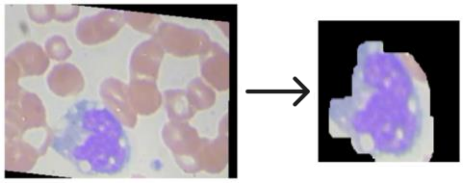


Figure 6. Image segmentation process.

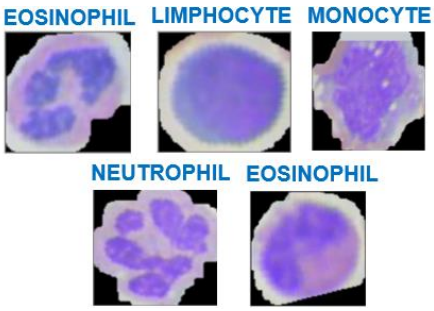


Figure 7. Sample of five images from the segmentation process.

TABLE II
CNN ARCHITECTURE

Layer (Type)	Output Form	Parameter#
conv2d	(None, 120, 120, 16)	448
max_pooling2d	(None, 60, 60, 16)	0
separable_conv2d	(None, 60, 60, 32)	688
batch_normalization	(None, 60, 60, 32)	128
max_pooling2d_1	(None, 30, 30, 32)	0
separable_conv2d_2	(None, 30, 30, 64)	2400
batch_normalization_1	(None, 30, 30, 64)	256
max_pooling2d_2	(None, 15, 15, 64)	0
separable_conv2d_4	(None, 15, 15, 128)	8896
batch_normalization_2	(None, 15, 15, 128)	512
max_pooling2d_3	(None, 7, 7, 128)	0
dropout	(None, 7, 7, 128)	0
separable_conv2d_6	(None, 7, 7, 256)	34176
batch_normalization_3	(None, 7, 7, 256)	1024
max_pooling2d_4	(None, 3, 3, 256)	0
Flatten	(None, 2304)	0
dense_ (Dense)	(None, 512)	1180160
dense_1 (Dense)	(None, 128)	65664
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 4)	260
Total params: 1.397.028		
Trainable params: 1.396.068		
Nontrainable params: 960		

included RMSProp, SGD, and Adam. The batch size was set to 32 to balance computational efficiency, learning stability, and model generalization capability [33]. A learning rate of 0.0001 was adopted to ensure stable and gradual parameter updates. The findings indicate that this learning rate effectively prevented divergence or excessively slow learning [34]. The number of epochs was set to 30, which was deemed sufficient to achieve convergence without overfitting, particularly given the implementation of regularization techniques such as dropout and batch normalization.

As presented in Table III, the results indicate that the Adam optimizer yielded the best performance, achieving a training accuracy of 98.65% and a validation accuracy of 97.94%, along with the lowest loss values of 4.35% for training data and 6.35% for validation data. RMSProp also demonstrated competitive performance but was slightly inferior to Adam, with a training

TABLE III
TRAINING, VALIDATION AND LOSS FOR EACH OPTIMIZER

Optimizer	Accuracy (%)	Loss (%)	Validation Accuracy (%)	Validation Loss (%)
RMSProp	98.63	5.08	96.87	13.02
SGD	83.46	40.39	87.57	30.73
Adam	98.65	4.35	97.94	6.35

accuracy of 98.63%, a validation accuracy of 96.87%, and a higher validation loss compared to Adam. The SGD optimizer exhibited the lowest performance among the three, with a training accuracy of 83.46% and a validation accuracy of 87.57%, as well as substantially higher loss values of 40.39% for training data and 30.73% for validation data. In terms of computational efficiency, the training time required for each optimizer was as follows: SGD took 3.82 minutes, Adam required 3.59 minutes, and RMSProp needed 3.82 minutes. Based on these results, the Adam optimizer provided the best overall performance with superior computational efficiency compared to the other optimizers.

Following the optimizer evaluation, the optimal CNN model was identified using the Adam optimizer with a batch size of 32, a learning rate of 0.0001, and 30 epochs. The performance of the optimal CNN model, comparing accuracy and loss metrics between training and validation data, is illustrated in Figure 8.

Figure 8, generated from program processing, depicts the accuracy and loss trends throughout model training. The accuracy graph indicates that training accuracy (blue line) steadily increases from 0.3 to nearly 1.0 by epoch 20, while validation accuracy (orange line) rises rapidly during the initial epochs (0–5) before stabilizing near the training accuracy. This pattern suggests strong model generalization without signs of overfitting. In the loss graph, training loss (blue line) sharply decreases in the early training phase (0–5 epochs), reflecting the optimizer's effectiveness in minimizing errors. Validation loss (orange line) follows a similar trend with minor fluctuations in the early epochs (3–5) before stabilizing, indicating weight adjustments in response to validation data. The consistent loss reduction approaching zero by the final epochs signifies effective model convergence. Overall, these trends demonstrate that the model achieves high accuracy and low loss with optimal performance, supported by the effectiveness of the optimizer in managing the training.

D. EVALUATION

This stage is conducted to assess the model's performance in predicting test data that has never been seen before. The optimal model, obtained from parameter tuning experiments, was tested to evaluate its generalization capability. To compute accuracy, precision, recall, F1 score, and specificity values, a confusion matrix evaluation was conducted using the RMSProp, Adam, and SGD optimizers, as illustrated in Figure 9.

Figure 9 was generated through computational processing, visualizing the model's classification results in the form of a confusion matrix. Based on Figure 9, the Adam optimizer demonstrates the best performance in classifying white blood cells. Adam exhibits high accuracy with minimal misclassification errors, particularly in neutrophils, which are occasionally misclassified as eosinophils. This can be attributed to Adam's use of momentum and adaptive weighting, enhancing stability and accelerating convergence despite

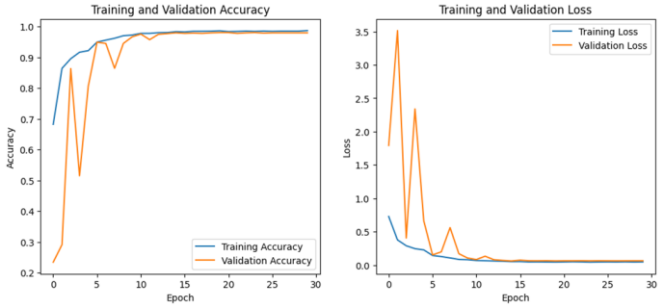


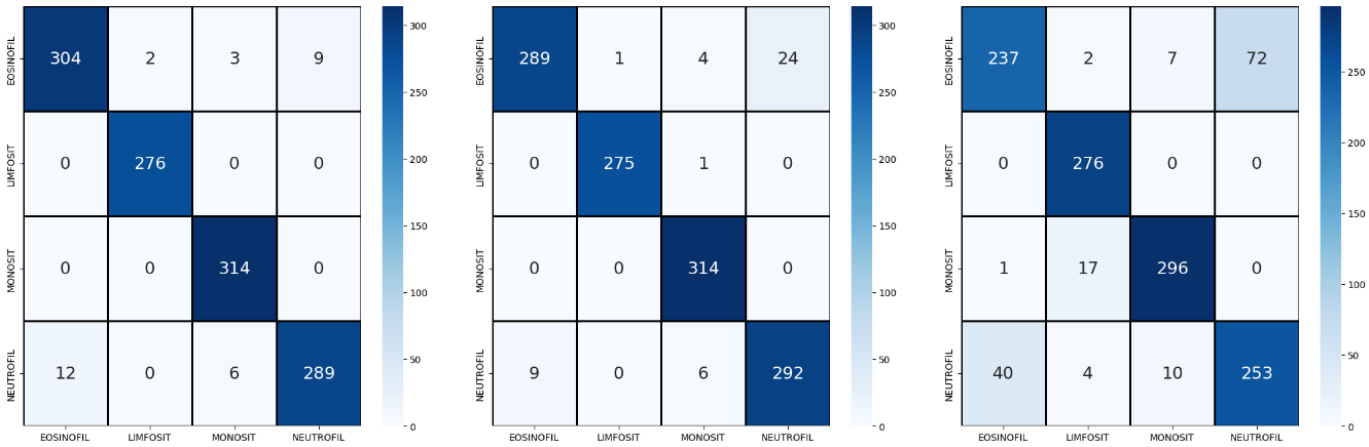
Figure 8. Accuracy and loss graph results.

indistinct inter-class features. RMSProp also performs well but exhibits slightly higher misclassification in eosinophils due to its focus on adaptive learning rate adjustments. While this contributes to stability, it is less effective in distinguishing classes with similar features. Conversely, SGD shows the highest misclassification rate, particularly in differentiating eosinophils and neutrophils, as its weight update mechanism is simpler and more sensitive to learning rate settings, making it less effective in handling visually similar classes. After extracting the values of TP, TN, FP, and FN from the confusion matrix, accuracy, precision, recall, F1 score, and specificity were calculated using (10) to (14).

According to the model evaluation results in Table IV, the Adam optimizer provides the best performance among the three tested optimizers. Adam achieves an accuracy of 97.37%, indicating that the model correctly classified most images, although accuracy alone is insufficient for comprehensive performance evaluation. A recall of 97.43% demonstrates the model's effectiveness in correctly identifying all white blood cells, minimizing the risk of false negatives, which could be crucial in medical applications, such as failure to diagnose severe conditions. A precision of 97.42% indicates that the model accurately classifies white blood cells into the correct category, reducing false positives that could lead to overdiagnosis and unnecessary medical procedures. With an F1-score of 97.42%, the model maintains a well-balanced trade-off between recall and precision, which is crucial for avoiding misdiagnoses, including underdiagnosis and overdiagnosis. Finally, a specificity value of 99.11% demonstrates the model's ability to avoid misclassifying negative cases, further reducing false positives and reinforcing the reliability of automated diagnostic results for medical professionals.

Subsequently, an evaluation using the ROC AUC curve was conducted to assess the model's capability to distinguish between positive and negative classes based on the area under the ROC curve. The ROC curve illustrates the model's performance for each class by comparing the true positive rate (TPR) and the false positive rate (FPR). The ROC curve results for the three optimizers are presented in Figure 10.

Figure 10 was generated through a computational process that visualizes the model evaluation metrics using the ROC curve and AUC values. Based on Figure 10, the y-axis represents the TPR, which indicates the proportion of correctly identified positive cases from the total actual positive class instances. The x-axis represents the FPR, denoting the proportion of misclassified negative instances from the total actual negative class instances. The blue, green, red, and purple lines illustrate the model's performance at different threshold values, while the dashed black line represents the performance

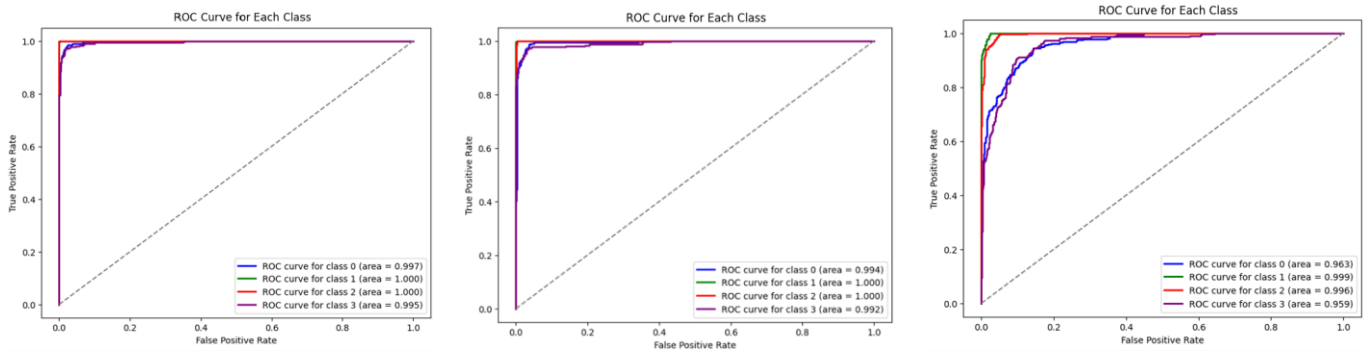


(a) (b) (c)

Figure 9. Confusion matrix, (a) Adam optimizer, (b) RMSProp optimizer, (c) SGD optimizer.

TABLE IV
MODEL EVALUATION OF EACH OPTIMIZER

Optimizer	Accuracy	Recall	Precision	F1 Score	Specificity
RMSProp	0.9630	0.9641	0.9641	0.9637	0.9875
SGD	0.8741	0.8780	0.8749	0.8750	0.9579
Adam	0.9737	0.9743	0.9742	0.9742	0.9911



(a) (b) (c)

Figure 10. ROC Curve and AUC Value, (a) Adam optimizer, (b) RMSProp optimizer, (c) SGD optimizer.

TABLE V
COMPARISON OF RESEARCH RESULTS

Research	Method	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	Specificity (%)	AUC
This study	CNN dan optimizer Adam	97.37	97.42	97.43	97.42	99.11	99.00
[8]	CNN fully connected	96.84	96.39	96.26	-	97.35	-
[9]	Meta learning dan color constancy	96.47	95.61	95.61	95.61	-	-
[10]	Decision tree	92.20	-	-	-	-	-
[11]	Con-SVM	85.96	-	-	-	-	-
[12]	KNN	94.30	-	-	-	-	-
[13]	DDRNet	91.98	-	-	96.00	-	-

of a random classifier. The ROC analysis results for the four classes of white blood cells indicate that the Adam optimizer demonstrated the best performance, achieving an average AUC of 1.00, signifying excellent class differentiation capability. The RMSProp optimizer ranked second with an average AUC of 0.992, reflecting strong performance but slightly lower than Adam. The SGD optimizer exhibits the lowest average AUC of 0.959, indicating suboptimal performance in distinguishing between the classes compared to Adam and RMSProp. Therefore, Adam was determined to be the most effective optimizer for white blood cell classification.

From a practical perspective, the Adam optimizer is the most suitable choice for automated diagnostic systems in medical facilities with adequate computational resources. However, in resource-constrained environments, RMSProp or even SGD may be considered as more efficient alternatives. For real-world applications, further model training using data from diverse medical institutions could enhance generalization and reliability, ensuring the system's optimal contribution to accurate and trustworthy medical diagnostics.

Based on the study's findings, the classification of white blood cell images using the Adam optimizer with CNN

architecture demonstrated superior performance compared to previous studies employing different methods. Table V provides a comparative analysis between this study and the previous study.

Table V indicates that the combination of CNN and the Adam optimizer achieved the highest performance in white blood cell image classification compared to earlier methodologies. With accuracy, recall, precision, F1 score, and AUC values of 97.37%, 97.43%, 97.42%, 97.42%, and 99%, respectively, this model outperformed other approaches, including fully connected CNN, which was reported in prior studies with an accuracy of 96.84% [8], as well as meta-learning and color constancy methods, which achieved an accuracy of 96.47% [9]. Lower performance was also observed in classical methods such as decision trees with an accuracy of 92.2% [10], Convolutional SVM (Con-SVM) with 85.96% [11], and KNN with 94.3% [12]. Another study utilizing DDRNet reported an accuracy of 91.98% [13]. Although DDRNet demonstrated a strong ability to handle complex features, its overall performance remained inferior to CNN with the Adam optimizer in this study. These findings highlight that the CNN-Adam combination provides a more adaptive solution for medical image analysis. This study confirms that this combination represents the optimal approach for white blood cell image classification.

V. CONCLUSION

This study has found that among the three tested optimizers—Adam, RMSProp, and SGD—the Adam optimizer demonstrated the best performance in training the CNN model for white blood cell classification. Based on the evaluation using a confusion matrix, Adam achieved an accuracy of 97.37%, recall of 97.43%, precision of 97.42%, F1 score of 97.42%, and specificity of 99.11%. These values indicate that Adam outperformed the other optimizers across all evaluation metrics. Compared to RMSProp, Adam exhibited a performance advantage with an accuracy difference of 1.07%, recall of 1.02%, precision of 1.01%, F1 score of 0.95%, and specificity of 0.36%. Meanwhile, the performance gap between Adam and SGD was significantly larger, with an accuracy difference of 10.96%, recall of 9.63%, precision of 9.93%, F1 score of 9.92%, and specificity of 3.32%. Additionally, the ROC-AUC curve analysis showed that the values for each class exceeded 90%, indicating that the model effectively differentiates between classes and falls within the category of excellent classification. Based on these findings, implementing the model into a web-based system is recommended, enabling image uploads and real-time classification to support practical applications in the medical field. However, this study has certain limitations, particularly in evaluation, as further validation with external datasets and testing in real-world clinical environments are required to ensure the model's effectiveness in actual clinical applications.

CONFLICTS OF INTEREST

The authors declare that during the research and writing of this scientific article entitled "Comparison of Optimizer Use in White Blood Cell Classification Employing CNN," the writing team has no conflict of interest with any party.

AUTHORS' CONTRIBUTIONS

Conceptualization, Dede Kurniadi and Rifky Muhammad Shidiq; methodology, Dede Kurniadi; software, Rifky Muhammad Shidiq; validation, Dede Kurniadi and Asri Mulyani; writing-original draft, Dede Kurniadi and Rifky Muhammad Shidiq; writing-reviewing and editing, Dede Kurniadi, Rifky Muhammad Shidiq, and Asri Mulyani; visualization, Rifky Muhammad Shidiq; supervision, Dede Kurniadi; funding, Dede Kurniadi (Institut Teknologi Garut).

ACKNOWLEDGMENT

The present research was made possible by the financial support and resources provided by the Institut Teknologi Garut. In addition, gratitude is extended to all individuals and entities who have contributed to and supported this endeavor, either directly or indirectly, throughout the research process.

REFERENCES

[1] B.J. Bain, "Performing a blood count," in *Blood Cells: A Practical Guide*, 6th ed. West Sussex, United Kingdom: John Wiley & Sons Ltd, 2022, ch. 2, pp. 17–63.

[2] W. King, K. Toler, and J. Woodell-May, "Role of white blood cells in blood- and bone marrow-based autologous therapies," *Biomed Res. Int.*, vol. 2018, no. 1, pp. 1–8, Jul. 2018, doi: 10.1155/2018/6510842.

[3] N. Dong, M. Zhai, J. Chang, and C. Wu, "A self-adaptive approach for white blood cell classification towards point-of-care testing," *Appl. Soft Comput.*, vol. 111, pp. 1–13, Nov. 2021, doi: 10.1016/j.asoc.2021.107709.

[4] M.A. Ali, F. Dornaika, and I. Arganda-Carreras, "White blood cell classification: Convolutional neural network (CNN) and vision transformer (ViT) under medical microscope," *Algorithms*, vol. 16, no. 11, pp. 1–17, Nov. 2023, doi: 10.3390/a16110525.

[5] T.A. Sadoon and M.H. Ali, "An overview of medical images classification based on CNN," *Int. J. Curr. Eng. Technol.*, vol. 10, no. 6, pp. 900–905, Nov./Dec. 2020, doi: 10.14741/ijcet/v.10.6.1.

[6] E.M. Dogo, O.J. Afolabi, and B. Twala, "On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification," *Appl. Sci.*, vol. 12, no. 23, pp. 1–36, Dec. 2022, doi: 10.3390/app122311976.

[7] P. Mooney, "Blood cell images." Kaggle. Access date: 13-Mar-2024. [Online]. Available: https://www.kaggle.com/datasets/paultimothymooney/blood-cells

[8] K.G. Kannan *et al*. "Classification of WBC cell classification using fully connected convolution neural network," *J. Phys., Conf. Ser.*, vol. 2466, No. 1, Apr. 2023, Art. No 012033, doi: 10.1088/1742-6596/2466/1/012033.

[9] E. Rivas-Posada, M.I. Chacon-Murguia, J.A. Ramirez-Quintana, and C. Arzate-Quintana, "Classification of leukocytes using meta-learning and color constancy methods*," J. Ilmiah Tek. Elekt. Komput. Inform.,* vol. 8, no. 4, pp. 486–499, Dec. 2022, doi: 10.26555/jiteki.v8i4.25192.

[10] C. Panjaitan, Y. Panjaitan, D. Sitanggang, and S.W. Tarigan, "Image processing for detection of dengue virus," *J. Sis. Inf. Ilmu Komput. Prima*, vol. 7, no. 2, pp. 26–34, Feb. 2024, doi: 10.34012/jurnalsisteminformasidanilmukomputer.v7i2.4799.

[11] A. Ekiz, K. Kaplan, and H.M. Ertunç, "Classification of white blood cells using CNN and Con-SVM," in *2021 29th Signal Process. Commun. Appl. Conf. (SIU)*, 2021, pp. 1–4, doi: 10.1109/SIU53274.2021.9477962.

[12] Z.E. Fitri, L.N.Y. Syahputri, and A.M.N. Imron, "Classification of white blood cell abnormalities for early detection of myeloproliferative neoplasms syndrome based on k-nearest neighbor," *Sci. J. Inform.*, vol. 7, no. 1, pp. 136–142, May 2020, doi: 10.15294/sji.v7i1.24372.

[13] M. Jawahar, L.J. Anbarasi, S. Narayanan, and A.H. Gandomi, "An attention-based deep learning for acute lymphoblastic leukemia classification," *Sci. Rep.*, vol. 14, pp. 1–20, Jul. 2024, doi: 10.1038/s41598-024-67826-9.

[14] Y. Shao *et al.*, "An improved BGE-Adam optimization algorithm based on entropy weighting and adaptive gradient strategy," *Symmetry*, vol. 16, no. 5, pp. 1–16, May 2024, doi: 10.3390/sym16050623.

[15] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, and S. Elmougy, "Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning," *Sci. Rep.*, vol. 13, pp. 1–16, May 2023, doi: 10.1038/s41598-023-35663-x.

[16] Y. Tian, Y. Zhang, and H. Zhang, "Recent advances in stochastic gradient descent in deep learning," *Mathematics*, vol. 11, no. 3, pp. 1–23, Feb. 2023, doi: 10.3390/math11030682.

[17] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *J. Big Data*, vol. 8, pp. 1–74, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[18] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th Int. Conf. Learn. Represent.*, 2021, pp.1–22.

[19] I. Adjabi, A. Ouahabi, A. Benzaoui, and A. Taleb-Ahmed, "Past, present, and future of face recognition: A review," *Electronics*, vol. 9, no. 8, pp. 1–52, Aug. 2020, doi: 10.3390/electronics9081188.

[20] W. Di, A. Bhardwaj, and J. Wei, *Deep Learning Essentials*. Birmingham, United Kingdom: Packt, 2018.

[21] Y. Wang, Z. Xiao, and G. Cao, "A convolutional neural network method based on Adam optimizer with power-exponential learning rate for bearing fault diagnosis," *J. Vibroengineering*, vol. 24, no. 4, pp. 666–678, Jun. 2022, doi: 10.21595/jve.2022.22271.

[22] Q. Tong, G. Liang, and J. Bi, "Calibrating the adaptive learning rate to improve convergence of ADAM," *Neurocomputing*, vol. 481, pp. 333–356, Apr. 2022, doi: 10.1016/j.neucom.2022.01.014.

[23] S. Nagendram *et al.*, "Stochastic gradient descent optimisation for convolutional neural network for medical image segmentation," *Open Life Sci.*, vol. 18, no. 1, pp. 1–15, Aug. 2023, doi: 10.1515/biol-2022-0665.

[24] Z. Wang, E. Wang, and Y. Zhu, "Image segmentation evaluation: A survey of methods," *Artif. Intell. Rev.*, vol. 53, pp. 5637–5674, Dec. 2020, doi: 10.1007/s10462-020-09830-9.

[25] Y. Said, A.A. Alsheikhy, T. Shawly, and H. Lahza, "Medical images segmentation for lung cancer diagnosis based on deep learning architectures," *Diagnostics*, vol. 13, no. 3, pp. 1–15, Feb. 2023, doi: 10.3390/diagnostics13030546.

[26] V. Chaudhary, P.K. Buttar, and M.K. Sachan, "Satellite imagery analysis for road segmentation using U-Net architecture," *J. Supercomput.*, vol. 78, pp. 12710–12725, Jul. 2022, doi: 10.1007/s11227-022-04379-6.

[27] I. Markoulidakis *et al.*, "Multiclass confusion matrix reduction method and its application on net promoter score classification problem," *Technologies*, vol. 9, no. 4, pp. 1–22, Dec. 2021, doi: 10.3390/technologies9040081.

[28] Ş.K. Çorbacıoğlu and G. Aksel, "Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value," *Turkish J. Emerg. Med.*, vol. 23, no. 4, pp. 195–198, Oct. –Dec. 2023, doi: 10.4103/tjem.tjem_182_23.

[29] V.R. Joseph, "Optimal ratio for data splitting," *Stat. Anal. Data Min., ASA Data Sci. J.*, vol. 15, no. 4, pp. 531–538, Aug. 2022, doi: 10.1002/sam.11583.

[30] T. Wongvorachan, S. He, and O. Bulut, "A comparison of undersampling, oversampling, and SMOTE methods for dealing with imbalanced classification in educational data mining," *Information*, vol. 14, no. 1, pp. 1–15, Jan. 2023, doi: 10.3390/info14010054.

[31] C. Yang *et al.*, "Impact of random oversampling and random undersampling on the performance of prediction models developed using observational health data," *J. Big Data*, vol. 11, pp. 1–18, Jan. 2024, doi: 10.1186/s40537-023-00857-7.

[32] A. Semma *et al.*, "Writer identification: The effect of image resizing on CNN performance," in *6th Int. Conf. Smart City Appl.*, 2021, pp. 501–507, doi: 10.5194/isprs-archives-XLVI-4-W5-2021-501-2021.

[33] T.A.A.H. Kusuma, K. Usman, and S. Saidah, "People counting for public transportations using you only look once method," *J. Tek. Inform.*, vol. 2, no. 1, pp. 57–66, Jun. 2021, doi: 10.20884/1.jutif.2021.2.2.77.

[34] T.S. Nabila and A. Salam, "Classification of brain tumors by using a hybrid CNN-SVM model," *J. Appl. Inform. Comput.*, vol. 8, no. 2, pp. 241–247, Dec. 2024, doi: 10.30871/jaic.v8i2.8277.