Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/67/20240612

Pneumonia image classification using convolutional neural network

Zhili He

Digital Media Technology, University of Electronic Science and Technology of China, Chengdu, 611731, China

2021080911026@std.uestc.edu.cn

Abstract. Throughout history, humanity has continually faced numerous unforeseen health crises, thereby emphasizing the utmost significance of harnessing cutting-edge technology and scientific approaches to proactively anticipate and tackle potential outbreaks. In the context of the COVID-19 pandemic, the utilization of CT scans has proven to be an indispensable asset in accurately detecting and diagnosing individuals afflicted with COVID-19. By virtue of their advanced imaging capabilities and ability to capture detailed internal images, CT scans have emerged as a pivotal diagnostic tool in the fight against this infectious disease. The objective of this study is to improve the precision and effectiveness of COVID-19 diagnosis by employing a convolutional neural network (CNN) model architecture specifically tailored for the analysis of CT images. We conducted a comprehensive analysis of the current body of literature to investigate the clinical features, imaging manifestations, and image-based diagnostic methods of COVID-19. This study examines the application of deep learning models in the diagnosis of COVID-19, specifically emphasizing the utilization of Convolutional Neural Networks (CNNs) as a robust method for analyzing medical images. In addition, we demonstrate experimental enhancements to the CNN model, achieving a diagnosis accuracy of up to 92.80% when evaluated on test data.

Keywords: COVID-19 diagnosis, CNN, medical image analysis, deep learning, experimental improvements.

1. Introduction

Despite the current global status of COVID-19 resembling that of a typical influenza, it is imperative to recognize the potential for unforeseen challenges that may arise in the future and to take proactive measures to enhance our preparedness for such circumstances. The course of history has witnessed numerous unanticipated health crises, thereby emphasizing the significance of harnessing state-of-the-art technology and scientific methodologies to preemptively identify and address potential epidemics. Notably, during the COVID-19 pandemic, the emergence of CT scans as invaluable tools for diagnosing COVID-19 infection has significantly contributed to the medical field [1]. By providing intricate details of lung lesions, these scans have facilitated physicians in rendering accurate and timely diagnoses. However, it is important to acknowledge that the analysis of medical imaging data can present certain challenges due to its intricate and diverse nature. Conventional methods of analysis may encounter limitations when confronted with the complexity of these images. Therefore, it becomes imperative to

DOI: 10.54254/2755-2721/67/20240612

explore innovative approaches that can overcome these limitations and enhance the accuracy and efficiency of medical image analysis. By leveraging advancements in artificial intelligence (AI) and machine learning, researchers and healthcare professionals can develop sophisticated algorithms capable of extracting meaningful insights from medical images. These AI-driven analysis techniques have the potential to revolutionize the field of medical imaging by improving diagnostic accuracy, expediting treatment decisions, and ultimately enhancing patient outcomes. This paper endeavors to enhance the precision and efficiency of diagnosing new coronavirus pneumonia (COVID-19) by employing a specialized Convolutional Neural Network (CNN) model architecture tailored for CT image analysis. By harnessing the power of deep learning technology and training on a vast repository of CT scan images, this approach seeks to elevate diagnostic accuracy and efficiency, empowering us to better navigate potential health crises in the future.

2. Literature Review

2.1. Clinical Characteristics and Imaging Manifestations of COVID-19

Since the inception of the COVID-19 pandemic, there has been a comprehensive exploration into its clinical features and imaging manifestations. The clinical presentation of COVID-19 commonly includes symptoms such as fever, cough, and dyspnea, which serve as primary indicators for diagnosis and management [2]. Additionally, imaging studies have revealed characteristic findings on chest radiographs and computed tomography (CT) scans, which play a crucial role in identifying and monitoring the disease progression.

In terms of imaging manifestations, ground-glass opacities (GGOs), consolidations, and bilateral involvement of lung parenchyma are frequently observed in COVID-19 patients [3]. GGOs, which appear as hazy areas with increased opacity but without obscuration of the underlying vessels, are considered hallmark features of COVID-19 pneumonia [4]. Consolidations, on the other hand, represent areas of lung tissue that have become filled with inflammatory exudates or fluid, leading to increased density on imaging.

In summary, the clinical features and imaging manifestations of COVID-19, including fever, cough, dyspnea, and characteristic radiographic findings such as GGOs, consolidations, and bilateral lung involvement, are indispensable in the comprehensive evaluation and management of patients affected by the disease.

2.2. Review of Image-based Diagnostic Methods for COVID-19

A plethora of image-based diagnostic methods have been proposed and scrutinized for their utility in diagnosing COVID-19. These methods span across diverse techniques, ranging from conventional image analysis algorithms to machine learning strategies, and advanced deep learning models. Researchers have meticulously examined the effectiveness of these approaches in identifying COVID-19-associated abnormalities in chest CT scans and X-ray images, underscoring their capacity for precise and effective diagnosis.

Traditionally, image analysis algorithms have been utilized for the detection and characterization of abnormalities in medical images. These algorithms leverage various image processing techniques, such as segmentation and feature extraction, to identify patterns indicative of COVID-19 infection. While these methods have demonstrated utility, they may be limited by their reliance on predefined rules and features, potentially hindering their adaptability to diverse imaging datasets [5].

Machine learning approaches offer a more flexible and data-driven alternative for COVID-19 diagnosis. By training on annotated image datasets, machine learning models can learn to distinguish between normal and abnormal imaging patterns associated with COVID-19. Techniques such as support vector machines (SVM), random forests, and logistic regression have been explored for their efficacy in this context, showcasing promising results in terms of diagnostic accuracy and efficiency [6].

In recent years, deep learning models, particularly Convolutional Neural Networks (CNNs), have emerged as powerful tools for medical image analysis. CNNs excel at automatically extracting

DOI: 10.54254/2755-2721/67/20240612

hierarchical features from raw image data, enabling them to effectively discern subtle abnormalities indicative of COVID-19 infection. Studies have demonstrated the superiority of deep learning models in COVID-19 diagnosis, showcasing their ability to outperform traditional machine learning approaches in terms of both accuracy and speed [7].

In conclusion, the field of image-based diagnostic methods for COVID-19 is characterized by a diverse array of techniques, ranging from traditional image analysis algorithms to advanced deep learning models. These methods hold considerable promise for enhancing the accuracy and efficiency of COVID-19 diagnosis, thus facilitating timely and effective management of the disease.

2.3. Basic Principles and Applications of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have emerged as a cornerstone in medical image analysis, owing to their remarkable proficiency in feature extraction and classification tasks. Comprising multiple layers, CNNs possess a hierarchical architecture that facilitates the automatic learning of intricate image representations from input data. These layers typically include convolutional layers, pooling layers, and fully connected layers, each playing a distinct role in the feature extraction process [8].

In CNNs, convolutional layers act as feature extractors by applying a series of filters to the input image, thereby detecting spatial patterns and features at different scales. These filters, also known as kernels, convolve across the input image, generating feature maps that highlight relevant patterns indicative of disease pathology. Subsequently, pooling layers are employed to downsample the feature maps, reducing their spatial dimensions while preserving essential features. This process helps in mitigating computational complexity and enhancing the network's robustness to spatial variations [9].

Furthermore, fully connected layers serve as classifiers, integrating the extracted features to make predictions regarding the presence or absence of a particular condition. Through the process of forward propagation, CNNs can iteratively refine their parameters to optimize performance on the given task, such as COVID-19 diagnosis from chest CT scans.

The versatility and effectiveness of CNNs have led to their widespread adoption in various medical imaging applications, including the detection and classification of COVID-19-related abnormalities on chest CT scans. By leveraging the hierarchical representations learned from vast datasets, CNNs demonstrate superior performance in discerning subtle imaging features associated with COVID-19 infection, thereby aiding clinicians in accurate diagnosis and treatment planning [10].

3. Methodology

3.1. Dataset

In this research, a retrospective cohort of pediatric patients aged one to five years old from Guangzhou Women and Children's Medical Center in Guangzhou was selected to obtain chest X-ray pictures (anterior-posterior). Thoracic X-ray imaging was conducted as a component of patients' routine clinical care. Sample images from these classes are given in Fig.1.

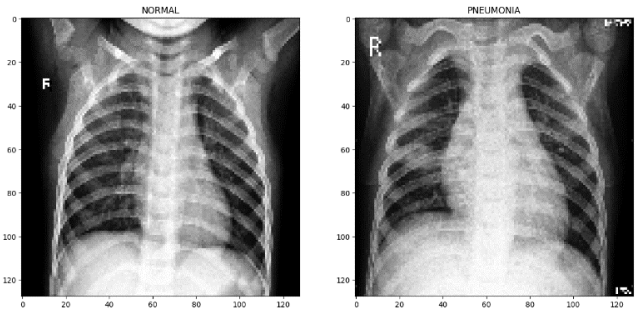

Figure 1. Images and CT images of pneumonia patients in the training set

DOI: 10.54254/2755-2721/67/20240612

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are total 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). In the training set, there are a total of 1341 images labeled as normal and 3875 images labeled as pneumonia, accounting for 25.7% and 74.3% of the total, respectively. At the same time, in the test set, there are 234 normal labeled images accounting for 37.5% and 390 pneumonia labeled images accounting for 62.5%(Fig.2).

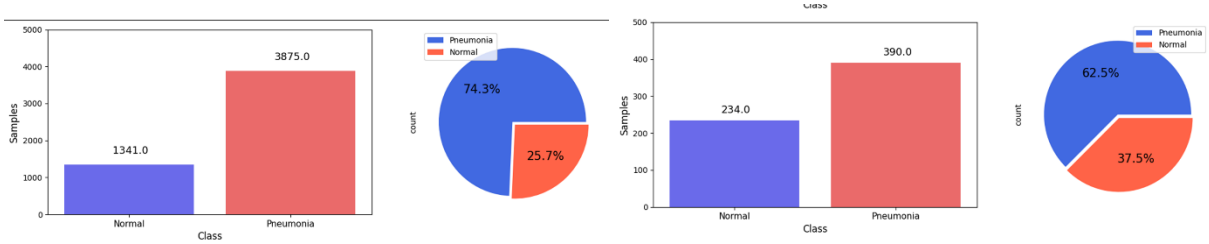

Figure 2. Data set distribution proportion

3.2. Data and Image pre-processing

During the training process of convolutional neural networks, data quality and data distribution have a huge impact on the performance obtained from training. As long as the data performance is good enough, the effect will not be too bad even without too much training. First of all, when the author counted the image information, it found that the resolution of all training images was not the same and most of the images were too large. This will cause the parameters or calculation amount of the model to increase accordingly. The author applied resize() method in the cv2 library to reshape all images to 224*224, which is the most classic size for deep learning input images [11]. You can intuitively feel the changes before and after image reshaping in Fig.3.

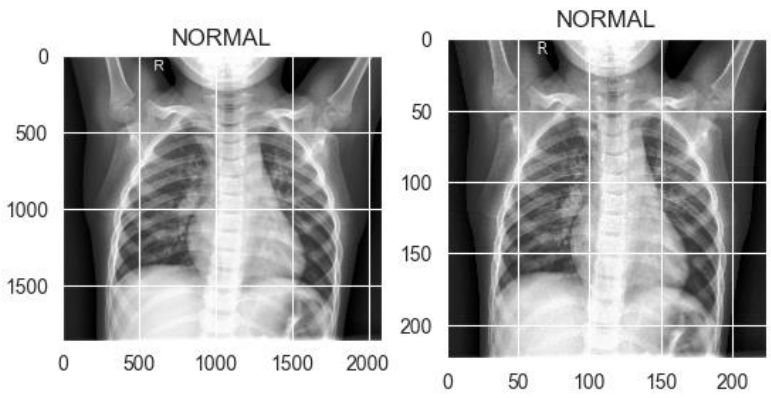

Figure 3. The 2090*1898 image is scaled to 224*224

By observing the histogram of the image, we can clearly understand the characteristics of the training image and perform preprocessing to achieve a more ideal training effect. Histogram equalization is a method in the field of image processing that uses image histograms to adjust contrast, and is a means of image enhancement. Grayscale inversion refers to linearly or nonlinearly inverting the grayscale range of an image to produce an image with the opposite grayscale of the input image. The lungs appear black in the CT image, so grayscale inversion is performed to observe the transformation effect. Here are the images obtained by performing the above-mentioned processing on the data image and their histogram distribution shown in Fig.4 [12-13].

DOI: 10.54254/2755-2721/67/20240612

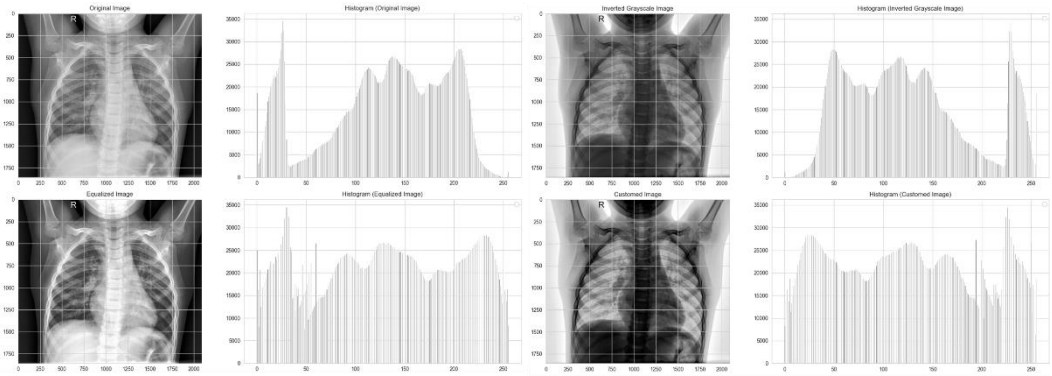

Figure 4. From top to bottom, they are 1 original image, 2 histogram equalization, 3 grayscale inversion, 4 grayscale inversion + histogram equalization.

After processing the training images themselves, our focus will now shift towards the data distribution. In the training set, images with label NORMAL only account for 25.7%, which may lead to data imbalance. The so-called imbalance means that the sample sizes of different categories are very different, or a small number of samples represent key data, and good learning of the patterns of a small number of samples is required. The presence of an imbalanced sample distribution can lead to a classification task where the short sample size contains an insufficient number of features, hence impeding the extraction of patterns. Furthermore, even if a classification model is successfully developed, it is susceptible to over-reliance on the limited data samples, so giving rise to issues related to over-fitting. The model's accuracy and robustness will be subpar when applied to new data. Fig.5 is the Confusion Matrix obtained without any preprocessing training on the data set. It shows that a large number of images with NORMEL labels are judged to be pneumonia label images. So, the author initially determined that this was caused by unbalanced sample distribution. In order to make up for the defects caused by unbalanced sample distribution and prevent overfitting, the author used ImageDataGenerator, which is a useful tool to flip and shift the images randomly, to perform data augmentation on the training set [14-15].

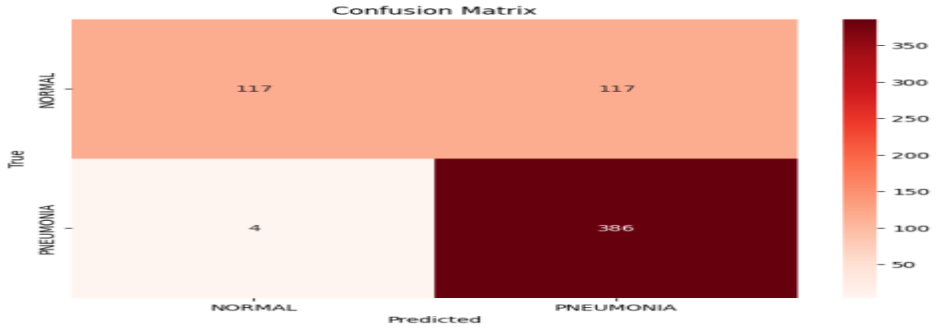

Figure 5. Without any processing on the data set, the confusion matrix obtained

3.3. Convolutional Neural Network Model Structure

The model proposed in this study is a convolutional neural network (CNN) [16], which is used for CT image classification tasks. The following is an overview of each layer of the model, so does TABLE 1:

1) Input layer: accepts a grayscale image of 224x224 size as input shaped in (224,224,1).

2) Convolution layer (Conv2D): The first convolution layer contains 32 convolution kernels, each convolution kernel size is 3x3, and uses the ReLU activation function for feature extraction. Convolutional layers are a type of neural network layer commonly used in deep learning to extract features from input data. In the convolutional layer, the model learns local features of image or sequence data by applying convolution operations.

DOI: 10.54254/2755-2721/67/20240612

Convolution is a linear operation that produces output features by sliding a narrow window, known as a convolution kernel or filter, over the input data (figure 6). This process involves multiplying the convolution kernel with a particular area of the data and then summing the results. This process can be understood as filtering the input data with a convolution kernel and extracting feature information at different locations.

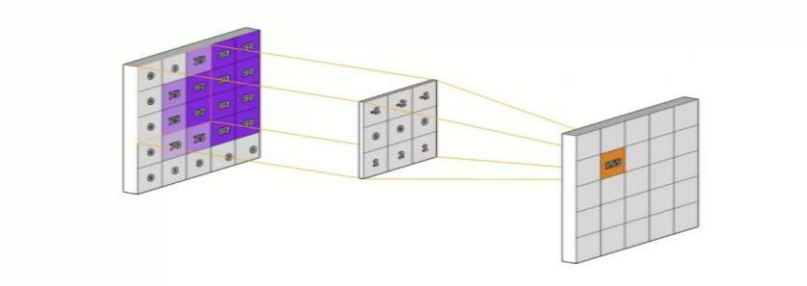

Figure 6. The role of the convolutional layer is to extract useful features from the input data, allowing the model to learn the local pattern and structural information of the data.

3) Batch Normalization layer (BatchNormalization): Standardizing the output of each batch helps accelerate convergence and stabilize the training process. Batch Normalization speeds up the training process by normalizing each input so that the distribution of each feature is kept within a smaller range [17].

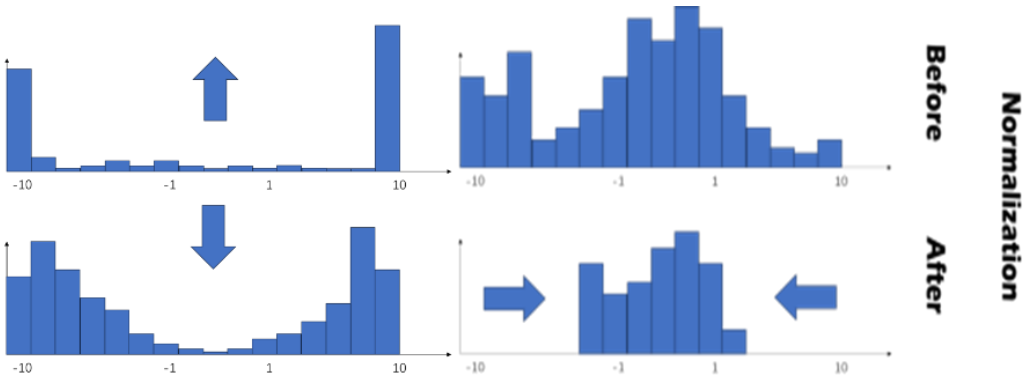

Figure 7. example of tanh activation function with Batch Normalization

4) Maximum pooling layer (MaxPooling2D): Use a 2x2 pooling window to perform maximum pooling operations to reduce the size of the feature map. Pooling Layer is a type of neural network layer commonly used in deep learning. It is used to reduce the size of feature maps, reduce the amount of calculation, and enhance the model's learning ability for translation invariance.

The pooling layer reduces the dimensionality of the feature map by pooling local areas of the input data. Commonly used pooling operations include Max Pooling and Average Pooling. In max pooling, the pooling operation selects the maximum value in the local area as the output; while in average pooling, the pooling operation selects the average value of the local area as the output. You can see the principle of the pooling layer more intuitively in Fig.8.

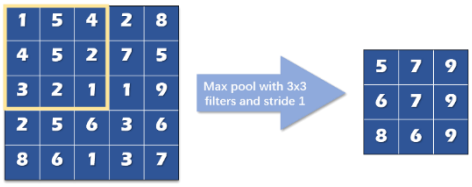

Figure 8. An example of the use of Max Pooling.

DOI: 10.54254/2755-2721/67/20240612

5) Dropout: Randomly discard some neurons to prevent overfitting [18].

6) Fully connected layer (Dense): Contains 128 neurons and uses the ReLU activation function for nonlinear transformation. The Fully Connected Layer, alternatively referred to as the Dense Layer, is a prevalent layer category inside neural networks. A completely connected layer consists of neurons that are connected to all neurons in the preceding layer. Each connection is assigned a weight, which is acquired through the training procedure. The function of the fully connected layer is to perform non-linear mapping and combination of input features to obtain higher-level feature representation. It will show you how fully connected layer classify Pneumonia in Fig.9.

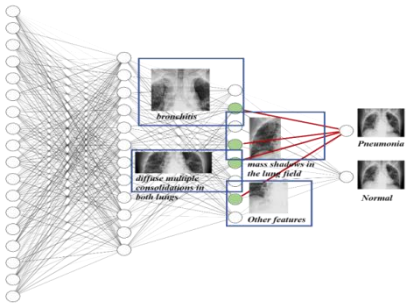

Figure 9. The fully connected layer adjusts the weights during training based on the image features of different parts of the kitten. When the final image is input, the probability of being Pneumonia is predicted based on the weights.

7) Output layer (Dense): Contains 1 neuron, uses the Sigmoid activation function for binary classification output, and outputs 0 or 1 to indicate whether it belongs to a certain category.

The model has a total of 14 layers, including 4 convolutional layers, 4 pooling layers, 4 batch normalization layers, 2 dropout layers and 2 fully connected layers. The reasonable combination of these layers enables the model to have good feature extraction capabilities and generalization capabilities, making it suitable for image classification tasks. By adding batch normalization and dropout layers, the generalization ability and robustness of the model are improved.

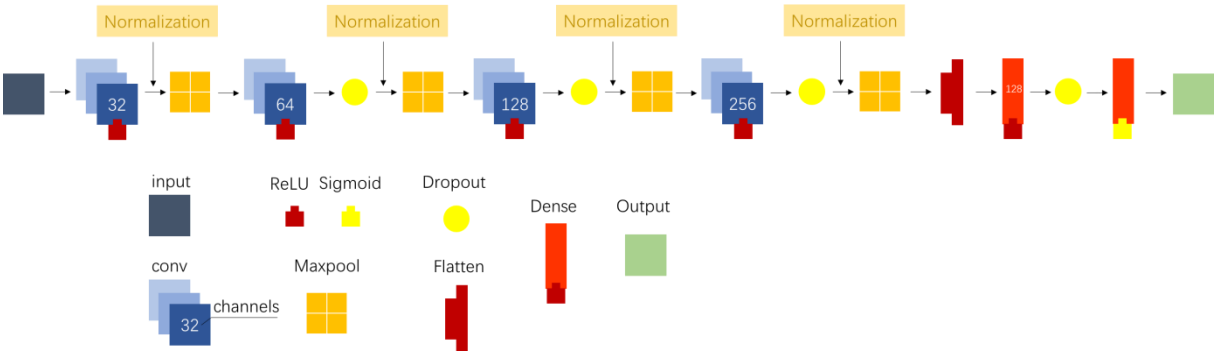

Figure 10. Overview of Each Layer of The Model

3.4. Model Compilation

Model compilation is like a builder to a delicate house. After building the model, we need to compile the model for training. Only by continuously converging and improving parameters can we achieve the best results. The model.compile() function is one of the APIs provided by the Keras framework. Its emergence can be traced back to the early days of the deep learning framework (about 2015). The initial version of Keras is a high-level neural network API and can run on multiple deep learning backends (such as TensorFlow, Theano, etc.). As Keras continues to grow in popularity and growth in the deep learning community, people are starting to make more use of Keras itself and its default backend TensorFlow.

DOI: 10.54254/2755-2721/67/20240612

The model.compile() function is designed as a compiler to link the model's graph structure definition with the calculation engine to enable optimization, loss function selection, and configuration of the training process.

3.4.1. Optimizer. The optimizer's primary purpose is to modify and compute the network parameters that impact both model training and model output. This is done with the aim of achieving the optimal value, which is determined by minimizing or maximizing the loss function. The majority of widely used optimizers in deep learning rely on gradient descent. This implies that they iteratively calculate the gradient of a specified loss function L and adjust the parameters in the opposite direction (thus descending towards an imagined global minimum).

Which optimizer is best? Which optimization algorithm should be chosen? No consensus has yet been reached. Currently, the most popular and highly used optimizers (algorithms) include SGD, SGD with momentum, RMSprop, RMSProp with momentum, AdaDelta, and Adam [19]. After experiments, the best optimizer for this study is Nadam (Statistics Shown in TABLE 2,TABLE 3.) [20]. The parameter update formula is as follows:

$$g_t = \frac{\partial(\text{Loss})}{\partial(\theta_t)} \tag{1}$$

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t} \tag{2}$$

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

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}}+\epsilon}\left(\beta_1 \widehat{m_t} + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right) \tag{4}$$

In Formula 1, g_t represents the gradient of the loss function Loss with respect to the parameter θ_t.This serves as the basis for updating parameters in optimization algorithms, indicating the direction of the gradient at the current parameter values. In Formula 2, $\widehat{m_t}$ is the corrected first moment estimate m_t. By dividing by $1-\beta_1^t$, it corrects the bias in the first moment estimate, making the estimation of gradients more accurate especially during early training stages. In Formula 3, m_t is the first moment estimate, representing the exponentially weighted average of past gradients. This equation updates the first moment estimate using exponential moving average, where β_1 controls the decay rate of gradients. Finally in formula 4, θ_{t+1} denotes the parameter value at the next time step $t+1$. This equation describes the parameter update process, where η is the learning rate controlling the step size of parameter updates. $\widehat{v_t}$ represents the corrected second moment estimate, calculated as the exponentially weighted average of past squared gradients divided by the bias-corrected term. ϵ is a small constant added for numerical stability. The entire equation involves a weighted average of the first moment estimate $\widehat{m_t}$ and the second moment estimate $\widehat{v_t}$, adjusted by the learning rate for parameter updates.

These equations constitute the core update process of the NAdam algorithm, aiming to minimize the loss function and update parameters iteratively.

In a nutshell, the Nadam optimizer is an optimizer that combines Nesterov momentum and adaptive learning rate methods, which can provide faster convergence to local optimal solutions or global optimal solutions when training deep learning models.

Table 1. Nadam Performs best both in test_accuracy and test_loss

optimizer	*test_accuracy*	*test_loss*	*optimizer*	test_accurancy	test_loss
Nadam	0.9263	0.2109	*Nadam*	0.9263	0.2109
RMSprop	0.9103	0.5744	*SGD*	0.8974	0.2821
Adam	0.899	0.2997	*Adam*	0.899	0.2997
SGD	0.8974	0.2821	*AdaGrad*	0.859	0.3266
AdaGrad	0.859	0.3266	*Adadelta*	0.774	0.4621
Adadelta	0.774	0.4621	*RMSprop*	0.9103	0.5744

DOI: 10.54254/2755-2721/67/20240612

3.4.2. Loss Function. The loss function is an operating function used to measure the difference between the model's predicted value $f(x)$ and the true value Y. It is a non-negative real-valued function, usually using $L(Y, f(x))$. This suggests that a smaller loss function is indicative of a higher level of model resilience. Because the pneumonia CT image recognition discussed in this paper is a binary recognition problem, the loss function we choose is naturally 'binary_crossentropy' [21]. The algorithm of binary cross entropy:

$$L(Y, f(x)) = -(Y \cdot \log(f(x)) + (1 - Y) \cdot \log(1 - f(x))) \tag{5}$$

Binary cross-entropy loss is a metric used to evaluate the performance of binary classification models. It measures the discrepancy between predicted probabilities $f(x)$ and actual binary labels Y.

When the true label Y is 1, the loss decreases as the predicted probability $f(x)$ approaches 1, indicating a confident prediction of the positive class. Conversely, if the predicted probability approaches 0 when the true label is 1, the loss increases significantly. This behavior is aligned with the logarithmic nature of the loss function.

In essence, binary cross-entropy loss encourages the model to make confident and accurate predictions, penalizing overly confident incorrect predictions and rewarding confident correct predictions.

3.4.3. Callbacks. Callbacks are a powerful tool for customizing the behavior of Keras models during training, evaluation, or inference. Examples include tf.keras.callbacks.TensorBoard, which uses TensorBoard to present training progress and results, and tf.keras.callbacks.ModelCheckpoint, which periodically saves the model during training.

4. Result and evaluation

In this section, we present the results and evaluation of the convolutional neural network (CNN) model we designed and trained for pneumonia detection. We first show the performance of the model on the test data set, including evaluation indicators such as accuracy, recall, and precision [22]. Finally, we evaluate the specific performance of the model through the F1 score and confusion matrix [23].

4.1. Accuracy, Recall, Precision and Specificity

Accuracy is the most commonly used and most intuitive performance indicator, and its definition is as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+TN+FP} \tag{6}$$

The recall rate can be defined as the ratio of correctly predicted positive samples to the total number of samples that are really positive:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{7}$$

The precision rate represents: among all samples classified as positive examples, the proportion of true positive examples is defined as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{8}$$

The semantics of specificity are: among the actual negative samples, how likely it is to be predicted. This definition is very similar to recall rate. The only difference between the two is that the objects are different. Recall rate is for positive examples, while specificity is for negative examples. .

$$\text{Specificity} = \frac{TN}{FP+TN} \tag{9}$$

Most comprehensively, the experimental data shows quite good results. The accuracy rate reached 92.79%, indicating that the model performed well in correctly classifying all samples. The recall rate is

DOI: 10.54254/2755-2721/67/20240612

95.89%, which means that the model successfully captures the majority of samples that are actually positive examples. The accuracy rate reached 92.80%, indicating that the model showed high accuracy in correctly classifying positive samples. Finally, the specificity is 87.60%, which shows that the model also indicates a high ability to correctly classify negative samples. Taking these indicators together, the results of the experimental data are relatively ideal.

4.2. Confusion Matrix

The confusion matrix provides a detailed view of the prediction accuracy of the classification model on different categories. By comparing the actual categories and model-predicted categories, indicators such as accuracy, recall, precision, and F1 score of the model can be calculated to comprehensively evaluate model performance (Fig.11).

$$F1_{Score} = \frac{2*Precision*Recall}{Precision+Recall} \tag{10}$$

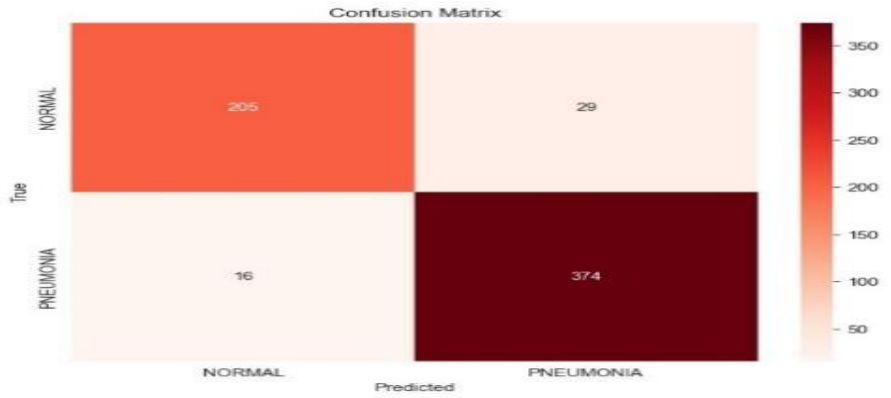

Figure 11. the Confusion matrix of the model

The F1-Score allows for the simultaneous evaluation of the model's misdiagnosis rate and screening rate by considering both precision and recall. The model's F1 score after evaluation is around 0.9432. It shows that the model performs well in maximizing precision and recall at the same time, and has high classification accuracy.

4.3. Evaluation

After training the model, a line chart is generated based on the accuracy and loss obtained for each batch (fig.12). We can clearly see that train_accuracy and train_loss gradually converge. However, when observing validation_accuracy, there are constant fluctuations. It is initially speculated that it is due to too little data in the validation set.

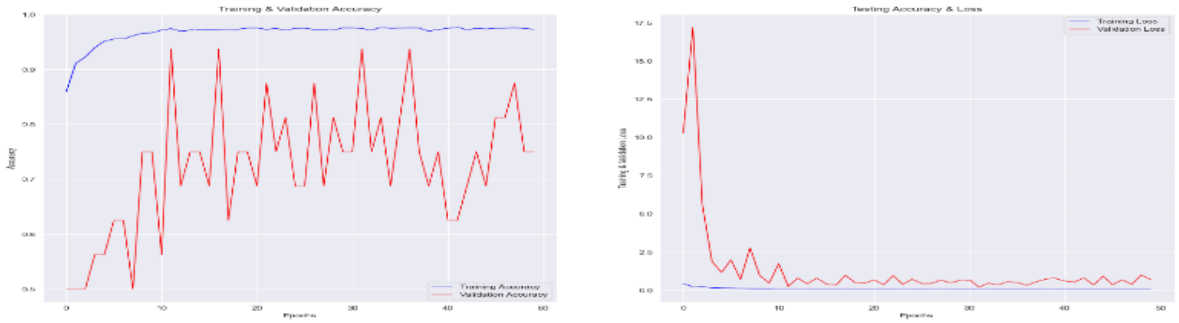

Figure 12. The variation curve of accuracy and loss

DOI: 10.54254/2755-2721/67/20240612

5. Conclusion

In this study, we successfully improved the performance of CNN models through methods such as augmenting and balancing the dataset, image enhancement, batch normalization, and increasing the depth of the model through the stacking of multiple convolutional layers. However, we are also aware of some shortcomings. First, we did not compare with other deep learning models, which limits a comprehensive evaluation of the model performance. Secondly, there is still room for further improvement in the research, especially in distinguishing bacterial infections from viral infections, which we can explore more deeply. In addition, due to the difficulty of manual labeling of data set labels, we propose a direction for future improvement, which is to build a student-teacher deep learning model and implement automatic labeling functions to improve the efficiency and accuracy of data set labeling. Looking forward, with the continuous development of deep learning and medical imaging technology, we are expected to shift this research focus to incorporating the EfficientNet model into our diagnostic framework [24]. EfficientNet has an efficient and lightweight architecture, making it particularly suitable for deployment on embedded devices and portable medical diagnostic instruments. By integrating EfficientNet into our diagnostic framework, we expect to achieve improvements in sensitivity and specificity in COVID-19 diagnosis, providing more accurate and reliable results. In addition, the simplification and miniaturization of medical equipment can be achieved using EfficientNet, which is expected to promote the development of portable home medical diagnostic equipment. This development will help individuals monitor and manage health conditions in a timely manner and promote disease prevention and early detection.

References

[1] Lee, Elaine YP, Ming-Yen Ng, and Pek-Lan Khong. (2020). "COVID-19 pneumonia: what has CT taught us?." The Lancet Infectious Diseases, 20.4 : 384-385.

[2] Guo, Yan-Rong, et al. (2020). The origin, transmission and clinical therapies on coronavirus disease 2019 (COVID-19) outbreak–an update on the status. Military medical research, 7: 1-10.

[3] Wong, Ho Yuen Frank, et al. (2020). Frequency and distribution of chest radiographic findings in patients positive for COVID-19. Radiology, 296.2: E72-E78.

[4] Zu, Zi Yue, et al. (2020). Coronavirus disease 2019 (COVID-19): a perspective from China. Radiology 296.2: E15-E25.

[5] Wynants, Laure, et al. (2020). Prediction models for diagnosis and prognosis of covid-19: systematic review and critical appraisal. BMJ: 369.

[6] Li, Lin, et al. (2020). Using artificial intelligence to detect COVID-19 and community-acquired pneumonia based on pulmonary CT: evaluation of the diagnostic accuracy. Radiology, 296.2 : E65-E71.

[7] Wang, Shuai, et al. (2021). A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). European radiology, 31: 6096-6104.

[8] Shen, Dinggang, Guorong Wu, and Heung-Il Suk. (2017). Deep learning in medical image analysis. Annual review of biomedical engineering, 19: 221-248.

[9] Zhang, Jianpeng, et al. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. arXiv.

[10] Chen, Jun, et al. (2020). Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography." Scientific reports, 10.1: 19196.

[11] Russakovsky, Olga, et al. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115: 211-252.

[12] Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd ed.). Prentice Hall.

[13] Pizer, Stephen M., et al. (1987). Adaptive histogram equalization and its variations." Computer vision, graphics, and image processing, 39.3: 355-368.

[14] Perez, Luis, and Jason Wang. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv.

DOI: 10.54254/2755-2721/67/20240612

[15] Shorten, Connor, and Taghi M. Khoshgoftaar. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6.1: 1-48.
[16] LeCun, Yann, et al. (1998). Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86.11: 2278-2324.
[17] Ioffe, Sergey, and Christian Szegedy. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. pmlr.
[18] Srivastava, Nitish, et al. (2014). Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research, 15.1: 1929-1958.
[19] Kingma, Diederik P., and Jimmy Ba. (2014). Adam: A method for stochastic optimization. ArXiv,1412.6980 .
[20] Dozat, Timothy. (2016). Incorporating nesterov momentum into adam. ICLR.
[21] Bishop, Christopher M. (2006). Pattern recognition and machine learning. Springer google schola, 2: 5-43.
[22] Sokolova, Marina, and Guy Lapalme. (2009). A systematic analysis of performance measures for classification tasks. Information processing & management, 45.4: 427-437.
[23] Goutte, Cyril, and Eric Gaussier. (2005). A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. European conference on information retrieval. Berlin, Heidelberg: Springer Berlin Heidelberg.
[24] Tan, Mingxing, and Quoc Le. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR.