Hindawi
International Journal of Intelligent Systems
Volume 2024, Article ID 8997093, 38 pages
https://doi.org/10.1155/2024/8997093

Research Article

A Deep Learning System for Detecting Cardiomegaly Disease Based on CXR Image

Shaymaa E. Sorour,[1,2] **Abeer A. Wafa**,[3] **Amr A. Abohany**,[4] **and Reda M. Hussien**[4]

[1]*Department of Management Information Systems, College of Business Administration, King Faisal University, Al-Ahsa 31982, Saudi Arabia*
[2]*Faculty of Specific Education, Kafrelsheikh University, Kafrelsheikh 33511, Egypt*
[3]*Faculty of Computer and Artificial Intelligence, Helwan University, Helwan, Egypt*
[4]*Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh, Egypt*

Correspondence should be addressed to Shaymaa E. Sorour; ssorour@kfu.edu.sa

Received 14 November 2023; Revised 30 January 2024; Accepted 1 February 2024; Published 23 February 2024

Academic Editor: Vasudevan Rajamohan

The potential of technology to revolutionize healthcare is exemplified by the synergy between artificial intelligence (AI) and early detection of cardiomegaly, demonstrating the power of proactive intervention in cardiovascular health. This paper presents an innovative approach that leverages advanced AI algorithms, specifically deep learning (DL) technology, for the early detection of cardiomegaly. The methodology consists of five key steps, including data collection, image preprocessing, data augmentation, feature extraction, and classification. Utilizing chest X-ray (CXR) images from the National Institutes of Health (NIH), the study applies rigorous image preprocessing operations, including color transformation and normalization. To enhance model generalization, data augmentation is employed, paving the way for two distinct DL models, a convolutional neural network (CNN) developed from scratch and a pretrained residual network with 50 layers (ResNet50), and adapted to the problem domain. Both models are systematically evaluated with five optimizers, revealing the AdaMax optimizer's superiority for the CNN model and AdaGrad's efficacy for the modified ResNet50. The proposed CNN with AdaMax achieves an impressive 99.91% accuracy, outperforming recent techniques in precision, recall, and $F1-$ score. This research underscores the transformative potential of AI in cardiovascular health diagnostics, emphasizing the significance of timely intervention.

1. Introduction

Cardiomegaly, or an enlarged heart, may be a sign of cardiac insufficiency in a variety of medical conditions, including high blood pressure, coronary artery disease, heart valve disease, and pulmonary hypertension. Detecting cardiomegaly early on is of utmost importance, and one of the most widely used noninvasive and affordable medical imaging tests for early diagnosis is chest X-ray (CXR) imaging [1]. Recent advances in deep learning (DL), combined with the availability of comprehensive CXR databases, have greatly improved the performance of computer-aided detection for cardiomegaly. These methods have achieved results that are comparable to those of human radiologists [2–6]. However, it is important to note that current DL-based detection methods rely heavily on binary classification of the entire CXR, using image-level label-dependent learning to detect cardiomegaly. Although classification-based methods have limitations due to the unclear process by which DL algorithms arrive at conclusions, segmentation-based methods can effectively identify the boundaries of the lungs and heart on CXRs, enabling the automatic calculation of the cardiothoracic ratio (CTR). This index is highly valuable for evaluating cardiac enlargement. It is essential to possess a thorough comprehension of the standard ranges for the CTR since these ranges are subject to variation due to factors such as age, gender, and population demographics. By comparing the calculated CTR value to established norms, we can identify possible indications of

cardiac enlargement. An elevated CTR value signals an imbalance between heart size and thoracic dimensions, which can be a sign of cardiomegaly. The CTR is determined by dividing the maximum horizontal cardiac diameter by the maximum horizontal thoracic diameter. The normal range for CTR is between 0.42 and 0.50. If the value is higher than 0.50, it indicates cardiomegaly [7]. To accomplish this, it is important to identify key anatomical landmarks in the chest's posterior-anterior (PA) projection. Calibrated measurement tools are used to determine the maximal horizontal diameter of the cardiac silhouette, which includes the outer boundaries of the heart, such as the left ventricular border and the right atrial margin. At the same time, the thoracic cavity's widest transverse diameter is measured, establishing the dimensions of the anatomical context in which the heart resides. The quotient of these two measurements, the cardiac diameter and the thoracic width, gives rise to the CTR.

Interpreting the CTR requires a nuanced understanding of normative ranges, which vary based on many factors such as age, gender, and population demographics. Comparing the calculated CTR against established norms is crucial in detecting cardiac enlargement. Elevated CTR values indicate an altered equilibrium between heart size and thoracic dimensions, which can signify cardiomegaly. To calculate the CTR, you need to follow these key steps:

(1) *Identify Cardiac Borders*. Discern the outer boundaries of the cardiac silhouette on the CXR image. This involves identifying anatomical landmarks, such as the left ventricular border on the left side and the right atrial border on the right side.

(2) *Measure Cardiac Width*. Using calibrated measurement tools, ascertain the widest horizontal diameter of the cardiac silhouette. This encompasses the distance between the identified borders, encapsulating the entire extent of the heart's dimensions visible on the CXR.

(3) *Measure Thoracic Width*. Simultaneously, measure the widest transverse diameter of the thoracic cage. This measurement corresponds to the broadest dimension of the chest cavity as depicted on the CXR.

(4) *Calculate the CTR*. Divide the measured cardiac width by the measured thoracic width to obtain the CTR value. Mathematically, it is expressed as

$$\text{CTR} = \frac{(\text{Width of the Heart})}{(\text{width of the thorax})}. \quad (1)$$

The CTR value is a unitless ratio that quantifies the heart's size relative to the thoracic cavity. An elevated CTR value suggests cardiomegaly. However, the quantitative datum is not considered in isolation. Its clinical significance is accentuated through contextual integration with patient-specific variables, including symptoms, medical history, and associated diagnostic data. A thorough analysis of an elevated CTR can prompt further clinical exploration, including specialized consultations, corroborative diagnostic investigations, and targeted therapeutic interventions.

In summary, the CTR from CXR is a multidimensional diagnostic tool that provides early insights into cardiomegaly. By embracing quantitative and qualitative aspects of this approach, healthcare practitioners gain a nuanced framework for identifying cardiac enlargement, leading to proactive clinical management and better patient outcomes.

DL is a powerful technique [8] that involves training artificial neural networks to analyse data and make predictions. Its ability to identify complex patterns and accurately predict outcomes based on large datasets has made it increasingly popular across various fields. For instance, DL can be utilized to detect cardiomegaly (enlarged heart) in CXR images, offering a practical solution to diagnose this condition. The importance of DL in the early detection of cardiomegaly through CXR images lies in the following:

(1) *Feature Extraction*. DL models can automatically learn relevant features from CXR images without the need for explicit feature engineering. This enables the model to capture intricate details that might be indicative of cardiomegaly.

(2) *Complex Pattern Recognition*. Cardiomegaly detection involves recognizing subtle variations in heart size, shape, and contour. DL models excel at identifying complex and nonlinear patterns within the data, which may not be easily discernible by traditional methods.

(3) *Scalability*. DL models can be trained on large datasets, allowing them to generalize from a diverse range of CXR images. This scalability enables better performance in identifying various manifestations of cardiomegaly across different patient populations.

(4) *Continuous Learning*. DL models can be fine-tuned and updated as new data become available. This adaptability ensures that the model remains relevant and accurate over time, improving its effectiveness in detecting cardiomegaly.

(5) *Reduction of Human Error*. Automated analysis of CXR images using DL can minimize human subjectivity and errors that may arise from manual interpretation, leading to more consistent and reliable results.

(6) *Early Intervention*. By accurately detecting signs of cardiomegaly in its early stages, healthcare professionals can initiate appropriate interventions, such as further diagnostic tests or treatment plans, to mitigate the progression of cardiovascular diseases.

DL technology for early detection of cardiomegaly involves constructing neural networks with multiple layers that can learn to map CXR images to relevant diagnostic outcomes. These networks are trained using labeled datasets, where CXR images are paired with corresponding labels indicating the presence or absence of cardiomegaly. The learning process involves optimizing the model's parameters to minimize the difference between predicted outcomes and actual labels, thereby enabling the model to generalize to new, unseen CXR images. Overall, the application of DL technology in the early detection of cardiomegaly through CXR images offers

a sophisticated and data-driven approach to improve diagnostic accuracy, potentially leading to enhanced patient outcomes and better management of cardiovascular health.

1.1. Motivations. The primary objective of this paper is to develop a DL model for early identification of cardiomegaly. The paper proposes two distinct DL models to solve a given problem. The first model is a convolutional neural network (CNN) which has been developed from scratch. The second model is based on a pretrained residual network with 50 layers (ResNet50) which has been modified to make it more appropriate for the problem at hand. To evaluate the proposed models, the research considers five types of optimizers, namely, Adaptive Gradient (AdaGrad), Adaptive Moment Estimation (Adam), Adaptive Moment Estimation with Infinity Norm (AdaMax), Nesterov-accelerated Adaptive Moment Estimation (NAdam), and Root Mean Square Propagation (RMSprop). The effects of each optimizer on the performance, in terms of accuracy, precision, recall, *F*1-score, and processing time, of the models are analysed and discussed in detail.

The proposed model comprises seven distinct steps. Initially, a relevant dataset for cardiomegaly is gathered. After that, the unstructured data are preprocessed, converting them into a structured format for classification purposes. Next, the data are split into training and testing sets, with data augmentation being employed to improve the model's predictive abilities. Simultaneously, DL models are utilized for both feature extraction and classification. Finally, each DL model is compiled with five types of optimizers, and their effectiveness is measured using predefined metrics. Based on the evaluation results, the best optimizer for each DL model is determined, ultimately leading to the identification of the best-performing DL model for early cardiomegaly detection.

1.2. Contributions. This paper introduces a pioneering methodology that incorporates both deep learning model diversity and optimization strategy comprehensiveness. The standout features and contributions, particularly considering the noteworthy results obtained, are outlined as follows.

1.2.1. Architectural Innovation

(1) The paper goes beyond conventional methods by employing two distinct DL models—a custom-built CNN and a modified ResNet50.

(2) This dual-model paradigm allows for a nuanced comparison, showcasing the ingenuity of a model created from scratch against the adaptability of a modified pretrained model. The exceptional results affirm the efficacy of both approaches in early cardiomegaly detection.

1.2.2. Optimizer Diversity and Performance Metrics

(1) The study systematically evaluates both models under the influence of five diverse optimization algorithms namely, AdaGrad, Adam, AdaMax, NAdam, and RMSprop.

(2) The comprehensive exploration of optimization strategies is coupled with a multifaceted evaluation using metrics like accuracy, precision, recall, and *F*1-score. The exceptional achievements, such as 99.91% accuracy with AdaMax for the proposed CNN and 99.73% accuracy with AdaGrad for the modified ResNet50, underscore the significance of this dual exploration.

1.2.3. Efficiency as a Decisive Criterion

(1) The paper introduces computational efficiency as a crucial criterion, evaluating the speed of predictions for each model-optimizer combination.

(2) Recognizing AdaMax as the fastest optimizer for the proposed CNN and AdaGrad for the modified ResNet50 not only emphasizes the practical implications for real-time clinical scenarios but also highlights the efficiency gains achieved through strategic optimizer selection.

1.2.4. Optimal Configurations and Model Superiority Determination

(1) The study not only identifies optimal configurations for each model-optimizer pair but also determines the superior model-optimizer combination for both the proposed CNN and the modified ResNet50.

(2) The exceptional results, such as 99.91% accuracy, 100% precision, 99.40% recall, and 99.7% *F*1-score with AdaMax for the proposed CNN, and 99.73% accuracy, 98.8% recall, and 99.1% *F*1-score with AdaGrad for the modified ResNet50, reinforce the practical applicability and superiority of these configurations.

So, these findings establish the study as a pivotal contribution toward improving the effectiveness and practicality of DL in medical image analysis.

1.3. Structure. The rest of the paper is structured into various sections. Initially, the prior research is presented in Section 2. After that, Section 3 discusses a proposed system for early detection of cardiomegaly, which is further divided into two subsections: Section 3.1 presents suggested DL models, while Section 3.2 provides suggested optimizers for each DL model. Proceeding further, Section 4 presents the analysis and experimental findings, which are then compared to some state-of-the-art methods. Finally, the paper concludes in Section 5 by summarizing the findings and suggesting further research.

2. Literature Review

The pursuit of precise and timely identification of cardiomegaly, an imperative indicator of underlying cardiovascular pathologies, is a paramount objective in medical imaging. Within this context, the examination of chest

radiographs, or CXRs, emerges as a pivotal diagnostic modality. However, the inherently intricate and context-dependent nature of cardiomegaly poses challenges to traditional interpretative paradigms. To address this, a surge of interest has been directed toward harnessing the capabilities of DL methodologies, particularly CNNs, to facilitate the automated detection of cardiomegaly from CXR images. This section embarks on an incisive exploration of the scholarly landscape, traversing methodologies underpinning the application of CNNs in this domain as in Table 1. The investigation encompasses the diverse spectra of data augmentation techniques, architectural configurations, and training modalities that underlie the DL paradigms employed in tackling the complexity of cardiomegaly detection. Additionally, an analytical examination of reported evaluation metrics, such as precision-recall curves and *F*1-scores, resonates with the overarching goal of elucidating the advancements and potential limitations of CNN-based strategies in enabling robust and clinically relevant identification of cardiomegaly. The synthesis of these endeavors forms a crucial substrate, illuminating the trajectory of employing DL, specifically in the realm of CXR-based cardiomegaly detection, thus paving the way for a more comprehensive understanding of the scientific landscape and its implications. Lin et al. [9] introduced an innovative classifier for automated cardiomegaly-level screening in frontal PA view CXR images. The multilayer one-dimensional (1D) CNN employed dual-round 1D convolutional processes, enhancing the recognition of normal conditions and different cardiomegaly levels. The classification layer utilized a grey relational analysis-based classifier for computational simplicity. Evaluation through 10-fold cross-validation demonstrated promising results with precision of 97.80%, recall of 98.20%, accuracy of 98.00%, and an impressive *F*1-score of 97.99%. The study utilized datasets from the National Institutes of Health (NIH) CXR image collection, showcasing the potential of advanced image analysis in medical diagnostics. Wu et al. [10] developed a hybrid approach employing both 2D (two-dimensional) and 1D CNN-based classifiers for swift cardiomegaly screening in clinical applications, specifically utilizing frontal PA CXR scans. The methodology integrates 2D and 1D convolutional processes, along with a multilayer interconnected classification network, to enhance feature extraction and pattern recognition tasks. This holistic strategy aims to enrich the original CXR images while eliminating undesired artifacts. The classifier is trained and assessed using 10-fold cross-validation. The experimental results exhibit promising outcomes aligned with the intended medical purpose, boasting a precision of 97.60%, recall of 99.20%, accuracy of 98.40%, and an impressive *F*1-score of 98.38%. Notably, datasets from the NIH CXR image collection were utilized.

Chen et al. [11] presented a novel approach for cardiomegaly detection on CXR images, introducing a dual attention network named CXRDANet. The CXRDANet incorporates a channel attention module (CAM) and a spatial attention module (SAM), strategically enhancing features associated with the lesion region. Experimental results showcase the effectiveness of their technique, achieving an accuracy of 90.50%, a recall of 94.45%, and an *F*1-score of 90.59%. The training and test sets involve nonoverlapping CXR images of cardiomegaly and normal cases sourced from CXR14 and NLM-CXR within the NIH CXR image collection. Zhou et al. [12] employed a transfer learning approach in their study to identify cardiomegaly in CXR images, as detailed in their work. The researchers achieved superior results by utilizing a combination of InceptionV3 and ResNet50 pretrained models, yielding impressive AUCs of 86.0%, surpassing the outcomes of previous similar studies. Their research highlights the effectiveness of employing transfer learning methodologies to develop a CXR computer-aided detection (CAD) system. The proposed technique exhibits an accuracy of 79.7% and an *F*1-score of 80.00%, validated on the CXR8 dataset sourced from the NIH CXR image collection. Innat et al. [13] introduced Cardio-XAttentionNet, a DL network designed for accurate cardiomegaly classification and localization in CXRs. The model incorporates a convolutional attention mapping technique, enhancing the global average pooling (GAP) system through the addition of a weighting term to form an efficient attention mapping mechanism (AMM). This innovation enables the detection of cardiomegaly through both image-level classification and pixel-level localization, even with image-level labeling alone. The proposed attention mapping network's backbone integrates robust ConvNet designs. The best-performing single model demonstrates impressive overall accuracy, recall, *F*-1 measure, and area under the curve (AUC) scores of 87.00%, 85.00%, 86.00%, and 89.00%, respectively. Training of the model utilized the CXR14 dataset from the NIH.

Ajmera et al. [14] presented a DL-based algorithm aimed at automating CTR computation, facilitating swift cardiomegaly diagnosis and enhancing radiological workflow. The algorithm employs the Attention U-Net DL architecture for automatic CTR calculation. An observer performance test was conducted to assess radiologists' efficacy in identifying cardiomegaly, both with and without the assistance of artificial intelligence. The U-Net model exhibits commendable performance metrics, with a recall of 80.00%, precision reaching 99.00%, and an *F*1-score of 88.00%. These results underscore the potential of the DL algorithm in aiding radiologists in the efficient and accurate diagnosis of cardiomegaly. Sarpotdar et al. [15] introduced a DL-based, customized, and retrained U-Net model designed for the diagnosis of cardiomegaly. Their model integrates data preprocessing, image enhancement, image compression, and classification processes to optimize computation time. Utilizing a CXR image dataset for simulation, the study achieves notable diagnostic metrics, including an accuracy of 94%, recall of 96.2%, and specificity of 92.5%. The evaluation was based on the CXR8 dataset sourced from the NIH CXR image collection. Candemir et al. [2] proposed a method for cardiomegaly detection in CXRs, as outlined in their work. The algorithm comprises two key stages: identification of heart and lung regions in CXRs and extraction of radiographic index from their boundaries. A lung detection algorithm, extended for automatic determination of heart

Table 1: Comparison between different early detection methods of cardiomegaly.

Authors (year)	Dataset	Methodology	Results
Lin et al. [9] (2022)	NIH	One-dimensional (CNN)	Precision of 97.80%, recall of 98.20%, accuracy of 98.00%, and *F*1-score of 97.99%
Wu et al. [10] (2022)	NIH	Mix of 1D and 2D CNN	Precision of 97.60%, recall of 99.20%, accuracy of 98.40%, and *F*1-score of 98.38%
Chen et al. [11] (2022)	NIH	CXRDANet	An accuracy of 90.50%, recall of 94.45%, and *F*1-score of 90.59%
Zhou et al. [12] (2019)	NIH	A mix of InceptionV3 and ResNet50	An accuracy of 79.70% and *F*1-score of 80.00%
Innat et al. [13] (2023)	NIH	Cardio-XAttentionNet	Accuracy of 87.00%, recall of 85.00%, *F*1-score of 86.00%, and AUC of 89.00%
Ajmera et al. [14] (2022)	1257 CXRs	Attention U-Net	Recall of 80.00%, precision of 99.00%, and *F*1-score of 88.00%
Sarpotdar et al. [15] (2022)	NIH	U-Net	Accuracy of 94.00%, recall of 96.20%, and specificity of 92.50%
Candemir et al. [2] (2016)	JSRT dataset and Indian dataset	A novel automated method	Accuracy of 77.00% and sensitivity of 77.00%
Bougias et al. [16] (2021)	2000 X-rays	Google's InceptionV3, VGG16, VGG19, and SqueezeNet networks	Sensitivity of 84.00%, specificity of 83.00%, PPV of 83.00%, NPV of 84.00%, and accuracy of 84.50%
Ribeiro et al. [17] (2023)	VinDr-CXR	ResNet50 v2	Accuracy of $91.80 \pm 0.7\%$, precision of $74.00 \pm 2.7\%$, sensitivity of $87.00 \pm 5.5\%$, specificity of $92.90 \pm 1.2\%$, *F*1-score of $79.80 \pm 1.9\%$, and AUROC of $90.00 \pm 0.7\%$

boundaries, was employed to locate these regions. Standard models of heart and lung regions, learned from a public CXR dataset with boundary markings, were registered to patient CXR images to estimate region locations. Traditional and recently published radiographic indexes were utilized for index computation. The classifier combined these indexes, successfully identifying patients with cardiomegaly with an accuracy of 77.00% and sensitivity (recall) of 77.00%. The evaluation was conducted on the JSRT Set (compiled by the Japanese Society of Radiological Technology), comprising 247 CXRs, and the Indiana Set, consisting of approximately 4,000 frontal and lateral CXRs with various lung abnormalities. From the latter, 250 frontal CXRs with cardiomegaly and 250 normal cases were selected for testing. In a recent investigation conducted by Bougias et al. [16], the analysis of CXR aimed to discern cases of cardiomegaly. The study utilized a dataset comprising 2000 CXRs, evenly divided between normal and confirmed cardiomegaly cases. Several DL networks, including Google's InceptionV3, VGG16, VGG19, and SqueezeNet, were employed to extract deep features from the CXRs. Subsequently, these extracted features were utilized for classifying CXRs as either indicative or not indicative of cardiomegaly through a logistic regression algorithm. Among the DL networks, VGG19 demonstrated the most favorable results, achieving an overall accuracy of 84.5%. The other networks exhibited varying accuracy ranges, with sensitivities ranging from 64.1% to 82%, specificities ranging from 77.1% to 81.1%, positive predictive values (PPVs) ranging from 74% to 81.4%, and negative predictive values (NPVs) ranging from 68% to 82%.

Ribeiro et al. [17] introduced a DL model designed to detect cardiomegaly using CXR images, accompanied by an evaluation of three local explainable methods: Grad-CAM, LIME, and SHAP. The study's findings reveal that their DL model achieved an accuracy of $91.8 \pm 0.7\%$, precision of $74.0 \pm 2.7\%$, sensitivity of $87.0 \pm 5.5\%$, specificity of $92.9 \pm 21.2\%$, $F1$-score of $79.8 \pm 21.9\%$, and an area under the receiver operating characteristic curve (AUROC) of $90.0 \pm 20.7\%$. The evaluation was conducted on the VinDr-CXR dataset, comprising CXR scans obtained retrospectively from two major hospitals in Vietnam. Table 1 summarizes the findings of the literature review.

Singh et al. [18] proposed using DL techniques to diagnose glaucoma, which is the second leading cause of permanent blindness. Their study showed that Inception-ResNet-v2 and Xception models outperformed other models in detecting glaucoma from fundus pictures. The automated system has great potential to improve early detection of glaucoma while reducing human efforts and time.

In a recent study by Khanna et al. [19], CT images and test kits were utilized to predict COVID-19 with success. By employing AI-powered chest X-ray analysis, automated analysis has been shown to greatly enhance accuracy and speed up the diagnostic process. The most dependable models, including the ensemble deep transfer learning CNN model and hybrid LSTM-CNN, have demonstrated the ability to deliver prompt and precise predictions, offering a meaningful advancement for clinical practice.

Khanna et al. [19] presented three innovative CNNs that can detect and classify diabetic retinopathy into five categories. The proposed models outperform many existing models in terms of classification, achieving a maximum accuracy of 0.9545, a maximum $F1$-score of 0.9685, a maximum sensitivity of 0.9566, and a maximum AUC score of 0.9769. The experiment results demonstrate that the proposed models are effective in identifying retinal issues in diabetic patients, leading to better diagnosis and preventing vision loss.

Table 1 introduces the state-of-the-art studies in the area of detecting cardiomegaly disease based on CXR images.

In discussing medical and disease data, it is crucial to prioritize accuracy and sensitivity since human life is at stake. Previous studies have shown that many of them need more balanced data, leading to subpar results in addressing medical problems. Our paper aims to address this issue by introducing changes to the same dataset used in these studies to balance the data as in Section 4.1 and achieve more precise and reliable outcomes. Our goal is to explore multiple DL models for early cardiomegaly detection using different optimizers to determine the best one for each model. By doing so, we aim to present a model that outperforms previous studies in the same field.

3. The Proposed Methodology

An early detection system for cardiomegaly is proposed in this paper, which utilizes specific DL models. The system involves several key steps, starting with the preprocessing of CXR images using a dedicated pipeline. This pipeline includes techniques such as data resizing, labeling, normalization, and color transformation. The preprocessed dataset is then divided into training and testing sets, and data augmentation is performed for both sets. The recommended DL models are finally constructed and trained based on these sets, and their performance is evaluated. The DL approach that has been suggested involves the use of two DL models, namely, CNNs and ResNet50. Additionally, five different optimizers: AdaGrad, Adam, NAdam, AdaMax, and RMSprop, will be utilized to observe their impact on each proposed model and its outcomes. Automated feature extraction and classification tasks are carried out by the DL models presented. The training accuracy and validation loss values are evaluated periodically. After that, the effectiveness of the available DL models is measured in terms of various assessment measures such as accuracy, precision, recall, $F1$-score, and processing time. The general design of the suggested system for early detection of cardiomegaly is illustrated in Figure 1.

3.1. Suggested Deep Learning Models. Two types of DL models, namely, CNNs and ResNet50 models, are recommended for use. These models are briefly explained in the following subsections. Additionally, each of the models is compiled with five different optimizers, namely, AdaGrad, Adam, NAdam, AdaMax, and RMSprop, which are also briefly explained in Section 3.2. The primary objective of using these models with these optimizers is to determine

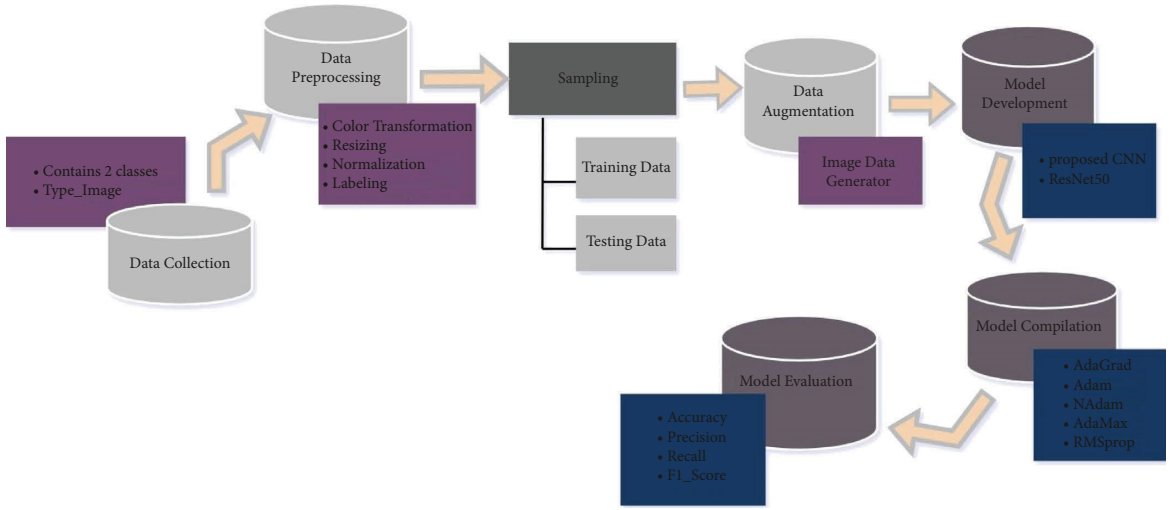

Figure 1: The overall design of the proposed system for early detection of cardiomegaly.

how each optimizer affects each proposed model and identify the best optimizer from the five that balances testing time and detection accuracy.

3.1.1. Convolutional Neural Network. Convolutional neural networks (CNNs) have brought about a revolution in the field of DL, with their incredible success in various domains such as image classification [20], object detection [21], and image segmentation [22]. As a result, there has been a significant increase in interest from both academia and industry in recent years. CNNs aim to make use of the spatial hierarchies in the data by utilizing convolutional and pooling layers, as explained in [23]. Convolutional layers are considered the key component of CNNs, which apply convolution operations to the input data through the use of a small filter, also known as a kernel that is positioned over the spatial dimensions of the input image. For a 2D input image and a 2D filter, the convolution operation can be mathematically expressed as

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m.j+n) * K(m,n), \tag{2}$$

where $S(i,j)$ is the value at position (i,j) in the output feature map, I represents the input image, and K is the filter/kernel. The summation runs over all m and n values that correspond to the filter's dimensions. After convolution, nonlinear activation functions like rectified linear unit (ReLU) are often applied elementwise to introduce nonlinearity into the model. Mathematically, the ReLU function is defined by the following equation:

$$\mathrm{ReLU}(x) = \max(0,x). \tag{3}$$

Pooling layers, such as MaxPooling, are considered another crucial component of CNNs. Pooling layers help reduce the spatial dimensions of the feature maps while retaining important information. MaxPooling, for example, involves selecting the maximum value within a small region of the input. This operation is expressed as

$$\mathrm{MaxPooling}(i,j) = \max\begin{pmatrix} I(i,j) & I(i,j+1) \\ I(i+1,j) & I(i+1,j+1) \end{pmatrix}. \tag{4}$$

The combination of convolutional layers and pooling layers helps CNNs capture and hierarchically represent features at different levels of abstraction. This ability is crucial for recognizing complex patterns and structures within images. In addition to convolutional and pooling layers, CNNs often contain fully connected layers near the end of the architecture. A fully connected layer, also known as a dense layer, is a fundamental building block in neural networks. It is a type of layer where each neuron or node is connected to every neuron in the previous layer, forming a fully connected graph. This layer is commonly used to capture complex relationships and patterns within the data, making it suitable for tasks such as classification and regression. Mathematically, let us consider a fully connected layer that takes input from a previous layer with n neurons. Each neuron in the fully connected layer applies a weighted sum of the input values followed by an activation function to produce its output. The output y_i of the i^{th} neurons in the fully connected layer can be calculated using the following equation:

$$y_i = \mathrm{activation}\left(\sum_{j=1}^{n} w_{ij}\,.\,x_i + b_i\right), \tag{5}$$

where w_{ij} is the weight connecting the j^{th} neuron in the previous layer to the i^{th} neuron in the fully connected layer, x_j is the output of the j^{th} neuron in the previous layer, b_i is the bias term for the i^{th} neuron in the fully connected layer, and the activation function introduces nonlinearity to the output.

In the context of neural networks, multiple fully connected layers can be stacked together, typically with activation functions in between, to form deep architectures. The deep neural network can learn increasingly complex and abstract representations of the input data as it goes through these layers. Fully connected layers have a significant number of parameters, often leading to a high computational load and potential overfitting. Regularization techniques like dropout and weight decay are commonly employed to mitigate overfitting. Additionally, global average pooling and convolutional layers have gained popularity, especially in convolutional neural networks, as they reduce the number of parameters and capture spatial hierarchies more effectively. A typical CNN architecture along with these layers is depicted in Figure 2. In this paper, the suggested CNN model involves several convolutional and pooling layers to discover the preprocessed images' features and do the classification task. The following layers make up the recommended CNN model:

(1) The first layer is a convolutional (Conv2D) layer with 32 filters of size 3 X 3, and incorporates the ReLU activation function. This is formally presented in equation (3), with the layer receiving an input tensor with dimensions of (224, 224, 3).

(2) The second layer is a max-pooling (MaxPooling2D) layer, which has the maximum output from the neighborhood, with a pool size of 2×2. This layer reduces the spatial dimensions of the feature maps outputted by the convolutional layer, as explained in equation (5).

(3) The third layer is another convolutional layer (Conv2D) with 64 filters and the ReLU activation function.

(4) The fourth layer is another max-pooling (MaxPooling2D) layer with a pool size of 2×2.

(5) The fifth layer is another convolutional layer (Conv2D) with 128 filters and the ReLU activation function.

(6) The sixth layer is another max-pooling (MaxPooling2D) layer with a pool size of 2×2.

(7) The seventh layer is another convolutional layer (Conv2D) with 128 filters and the ReLU activation function.

(8) The eighth layer is another max-pooling (MaxPooling2D) layer with a pool size of 2×2.

(9) The ninth layer is a flattened layer (Flatten) that converts the 2D feature maps outputted by the previous layer into a 1D vector.

(10) The tenth layer is a dense layer (Dense). It is fully connected because all input neurons are considered by each output neuron, with 128 units and a ReLU activation function.

(11) In the final layer, there is a dense layer that contains 2 units and utilizes the softmax activation function. This function generates a probability distribution for the two classes of data. The softmax function is proficient in transforming a vector of K real values into a vector of K real values that add up to 1. It can handle values that are positive, negative, zero, or greater than one and transforms them into values ranging from 0 to 1. The mathematical representation for softmax is represented in the following equation:

$$f_j(Z) = \frac{e^Z j}{\sum_k e^Z k}. \tag{6}$$

In conclusion, following the comprehensive delineation of the 11-layer CNN architecture proposed for precocious detection of cardiomegaly, the pseudocode for the recommended CNN framework is delineated in Algorithm 1 and 2 that provide the pseudocode for the proposed CNN model and the modified ResNet50 model, respectively. In addition, the architecture of the suggested CNN model is illustrated in Figure 3, which provides a visual representation of a neural network's architecture. It offers a structured and static view of how the various layers are connected and how data flow through the network. It illustrates the arrangement of input and output layers, hidden layers, and any branching or merging of connections. It captures the overall structure of the mode and gives a snapshot of the model's design, making it suitable for conveying a high-level overview of the neural network's layout and connections, and Figure 4 offers an interactive and dynamic representation of a neural network's architecture. This interactive nature empowers users to explore intricate model configurations, understand data flow, and delve into specific layer properties, making it particularly beneficial for educational purposes and comprehending complex neural network architectures.

3.1.2. Residual Network with 50 layers (ResNet50). Learning through residual mechanisms is a simplified process that focuses on the difference between the input and desired mapping rather than attempting to learn the entire mapping directly. This approach allows for effective learning, even in situations where the network is extremely deep. In mathematical terms, input is denoted as x, and output is denoted as $H(x)$, as shown in Figures 5–14, and can be calculated using the following equation:

$$H(x) = F(x) + X, \tag{7}$$

where $F(x)$ represents the residual function that the block needs to learn. By rearranging equation (7), we can see that the network is learning to approximate the residual function $F(x)$ as shown in the following equation:

$$F(x) = H(x) - X. \tag{8}$$

The ResNet50 is a convolutional neural network architecture that has played a significant role in advancing the field of computer vision and image recognition. It was

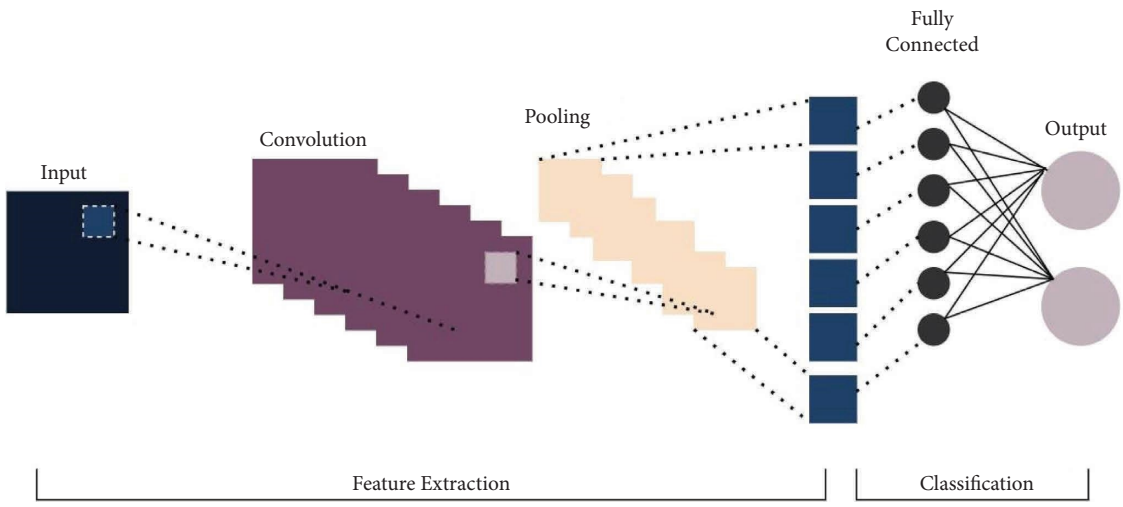

Figure 2: The CNN architecture.

developed by He et al. [24]. The ResNet50 addresses the problem of vanishing gradients, which can often hinder the learning of intense neural networks. This problem occurs when the gradients become very small during back-propagation as they pass through layers, making it challenging for the network to learn effectively. To solve this issue, ResNet50 introduces residual blocks, which help the network learn identity mappings more easily, thereby allowing for the training of extremely deep networks. ResNet50 architecture consists of 50 layers as shown in Figure 15 and is organized in a series of building blocks, each comprising multiple convolutional layers, batch normalization, and activation functions. These blocks are repeated multiple times, with shortcut connections (also known as skip connections) that allow the gradient to flow directly through the layers, thereby mitigating the vanishing gradient problem. In summary, ResNet50, Figure 15, is a groundbreaking neural network architecture that tackles the challenges of training very deep networks. By introducing residual blocks with skip connections, it enables the efficient training of networks with dozens of layers. Mathematically, ResNet50 employs the concept of residual functions, which helps the network learn the difference between the desired output and the input, making it easier to optimize and train deep networks effectively.

In this paper, alterations were applied to the ResNet50 architecture to adapt it for the targeted classification task, as depicted from Figure 5 to 13 , and Figure 16, as follows:

(1) *Exclusion of Top Classification Layer*. The ResNet50 model, as originally designed, includes a classification layer with 1000 units, suitable for classifying images into one of the 1000 categories in the ImageNet dataset. In this paper, the top classification layer was excluded. This means that the final classification layer of the ResNet50 model is not used.

(2) *Freezing Layers*. After excluding the top classification layer, the code iterates through all the layers in the ResNet50 model and sets their trainable attribute to False. By doing so, the weights of these layers are frozen, preventing them from being updated during subsequent training. This step retains the pretrained feature extraction capabilities of ResNet50 while preventing them from being altered during the customization process.

(3) *Additional Layers for Customization*. The proposed model then proceeds to add new layers to the model, which are fully connected (Dense) layers. These new layers allow you to fine-tune the model for the specific classification task. The original ResNet50 features are followed by these custom layers, which help capture more task-specific patterns and information.

(4) *Flatten Layer*. A Flatten layer is inserted after the frozen ResNet50 layers. This layer reshapes the output from the previous layers into a one-dimensional vector, preparing the data for the subsequent fully connected layers.

(5) *Dense Layers*. Three additional dense layers are added, each with 100 units and ReLU activation function. These layers introduce nonlinearity to the model and enable it to learn more intricate patterns and relationships in the data.

(6) *Output Layer*. Finally, a dense layer with 2 units and softmax, equation (6), activation function is added. This layer produces class probabilities for the two classes in a binary classification problem. The softmax activation ensures that the predicted probabilities sum up to 1, providing a probability distribution over the classes.

Overall, the changes made to the ResNet50 model can be seen in Figure 16, which is a modified version of Figure 14 which involves excluding the original top classification layer, freezing the ResNet50 layers to preserve their pretrained features, adding custom fully connected layers for task-

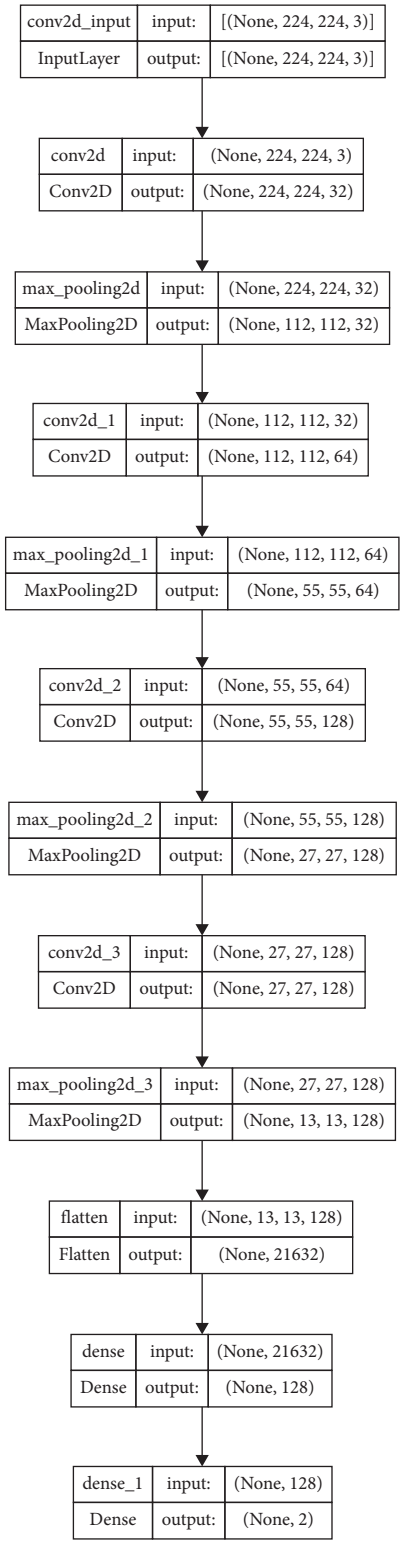

Figure 3: The architecture of the suggested CNN model.

specific learning, and creating a new output layer for your specific classification problem. These modifications enable the utilization of the pretrained ResNet50 features, while customizing the model for the specific image classification requirements, with the architecture shown in Figure 15 and the alterations detailed in Figure 17.

ResNet50 model is specifically chosen for this problem rather than the other pretrained model because of the following:

(1) *Residual Connections for Gradient Flow*. ResNet50 introduces residual connections, alleviating the vanishing gradient challenge inherent in deep neural networks. This architectural innovation facilitates smoother gradient flow during backpropagation, enabling the effective training of significantly deep models.

(2) *Depth Scaling and Performance Maintenance*. ResNet50 exhibits the remarkable capability to scale in depth without encountering the degradation problem observed in shallower architectures. This phenomenon underscores the network's resilience to diminishing performance with increasing depth.

(3) *Efficient Parameter Utilization*. ResNet50 achieves competitive performance with a reduced parameter count, implying efficient extraction and utilization of hierarchical features. This attribute underscores the network's ability to discern salient patterns with parsimonious parameterization.

(4) *Transfer Learning Prowess*. Pretrained on expansive datasets like ImageNet, ResNet50 serves as a potent feature extractor, demonstrating its effectiveness in transfer learning across diverse image classification tasks. The network's learned representations contribute significantly to enhanced generalization.

(5) *Architectural Adaptability*. The architecture of ResNet50, featuring skip connections, imparts adaptability beyond image classification. Its applicability extends seamlessly to various computer vision tasks, encompassing object detection and segmentation.

(6) *Benchmark Performance*. ResNet50 consistently attains state-of-the-art performance in benchmark datasets and competitions, underscoring its efficacy in discerning intricate patterns and features within images.

Figures 5–13 and 16 provide a visual representation of the architecture of a neural network. They offer a structured and static view of how the various layers are connected and how data flow through the network. These figures illustrate the arrangement of input and output layers, hidden layers, and any branching or merging of connections. They also capture the overall structure of the model, giving a snapshot of the neural network's design, which makes it suitable for conveying a high-level overview of the layout and connections of the model.

3.2. Suggested Optimizers. Optimization and optimizers play a crucial role in training DL models effectively [25–29]. The main objective of the optimization is to discover the collection of model parameters that can reduce a specified loss function. Optimizers refer to algorithms that aid in directing the modifications of these parameters while undergoing the training procedure.

Here is a closer look at optimization and some commonly used optimizers in DL:

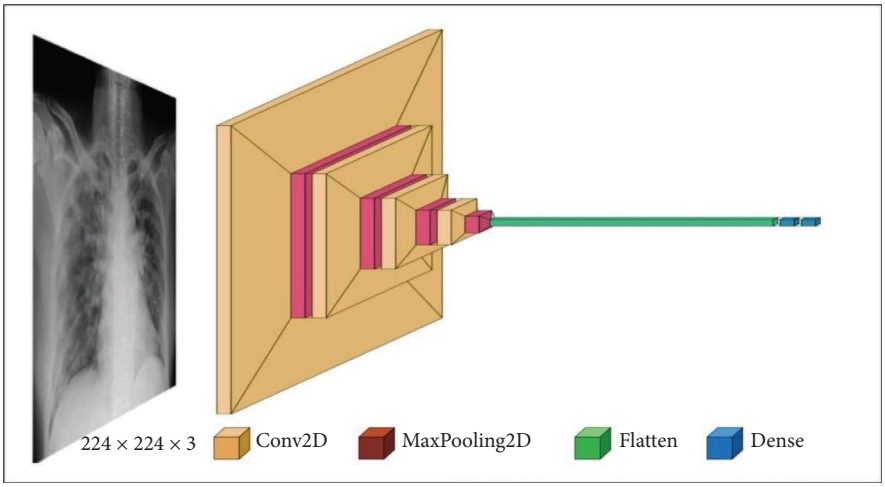

Figure 4: The suggested CNN model visualization.

Input:
Cardiomegaly dataset
T—fixed number of allowed epochs
Output:
One of the two classes
(1) Start
(2) Resize images of the cardiomegaly dataset to 224×224;
(3) Augment the training images with strategies, including random rotation, random zooming of images, random flipping of images in both horizontal and vertical directions, and random width and height shifting of images; shuffle and split the resized images into training, validation, and testing image datasets;
(4) $t \leftarrow 1$;
(5) while $t < T$ do
(6) for batch size $b = 1:85$ do
(7) Train the model on the augmented training images dataset for each batch b based on the fitness function fit();
(8) Compile the model with five types of optimizers, which are AdaGrad, Adam, NAdam, AdaMax, and RMSprop;
(9) end for
(10) Evaluate the training accuracy value and the validation loss value for each epoch t;
(11) if the validation loss value does not enhance for certain epochs then
(12) Go to step 5
(13) end if
(14) end while
(15) Evaluate the trained model on the testing images dataset based on the predict() function to obtain predictions for the testing images;
(16) Calculate the accuracy score using the predicted labels and the true labels;
(17) Choose one of the two classes;
(18) Compare the results of each optimizer and find the best to this proposed model;
(19) End

Algorithm 1: The proposed CNN model.

(1) *Optimization Problem*. In DL, the optimization problem [30] involves finding the values of model parameters that minimize a loss function. This loss function quantifies the difference between the model's predictions and the actual target values. The optimization process aims to iteratively adjust the model's parameters to reduce the loss and improve its performance on the training data.

(2) *Gradient Descent*. Gradient descent [31–34] is the foundation of most optimization algorithms in DL. It involves calculating the gradient (partial derivatives) of the loss function with respect to each parameter and updating the parameters in the direction that reduces the loss. The general update rule is

$$\text{parameter} = \text{parameter} - \text{learning}_r\text{ate} * \text{gradient}. \tag{9}$$

Here, the learning rate controls the step size of each update.

Input:
Cardiomegaly dataset
T—fixed number of allowed epochs
Output:
One of the two classes
(1) Start
(2) Resize images of the cardiomegaly dataset to 224×224;
(3) Augment the training images with strategies, including random rotation, random zooming of images, random flipping of images in both horizontal and vertical directions, and random width and height shifting of images;
(4) Shuffle and split the resized images into training, validation, and testing image datasets;
(5) Use the pretrained model ResNet50 and make some changes which are, excluding the original top classification layer, freezing the ResNet50 layers to preserve their pretrained features, adding custom fully connected layers for task-specific learning, and creating a new output layer for your specific classification problem;
(6) $t \leftarrow 1$;
(7) while $t < T$ do
(8) for batch size $b = 1:32$ do
(9) Train the model on the augmented training images dataset for each batch b based on the fitness function fit();
(10) Compile the model with five types of optimizers, which are AdaGrad, Adam, NAdam, AdaMax, and RMSprop;
(11) end for
(12) Evaluate the training accuracy value and the validation loss value for each epoch t;
(13) if the validation loss value does not enhance for certain epochs then go to 7
(14) end if
(15) end while
(16) Evaluate the trained model on the testing images dataset based on the predict() function to obtain predictions for the testing images;
(17) Calculate the accuracy score using the predicted labels and the true labels;
(18) Choose one of the two classes;
(19) Compare the results of each optimizer and find the best to these proposed models;
(20) End

Algorithm 2: The modified ResNet50 model.

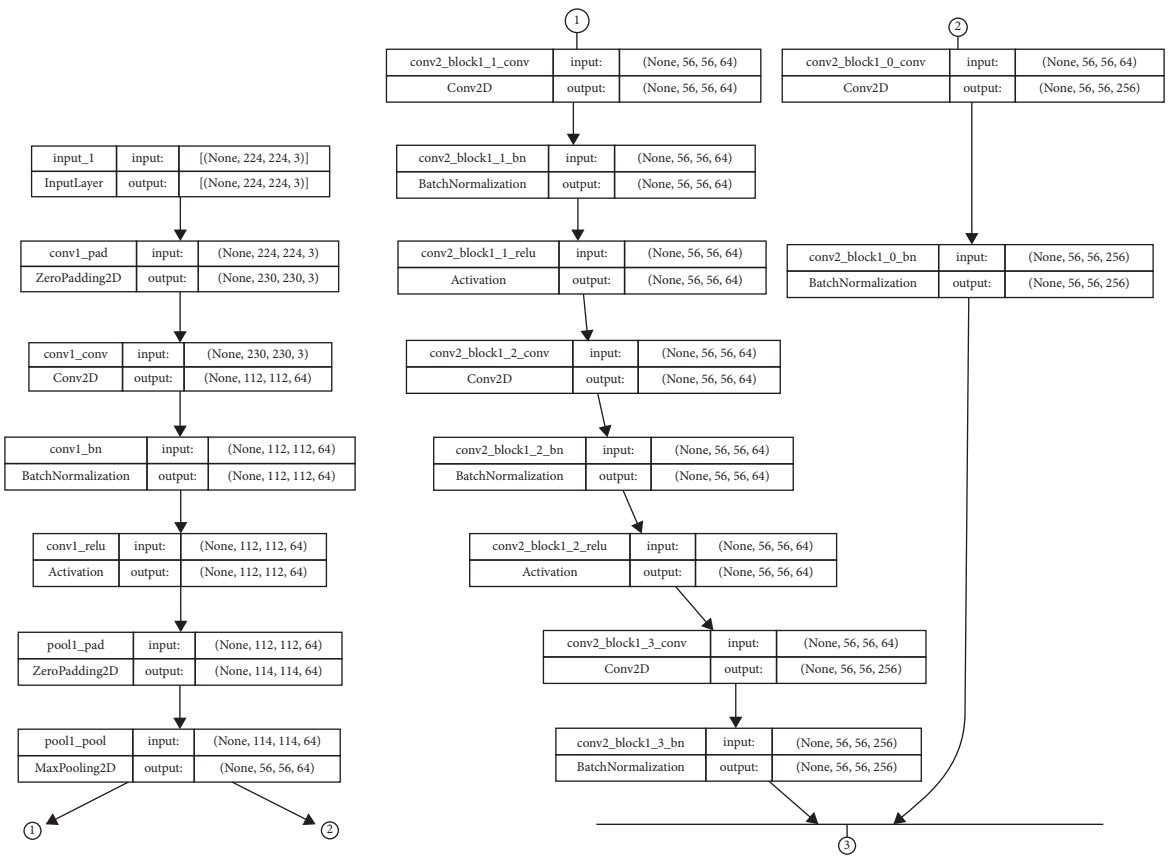

Figure 5: ResNet50 model.

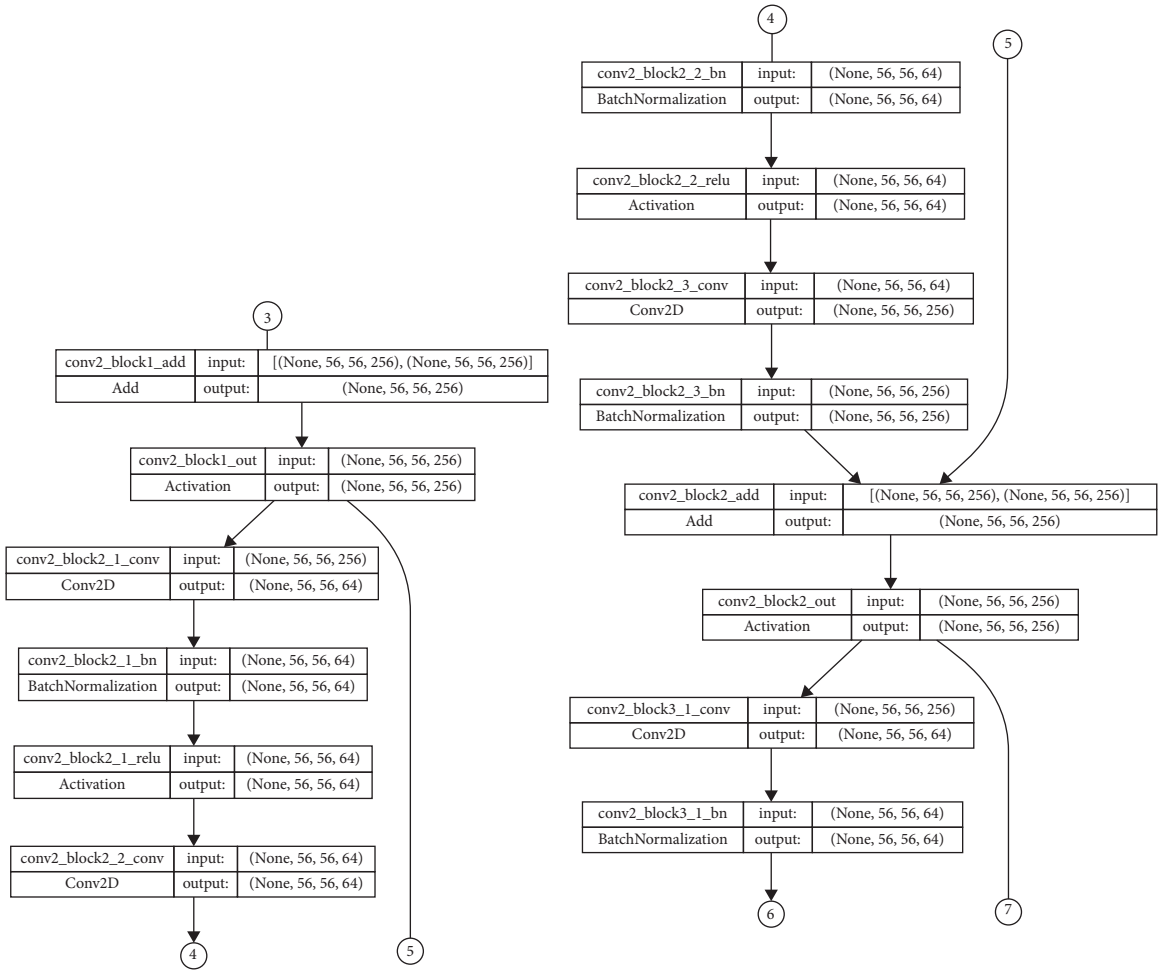

Figure 6: ResNet50 model (cont.).

(3) *Stochastic Gradient Descent (SGD)*. Stochastic gradient descent [35–37] is a variation of gradient descent where instead of computing the gradient using the entire training dataset, it is computed using a randomly selected mini-batch of the data. This introduces randomness and helps the optimization process escape local minima and converge faster.

   (i) *Optimizer Algorithms*. Several optimizer algorithms [25, 26, 38, 39] enhance the basic gradient descent approach. These algorithms incorporate adaptive learning rates, momentum, and other techniques to improve convergence and training efficiency. Some popular optimizers include AdaGrad, Adam, NAdam, AdaMax, and RMSprop, which will be explained in detail in the following subsubsections.

   (ii) *Learning Rate Scheduling*. Dynamic adjustment of the learning rate during training is known as learning rate scheduling [40–43]. It involves decreasing the learning rate over time to allow the optimization process to converge more accurately. Common scheduling strategies include step decay, exponential decay, and learning rate warmup.

   (iii) *Regularization*. Regularization techniques [44–47] like $L1$ and $L2$ regularization add penalty terms to the loss function to prevent overfitting. They discourage large parameter values and promote simpler models that generalize better to new data.

   (iv) *Second-Order Optimization*. While less commonly used in DL due to computational complexity, second-order optimization methods [5, 48–50] like Newton's method and variants use second-order information (Hessian matrix) to guide updates. These methods can converge faster but require more computational resources.

The choice of optimizer depends on the specific problem, model architecture, and dataset characteristics. Experimentation and tuning are necessary to find the most suitable optimizer and hyperparameters for a given task.

3.2.1. AdaGrad. Adaptive Gradient (AdaGrad) algorithm [51–54] is a gradient-based optimization algorithm that aims to improve the efficiency of learning in the context of training machine learning (ML) models, especially in scenarios involving sparse data and features. One of the central

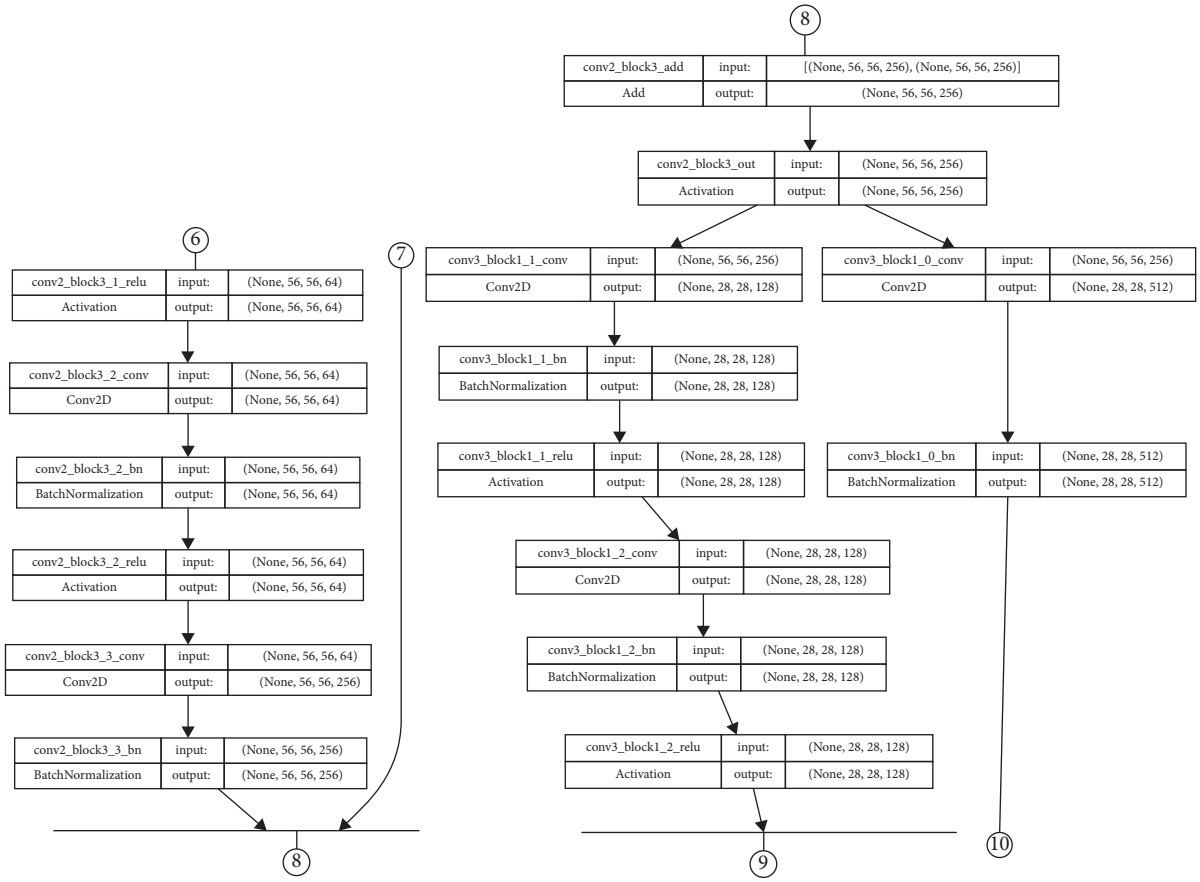

Figure 7: ResNet50 model (cont.).

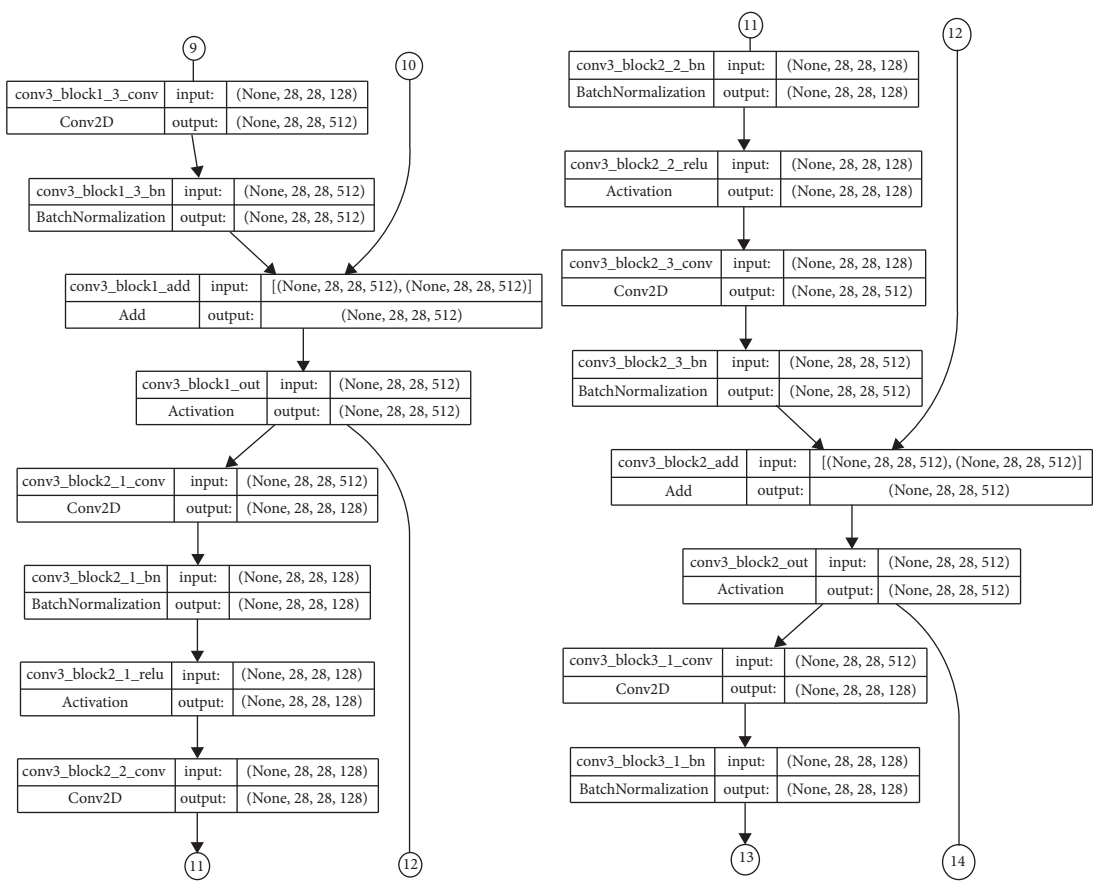

Figure 8: ResNet50 model (cont.).

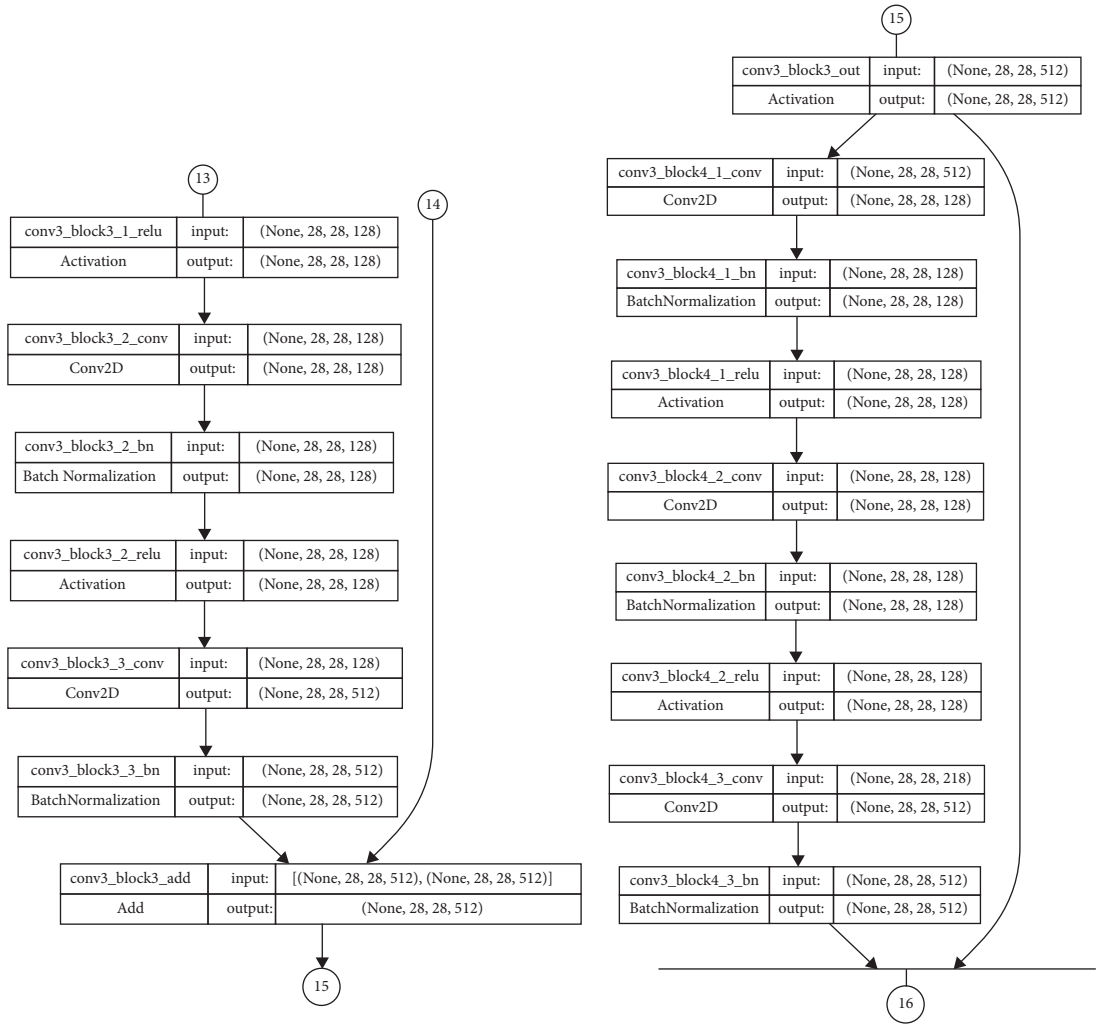

Figure 9: ResNet50 model (cont.).

challenges in optimization is finding an optimal set of parameters that minimizes a given loss function. AdaGrad tackles this challenge by adjusting the learning rates for each parameter based on historical information about how the gradients have been changing over time. At its core, AdaGrad's innovation lies in the way it adapts the learning rates for different parameters by taking into account the historical behavior of gradients. This adaptation is particularly important because traditional optimization methods often use a single learning rate for all parameters, which can lead to slow convergence or even divergence in complex optimization landscapes. Mathematically, let us break down the key steps of AdaGrad:

(1) *Initialization*. For i^{th} parameter w, AdaGrad initializes a parameter vector w_i and a variable G_i to zero. The parameter vector w_i represents the parameter to be optimized, and G_i keeps track of the historical sum of squared gradients.

(2) *Gradient Calculation*. At iteration t of the optimization process, the gradient g_t of the loss function with respect to the parameter vector w_i is computed.

Historical Gradient Accumulation. AdaGrad's adaptation is based on the accumulation of historical gradient information G_i which is updated as

$$G_i = G_{i-1} + \left(g_{t,i}\right)^2, \tag{10}$$

where $g_{t,i}$ represents the gradient of the loss function with respect to parameter w_i at iteration t.

Parameter Update. The parameter w_i is updated using the adaptive learning rate. Mathematically, w_i is expressed as

$$w_i = w_{i-1} + \frac{\eta}{\sqrt{G_i + \epsilon}} \cdot g_{t,i}, \tag{11}$$

where η represents the base learning rate, determining the step size of the update, and ϵ is a small positive constant added for numerical stability to avoid division by zero.

The critical insight of AdaGrad is that parameters with small historical gradients (indicating slow updates) will receive larger learning rate adjustments, while parameters with larger historical gradients (indicating fast updates) will experience smaller learning rate adjustments. This dynamic

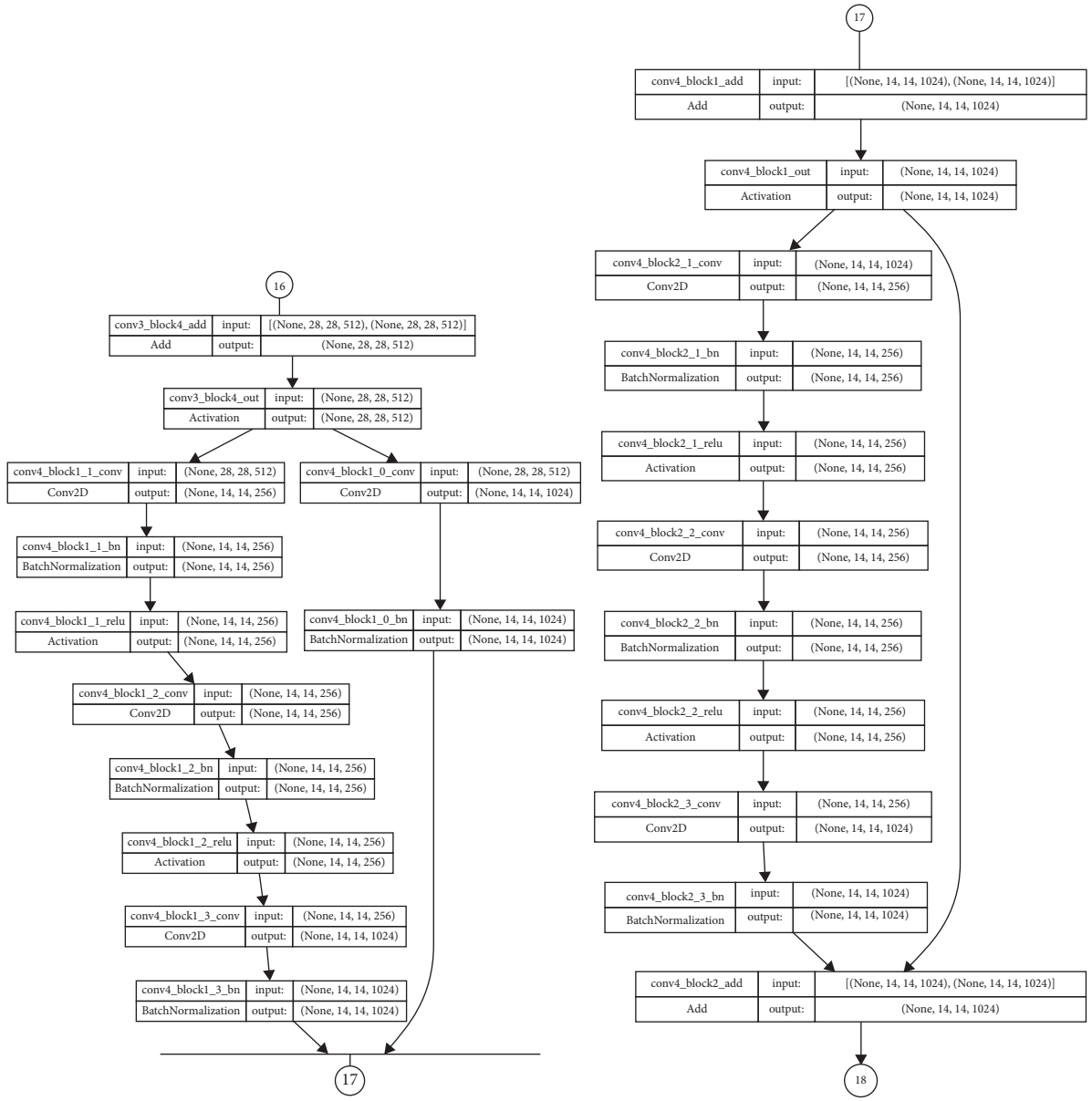

Figure 10: ResNet50 model (cont.).

adjustment enables the algorithm to navigate the optimization landscape more efficiently, converging faster in regions with complex dynamics. However, while AdaGrad's adaptivity can be advantageous, it also has its limitations. As training progresses, the historical gradient sum G_i continues to accumulate, potentially leading to overly small learning rates that hinder convergence. This phenomenon, often referred to as the "learning rate decay," has prompted the development of variants like RMSprop and Adam, which aim to strike a balance between adaptivity and convergence stability. Overall, AdaGrad is a pioneering optimization algorithm that tailors the learning rates of parameters based on historical gradient information. By adapting learning rates to the characteristics of each parameter, AdaGrad enhances the convergence efficiency, making it particularly effective in situations where data are sparse or features have varying scales. However, its tendency to decrease learning rates over time has motivated the creation of subsequent optimization algorithms that offer improved convergence behavior while retaining the adaptivity concept.

3.2.2. Adam. Adaptive Moment Estimation (Adam) [35, 52, 55, 56] is an optimization algorithm designed to enhance the training of ML models, particularly deep neural networks. It builds upon concepts from both the momentum and Root Mean Square Propagation (RMSprop), which is described later in Section 3.2.5, optimizers to offer efficient and adaptive updates to model parameters during the training process. Adam's innovation lies in its ability to adjust both the learning rates and momentum for each parameter based on historical gradient information, leading to improved convergence properties in various optimization landscapes. Mathematically, let us delve into the fundamental aspects of the Adam optimizer:

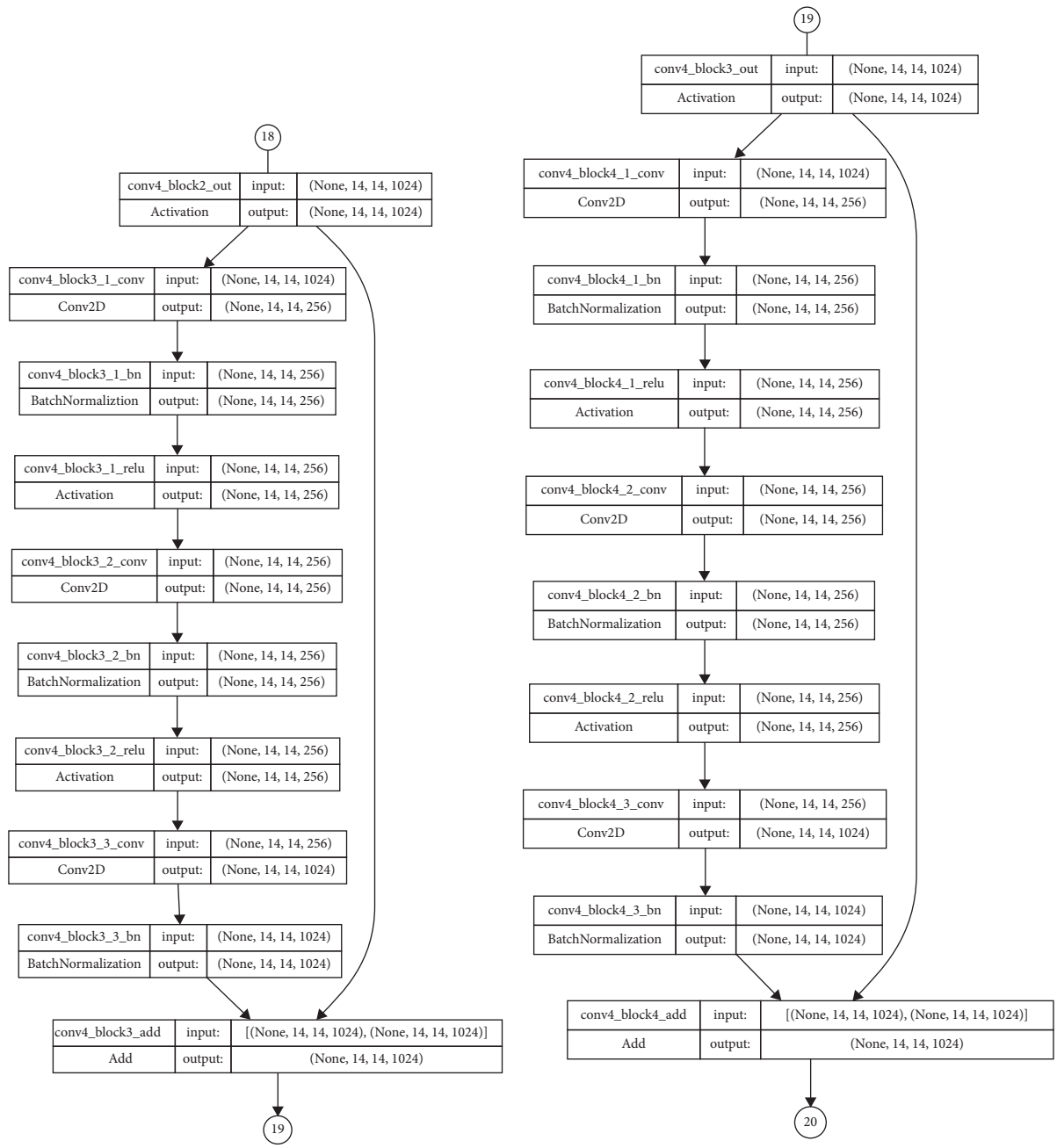

Figure 11: ResNet50 model (cont.).

(1) *Initialization*. Adam initializes two moment vectors, m and v, for each parameter. The m vector tracks the exponentially moving average of past gradients, and the v vector tracks the exponentially moving average of squared past gradients.

(2) *Exponential Moving Averages*. In each iteration of training, the gradients g_t of the loss function concerning the parameters are calculated. Adam then updates the m and v vectors using equations (12) and (13).

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . g_t, \tag{12}$$

$$v_t = \beta_2 . v_{t-1} + (1 - \beta_2) . g_t^2, \tag{13}$$

where β_1 and β_2 are hyperparameters that control the exponential decay rates of the moving averages.

(3) *Bias Correction*. The m and v vectors are biased toward zero, especially during the initial iterations. To address this bias, at iteration t, Adam performs bias

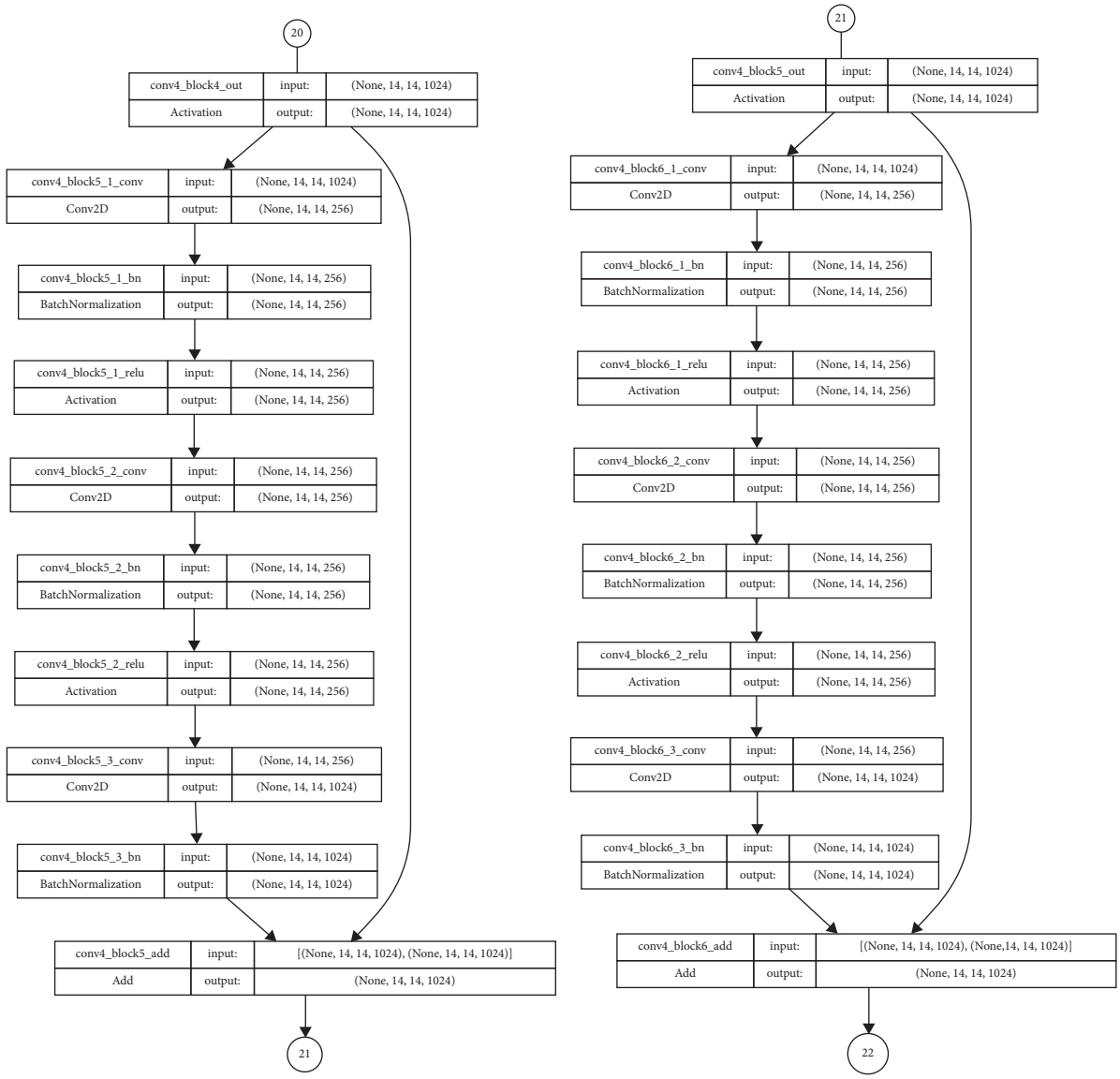

Figure 12: ResNet50 model (cont.).

correction by adjusting the vectors using equations (14) and (15):

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}, \tag{14}$$

$$\widehat{v_t} = \frac{v_t}{1-\beta_2^t}. \tag{15}$$

(4) *Parameter Update*. In the final step, the model's parameters w is updated utilizing the bias-corrected moment estimates as described in equation (16):

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{\widehat{v_t}+\epsilon}} \cdot \widehat{m_t}, \tag{16}$$

where η is the learning rate, controlling the step size of the update, and ϵ is a small constant added for numerical stability to prevent division by zero.

Adam's key strength lies in its adaptive learning rates and momentum adjustments. The m vector behaves like momentum, helping to accelerate the optimization process, while the v vector adapts the learning rates based on the historical information of the squared gradients. This combination allows Adam to handle sparse gradients and varying landscape geometries effectively. However, it is worth noting that while Adam offers excellent performance in many scenarios, its hyperparameters (β_1, β_2, and ϵ) need careful tuning. Incorrect tuning can result in suboptimal convergence or training instability. Furthermore, in cases where the loss landscape is particularly noisy or ill-conditioned, variations of Adam or other optimizers might be more suitable. Overall, Adam is an advanced optimization algorithm that combines momentum

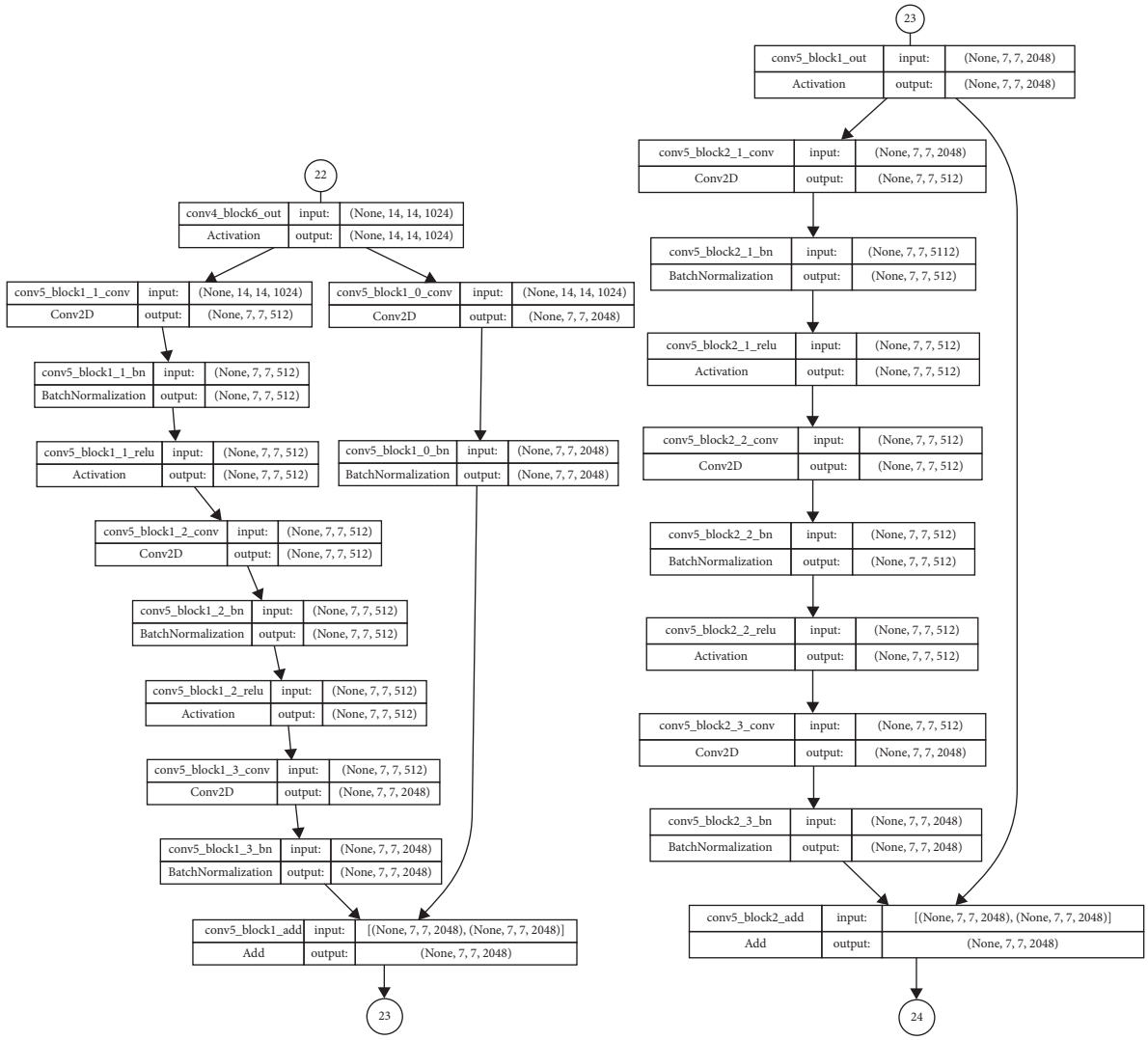

Figure 13: ResNet50 model (cont.).

and adaptive learning rates to optimize ML models effectively. Its mathematical framework enables it to adaptively adjust learning rates and momentum, making it well-suited for training deep neural networks and handling challenges like sparse gradients and varying scales. However, its success relies on hyperparameter tuning and understanding the characteristics of the optimization landscape.

3.2.3. AdaMax. Adaptive Moment Estimation with Infinity Norm (AdaMax) [34, 57–59] is an optimization algorithm that builds upon the principles of the Adam optimizer while addressing some of its limitations. Specifically, AdaMax is designed to offer more stable and effective updates by modifying the way the moments (exponential moving averages of gradients and squared gradients) are computed. By introducing a new term called "infinity norm" and modifying the denominator of the update equation, AdaMax aims to mitigate some of the potential issues that can arise during optimization. Let us explore the mathematical operations and key concepts of AdaMax:

(1) *Initialization*. Just like other adaptive optimization algorithms, AdaMax initializes two moment vectors, m and v, for each parameter. These moments track the exponentially weighted averages of past gradients and squared gradients, respectively. Additionally, an initial value for the infinity norm u is set.

(2) *Exponential Weighted Averages*. At iteration t, the gradients g_t of the loss function with respect to the parameters are calculated. AdaMax then updates the moments m and v using equations (17) and (18):

$$3.m_t = \beta_1.m_{t-1} + (1-\beta_1)\,.\,g_t, \tag{17}$$

$$v_t = \max\left(\beta_2.v_{t-1}, \left|g_t\right|\right), \tag{18}$$

where β_1 and β_2 are the exponential decay rates for the moving averages.

(3) *Parameter Update*. AdaMax performs the parameter update using the modified moments:

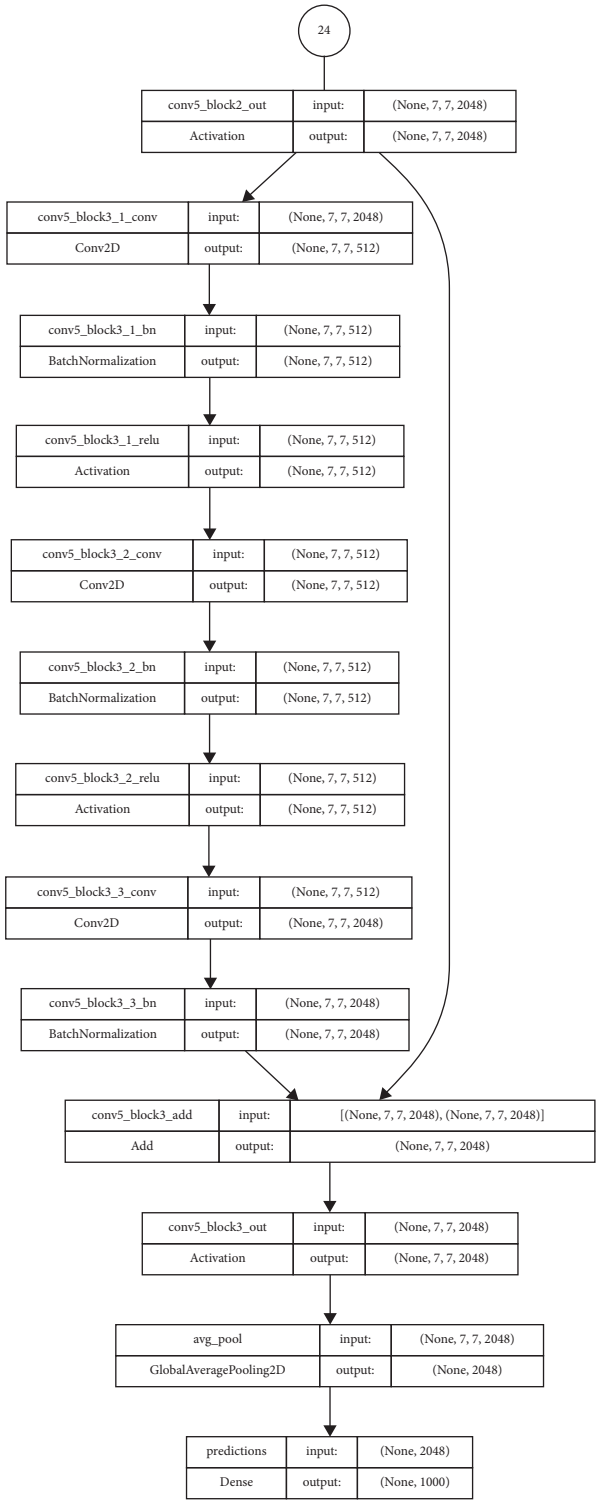

Figure 14: ResNet50 model (cont.).

$$w_t = w_{t-1} - \frac{\eta}{v_t} \cdot m_t, \tag{19}$$

where η is the learning rate, controlling the step size of the update.

The key innovation of AdaMax lies in the computation of the v moment. Instead of using the squared gradient directly, AdaMax employs the infinity norm $|g_t|$, which is the maximum absolute value of the gradient components. By doing so, AdaMax introduces a form of regularization on the second moment that can help address some of the convergence issues observed in Adam when the moving average of squared gradients gets close to zero. Moreover, AdaMax's use of the infinity norm can make the updates more robust, particularly in cases where individual gradient components are exceptionally large. This adaptation can prevent the v moment from becoming too small, thus maintaining stable learning rates throughout training. Overall, AdaMax is an extension of the Adam optimizer that employs the infinity norm and a modified denominator in the update equation to provide more reliable and effective parameter updates. Its mathematical alterations are designed to address some of the potential challenges in optimization landscapes, making it a valuable optimization technique for training ML models, especially deep neural networks. However, like other adaptive optimizers, proper hyperparameter tuning is essential to harness its full potential.

3.2.4. NAdam. Nesterov-accelerated Adaptive Moment Estimation (NAdam) [34, 60–62] is an optimization algorithm that builds upon the Adam optimizer by incorporating Nesterov's accelerated gradient technique. NAdam aims to combine the benefits of both the Nesterov Accelerated Gradient (NAG) and Adam optimizers to improve convergence and handling of noisy or ill-conditioned optimization landscapes. By adopting the momentum update in NAG with the moment-based update in Adam, NAdam seeks to achieve enhanced optimization performance. To understand NAdam more thoroughly, let us delve into its mathematical operations and key concepts:

(1) *Initialization*. Like other adaptive optimization algorithms, NAdam initializes two moment vectors, m and v, for each parameter. These moments track the exponentially weighted averages of past gradients and squared gradients, similar to Adam. Additionally, NAdam also initializes the Nesterov momentum term $|g_{t-1}|$ for each parameter.

(2) *Exponential Weighted Averages*. In each iteration of training, the gradients g_t of the loss function with respect to the parameters are calculated. NAdam then updates the moments m and v using similar formulas as Adam in equations (20) and (21).

(3) *Nesterov Momentum Update*. NAdam introduces the Nesterov momentum term g_{t-1} into the momentum update step. This term incorporates information from the previous iteration's gradient as expressed in the following equation:

$$g_{t-1} = \beta_1 \cdot g_{t-2} + (1 - \beta_1) \cdot g_{t-1}. \tag{20}$$

(4) *Combined Parameter Update*. The NAdam parameter update combines the momentum update from Nesterov's technique and the moment-based update from Adam. The parameter w is updated using the following equation:

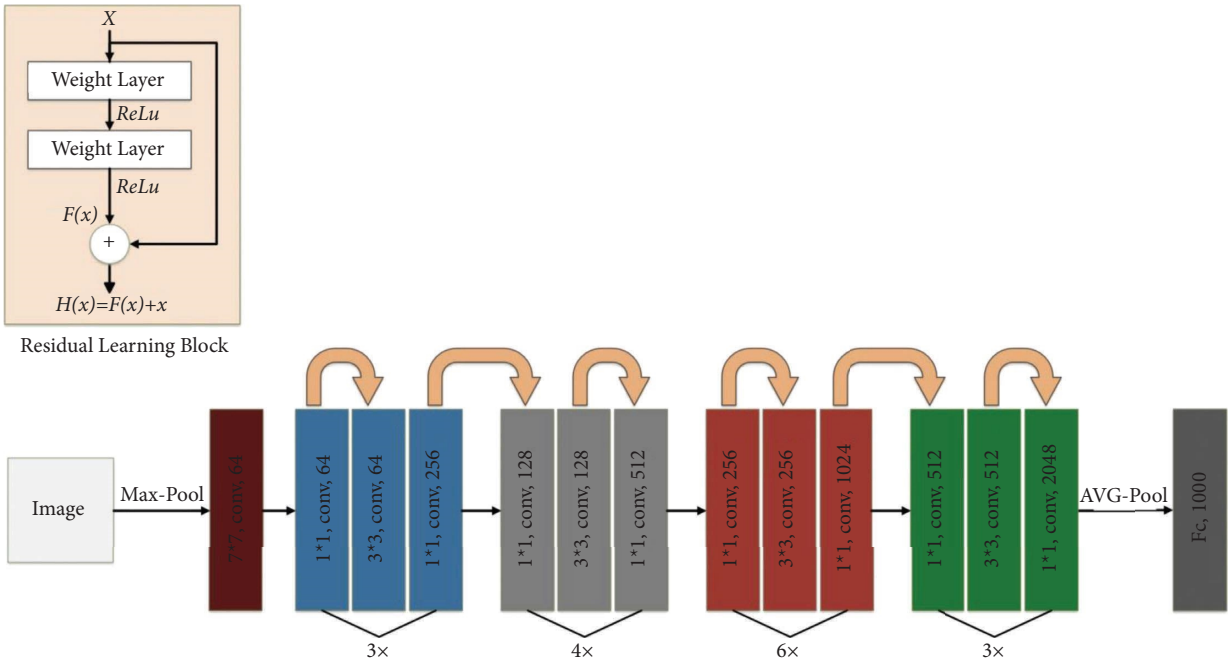

Figure 15: The architecture of ResNet50.

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{v_t} + \epsilon} \cdot (\beta_1 \cdot g_{t-1} + (1 - \beta_1) \cdot g_t). \quad (21)$$

The strength of NAdam lies in its combination of Nesterov's accelerated gradient and the adaptive moment updates. By incorporating the Nesterov term into the momentum update, NAdam improves convergence properties by adjusting the direction of the update based on past gradients. This can be particularly beneficial when navigating through regions with complex and noisy landscapes. Overall, NAdam is an optimization algorithm that marries the concepts of Nesterov's accelerated gradient and the adaptive moment-based updates of Adam. Its hybrid approach aims to provide improved optimization performance, making it a suitable candidate for training ML models, especially deep neural networks. As with other adaptive optimizers, proper tuning of hyperparameters is crucial for achieving optimal results.

3.2.5. RMSprop. Root Mean Square Propagation (RMSprop) [23, 34, 63–68] is an optimization algorithm designed to address some of the limitations of traditional gradient descent methods, particularly in handling varying scales of gradients and accelerating convergence in DL models. RMSprop adapts the learning rates of individual parameters based on the historical information of past gradients, allowing for more efficient updates and better optimization performance. Let us explore the mathematical operations and key concepts of RMSprop:

(1) *Initialization*. RMSprop initializes a variable s for each parameter, representing the exponentially moving average of squared gradients. This variable is initialized to zero.

(2) *Exponential Moving Averages*. In each iteration of training, the gradients g_t of the loss function with respect to the parameters are calculated. RMSprop then updates the s values using the following equation:

$$s_t = \beta . s_{t-1} + (1 - \beta) \cdot g_t^2, \quad (22)$$

where β is a hyperparameter that controls the exponential decay rate of the moving average.

(3) *Parameter Update*. RMSprop performs the parameter update using the adapted learning rate.

$$w_t = \text{wt} - 1 - \frac{\eta}{s_t} \cdot g_t. \quad (23)$$

The main innovation of RMSprop lies in its adaptation of the learning rate based on the moving average of squared gradients. By doing so, RMSprop can effectively handle situations where some gradients have larger magnitudes than others. This adaptive learning rate adjustment allows the optimizer to navigate optimization landscapes more smoothly and converge faster. Moreover, RMSprop's incorporation of the squared gradient magnitude in the denominator of the update equation contributes to a natural scaling of the learning rates, preventing the learning rates from becoming too small, as can happen in standard gradient descent methods. This property can make RMSprop particularly useful for training deep neural networks, where gradients can exhibit a wide range of magnitudes. Overall, RMSprop is an optimization algorithm that adapts the learning rates of parameters based on the historical information of squared gradients. Its mathematical framework enables it to handle varying scales of gradients and accelerate convergence in DL models. While RMSprop has been

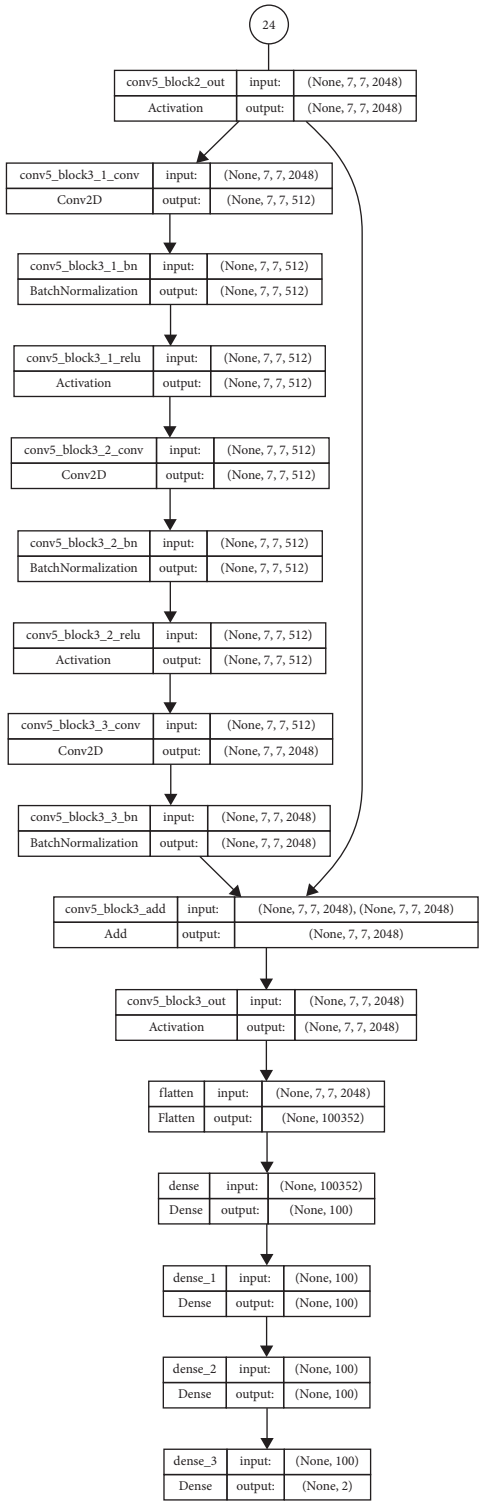

Figure 16: The modified ResNet50 model.

proven effective in many scenarios, it is important to note that proper hyperparameter tuning is crucial for achieving optimal results, as the choice of β and ϵ can significantly impact the optimization process.

4. Analytical Insights and Experimental Outcomes

The proposed models along with the recommended optimizers were subjected to experimentation, the datasets used for both training and testing were evaluated, and the final results were obtained by averaging all the assessment metrics. Additionally, the time taken for training and testing was also recorded. Further details on the datasets used to evaluate the performance of the proposed model can be found in Section 4.1. Workplace characteristics and hyperparameter values are recorded in Section 4.2, followed by performance indicators in Sections 4.3 and 4.4, which describe experimental analysis and result discussion.

4.1. Dataset Description. The evaluation of the proposed and state-of-the-art models is conducted in this study using data from a CXR provided by the NIH clinical center. This CXR can give an initial indication of cardiomegaly by evaluating the size and shape of the heart. NIH, a research hospital in the United States, has recently published over 100,000 anonymized frontal-view CXR images. The NIH gathered a scanning dataset of over 30,805 individuals, many of whom had advanced lung illnesses. All of the photographs have a high resolution of 1024×1024. Figure 18 provides visual dataset samples. There are up to 15 different thoracic categories, such as atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, pneumothorax, consolidation, edema, emphysema, fibrosis, pleural thickening, hernia, and normal images. Although the picture label is derived using natural language processing (NLP), the publisher guarantees that the NLP labeling precision surpasses 90%. The CXR8 also provides 8 class designations. The dataset is offered by Kaggle but with some changes to be more suitable for our problem of early detection of cardiomegaly and to make the data more balanced. The dataset is composed of 5552 images, divided into two categories: 2776 images are labeled as No Cardiomegaly, indicating normal conditions, and the other 2776 images are categorized under Cardiomegaly, signifying the presence of the disease, as illustrated in Figure 19. These images are preprocessed through resizing, color transformation, and normalization and then labeled. Image preprocessing is a crucial initial step in computer vision and image analysis pipelines, aiming to enhance the quality and consistency of images prior to any further analysis or processing.

4.1.1. Resizing [69–71]. The first step is to adjust the image's dimensions while keeping its aspect ratio intact. This is necessary to make all the images the same size, which ensures consistency and compatibility for further processing. Resizing images not only makes computing easier but also helps to compare images fairly across various resolutions, making analysis more accurate.

4.1.2. Color Transformation [72, 73]. In the second step, the color space of an image is manipulated to enhance certain image characteristics. For example, RGB (Red, Green, Blue)

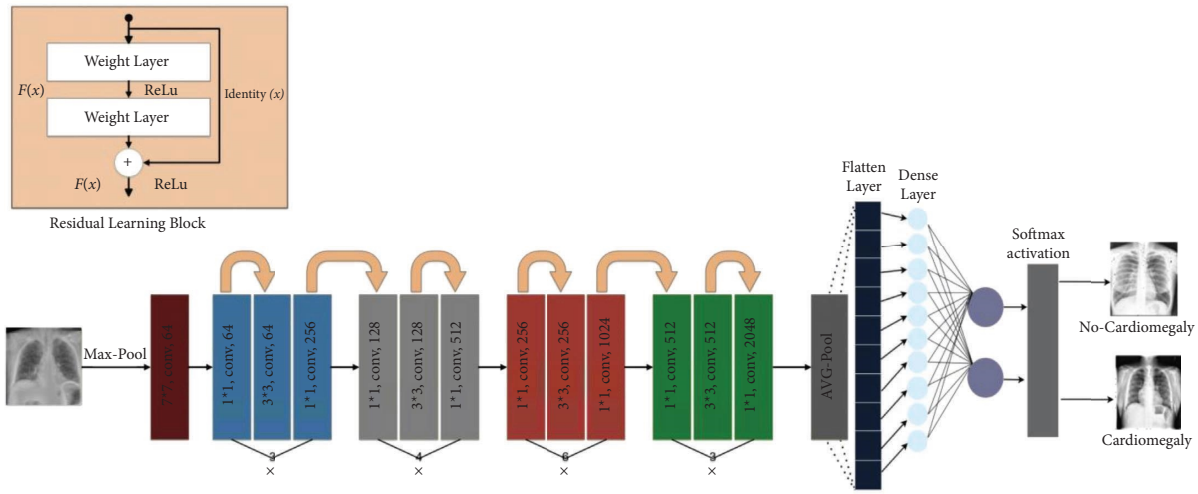

Figure 17: The architecture of modified ResNet50 for binary classification.

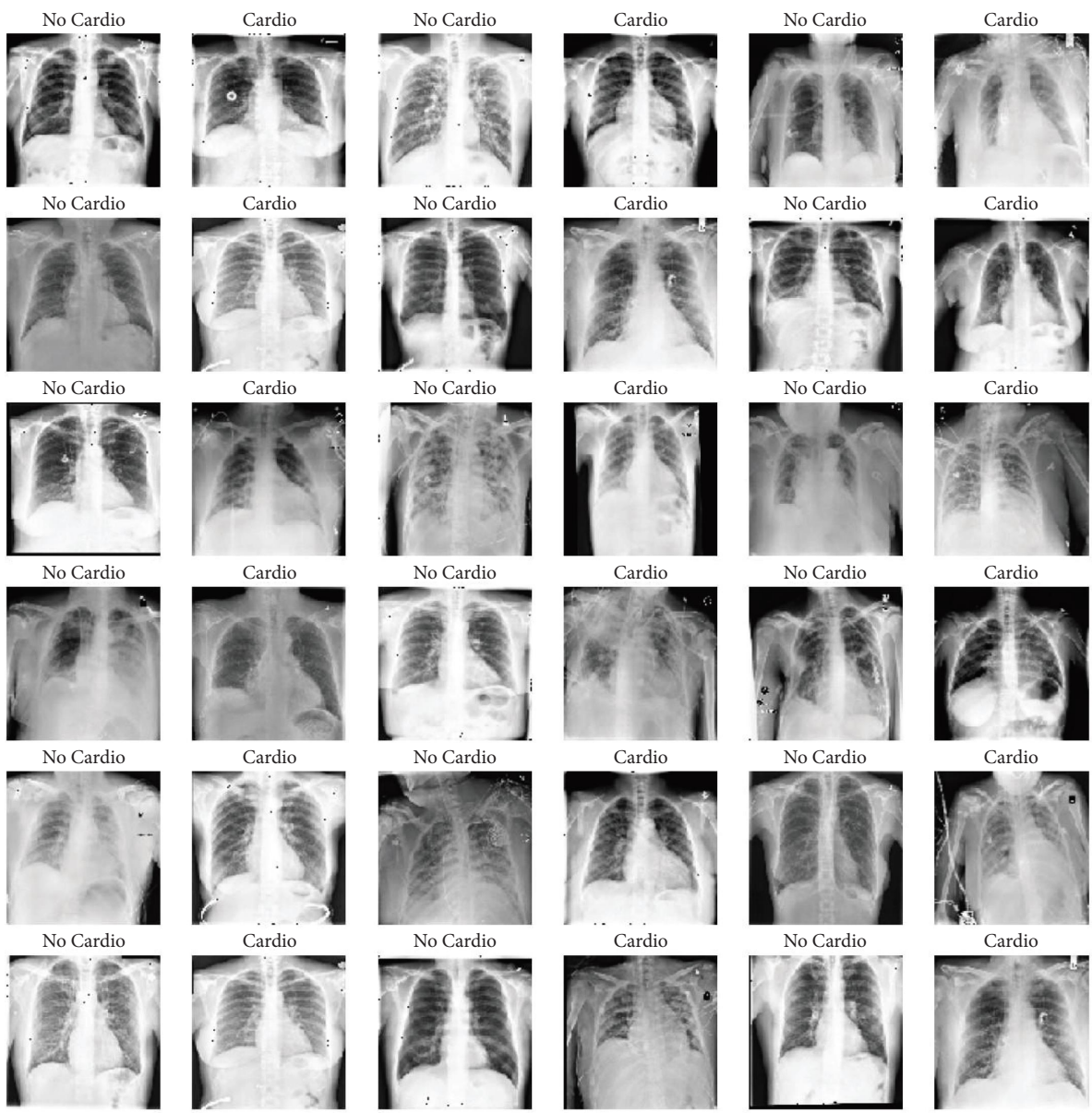

Figure 18: Visualization sample of the data.

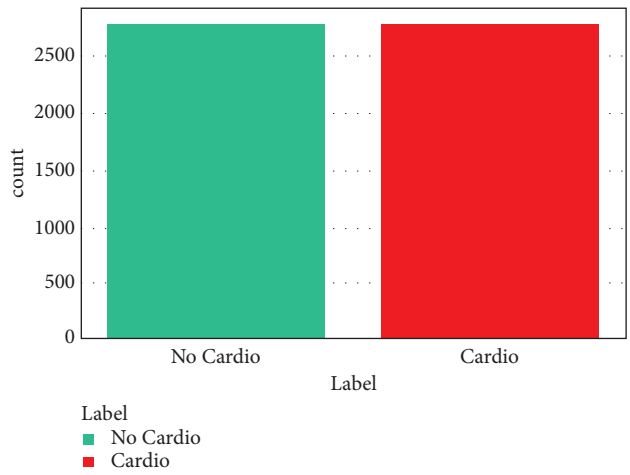

Figure 19: Dataset histogram.

and HSV (Hue, Saturation, Value) are distinct color spaces that represent color information within an image. Transforming an image from one color space to another can accentuate or suppress certain features, which can be useful in tasks such as feature extraction or pattern recognition.

4.1.3. Normalization [74, 75]. The third step involves adjusting pixel values to ensure they comply with a standardized scale. This is important to reduce the impact that changing lighting conditions or sensor characteristics can have on the appearance of an image. Normalization techniques typically involve scaling pixel values to fall within a specific range, such as [0, 1] or [−1, 1], or performing mean subtraction and division by standard deviation. Normalization ensures that the statistical distribution of image data is consistent, which helps to stabilize the training and learning processes of subsequent ML models.

4.1.4. Labeling [76, 77]. The final step in the process involves annotating images with relevant information that identifies the objects or regions within them. In supervised learning, labels serve as the ground truth for training and evaluating ML models. Proper labeling requires accurately associating images with the appropriate classes or categories, allowing algorithms to effectively learn and generalize from the provided data.

In this study, the data are labeled as 0 for no cardiomegaly and 1 for cardiomegaly. The data are then shuffled and split into 80% training and 20% testing sets, as shown in Table 2.

Next, the dataset is enhanced with data augmentation using the ImageDataGenerator technique [78–83]. Data augmentation is a crucial technique in computer vision, especially when there is a shortage of annotated data for training ML models. This technique generates various versions of the existing dataset by applying a range of transformations to the original images. In the Keras library, the ImageDataGenerator class enables this process by providing a simple way to implement data augmentation during the model's training phase.

Table 2: Dataset's subset size.

Subset	Records
Training	4441
Testing	1111

The ImageDataGenerator technique [80–83] operates by applying a set of predefined image transformations, such as rotations, flips, shifts, zooms, and shears, to the input images. These transformations introduce controlled perturbations that simulate real-world variations in imaging conditions. By augmenting the dataset with these transformed images, the effective size of the training data increases substantially, which can mitigate overfitting and enhance the model's generalization capabilities.

The advantages of data augmentation [84] are manifold. Firstly, augmented data enrich the training dataset with diverse instances, effectively expanding the model's exposure to a wider range of scenarios. This exposure contributes to improved robustness, as the model learns to recognize objects or patterns despite variations in lighting, orientation, and other factors. Secondly, augmented data can mitigate the risk of overfitting, a phenomenon where a model learns to memorize the learning data instead of generalizing from it. The inherent noise introduced by the transformations helps regularize the learning process, preventing the model from becoming excessively specialized to the training samples. Lastly, data augmentation can also save valuable time and resources, as it reduces the dependency on collecting and annotating an extensive dataset. Through augmentation, a relatively smaller dataset can be effectively amplified, potentially leading to comparable or even superior model performance.

4.2. Modelling Context. The computational outcomes were produced using a computer equipped with an Intel Core i7 CPU, 64 GB of RAM, and an NVIDIA GTX 1050i GPU. Additionally, coding resources like Python, Keras, TensorFlow, and Sklearn were employed for executing the programming assignments. The hyperparameters and standard parameter configurations for the presented models, including the maximum number of epochs and the loss function, are detailed in Table 3. These hyperparameters were set by using both grid search and random search techniques for hyperparameter tuning. Grid search exhaustively explores a predefined set of hyperparameter values, as the number of epochs was in the range from 1 to 100, the learning rate was in the range from 0.0001 to 0.1, and batch size was in the range from 16 to 120, while random search randomly samples from a given distribution. This combination helps find optimal configurations more efficiently.

4.3. Performance Metrics. Recall, precision, accuracy, and $F1$-score are some of the assessment measures used to assess the presented models. These metrics are dependent on the assessment parameters for predictive models known as True Positive (T_P), True Negative (T_N), False Positive (F_P), and

Table 3: Parameter setting for the proposed models.

Model	Epochs	Batch size	Total parameters	Trainable parameters	Nontrainable parameters	Learning rate	Loss function
Proposed CNN	50	85	3,010,114	3,010,114	0	0.001	binary_Crossentropy
Modified ResNet50	15	32	33,643,414	10,055,702	23,587,712	0.001	binary_Crossentropy

False Negative (F_N). Recall [85] is mathematically defined as T_P divided by the product of T_P and F_N, as shown in the following equation:

$$\text{Recall} = \frac{T_P}{T_P + F_N}. \tag{24}$$

Precision [86] is computed using equation (25) and is equal to T_P divided by the product of T_P and F_P:

$$\text{Precision} = \frac{T_P}{T_P + F_P}. \tag{25}$$

Equation (26) describes accuracy as follows:

$$\text{Accuracy} = \frac{T_N + T_P}{T_P + F_P + T_N + F_N}. \tag{26}$$

$F1$-score [87] is equal to 2 times precision times recall times the sum of precision and recall divided by the precision plus recall. It is written as

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{27}$$

4.4. Investigative Comparison. This paper suggests a framework to address the problem of early cardiomegaly detection. Two different DL models were used for this objective and each one is developed with five types of optimizers, which are AdaGrad, Adam, AdaMax, NAdam, and RMSprop, to find the best optimizer for each proposed model. The goal is to develop two DL models and find the best optimizer from the fifth for each one of them that performs at its best regarding processing speed and detection accuracy and then find the best-proposed model from the two.

4.5. Results of the Proposed CNN Models. In this section, the simulated outcomes of the suggested CNNs with the five types of optimizers are compared to find the best optimizer for the proposed CNN model. Table 3 shows that the suggested CNNs with the five types of optimizers are all trained using 50 epochs with a batch size of 85.

4.5.1. Results of the Proposed CNN with AdaGrad. The simulated outcomes of the proposed CNN-AdaGrad are covered in this section. The accuracy, loss curves, and confusion matrix of the proposed CNN-AdaGrad are shown in Figures 20–22, respectively. The proposed CNN-AdaGrad obtained an accuracy of 96.07% for training. Table 4 shows the proposed CNN-AdaGrad evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and $F1$-score of 95.5%, 96.89%, 73.1%, and 83.3%, respectively.

4.5.2. Results of the Proposed CNN with Adam. This section covers the simulated outcomes of the proposed CNN-Adam. The accuracy, loss curves, and confusion matrix of the proposed CNN-Adam are shown in Figures 23–25, respectively. The proposed CNN-Adam obtained an accuracy of 100.00% for training. Table 5 shows the proposed CNN-Adam evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and $F1$-score of 99.64%, 100.00%, 97.39%, and 98.7%, respectively.

4.5.3. Results of the Proposed CNN with AdaMax. This section covers the simulated outcomes of the proposed CNN-AdaMax. The accuracy, loss curves, and confusion matrix of the proposed CNN-AdaMax are shown in Figures 26–28, respectively. The proposed CNN-AdaMax obtained an accuracy of 100.00% for training. Table 6 shows the proposed CNN-AdaMax evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and $F1$-score of 99.91%, 100.00%, 99.40%, and 99.7%, respectively.

4.5.4. Results of the Proposed CNN with NAdam. The simulated outcomes of the proposed CNN-NAdam are covered in this section. The accuracy, loss curves, and confusion matrix of the proposed CNN-NAdam are shown in Figures 29–31, respectively. The proposed CNN-NAdam obtained an accuracy of 100.00% for training. Table 7 shows the proposed CNN-NAdam evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and $F1$-score of 99.55%, 100.00%, 97.1%, and 98.5%, respectively.

4.5.5. Results of the Proposed CNN with RMSprop. The simulated outcomes of the proposed CNN-RMSprop are covered in this section. The accuracy, loss curves, and confusion matrix of the proposed CNN-RMSprop are shown in Figures 32–34, respectively. The proposed CNN-RMSprop obtained an accuracy of 99.50% for training. Table 8 shows the proposed CNN-RMSprop evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and $F1$-score of 99.37%, 98.9%, 96.89%, and 97.8%, respectively.

4.6. Results of the Modified ResNet50 Models. In this section, the simulated outcomes of the modified ResNet50 with the five types of optimizers are compared to find the best optimizer for the modified ResNet50 model. Table 3 shows that the modified ResNet50 models with the five types of optimizers are all trained using 15 epochs with a batch size of 32.

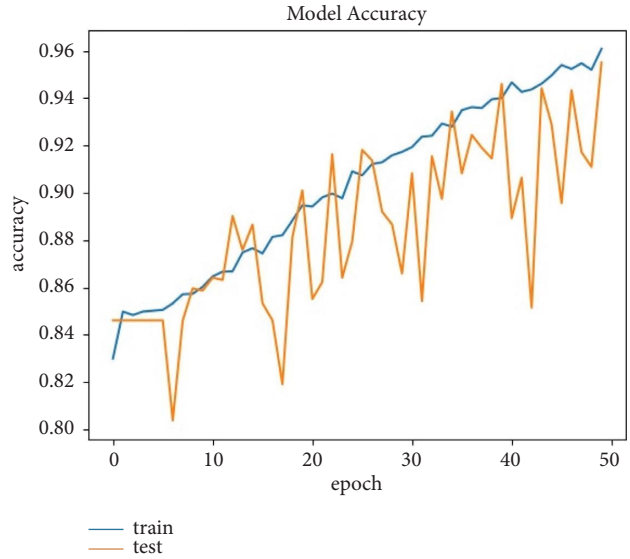

Figure 20: Accuracy curve of the proposed CNN-AdaGrad.

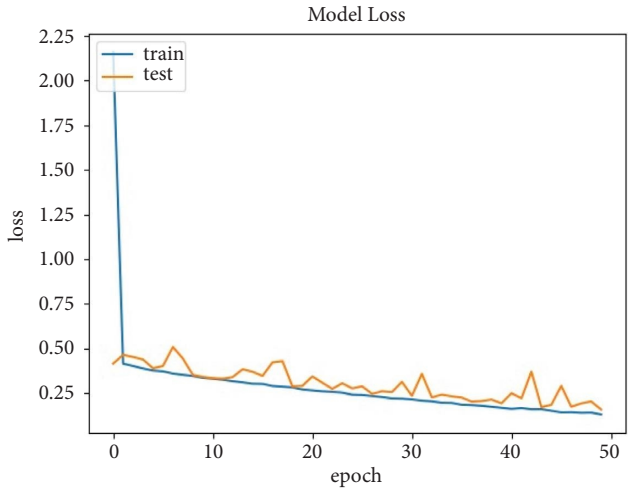

Figure 21: Loss curve of the proposed CNN-AdaGrad.

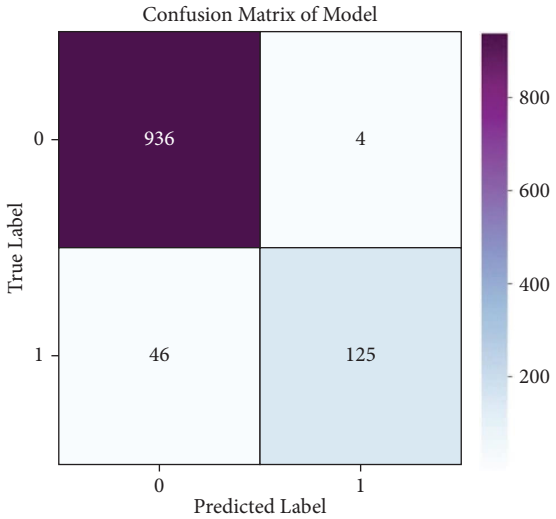

Figure 22: Confusion matrix of the proposed CNN-AdaGrad.

Table 4: Performance overview of the proposed CNN-AdaGrad.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.95	1.00	0.97
1 (cardiomegaly)	0.97	0.73	0.83
Macro avg.	0.96	0.86	0.90
Weighted avg.	0.96	0.95	0.95
Accuracy (%)	95.5		

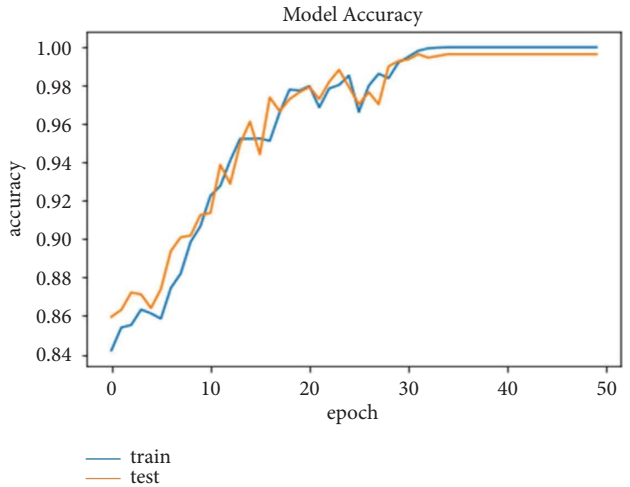

Figure 23: Accuracy curve of the proposed CNN-Adam.

4.6.1. Results of the Modified ResNet50 with AdaGrad. The simulated outcomes of the modified ResNet50-AdaGrad optimizer are presented here. The accuracy, loss curves, and confusion matrix of the modified ResNet50-AdaGrad are shown in Figures 35–37, respectively. The modified ResNet50-AdaGrad obtained an accuracy of 100.00% for training. Table 9 shows the modified ResNet50-AdaGrad evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and *F*1-score of 99.73%, 99.4%, 98.9%, and 99.1%, respectively.

4.6.2. Results of the Modified ResNet50 with Adam. The simulated outcomes of the modified ResNet50-Adam are covered in this section. The accuracy, loss curves, and confusion matrix of the modified ResNet50-Adam are shown in Figures 38–40, respectively. The modified ResNet50-Adam obtained an accuracy of 100.00% for training. Table 10 shows the modified ResNet50-Adam evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and *F*1-score of 99.19%, 100.00%, 95.1%, and 97.5%, respectively.

4.6.3. Results of Modified ResNet50 with AdaMax. The simulated outcomes of the modified ResNet50-AdaMax are covered in this section. The accuracy, loss curves, and confusion matrix of the modified ResNet50-AdaMax are shown in Figures 41–43, respectively. The modified ResNet50-AdaMax obtained an accuracy of 100.00% for training. Table 11 shows the modified ResNet50-AdaMax

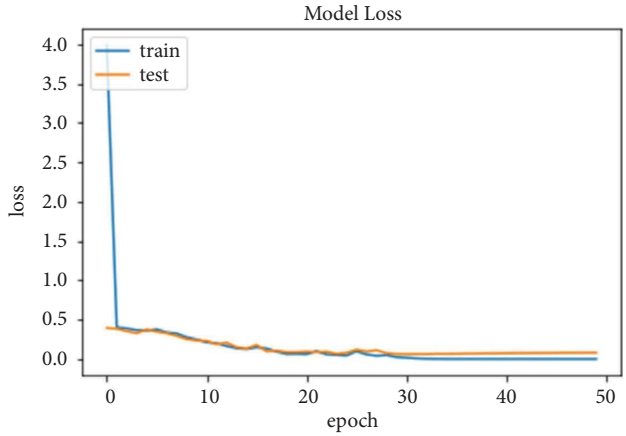

Figure 24: Loss curve of the proposed CNN-Adam.

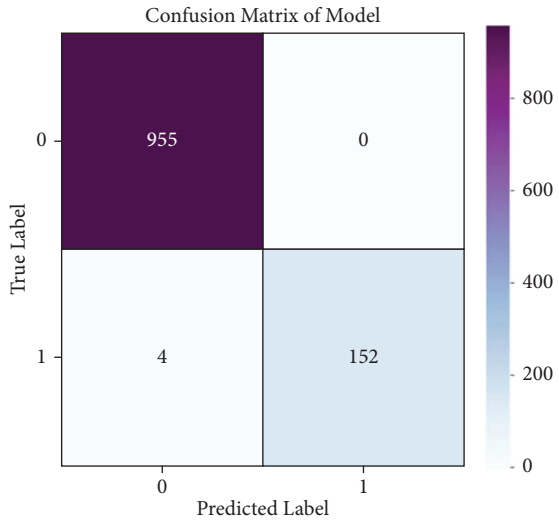

Figure 25: Confusion matrix of the proposed CNN-Adam.

Table 5: Performance overview of the proposed CNN-Adam.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	1.00	1.00	1.00
1 (cardiomegaly)	1.00	0.97	0.99
Macro avg.	1.00	0.99	0.99
Weighted avg.	1.00	1.00	1.00
Accuracy (%)	99.64		

evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and *F*1-score of 99.64%, 100.00%, 97.50%, and 98.70%, respectively.

4.6.4. Results of Modified ResNet50 with NAdam. The simulated outcomes of the modified ResNet50-NAdam optimizer are covered in this section. The accuracy, loss curves, and confusion matrix of the modified ResNet50-NAdam are shown in Figures 44–46, respectively. The modified ResNet50-NAdam obtained an accuracy of 100.00% for training. Table 12 shows the modified ResNet50-NAdam evaluation metrics in the testing stage. It

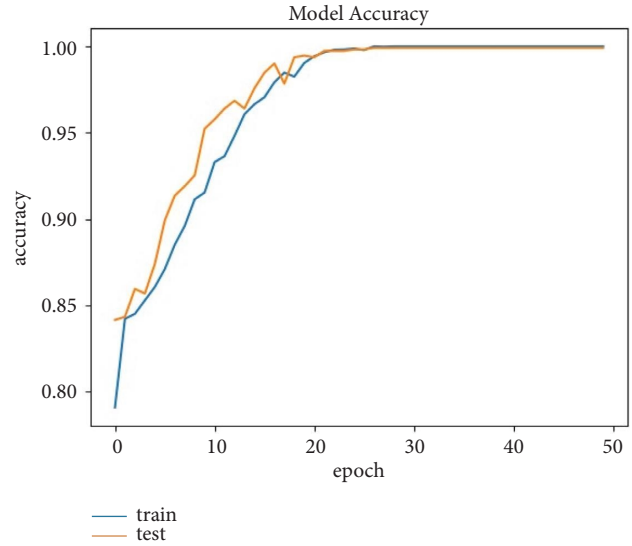

Figure 26: Accuracy curve of the proposed CNN-AdaMax.

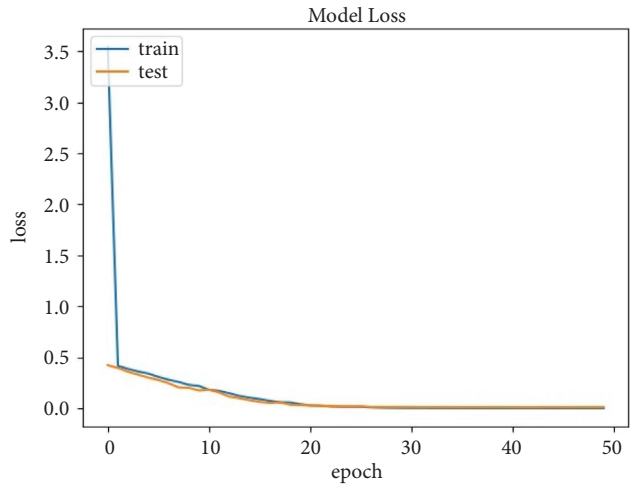

Figure 27: Loss curve of the proposed CNN-AdaMax.

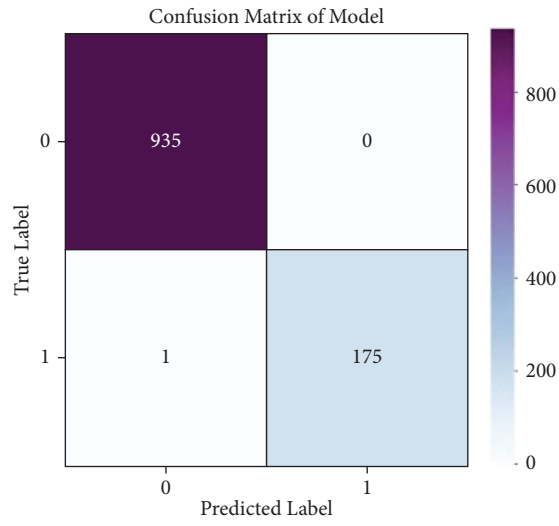

Figure 28: Confusion matrix of the proposed CNN-AdaMax.

Table 6: Overview of the proposed CNN-AdaMax.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	1.00	1.00	1.00
1 (cardiomegaly)	1.00	0.99	1.00
Macro avg.	1.00	1.00	1.00
Weighted avg.	1.00	1.00	1.00
Accuracy (%)	99.91		

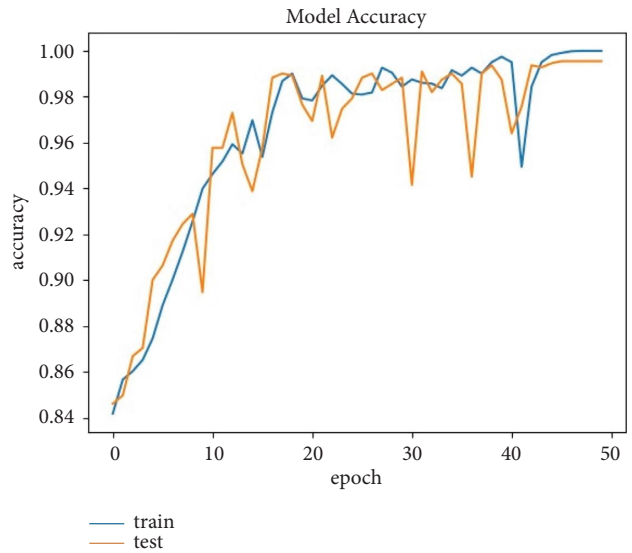

Figure 29: Accuracy curve of the proposed CNN-NAdam.

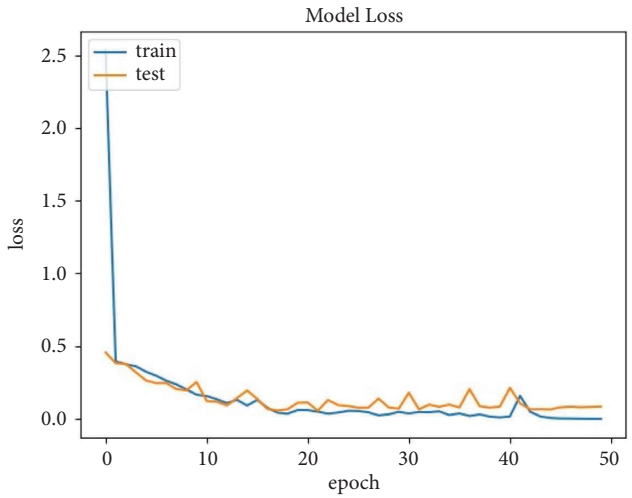

Figure 30: Loss curve of the proposed CNN-NAdam.

achieved an accuracy, precision, recall, and *F*1-score of 99.46%, 100.00%, 96.6%, and 98.3%, respectively.

4.6.5. Results of Modified ResNet50 with RMSprop. The simulated outcomes of the modified ResNet50-RMSprop optimizer are covered in this section. The accuracy, loss curves, and confusion matrix of the modified ResNet50-RMSprop are shown in Figures 47–49, respectively. The modified ResNet50-RMSprop obtained an accuracy of

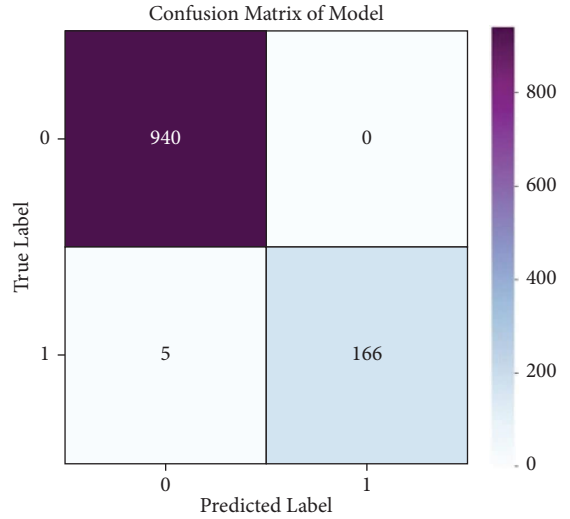

Figure 31: Confusion matrix of the proposed CNN-NAdam.

Table 7: Performance overview of the proposed CNN-NAdam.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.99	1.00	1.00
1 (cardiomegaly)	1.00	0.97	0.99
Macro avg.	1.00	0.99	0.99
Weighted avg.	1.00	1.00	1.00
Accuracy (%)	99.55		

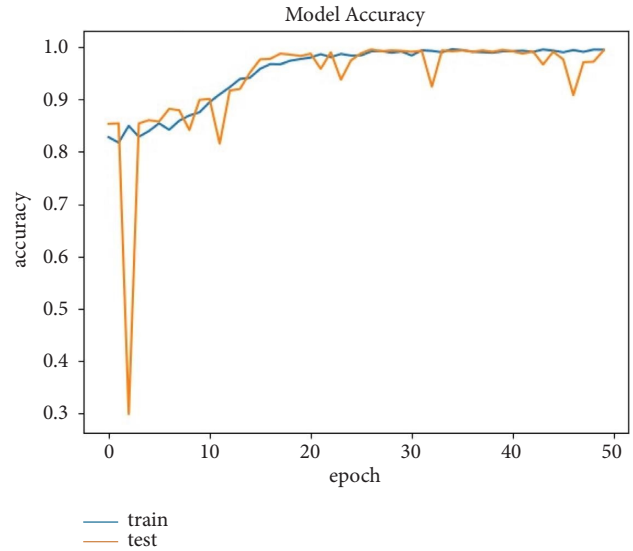

Figure 32: Accuracy curve of the proposed CNN-RMSprop.

98.96% for training. Table 13 shows the modified ResNet50-RMSprop evaluation metrics in the testing stage. It achieved an accuracy, precision, recall, and *F*1-score of 96.94%, 99.3%, 80.2%, and 88.7%, respectively.

4.6.6. Result Discussion. Table 14 shows the experimental results of the proposed CNN with the suggested optimizers; it can be observed that AdaGrad optimizer is the last in all

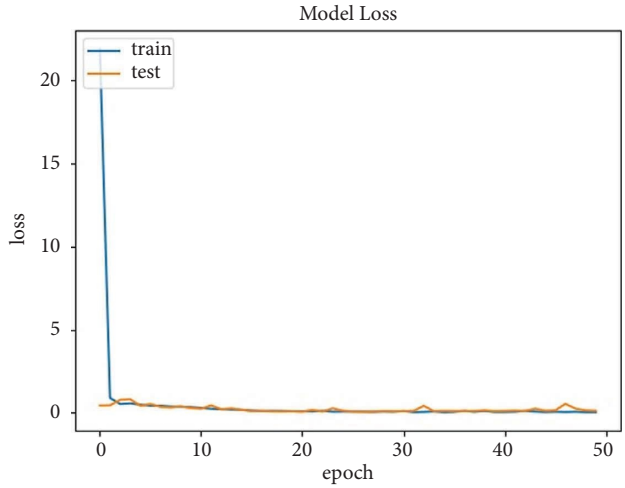

Figure 33: Loss curve of the proposed CNN-RMSprop.

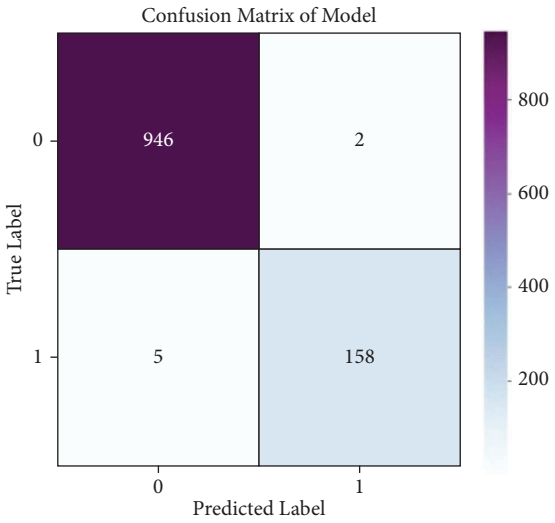

Figure 34: Confusion matrix of the proposed CNN-RMSprop.

Table 8: Performance overview of the proposed CNN-RMSprop.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.99	1.00	1.00
1 (cardiomegaly)	0.99	0.97	0.98
Macro avg.	0.99	0.98	0.99
Weighted avg.	0.99	0.99	0.99
Accuracy (%)	99.37		

performance measures. Adam optimizer ranks first in precision and second in accuracy, recall, *F*1-score, and loss. AdaMax optimizer is the best in all performance measures, and it is also the lowest in loss. NAdam optimizer ranks first in precision and third in accuracy, recall, *F*1-score, and loss. RMSprop optimizer ranks second in precision and fourth in accuracy, recall, *F*1-score, and loss.

Based on the duration required for the proposed CNN model, it is apparent that during the training phase, AdaGrad, Adam, and RMSprop optimizers required equal time

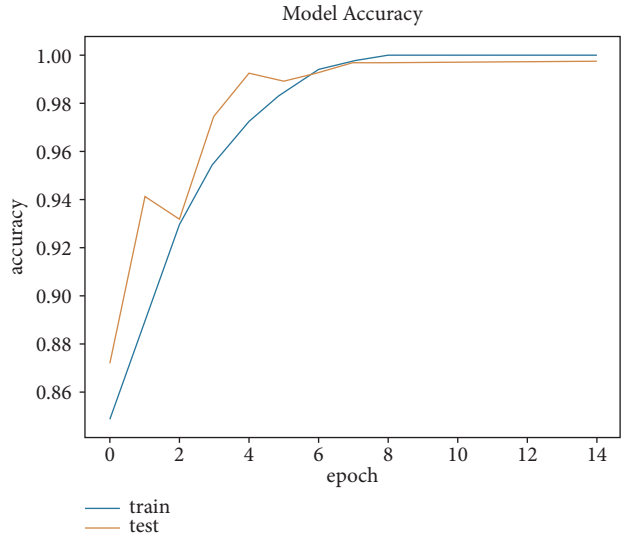

Figure 35: Accuracy curve of the modified ResNet50-AdaGrad.

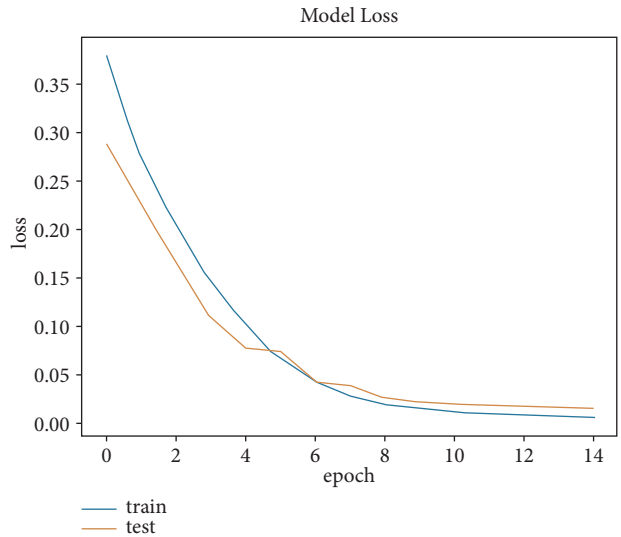

Figure 36: Loss curve of the modified ResNet50-AdaGrad.

and ranked second. AdaMax and NAdam, on the other hand, took the least amount of time and thus ranked first. Moving on to the testing phase, AdaMax proved to be the most efficient, taking the shortest time. NAdam came in second, while Adam and RMSprop took the same amount of time and ranked third. Finally, AdaGrad consumed the most time and ranked last.

Table 15 shows the experimental results of the modified ResNet50 with the suggested optimizers. It can be observed that AdaGrad optimizer ranks first in accuracy, recall, and *F*1-score and second in precision and loss. Adam optimizer ranks first in precision and fourth in accuracy, recall, *F*1-score, and loss. AdaMax optimizer ranks first in precision and loss and second in accuracy, recall, and *F*1-score. NAdam optimizer ranks first in precision and third in accuracy, recall, *F*1-score, and loss. RMSprop is the last in all performance measures.

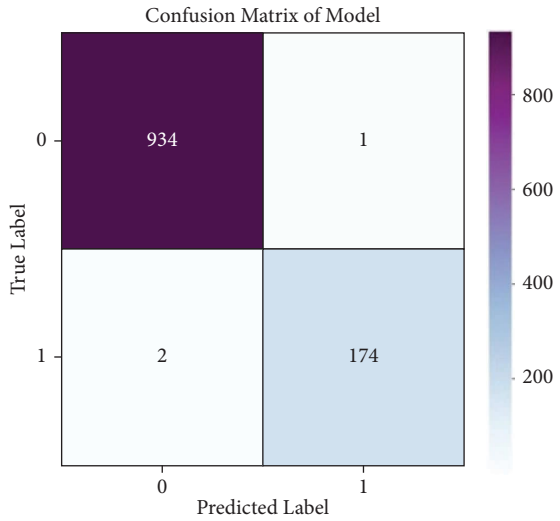

Figure 37: Confusion matrix of the modified ResNet50-AdaGrad.

Table 9: Performance overview of the modified ResNet50-AdaGrad.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	1.00	1.00	1.00
1 (cardiomegaly)	0.99	0.99	0.99
Macro avg.	1.00	0.99	0.99
Weighted avg.	1.00	1.00	1.00
Accuracy (%)	99.73		

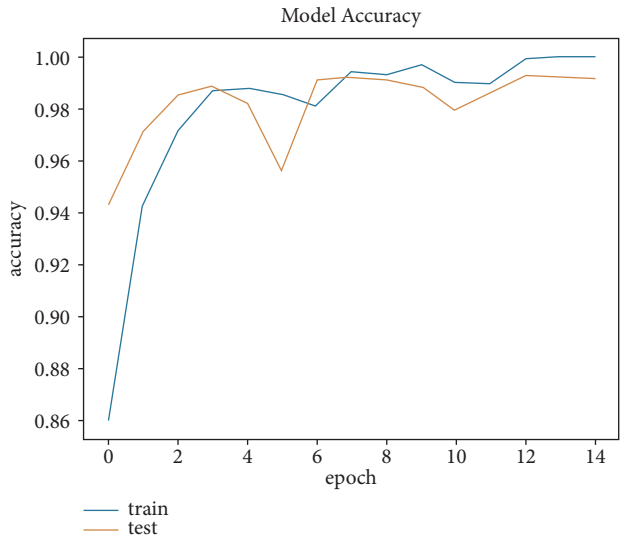

Figure 38: Accuracy curve of the modified ResNet50-Adam.

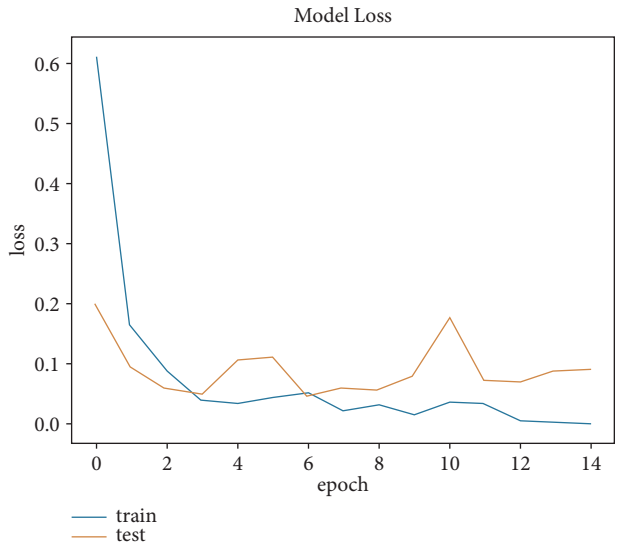

Figure 39: Loss curve of the modified ResNet50-Adam.

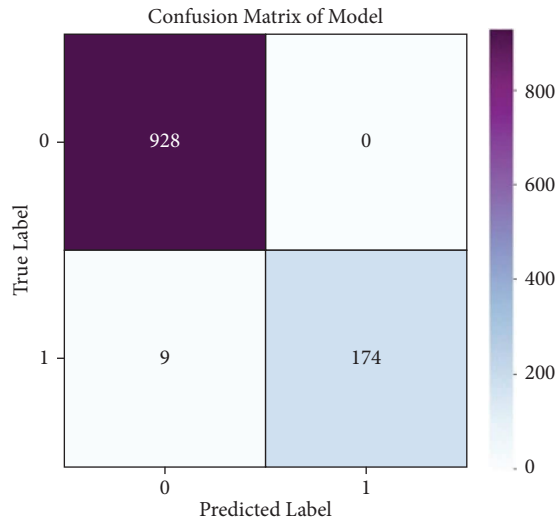

Figure 40: Confusion matrix of the modified ResNet50-Adam.

Table 10: Performance overview of the modified ResNet50-Adam.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.99	1.00	1.00
1 (cardiomegaly)	1.00	0.95	0.97
Macro avg.	1.00	0.98	0.98
Weighted avg.	0.99	0.99	0.99
Accuracy (%)	99.19		

It is evident from the duration needed for the adjusted ResNet50 model that Adam and RMSprop optimizers took an equal amount of time and were second in rank during the training phase. In contrast, AdaGrad, AdaMax, and NAdam took the least amount of time and ranked first. During the testing phase, AdaGrad demonstrated the highest efficiency, taking the shortest time. NAdam followed in second place, while AdaMax came in third. Adam and RMSprop took the same amount of time and ranked last.

Finally, it can be observed that the proposed CNN-AdaMax is the best in all performance measures, faster in both training and testing time, and lowest in loss.

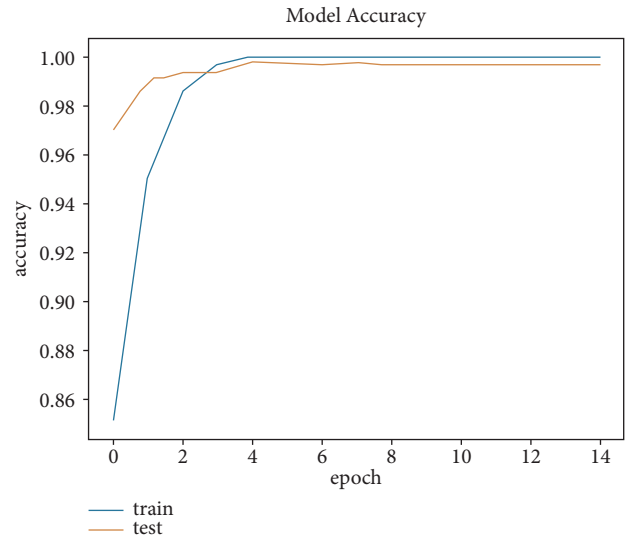

Figure 41: Accuracy curve of the modified ResNet50-AdaMax.

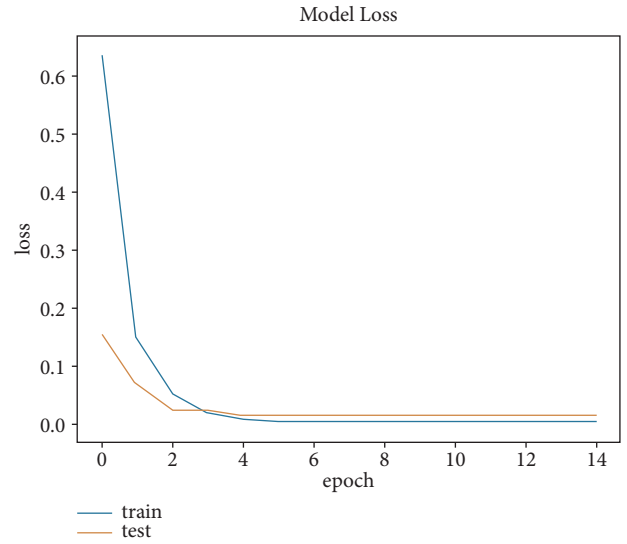

Figure 42: Loss curve of the modified ResNet50-AdaMax.

4.7. Benchmarking against Cutting-Edge Models. Table 16 compares the proposed models with the recent models using the same dataset. The proposed CNN-AdaMax optimizer ranked first in all performance metrics, while the proposed CNN-Adam optimizer, the proposed CNN-NAdam optimizer, the modified ResNet50-Adam optimizer, the modified ResNet50-AdaMax optimizer, and the modified ResNet50-NAdam optimizer ranked first in precision. The modified ResNet50-AdaGrad ranked second in terms of accuracy, precision, recall, and *F*1-score.

It is important to note that the choice of optimizer depends on various factors, including the architecture of the neural network, the nature of the dataset, and the training goals. In this study, the AdaMax optimizer with the proposed CNN, utilized dataset, and same hyperparameters is the best in all performance measures because as previously explained in Section 3.2.3, AdaMax is a variant of the Adam optimizer that provides certain advantages, particularly for training the proposed CNN:

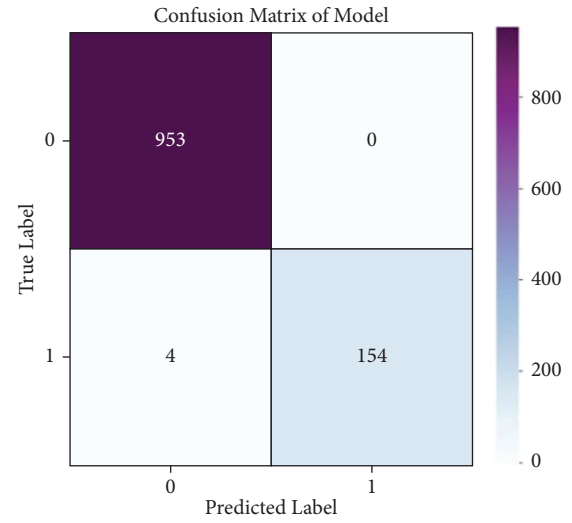

Figure 43: Confusion matrix of the modified ResNet50-AdaMax.

Table 11: Performance overview of the modified ResNet50-AdaMax.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	1.00	1.00	1.00
1 (cardiomegaly)	1.00	0.97	0.99
Macro avg.	1.00	0.99	0.99
Weighted avg.	1.00	1.00	1.00
Accuracy (%)	99.64		

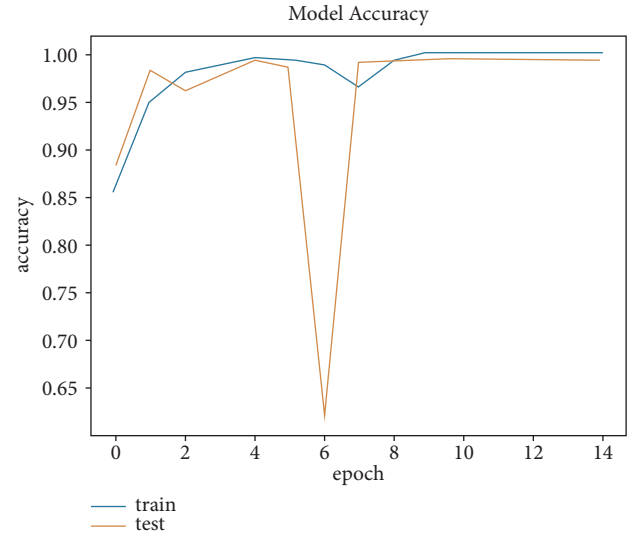

Figure 44: Accuracy curve of the modified ResNet50-NAdam.

(1) *Sparse Gradient Handling.* AdaMax's parameter update rule includes the infinity norm (L∞ norm) of the gradients, which can be helpful for handling sparse gradients. This property can be beneficial when training deep models with a large number of parameters, as sparse gradients can be an issue during optimization.

(2) *Convergence Speed.* AdaMax is known to have better convergence properties compared to plain Adam in

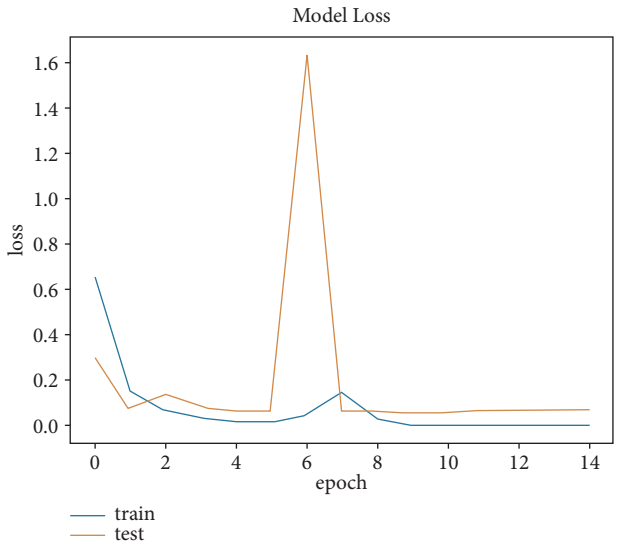

Figure 45: Loss curve of the modified ResNet50-NAdam.

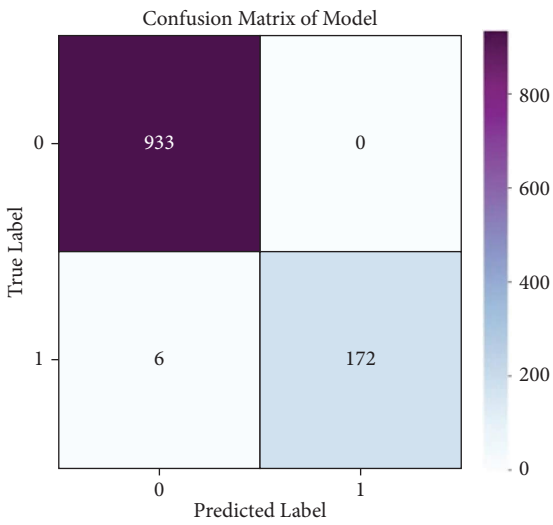

Figure 46: Confusion matrix of the modified ResNet50-NAdam.

Table 12: Performance overview of the modified ResNet50-NAdam.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.99	1.00	1.00
1 (cardiomegaly)	1.00	0.97	0.98
Macro avg.	1.00	0.98	0.99
Weighted avg.	0.99	0.99	0.99
Accuracy (%)	99.46		

certain cases. This can be beneficial when training a CNN from scratch, as faster convergence can lead to quicker training times.

(3) *Stability*. AdaMax's formulation can make it more stable than other optimizers, such as plain Adam. This stability can help avoid oscillations during training and improve the overall training process.

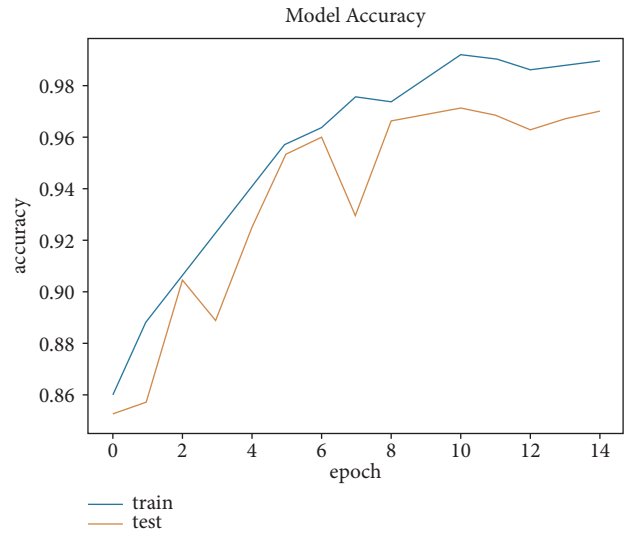

Figure 47: Accuracy curve of the modified ResNet50-RMSprop.

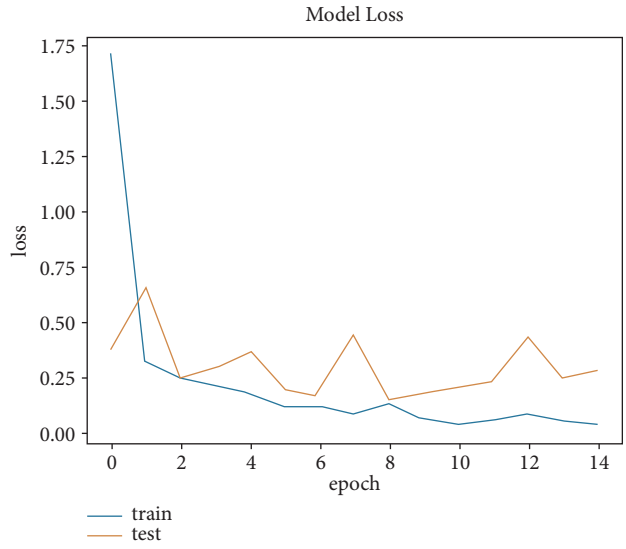

Figure 48: Loss curve of the modified ResNet50-RMSprop.

On the other hand, AdaGrad with the modified ResNet50 model in our problem case and with the same hyperparameters is the first in terms of accuracy, recall, and *F*1-score and also is faster in training and testing time because as previously explained in Section 3.2.1, AdaGrad is an optimizer that adapts the learning rates of individual parameters based on their historical gradient information. It has certain characteristics that could be advantageous when training complex architectures like ResNet50:

(1) *Adaptivity to Sparse Features*. AdaGrad's adaptive learning rate mechanism works well with sparse features. ResNet50 has skip connections that can introduce sparse-like activations through the network. AdaGrad's adaptivity to such features can lead to more effective updates during training.

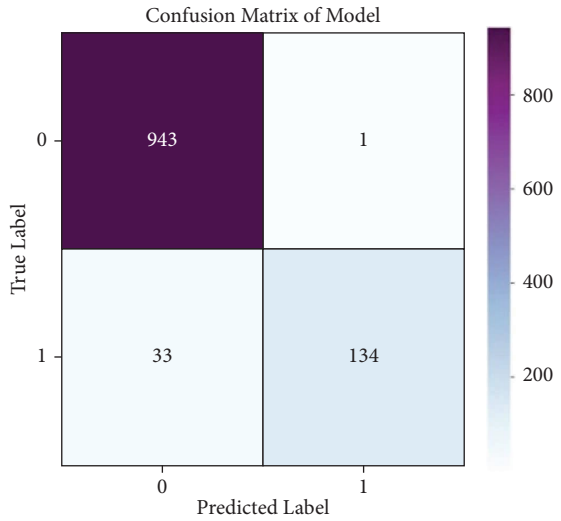

Figure 49: Confusion matrix of the modified ResNet50-RMSprop.

Table 13: Performance overview of the modified ResNet50-RMSprop.

	Precision	Recall	*F*1-score
0 (no cardiomegaly)	0.97	1.00	0.98
1 (cardiomegaly)	0.99	0.80	0.89
Macro avg.	0.98	0.90	0.93
Weighted avg.	0.97	0.97	0.97
Accuracy (%)	96.94		

Table 14: Comparison between the five optimizers of the proposed CNN model.

Model	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Loss	Training time (m)	Testing time (ms)
CNN-AdaGrad	95.50	96.89	73.10	83.30	0.158	6	14
CNN-Adam	99.64	100.00	97.39	98.7	0.080	6	13
CNN-AdaMax	99.91	100.00	99.40	99.7	0.001	5	10
CNN-NAdam	99.55	100.00	97.1	98.5	0.083	5	12
CNN-RMSprop	99.37	98.9	96.89	97.8	0.105	6	13

Table 15: Comparison between the five optimizers of the modified ResNet50 model.

Model	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)	Loss	Training time (m)	Testing time (ms)
ResNet50-AdaGrad	99.73	99.4	98.9	99.1	0.0158	8	87
ResNet50-Adam	99.19	100.00	95.1	97.5	0.087	9	98
ResNet50-AdaMax	99.64	100.00	97.5	98.7	0.009	8	96
ResNet50-NAdam	99.46	100.00	96.6	98.3	0.062	8	93
ResNet50-RMSprop	96.94	99.3	80.2	88.7	0.274	9	98

(2) *Weight Decay Adaptation*. AdaGrad's accumulation of historical gradients effectively reduces the learning rate for frequently updated parameters. This can act as a form of implicit weight decay, which can help prevent overfitting in complex models like ResNet50.

(3) *Convergence with Sparse Data*. If the dataset used for training ResNet50 contains sparse or irregularly distributed features, AdaGrad's learning rate adaptation can aid in navigating the optimization landscape and achieving convergence.

Table 16: Comparison between the proposed models and the state-of-the-art methods.

Method	Accuracy (%)	Precision (%)	Recall (%)	*F*1-score (%)
1D CNN (Lin et al. [9], 2022)	98.00	97.80	98.20	97.99
Mix of 1D and 2D CNN (Wu et al. [10], 2022)	98.40	97.60	99.20	98.38
CXRDANet (Chen et al. [11], 2022)	90.50	—	94.45	90.59
A mix of InceptionV3 and ResNet50 (Zhou et al. [12], 2019)	79.70	—	—	80.00
Cardio-XAttentionNet (Innat et al. [13], 2023)	87.00		85.00	86.00
Attention U-Net (Ajmera et al. [14], 2022)	—	99.00	80.00	88.00
U-Net (Sarpotdar et al. [15], 2022).	94.00	—	96.20	—
Novel automated method (Candemir et al. [88], 2016)	77.00	—	77.00	—
VGG19 (Bougias et al. [16], 2021)	84.50		84.00	
ResNet50 v2 (Ribeiro et al. [17], 2023)	91.80 ± 0.7	74.00 ± 2.7	87.00 ± 5.5	79.80 ± 1.9
Proposed CNN-AdaGrad	95.50	96.89	73.10	83.30
Proposed CNN-Adam	99.64	100.00	97.39	98.70
Proposed CNN-AdaMax	99.91	100.00	99.40	99.70
Proposed CNN-NAdam	99.55	100.00	97.10	98.50
Proposed CNN-RMSprop	99.37	98.90	96.89	97.80
Modified ResNet50-AdaGrad	99.73	99.40	98.90	99.10
Modified ResNet50-Adam	99.19	100.00	95.10	97.50
Modified ResNet50-AdaMax	99.64	100.00	97.50	98.70
Modified ResNet50-NAdam	99.46	100.00	96.60	98.30
Modified ResNet50-RMSprop	96.94	99.30	80.20	88.70

Overall, it is important to note that while AdaMax and AdaGrad or other optimizers have these potential advantages, their effectiveness can still depend on factors such as the specific dataset, hyperparameter tuning, and other architecture-related choices. So, in order to find out which optimizer is most effective for a particular task, experimentation is crucial.

5. Conclusion

Early detection of cardiomegaly is crucial for managing cardiovascular health. This research highlights the importance of timely identifying the condition to enable proactive interventions, ultimately leading to better patient outcomes. Advanced imaging techniques, such as CXR imaging, are noninvasive and effective for assessing cardiac morphology and size. They enable clinicians to determine cardiac chamber enlargement and overall cardiac dimensions accurately. So, in our paper, one of the finest datasets for the detection process is CXR. DL-based models are used to categorize images in the medical area. This paper introduced two DL models: one from scratch which is the proposed CNN and a pretrained model ResNet50 with some modifications, to be suitable for our binary classification problem. Each model is compiled with five types of optimizers which are AdaGrad, Adam, AdaMax, NAdam, and RMSprop. For the proposed CNN which is trained using 50 epochs with a batch size of 85, AdaMax optimizer is the best in all performance measures and it is also the fastest of them. For modified ResNet50 which is trained using 15 epochs with a batch size of 32, the AdaGrad optimizer is better in terms of accuracy, recall, and F1-score and also is the fastest of them, but Adam, AdaMax, and NAdam optimizers are the better in precision. Overall, the proposed CNN-AdaMax is the best in all performance measures and the fastest. The results of this study confirm the diagnostic power of such imaging modalities in detecting cardiomegaly. Therefore, they should be considered for integration into routine clinical practice for early detection and intervention. Early detection of cardiomegaly not only advances scientific understanding but also reduces the burden of advanced cardiac pathology on individuals and healthcare systems. Further research should investigate the long-term implications of early detection and explore the interplay of various imaging technologies. This will contribute to the refinement of clinical strategies and the enhancement of patient care in cardiovascular health. Although the study achieved impressive accuracies of 99.91% using AdaMax for the proposed CNN and 99.73% using AdaGrad for the modified ResNet50, there are still some limitations. The dataset used for this study includes 5552 images, with equal representation of no-cardio and cardio cases, but there may be biases that could affect the generalizability of the results. Additionally, the complex architecture of ResNet50 makes it difficult to interpret the model's results, which is crucial in medical applications. The high resource requirements for training ResNet50 also raise concerns about practical deployment. To overcome these limitations, future research will focus on exploring the dataset in detail to identify and address any biases, working collaboratively with domain experts to improve interpretability, and investigating optimization strategies such as model compression and hardware acceleration to enable efficient deployment in resource-constrained environments.

Abbreviations

DL: Deep learning
ResNet50: Residual network with 50 layers
NIH: National Institutes of Health
AdaGrad: Adaptive Gradient
NAdam: Nesterov-accelerated Adaptive Moment Estimation

RMSprop:	Root Mean Square Propagation
AdaMax:	Adaptive Moment Estimation with Infinity Norm
VGG19:	Visual Geometry Group with 19 layers
AI:	Artificial intelligence
CTR:	Cardiothoracic ratio
CXRDANet:	Chest X-ray with dual attention network
PA:	Posterior-anterior
AUC:	Area under the curve
GAP:	Global average pooling
SGD:	Stochastic gradient descent
CNN-Adam:	CNN with Adam optimizer
CNN-NAdam:	CNN with NAdam optimizer
ResNet50-AdaGrad:	ResNet50 with AdaGrad optimizer
ResNet50-AdaMax:	ResNet50 with AdaMax optimizer
ResNet50-RMSprop:	ResNet50 with RMSprop optimizer
ML:	Machine learning
CNNs:	Convolutional neural networks
VGG16:	Visual Geometry Group with 16 layers
Adam:	Adaptive Moment Estimation
TN:	True Negative
FP:	False Positive
FN:	False Negative
TP:	True Positive
NLP:	Natural language processing
CXR:	Chest X-ray
CAM:	Channel attention module
SAM:	Spatial attention module
AMM:	Attention mapping mechanism
JSRT:	Japanese Society of Radiological Technology
CNN-AdaGrad:	CNN with AdaGrad optimizer
CNN-AdaMax:	CNN with AdaMax optimizer
CNN-RMSprop:	CNN with RMSprop optimizer
ResNet50-Adam:	ResNet50 with Adam optimizer
ResNet50-NAdam:	ResNet50 with NAdam optimizer.

Data Availability

Data are available on request from the corresponding author.

Disclosure

Intellectual property protection measures have been thoroughly discussed, and there are no impediments to its dissemination, including publication timing. Accordingly, we confirm our adherence to our institutions' guidelines on intellectual property.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

All authors have reviewed and approved the manuscript, and no supplementary contributors who meet authorship criteria have been excluded. The sequence of authors has been agreed upon mutually.

Acknowledgments

The authors extend their sincere thanks to the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia, for their generous financial support, which was crucial for the research project (GRANT 5,740).

References

[1] A. A. Novikov, D. Lenis, D. Major, J. Hladuvka, M. Wimmer, and K. Buhler, "Fully convolutional architectures for multiclass segmentation in chest radiographs," *IEEE Transactions on Medical Imaging*, vol. 37, no. 8, pp. 1865–1876, 2018.

[2] S. Candemir, S. Rajaraman, G. Thoma, and S. Antani, "Deep learning for grading cardiomegaly severity in chest x-rays: an investigation," in *2018 IEEE Life Sciences Conference (LSC)*, pp. 109–113, IEEE, Montreal, Canada, October 2018.

[3] Q. Que, Z. Tang, R. Wang et al., "CardioXNet: automated detection for cardiomegaly based on deep learning," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 612–615, IEEE, Honolulu, HI, USA, July 2018.

[4] T. Gupte, M. Niljikar, M. Gawali, V. Kulkarni, A. Kharat, and A. Pant, "Deep learning models for calculation of cardiothoracic ratio from chest radiographs for assisted diagnosis of cardiomegaly," in *2021 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*, pp. 1–6, IEEE, Durban, South Africa, August 2021.

[5] P. Xu, F. Roosta, and M. W. Mahoney, "Second-order optimization for non-convex machine learning: an empirical study," in *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 199–207, SIAM, Philadelphia, PA, USA, July 2020.

[6] Z. C. Taçyildiz, E. Kiliç, A. Budak, and H. Karataş, "Cardiothoracic ratio calculation and cardiomegaly detection based on object detection," in *2021 29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, Istanbul, Turkey, June 2021.

[7] K. Dimopoulos, G. Giannakoulas, I. Bendayan et al., "Cardiothoracic ratio from postero-anterior chest radiographs: a simple, reproducible and independent marker of disease severity and outcome in adults with congenital heart disease," *International Journal of Cardiology*, vol. 166, no. 2, pp. 453–457, 2013.

[8] S. E. Sorour, A. A. A. El-Mageed, K. M. Albarrak, A. K. Alnaim, A. A. Wafa, and E. El-Shafeiy, "Classification of Alzheimer's disease using MRI data based on Deep Learning Techniques," *Journal of King Saud University-Computer and Information Sciences*, vol. 36, no. 2, Article ID 101940, 2024.

[9] C.-H. Lin, F. Z. Zhang, J. X. Wu et al., "Posteroanterior chest X-ray image classification with a multilayer 1D convolutional neural network-based classifier for cardiomegaly level screening," *Electronics*, vol. 11, no. 9, p. 1364, 2022.

[10] J.-X. Wu, C.-C. Pai, C.-D. Kan, P.-Y. Chen, W.-L. Chen, and C.-H. Lin, "Chest X-ray image analysis with combining 2D and 1D convolutional neural network based classifier for rapid cardiomegaly screening," *IEEE Access*, vol. 10, pp. 47824–47836, 2022.

[11] L. Chen, T. Mao, and Q. Zhang, "Identifying cardiomegaly in chest x-rays using dual attention network," *Applied Intelligence*, vol. 52, no. 10, pp. 11058–11067, 2022.

[12] S. Zhou, X. Zhang, and R. Zhang, "Identifying cardiomegaly in ChestX-ray8 using transfer learning," in *MEDINFO 2019: Health and Wellbeing E-Networks for All*, pp. 482–486, IOS Press, Amsterdam, Netherlands, 2019.

[13] M. Innat, M. F. Hossain, K. Mader, and A. Z. Kouzani, "A convolutional attention mapping deep neural network for classification and localization of cardiomegaly on chest X-rays," *Scientific Reports*, vol. 13, no. 1, p. 6247, 2023.

[14] P. Ajmera, A. Kharat, T. Gupte et al., "Observer performance evaluation of the feasibility of a deep learning model to detect cardiomegaly on chest radiographs," *Acta Radiologica Open*, vol. 11, no. 7, Article ID 205846012211073, 2022.

[15] S. S. Sarpotdar, "Cardiomegaly detection using deep convolutional neural network with U-net," 2022, https://arxiv.org/abs/2205.11515.

[16] H. Bougias, E. Georgiadou, C. Malamateniou, and N. J. A. R. Stogiannos, "Identifying cardiomegaly in chest X-rays: a cross-sectional study of evaluation and comparison between different transfer learning methods," *Acta Radiologica*, vol. 62, no. 12, pp. 1601–1609, 2021.

[17] E. Ribeiro, D. A. Cardenas, J. E. Krieger, and M. A. Gutierrez, "Interpretable deep learning model for cardiomegaly detection with chest X-ray images," in *Anais do XXIII Simpósio Brasileiro de Computação Aplicada à Saúde*, pp. 340–347, SBC, London, UK, 2023.

[18] L. K. Singh, Pooja, H. Garg, and M. Khanna, "Deep learning system applicability for rapid glaucoma prediction from fundus images across various data sets," *Evolving Systems*, vol. 13, no. 6, pp. 807–836, 2022.

[19] M. Khanna, L. K. Singh, S. Thawkar, and M. Goyal, "Deep learning based computer-aided automatic prediction and grading system for diabetic retinopathy," *Multimedia Tools and Applications*, vol. 82, pp. 1–48, 2023.

[20] M. Sorić, D. Pongrac, and I. Inza, "Using convolutional neural network for chest X-ray image classification," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1771–1776, IEEE, Opatija, Croatia, September 2020.

[21] A. Bhat, "Automated detection of COVID-19 from X-ray images using deep convolutional neural networks," in *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pp. 2076–2081, IEEE, Greater Noida, India, December 2021.

[22] R. D. Portela, J. R. G. Pereira, M. G. F. Costa, and C. F. F. Costa Filho, "Lung region segmentation in chest x-ray images using deep convolutional neural networks," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 1246–1249, IEEE, Montreal, Canada, July 2020.

[23] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, vol. 33, 2021.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[25] R. Sun, "Optimization for deep learning: theory and algorithms," 2019, https://arxiv.org/abs/1912.08957.

[26] R.-Y. Sun, "Optimization for deep learning: an overview," *Journal of the Operations Research Society of China*, vol. 8, no. 2, pp. 249–294, 2020.

[27] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th international conference on international conference on machine learning*, pp. 265–272, Bellevue, WA, USA, June 2011.

[28] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1285–1290, IEEE, Valencia, Spain, July 2017.

[29] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer, "Scalable second order optimization for deep learning," 2020, https://arxiv.org/abs/2002.09018.

[30] A. L. Friesen and P. Domingos, "Deep learning as a mixed convex-combinatorial optimization problem," 2017, https://arxiv.org/abs/1710.11573.

[31] H. Pan, X. Niu, R. Li, Y. Dou, and H. Jiang, "Annealed gradient descent for deep learning," *Neurocomputing*, vol. 380, pp. 201–211, 2020.

[32] S. Arora, Z. Li, and A. Panigrahi, "Understanding gradient descent on the edge of stability in deep learning," in *International Conference on Machine Learning*, pp. 948–1024, Baltimore, MD, USA, July 2022.

[33] J. Zhang, "Gradient descent based optimization algorithms for deep learning models training," 2019, https://arxiv.org/abs/1903.03614.

[34] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, https://arxiv.org/abs/1609.04747.

[35] N. Ketkar, "Stochastic gradient descent," *Deep learning with Python: A hands-on introduction*, Springer, Berlin, Germanypp. 113–132, 2017.

[36] P. Netrapalli, "Stochastic gradient descent and its variants in machine learning," *Journal of the Indian Institute of Science*, vol. 99, no. 2, pp. 201–213, 2019.

[37] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics*, pp. 177–186, Springer, Paris France, August 2010.

[38] G. Perin and S. Picek, "On the influence of optimizers in deep learning-based side-channel analysis," in *Selected Areas in Cryptography: 27th International Conference*, pp. 615–636, Springer, Halifax, Canada, October 2021.

[39] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, "Model-based deep learning: on the intersection of deep learning and optimization," *IEEE Access*, vol. 10, pp. 115384–115398, 2022.

[40] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," 2019, https://arxiv.org/abs/1910.07454.

[41] Z. Xu, A. M. Dai, J. Kemp, and L. Metz, "Learning an adaptive learning rate schedule," 2019, https://arxiv.org/abs/1909.09712.

[42] M. D. Zeiler, "Adadelta: an adaptive learning rate method," 2012, https://arxiv.org/abs/1212.5701.

[43] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472, IEEE, Santa Rosa, CA, USA, March 2017.

[44] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Readings in Computer Vision*, pp. 638–643, 1987.

[45] P. J. Bickel, B. Li, A. B. Tsybakov et al., "Regularization in statistics," *Test*, vol. 15, no. 2, pp. 271–344, 2006.

[46] F. Girosi, M. Jones, and T. J. N. C. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[47] G. Leibbrandt, "Introduction to the technique of dimensional regularization," *Reviews of Modern Physics*, vol. 47, no. 4, pp. 849–876, 1975.

[48] R. Battiti, "First-and second-order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141–166, 1992.

[49] R. Zdunek and A. Cichocki, "Nonnegative matrix factorization with constrained second-order optimization," *Signal Processing*, vol. 87, no. 8, pp. 1904–1916, 2007.

[50] N. Agarwal, B. Bullins, and E. Hazan, "Second-order stochastic optimization for machine learning in linear time," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 4148–4187, 2017.

[51] A. Lydia and S. Francis, "Adagrad—an optimizer for stochastic gradient descent," *Journal of Computing and Information Sciences*, vol. 6, no. 5, pp. 566–568, 2019.

[52] N. Zhang, D. Lei, and J. Zhao, "An improved Adagrad gradient descent optimization algorithm," in *2018 Chinese Automation Congress (CAC)*, pp. 2359–2362, IEEE, Xi'an, China, November 2018.

[53] Z. Wei, Z. Huang, and J. Zhu, "Position control of magnetic levitation ball based on an improved adagrad algorithm and deep neural network feedforward compensation control," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8935423, 13 pages, 2020.

[54] M. N. Halgamuge, E. Daminda, and A. Nirmalathas, "Best optimizer selection for predicting bushfire occurrences using deep learning," *Natural Hazards*, vol. 103, no. 1, pp. 845–860, 2020.

[55] S. Bock and M. Weiß, "A proof of local convergence for the Adam optimizer," in *2019 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, Budapest, Hungary, July 2019.

[56] Z. Zhang, "Improved Adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pp. 1-2, Ieee, Banff, Canada, June 2018.

[57] M. Maurya and N. Yadav, "A comparative analysis of gradient-based optimization methods for machine learning problems," in *International Conference on Data Analytics and Computing*, pp. 85–102, Springer, Berlin, Germany, January 2022.

[58] L. R. Campos, P. Nogueira, and E. Nascimento, "Tuning a fully convolutional network for velocity model estimation," in *Offshore Technology Conference Brasil*, Rio de Janeiro, Brazil, October 2019.

[59] M. Obayya, M. S. Maashi, N. Nemri et al., "Hyperparameter optimizer with deep learning-based decision-support systems for histopathological breast cancer diagnosis," *Cancers*, vol. 15, no. 3, p. 885, 2023.

[60] A. Pavate, R. J. D. S. Bansode, and D. Analytics, *An Analysis of Derivative-Based Optimizers on Deep Neural Network Models*, CRC Press, Boca Raton, FL, USA, 2021.

[61] G. Amoudi, R. Albalawi, F. Baothman, A. Jamal, H. Alghamdi, and A. Alhothali, "Arabic rumor detection: a comparative study," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 12511–12523, 2022.

[62] M. Munsarif, E. Noersasongko, P. N. Andono, A. Soeleman, and M. Sam'an, "An improved convolutional neural networks based on variation types of optimizers for handwritten digit recognition," 2022, https://ssrn.com/abstract=4055758.

[63] D. Bahrami and S. Zadeh, "Gravity optimizer: a kinematic approach on optimization in deep learning," 2021, https://arxiv.org/abs/2101.09192.

[64] B. Wang, Q. Meng, W. Chen, and T.-Y. Liu, "The implicit bias for adaptive optimization algorithms on homogeneous neural networks," in *International Conference on Machine Learning*, pp. 10849–10858, PMLR, New York, NY, USA, May 2021.

[65] R. Poojary and A. Pai, "Comparative study of model optimization techniques in fine-tuned CNN models," in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–4, IEEE, Ras Al Khaimah, UAE, November 2019.

[66] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of Adam and rmsprop," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11127–11135, Long Beach, CA, USA, June 2019.

[67] M. Yaqub, J. Feng, M. Zia et al., "State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images," *Brain Sciences*, vol. 10, no. 7, p. 427, 2020.

[68] U. Michelucci, *Applied Deep Learning with TensorFlow 2*, Springer, Berlin, Germany, 2022.

[69] B. J. Sullivan, R. Ansari, M. L. Giger, and H. MacMahon, "Effects of image preprocessing/resizing on diagnostic quality of compressed medical images [Chest Radiographs Application]," in *Proceedings., International Conference on Image Processing*, vol. 2, pp. 13–16, IEEE, Washington, DC, USA, October 1995.

[70] S. Perumal, T. Velmurugan, and A. Mathematics, "Preprocessing by contrast enhancement techniques for medical images," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 18, pp. 3681–3688, 2018.

[71] M. Brisinello, R. Grbić, M. Pul, and T. Anđelić, "Improving optical character recognition performance for low quality images," in *2017 International Symposium ELMAR*, pp. 167–171, IEEE, Zadar, Croatia, September 2017.

[72] S. D. Khirade and A. Patil, "Plant disease detection using image processing," in *2015 International conference on computing communication control and automation*, pp. 768–771, IEEE, Pune, India, February 2015.

[73] W. Förstner, "Image preprocessing for feature extraction in digital intensity, color and range images," in *Geomatic Method for the Analysis of Data in the Earth Sciences*, pp. 165–189, Springer, Berlin, Germany, 2003.

[74] M. Sharif, S. Mohsin, M. J. Jamal, and M. Raza, "Illumination normalization preprocessing for face recognition," in *2010 the 2nd conference on environmental science and information application technology*, vol. 2, pp. 44–47, IEEE, Wuhan, China, July 2010.

[75] J. Meier, R. Bock, G. Michelson, L. G. Nyúl, and J. Hornegger, "Effects of preprocessing eye fundus images on appearance based glaucoma classification," in *Computer Analysis of Images and Patterns: 12th International Conference, CAIP 2007*, pp. 165–172, Springer, Vienna, Austria, August 2007.

[76] F. Pérez-García, R. Sparks, S. J. C. M. Ourselin, and P. Biomedicine, "TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning," *Computer Methods and Programs in Biomedicine*, vol. 208, Article ID 106236, 2021.

[77] G. Caseneuve, I. Valova, N. LeBlanc, and M. J. P. Thibodeau, "Chest X-ray image preprocessing for disease classification," *Procedia Computer Science*, vol. 192, pp. 658–665, 2021.

[78] J. Shijie, W. Ping, J. Peiyi, and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," in *2017 Chinese automation congress (CAC)*, pp. 4165–4170, IEEE, Jinan, China, October 2017.

[79] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, pp. 117–122, IEEE, Poland, May 2018.

[80] S. Gu, M. Pednekar, and R. Slater, "Improve image classification using data augmentation and neural networks," *SMU Data Science Review*, vol. 2, no. 2, p. 1, 2019.

[81] D. Paper and D. Paper, "Increase the diversity of your dataset with data augmentation," *State-of-the-Art Deep Learning Models in TensorFlow*, Springer, Berlin, Germanypp. 37–64, 2021.

[82] R. Poojary, R. Raina, and A. Kumar Mondal, "Effect of data-augmentation on fine-tuned CNN model performance," *IAES International Journal of Artificial Intelligence*, vol. 10, no. 1, p. 84, 2021.

[83] A. Soliman and J. Terstriep, "Keras Spatial: extending deep learning frameworks for preprocessing and on-the-fly augmentation of geospatial data," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pp. 69–76, Chicago, IL, USA, November 2019.

[84] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017, https://arxiv.org/abs/1712.04621.

[85] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval*, vol. 12, no. 4, pp. 461–486, 2009.

[86] A. K. A. De Medeiros, A. Guzzo, G. Greco et al., "Process mining based on clustering: a quest for precision," in *Business Process Management Workshops: BPM 2007 International Workshops*, pp. 17–29, Springer, Brisbane, Australia, September 2008.

[87] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "Combining evaluation metrics via the unanimous improvement ratio and its application to clustering tasks," *Journal of Artificial Intelligence Research*, vol. 42, pp. 689–718, 2011.

[88] S. Candemir, S. Jaeger, W. Lin, Z. Xue, S. Antani, and G. Thoma, "Automatic heart localization and radiographic index computation in chest x-rays," in *Medical Imaging 2016: Computer-Aided Diagnosis*, vol. 9785, pp. 302–309, SPIE, Bellingham, WA, USA, 2016.