Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 09, No. 02, June 2023.

AUTOMATIC DETECTION OF PNEUMONIA USING CONCATENATED CONVOLUTIONAL NEURAL NETWORK

Ahmad T. Al-Taani and Ishraq T. Al-Dagamseh

(Received: 12-Dec.-2022, Revised: 8-Mar.-2023 and 14-Mar.-2023, Accepted: 30-Mar.-2023)

ABSTRACT

Pneumonia is a life-threatening disease and early detection can save lives. Many automated systems have contributed to the detection of this disease and currently, deep-learning models have become among the most widely used models for building these systems. In this study, two deep-learning models are combined: DenseNet169 and pre-activation ResNet models and used for automatic detection of pneumonia. Two methods are used to deal with the problem of unbalanced data: class weight, which enables to control the percentage of data to be used from the original data for each class of data, while the other method is resampling, in which modified images are produced with an equal distribution using data augmentation. The performance of the proposed model is evaluated using a balanced dataset that consists of 5856 images. Achieved results were promising compared to those obtained by several previous studies. The model achieved a precision value of 98%, an area under curve (AUC) based on ROC of 97% and a loss value of 0.23.

KEYWORDS

Deep learning, Convolutional neural networks, Chest X-rays, Pneumonia, Pre-activation ResNet, DenseNet169.

1. INTRODUCTION

Pneumonia is a disease that targets the lungs. The cause of the disease, whether it is a virus, bacteria or fungus [1], seeks to damage one or both lungs. According to a report issued by the World Health Organization (WHO), it has been indicated that pneumonia is one of the most serious life-threatening diseases, especially for children under the age of five, as it kills about 1.2 million children of this age category around the world [2]. Other statistics have shown how dangerous and threatening pneumonia can be to a person's life. For example, a study indicated that there are about 4.5 million people infected with pneumonia around the world each year and more than 50,000 people die as a result of this disease in the USA each year [3]. In another study in 2012, it was indicated that there were about 28,952 deaths in the UK that were caused by pneumonia that year. In two statistical studies in 2016, the first concluded that about 880,000 children under the age of two died as a result of this disease [4], while the second in the same year indicated that Indian reports indicated that the highest death rates around the world were due to pneumonia with about 158,176 deaths in that year [5]. According to a report by the WHO, one out of every three deaths is due to pneumonia in India [6]. Figure 1 shows three X-ray images representing a normal, bacterial pneumonia and viral pneumonia images.

Figure 1. Normal, bacterial and viral pneumonia images [17].

From Figure 1, we can clarify the difference between the images, where in the normal chest image in the left, specifically by focusing on the lungs area, which appears in a black color, the presence of gases and their normal exchange are expressed. The middle image expresses a lung infected with pneumonia of a bacterial type and the affected area is the one to which the two arrows point which

A. T. Al-Taani and I. T. Al-Dagamseh are with Department of Computer Science, Yarmouk University, Irbid, Jordan. E-mails: ahmadta@yu.edu.jo and ishraq500@gmail.com

appears as a blurring or gray area. The image in the right shows two lungs infected with a viral type of pneumonia which appears in a hazy color. The difference between the bacterial and viral types is that the bacterial type infects one side of the lungs, while the viral one may infect both lungs.

Radiologists focus on identifying white spots and foggy or blurry areas to locate the injury, but the fact that the X-ray images of the chest are in black and white causes another challenge for the diagnostician to make a proper diagnosis of the disease [7]. This needs experience and accuracy. Another problem is that the symptoms of this disease overlap which the symptoms of other diseases such as lung cancer, which appears as opacity in the picture [8]. Therefore, there is a need to build automated systems capable of diagnosing this disease with greater accuracy and less time. The stages of developed algorithms and methods used for diagnosing pneumonia can be divided into three main stages: the oldest stage which started in the nineties used textual analysis to detect the CXR images [9]-[10]. As for the intermediate stage, it started with the emergence of machine-learning algorithms like support vector machine (SVM), k-nearest neighbor (KNN), Naïve bayes (NB) and random forest (RF) [11]-[12]. Research in the current period; i.e., the third stage, focused on the use of deep-learning models, specifically convolutional neural network (CNN) models, such as VGGNetm ExNet and GoogleNet [13] and ResNet models, like ResNet-38, ResNet-50 and Resnet-101 [3], [14]-[15]. Deep-learning models are mostly used for feature extraction and some focus on identifying regions of interest in the image; i.e., identifying the most important and effective area in diagnosing the disease [8].

In this paper, we proposed a new model consisting of two integrated deep-learning models for the phase of extracting features from CXR images to automatically diagnose pneumonia and those models are: pre-activation ResNet and DensNet169. After a critical analysis of the literature, we have deduced that the pre-activation ResNet model has achieved better results than the ResNet model in the medical field for instance, in [16], it obtained efficient results for the detection of prostate cancer. We have also found that those two models are the most common models used by different researchers for solving similar problems in the medicine field like Alzheimer disease [47], but these modes are not used for the detection of pneumonia. The decision of combining the two models is taken experimentally, after conducting the experiments using each model individually; it was found that the combined model has achieved better results that the results obtained when applying each model individually. The other goal of this study is to show the importance of supporting the numerical results to evaluate the performance of the model with what is called the performance curve, which shows how the model learns during the training and validation processes.

The rest of this paper is organized as follows: In Section 2, we present the most prominent previous studies, specifically those that focused on deep-learning models and supported their results with a performance curve. Section 3 presents the proposed model in detail. Section 4 presents and discusses the experimental results. Finally, conclusions drawn from this study and directions for future work are presented in Section 5.

2. RELATED WORKS

Most of the recent studies, specifically those that were used to build systems for automatic detection of pneumonia, focused on the use of deep-learning models in general and CNN models in particular, as they have a high ability to extract the smallest details from images. We have highlighted in this section the most related studies that used deep-learning algorithms on the same database proposed in [17].

Most of the previous approaches used pre-trained models on large datasets. In [18], three types of pre-trained CNN models are used in the early feature-extraction stage; these include AlexNet, VGG-16 and VGG-19. Five algorithms are used, while in the classification stage, including KNN, decision tree (DT), linear discriminant analysis (LDA), linear regression (LR) and SVM are used. The best accuracy they reached was about 99%, but the performance curve showed a higher fluctuation in the validation phase than in the training phase.

In [19], a DenseNet-121 pre-trained model is used as a feature extractor with a deep neural network as a classifier. The best accuracy reached is 98% after conducting three experiments on the model, while the shape of the performance curve was unstable in the first and third experiments and reached a more

stable shape in the second experiment.

Another study based on pre-trained models is proposed by Chouhan et al. [1] using five pre-trained CNN models: ALexNet, DenseNet121, InceptionV3, ResNet18 and Google Net. The proposed model achieved an accuracy of 96.39% and slightly squiggly performance curves for all models.

In [20], a pre-trained model (Xception) is used in the feature-extraction step and an under-sampling technique is used to re-balance the dataset. The proposed model reached a value of 96% for AUC measure, a recall of 99%, a precision of 84%, an F1-score of 0.91 and a validation loss of 0.04. The performance curves started to fluctuate until the 20^{th} epoch, which is where the stability started perfectly.

In [21], five pre-trained CNN models: Xception, Inception-v3, VGG-16, ResNet50 and DenseNet201, are suggested to extract features from chest X-rays (CXR), then extracted features by every model are combined using Dempster–Shafer theory. The performance curve measure is not used for the evaluation of the model; only accuracy, precision and F1-score metrics are used. The model obtained an accuracy of 97%, a recall of 98% and an AUC of 99%.

In [22], Islam et al. proposed a model which consisted of two pre-trained CNN models: SqueezeNet and Inception-V3, to extract attributes from CXR images. SVM, KNN and Artificial Neural Network (ANN) are used for classification. The best reported results were obtained when using the ANN, which gave an accuracy and a sensitivity of 98% as well as a specificity and a precision of 99%.

Ayan and Ünver [23] compared the performance of the two pre-trained models Vgg16 and Xception. The results showed that the Vgg16 model outperformed the Xception model with an accuracy of about 87%, while the Xception model achieved an accuracy of 82%. For the performance curves, turbulence and fluctuations are evident in the performance curve with respect to the validation stage of the VGG16 model, while the performance curve seemed less turbulent with respect to the Xception model with a difference in the results of the two curves in the training and the validation phases.

A pre-trained ResNet-50 model is employed by Hussain et al. [24] for classifying injured and non-injured CXRs; data augmentation is also used to increase the number of images in the dataset. The model obtained an accuracy of about 90%, while the performance curve is very turbulent and has little stability in the validation phase.

Singh et al. [25] built a modified model of ResNet, called Quaternion CNN. The proposed model differs from the original ResNet model in which every convolutional layer is replaced by a Quaternion convolutional layer; also, every Relu layer is replaced by an exponential linear unit. The proposed model reached an accuracy of 93%, a loss value of 0.25% and an F1-score of 94%. The shape of the performance curve was turbulent until the fortieth step and then seemed perfect; also, the loss scale was stable in shape and the difference between the training curve and the validation curve was increasing over time.

Rači et al. [26] applied the traditional CNN model layer by layer where the accuracy of that model reached 90%. The performance curve for the validation stage was highly fluctuating and more pronounced in the loss scale than in the performance curve for the accuracy measure, which, although fluctuated, was more fluid and the difference between the two curves in the training and validation stages is less. In [3], CNN and ResNet models are combined and used for the detection of pneumonia. Also, Jain et al. [27] used six CNN models and four pre-trained models: VGG16, VGG19, Inception V3 and ResNet50, for pneumonia detection. The models obtained good results, while the performance curve for all models was unstable with a clear difference between the performances of the training and validation curves.

Mabrouk et al. [45] proposed the use of three CNN pre-trained models including DenseNet169, MobileNetV2 and Vision Transformer for the detection of pneumonia using ImageNet chest X-rays database. Experimental results showed that the proposed approach obtained good results compared to other previous approaches, with an accuracy of 93.91% and an F1-score of 93.88%.

Rahman et al. [46] used four CNN pre-trained models including AlexNet, ResNet18, DenseNet201 and SqueezeNet for the detection of pneumonia tested on a dataset consisting of 5247 chest X-ray images. The authors reported promising classification results compared to other approaches.

In [48], different transfer learning techniques are used to detect pneumonia from chest X-rays. Experiments showed that promising results are obtained by the proposed approach.

In [49], a deep-learning (MobileNet) model is proposed for the detection of pneumonia using chest X-ray images. The dataset proposed by Kermany [17] is used for training and testing the model. The model achieved an accuracy of 97.34% for training, an accuracy of 87.5% for validation and an accuracy of 94.23% for testing.

In [50], A CNN model using a dropout in the conv layer is proposed for the detection of pneumonia from chest X-rays. Kermany dataset [17] is used for training and testing the proposed approach. Achieved results were promising compared to recent approaches (accuracy = 97.2%, recall = 97.3%, precision = 97.4% and AUC = 0:982). Table 1 presents a summary of the key papers reviewed in this study using the same dataset [17].

Table 1. Summary of the key papers.

Ref.	Approach	Limitations	Best Results
[1]	CNN model with pre-trained models	Slightly zigzag curves in the performance curve.	Accuracy: 96.39%, Recall: 99%
[3]	CNN with lightened image on increased contrast with ResNet	A low value is a measure of accuracy with no consideration of the performance curve.	Accuracy: 78.73%
[19]	Lightweight Deep ANN	Unstable performance curve.	Accuracy: 98%
[20]	Xception pre-trained model	Different results for different performance measures.	AUC: 0.96%, Recall: 0.99, Precision: 0.84%, F1-score: 91%
[23]	Vgg16 and Xception models	Oscillating curves of both models. Difference between the performance curve for the training and validation phases of the Xception model.	Accuracy: 87%
[26]	CNN	The performance curve was too squiggly in the validation phase.	Accuracy: 90%
[27]	CNN, VGG16, VGG19, ResNet50 and InceptionV3.	The performance curve is unstable for all models.	Accuracy: 92% and 88%,

3. MATERIALS AND METHODS

3.1 Materials

The dataset proposed by Kermany et al. [17] is used to evaluate the proposed approach. The dataset consists of 5856 CXR images of pneumonia of two classes: infected images and not infected images. This dataset is for children within ages between 1 and 5 years and classified by specialists in the "Women and Children's Medical Center" at Guangzhou [42]. The dataset is divided into 89% for training, 10% for testing and 1% for validation. This distribution is presented in Table 2.

Table 2. The original distribution of the dataset.

	Normal	Pneumonia	Total
Training	1341	3875	5216
Testing	234	390	624
Validation	8	8	16
Total	1583	4273	5856

It can be seen from Table 2 that the percentage of pneumonia images reached about 73%, while the percentage of normal images was about 27% for the whole dataset. The percentage of pneumonia images in the data test reached about 63% and the percentage of normal images is about 37%.

The second experiment is conducted after redistributing the dataset into 75% for training, 22% for validation and 3% for validation, as presented in Table 3.

Table 3. The redistribution of the dataset.

	Pneumonia	**Normal**	**Total**
Training	3207 (73%)	1185 (27%)	4392 (75%)
Testing	811 (63%)	477 (37%)	1288 (22%)
Validation	88 (50%)	88 (50%)	176 (3%)
Total	4229 (72%)	1627 (28%)	5856 (100%)

The third experiment is conducted after applying data augmentation to the original dataset with 75% for training, 22% for testing and 3% for validation, as presented in Table 4. The distribution here was an equal distribution for each class in the training, testing and validation, since we are trying to test the effect of this distribution of the classes on the performance of the model. The dataset size after augmentation reached 29,890 images, divided as 22,417 for training, 6575 for testing and 898 for validation (Table 4).

Table 4. The distribution of the dataset after augmentation.

	Pneumonia	**Normal**	**Total**
Training	11,432	10,985 (49%)	22,417 (75%)
Testing	3,419 (52%)	3,156 (48%)	6,575 (22%)
Validation	449 (50%)	449 (50%)	898 (3%)
Total	4229 (72%)	1627 (28%)	29,890 (100%)

3.2 Parameter Settings

Parameter setting is one of the most sensitive steps and affects the results of the model; it is used to reduce network error. The values of these parameters are chosen experimentally for all experiments, as presented in Table 5. Parameters' values are also related to the type of optimizer used; for instance, a learning rate of 0.0001 is used for the Adam optimizer for updating the network weights. 30 epochs are used, which indicates the number of times the training dataset is divided according to the patch size across the network and accordingly, the parameters in the network are updated. Default steps per epoch that are equal to several samples in the training dataset are divided by the batch size and default steps between validation epochs that are equal to several samples in testing dataset are divided by the batch size for fitting the model with data step. L2 regularization is used with the SGD optimizer to overcome the overfitting problem, where the learning rate is changed to 0.00001 and the number of epochs is also changed to 40; this value represents the number of forward and backward stages to be used in the training and verification stages. Dropout regularization is used in the resampling stage instead of L2 regularization.

Table 5. Training hyper-parameters.

Parameter	**Value**
Batch size	24
Optimizer	Adam, SGD
Learning rate	0.0001(Adam) and 0.00001(SGD)
Number of	30,40

Steps per epoch	Number of training samples / batch size
Validation steps	Number of testing samples / batch size
Regularization	L2, Dropout

3.3 Evaluation Metrics

The performance of the proposed model is evaluated using five metrics: accuracy, precision, recall and AUC and it is the rate between the TP and FP. In general, it is used in binary classification models and it is used to evaluate models that take into account the same percentage of importance for the classification of both types of samples.

Binary cross entropy is also used as a loss function to measure the difference between the estimated probability and the actual probability [43]. These metrics are formulated using Formulae 1 - 4.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$AUC = \int_0^1 \frac{TP}{P}\, d\frac{FP}{N} = \frac{1}{P\,N} \int_0^N TP\, d\, FP \tag{4}$$

where, TP, FN, FP and TN refer to True Positive, False Negative, False Positive and True Negative, respectively. TP represents the number of correct classifications of positive observations of phenomena-infected images; FN represents the number of incorrect classifications of positive observations (normal images) that are incorrectly labeled as phenomena-infected images. FP represents the number of incorrect classifications of negative observations and TN represents the number of correct classifications of negative observations.

3.4 Methods

The overall architecture of the proposed approach is presented in Figure 2. It comprises three phases: pre-processing, data augmentation and normalization, as well as feature extraction and classification.

Since the dataset used in this study is unbalanced, two different techniques are used to tackle this problem: the class-weight and the resampling techniques. In the first technique, the weights of the different classes in the dataset are rebalanced so that the model is prevented from bias towards the most present class. This technique uses a kind of hyper-parameters added during the fitting or training processes; i.e., it is an amendment to the existence and weights of the classes so that the presence of the two classes is balanced. Two experiments are conducted; in the first experiment, the same weight is given for both classes, while in the second experiment, higher weight is given to the class with the least presence (the normal class). For the second technique (resampling), the number of images is

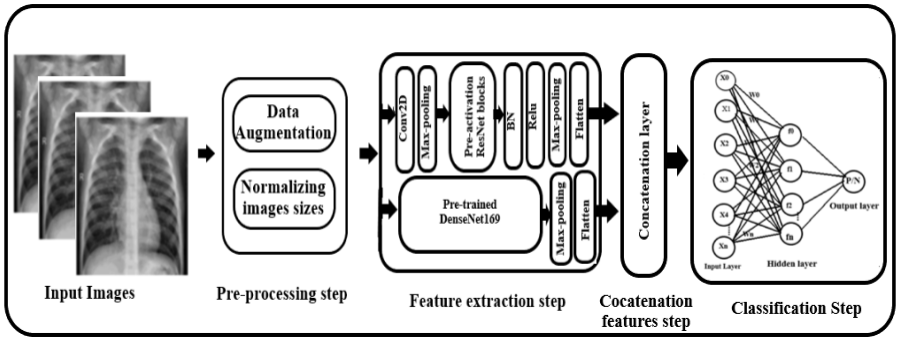

Figure 2. The proposed approach.

increased for the least frequent category based on the data augmentation method, so that the number of images for both categories is balanced.

3.4.1 Pre-processing

Data augmentation is used to increase the volume of the dataset, since deep-learning models need large data to give a high performance. It is also used to address the problem of overfitting, i.e., the model achieves good results on the training data, while it doesn't perform well on new unseen data [28]. Eight augmentation methods are applied online in this study as presented in Table 6. All these operations are performed randomly to produce new images, where more than one operation was applied [29]. Image sizes for the dataset are normalized to 224x224 to be suitable for the DenseNet169 pre-trained model.

3.4.2 Feature Extraction

Two CNN models are used in this study, including pre-activation ResNet and DenseNet169. The pre-activation ResNet model is built from scratch; i.e., defined layer by layer and the DenseNet169 model is applied using the weights used during the training phase on the ImageNet dataset.

Pre-activation ResNet Model

Pre-activation ResNet is a special version of ResNet proposed by He et al. [30]. The ResNet model has two different architectures based on the flow of data. The first architecture follows a main path, while the second architecture takes a shortcut path; i.e., data can travel between layers and bypass the others depending on the type of main path [31].

Table 6. The proposed data augmentation methods.

Augmentation process	**Description**
Rescale (1/255)	Transforming the image pixel values (0 – 255) to values between 0 and 1.
Rotation = 40	Rotating images between 1° and 40° degrees.
Width-shift = 0.2	Shifting images horizontally with a percent of 0.2.
Height-shift = 0.2	Shifting images vertically with a percent of 0.2.
Zoom-range = 0.2	Scaling and inverse scaling of images.
Horizontal flip = True	Flipping images horizontally.
Fill-mode = 'nearest'	Filling the nearest of the body shape with points (the border of the image).
Shear-range = 0.2	Shearing of images.

The first architecture consists of an arrangement of layers: a convolutional layer, a batch normalization layer and a rectified linear unit (Relu) layer or a bottleneck. The second architecture is based on the same principle with a different arrangement of the layers, so that their arrangement in the main path is as follows: BN layer, a Relu layer and a convolutional layer (conv layer). Figure 3 shows the two different architectures of the ResNet. In this study, we used a pre-activation ResNet architecture with 9 pre-activation residual blocks.

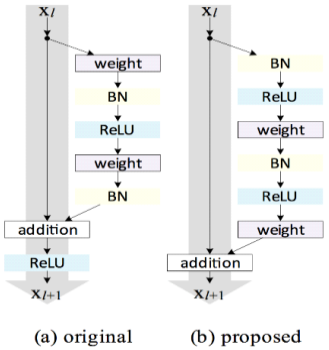

Figure 3. ResNet *vs.* pre-activation ResNet [30].

Equation 5 represents the block of the pre-activation ResNet, where the output layer in the pre-activation ResNet is $Xl + 1$ and the shortcut path Xl is the input features. F is the path,°F is the activation function and Wl is the weight.

$$\mathrm{X}l + 1 = \mathrm{X}l + \mathrm{F}(°\mathrm{F}(\mathrm{X}l), \mathrm{W}l) \quad (5)$$

BN layer is based on the principle of estimating the means and standard deviations of the inputs of each layer for each small batch or subset of the training set. Bjorck et al. [32] defined BN as "a normalization method for layer activation in the hidden layers of deep-learning models". BN layer is used to increase the training speed and to improve the accuracy in deep-learning models [33].

Relu is the most widely used type of activation function in deep-learning models. It defines the output of the hidden layer inputs and works by converting all numbers less than zero into zero while keeping the other numbers. Equation 6 represents how the Relu works by converting each number less than zero into be zero and keeping the other numbers at their values.

$$f(x) = max(0, x) \quad (6)$$

Conv layer is the third layer on the pre-activation ResNet block. It works by moving the filter on the images to extract the features. This process is done by multiplying the value of the image's pixel array with the filter number, which produces what is called a feature map [34]. In practice, that sequence is defined iteratively again in the main path as we define one conv layer in the shortcut path. As for the layers before those blocks, the conv layer is defined, followed by the max-pooling layer to reduce the size of feature maps. After defining the blocks, we must define (BN => Relu) [35]. The next layer is the pooling and flatten layer; its function is to reshape the feature maps from a two-dimensional array to a one-dimensional array to be fed into the ANN classifier after merging it with other features from the DenseNet169model. Figure 4 shows the main layers of the proposed pre-activation ResNet model.

Pre-trained DenseNet169 Model

DenseNet169 is a type of densely connected convolutional layer with 169 layers. It is proposed by Huang et al. [36] and considered as one of the most popular CNN models used by researchers for classification and segmentation tasks. DenseNet169 is trained on many large datasets, but the volume of these datasets is less than those used by some researchers like Kundu et al. [37] and El Asnaoui et al. [38]. In this study, we used the DenseNet169 model with the ImageNet datasets for feature extraction and ANN was used for classification. Figure 5 shows a DenseNet model with 5 layers. The pre-trained DenseNet169 model is applied after resizing the image size into 224x224. The features extracted from the two models are combined using the concatenation layer [39].

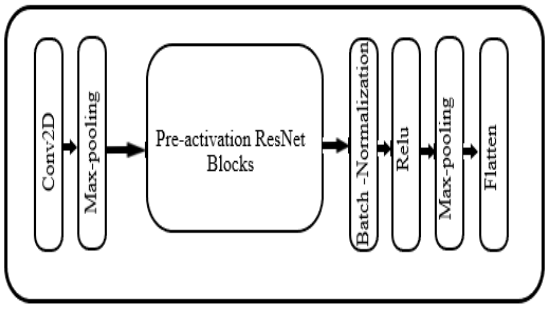

Figure 4. The proposed pre-activation ResNet.

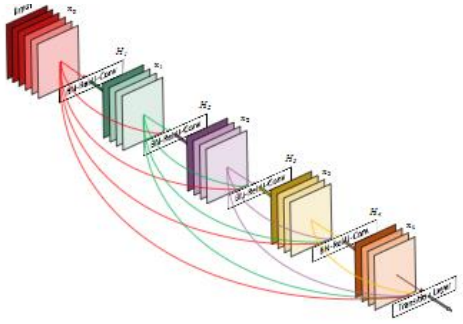

Figure 5. DenseNet with 5-layer connection [35].

3.4.3 Classification

The network consists of one hidden layer, 450 neurons, a Relu activation function and one output layer with one neuron. The number of layers and the number of neurons are determined experimentally; i.e., one hidden layer and 450 neurons gave the best results. ANN is used for classification and the sigmoid activation function is also used in this study, since it is commonly used for similar tasks [40]. Feature extraction is performed using two models based on a certain number of samples or specific batch.

4. EXPERIMENTAL RESULTS

In this section, we present the results and performance curves that express how the model performed during the training and validation phases. In the three main stages: original distribution stage, class weight stage and re-sampling stage, the results of each stage are divided into sub-stages based on the type of optimizer used and other additional criteria. The metrics used for evaluation are: precision, recall, accuracy, loss and AUC. The model's performance in the training phase is represented in blue for all curves and the orange curve expresses the model's performance during the validation phase.

A different number of layers is used for the two models in the feature-extraction stage depending on the used optimizer. For instance, 85 layers are used for pre-activation ResNet and DenseNet169 used with the Adam optimizer, while 37 layers are used with the SGD optimizer. For the classification step, an ANN is used with one hidden layer containing 450 neurons as well as a Relu activation function and one output layer with one neuron and a sigmoid activation function. Since we applied the pre-activation ResNet model from scratch, we tried to apply it with different numbers of layers. After a number of attempts, we found that the proposed model obtained best results when the number of layers is 85 for the Adam optimizer (including feature-extraction layers and two classification layers) and 37 layers for the SGD optimizer.

These classes are designed for each of the three main stages, dealing with the original data, class weight and re-sampling stages. Numbers of features extracted using the pre-activation ResNet and DenseNet169 models are 200704 and 14976, respectively. The input size of the classifier for all experiments is 224x224x3.

4.1 Results Using the Original Dataset

The performances of the proposed models are represented using performance curves which express the learning behaviour of the models during training and validation stages. The curve consists of two levels: the x-axis expresses the number of epochs or steps that the model walks back and forth during training and validation. The y-axis expresses the number of metrics used in the evaluation. Each Cartesian level contains two curves. The first is the training curve (blue curve), which expresses the performance of the model when trained on the training data. The second is the validation curve (orange curve), which expresses the performance of the model in the validation stage.

4.1.1 Results Using Adam Optimizer

Figure 6 shows how the model behaviour during the training and the validation phases when using the Adam optimizer. It can be seen from the Figure that the model was stable during the training stage, while it was unstable during the validation stage. We found that the dropout has increased the loss, which is worthy to be investigated more in future research. It can be seen also from Figure 6 how the model was learned throughout the training and validation periods. Looking at the movement of the training and validation curves, it can be noticed that the movement of the training curve (blue-color curve) is stable throughout the period, while the movement of the validation curve for all the metrics was very mobile and completely unstable, ending with very poor results at the last epoch.

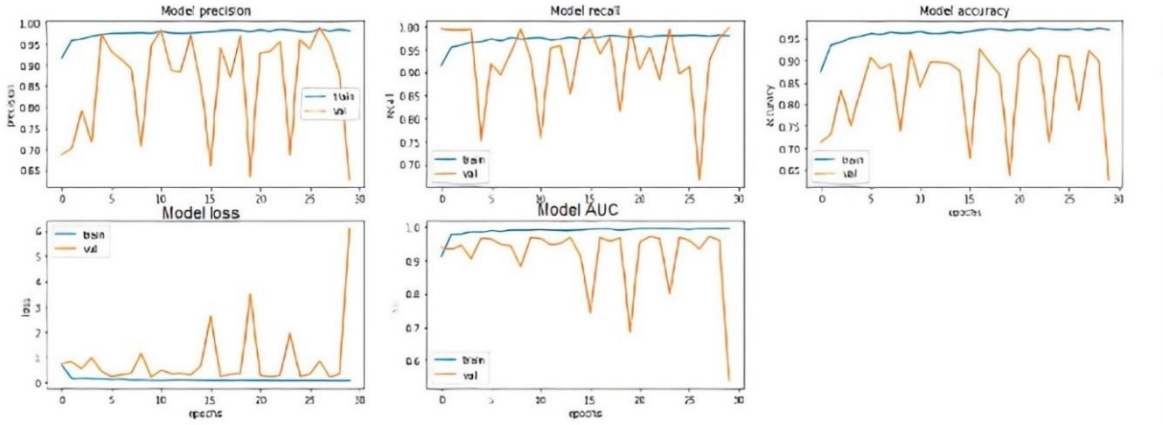

Figure 6. Results using the original dataset and the Adam optimizer.

4.1.2 Results Using SGD Optimizer

Figure 7 shows the model behaviour during the training and the validation phases when using the SGD optimizer. It can be seen from this Figure that the curves began to stabilize clearly. We also found that there is a difference between the performance of the training curve and that of the validation curve. It can be also noticed that the curves started to stabilize a little compared to the previous stage after reducing the number of layers and reducing the learning rate and the length value of l2. Also, there is a difference between the performance of the training curve and that of the validation curve.

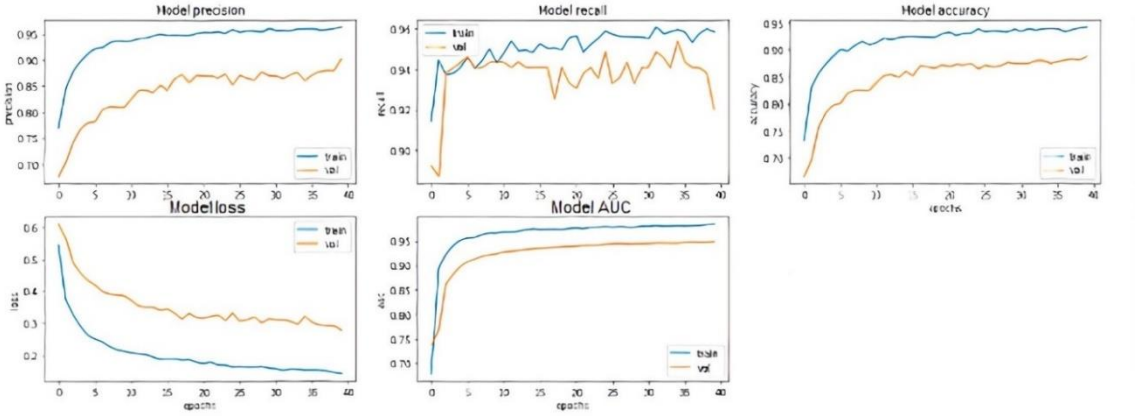

Figure 7. Results using the original dataset and the SGD optimizer.

4.2 Results Using Class Weighted Dataset

4.2.1 Results Using Adam Optimizer

Figure 8 shows the results of the proposed model when using the Adam optimizer with a class weight dataset. The class weighting technique is used to tackle the problem of imbalanced data and to improve the loss values from the five measures. The model's performance curves in the validation stage express the inability of the model to generalize all the validation data, which is shown by the highly volatile curve in the validation stage (orange color).

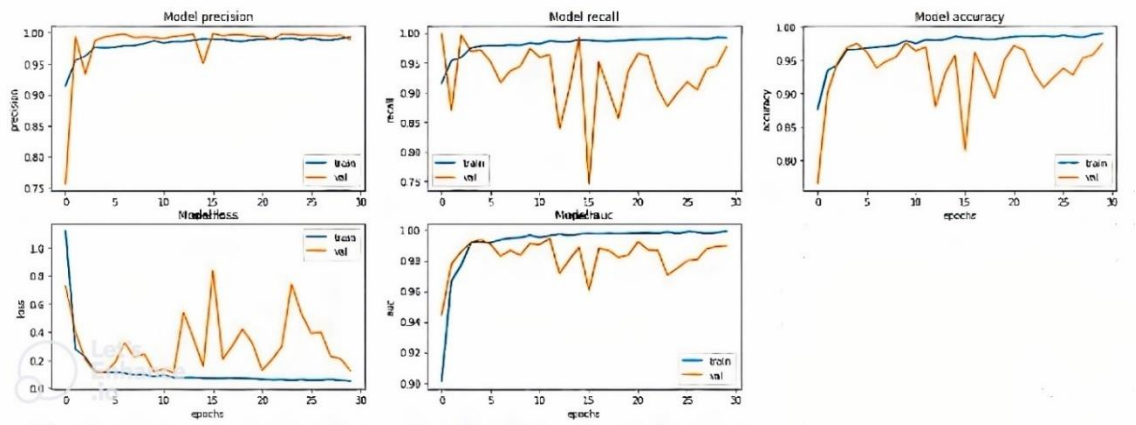

Figure 8. Results using class weight and the Adam optimizer.

After an analysis of the results for this stage (Figure 8), we noticed that the precision started with slightly high values at the beginning of the training with a value of 93% and began to increase with the passage of learning time, where the values were stabilized at epoch 10 until the end of the specified period at the 30th epoch, to stop at a value of 99%. For the validation curve, the precision started with low values at the beginning of the retraining, but it quickly adjusted the values to reach the highest level at epoch 2, then it suddenly decreased to reach a value of 94% at the third epoch. Then, it returned to high and stable values from epoch 5 to epoch 14 and suddenly decreased by one epoch and returned to rising and stabilized again until the end of the period, reaching a value of 98% at the 30th epoch.

The recall values were stable from the beginning of the fifth epoch until the end of the period to stop at a value of 99%. For the validation curve, it started with very high values, but it was fluctuating throughout the training period and reached its lowest level in the middle of the period to go up, but with high fluctuations. It reached a high value of 99% at the end of the period.

The accuracy values started with the lowest value of 88% at the first epoch, but it continued to improve and began to stabilize at epoch 15 until the end of the period to stop at a value of 99%. For the validation curve, it was very low at 75%, but it quickly rose at the third epoch; however, its behaviour was very volatile throughout the period, but it stopped at the end of the period at a good value of 97%.

For the loss metric, it is concluded that the model during training continued to decline from the beginning of the third epoch with great stability in the curve and reached a value of 0.002 at the end. While the behaviour of the model during the validation period was fluctuating from the beginning, but it stopped at a small error rate of 0.01 at the last epoch.

Lastly, the AUC values in the training phase started to stabilize early at epoch 3 until the end of the period to stop at a value of 99%, while the performance of the model in the validation stage was fluctuating throughout the period to stop at a value of 97%.

4.2.2 Results Using SGD Optimizer

Several attempts and different values have been tried for the weights of the classes at this stage. In the first attempt, we used equal class weight with the SGD optimizer with less number of pre-activation blocks (three blocks) to minimize the complexity of the model. In the second attempt, we have given a higher weight for class 0, which had a lower distribution between the two classes; the original distribution for class 0 is multiplied by 1.5, while the original weight for class 1 is kept unchanged.

- ***Results Using Equal Class Weights***

Figure 9 shows the results of the proposed model when using the SGD optimizer with equal class weights.

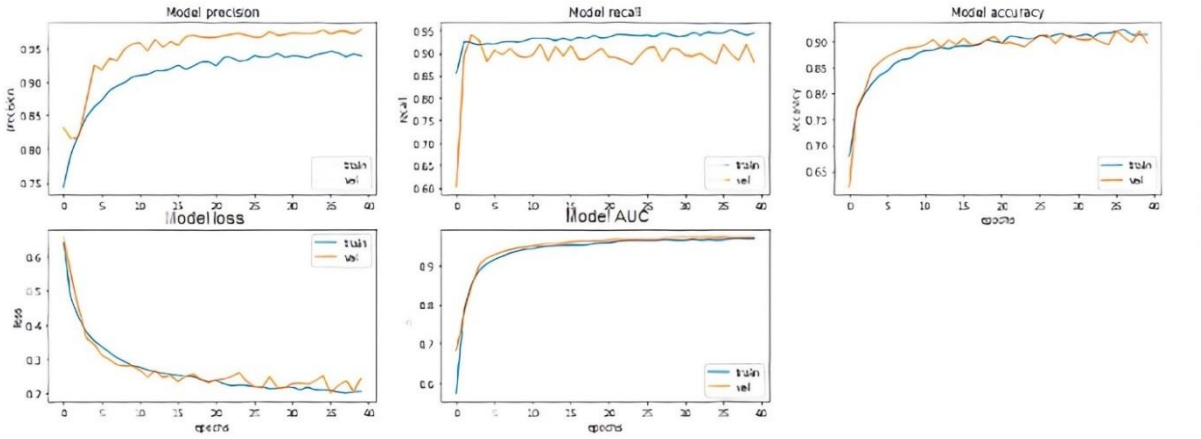

Figure 9. Results using equal class weights and the SGD optimizer.

After a careful analysis of the results in Figure 9, we notice that the precision started with slightly low values from the beginning of the training, but it quickly began to gradually rise from epoch 5 to epoch 40 (end of training). The performance curve was almost stable with very few bends reaching a precision value of about 95%. The model's behavior during the validation period was almost stable with very few zigzags that continued until the end of the period to reach a value of 97%.

The recall values during the training stage were stable from the beginning of the third epoch until the end of the training, stopping at a value of 93%, while during the validation stage, the recall started with low values, but it fluctuated slightly throughout the period reaching an average value of 88%.

The accuracy during the training phase started with a low value of 68% at the first epoch, but it continued to rise and began to stabilize at epoch 10 until the end of the training to stop at a value of

92%, while in the validation phase, its behavior was slightly volatile throughout the period and stopped at the end of the period at a value of 89%.

It is also noticed that the behaviour of the model during the training phase continued to decline from the beginning of epoch 3 reaching a loss value of about 0.3. It continued to decline until the end of the period to reach a loss value of about 0.19. The values for the AUC measure during the training and validation phases started stable from the beginning to the end of the period, stopping at a value of about 97%.

- ***Results Using Different Class Weights***

Metrics' values and numbers improved when changing the weight of the least available samples in the data, which is the normal class, by 1.5 compared to 1.0 for the highest available samples. Although the curves are less stable than in the previous stage, as shown in Figure 10, there were slightly better results as numbers, where the precision ratio reached 98% and the stability of performance relative to the AUC metric remained at 97% and the recall metric remained at 88%, while the accuracy metric improved by 0.01. Its value becomes 90% and the loss ratio becomes 0.23.

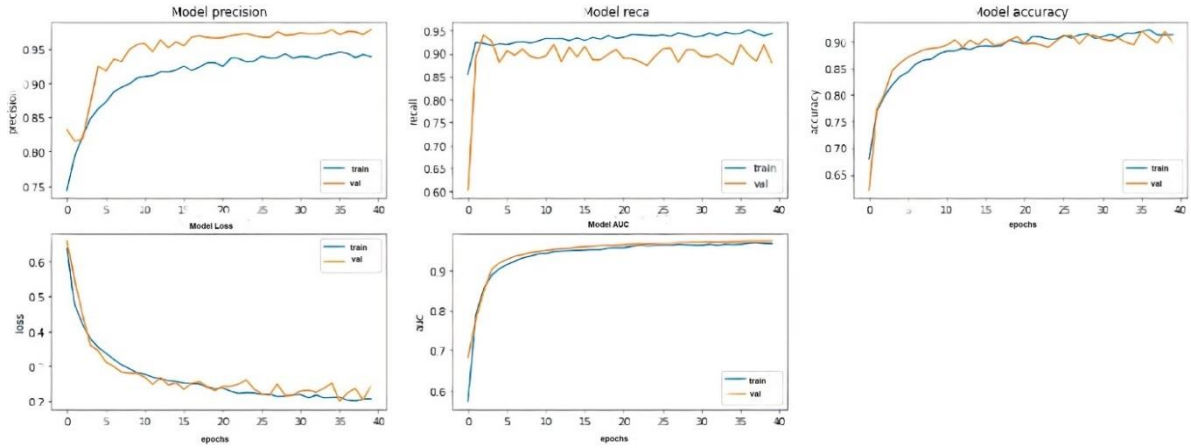

Figure 10. Results using different class weights and the SGD optimizer.

4.3 Results Using Re-sampled Dataset

In this stage, we added a dropout layer with values of 0.8 and 0.9. This layer is used to decrease the complexity of the nodes in the ANN in a dense layer by randomly picking nodes to make them idle or disabled for a certain time; this step is also used to reduce the effect of the overfitting problem. Table 7 shows the hyper-parameters used in this stage for both Adam and SGD optimizers.

Table 7. Training hyper-parameters in the re-sampling stage.

Parameter	Value
Batch size	24
Learning rate	Adam: 0.0001; SGD: 0.00001
Number of epochs	30, 40
Steps per epoch	No. of training samples: batch-size
Validation steps	No. of testing samples: batch-size
Regularization method	L2, dropout

4.3.1 Results Using Adam Optimizer

Figure 11 shows the performance curves of the model using the re-sampling method. The model was good in the training stage, but its performance was not stable in the validation stage, with a very high loss value, so we applied another optimizer with a less-complexity model in the next step.

4.3.2 Results Using SGD Optimizer

Figure 12 shows the results for the different metrics in the re-sampling stage using the SGD optimizer.

In this stage, we noticed that the resulting curves became smoother than in the re-sampling stage with Adam optimizer, but we need to increase the dropout ratio to minimize the difference between the training and the testing results, so we applied a model with just 3 blocks for the pre-activation ResNet model.

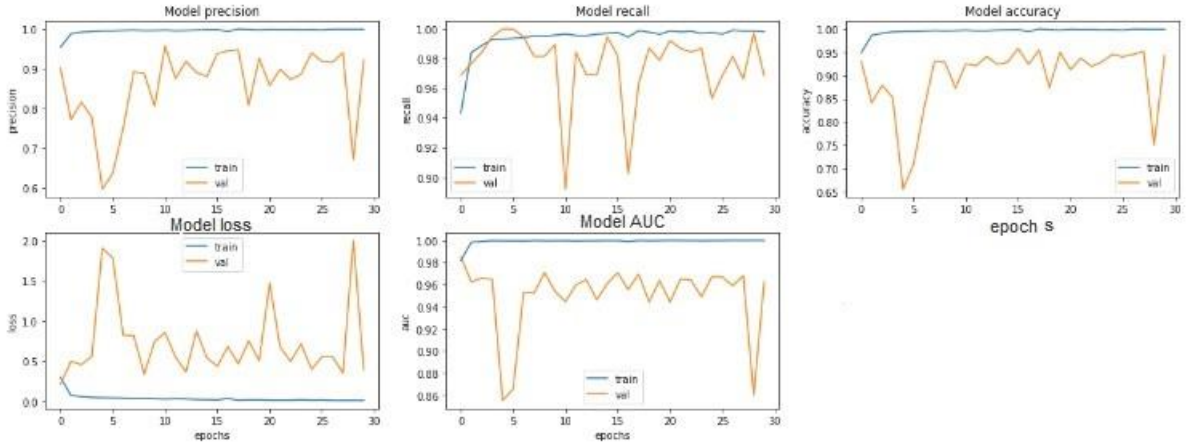

Figure 11. Results using the re-sampled dataset and the Adam optimizer.

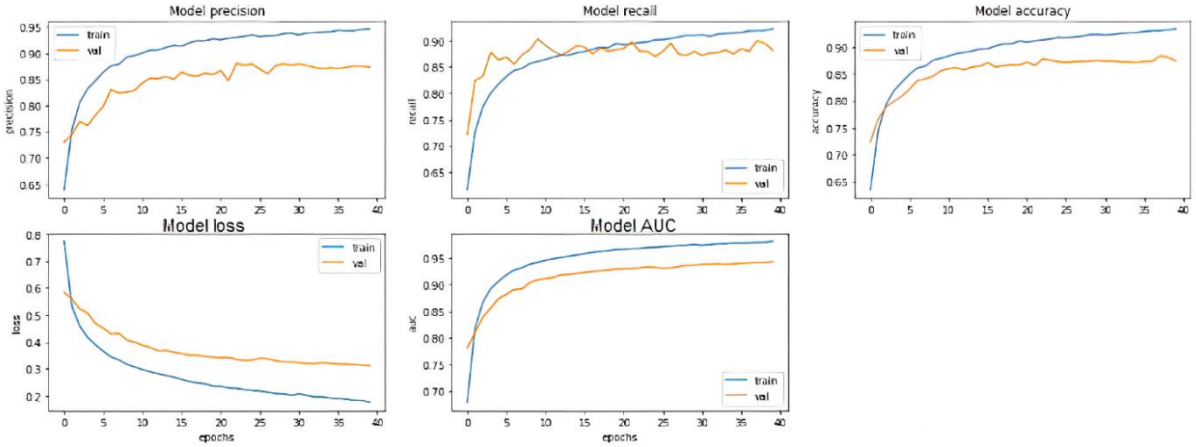

Figure 12. Results using the re-sampled dataset and the SGD optimizer.

The training stage started with a small precision value at the beginning of the training and continued to improve and rise with a steady movement until it reached a precision value of about 97%, while in the validation stage, it started with a good start and continued to improve with a slight rise. The curve formed a little zigzag and the training stopped at epoch 40 by a value of 88%.

The recall values at the training stage started with a value of about 63% and continued to rise gradually with stable movement until it reached a value of 93%, while at the validation stage, the recall started at a value of about 82%, while the shape of the curve was fluctuating by a small percentage and the recall ended at a value of 89%. The movement of the training curve was increasing as the number of epochs increases with a smooth curve stopping at an accuracy value of about 95% and the validation curve started to stabilize at approximately epoch 10 until it reached epoch 40 with an accuracy of about 88%.

The loss values at both the training and the validation stages were stable reaching a value of about 0.11at the training stage and a value of about 0.27 at the validation stage. The movement of the two curves for the AUC metric was stable and gradually increasing, with the training curve stopping at a value of about 99% and the validation curve stopping at a value of about 95%.

4.4 Results' Comparison for the Three Stages

The experiments in this study are divided into three main stages. The first stage is to train the model on the data with its original distribution and the remaining two stages deal with the weighting stage

and the resampling stage The goal in this study is to reach a good performance numerically as well as behaviourally through its behaviour in the performance curve, so that we reach a stable model as much as possible. The results are presented through the performance curves in the previous paragraph in detail and Table 8 summarizes the results for each experiment.

The model obtained a bad and unstable performance when trained on the data with its original distribution. Referring to Table 8, it shows bad numerical results, so there was a need to increase the percentage of data in the performance validation stage and using techniques to deal with the irregular distribution in performance. The goal of this study is to make a trade-off and to combine the stability of the performance curve and the numerical results to choose the best results.

Table 8. Results' comparison for the three stages.

Stage	Results	Precision	Recall	Accuracy	AUC	Loss
Original distribution stage (Adam optimizer)	Training	0.74	0.99	0.74	0.53	4.42
	Validation	0.62	0.99	0.62	0.54	6.10
Original distribution stage (SGD optimizer)	Training	0.96	0.95	0.94	0.98	0.14
	Validation	0.90	0.92	0.88	0.94	0.27
Class weight stage (Adam optimizer)	Training	0.99	0.99	0.99	0.99	0.002
	Validation	0.97	0.99	0.97	0.97	0.01
Equal class weight stage (SGD optimizer)	Training	0.95	0.93	0.92	0.97	0.19
	Validation	0.97	0.88	0.89	0.97	0.24
Different class weight stage (SGD optimizer)	Training	0.96	0.93	0.92	0.97	0.18
	Validation	0.98	0.88	0.90	0.97	0.23
Re-sampling stage (SGD optimizer and dropout of 0.8)	Training	0.974	0.939	0.957	0.992	0.115
	Validation	0.880	0.890	0.883	0.953	0.279
Re-sampling stage (SGD optimizer and dropout of 0.9)	Training	0.965	0.927	0.946	0.987	0.142
	Validation	0.876	0.907	0.888	0.948	0.292

Despite the high results of the class weight stage experiment with Adam optimizer, its performance curves were zigzag, which means that the model has no guaranteed performance. Therefore, the results of the different class weight experiment are nominated with SGD optimizer to combine good performance and acceptable shape of performance curves.

4.5 Comparison with Other Approaches

The results of the proposed approach are compared with the results of nine previous approaches which used the original distribution of the dataset proposed in [17]. This comparison is presented in Table 9. The proposed approach achieved promising results compared to other approaches, especially for the precision metric.

Table 9. Results' comparison with others approaches.

Ref.	**Algorithm**	**Accuracy**	**Recall**	**Precision**	**AUC**	**Loss**
[3]	CNN	78	-	-	-	-
[15]	ResNet-34	92	99	90	-	1.6
[27]	VGG16	87	96	-	-	0.3
	VGG19	88	95	-	-	0.3
	Inception V3	70	84	-	-	0.9
	ResNet50	77	97	-	-	0.6

[29]	ResNet152V2	99	99	99	99	-
	MobileNetV2	96	99	95	97	-
	CNN	92	92	95	96	-
	LSTM	91	92	93	95	-
[20]	Xception	-	99	84	96	0.04
[24]	Pre-trained ResNet50	90	93	93	89	0.03
[23]	VGG16	87	-	-	-	0.3
	Xception	82	-	-	-	0.45
[2]	CNN model	84	-	-	-	0.8
[37]	Xception	83	-	95	-	0.6
	DenseNet201	93	-	99	-	1.9
	MobileNet_v2	96	-	98	-	0.24
	VGG19	85	-	80	-	1.3
	CNN	84	-	94	-	0.4
	VGG16	86	-	87	-	1
	Inception_v3	94	-	93	-	1.76
	ResNet 50	96	-	98	-	1.5
	Inception_ResNet_V2	96	-	98	-	1.1
Proposed model	Pre-activation ResNet with DenseNet169	90	88	98	97	0.23

Comparing the results in terms of accuracy shows that better results are achieved compared to many previous approaches, including ResNet-34 [15], MobileNetV2, CNN and LSTM [29], the pre-trained Xception model [20], ResNet-50 [24], Xception, VGG19 and CNN [37], while the difference was clear between our results and those of other models in measuring AUC; other models include MobileNetV2, CNN, LSTM [29], the pre-trained Xception model [20], ResNet-50 [24]. The last column shows the difference between our results in the loss metric compared to ResNet-34 [15], [27] and its four algorithms [23], [37].

4.6 Results Discussion

The proposed approach was able to reach a performance that combines good results numerically and behaviourally acceptable in the performance curve. In addition, we overcame many challenges that we encountered during this study, like the small size of the test data, which appeared for the first time in the performance curve. To overcome this challenge, we repartitioned the training and validation data from scratch.

Then, we used different techniques to deal with the imbalance of data distribution; namely, the categorical weight and re-sampling techniques with the increase of data. Another challenge was the problem of overfitting, which we addressed by reducing the number of ResNet pre-activation layers and using an organization type suitable for the two techniques implemented in the previous challenge. L2 is used with the class weighing technique and the dropout is best suited with the re-sampling technique. In addition, the optimizer was changed from Adam to SGD, where the use of the SGD optimizer had a positive effect in terms of model performance stability compared to the results obtained when applying the Adam optimizer. The explanation for this phenomenon is that the SGD optimizer splits the batch number of features equal to the number of those features, making the performance movement more stable, specifically in the loss scale and thus allowing for the possibility of updating the weights for each training sample. The second conclusion is that the method for dealing with unbalanced data has a role in determining the type of regularization method that is the best, where the L2 method was the best suited one for the class weight technique, while the dropout method was the best suited for the re-sampling technique, which may be due to the volume of data used in each

stage. The results were very good on the precision, AUC and loss metrics. However, the results were modest in terms of performance on the measures of accuracy and recall. The reason according to our analysis was the inequality of weight for each class, so it may cause a conflict in the classification of the two classes. When comparing our results with those of other models, we got good results on most metrics, except for recall. Finally, we can also say that the proposed approach was able to outperform many models used in previous studies in terms of reaching stable performance and among these studies that reached fluctuating performance regardless of the results as values are [22]-[24], [27] and [44]. In addition, the results have a small loss rate and the difference was obvious compared to the two studies [26] and [2], which had an error rate of more than 1.00.

Despite that the proposed approach gave promising results compared with the state-of-the-art approaches, it has some limitations, such as:

- The used database consists of X-ray images of children between 1 and 5 years old, which may make the approach not comprehensive for other age groups.
- The process of parameter setting for the pre-activation ResNet model is performed experimentally to choose the most appropriate parameters.

5. CONCLUSIONS

AI applications have proven their efficiency in the early diagnosis of many diseases, which contributed to saving the lives of many patients. In this study, we proposed an approach by merging different deep-learning models to extract features from CXR images for the automatic detection of pneumonia disease. Two models are proposed: pre-activation ResNet and pre-trained DenseNet169. Most of the previous researchers focused on highlighting numerical results without considering the estimate of performance in the performance curve in the training and validation stages, while our focus in this research was more on combining the stability of the performance curve in addition to the numerical values of the performance measures. The proposed approach has obtained a well-stable curve compared with those obtained by many previous studies [19] and [22]. As numerical results, we achieved a precision of 98%, an AUC of 97% and a loss of 23%, while the performance was modest in terms of recall and accuracy. From the findings of the proposed approach, we conclude the following points:

- The nature and distribution of data have a significant impact on the performance of the model and each type of data has its own appropriate way to deal with.
- The type of optimizer used may increase the quality and stability of the model's performance or *vice versa*. It was evident in the results that the model became more stable when applying the SGD optimizer compared to its performance with the Adam optimizer.
- Reducing the number of layers or blocks in the pre-activation ResNet model contributes to stability and better results as well.
- The more stable the performance curve, the more assured its performance.
- Evaluating the performance of the model based on the final results of the performance measures as numbers alone is not sufficient to prove the quality and effectiveness of the model.
- The proposed approach was able to outperform those used in many previous studies, to which we dedicated a special section.
- Since the size of the proposed dataset was not sufficient to achieve satisfactory results, we had to redistribute it again. The distribution was also unbalanced and we handled it in two ways: category weighting and re-sampling as the data increased.
- The proposed approach was able to obtain a more stable performance curve than those obtained in many previous works, as detailed in the discussion section.

For future work, we suggest applying the proposed approach to data on COVID-19 pandemic and to integrate ResNet pre-activation with other CNN models.

REFERENCES

[1] V. Chouhan, S. K. Singh, A. Khamparia et al., "A Novel Transfer Learning Based Approach for Pneumonia Detection in chest X-ray Images," Applied Sciences, vol. 10, no. 2, p. 559, 2020.

[2] A. Sharma, M. Negi, A. Goyal, R. Jain and P. Nagrath, "Detection of Pneumonia Using ML & DL in Python. IOP Conference Series: Materials Science and Engineering," Proc. of the 1st Int. Conf. on Computational Research and Data Analytics (ICCRDA 2020), Rajpura, India, DOI: 10.1088/1757-899X/1022/1/012066, 2021.

[3] C. J. Saul, D. Y. Urey and C. D. Taktakoglu, "Early Diagnosis of Pneumonia with Deep Learning," ArXiv. /abs/1904.00937, DOI: 10.48550/arXiv.1904.00937, 2019.

[4] R. Sarkar, A. Hazra, K. Sadhu and P. Ghosh, "A Novel Method for Pneumonia Diagnosis from Chest X-Ray Images Using Deep Residual Learning with Separable Convolutional Networks," Proc. of the Computer Vision and Machine Intelligence in Medical Image Analysis Conf., Part of the Advances in Intelligent Systems and Computing Book Series, vol. 992, DOI: 10.1007/978-981-13-8798-2_1, 2020.

[5] A. Tilve, S. Nayak, S. Vernekar, D. Turi, P.R. Shetgaonkar and S. Aswale, "Pneumonia Detection Using Deep Learning Approaches," Proc. of the Int. Conf. on Emerging Trends in Information Technology and Engineering (ic-ETITE), pp. 1-8, DOI: 10.1109/ic-ETITE47903.2020.152, 2020.

[6] D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan and A. Mittal, "Pneumonia Detection Using CNN Based Feature Extraction," Proc. of the IEEE Int. Conf. on Electrical, Computer and Communication Technologies (ICECCT), pp. 1-7, DOI: 10.1109/ICECCT.2019.8869364, 2019.

[7] A. M. Alqudah, S. Qazan and I. S. Masad, "Artificial Intelligence Framework for Efficient Detection and Classification of Pneumonia Using Chest Radiography Images," J. Medical and Biological Eng., vol. 41, pp. 599–609, DOI: 10.1007/s40846-021-00631-1, 2021.

[8] A. K. Jaiswal et al., "Identifying Pneumonia in Chest X-rays: A Deep Learning Approach," Measurement, vol. 145, pp. 511–518. DOI: 10.1016/j.measurement.2019.05.076, 2019.

[9] S. Kido, J. Ikezoe, H. Naito, S. Tamura and S. Machi, "Fractal Analysis of Interstitial Lung Abnormalities in Chest Radiography," Radiographics, vol. 15, no. 6, pp. 1457-1464, DOI: 10.1148/radiographics.15.6.8577968, 1995.

[10] T. Ishida, S. Katsuragawa, K. Ashizawa et al., "Application of Artificial Neural Networks for Quantitative Analysis of Image Data in Chest Radiographs for Detection of Interstitial Lung Disease," Journal of Digital Imaging, vol. 11, no. 4, pp. 182–192, DOI: 10.1007/BF03178081, 1998.

[11] T. Rafael et al., "Comparative Performance Analysis of Machine Learning Classifiers in Detection of Childhood Pneumonia Using Chest Radiographs," Procedia Computer Science, vol. 18, pp. 2579-2582, DOI: 10.1016/j.procs.2013.05.444, 2013.

[12] S. Reza, O. B. Amin and M. M. A. Hashem, "A Novel Feature Extraction and Selection Technique for Chest X-ray Image View Classification," Proc. of the 5th Int. Conf. on Advances in Electrical Engineering (ICAEE), pp. 189-194, DOI: 10.1109/ICAEE48663.2019.8975457, 2019.

[13] G. Verma and S. Prakash, "Pneumonia Classification using Deep Learning in Healthcare," Int. J. of Innovative Technology and Exploring Engineering (IJITEE), vol. 9, no. 4, pp. 1715–1723, 2020.

[14] N. Ansari, A. Faizabadi, S. Motakabber and M. Ibrahimy, "Effective Pneumonia Detection Using ResNet based Transfer Learning," Test Eng. and Management, vol. 82, pp. 15146 – 15153, 2020.

[15] K. Kadam, S. Ahirrao, H. Kaur, P. Shraddha and A. Pawar, "Deep Learning Approach for Prediction of Pneumonia," Int. J. of Scientific and Technology Research, vol. 8, no. 10, pp. 2986–2989, 2019.

[16] S. Yoo, I. Gujrathi, M. A. Haider and F. Khalvati, "Prostate Cancer Detection Using Deep Convolutional Neural Networks," Scientific Reports, vol. 9, p. 19518, DOI: 10.1038/s41598-019-55972-4, 2019.

[17] D. S. Kermany et al., "Identifying Medical Diagnoses and Treatable Diseases by Image-based Deep Learning," Cell, vol. 172, no. 5, pp. 1122-1131.e9, DOI: 10.1016/j.cell.2018.02.010, 2018.

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

[19] B. Almaslukh, "A Lightweight Deep Learning-Based Pneumonia Detection Approach for Energy-Efficient Medical Systems," Wireless Communications and Mobile Computing, DOI: 10.1155/2021/5556635, 2021.

[20] J. E. Luján-García et al., "A Transfer Learning Method for Pneumonia Classification and Visualization," Applied Sciences, vol. 10, no. 8, p. 2908, DOI: 10.3390/app10082908, 2020.

[21] S. Ben Atitallah, M. Driss, W. Boulila, A. Koubaa and H. ben Ghézala, "Fusion of Convolutional Neural Networks Based on Dempster–Shafer Theory for Automatic Pneumonia Detection from Chest X-ray Images," Int. J. of Imaging Systems and Technology, vol. 32, no. 2, pp. 658–672, DOI: 10.1002/ima.22653, 2022.

[22] K. T. Islam, S. Wijewickrema, A. M. Collins and S. O'Leary, "A Deep Transfer Learning Framework for Pneumonia Detection from Chest X-ray Images," Proc. of the 15th Int. Conf. on Computer Vision Theory and Applications, pp. 286-293, DOI: 10.5220/0008927002860293, 2020.

[23] E.Ayan and H. M. Ünver, "Diagnosis of Pneumonia from Chest X-ray Images Using Deep Learning," Proc. of the 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), pp. 1-5, DOI: 10.1109/EBBT.2019.8741582, 2019.

[24] S. M. H. Hussain, S. M. Raju and A. R. Ismail, "Predicting Pneumonia and Region Detection from X-ray Images Using Deep Neural Network," arXiv: 2101.07717, DOI: 10.48550/arXiv.2101.07717, 2021.

[25] S. Singh and B. K. Tripathi, "Pneumonia Classification Using Quaternion Deep Learning," Multimedia Tools and Applications, vol. 81, pp. 1743–1764, DOI: 10.1007/s11042-021-11409-7, 2022.

[26] L. Račić, T. Popović, S. čakić and S. Šandi, "Pneumonia Detection Using Deep Learning Based on Convolutional Neural Network," Proc. of the 25th Int. Conf. on Information Technology (IT), pp. 1-4, DOI: 10.1109/IT51528.2021.9390137, 2021.

[27] R. Jain, P. Nagrath, G. Kataria, V. S Kaushik and J. Hemanth D., "Pneumonia Detection in Chest X-ray Images Using Convolutional Neural Networks and Transfer Learning," Measurement, vol. 165, p. 108046, DOI: 10.1016/j.measurement.2020.108046, 2020.

[28] C. Shorten and T. M. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," Journal of Big Data, vol. 60, Article no. 60, DOI: 10.1186/s40537-019-0197-0, 2019.

[29] N. M. Elshennawy and D. M. Ibrahim, "Deep-Pneumonia Framework Using Deep Learning Models Based on Chest X-ray Images," Diagnostics, vol. 10, no. 9, p. 649, 2020.

[30] K. He, X. Zhang, S. Ren and J. Sun, "Identity Mappings in Deep Residual Networks," Proc. of European Conf. on Computer Vision (ECCV 2016), Part of the Lecture Notes in Computer Science, vol. 9908, DOI: 10.1007/978-3-319-46493-0_38, 2016.

[31] K. He, X. Zhang, Sh. Ren and J. Sun, "Deep Residual Learning for Image Recognition," Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 770-778. [Online], Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7780459, 2016.

[32] W. Shang, J. Chiu and K. Sohn, "Exploring Normalization in Deep Residual Networks with Concatenated Rectified Linear Units," Proc. of the 31st AAAI Conference on Artificial Intelligence (AAAI-17), vol. 31, no. 1, pp. 509-1516, DOI: 10.1609/aaai.v31i1.10759, 2017.

[33] J. Bjorck, C. Gomes, B. Selman and K. Q. Weinberger, "Understanding Batch Normalization," arXiv: 1806.02375, DOI: 10.48550/arXiv.1806.02375, 2018.

[34] A.Khan, A.Sohail, U. Zahoora and A. S. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," Artificial Intelligence Review, vol. 53, pp. 5455-5516, DOI: 10.1007/s10462-020-09825-6, 2020.

[35] A. Desarda, "Build a Custom ResNetV2 with the Desired Depth from Scratch," Towards Data Science, [Online], Available: https://towardsdatascience.com/build-a-custom-resnetv2-with-the-desired-depth-92892ec79d4b, 2020, Last Accessed 31/10/2022.

[36] G. Huang, Z. Liu, L. Van der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 2261-2269, DOI: 10.1109/CVPR.2017.243, 2017.

[37] R. Kundu, R. Das, Z.W. Geem, G-T Han and R. Sarkar, "Pneumonia Detection in Chest X-ray Images Using an Ensemble of Deep Learning Models," PLoS ONE, vol. 16, no. 9, p. e0256630, DOI: 10.1371/journal.pone.0256630, 2021.

[38] K. El Asnaoui, Y. Chawki and A. Idri, "Automated Methods for Detection and Classification Pneumonia Based on X-Ray Images Using Deep Learning," Proc. of the Artificial Intelligence and Blockchain for Future Cybersecurity Applications, Part of the Studies in Big Data Book Series, vol. 90, DOI:10.1007/978-3-030-74575-2_14, 2021.

[39] Q. Wu and F. Wang, "Concatenate Convolutional Neural Networks for Non-intrusive Load Monitoring across Complex Background," Energies, vol. 12, no. 8, p. 1572, DOI: 10.3390/en12081572, 2019.

[40] J. Brownlee, "How to Choose an Activation Function for Deep Learning," Machine Learning Mastery, [Online], Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/, 2021, Last Accessed 31/10/2022.

[41] J. Collis, "Glossary of Deep Learning: Bias," Deeper Learning, [Online], Available https://medium.com/deeper-learning/glossary-of-deep-learning-bias-cf49d9c895e2, 2017, Last Accessed on 31/10/2022.

[42] P. Mooney, "Chest X-ray Images (Pneumonia)," Kaggle, [Online], Available: www.kaggle.com/paultimothymooney/chest-xray-pneumonia, 2018, Last Accessed on 31/10/2022.

[43] M. Vakili, M. Ghamsari and M. Rezaei, "Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification," arXiv: 2001.09636v1, DOI: 10.48550/arXiv.2001.09636, 2020.

[44] G. Labhane, R. Pansare, S. Maheshwari, R. Tiwari and A. Shukla, "Detection of Pediatric Pneumonia from Chest X-ray Images using CNN and Transfer Learning," Proc. of the 3rd Int. Conf. on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), pp. 85-92, DOI: 10.1109/ICETCE48199.2020.9091755, 2020.

[45] A. Mabrouk, R.P. Díaz Redondo, A. Dahou, M. Abd Elaziz and M. Kayed, "Pneumonia Detection on Chest X-ray Images Using Ensemble of Deep Convolutional Neural Networks," Applied Sciences, vol.

12, p. 6448, DOI: 10.3390/app12136448, 2022.
[46] T. Rahman et al., "Transfer Learning with Deep Convolutional Neural Network (CNN) for Pneumonia Detection Using Chest X-ray," Applied Sciences, vol. 10, p. 3233, DOI: 10.3390/app10093233, 2020.
[47] S. Pallawi and D. Kumar Singh, "Preview and Analysis of Deep Neural Network for Alzheimer's Disease Classification using Brain Medical Resonance Imaging," Cognitive Computation and Systems, vol. 5, no. 1, pp. 1-13, DOI: 10.1049/ccs2.12072, 2022.
[48] D. K. Jain, T. Singh, P. Saurabh, D. Bisen, N. Sahu, J. Mishra and H. Rahman, "Deep Learning-aided Automated Pneumonia Detection and Classification Using CXR Scans," Computational Intelligence Neuroscience, vol. 2022, Article ID 7474304, DOI: 10.1155/2022/7474304, 2022.
[49] M. Trivedi and A. Gupta, "A Lightweight Deep Learning Architecture for the Automatic Detection of Pneumonia Using Chest X-ray Images," Multimedia Tools Applications, vol. 81, pp. 5515–5536, DOI: 10.1007/s11042-021-11807-x, 2022.
[50] P. Szepesi and L. Szilágyi, "Detection of Pneumonia Using Convolutional Neural Networks and Deep Learning," Biocybernetics and Biomedical Engineering, vol. 42, no. 3, pp. 1012-1022, DOI: 10.1016/j.bbe.2022.08.001, 2022.

ملخص البحث:

يُعدّ مرض ذات الرّئة من الأمراض المهدّدةِ للحياة، حيث يمكن للكشْف المبكّر عنه إنْقاذ الأرواح. وهناك العديد من الأنظمة الأوتوماتيكية التي ساهمت في الكشْف عن الإصابة بهذا المرض، وأصبحت النّماذج القائمة على التّعلُّم العميق أوسع النّماذج انتشاراً التي تُستخدم في بناء أنظمة الكشْف هذه.

في هذه الدراسة يتمّ جمْع نموذجين من نماذج التعلُّم العميق واستخدامها في الكشْف عن مرض ذات الرّئة. وقد تمّ استخدام طريقتين للتّعامل مع مشكلة عدم توازُن البيانات، وهما: وَزْن البيانات بحيث تمكّن هذه الطّريقة من التحكُّم في النِّسبة المئوية من البيانات الأصلية التي يُمكن استخدامها لكلّ صِنْفٍ من البيانات، وإعادة أخْذ العيّنات وفيها يتمّ إنتاج صورٍ معدَّلة ذات توزيعٍ متساوٍ للبيانات.

تم تقييم النّموذج المقترح باستخدام مجموعة بيانات متوازنة تتكون من (5856) صورة. وكانت النّتائج واعِدَة مقارنةً بنماذج استُخدمت في دراساتٍ سابقة؛ إذ كانت نسبة الضَّبْط (98%)، والمساحة تحت المنحنى (97%)، بينما بلغت قيمة الفَقْد (0.23).