Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 199-209
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Radiography Image Classification Using Deep Convolutional Neural Networks

Ahmad Chowdhury, Haiyi Zhang

Department of Computer Science, Acadia University, Wolfville, Canada
Email: 0304974c@acadiau.ca, haiyi.zhang@acadiau.ca

How to cite this paper: Chowdhury, A. and Zhang, H.Y. (2024) Radiography Image Classification Using Deep Convolutional Neural Networks. *Journal of Computer and Communications*, **12**, 199-209.
https://doi.org/10.4236/jcc.2024.126012

Received: March 7, 2024
Accepted: June 25, 2024
Published: June 28, 2024

Abstract

Research has shown that chest radiography images of patients with different diseases, such as pneumonia, COVID-19, SARS, pneumothorax, etc., all exhibit some form of abnormality. Several deep learning techniques can be used to identify each of these anomalies in the chest x-ray images. Convolutional neural networks (CNNs) have shown great success in the fields of image recognition and image classification since there are numerous large-scale annotated image datasets available. The classification of medical images, particularly radiographic images, remains one of the biggest hurdles in medical diagnosis because of the restricted availability of annotated medical images. However, such difficulty can be solved by utilizing several deep learning strategies, including data augmentation and transfer learning. The aim was to build a model that would detect abnormalities in chest x-ray images with the highest probability. To do that, different models were built with different features. While making a CNN model, one of the main tasks is to tune the model by changing the hyperparameters and layers so that the model gives out good training and testing results. In our case, three different models were built, and finally, the last one gave out the best-predicted results. From that last model, we got 98% training accuracy, 84% validation, and 81% testing accuracy. The reason behind the final model giving out the best evaluation scores is that it was a well-fitted model. There was no overfitting or underfitting issues. Our aim with this project was to make a tool using the CNN model in R language, which will help detect abnormalities in radiography images. The tool will be able to detect diseases such as Pneumonia, Covid-19, Effusions, Infiltration, Pneumothorax, and others. Because of its high accuracy, this research chose to use supervised multi-class classification techniques as well as Convolutional Neural Networks (CNNs) to classify different chest x-ray images. CNNs are extremely efficient and successful at reducing the number of parameters while maintaining the quality of the primary model. CNNs are also trained to recognize the edges of various objects in any

batch of images. CNNs automatically discover the relevant aspects in labeled data and learn the distinguishing features for each class by themselves.

Keywords

CNN, Radiography, Image Classification, R, Keras, Chest X-Ray, Machine Learning

1. Introduction

Chest x-ray images are one of the most popular and efficient types of screening and diagnosis procedures for detecting various thoracic diseases. In research, it was discovered that people infected with numerous diseases such as Pneumonia, COVID-19, SARS, Pneumothorax, and others all show some form of abnormality in their chest radiography images. All of these anomalies in chest X-ray images can be recognized using various deep-learning techniques [1].

Convolutional Neural Networks (CNNs) are used in medical fields for a variety of purposes, including the classification of chest X-rays and localization of common thoracic diseases [2]. As we all know, accurately interpreting a chest radiology image is extremely difficult. Even well-trained radiologists make mistakes because it is difficult to distinguish between different chest x-ray images. After all, many diseases have similar visual features. As a result, this chest X-ray diagnosis tool can be used to aid and support radiologists by reducing detection and analyzing time. This tool will also be able to classify thoracic diseases with the greatest degree of certainty [3]. This tool can also be used in hospitals and diagnostic centers for X-ray screening, which reduces patient wait time and can help speed up diagnostic workflow in an emergency. Our project aims to create a chest X-ray classifier that will serve as an additional resource for all expert radiologists.

Because of the high accuracy of Convolutional Neural Networks (CNNs), this project intends to use a supervised multi-class classification technique and Convolutional Neural Networks (CNNs) [4] to classify different chest X-ray images. Keras, a deep learning framework, and the TensorFlow backend are used in this project. Several deep learning-based approaches have been proposed for classifying pneumonia, lung cancer, etc. and most of them have proven that they could achieve human-level diagnostic performance.

Although we cannot achieve 100% accuracy with the data we currently have, we can certainly build and optimize a model that achieves nearly 99% training accuracy. We trained our model on over 17,000 chest X-ray images and used a customized dataset that included 6 different classes of chest X-ray images that were manually labeled and saved. They are: "Covid-19", "Effusions", "Infiltration", "No-Findings", "Pneumonia", and "Pneumothorax". The chest x-ray image data were obtained from "NIH Chest X-Ray Dataset," [5] one of the largest chest X-ray datasets. To put our model to the test, we used 900 previously un-

seen images that were also manually labeled and saved [6]. of our models, which we created from scratch, obtained a training evaluation score of 97% and a testing evaluation score of 72.13%. It was an overfitted model, so to counter that, we tweaked/fine-tuned our previous model multiple times by changing the batch size, learning rate, adding max-pooling and dropout. Finally, the testing evaluation result increased from 72.13% to 81%.

2. Background

The emergence of deep learning models for automatic diagnosis of thorax diseases bolstered by medical imagery technology, notably on chest radiology, is one recent technological progress in the medical imaging field. ChestNet, [7] which involves the involvement of two branches (classification and attention) and has shown excellent results in the detection of disease of the thorax by utilization of contrast radiography of the lungs. The effectiveness of evaluation against the above-mentioned best-performing deep learning models emphasized the need for automated programs during the diagnosis. Furthermore, CNNs are also making it possible to extract the most important characteristics from X-ray pictures with the invisible purpose of diagnosing COVID-19 [8]. Transfer learning has made it possible for the advancement of detection and classification of COVID-19 abnormalities, however, challenges still exist like data cleanliness and the availability of labeled Covid-19 data. Further, a 121-layer CNN architecture [9], pre-trained with a large global dataset of the normal frontal view of chest x-rays, achieved a better classification accuracy than the board radiologists. However, it appears to be focused only on frontal view and cannot tell a patient's medical history, and that is why other research is mandatory.

3. Methodology

3.1. Introduction

This chapter describes the methodology of building the chest X-ray image diagnosis tool using a deep convolutional neural network (CNN). This chapter shows the whole processing application, how the data was cleaned and preprocessed, and then how the model was trained and built. The data scaling and Normalization part is described in section 3.3 and some other training processes are described in sections 3.4 and 3.5.

3.2. Data Description

For this research, the dataset is organized into three folders (train, validation, and test). All three train, validation, and test folders contain 6 subfolders ("Covid-19", "Effusions", "Infiltration", "No-Findings", "Pneumonia" and "Pneumothorax"), one for each class. For building the model, we are using in total of 17,105 images and separate 900 images for testing this model [10]. The dataset is divided into an 80:20 (%) ratio. So, for training, there are 13,500 images and for validation, there are 2705 images.

3.3. Data Scaling and Normalization

For this research, the dataset is organized into three folders (train, validation, and test). All three train, validation, and test folders contain 6 subfolders ("Covid-19", "Effusions", "Infiltration", "No-Findings", "Pneumonia" and "Pneumothorax"), one for each class. For building the model, we are using in total of 17,105 images and separate 900 images for testing this model [10]. The dataset is divided into an 80:20 (%) ratio. So, for training, there are 13,500 images and for validation, there are 2705 images.

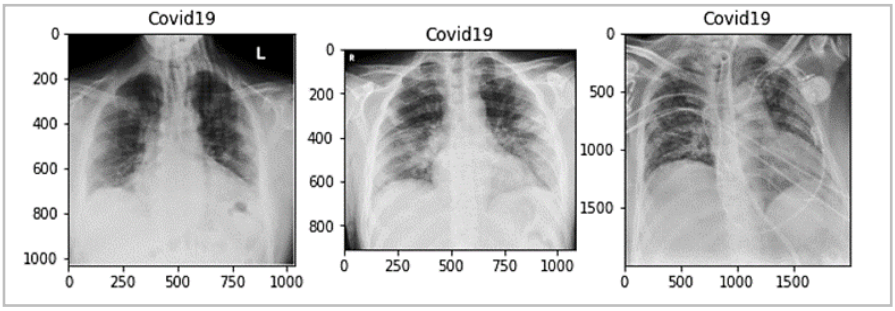

Figure 1. Unscaled training and validation data (Covid-19).

In **Figure 1** we can see that the images are of different sizes. Differences in the scales across input variables may increase the difficulty of the problem being modeled. As our training and validation images are not scaled properly, we have to first scale them using the Normalization method. Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1. Based on data inspection, images are scaled to a size of 224 by 224 and normalized to values (0, 1). This process will help our model to learn more steadily as now all the image data have the same size (224, 224). We divide the training, testing, and validation datasets by 255, simply to rescale and normalize the data, which will eventually improve the performance of the activation function (see **Figure 2**).

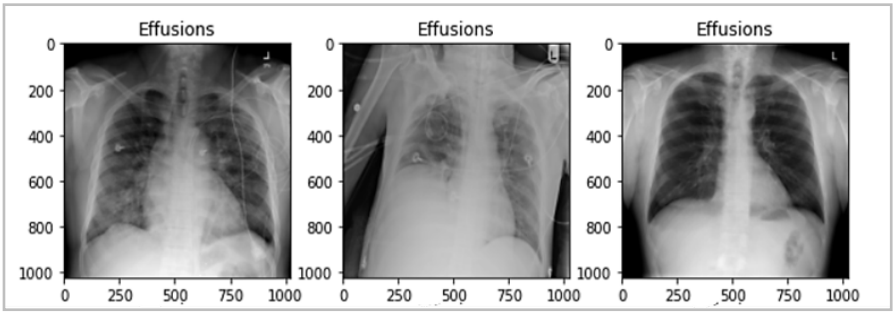

Figure 2. Rescaled training and validation data (Effusions).

3.4. Data Augmentation

The data that we choose for training our CNN model undoubtedly plays the most important role. To make a deep learning model work perfectly, we need a

vast amount of data. If the amount of data is huge, we also need huge computing power (GPUs). On the internet, there is no shortage of data availability, but fetching the right type of data that matches the exact use case of our experiment is a daunting task. The x-ray image data that we are looking at has to have good diversity as the object of interest needs to be present in varying sizes, lighting conditions, and right poses. As we made a customized dataset for our project because there were some hardware and computing limitations, we ended up with around 17,100 images which is not very ideal if we want to make our model work perfectly. This will cause our model to overfit but fortunately, we can easily overcome this problem by using data augmentation [11]. In data augmentation, we can generate a lot of new data with the help of our existing data, which will eliminate the problem of having less data.

3.5. Proposed CNN Model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input sensor and one output sensor. Models in Keras can come in two forms—Sequential and Functional API. The sequential model is widely used because it allows us to easily stack sequential layers of the network in order from input to output. First, we instantiated our Sequential model object and then, we added layers to it one by one using the add () method. We added a 2D convolutional layer to process the chest X-ray input images. The first argument passed to the Conv2D() layer function is the number of output channels—in this case, we have 32 output channels in the first convolutional layer. The next input is the kernel size, which in this case we have chosen to be a 3 × 3 moving window, followed by the strides in the *x* and *y* directions (1, 1). After that, we used Dropout just to prevent the model from overfitting. Next, the activation function (ReLU) [12] and finally, we supplied the model with the size of the input to the layer, which is (64, 64). After that, we used a second Conv2D() layer and added a 2D max pooling layer to shrink the input size. We simply specified the size of the pooling in the *x* and *y* directions—(2, 2) in this case and the strides. Next, we added another convolutional + max pooling layer with 32 output channels (see **Figure 3**).

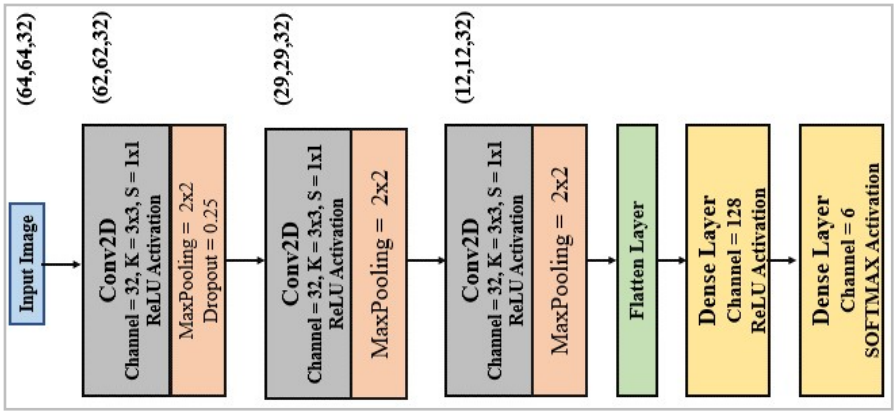

Figure 3. Schematic of the proposed sequential model architecture for input shape (64, 64).

After building all three convolutional hidden layers, we flatten the output and enter it into the fully connected layers [13]. Using the Dense () layer in Keras, we declared the fully connected layers. In the first Dense layer, there are 128 output channels, with the same activation function, but the last Dense layer has only 6 output channels as we are classifying between 6 different classes, and in the last Dense layer, we used the SoftMax activation function because of the categorical class mode. After that, we optimized and compiled the whole model. For optimizing the model, we used the Adam [14] optimizer with a learning rate of 0.001. In **Figure 4** we can see the schematic of the proposed sequential model. The Sequential Model that we built has a total of 167,750 parameters. From there, all were trainable parameters.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 62, 62, 32)        896
 max_pooling2d_3 (MaxPooling2D  (None, 31, 31, 32)     0
 )
 dropout_3 (Dropout)         (None, 31, 31, 32)        0
 conv2d_2 (Conv2D)           (None, 29, 29, 32)        9248
 max_pooling2d_2 (MaxPooling2D  (None, 14, 14, 32)     0
 )
 dropout_2 (Dropout)         (None, 14, 14, 32)        0
 conv2d_1 (Conv2D)           (None, 12, 12, 32)        9248
 max_pooling2d_1 (MaxPooling2D  (None, 6, 6, 32)       0
 )
 dropout_1 (Dropout)         (None, 6, 6, 32)          0
 flatten_1 (Flatten)         (None, 1152)              0
 dense_2 (Dense)             (None, 128)               147584
 dense_1 (Dense)             (None, 6)                 774
=================================================================
Total params: 167750 (655.27 KB)
Trainable params: 167750 (655.27 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 4. Sequential model summary with trainable parameters.

4. Result Analysis and Discussion

In this chapter, the performances of the proposed schemes are presented with a visual interpretation. In this project we built multiple sequential models from scratch, using the same customized dataset. All three models showed different results, some performed better, and some didn't. We achieved 98% training accuracy in two of our three models. All three models training accuracy, validation accuracy, training loss, validation loss, etc. are shown and described with visual representation in **Figures 5-7**.

4.1. Sequential Model Training and Validation Accuracy

This is the model which we built from scratch. In this model, we used three convolutional layers, three pooling layers, and two dense layers. We used 70 epochs while training this model. After the training was done, we saw a maximum of 98% training accuracy and a maximum of 84% validation accuracy. The final training loss was 0.0479.

```
422/422 [==============================] - 139s 330ms/step - loss: 0.0479 - accuracy: 0.9838
[1] "Model evaluation result --> Training Data"
      loss   accuracy
0.04790983 0.98377776
```

Figure 5. Training accuracy and loss score.

```
85/85 [==============================] - 28s 332ms/step - loss: 0.7017 - accuracy: 0.8362
[1] "Model evaluation result --> Validation Data"
     loss  accuracy
0.7017277 0.8362292
```

Figure 6. Validation accuracy and loss score.

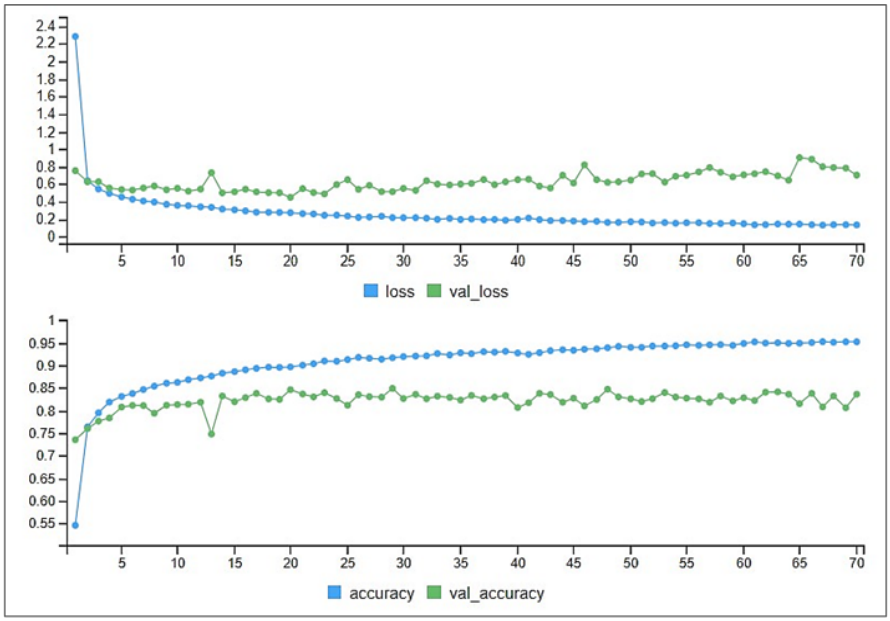

Figure 7. Plotting of Training-Validation accuracy and loss.

In the plotting figure above (**Figure 7**), we can see our training accuracy keeps getting high and training loss keeps getting low. In the figure above we can also see our training accuracy is higher than the validation accuracy. Also, not in a single moment did we see the training loss go high up. At the end of the training, we saw training loss was less than the validation loss, which resulted in our model performing better and we didn't face any overfitting or underfitting issues. The training accuracy graph is shown using the blue line and the validation accuracy graph is shown using the green line.

4.2. Model Evaluation

All three models showed different test results. Some predicted better and some didn't. We achieved the highest 81% testing accuracy in one of our three models. The rest two achieved on average 72% testing accuracy score. We tested all three models with unseen test images. We first scaled and normalized all the testing images for better results. To see how our model is performing with unseen chest X-ray images, we performed the "confusion matrix" and "classification report" techniques [12]. Both of these operations were successfully performed with the

help of the "sklearn matrics" library. All three models' testing evaluation scores, confusion matrix, and classification reports are shown and described with visual representation in **Figures 8-10**.

4.3. Sequential Model Test Evaluation Score

This is the testing evaluation score of the final model which we built from scratch. After the evaluation of unseen test images was done, we saw a maximum of 81% testing score. This means this particular model's predicted accuracy score is 0.81.

```
26/26 [==============================] - 8s 283ms/step - loss: 0.9201 -
accuracy: 0.8095
[1] "Model evaluation result --> Test Data"
     loss  accuracy
0.9201103 0.8095238
```

Figure 8. Testing accuracy and loss score.

4.4. Sequential Model Confusion Matrix

We trained our multiclass classification CNN model using Keras and we also evaluated the model on our test set of images using the Confusion Matrix.

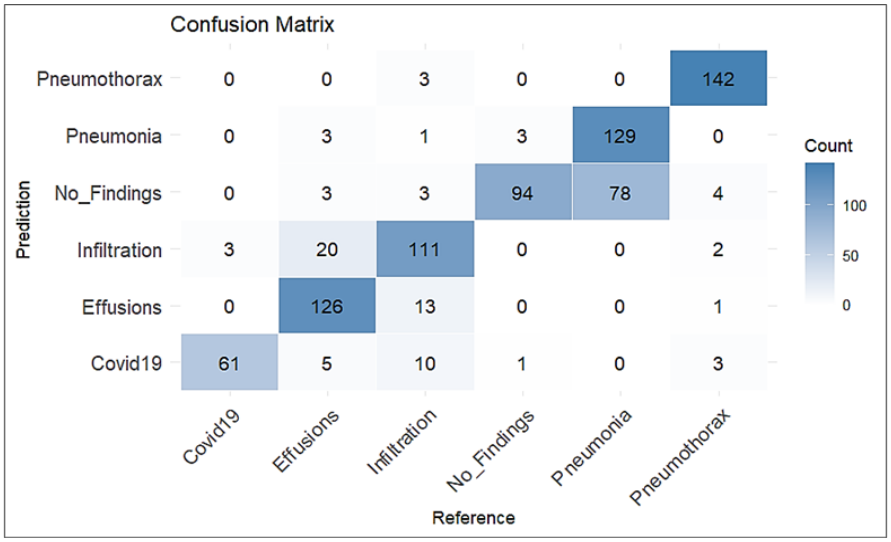

Figure 9. Confusion Matrix of test data.

It is a specific table layout that allows visualization of the performance of a model. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. It helped us describe the performance of our chest x-ray classifier on a set of test data for which the true values are known. If the accuracy score is close to "1", then it is a great prediction. From the figure below we can see, "Pneumothorax" predicted 142 images correctly out of 145 images, which is almost a perfect result. If we compare all six classes' prediction scores, we can see on average all the classes scored 0.80, which is very good.

4.5. Sequential Model Classification Report

In **Figure 11** below we can see "Pneumothorax" has the best F1 and Recall score. The F1 score of "Pneumothorax" is 0.95 and the Recall score is 0.97. On average all the classes' F1 score is 0.80, meaning they gave out good prediction results. If we take a look at the overall balanced accuracy score, we can see apart from

```
[1] "Classification Report:"
                     Sensitivity Specificity Pos Pred Value
Class: Covid19         0.7625000   0.9959405      0.9531250
Class: Effusions       0.9000000   0.9543446      0.8025478
Class: Infiltration    0.8161765   0.9560761      0.7872340
Class: No_Findings     0.5164835   0.9937206      0.9591837
Class: Pneumonia       0.9485294   0.8857980      0.6231884
Class: Pneumothorax    0.9793103   0.9851632      0.9342105
                     Neg Pred Value Precision    Recall        F1
Class: Covid19           0.9748344 0.9531250 0.7625000 0.8472222
Class: Effusions         0.9788520 0.8025478 0.9000000 0.8484848
Class: Infiltration      0.9631268 0.7872340 0.8161765 0.8014440
Class: No_Findings       0.8779473 0.9591837 0.5164835 0.6714286
Class: Pneumonia         0.9885621 0.6231884 0.9485294 0.7521866
Class: Pneumothorax      0.9955022 0.9342105 0.9793103 0.9562290
                     Prevalence Detection Rate Detection Prevalence
Class: Covid19       0.0976801      0.07448107            0.07814408
Class: Effusions     0.1709402      0.15384615            0.19169719
Class: Infiltration  0.1660562      0.13553114            0.17216117
Class: No_Findings   0.2222222      0.11477411            0.11965812
Class: Pneumonia     0.1660562      0.15750916            0.25274725
Class: Pneumothorax  0.1770452      0.17338217            0.18559219
                     Balanced Accuracy
Class: Covid19               0.8792202
Class: Effusions             0.9271723
Class: Infiltration          0.8861263
Class: No_Findings           0.7551020
Class: Pneumonia             0.9171637
Class: Pneumothorax          0.9822368
```

Figure 10. Classification Report of multiclass sequential model.

	Features	Trainable Parameters	Training Score	Validation Score	Prediction Score
Version 1	i. Without Batch Normalization. ii. Without Dropout. iii. Optimizer: "**Adam**"	**167,750**	**0.98**	**0.81**	**0.72**
Version 2	i. With Batch Normalization. ii. With Dropout. iii. Optimizer: **"SGD"**	**16,851,398**	**0.85**	**0.83**	**0.73**
Version 3	i. Without Batch Normalization. ii. With MaxPooling iii. With Dropout iv. More Hidden Layers. v. Optimizer: **"Adam"**	**167,750**	**0.98**	**0.84**	**0.81**

Figure 11. All three models training, validation, and testing score comparison.

"No-Findings", all the other classes performed well. If we calculate the average of those balanced accuracy scores, we can see that it is almost 90%, which again defines how well-fitted our model is and how well it performed when we tested it with completely unseen data. The rows define six different classes of our model. The columns define different metrics such as precision, recall, F1 score, etc.

4.6. Model Comparison

Below is the table-like figure we can see the model comparison of the three models we built and trained for this project. Version 1 didn't perform well, it got only a 72% prediction score. For version 3, we added a max pooling layer, drop out, and more hidden layers. As a result, our prediction score jumped up from 72% to 81%.

5. Conclusion

The most important things we have to consider while doing a project like this are Data Collection, Data Labeling, and Data Cleaning. We took data for our project from an open-source database, so we have to keep in mind that all the data may not be correctly labeled and therefore it will be challenging for our model to distinguish different chest x-ray images accurately as many diseases have similar visual features. Furthermore, research and effort will be required to make this tool work perfectly. In the future, we need to use a bigger and better dataset, more hardware, and GPU power to store and train data. We can also use multiple feature selection in the future which will help to distinguish between diseases with similar features. Other than chest X-rays, this classification tool may also be used in the future to detect and diagnose other X-ray images like fractures, bone dislocations, CT scans, etc.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Rajpurkar, P., Irvin, J., Ball, R.L., Zhu, K., Yang, B., Mehta, H., *et al.* (2018) Deep Learning for Chest Radiograph Diagnosis: A Retrospective Comparison of the Chexnext Algorithm to Practicing Radiologists. *PLOS Medicine*, **15**, e1002686. https://doi.org/10.1371/journal.pmed.1002686

[2] Anwar, S.M., Majid, M., Qayyum, A., Awais, M., Alnowami, M. and Khan, M.K. (2018) Medical Image Analysis Using Convolutional Neural Networks: A Review. *Journal of Medical Systems*, **42**. https://doi.org/10.1007/s10916-018-1088-1

[3] Mazurowski, M.A., Buda, M., Saha, A. and Bashir, M.R. (2018) Deep Learning in Radiology: An Overview of the Concepts and a Survey of the State of the Art with Focus on Mri. *Journal of Magnetic Resonance Imaging*, **49**, 939-954. https://doi.org/10.1002/jmri.26534

[4] Mahmud, T., Rahman, M.A. and Fattah, S.A. (2020) Covxnet: A Multi-Dilation Convolutional Neural Network for Automatic COVID-19 and Other Pneumonia Detec-

tion from Chest X-Ray Images with Transferable Multi-Receptive Feature Optimization. *Computers in Biology and Medicine*, **122**, 103869. https://doi.org/10.1016/j.compbiomed.2020.103869

[5] R. (NIH/CC/DRD) [E] Summers (2017) NIH Chest X-Ray Dataset. National Institutes of Health-Clinical Center. https://nihcc.app.box.com/v/ChestXray-NIHCC.

[6] Hsu, J., Lu, P. and Khosla, K. Predicting Thorax Diseases with NIH Chest X-Rays. 4-8. https://pdfs.semanticscholar.org/6919/75f1a12148d6c30a442dd1415ff381dd25d2.pdf

[7] Wang, H. and Xia, Y. (2018) ChestNet: A Deep Neural Network for Classification of Thoracic Diseases on Chest Radiography. 1-8.

[8] Wang, L., Lin, Z.Q. and Wong, A. (2020) Covid-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images. *Scientific Reports*, **10**. https://doi.org/10.1038/s41598-020-76550-z

[9] Rajpurkar, P., *et al.* (2017) CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning.

[10] Cohen, J.P. (2019) Covid-19 Chest X-Ray Dataset. https://github.com/ieee8023/covid-chestxray-dataset

[11] Wong, S.C., Gatt, A., Stamatescu, V. and McDonnell, M.D. (2016). Understanding Data Augmentation for Classification: When to Warp? 2016 *International Conference on Digital Image Computing*: *Techniques and Applications* (*DICTA*). https://doi.org/10.1109/dicta.2016.7797091

[12] Pham, H.H., Le, T.T., Tran, D.Q., Ngo, D.T. and Nguyen, H.Q. (2019) Interpreting Chest X-Rays via CNNs That Exploit Disease Dependencies and Uncertainty Labels.

[13] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q. (2017) Densely Connected Convolutional Networks. 2017 *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*). https://doi.org/10.1109/cvpr.2017.243

[14] Kingma, D.P. and Ba, J.L. (2015) Adam: A Method for Stochastic Optimization. 3*rd Int. Conf. Learn. Represent. ICLR* 2015-*Conf. Track Proc.*, 1-15.