Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 1 (2025) 118 - 129 e-ISSN: 2580-0760

Application of VGG16 in Automated Detection of Bone Fractures in X-Ray Images

Resky Adhyaksa[1], Bedy Purnama[2*]
[1]School of Computing, Telkom University, Bandung, Indonesia
[2]Center of Excellence Artificial Intelligence for Learning and Optimization, Telkom University, Bandung, Indonesia
[1]reskyadhyaksa@student.telkomuniversity.ac.id, [2]bedypurnama@telkomuniversity.ac.id

Abstract

The purpose of this research is to determine whether or not a deep learning model called VGG16 can automatically identify bone fractures in X-ray pictures. The dataset, sourced from Kaggle, includes 10,522 images of human hand and foot bones, which underwent preprocessing steps such as normalization and resizing to 224x224 pixels to enhance data quality. The study utilizes the VGG16 architecture, pre-trained on ImageNet, as a base model, with transfer learning applied to adapt the model for fracture detection by fine-tuning its weights. This architecture consists of five blocks of convolutional and max-pooling layers to effectively extract and enhance information from the images for precise classification. The training and testing phases utilized an 80:20 split of the data, employing binary cross-entropy as the loss function and the Adam optimizer for efficient weight updates. The model achieved high performance, with an accuracy of 99.25%, precision of 98.62%, recall of 98.88%, and an F1-score of 99.16% over 25 epochs with a batch size of 128. Experimental results indicate that smaller batch sizes generally enhance accuracy and reduce loss values, with batch sizes of 128 and 16 yielding optimal performance. The study's findings underscore the potential of VGG16 in improving diagnostic accuracy and reliability in medical imaging, providing a robust tool for fracture detection. Future research should continue exploring hyperparameter optimization to further enhance model performance while balancing computational efficiency.

Keywords: *Deep Learning; Bone Fracture; X-Ray Images; Convolutional Neural Network; VGG16*

How to Cite: R. Adhyaksa and B. Purnama, "Application of VGG16 in Automated Detection of Bone Fractures in X-Ray Images", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 9, no. 1, pp. 118 - 129, Feb. 2025.
DOI: https://doi.org/10.29207/resti.v9i1.6101

1. Introduction

A fracture is a medical condition that occurs when the continuity of bone is disrupted, usually due to significant trauma or injury [1]. Fractures can vary from a minor crack to a completely separate bone. This condition not only causes severe pain but can also result in functional impairment of the affected body part. Broken bones often require appropriate medical treatment, including immobilization using a cast or splint and, in some cases, surgery to ensure optimal healing.

Bone fractures are a common health problem with a significant impact on the quality of life of sufferers. In addition to intense pain, fractures often lead to decreased mobility and body function, hampering daily activities [2]. The main causes of fractures include direct trauma such as accidents, falls, or hard impacts, as well as certain medical conditions such as osteoporosis that can increase the risk of fracture. Understanding the types of fractures, their symptoms, and effective treatment methods is crucial in recovery. With prompt and appropriate treatment, the prognosis of fracture patients can be improved, minimizing the risk of long-term complications and ensuring a return to normal function of the injured bone.

Fractures are usually detected and analyzed through X-rays, which allows doctors to identify and diagnose fractures. Various studies have been developed for early fracture detection methods using X-rays to improve the accuracy and speed of diagnosis. One such study utilized the Deep Neural Network (DNN) method for fracture classification and achieved 92.44% accuracy in predicting fracture samples on X-ray images [3].

Other research also focuses on the application of deep learning methods, utilizing the MURA dataset from Stanford and the Deep Convolutional Neural Network

Received: 05-10-2024 | Accepted: 23-11-2024 | Published Online: 14-02-2024

(DCNN) technique with the AlexNet model [4]. This research obtained an accuracy of 86.67%, which is lower than previous research. However, it provides a new picture in conducting fracture research.

In addition, there is also research that focuses on the classification of X-ray images to detect pneumonia [5]. This research has similarities with the research conducted, which both use the VGG16 model. In the study on pneumonia, the VGG16 model achieved an accuracy value of 95% in image classification. This success shows the significant potential of the VGG16 model in analyzing and classifying medical images.

VGG16 is a Convolutional Neural Network (CNN) model that plays a key role in deep learning by extracting high-level features from images through its 16-layer architecture. This network has been specifically designed to capture intricate details by applying convolutional filters across various layers, making it highly effective for image classification tasks. In a breast cancer classification study using the VGG16 model as the classification algorithm, an accuracy of 89.6% was obtained [6]. The use of VGG16 in breast cancer classification shows its potential in supporting early diagnosis and improving detection accuracy, which is crucial for timely treatment and care for patients. The model's ability to learn hierarchical features from raw image data demonstrates its core function in CNN-based deep learning.

Another research study [7] proposed a technique for diagnosing bone fractures by leveraging texture features based on income inequality and support vector machines (SVM). The study was focused on accurately and efficiently identifying and categorizing both broken and intact bones. The proposed model significantly improved the detection performance by using the Gray Level Co-occurrence Matrix (GLCM) texture features in conjunction with the Gini Index. The results indicated that the SVM model achieved an accuracy of 95% in classifying fractured bones, demonstrating a notable improvement over previous methods.

Recent research into fracture classification using artificial intelligence has shown significant progress in classification accuracy [8]. This research utilized the BoneView algorithm from the GLEAMER company to detect fractures [9]. The algorithm achieved an accuracy of 97%, the highest value compared to previous studies. This result confirms the superiority of BoneView in improving fracture detection accuracy, making an important contribution to the field of diagnostic radiology.

Various deep learning models have been employed in medical imaging, such as Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs), for tasks like fracture classification and disease detection. Among these, models like AlexNet, ResNet, and EfficientNet have demonstrated high accuracy in image classification. Despite the advancements in these models, VGG16 was chosen for this study due to its simplicity, effectiveness in feature extraction, and consistent performance in medical image classification tasks. While models like ResNet and EfficientNet are known for their superior performance in many classification tasks due to their deeper architectures and optimization techniques, VGG16 offers a more straightforward and interpretable structure, making it easier to fine-tune for specific applications such as fracture detection.

This study aims to contribute to two main aspects based on previous research. First, this study uses a dataset that has gone through a preprocessing stage with a normalization approach, aiming to reduce noise and improve data quality. Secondly, this study applies the VGG16 classification method, which is more accurate than previous studies. With this approach, this study achieved high accuracy and offered a method that can be adopted to improve fracture detection in the future. The results from this study show significant potential in improving the quality of radiology diagnosis and provide a solid foundation for developing more effective fracture detection methods.

2. Research Methods

In the research conducted, there is a flowchart that explains each step of problem-solving based on Figure 1. The first stage is data fetching, collected from Kaggle, an open-source website. A total of 10,522 data were found in this process. The collected data underwent a preprocessing stage to remove and reduce the noise.

The next stage involves creating the VGG16 layer model, which will serve as the main algorithm for solving this problem. After creating the model and processing the data, we divide the data into two parts with a ratio of 80:20 for training and testing purposes. The training stage utilizes the training data, while the prediction stage uses the testing data. Finally, we evaluate the model to assess the performance of the trained and tested models. The research flowchart system will be summarized in Figure 1.

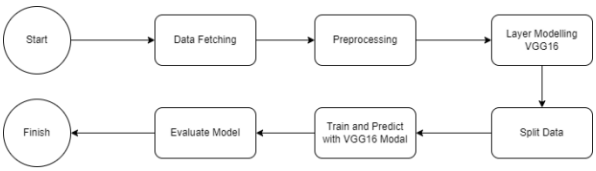

Figure 1. Research Flowchart

2.1 Dataset

The data is sourced from the open-access platform Kaggle [10], which provides a free collection of X-ray data of broken human hand and foot bones. The dataset consists of 10,522 images, split into three main parts for training, testing, and validation. The split follows a common practice in deep learning to reserve a portion of the dataset for testing and validation in order to measure the model's performance on unseen data.

This is an open access article under the CC BY-4.0 license

The reason for using an 80:20 data split in this study is to balance the need for sufficient training data while preserving enough data to accurately test the model's generalization. With a larger training set (80%), the model has more data to learn from, which helps improve its performance. The 20% testing set is reserved for evaluating the model's ability to generalize to new, unseen data, ensuring that the results are not overly biased toward the training data.

This split is commonly used in deep learning tasks because it offers a good trade-off between training accuracy and generalization. However, there is always a risk of overfitting, particularly when working with complex models like VGG16. Overfitting occurs when a model performs well on the training data but poorly on the test data, indicating that it has learned noise or irrelevant patterns rather than generalizable features. To mitigate this, the model was validated using an additional validation set consisting of 798 images (337 fractured and 461 normal), which helps monitor performance during training and ensures the model does not overfit.

Additionally, data augmentation techniques such as rotation, zooming, and horizontal flipping were applied to artificially increase the diversity of the training set. These techniques are essential for preventing overfitting by exposing the model to a wider variety of input conditions, thereby enhancing its ability to generalize across different types of X-ray images. The dataset used is presented in Figure 2.

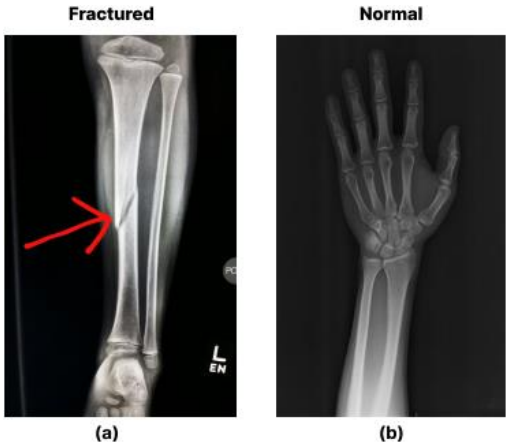

Figure 2. (a) Fractured Human Wrist, (b) Normal Human Wrist.

Figure 2 presents two X-ray images of a human hand, providing a detailed comparison between a fractured hand and a normal hand. Part (a) of the figure illustrates an X-ray of a hand with fractured bones. The image reveals a significant disruption in the continuity of the bones, indicative of a fracture. This type of injury typically results from various forms of trauma, such as accidents, falls, or severe impacts [1]. The X-ray clearly shows the fractured areas, where the bone has lost its structural integrity. These fractures may appear as clean breaks, cracks, or even multiple shattered fragments, depending on the severity of the injury. The visibility of these disruptions is crucial for medical professionals to diagnose the exact nature and extent of the fracture, which in turn informs the appropriate treatment plan, such as casting, splinting, or surgical intervention.

Part (b), in contrast, depicts an X-ray of a hand with normal bones. The bones in this image are intact, with no signs of fractures or damage. The smooth, continuous outlines of the bones indicate a healthy skeletal structure. This intact bone structure is essential for normal hand function, providing the necessary support for movement, dexterity, and strength. The comparison with the fractured hand in part (a) underscores the impact that bone integrity has on overall hand function and health. The normal X-ray serves as a baseline, highlighting what a healthy hand should look like, free from any abnormalities or injuries.

By juxtaposing these two images, Figure 2 effectively demonstrates the stark differences between a fractured and a normal hand. The fractured hand shows clear signs of trauma with interrupted bone continuity, while the normal hand exhibits a flawless bone structure. This visual comparison is instrumental in understanding the implications of bone fractures, emphasizing the importance of timely and accurate diagnosis and the need for proper medical treatment to restore bone integrity and functionality.

2.2 Preprocessing

The preprocessing phase plays a crucial role in image and text data classification. During this phase, we trim or transform parts of the data to remove noise that could interfere with the model training process [11]. Preprocessing employs techniques such as normalization, duplication removal, filling missing values, and data transformation. Handling large and complex data sets requires substantial memory and processing time. By performing preprocessing, we can reduce the data size, thereby speeding up the training time and lessening the load on the computing system [12].

This research applied several preprocessing steps to the dataset to ensure the data was ready for use in the VGG16 model. The preprocessing steps included shifting samples by 20%, enlarging images by 20%, and horizontally flipping images for a portion of the training data. Additionally, data normalization was performed to ensure that the pixel values of the photos were within a uniform range, specifically between -1 and 1 [13].

Normalization using the range of -1 to 1 has several important advantages, especially in the context of training the VGG16 model. Many activation functions used in neural networks, such as tanh and ReLU, can work more effectively with data within the range of -1 to 1 [14]. Tanh is a hyperbolic tangent activation function that outputs values between -1 and 1. It is particularly useful because it centers the data around zero, which helps mitigate issues with saturation that can occur in deeper networks, thereby improving

This is an open access article under the CC BY-4.0 license

convergence during training. ReLU, on the other hand, outputs the input directly if it is positive; otherwise, it outputs zero. This non-linear activation function allows models to handle positive values effectively, promoting sparse activation and speeding up training. ReLU has become popular due to its simplicity and efficiency in dealing with the vanishing gradient problem, making it suitable for deep networks. Furthermore, data normalized within a smaller and symmetrical range can accelerate the training process by making gradient descent more stable and efficient, providing reassurance about the stability and efficiency of our training process.

A formula can be used to find the range value in normalization. The formula is called Feature Scaling, which is used to change the data value to be in the range of 0 to 1. The feature scaling formula is modified to achieve the -1 to 1 in normalization. It becomes like the following Formula 1 [15]. In this research, the process focused on the entire area of the images rather than specific regions of interest (ROI). This approach was taken to ensure that the model could capture all possible features present in the X-ray images, regardless of their location within the frame.

$$x' = 2 \times \left(\frac{x_i - \min(x)}{\max(x) - \min(x)}\right) - 1 \quad (1)$$

Based on Formula 1 [15], the normalization value is calculated by converting the existing data values into a range of 0 to 1. This process is done by subtracting the x_i value with $\min(x)$ and then dividing it by the difference between $\max(x)$ and $\min(x)$. After that, the result will be multiplied by two and reduced by one to get a value from -1 to 1. In this instance, x_i represents the initial data value, while $\min(x)$ and $\max(x)$ denote the smallest and largest values within the dataset, respectively.

This preprocessing stage includes only a few steps as the dataset used has gone through several preprocessing stages before. The dataset has undergone image rotations with rotations ranging from 15% to 90%, which increased the amount of data to about 10,522 images. Therefore, in this study, the preprocessing stage focused more on normalizing and adjusting the images to fit the model to be used.

In addition to normalization, the data samples were also converted to a length and width of 224 pixels to match the input required by the VGG16 model. This process ensures that all images in the dataset have a consistent size, which facilitates the model training process. With this combination of steps, data preprocessing aims to improve data quality and support optimal model performance [16].

2.3 VGG16 Layer Modeling

Advanced deep-learning models in medical imaging have revolutionized diagnostic accuracy and efficiency. The VGG16 convolutional neural network, in particular, has shown great promise in various classification tasks. This study specifically focuses on using the VGG16 model to classify fractured bones. By leveraging the model's ability to capture detailed features in medical images, we strive to enhance the diagnostic process for detecting fractures. This approach has the potential to improve accuracy and offer a reliable tool for medical practitioners, resulting in superior patient outcomes and more streamlined clinical workflows [17].

The deep learning model employed in this study focuses on the VGG16 architecture, which comprises several blocks [18]. The model includes five blocks designed to process the input shape defined by the image matrix vector size. These layers progressively extract and refine features from the input images, facilitating accurate classification of fractured bones. Table 1 summarizes the detailed composition of these blocks. This structured approach enhances the model's ability to learn intricate patterns and ensures robust performance across different datasets, establishing it as a dependable resource for medical diagnostics.

Table 1. VGG16 Architecture Model

Block	Layer Type	Number of Filters	Filter Size	Activation Function
Block 1	Convolutional Layer	64	3x3	ReLU
	Convolutional Layer	64	3x3	ReLU
	Max-Pooling Layer	-	2x2	-
Block 2	Convolutional Layer	128	3x3	ReLU
	Convolutional Layer	128	3x3	ReLU
	Max-Pooling Layer	-	2x2	-
Block 3	Convolutional Layer	256	3x3	ReLU
	Convolutional Layer	256	3x3	ReLU
	Convolutional Layer	256	3x3	ReLU
	Max-Pooling Layer	-	2x2	-
Block 4	Convolutional Layer	512	3x3	ReLU
	Convolutional Layer	512	3x3	ReLU
	Convolutional Layer	512	3x3	ReLU
	Max-Pooling Layer	-	2x2	-
Block 5	Convolutional Layer	512	3x3	ReLU
	Convolutional Layer	512	3x3	ReLU
	Convolutional Layer	512	3x3	ReLU
	Max-Pooling Layer	-	2x2	-

Based on Table 1, The first block begins the feature extraction by applying two convolutional layers, each with 64 filters and a small receptive field, aimed at identifying fundamental image characteristics like edges and textures. Following these layers, a max-pooling layer diminishes the spatial dimensions of the

This is an open access article under the CC BY-4.0 license

feature maps, thus lowering the computational burden and introducing translational invariance [19].

In the second block, feature extraction's complexity increases with convolutional layers containing 128 filters. Like the first block, two convolutional layers are applied sequentially, and then a max-pooling layer is utilized. This architecture enables the model to identify more detailed patterns and characteristics from the input images [20].

The third block extends the feature extraction capability by employing three convolutional layers, each with 256 filters. The additional convolutional layer in this block enhances the model's ability to learn more complex and abstract representations of the input data [21]. The max-pooling layer at the end of this block continues to reduce the spatial dimensions while preserving the learned features.

The fourth block follows a similar pattern but with increased filters in the convolutional layers, set to 512. This increment allows the model to capture finer details and more complex patterns within the images. The arrangement of three convolutional layers, succeeded by a max-pooling layer, guarantees that the extracted features are comprehensive and hierarchically organized.

Finally, the fifth block mirrors the structure of the fourth block, utilizing three convolutional layers with 512 filters each. This block further refines the feature maps, capturing the most intricate details necessary for accurate classification. The concluding max-pooling layer reduces the feature map dimensions, preparing the refined features for subsequent stages of the model, such as fully connected layers and classification tasks [22].

In each convolutional layer, there is a general formula used to calculate the parameters of the layer. First, the convolutional layer needs to determine the value of the width. The width in a convolutional layer is a concept that describes the layer's capacity to process and represent information from the input data. Determining the width value first is crucial because it influences many aspects of the network's operation and performance. The search for the width value can be detailed using Formula 2 [23].

$$width = \frac{c_i {k_i}^2}{g_i} \quad (2)$$

In Formula 2[23], the value of the number of input channels is denoted by c_i, which indicates the depth of the input feature map. c_i signifies the amount of information that can be processed by the layer. Next, k_i represents the kernel size of the convolutional operation, such as 3x3, 5x5, and others. The kernel size determines the spatial area of the input that will be processed by each neuron in the output layer. Larger kernels can capture more spatial details from the input. Finally, g_i represents the number of groups in each convolutional operation. Group convolutions divide the input channels into several smaller groups, where each group is processed independently. After calculating the width using Formula 2 [23], this value is then inserted into Formula 3 [23], which explains how a neural network will be processed within the convolutional layer.

$$H_L = \log(r_{L+1}^2 c_{L+1}) - \sum_{i=1}^{L} \log(\frac{c_i {k_i}^2}{g_i}) \quad (3)$$

Formula 3 [23] consists of two main parts used to measure the entropy or expressive capacity of a convolutional network. The first part, $\log(r_{L+1}^2 C_{L+1})$, describes the total amount of information that can be represented by the feature map in the final output layer, where r_{L+1} is the spatial resolution and c_{L+1} is the number of output channels. The second part, $\sum_{i=1}^{L} \log(\frac{c_i {k_i}^2}{g_i})$, is the summation of the logarithms of each convolutional layer, where c_i is the number of input channels, k_i is the kernel size, and g_i is the number of groups.

The ReLU activation function is employed in this layer to incorporate non-linearity into the model [24]. Without non-linear activation functions, the neural network would only be capable of performing linear operations, which means the network's ability to learn and model complex data would be highly limited. ReLU helps the network learn more complex representations [25].

However, alternative activation functions also contribute uniquely to the performance of neural networks. For example, Tanh, the hyperbolic tangent function scales inputs to a range of -1 to 1, which helps center the data and can lead to faster convergence compared to ReLU in some cases [26]. However, it may still suffer from the vanishing gradient problem for extreme input values. Another function is are sigmoid, the sigmoid function outputs values between 0 and 1, making it suitable for binary classification tasks [26]. However, like Tanh, it can lead to vanishing gradients, particularly in deeper networks, making it less favorable for hidden layers in modern architectures.

The ReLU function is adjusted by introducing additional parameters, allowing further tuning of the standard activation function [27]. The ReLU activation function formula is given in Formula 4 [28]. Based on this formula, gamma and beta can simplify the function to a more conventional form.

$$\sigma(x) = (\alpha \cdot \max\{0, \psi\} + \gamma, \beta \cdot \max\{0, \varphi(-x)\} + \delta) \quad (4)$$

Based on Formula 4 [28], the value of α is a parameter that scales the positive part of the input after applying the function ψ. The value of β scales the negative part of the input after applying the function φ. The parameter γ acts as a bias added to the positive part of the input. Similarly, δ functions like γ; it is a bias added to the negative part of the input after being inverted by φ.

This is an open access article under the CC BY-4.0 license

The function is split into two primary sections, each designed to improve the neural network's flexibility and ability to learn data representations effectively. The first part will handle the computation of positive values, as detailed in Formula 5 [28]. The second part will handle the computation of negative values, as detailed in Formula 5 [28].

$$(\alpha \cdot \max\{0, \psi\} + \gamma) \quad (5)$$

The first part in Formula 5 [28] is responsible for processing the positive input, controlling the scale with α, and adding bias with γ. It handles the positive portion of the input x. The function $\max\{0, \psi\}$ ensures that only the positive values of the input are passed through, while the negative values are set to zero. The parameter α determines the magnitude of the output from the positive portion. By multiplying the result of $\max\{0, \psi\}$ by α, we can control the sensitivity or influence of the positive input on the final output.

Then, the parameter γ is added as an additional bias that can shift the positive output upward or downward. This can help further adjust the output value and provide additional flexibility for model tuning. After understanding what the ReLU function does in the first part, the second part will be detailed in Formula 6 [28].

$$(\beta \cdot \max\{0, \varphi(-x)\} + \delta) \quad (6)$$

The second part of the ReLU function in Formula 6 [28] is responsible for processing the negative portion of the input x. The function $\max\{0, \varphi(-x)\}$ ensures that only the negative values of the input, inverted to positive, are passed through, while the positive values are set to zero. The parameter β determines the magnitude of the output from the negative portion. By multiplying the result of $\max\{0, \varphi(-x)\}$ by β, we can control the sensitivity or influence of the negative input on the final output.

Next, adding bias using the parameter δ will serve as an additional bias that can shift the negative output upward or downward. This provides further flexibility in adjusting the output value. By setting φ and δ to zero, the function becomes more straightforward and more similar to the conventional ReLU function while still maintaining additional flexibility for scaling adjustments through φ and β.

There is a simplification in the use of these two functions by setting γ and δ to zero. This simplification transforms the complex ReLU activation function into a more conventional form. Formula 7 [28] will display the decomposition of this simplified function.

$$\sigma(x) = (\alpha \cdot \max\{0, x\}, \beta \cdot \max\{0, (-x)\}) \quad (7)$$

Based on Formula 7 [28] compared to the previous Formula 4 [28], it is found that the values $\psi(x)$ and $\psi(-x)$ are chosen as identity mappings to x. After this mapping, each part operates as follows: $\alpha \cdot \max\{0, x\}$ multiplies the positive input x by α. If x is greater than 0, the output is αx. If x is less than or equal to 0, the output is 0. Then, the second part acts as an inverted ReLU function that multiplies the negative input x by β. If x is less than 0, the output is $\beta(-x)$. If x is greater than or equal to 0, the output is 0.

Then, In Formula 4 [28], the values of γ and δ are set to zero, resulting in no additional bias shift. This makes the function more straightforward and more similar to the conventional ReLU. However, there is additional flexibility through α and β, which allows for the adjustment of sensitivity to positive and negative inputs separately.

Every block in the model contains a Max-Pooling layer that reduces the feature map dimensions by half, affecting both its width and height [29]. Additionally, max pooling helps the network become more robust to shifts in the position of features within the image. The operation of the max pooling layer in altering dimensions is governed by Formula 8 [19], which determines the output value for each position in the resultant feature map.

$$y_{kij} = \max_{(p,q)\in R_{ij}} x_{kpq} \quad (8)$$

Formula 8 [19] presented indicates that the output value y_{kij} at position (i, j) of the k-th feature map is obtained by taking the maximum value of the elements x_{kpq} within the pooling region R_{ij}. More specifically, the expression $\max_{(p,q)\in R_{ij}} x_{kpq}$ shows that for each position (i, j), there is a pooling region R_{ij} encompassing several elements in the original feature map. Among the elements in this pooling region, the highest value is selected as the output value y_{kij}. This process aims to reduce the dimensionality of the feature map while retaining the most significant features. Max pooling helps convolutional neural networks become more robust to shifts in the position of features within the image, and it also reduces the number of parameters and computational requirements in the network.

The model's compilation process utilizes the Adam optimizer for its efficiency and adaptability in training deep-learning models [30]. Adam, or Adaptive Moment Estimation, is a popular algorithm known for handling gradient changes swiftly and efficiently [31]. It combines the benefits of AdaGrad and RMSProp by updating weights using two types of momentum: the first measures the exponential average of the gradients, and the second measures the exponential average of the squared gradients. To ensure accuracy, Adam applies bias correction to both momenta. This process allows for more stable and faster convergence. Before updating weights with Adam, we calculate the first and second moving momenta using Formulas 9 and 10 [32].

$$m_t = (1 - \beta_1) \sum_{i=0}^{t} \beta_1^{t-i} g_i \quad (9)$$

$$v_t = (1 - \beta_2) \sum_{i=0}^{t} \beta_2^{t-i} {g_i}^2 \quad (10)$$

Formula 9 [32] calculates the first moving momentum, m_t, where β_1 is the exponential decay rate for the first moment, and g_i represents the gradient at each time step

This is an open access article under the CC BY-4.0 license

i. This smooths the gradient estimates, helping to stabilize the training process. The second moving momentum at Formula 10 [32] spilled, v_t, captures the exponential average of the squared gradients. Here, β_2 is the exponential decay rate for the second moment. This second moment helps to scale the gradients, allowing the optimizer to adjust the learning rate adaptively for each parameter, ensuring more efficient convergence. After knowing the values of m_t and v_t, the bias correction of the momentums can now be calculated at Formula 11 [32] and Formula 12 [32].

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{11}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{12}$$

Using the values of m_t and v_t from the previous momentum calculations, the bias correction for the first momentum, given by Formula 11 [32], and for the second momentum by the Formula 12 [32]. Here, β_1 and β_2 are the exponential decay rates used in the initial momentum calculations. This bias correction facilitates the creation of more accurate estimates of the exponential averages of gradients and squared gradients, promoting a more stable and efficient weight update process. After all the values were calculated now, the updated weight can be calculated at Formula 13 [32].

$$w_t = w_{t-1} n \frac{\hat{m}_t}{\sqrt{\hat{v}_t+\varepsilon}} \tag{13}$$

In Formula 13 [32], the value of w_{t-1} is the previous weight, n is the learning rate, $\hat{m}_t$ is the bias-corrected first momentum, $\hat{v}_t$ is the bias-corrected second momentum, and ε is a small constant added to prevent division by zero. This formula ensures that the weight updates are scaled appropriately by the first and second-moment estimates, enabling the optimizer to handle gradient changes more efficiently and ensuring a stable convergence process.

The loss function employed is binary cross-entropy, which is particularly suitable for binary classification tasks such as this one [33]. Additionally, the model's performance is evaluated based on accuracy metrics, providing a clear measure of its classification effectiveness [34]. This combination of optimization, loss function, and performance metrics ensures that the model is both well-tuned and capable of delivering reliable diagnostic results.

2.3 Evaluation

To assess the performance of the VGG16 model in classifying bone fractures, a confusion matrix has been utilized to evaluate the model after training. The confusion matrix provides a comprehensive breakdown of the model's predictions versus the actual classifications [35]. It is a tabular representation that allows us to visualize the performance of a classification algorithm by detailing the counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The confusion matrix will be elaborated upon in Table 2.

Table 2. Confusion Matrix

		Prediction	
Actual		Positive	Negative
	Positive	TP	FN
	Negative	FP	TN

From Table 2, we can calculate several important metrics to evaluate the model's performance: accuracy, precision, recall, and F1 score. First, we will explain accuracy, which measures how accurately the model predicts the correct data points. It reflects the overall effectiveness of the model in classifying instances. In contrast, precision assesses the proportion of true positive predictions among all positive predictions made by the model, indicating the reliability of the positive class predictions. Accuracy will be detailed in Formula 14 [35].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{14}$$

Precision is a critical metric in the evaluation of classification models, particularly in fields where accurate predictions are vital, such as medical diagnostics. Precision quantifies the accuracy of the model's positive predictions by determining the proportion of predicted positive instances that are indeed positive. Formally, it addresses the question: "Of all instances classified as positive, how many are truly positive?" A high precision value indicates that the model is proficient at minimizing false positives, thereby enhancing its reliability. Precision is calculated using the formula shown in Formula 15 [35].

$$Precision = \frac{TP}{TP+FP} \tag{15}$$

Recall is another essential metric for evaluating the performance of a classification model. Often referred to as sensitivity or the true positive rate, recall quantifies the model's ability to correctly identify all relevant positive instances within a dataset. Specifically, it addresses the question: "Of all actual positive instances, how many did the model correctly identify?". Recall is mathematically represented by Formula 16 [35].

$$Recall = \frac{TP}{TP+FN} \tag{16}$$

F1 Score is a vital metric that combines both precision and recall to provide a single performance measure for classification models. The F1 Score is particularly useful in scenarios where there is an imbalance between the positive and negative classes, as it seeks to find the balance between the two metrics. This measure is defined mathematically as the harmonic mean of precision and recall, calculated using Formula 17 [35].

$$F1\ Score = 2 \times \frac{Precision+Recall}{Precision \times Recall} \tag{17}$$

In conclusion, the evaluation of the VGG16 model for classifying bone fractures through the use of a confusion matrix has provided valuable insights into its

This is an open access article under the CC BY-4.0 license

performance metrics. By detailing true positives, false positives, true negatives, and false negatives, the confusion matrix has enabled a comprehensive analysis of the model's classification abilities. The calculated metrics of accuracy, precision, recall, and F1 score serve as critical indicators of the model's effectiveness. Accuracy reflects the overall correctness of predictions, while precision and recall assess the reliability and sensitivity of the positive classifications, respectively.

3. Results and Discussions

This section presents the model's classification results from the previously designed model for analysis and discussion. The results include the confusion matrix, training history, and experiments conducted using various batch sizes. These findings give a thorough insight into the model's accuracy and proficiency in identifying fractured bones.

3.1 Results

The effectiveness of the VGG16 model in medical image classification, particularly for detecting fractures, hinges on its robust training process and performance metrics. By meticulously analyzing the model's training outcomes, we gain valuable insights into its capability to classify bone fractures accurately. The results of this training phase are crucial for understanding how well the model can generalize to new, unseen data, thereby providing a reliable tool for medical diagnostics.

At this stage, we analyze the implementation of the VGG16 layer model for fracture classification. The research involves training the model for ten epochs with a batch size of 128, resulting in a notable classification accuracy of 99.25% and a precision of 98.62%. We obtained this accuracy using a test dataset of 4,083 samples of 3,366 regular and 267 fracture data points. We will also evaluate the training results using a confusion matrix, as illustrated in Table 3.

Table 3. Training Result Matrix

		Prediction	
Actual		Fracture	Normal
	Fracture	237	1
	Normal	1	267

Table 3 illustrates the classification results of the model on the test data, which consists of two classes: Fractured and Normal. The confusion matrix reveals that the model correctly identified 237 samples as Fractured, meaning the model accurately classified 234 indeed fractured samples. Additionally, the model correctly identified 267 samples as Normal, indicating the model accurately classified 267 samples without fractures. However, the model misclassified one sample that was actually Normal as Fractured. Conversely, the model misclassified one sample that was actually Fractured as Normal.

The model training process also records the values of accuracy, validation accuracy, loss, and validation loss throughout the training. During the training using ten batches, the model stores these values in a variable named history. Subsequently, the history will be summarized in Figure 3 to visualize these values. This monitoring aims to observe the model's adjustment to the data during training and evaluate its effectiveness in preventing overfitting and underfitting issues.

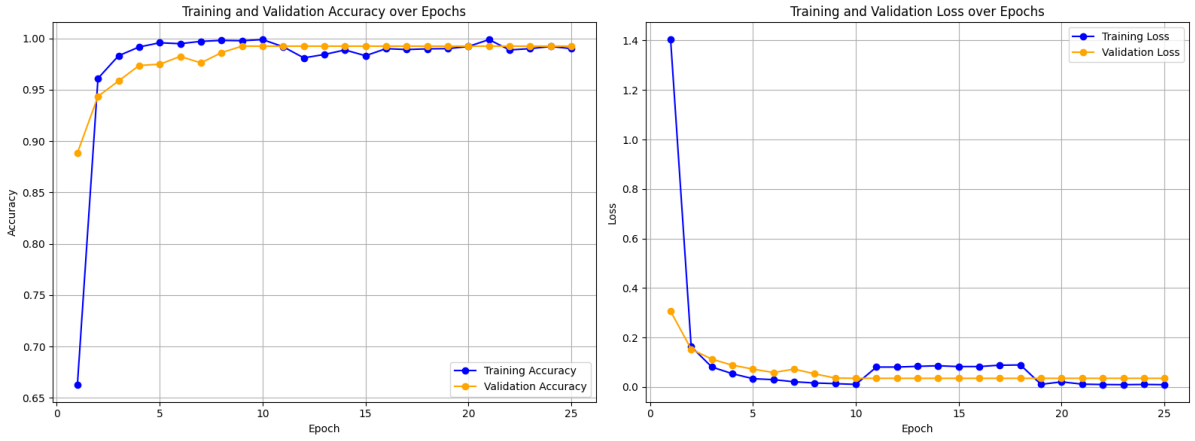

Figure 3. Training History Epochs 1-25

Based on the Figure 3, the initial epoch of the model recorded an accuracy of 66.29% with a loss of 1.4037, highlighting the initial inaccuracies as it began its learning process. Notably, the validation accuracy was higher at 88.85%, accompanied by a validation loss of 0.3058, suggesting some capacity for generalization to unseen data from the outset. As training advanced, considerable improvements were observed in the second epoch, where training accuracy surged to 96.11% and the loss decreased to 0.1631. Validation accuracy also improved to 94.36%, with a validation loss of 0.1527, indicating effective learning and enhanced predictive accuracy.

By the third epoch, the model attained an accuracy of 98.32%, with a further reduction in loss to 0.0805. The validation accuracy reached 95.86%, and the loss fell to 0.1124, reflecting ongoing improvement and accuracy in predictions. In the fourth epoch, training accuracy peaked at 99.17%, accompanied by a low loss of 0.0536. The validation accuracy reached 97.37%, with

This is an open access article under the CC BY-4.0 license

a corresponding loss of 0.0879, suggesting that the model was increasingly reliable in its predictions.

During the fifth and sixth epochs, both training and validation accuracies continued to rise. In the sixth epoch, training accuracy was reported at 99.49%, with a loss of 0.0294, while validation accuracy climbed to 98.25%, with a loss of 0.0581. These findings illustrate the model's growing effectiveness and consistency. By the eighth epoch, training accuracy hit 99.81%, accompanied by a minimal loss of 0.0160, while validation accuracy was at 98.62%, with a loss of 0.0529. Such high accuracy indicates that the model had learned to generalize effectively.

In the ninth and tenth epochs, the model achieved remarkable training accuracies of 99.78% and 99.89%, respectively, with training loss values dropping to 0.0132 and 0.0103. The validation accuracy remained high at 99.25% in both epochs, and loss values stabilized around 0.0349, suggesting that the model had attained high precision and consistent performance.

From the eleventh to the twenty-fifth epochs, the model consistently maintained high accuracy levels with minimal fluctuations. Training accuracy hovered between 98% and 99%, while validation accuracy remained stable at 99.25%, with a low validation loss of approximately 0.0349. These final epochs demonstrate the model's robustness and reliability in predicting fracture classifications, indicating a strong ability to generalize to unseen data with minimal errors.

These training results demonstrate that the model has achieved very high accuracy and minimal error in the training and validation data. The model can generalize patterns from the training data to the validation data, as reflected by the high and consistent validation accuracy and low loss values. The classification report will be summarized in Table 4.

Table 4. Bone Fracture Classification Report

Class	Precision	Recall	F1-Score	Accuracy
Fractured	1.00	0.98	0.99	0.99
Normal	0.99	1.00	0.99	0.99

Table 4 presents the performance metrics for the classification model in distinguishing between fractured and normal cases. The metrics used to evaluate the model include Precision, Recall, F1-Score, and Accuracy, each for both the Fractured and Normal classes. For the Fractured class, the model achieved a perfect Precision of 1.00, indicating that all instances identified as fractured were indeed fractured. This exceptional performance can be attributed to the high-quality dataset of 10,522 X-ray images, which provided a diverse set of examples for the model to learn from, thus enhancing its ability to accurately classify instances. The Recall rate for the Fractured class was 0.98, meaning the model correctly identified 98% of the actual fractured cases.

The VGG16 architecture's strength in feature extraction played a crucial role here; its deep convolutional layers are adept at identifying complex patterns in images, enabling the model to distinguish between fractured and normal bones effectively. The F1-Score, which is the harmonic mean of Precision and Recall, was 0.99 for this class, reflecting a balanced performance. The Accuracy for the Fractured class was also 0.99, suggesting that 99% of the total instances were correctly classified as either fractured or not fractured.

In the case of the Normal class, the model achieved a Precision of 0.99, showing that 99% of the instances identified as normal were truly normal. The Recall rate was perfect at 1.00, indicating that the model identified all the normal instances correctly. The F1-Score for the Normal class was 0.99, indicating a high level of precision and recall. Similarly, the Accuracy for the Normal class was 0.99, which means the model correctly classified 99% of the instances as either normal or not normal. Overall, the model's performance metrics show high precision, recall, F1-score, and accuracy across both classes, demonstrating its effectiveness and reliability in distinguishing between fractured and normal cases.

3.2 Experimental

To enhance the performance of the fracture classification model employing the VGG16 architecture, a number of experiments were done at this stage, involving various hyperparameter adjustments. The hyperparameters that were modified include batch size and the number of epochs. Each experiment was conducted meticulously to ensure that the results obtained provide a clear picture of how changes in hyperparameters affect the model's performance. The outcomes of the five tests carried out are consolidated in Table 5, providing a comprehensive overview of the accuracy and loss values attained for each set of hyperparameters.

Table 5. Experimental Hyperparameter Batch Size

Epochs	Batch Size	Accuracy	Loss
25	256	0.9872	0.0536
	128	0.9925	0.0135
	64	0.986	0.0805
	32	0.9899	0.0433
	16	0.9900	0.0241

In the original experiment, a total of 256 batches were utilized, and the specimen underwent training for ten epochs. The results showed an accuracy of 98.32% with a loss value of 0.0536. The second experiment used the same number of epochs but with a smaller batch size of 128. The accuracy increased to 99.25% with a lower loss value of 0.0135, indicating that reducing the batch size can enhance the model's performance.

In the third attempt, the sample size was reduced to 64, yielding a success rate of 98.69% and an error value of 0.0805. Although the accuracy remained high, the higher loss value suggests that this batch size may not be optimal for training the model. For the fourth trial,

This is an open access article under the CC BY-4.0 license

the batch size was decreased to 32, resulting in a success rate of 98.99% and an error value of 0.0433.

For the fifth experiment, a sample size of 16 was utilized, yielding a success rate of 99.00% and an incorrect value of 0.0241. From these experiments, smaller batch sizes improve model accuracy and reduce loss values. However, a batch size that is too small can lead to significantly higher loss values, as seen with the batch size of 64. Therefore, a batch size of 128 or 16 batch sized for achieving a balance between high accuracy and low loss values.

Based on the analysis of training data across various batch sizes, a clear pattern emerges regarding the impact of batch size on accuracy and loss. The batch size of 128 yields the highest accuracy (0.9901) and the lowest loss (0.0232), suggesting that this size may be the optimal choice in this context. Meanwhile, although a batch size of 16 also achieves an excellent accuracy (0.9900) and a low loss (0.0241), larger batch sizes, such as 256, tend to show less satisfactory accuracy and higher loss (0.9872 and 0.0536, respectively). These findings align with previous research indicating that smaller batch sizes can aid in achieving better generalization, whereas larger batch sizes may expedite training but risk diminishing accuracy if not properly calibrated [36]. This underscores the importance of hyperparameter tuning in deep learning models to achieve optimal performance.

In addition to employing VGG16, this study also compares its performance against other deep learning architectures such as AlexNet, DenseNet121, DenseNet169, and DenseNet201. These models were selected based on their established success in image classification tasks, with each offering unique strengths. AlexNet is recognized for pioneering the use of deep convolutional networks, while the DenseNet architectures (121, 169, and 201) are appreciated for their efficient feature reuse through densely connected layers, which help mitigate the vanishing gradient problem in deep networks. The testing results for these alternative architectures are presented in Table 6.

Table 6. Experiments Using Other Model Architectures

Model Name	Accuracy	Loss
VGG16	0.9925	0.0135
AlexNet	0.8794	0.2930
DenseNet121	0.9139	0.2187
DenseNet169	0.9899	0.0433
DenseNet201	0.9601	0.0241

Table 6 presents a comparison of five deep learning models: VGG16, AlexNet, DenseNet121, DenseNet169, and DenseNet201, evaluating their performance based on accuracy and loss. VGG16 achieves the highest performance, with an accuracy of 99.01% and a minimal loss of 0.0175, highlighting its strong generalization ability. DenseNet169 closely follows with a high accuracy of 98.99% and a slightly higher loss of 0.0433. DenseNet201 also demonstrates solid results, achieving a 96.01% accuracy and a loss of 0.0241, although it falls slightly behind the top-performing models.

Conversely, AlexNet records the lowest accuracy of 87.94% and the highest loss at 0.2930, indicating weaker performance compared to the other models. DenseNet121 surpasses AlexNet with an accuracy of 91.39% and a loss of 0.2187, but it does not perform as well as the more complex DenseNet169. Overall, DenseNet models, especially DenseNet169, strike a strong balance between accuracy and loss, outperforming AlexNet while approaching VGG16's performance levels.

3.3 Discussion

The objective of this work was to improve the accuracy of a fracture classification model by utilizing the VGG16 architecture and conducting a series of tests with different hyperparameters. The experiments highlighted the significant impact of batch size on the model's performance, demonstrating that smaller batch sizes generally improve accuracy and reduce loss values. Specifically, batch sizes of 128 and 16 proved the most effective, achieving an optimal balance between high accuracy and low loss.

The dataset was divided for experimentation using an 80:20 split, with 80% of the data used for training and 20% for testing. The splitting was conducted based on the researcher's own requirements, incorporating experimental setups that included data augmentation techniques. Data augmentation, such as rotation, zooming, and flipping, was applied to increase the diversity of training samples and avoid overfitting. Additionally, specific regions of the X-ray images (normal versus fracture) were considered to ensure a balanced representation of classes. This method helped maintain consistency between normal and fractured bone categories and allowed for more robust testing of the model's ability to generalize across unseen data.

Within the framework of the VGG16 model, the analysis report indicated excellent reliability, recall, and F1-score for both classes. Specifically, the fractured class achieved a precision of 100%, a recall of 98%, and an F1 score of 99%. The average class also exhibited impressive performance metrics. The model achieved a precision of 99%, indicating a high accuracy in identifying the correct class labels. The recall was perfect at 100%, demonstrating that the model successfully identified all instances of the class. Additionally, the F1 score was 99%, reflecting a balanced and robust performance in both precision and recall. These results underscore the model's robust capability to accurately distinguish between fractured and regular instances, as evidenced by the overall accuracy of 99%.

However, there are several challenges and trade-offs associated with the use of VGG16. First, the model is computationally expensive due to its depth and the large number of parameters. This often leads to slower

This is an open access article under the CC BY-4.0 license

training times, particularly when working with high-resolution medical images. Additionally, while smaller batch sizes improved performance, a batch size of 64 led to higher loss values, suggesting that there is a threshold beyond which further reduction in batch size may not yield additional benefits. This highlights the challenge of balancing hyperparameter tuning with computational efficiency.

Comparing the other research [3] using Deep Neural Networks (DNN) achieves 95% accuracy at 10% of test data and 93% at 15% of test data. Other research [4] using the Deep Convolutional Neural Network (DCNN) technique with the AlexNet model, achieved 86.67% accuracy for gaining the testing data. Lastly, research [7] using the Gray Level Co-occurrence Matrix gained an accuracy of 95%. These comparisons will be presented in Table 7.

Table 7. Comparisons Between Latest Research

Model Name	Accuracy
DNN [3]	0.8794
DCNN [4]	0.9139
GLCM [7]	0.9899
VGG16	0.9925

However, it is essential to note that while smaller batch sizes generally improve model performance, petite batch sizes like 64 led to higher loss values despite maintaining high accuracy. This indicates a threshold below which further reduction in batch size may not yield additional benefits and may degrade the model's performance.

In conclusion, this study's findings provide valuable insights into the optimal configuration of hyperparameters for training deep-learning models in medical image classification. The VGG16 model, when fine-tuned with appropriate batch sizes, demonstrates exceptional accuracy and robustness in fracture detection. Future research should explore the balance between batch size and other hyperparameters to enhance model performance further while mitigating computational costs and training time.

4. Conclusions

This study successfully demonstrated the application of the VGG16 architecture for the automatic detection of bone fractures in X-ray images. The model achieved high performance, with an accuracy of 99%, precision of 98.32%, recall of 98%, and an F1-score of 98.16%. These results confirm the potential of deep learning models, particularly VGG16, in enhancing diagnostic accuracy in medical imaging. Additionally, comparative experiments with other models like AlexNet and DenseNet architectures provided valuable insights into how different architectures perform in the context of medical image classification. However, despite these promising results, several limitations must be acknowledged. First, the dataset, while extensive, may not fully capture the diversity of real-world clinical scenarios, such as variations in imaging equipment or patient demographics. This could limit the model's generalizability when applied to different clinical settings. Another limitation is the use of transfer learning, which, while effective, might not be fully optimized for this specific task. Future research should explore custom architectures or domain-specific pre-trained models to further enhance performance. Moreover, while the VGG16 model demonstrated strong accuracy, the computational cost and memory requirements of deep networks remain a challenge. This study did not address these issues in depth, and future work could investigate more efficient models or techniques like model pruning and quantization to reduce the computational overhead. Finally, the study primarily focused on fracture detection but did not explore other important clinical aspects, such as the severity or type of fracture. Expanding the model to provide more nuanced diagnostic insights could significantly increase its clinical utility. In conclusion, while the VGG16 model offers a robust solution for fracture detection, future research should address these limitations by improving model generalizability, reducing computational complexity, and expanding the scope of clinical applications.

Acknowledgements

We sincerely thank Telkom University for its financial support, which made the completion of this research possible.

References

[1] S. Rao Karanam, Y. Srinivas, and S. Chakravarty, “A systematic review on approach and analysis of bone fracture classification,” *Mater Today Proc*, vol. 80, pp. 2557–2562, Jan. 2023, doi: 10.1016/j.matpr.2021.06.408.

[2] J. Iliaens, J. Onsea, H. Hoekstra, S. Nijs, W. E. Peetermans, and W. J. Metsemakers, “Fracture-related infection in long bone fractures: A comprehensive analysis of the economic impact and influence on quality of life,” *Injury*, vol. 52, no. 11, pp. 3344–3349, Nov. 2021, doi: 10.1016/j.injury.2021.08.023.

[3] D. P. Yadav and S. Rathor, “Bone Fracture Detection and Classification using Deep Learning Approach,” in *2020 International Conference on Power Electronics and IoT Applications in Renewable Energy and its Control, PARC 2020*, Institute of Electrical and Electronics Engineers Inc., Feb. 2020, pp. 282–285. doi: 10.1109/PARC49193.2020.236611.

[4] A. Noureen, M. A. Zia, A. Adnan, and M. Hashim, “Analysis and Classification of Bone Fractures Using Machine Learning Techniques,” in *E3S Web of Conferences*, EDP Sciences, Aug. 2023. doi: 10.1051/e3sconf/202340902015.

[5] K. A. Putri, W. Fawwaz, and A. Maki, “Enhancing Pneumonia Disease Classification using Genetic Algorithm-Tuned DCGANs and VGG-16 Integration,” *Open Access Journal*, vol. 6, no. 1, pp. 11–22, 2024, doi: 10.35882/jeemi.v6i1.349.

[6] D. Albashish, R. Al-Sayyed, A. Abdullah, M. H. Ryalat, and N. Ahmad Almansour, “Deep CNN Model based on VGG16 for Breast Cancer Classification,” in *2021 International Conference on Information Technology, ICIT 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 805–810. doi: 10.1109/ICIT52682.2021.9491631.

This is an open access article under the CC BY-4.0 license

[7] D. P. Yadav and G. Sharma, "Human Bone fracture prognosis using Income inequality based Texture Feature and Support Vector Machine," *IOP Conf Ser Mater Sci Eng*, vol. 1116, no. 1, p. 012137, Apr. 2021, doi: 10.1088/1757-899x/1116/1/012137.

[8] M. F. Russe *et al.*, "AI-based X-ray fracture analysis of the distal radius: accuracy between representative classification, detection and segmentation deep learning models for clinical practice," *BMJ Open*, vol. 14, no. 1, Jan. 2024, doi: 10.1136/bmjopen-2023-076954.

[9] J. Oppenheimer, S. Lüken, B. Hamm, and S. M. Niehues, "A Prospective Approach to Integration of AI Fracture Detection Software in Radiographs into Clinical Workflow," *Life*, vol. 13, no. 1, Jan. 2023, doi: 10.3390/life13010223.

[10] Madushani Rodrigo, Mohan Kumar, Abdelaziz Faramawy, and Harsha Arya, "Bone Fracture Multi-Region X-ray Data," Apr. 2024.

[11] G. J. Chowdary, G. Suganya, M. Premalatha, and S. Ganapathy, "Impact Of Machine Learning Models In Pneumonia Diagnosis With Features Extracted From Chest X-Rays Using VGG16," 2021.

[12] M. Abdel-Nasser, J. Melendez, A. Moreno, and D. Puig, "The impact of pixel resolution, integration scale, preprocessing, and feature normalization on texture analysis for mass classification in mammograms," *Int J Opt*, vol. 2016, 2016, doi: 10.1155/2016/1370259.

[13] G. Murtaza *et al.*, "Deep learning-based breast cancer classification through medical imaging modalities: state of the art and research challenges," *Artif Intell Rev*, vol. 53, no. 3, pp. 1655–1720, Mar. 2020, doi: 10.1007/s10462-019-09716-5.

[14] J.-M. Jo, "빅데이터의 정규화 전처리과정이 기계학습의 성능에 미치는 영향 조준모 * Effectiveness of Normalization Pre-Processing of Big Data to the Machine Learning Performance", doi: 10.13067/JKIECS.2019.14.3.547.

[15] I. Izonin, R. Tkachenko, N. Shakhovska, B. Ilchyshyn, and K. K. Singh, "A Two-Step Data Normalization Approach for Improving Classification Accuracy in the Medical Diagnosis Domain," *Mathematics*, vol. 10, no. 11, Jun. 2022, doi: 10.3390/math10111942.

[16] N. ŞENGÖZ, T. YİĞİT, Ö. ÖZMEN, and A. H. ISIK, "Importance of Preprocessing in Histopathology Image Classification Using Deep Convolutional Neural Network," *Advances in Artificial Intelligence Research*, vol. 2, no. 1, pp. 1–6, Feb. 2022, doi: 10.54569/aair.1016544.

[17] P. Desai, J. Pujari, C. Sujatha, A. Kamble, and A. Kambli, "Hybrid Approach for Content-Based Image Retrieval using VGG16 Layered Architecture and SVM: An Application of Deep Learning," *SN Comput Sci*, vol. 2, no. 3, May 2021, doi: 10.1007/s42979-021-00529-4.

[18] S. Chakrabarti *et al.*, *Compressed Residual-VGG16 CNN Model for Big Data Places Image Recognition*.

[19] H. J. Jie and P. Wanda, "Runpool: A dynamic pooling layer for convolution neural network," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 66–76, Jan. 2020, doi: 10.2991/ijcis.d.200120.002.

[20] A. Nasiri, A. Taheri-Garavand, and Y. D. Zhang, "Image-based deep learning automated sorting of date fruit," *Postharvest Biol Technol*, vol. 153, pp. 133–141, Jul. 2019, doi: 10.1016/j.postharvbio.2019.04.003.

[21] M. F. Hashmi, S. Katiyar, A. G. Keskar, N. D. Bokde, and Z. W. Geem, "Efficient pneumonia detection in chest xray images using deep transfer learning," *Diagnostics*, vol. 10, no. 6, Jun. 2020, doi: 10.3390/diagnostics10060417.

[22] A. Zafar *et al.*, "A Comparison of Pooling Methods for Convolutional Neural Networks," Sep. 01, 2022, *MDPI*. doi: 10.3390/app12178643.

[23] X. Shen *et al.*, "DeepMAD: Mathematical Architecture Design for Deep Convolutional Neural Network." [Online]. Available: https://github.com/alibaba/

[24] C. Banerjee, T. Mukherjee, and E. Pasiliao, "The Multi-phase ReLU Activation Function," in *ACMSE 2020 - Proceedings of the 2020 ACM Southeast Conference*, Association for Computing Machinery, Inc, Apr. 2020, pp. 239–242. doi: 10.1145/3374135.3385313.

[25] D. Saikrishna *et al.*, "Pneumonia Detection Using Deep Learning Algorithms," in *Proceedings of 2021 2nd International Conference on Intelligent Engineering and Management, ICIEM 2021*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021, pp. 282–287. doi: 10.1109/ICIEM51511.2021.9445310.

[26] H. Zheng, H. Qin, B. Wang, Z. Wu, M. Xiao, and L. Tan, "Adaptive Friction in Deep Learning: Enhancing Optimizers with Sigmoid and Tanh Function," Aug. 2024.

[27] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," *Ann Stat*, vol. 48, no. 4, pp. 1875–1897, Aug. 2020, doi: 10.1214/19-AOS1875.

[28] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An Empirical Study on Generalizations of the ReLU Activation Function," 2019, doi: 10.1145/3299815.

[29] Z. P. Jiang, Y. Y. Liu, Z. E. Shao, and K. W. Huang, "An improved VGG16 model for pneumonia image classification," *Applied Sciences (Switzerland)*, vol. 11, no. 23, Dec. 2021, doi: 10.3390/app112311185.

[30] H. Sadr, M. M. Pedram, and M. Teshnehlab, "Multi-View Deep Network: A Deep Model Based on Learning Features from Heterogeneous Neural Networks for Sentiment Analysis," *IEEE Access*, vol. 8, pp. 86984–86997, 2020, doi: 10.1109/ACCESS.2020.2992063.

[31] S. R. Islam, S. P. Maity, A. K. Ray, and M. Mandal, "Deep learning on compressed sensing measurements in pneumonia detection," *Int J Imaging Syst Technol*, vol. 32, no. 1, pp. 41–54, Jan. 2022, doi: 10.1002/ima.22651.

[32] D. O. Melinte and L. Vladareanu, "Facial expressions recognition for human–robot interaction using deep convolutional neural networks with rectified adam optimizer," *Sensors (Switzerland)*, vol. 20, no. 8, Apr. 2020, doi: 10.3390/s20082393.

[33]. Usha Ruby Dr.A, "Binary cross entropy with deep learning technique for Image classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 5393–5397, Aug. 2020, doi: 10.30534/ijatcse/2020/175942020.

[34] M. Toğaçar, B. Ergen, Z. Cömert, and F. Özyurt, "A Deep Feature Learning Model for Pneumonia Detection Applying a Combination of mRMR Feature Selection and Machine Learning Models," *IRBM*, vol. 41, no. 4, pp. 212–222, Aug. 2020, doi: 10.1016/j.irbm.2019.10.006.

[35] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, "Multi-label Classifier Performance Evaluation with Confusion Matrix," Academy and Industry Research Collaboration Center (AIRCC), Jun. 2020, pp. 01–14. doi: 10.5121/csit.2020.100801.

[36] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, Dec. 2020, doi: 10.1016/j.icte.2020.04.010.

This is an open access article under the CC BY-4.0 license