Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 10 No. 4 (2026) 1133 - 1143 e-ISSN: 2580-0760

Comparative Analysis of CNN Architectures and Optimizers for ALL Classification

Sugiyarto Surono[1], Fanita Damayanti[2*], Frenky Riski Gilang Pratama[3] , Attarik Mohammad[4], Maretta Mia Audina[5], Marfungah Wati[6]
[1,2,3,4,5,6]Mathematics, Faculty of Applied Science and Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

*2200015040@webmail.uad.ac.id

Abstract

Deep learning has become an important approach for medical image analysis, particularly for automated image classification. Among its various architectures, Convolutional Neural Networks (CNNs) are frequently employed to extract and distinguish visual patterns from medical images. However, the performance of a CNN may vary depending on the optimization algorithm used during the training process. This study investigates the effect of four optimization methods on different CNN architectures for the classification of Peripheral Blood Smear (PBS) images. A publicly available Kaggle dataset comprising 3,242 images and four diagnostic categories was used in the experiments. Six architectures, namely ResNet50V2, Xception, EfficientNetB2, EfficientNetB4, EfficientNetV2-S, and EfficientNetV2-M, were evaluated in combination with Adam, AdamW, NAdam, and RMSprop. Before training, the images underwent resizing, normalization, and augmentation procedures. The dataset was subsequently partitioned into training, validation, and test subsets using proportions of 80%, 10%, and 10%, respectively. Classification performance was analyzed using accuracy, loss, and confusion matrix-based evaluation. Among all tested combinations, Xception with RMSprop produced the best result, obtaining 99.69% accuracy and a loss value of 0.0194. Conversely, the lowest result was observed for ResNet50V2 with RMSprop, which achieved 93.85% accuracy. The experimental findings indicate that the optimizer can substantially influence CNN classification performance, and the effectiveness of an optimization method is dependent on the architecture with which it is combined.

Keywords: CNN; deep learning; optimizer; RMSProp; Xception

How to Cite: S. Surono, F. Damayanti, F. R. G. . Pratama, A. . Mohammad, M. M. Audina, and M. . Wati, "Comparative Analysis of CNN Architectures and Optimizers for ALL Classification", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 10, no. 4, pp. 1133–1143, Aug. 2026, doi: 10.29207/resti.v10i7506.
Permalink/DOI: https://doi.org/10.29207/resti.v10i7506

Received: January 19, 2026
Accepted: April 12, 2026
Available Online: August 23, 2026

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

Deep learning has emerged as a powerful computational paradigm for analyzing complex data and has been increasingly adopted in computer vision applications [1]. Unlike conventional approaches that often depend on manually engineered features, deep learning algorithms can learn meaningful representations directly from the input data, enabling them to capture complex patterns without requiring explicit feature construction [2], [3]. This capability has contributed to the widespread adoption of deep learning for numerous visual analysis problems, including image classification, object detection, and image segmentation. In many of these applications, deep learning methods have demonstrated superior predictive capabilities compared with traditional machine learning techniques [4], [5].

Within the deep learning domain, Convolutional Neural Networks (CNNs) are particularly suitable for processing visual information because their structure enables the extraction of spatial patterns from image data [6]. A CNN processes an image through a sequence of computational operations, including convolution, pooling, and nonlinear activation, allowing hierarchical visual information to be learned from the input. This characteristic makes CNN architectures well suited for tasks involving two-dimensional image data [7]. CNNs learn visual representations in a layered manner, where early layers capture basic patterns while deeper layers encode more abstract and object-related information [8]. This advantage makes CNN the main foundation in the development of modern image classification systems [9].

As the complexity of data increases and the demands for high accuracy increase, a range of deeper and more efficient CNN architectures have been developed [10]. ResNet introduced the concept of residual learning that allows for deep network training by overcoming the vanishing gradient problem [11]. Furthermore, Xception utilizes depthwise separable convolution to improve computing efficiency without sacrificing performance [12]. The EfficientNet family [13], including EfficientNetB2 and EfficientNetB4, carries a compound scaling strategy that systematically adjusts the depth, network width, and resolution of inputs resulting in optimal efficiency and accuracy ratios [14]. Subsequent developments on EfficientNetV2, such as EfficientNetV2-M and EfficientNetV2-S, focused on training speed and convergence stability, particularly on large and complex datasets [15].

The training outcome of a CNN is influenced not only by its architecture but also by the optimization strategy adopted during parameter learning [16]. An optimizer determines how the network weights are updated in response to the gradients generated during backpropagation. Consequently, the selected optimization method can affect the speed of convergence, training stability, and the model's generalization to previously unseen samples [17]. Among the optimization techniques commonly employed in deep learning are Adam, AdamW, NAdam, and RMSprop [18]. Adam has become a popular choice because it combines momentum estimation with adaptive step-size control, allowing parameter updates to be adjusted according to the observed gradients [19]. In contrast, AdamW modifies the regularization procedure by treating weight decay independently from the gradient-based update, which can provide improved control over model generalization [20]. NAdam extends the Adam optimization scheme by incorporating Nesterov momentum into its update mechanism, while RMSprop determines parameter-specific learning rates using an exponentially averaged estimate of squared gradients [21]. Despite their widespread use, these optimizers do not necessarily produce identical results across different CNN models. Their effectiveness may vary according to the architectural characteristics of the network and the properties of the dataset being analyzed [22].

In the classification of medical images, the challenges faced are often more complex than natural images due to the characteristics of medical data influenced by acquisition protocols, domain variations, label limitations, and noise or imperfect label issues [23]. In addition, medical imaging (especially histopathology/hematology) often experiences variations in imaging conditions such as differences in staining and shift domains between centers/tools, which makes model generalization more difficult [24]. The problem of class imbalance is also common in the classification of medical images and has been proven to need special treatment so that the performance of the minority class does not fall [25]. Acute Lymphoblastic Leukemia (ALL) is a type of blood cancer in which early and accurate identification plays an important role in determining appropriate treatment and achieving better clinical outcomes [26]. Deep learning-based approaches promise automation solutions that can handle such diversity and complexity better than conventional analytics [23]. Therefore, a systematic evaluation of the combination of CNN architecture and optimizer is critical to obtaining optimal performance and reliable models in clinical practice.

Several previous studies have applied CNN to the detection and classification of leukemia from microscopic blood imaging. Jeneessha et al. [27], demonstrating CNN's ability to distinguish very fine differences in blood cell morphology, they reported an average accuracy of 99.05% using ResNet-50 for the classification of 4 leukocyte types in the BCCD dataset, confirming the advantages of deep learning over traditional methods. However, very high accuracy figures on curated datasets also need to be read carefully (potential for data leakage or split that are not patient-based), so studies that emphasize rigorous validation remain important. In the case of ALL, various modern architectures have been tried. Thiriveedhi et al. [28] developed a CNN-based ALL-Net model and explainable AI, which achieved an accuracy of about 97.85% on 3,256 peripheral blood images. Abd El-Aziz et al. [29] using EfficientNetV2-S with a cross-validation scheme, it managed to obtain an accuracy of up to ~97.3% in differentiating ALL cells. Ghaderzadeh et al. [30] built a public ALL dataset containing ~3,562 blood-removal images from 89 patients (including 25 healthy individuals), and trained a variety of pretrained models; DenseNet201 is reported to achieve up to 99.85% accuracy (99.52% sensitivity, 99.89% specificity) on the ALL subtype classification task. In addition, Muduli et al. [31] evaluate a diverse range of pretrained CNN models (VGG16, ResNet50, Xception, EfficientNet-B0/V2B0, etc.) for ALL classification and demonstrate the high potential of modern CNN in this task. Mohan et al. [32] propose an ensemble framework that combines deep learning features (DenseNet-121, ResNet-34) with machine learning classifiers (SVM, K-NN, Random Forest), capable of improving accuracy by up to ~92.5% (F1-score 93.1%) on larger ALL C-NMC datasets. Although various architectures and approaches have been successfully applied, to date there have been no studies that specifically compare the influence of several optimizers (e.g. Adam, AdamW, NAdam, RMSprop) on the performance of cutting-edge CNN models such as EfficientNet and ResNet in the ALL classification.

Based on the above problems and knowledge gaps, this study aims to evaluate and compare the performance of four optimizers (Adam, AdamW, NAdam, RMSprop) on various CNN architectures (EfficientNet-B2, EfficientNet-B4, EfficientNetV2-S,

EfficientNetV2-M, ResNet50V2, and Xception) in the task of classifying blood cell microscopic images for ALL detection. We used an ALL public dataset [33] of 3,256 colored blood cell images from 89 patients (four classes), and conducted in-depth training experiments for each combination of architecture and optimizer. The results obtained are expected to provide empirical insight into which combination of CNN architecture and optimizer is most effective for the classification of ALL, so that it can serve as a reference for the development of a more accurate and efficient computer-aided diagnostic system in supporting the diagnosis of acute lymphoblastic leukemia.

2. Methods

The experimental procedure consisted of several sequential activities, beginning with the acquisition of the dataset and followed by data preparation, model development, and performance assessment. These activities were organized in a consistent manner to support a systematic and reproducible experimental process, allowing the resulting findings to be evaluated objectively. An overview of the methodological workflow is presented in Figure 1.

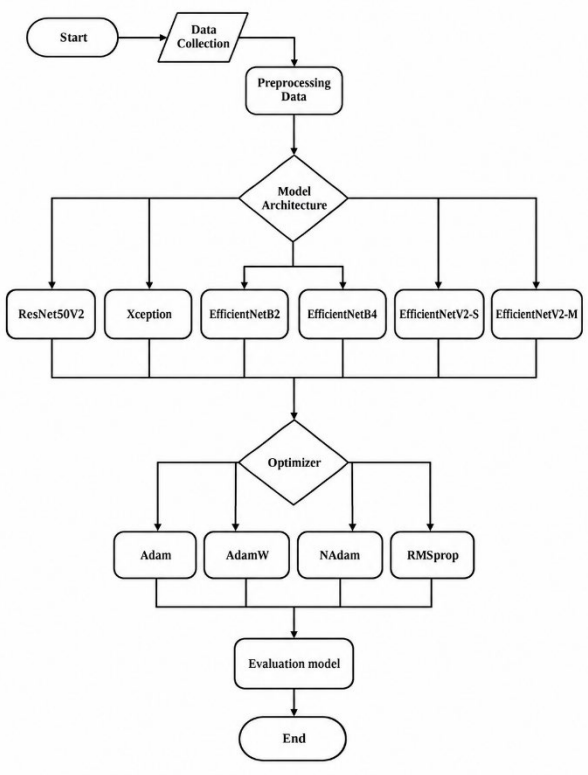


Figure 1. Flowchart

2.1 Dataset

This research employs a publicly available dataset of Peripheral Blood Smear (PBS) images obtained from the Kaggle repository at https://www.kaggle.com/datasets/mohammadamireshraghi/blood-cell-cancer-all-4class. The dataset contains 3,242 images collected for the analysis of blood cell conditions associated with Acute Lymphoblastic Leukemia (ALL). The images are grouped into four categories according to the morphological characteristics and malignancy stage of the cells, comprising 512 Benign images, 955 Malignant Early Pre-B images, 796 Malignant Pre-B images, and 979 Malignant Pro-B images. The complete set of images was subsequently utilized in the development and evaluation of classification models to differentiate normal and leukemic blood cells according to their morphological features.

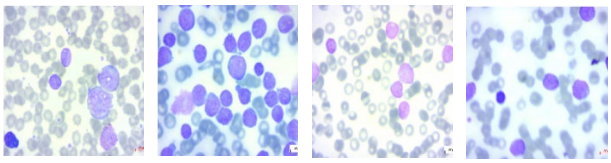


Figure 2. Dataset Kelas ALL (a) Benign, (b)Malignant Pre-B ALL, (c) Malignant Pro-B ALL, (d) Malignant Early Pre-B ALL

2.2 Preprocessing dan Split Dataset

Preprocessing is the initial stage in medical image processing that aims to improve data quality, reduce irrelevant variations, and prepare images in an appropriate format before the deep learning model training process [34]. In this study, preprocessing is focused on adjusting the image size to standardize the input dimensions according to the needs of each CNN architecture, namely EfficientNetB2, EfficientNetB4, EfficientNetV2-M, EfficientNetV2-S, ResNet50V2, and Xception. The image resizing process is carried out to ensure the compatibility of the model inputs, reduce the computational burden, and maintain important morphological information on blood cell images, which is crucial in the task of classifying medical images. To ensure accurate model evaluation and avoid overfitting, the medical imagery dataset is divided into three parts: training: validation: test, 80:10:10. This strategy allows monitoring of model performance during training while evaluating generalization capabilities against unknown data, as applied in a chest image classification study with CNN [35].

2.3 Augmentation

Data augmentation was employed to generate greater variation among the training samples while preserving their original class labels. This process involved modifying the input images through a range of transformations without altering the underlying class of each sample [35]. The use of augmented samples helps the model learn from more varied image patterns, thereby limiting its tendency to memorize the training data and supporting better performance on unseen samples. In this experiment, image diversity was generated through several geometric operations, namely rotation, width shifting, height shifting, shearing, zooming, horizontal flipping, and the use of a fill mode to handle newly exposed image regions.

2.4 Model CNN

Convolutional Neural Network (CNN) is one of the artificial neural network architectures that is specifically designed to process two-dimensional image data [36]. Conventional CNN is generally built using a sequential approach consisting of several main layers, namely convolution layers to extract local spatial features, pooling layers to reduce computational dimensions and complexity, and fully connected layers that function as classifiers based on the features that have been studied. Conventional CNN architecture is often used as a baseline model in medical image classification research due to its simple structure but effective in studying hierarchical representations of visual features. Examples of some CNN architecture models that will be used in this study are:

EfficientNetB2 belongs to the EfficientNet model series and employs a compound scaling strategy for adjusting the overall model configuration. Rather than increasing a single architectural dimension, this strategy simultaneously considers the number of network layers, the number of channels, and the resolution of the input images to achieve a more balanced model design [37]. The compound scaling approach allows for a proportional increase in model capacity across all three dimensions, resulting in better classification performance with relatively efficient computing requirements. This strategy enables EfficientNetB2 to achieve high accuracy with a lower number of parameters than conventional CNN architectures and unstructured magnified models.

EfficientNetB4 belongs to the EfficientNet model family and incorporates a coordinated scaling mechanism in its architectural design. The model adjusts several key dimensions simultaneously, including the number of layers, the channel width, and the resolution of the input images, rather than relying on changes to a single network dimension [37]. Compared to EfficientNetB2, EfficientNetB4 has a larger model capacity through proportional enhancement of all three dimensions, thus being able to extract more complex visual features. Despite having a higher number of parameters and computing requirements, EfficientNetB4 still maintains good computing efficiency and shows significant improvements in accuracy, especially on high-complexity image classification tasks, including medical imaging.

EfficientNetV2-S represents a compact configuration within the EfficientNetV2 series, developed with consideration for both model performance and computational requirements [38]. Its architecture employs Fused Mobile Inverted Bottleneck Convolution (Fused-MBConv) in the earlier stages and conventional MBConv blocks in subsequent deeper stages. This architectural arrangement is intended to improve the efficiency of the training process while supporting stable optimization. Compared with EfficientNetV2-M, the S variant contains fewer trainable parameters and requires lower computational resources. These characteristics make EfficientNetV2-S a practical option for image classification experiments involving moderate-sized datasets or hardware with constrained computational capacity, while still retaining the ability to learn discriminative visual features.

EfficientNetV2-M is one variant of the EfficientNetV2 family that was developed to improve training speed and convergence stability without sacrificing model accuracy [38]. The EfficientNetV2 architecture combines the use of fused mobile inverted bottleneck convolution (Fused-MBConv) at the initial layer and MBConv at the deeper layers, reducing training latency while maintaining robust feature extraction capabilities. As a medium-scale variant, EfficientNetV2-M has a larger model capacity than EfficientNetV2-S, making it more effective in handling large and complex datasets, including medical image classification tasks that require rich and detailed feature representation.

ResNet50V2 is a variant of the Convolutional Neural Network (CNN) architecture that is an evolution of ResNet50 with improvements to residual learning, specifically through the use of pre-activation residual blocks that move activation functions and batch normalization prior to convolutional operations [39]. This architectural design facilitates the propagation of information through deeper layers and contributes to more stable model optimization. As a result, the network can capture visual patterns at different levels of abstraction and maintain reliable predictive capability across various image classification scenarios. ResNet50V2 has also been adopted in numerous computer vision applications, particularly medical image analysis, where its hierarchical feature extraction capability can support the identification of disease-related patterns in radiographic images.

Xception adopts a convolutional design in which feature extraction is performed through two sequential operations. The first operation applies depthwise convolution independently to individual channels, followed by pointwise convolution using a 1×1 kernel to integrate information across channels [40]. This approach significantly reduces the number of parameters and computational load while maintaining or even improving representational capabilities over standard convolution; Xception also utilizes residual structures to smooth gradient flow in deep tissues. Due to its efficiency and performance, Xception is often used as the basis or starting point for modifications to medical image classification applications.

2.5 Optimizer

An optimizer is an algorithm used in neural network training to update the weights of the model based on the loss function gradient calculated through backpropagation. The main goal of the optimizer is to

find a weight value that minimizes the loss value, so that the model's ability to map features to the target label is optimal. In this study, several optimizers were used for model training, namely Adam, AdamW, NAdam, and RMSprop optimizers.

Adam is an adaptive and efficient optimization algorithm, often used as an alternative to conventional stochastic gradient descent [41]. This algorithm combines RMSProp and momentum, with momentum from the first moment of the biased gradient and the second moment calculated using an exponential moving average, with the following formulation [42].

The first moment is calculated using Equation 1.

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

m_t represents the first-moment estimate, β_1 is the exponential decay rate for the first moment, and g_t denotes the gradient at iteration (t). The second moment is calculated using Equation 2.

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

v_t represents the second-moment estimate and β_2 is the exponential decay rate for the second moment.

Because $m_0 = 0$ and $v_0 = 0$, the estimates of m_t and v_t are biased toward zero during the initial iterations. Therefore, bias correction is applied to the first and second moments using Equations 3 and 4, respectively:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (3)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (4)$$

The model parameters are subsequently updated according to Equation 5.

$$\hat{\theta}_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}}+\epsilon} \quad (5)$$

θ_t is the parameter of the model in the iteration, is the learning rate, $\hat{m}_t$ provides a stable direction (momentum), $\sqrt{\hat{v}_t}$ determines the magnitude of the step, and ϵ is a small constant.

AdamW incorporates momentum information into the parameter update process, allowing previous gradient directions to contribute to subsequent updates. This mechanism can help the optimization process move more efficiently toward a suitable solution during model training [43].

The first moment is calculated using Equation 6.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (6)$$

The second moment is calculated using Equation 7.

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

Bias correction is then applied to the first and second moments using Equations 8 and 9, respectively:

$$m_t \leftarrow \frac{m_t}{\alpha_1 \mu_t} \quad (8)$$

$$v_t \leftarrow \frac{v_t - m_t^2}{\alpha_2 \sigma_t} \quad (9)$$

The AdamW parameter update uses a decoupled weight decay formulation, as shown in Equation 10.

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t}+\varepsilon} - \eta\lambda\theta_t \quad (10)$$

θ_t is the parameter of the model, η is the learning rate, m_t is the average momentum of the gradient, v_t is the mean of the square of the gradient, ε is the small value, and λ is the weight decay.

NAdam is an adaptive optimization algorithm that combines Nesterov momentum with the adaptive learning rate mechanism of Adam [44]. This combination enables NAdam to incorporate the predictive behavior of Nesterov momentum into the parameter update process. The first moment is calculated using Equation 11.

$$m_t = \mu_t m_{t-1} + (1-\mu_t) g_t \quad (11)$$

m_t represents the first-moment estimate, μ_t is the momentum decay coefficient, and g_t denotes the gradient at iteration (t). The second moment is calculated using Equation (12):

$$v_t = v n_{t-1} + (1-v) g_t^2 \quad (12)$$

v_t represents the second-moment estimate and v is the exponential decay coefficient for the second moment.

Bias correction and Nesterov momentum are incorporated into the parameter update, as shown in Equation 13.

$$\theta_t = \theta_{t-1} - \alpha_t \frac{\hat{n}_t}{\sqrt{\hat{n}_t}+\varepsilon} \hat{m} \quad (13)$$

g_t is the gradient, m_t is the momentum estimate, n_t is the second moment estimate, $\hat{n}_t$ is the bias correction, α_t is the learning rate, and ε is the constanta.

RMSProp is an adaptive optimization algorithm designed to solve the problem of vanishing or fluctuating learning rate in SGD [45]. In RMSProp, the learning rate value of each parameter is adjusted based on the moving average of the gradient square. RMSProp adjusts the magnitude of parameter updates according to the recent behavior of the gradients. By incorporating information from previous gradient values, the optimizer can regulate the step size assigned to individual parameters during training. The method is generally described through two successive operations: maintaining an estimate based on the squared gradients and using this estimate to determine the subsequent parameter updates.

The moving average of the squared gradients is calculated using Equation 14.

$$v_t = \beta v_{t-1} + (1-\beta) g_t^2 \quad (14)$$

v_t is the mean value of the square of the gradient, β is the coefficient, v_{t-1} is the mean value of the square of the gradient in the previous step, and g_t^2 is the gradient square.

The model weights are subsequently updated using Equation 15.

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t}+\varepsilon} g_t \quad (15)$$

w_t is the weight of the model, w_{t+1} is the weight that has been updated, η is the learning rate, v_t is the average value of the square of the gradient, ε is the small value to prevent division by zero, g_t is the gradient, and $\frac{\eta}{\sqrt{v_t}+\varepsilon}$ is the adjustment factor to be stable when the gradient is large or small.

2.6 Evaluation

Model evaluation is conducted to determine the reliability of a classification system in assigning the correct class to unseen samples. The assessment is performed by examining the agreement between the predicted classes and their corresponding ground-truth labels, which provides a basis for judging the model's predictive capability and identifying areas that may require further improvement [46]. When the classes in a dataset are not evenly distributed, relying on a single performance indicator may not provide a comprehensive view of the classifier's behavior. Therefore, the confusion matrix was employed as the basis for evaluating the classification results. It records the correspondence between the predicted and actual classes through four possible outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [47]. These outcomes are subsequently used to derive accuracy, precision, recall, and F1-score.

Accuracy, as defined in Equation (16), represents the fraction of all test samples that are assigned to their correct classes. Precision, whose formulation is given in Equation (17), focuses on the correctness of positive predictions by comparing the number of correctly identified positive cases with all cases classified as positive. Recall, presented in Equation (18), evaluates how effectively the classifier detects positive instances by considering the proportion of correctly recognized positives among the actual positive cases. Finally, Equation (19) defines the F1-score, which provides a single performance value by combining precision and recall through their harmonic mean.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (16)$$

$$Precision_i = \frac{TP_i}{TP_i+FP_i} \quad (17)$$

$$Recall_i = \frac{TP_i}{TP_i+FN_i} \quad (18)$$

$$F1_i = 2\frac{Precision_i \cdot Recall_i}{Precision_i+Recall_i} \quad (19)$$

3. Results and Discussions

This section presents the results of the experiment along with a critical discussion regarding the influence of optimizer selection on the performance of various CNN architectures on image classification tasks. To maintain the validity of the interpretation, the discussion begins with a description of Technical Implementation and Reproducibility, so that the experimental procedures, training parameters, and computing environment can be replicated. After that, the Preprocessing Results were presented to show how the data sharing and augmentation strategies shaped the characteristics of the training data as well as the potential bias due to class imbalance. The discussion then focused on CNN Performance Evaluation, by comparing the overall performance of the model through accuracy, convergence dynamics on the training curve, and error patterns per class through confusion matrix and classification report.

3.1 Technical Implementation and Reproducibility

All experiments were implemented on Google Colab using NVIDIA T4 GPU acceleration to ensure that the model training process was efficient and consistent between configurations. Implementation was carried out with PyTorch as the main framework of deep learning, while the CNN architecture was invoked through timms (PyTorch Image Models) to load models such as Xception, ResNet50V2, EfficientNetB2, EfficientNetB4, EfficientNetV2-RW-S, and EfficientNetV2-RW-M. The quantitative evaluation process utilizes scikit-learn for the preparation of confusion matrix and classification reports, as well as visualization of training and evaluation results using Matplotlib/Seaborn. The dataset is obtained via the Kaggle API/CLI and organized in a uniform directory structure so that the notebook execution flow can be rerun without major modifications to the data path.

The experimental design was designed to make comparisons between optimizers fair, i.e. by maintaining the same training procedures on each architecture, and then replacing only the optimizers tested (Adam, AdamW, NAdam, RMSprop). The training used the CrossEntropyLoss loss function for multi-class classification and applied a two-stage learning transfer scheme: the initial stage of training the classifier/head with a learning rate of 0.001 for 10 epochs, then continued with full fine-tuning on all model parameters with a learning rate of 0.00001 for 10 epochs. To maintain convergence stability, adaptive learning rate adjustment is implemented using ReduceLROnPlateau which monitors validation loss so that the learning rate is lowered when validation performance improvements slow down.

The reproducibility aspect is maintained through locking the data composition using train: validation:test = 80%: 10%: 10% separation with a deterministic generator (seed 42) in the split process, as well as training configurations that are maintained identical across optimizer architecture combinations. The best model per configuration is stored in a validation performance-based state_dict (.pth) format so that results can be re-verified without retraining.

The final evaluation was carried out on the test set using accuracy and error analysis per class through confusion matrix and classification report, while learning dynamics were documented through training history (loss and accuracy train or val per epoch) to support convergence interpretation, generalization, and comparison between optimizers more comprehensively.

3.2 Preprocessing Result

The preprocessing stage is carried out to ensure that the image data has a consistent input format across the CNN architecture and to build a more robust training dataset through augmentation. At this stage, the dataset is first mapped into four classes (Benign, [Malignant] Pre-B, [Malignant] Pro-B, and [Malignant] early Pre-B), then divided into a subset of training/validation/testing with a ratio of 80%: 10%: 10%. The data distribution table in Table 1 shows the composition of each class in each subset. There is an imbalance in the number of samples between classes, especially the Benign class which has the least number of samples. This condition needs to be considered at the evaluation stage because it can affect the interpretation of aggregate metrics such as accuracy.

Table 1. Results of Split Datasets

Class	Training	Val	Test	Total
Benign	414	60	38	512
Pre-B ALL	762	89	104	955
Pro-B ALL	635	88	73	796
Early Pre-B ALL	782	87	110	979
Total	2593	324	325	3242

To maintain input compatibility across different architectures, the image is then resized to the standard input size of each model Table 2.

Table 2. Input size of each CNN architecture

Architecture	Input size (H×W)
ResNet50V2	224×224
Xception	299×299
EfficientNetB2	260×260
EfficientNetB4	320×320
EfficientNetV2- S	320×320
EfficientNetV2-M	320×320

This approach ensures that comparisons between architecture and optimizer are performed under input conditions consistent with common architecture design and transfer learning practices (e.g., ResNet based 224×224, Xception 299×299, and the EfficientNet B/V2 family, which generally uses higher resolutions). With consistent input size settings per architecture, observed performance differences can more easily be attributed to architectural and optimizer characteristics than input resolution mismatches.

In addition, data enrichment procedures were exclusively implemented on the learning subset through the torchvision. transforms framework to increase data diversity and mitigate overfitting. According to the implemented pipeline, each image was initially upscaled slightly to accommodate subsequent transformations, followed by stochastic operations including horizontal flipping with a probability of 0.5, vertical flipping with a probability of 0.2, random rotations within ±15 degrees, and affine transformations involving translations of up to 10% and scaling factors ranging between 0.9 and 1.1, without introducing extra rotation. Variations in illumination and color properties were further emulated using color perturbation techniques that adjust brightness, contrast, saturation, and hue within controlled ranges.

Subsequently, the processed images were restored to the required network input resolution, transformed into tensor representations, and standardized based on dataset-specific statistical parameters. Figure 3 illustrates representative outputs of this procedure, displaying one reference image alongside eight distinct, randomly generated transformed versions. The presence of dark regions in some samples originates from geometric operations that expose peripheral areas beyond the original field of view. Such transformations encourage the network to acquire feature representations that remain robust under changes in orientation, spatial positioning, and visual appearance, thereby preventing over-reliance on fixed visual patterns present in the training samples.

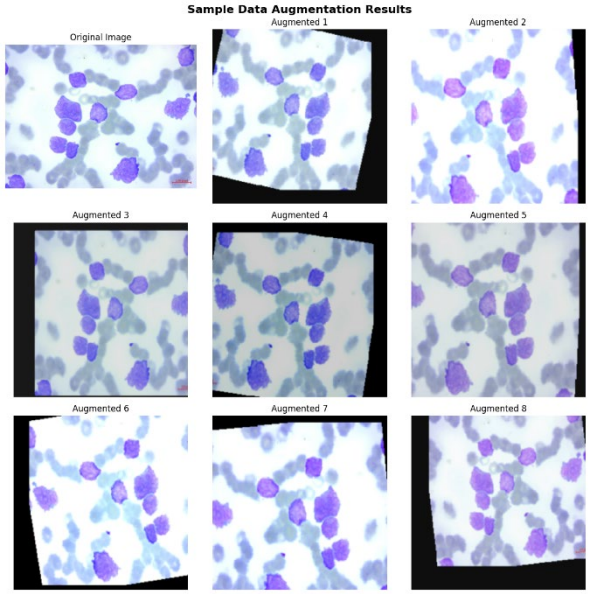


Figure 3. Data Augmentation Examples

3.3 CNN Model Performance

This section presents a comparative evaluation of the performance of six CNN architectures (Xception, ResNet50V2, EfficientNetB2, EfficientNetB4, EfficientNetV2-S, EfficientNetV2-M) using four optimizers (Adam, AdamW, NAdam, RMSprop) on the same test data. Performance is summarized through accuracy and cross-entropy loss in the test set (Table 3) as a concise indicator of prediction accuracy and probability quality of model output. To ensure that interpretation does not stop at aggregate numbers, the analysis is then complemented by a confusion matrix

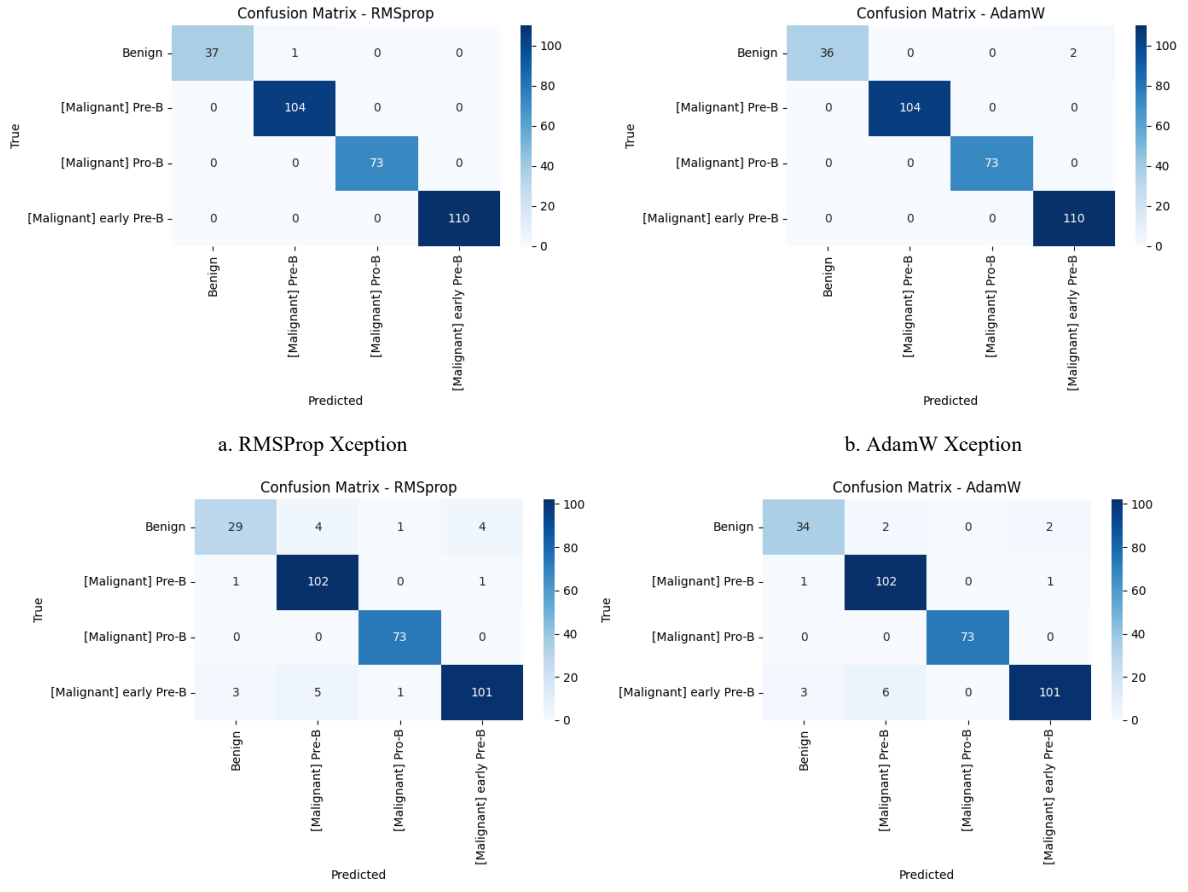
to review patterns of misclassification between classes, as well as classification reports per class that display precision, recall, and F1-score so that the strengths and weaknesses of the model can be understood more granularly, especially under unbalanced class distribution conditions.

Table 3. Comparison of test accuracy and test loss

Architecture	Adam		AdamW		NAdam		RMSProp	
	Acc	Loss	Acc	Loss	Acc	Loss	Acc	Loss
ResNet50V2	0.9600	0.1593	0.9538	0.1581	0.9569	0.1863	0.9384	0.2023
Xception	0.9877	0.0294	0.9939	0.0215	0.9908	0.0399	0.9969	0.0194
EfficientNetB2	0.9785	0.0643	0.9754	0.0878	0.9815	0.0634	0.9661	0.0702
EfficientNetB4	0.9877	0.0354	0.9908	0.0458	0.9908	0.0429	0.9877	0.0632
EfficientNetV2- S	0.9846	0.0387	0.9754	0.0880	0.9908	0.0483	0.9723	0.0664
EfficientNetV2-M	0.9908	0.0339	0.9877	0.0324	0.9877	0.0348	0.9877	0.0373

In general, the combination of Xception + RMSprop provides the best performance with an accuracy of 99.69% and a low loss of 0.0194, indicating highly accurate predictions and a more "certain" probability of output (low loss). On the other hand, ResNet50V2 tends to produce worse accuracy and losses than other architectures across all optimizers, with the most significant decline in RMSprop (93.85%; loss 0.2023). These findings show that differences in architecture and optimizers not only change the accuracy figures, but also affect the stability of optimization and the quality of prediction calibration (reflected in loss).

If you look at the optimizer section, NAdam shows very competitive performance on EfficientNetV2 architectures (e.g. EfficientNetV2-M and EfficientNetV2-S), while AdamW appears to be stable on some large architectures (e.g. Xception and EfficientNetB4). RMSprop is more "context-sensitive", it can be best on Xception, but it drops performance quite a bit on ResNet50V2 and EfficientNetV2-M. This pattern is important because it prevents overly general conclusions such as "a particular optimizer is always best", in fact, the effect of a strong optimizer is influenced by the architectural characteristics and dynamics of the gradient during fine-tuning.



a. RMSProp Xception

b. AdamW Xception

c. RMSProp ResNet50V2

d. AdamW ResNet50V2

Figure 4. Confusion Matrix

To clarify the model's success and failure, Figure 4 shows the confusion matrices of the two best-performing and two worst-performing architecture–optimizer combinations. This selection highlights the contrast between the highest and lowest performance, while the other combinations fall between these two extremes. A good confusion matrix is indicated by dominant values on the main diagonal, meaning most samples are correctly classified. In the best-performing model, such as Xception + RMSprop, off-diagonal values are very small, showing only a few misclassifications. In contrast, the worst-performing model, such as ResNet50V2 + RMSprop, shows more prominent off-diagonal values, indicating higher inter-class errors. This analysis is important because overall accuracy can hide weaknesses in minority classes, such as Benign, where a small number of errors may significantly reduce recall. The error patterns observed in the confusion matrices can be further explained through the interaction between the optimizer and network architecture, particularly in the case of RMSprop. The pronounced performance disparity of RMSprop between ResNet50V2 and Xception can be hypothesized to stem from the interaction between RMSprop's adaptive learning rate mechanism and the residual connection characteristics of the ResNet architecture. RMSprop adjusts the learning rate of each parameter based on the exponentially decaying moving average of squared gradients, causing parameters with high gradient variance to receive smaller update steps.

In the ResNet50V2 architecture, which relies heavily on identity-based residual connections to preserve gradient flow in deep networks, this adaptive mechanism may inadvertently attenuate the contribution of gradients along the residual branches, especially during the fine-tuning stage with a relatively small learning rate. As a result, weight adaptation in deeper layers becomes less optimal, preventing feature representations from fully adapting to the characteristics of the new data, which in turn leads to higher loss values and reduced accuracy. In contrast, Xception does not rely on deep identity residual connections, but instead employs depthwise separable convolutions that explicitly decouple spatial and channel-wise modeling. This design tends to produce more stable and localized gradient distributions across blocks, making it more compatible with RMSprop's parameter update strategy. Under these conditions, adaptive learning rate adjustment facilitates faster convergence and improved generalization performance, as reflected by higher accuracy and lower loss values. Therefore, the observed performance degradation of ResNet50V2 when using RMSprop is not merely empirical, but is likely influenced by a mismatch between RMSprop's optimization dynamics and the residual-dominated structure of the ResNet architecture, particularly in transfer learning–based fine-tuning scenarios.

4. Conclusions

Analysis of the experimental outcomes demonstrates that the pairing between network architecture and optimization strategy plays a decisive role in determining classification effectiveness for ALL images. Among the evaluated configurations, the Xception network optimized with RMSprop achieved the most favorable outcomes, as reflected by superior predictive accuracy and minimal error values. Conversely, ResNet50V2 exhibited comparatively weaker behavior across multiple optimization settings, with the most pronounced performance decline observed when RMSprop was employed. These discrepancies across configurations indicate that a universally optimal optimizer does not exist; rather, optimization success is closely linked to the structural properties of the network and its learning dynamics. The objectives of this work are therefore fulfilled by clearly illustrating performance distinctions across architecture optimizer pairings and emphasizing the necessity of carefully selecting model configurations to achieve reliable classification results. The optimal configuration identified in this work, namely Xception combined with RMSprop, carries meaningful implications for the design of automated medical image analysis systems. Deploying this configuration may enhance the robustness and predictive reliability of computer-assisted frameworks for ALL detection, thereby strengthening their ability to support clinical decision-making processes. More broadly, these findings offer practical insights for both researchers and practitioners when choosing suitable CNN structures alongside effective optimization strategies for related image classification problems. Recognizing that predictive capability is sensitive to training design allows system developers to better align model choices with the intrinsic characteristics of medical image data, ultimately enabling the construction of efficient and dependable diagnostic support systems for clinical use.

Despite these contributions, several constraints should be acknowledged. The experimental validation relied on a single publicly available dataset with restricted sample size and variability, which may limit the direct transferability of the results to real-world clinical settings. Furthermore, the investigation was confined to six network designs and four optimization algorithms, leaving open the possibility that alternative architectures or learning strategies could yield improved outcomes under different conditions. Future studies are therefore encouraged to examine the identified optimal configuration in practical deployment scenarios, such as integration into automated diagnostic pipelines and assessment using real patient data. Additional investigations may also consider alternative CNN families, expanded hyperparameter exploration, and diverse medical imaging modalities to reinforce and extend the present conclusions. Such efforts will contribute to the advancement of deep learning driven diagnostic

systems that are more accurate, resilient, and trustworthy in clinical environments.

Acknowledgements

The authors gratefully acknowledge the support provided by FAST Ahmad Dahlan University, Indonesia, particularly the Mathematics Laboratory. The facilities, resources, and assistance made available by the institution played an important role in supporting the implementation and completion of this research.

References

[1] N. Khadem, A. Nashir, and S. Rahmatyar, "The Role of Deep Learning in Advancing Computer Vision Applications: A Comprehensive Systematic Review," *Journal of Advanced Computer Knowledge and Algorithms*, vol. 3, no. 1, pp. 1–8, 2025, doi: https://doi.org/10.29103/jacka.v3i1.24732.

[2] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021, doi: https://doi.org/10.1016/j.mlwa.2021.100134.

[3] I. D. Mienye and T. G. Swart, "A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications," *Information*, vol. 15, no. 12, 2024, doi: 10.3390/info15120755.

[4] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif Intell Rev*, vol. 57, no. 4, p. 99, 2024, doi: 10.1007/s10462-024-10721-6.

[5] M. Trigka and E. Dritsas, "A Comprehensive Survey of Deep Learning Approaches in Image Processing," *Sensors*, vol. 25, no. 2, 2025, doi: 10.3390/s25020531.

[6] M. Krichen, "Convolutional Neural Networks: A Survey," *Computers*, vol. 12, no. 8, 2023, doi: 10.3390/computers12080151.

[7] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of Image Classification Algorithms Based on Convolutional Neural Networks," *Remote Sens (Basel)*, vol. 13, no. 22, 2021, doi: 10.3390/rs13224712.doi: 10.3390/rs13224712.

[8] J. N. Haus *et al.*, "MEMS Vibrometer for Structural Health Monitoring Using Guided Ultrasonic Waves," *Sensors*, vol. 22, no. 14, 2022, doi: 10.3390/s22145368.doi: 10.3390/s22145368.

[9] D. Park and J.-H. Lee, "Feasibility Evaluation of Computational Fluid Dynamics Approach for Inhalation Exposure Assessment: Case Study for Biocide Spray," *Applied Sciences*, vol. 11, no. 2, 2021, doi: 10.3390/app11020634.

[10] M. Tan and Q. Le, "EfficientNetV2: Smaller Models and Faster Training," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., in Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 10096–10106.

[11] L. Fan *et al.*, "Modulation Characteristics of High-Speed Transistor Lasers," *Applied Sciences*, vol. 12, no. 9, 2022, doi: 10.3390/app12094475.

[12] P.-M. Boarescu *et al.*, "Multi-Organ Protective Effects of Curcumin Nanoparticles on Drug-Induced Acute Myocardial Infarction in Rats with Type 1 Diabetes Mellitus," *Applied Sciences*, vol. 11, no. 12, 2021, doi: 10.3390/app11125497.

[13] K. Khotimah, S. Surono, and A. Thobirin, "Optimizing EfficientNet for imbalanced medical image classification using grey wolf optimization," *Computer Science and Information Technologies*, vol. 6, no. 2, pp. 112–121, 2025, doi: 10.11591/csit.v6i2.pp112-121.

[14] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020.

[15] M. Lavassani, J. Åkerberg, and M. Björkman, "Modeling and Profiling of Aggregated Industrial Network Traffic," *Applied Sciences*, vol. 12, no. 2, 2022, doi: 10.3390/app12020667.

[16] Z. Yang and X. Liu, "Roundoff Error Analysis of an Algorithm Based on Householder Bidiagonalization for Total Least Squares Problems," *Mathematics*, vol. 9, no. 20, 2021, doi: 10.3390/math9202550.

[17] A. Sabban, "Wearable Circular Polarized Antennas for Health Care, 5G, Energy Harvesting, and IoT Systems," *Electronics (Basel)*, vol. 11, no. 3, 2022, doi: 10.3390/electronics11030427.

[18] N. Campos Macias, W. Düggelin, Y. Ruf, and T. Hanne, "Building a Technology Recommender System Using Web Crawling and Natural Language Processing Technology," *Algorithms*, vol. 15, no. 8, 2022, doi: 10.3390/a15080272.

[19] W. Hu, "Stability of Impulsive Stochastic Delay Systems with Markovian Switched Delay Effects," *Mathematics*, vol. 10, no. 7, 2022, doi: 10.3390/math10071110.

[20] E. Rytel, A. Tajner-Czopek, A. Kita, A. Tkaczyńska, A. Z. Kucharska, and A. Sokół-Łętowska, "The Influence of the Production Process on the Anthocyanin Content and Composition in Dried Potato Cubes, Chips, and French Fries Made from Red-Fleshed Potatoes," *Applied Sciences*, vol. 11, no. 3, 2021, doi: 10.3390/app11031104.

[21] S. Azimi, C. De Sio, D. Rizzieri, and L. Sterpone, "Analysis of Single Event Effects on Embedded Processor," *Electronics (Basel)*, vol. 10, no. 24, 2021, doi: 10.3390/electronics10243160.

[22] Y. Zheng *et al.*, "Infrared Clinical Enamel Crack Detector Based on Silicon CCD and Its Application: A High-Quality and Low-Cost Option," *J Imaging*, vol. 7, no. 12, 2021, doi: 10.3390/jimaging7120259.

[23] S. K. Zhou *et al.*, "A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises.," *Proc IEEE Inst Electr Electron Eng*, vol. 109, no. 5, pp. 820–838, May 2021, doi: 10.1109/JPROC.2021.3054390.

[24] M. Asadi-Aghbolaghi *et al.*, "Learning generalizable AI models for multi-center histopathology image classification," *NPJ Precis Oncol*, vol. 8, no. 1, p. 151, 2024, doi: 10.1038/s41698-024-00652-4.

[25] I. Park, W. H. Kim, and J. Ryu, "Style-KD: Class-imbalanced medical image classification via style knowledge distillation," *Biomed Signal Process Control*, vol. 91, p. 105928, 2024, doi: https://doi.org/10.1016/j.bspc.2023.105928.

[26] H. Inaba and C.-H. Pui, "Advances in the Diagnosis and Treatment of Pediatric Acute Lymphoblastic Leukemia," *J Clin Med*, vol. 10, no. 9, 2021, doi: 10.3390/jcm10091926.

[27] P. Jeneessha and V. K. Balasubramanian, "Domain knowledge-infused pre-trained deep learning models for efficient white blood cell classification," *Sci Rep*, vol. 15, no. 1, p. 15608, 2025, doi: 10.1038/s41598-025-00563-9.

[28] A. Thiriveedhi, S. Ghanta, S. Biswas, and A. K. Pradhan, "ALL-Net: integrating CNN and explainable-AI for enhanced diagnosis and interpretation of acute lymphoblastic leukemia.," *PeerJ Comput Sci*, vol. 11, p. e2600, 2025, doi: 10.7717/peerj-cs.2600.

[29] A. A. Abd El-Aziz, M. A. Mahmood, and S. Abd El-Ghany, "A Robust EfficientNetV2-S Classifier for Predicting Acute Lymphoblastic Leukemia Based on Cross Validation," *Symmetry (Basel)*, vol. 17, no. 1, 2025, doi: 10.3390/sym17010024.

[30] M. Ghaderzadeh, M. Aria, A. Hosseini, F. Asadi, D. Bashash, and H. Abolghasemi, "A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images," *International Journal of Intelligent Systems*, vol. 37, no. 8, pp. 5113–5133, Aug. 2022, doi: https://doi.org/10.1002/int.22753.

[31] D. Muduli *et al.*, "Deep learning-based detection and classification of acute lymphoblastic leukemia with explainable AI techniques," *Array*, vol. 26, p. 100397, 2025, doi: https://doi.org/10.1016/j.array.2025.100397.

[32] S. B. Mohan, S. Sathya, S. Rajalaksmi, G. Gurumoorthy, and R. Sivanraju, "Efficient convolutional neural networks for acute lymphoblastic leukaemia prediction in computer vision," *Sci Rep*, 2025, doi: 10.1038/s41598-025-30777-w.

[33] A. Hosseini *et al.*, "A mobile application based on efficient lightweight CNN model for classification of B-ALL cancer from non-cancerous cells: A design and implementation study," *Inform Med Unlocked*, vol. 39, p. 101244, 2023, doi: https://doi.org/10.1016/j.imu.2023.101244.

[34] N. Guetta, A. Shabtai, I. Singh, S. Momiyama, and Y. Elovici, "Dodging Attack Using Carefully Crafted Natural Makeup," Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.06467

[35] H. Naveed, S. Anwar, M. Hayat, K. Javed, and A. Mian, "Survey: Image Mixing and Deleting for Data Augmentation," Feb. 2023, [Online]. Available: http://arxiv.org/abs/2106.07085

[36] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artif Intell Rev*, vol. 57, no. 4, Apr. 2024, doi: 10.1007/s10462-024-10721-6.

[37] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," Sep. 2020, [Online]. Available: http://arxiv.org/abs/1905.11946

[38] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," Jun. 2021, [Online]. Available: http://arxiv.org/abs/2104.00298

[39] H. Jiyan Mustaqim and O. Suria, "Comparative Analysis of ResNet50V2, ResNet152V2, and MobileNetV2 Architectures in Monkeypox Classification," 2025. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC

[40] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions."

[41] M. Reyad and A. M. Sarhan, "A modified Adam algorithm for deep neural network optimization," *Neural Comput Appl*, vol. 35, no. 23, pp. 17095–17112, 2023, doi: 10.1007/s00521-023-08568-z.

[42] 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," *Journal of Vibroengineering*, vol. 24, no. 4, pp. 666–678, Jun. 2022, doi: 10.21595/jve.2022.22271.

[43] P. Zhou, X. Xie, Z. Lin, K.-C. Toh, and S. Yan, "Win: Weight-Decay-Integrated Nesterov Acceleration for Faster Network Training," 2024. [Online]. Available: https://github.com/sail-sg/win.

[44] J. Tu, A. Yasin, and N. S. Mansor, "NadamClip: A Novel Optimization Algorithm for Improving Prediction Accuracy and Training Stability," *Processes*, vol. 13, no. 7, Jul. 2025, doi: 10.3390/pr13072145.

[45] R. Abdulkadirov, P. Lyakhov, and N. Nagornov, "Survey of Optimization Algorithms in Modern Neural Networks," Jun. 01, 2023, *MDPI*. doi: 10.3390/math11112466.

[46] S. Farhadpour, T. A. Warner, and A. E. Maxwell, "Selecting and Interpreting Multiclass Loss and Accuracy Assessment Metrics for Classifications with Class Imbalance: Guidance and Best Practices," *Remote Sens (Basel)*, vol. 16, no. 3, Feb. 2024, doi: 10.3390/rs16030533.

[47] M. M. Alam, F. S. Pribadi, R. A. Aprilianto, and A. R. Nurul'aini, "Stacking Ensemble Learning Model for Intrusion Detection in Electrical Substation," *Jurnal RESTI*, vol. 9, no. 5, pp. 1030–1037, Oct. 2025, doi: 10.29207/resti.v9i5.6502.