Classification of Pneumonia Infection from Chest X-rays using Convolution Neural Network

Galapon, Arthur V., Phan Thi Dieu Trang

February 2021

Abstract

A pneumonia detection deep learning algorithm was performed using chest x-ray images. A traditional convolutional neural network with two hidden convolutional layers was used to perform classification. The network was trained using $5,216$ training image set of pre-classified chest x-rays. Data augmentation was performed virtually due to the limited number of training set. A set of hyperparameter settings were varied to obtain the best possible model. Results show an accuracy of 93.269% with a 97% precision to detect normal chest x-ray images and 92% precision to detect pneumonia-infected chest x-ray.

1 Introduction

In 2017, WHO reported that pneumonia accounts for 18% of deaths of children under 5 years old [9]. Pneumonia is an acute respiratory inflammation of the lungs' air sacs or alveoli. Pneumonia may be caused by several infectious agents such as bacteria, fungi, and viruses. The most common of these is the *Streptococcus pneumoniae* which is the common cause of bacterial pneumonia in children. Patients with pneumonia may experience shortness of breath or difficulty in breathing, cough with phlegm or blood, fatigue, chest or muscle pains, and fever [5].

Aside from the aforementioned symptoms, pneumonia diagnosis can be confirmed via several tests such as chest x-rays, CT of lungs, chest ultrasound, chest MRIs, or lung biopsy. Among these tests, x-ray imaging is preferred over the other modalities due to its fast procedure and availability in

almost all hospital facilities. Radiologists will look for white spots in the lungs, known as infiltrates, which suggests infection. Other lung-related complications such as pleural effusions can also be detected via x-ray imaging.

Due to the pandemic caused by the CoVid-19 virus, the number of lung-related infections such as pneumonia has increased. It is, therefore, more urgent for radiologists to be able to make fast and accurate interpretation of the x-ray images. However, some details can still be missed out even for trained specialists and this may prove to be fatal especially for high-risks patients.

In order to address this need, computer-aided technologies can be implemented to help analyze these huge number of information. Deep learning algorithms such as convolution neural network can be used to facilitate classifications of x-ray images. Together with the physician, this can help speed-up diagnosis of lung-related diseases such as pneumonia.

Objectives of the Study

The goal of the project is to determine if the given chest x-ray can be classified as that of a normal lungs or as a lung with pneumonia using deep learning algorithms. The classification will be performed by a convolution neural network using an existing chest x-ray dataset. The final objective is to determine the best model that can predict with the highest accuracy by varying the different hyperparameters of model.

2 Theoretical Background

Convolutional Neural Network

Artificial Neural Networks (ANNs) are computational systems which are inspired by biological systems such as the nervous system. These machine learning algorithms can learn from data and are best suited for pattern recognition [1]. The basic structure of an ANN is mainly comprised of a number of interconnected computational nodes, often referred as neurons, in which the processes and 'learning' are distributed as shown in figure 1. Each neuron receives signals from other neurons and accumulates the result until it exceeds an activation threshold. Once the activation threshold is reached, the neuron will produce an output signal, otherwise, the neuron will remain inactive [2].

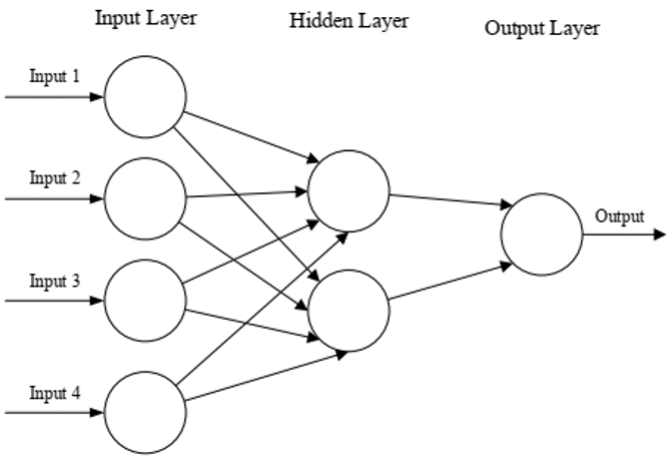


Figure 1: Size calibration of an image using *Set Scale* on ImageJ.

Convolutional Neural Network is a type of deep learning algorithm that is analogous to a traditional ANN. It is also comprised of neurons that self-optimize through learning. The only difference is that CNNs are mainly used in pattern recognition applications in other media such as images and sound files. Unlike ANNs, which tend to struggle with the computational complexity required to compute image data, CNNs can process larger image sizes in multiple channels. For a colored image of size 32 x 32 x 3, a single fully-connected neuron will have 3072 weights. This is still manageable computationally for conventional ANNs. However, for larger images such as that of a 200 x 200 x 3 image, the total amount of weights will be around $120,000$ for a single neuron. The total number of parameters would rapidly increase and may lead to possible overfitting and longer learning duration.

CNN Architecture: Input Layer

A typical CNN architecture consists of an input layer followed by multiple layers of convolution and pooling layers, and one or more fully connected layers as shown in figure 2.

The input layer is composed of the raw input image. Unlike computer systems, the human eyes can easily recognize images. Our brains are designed to process all the relevant information it receives from the different visual stimuli that our eyes see in as fast as 13ms [4]. However, this is not the case for computers. The way computers see images is in the form of a matrix as shown in figure 3. Each cell in the matrix, known as a pixel, carries a value which ranges from 0 to 255. A pixel value

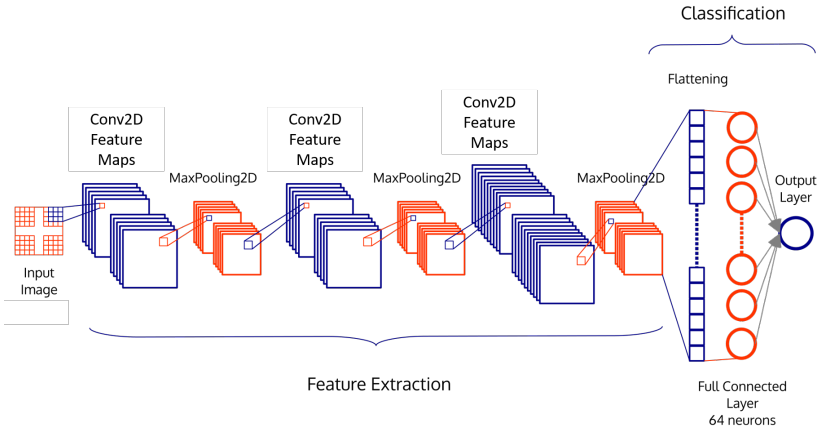


Figure 2: Convolutional Neural Network Architecture

of 0 indicates a black pixel, while a maximum pixel value corresponds to a white pixel. For colored images, the matrix comes in the form of a 3-channel matrix with each channel corresponding to a red, blue, and green matrices. This method of image interpretation reduces all image manipulation processes into a simple matrix operation.

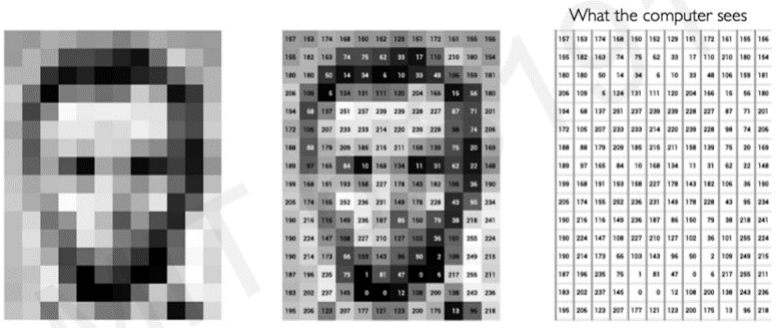


Figure 3: Matrix representation of images as 'seen' by a computer.

Convolution Layer

Feature extraction occurs at the convolutional layer using convolution filters or kernels. This is done using a combination of linear and non-linear operations such as convolution operation and activation functions.

Convolution is a mathematical operation between two function that shows the amount of overlap

of one function *g(x)* as it is shifted over another function *f(x)* [8]. In convolutional neural networks, the convolution operation is a specialized type of linear operation where a small array of numbers, often called as filters or kernel, is applied to the input which, in this case, is the input image matrix [10]. Element-wise product was performed between the filter and the input followed by a summation to obtain the output for that specific position as shown in figure 4. This operation is repeated over all positions. The resulting matrix is known as the feature map which represents different characteristics of the input matrix. Application of different filters will result to a different feature map. Thus, these filters or kernels are considered as feature extractors. The convolution operation can be expressed mathematically as:

$$s(t) = \int x(a)w(t-a) = (x * w)(t),$$

where w is the kernel, x is the input matrix, and s is the output feature map. For a two-dimensional image, the convolution of the image I and a two-dimensional kernel K, the convolution is expressed as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

The convolution operation, however, does not allow the center of the kernel to overlap the outermost element of the input. This leads to a shrinking of the output tensor which leads to loss of information on the image corners. To address this issue, padding techniques such as zero padding were applied. Rows and columns of zeros were added on each side of the input matrix to fit the center of the filter matrix on the outermost element, thus allowing it to keep the same in-plane dimension throughout the entire convolution operation. Without padding, each successive feature maps will reduce in size for each convolution operation.

The process of training a CNN model in the convolution layer includes the identification of the kernels that best works on the given training set. These kernels are automatically learned during the training process at the convolution layer. Figure 5 shows some of the filters learned using the chest x-ray dataset.

The outputs of the convolution process is passed through a non-linear activation function such

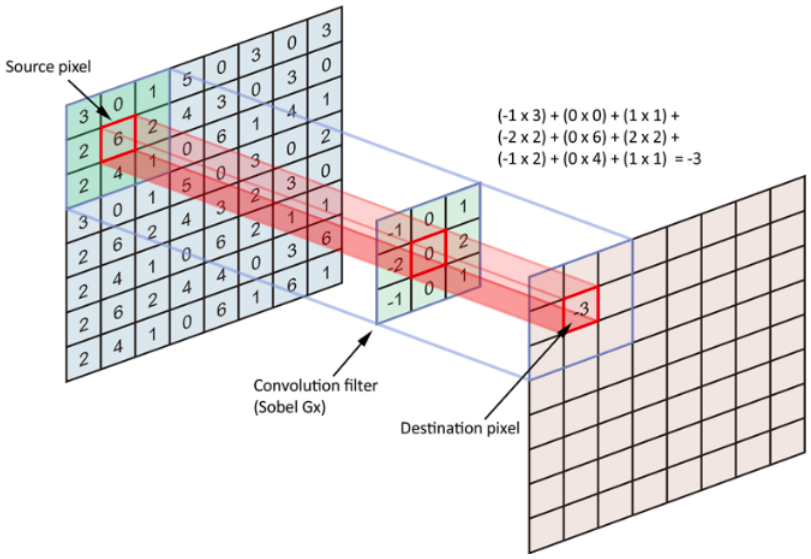


Figure 4: Convolution Operation Diagram

as 'ReLu' or rectified linear unit which simply computes the function $f(x) = max(0, x)$ as shown in figure 6. Other activation functions that can be used are the sigmoid and hyperbolic tangent.

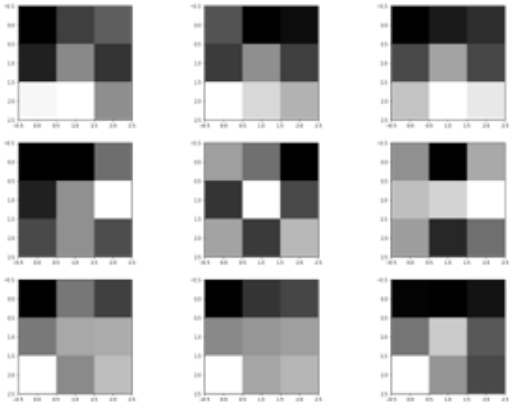


Figure 5: Sample filters generated from the chest x-ray training set.

Pooling Layer

The pooling layer performs down-sampling operations to reduce the in-plane dimensionality of feature maps. This will introduce a translation invariance to small shifts and distortion, and a reduction in the number of subsequent learnable parameters. Reducing the size of the input tensor helps im-

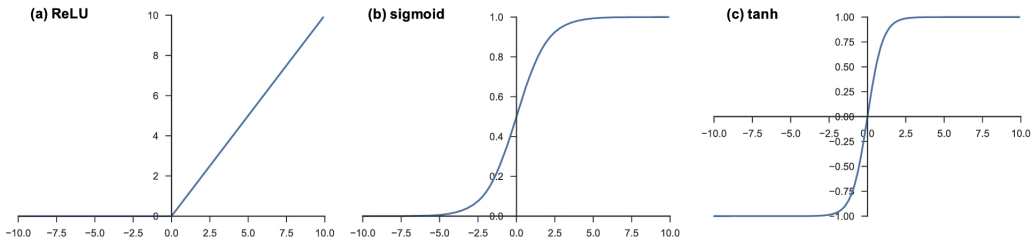


Figure 6: Activation Functions: Relu, Sigmoid, and Hyperbolic Tanget

prove the performance of the model and reduce the possibility of overfitting.

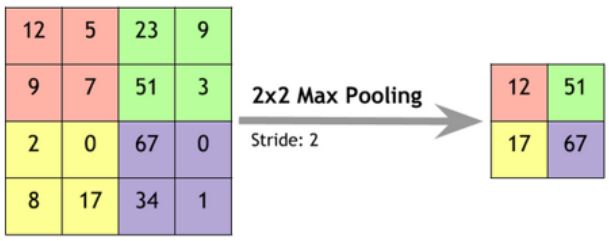


Figure 7: Diagram of max pooling operation.

The most frequently used form of pooling operation is max pooling. A Max pooling filter, typically of size 2x2, extracts patches from the feature maps, then outputs the maximum value in each patch while discarding all other values as shown in figure 7. Other pooling operations include the average of rectangular neighborhood, the L^2 norm of rectangular neighborhood, or a weighted average based on the distance from the central pixel [1].

Fully Connected Layer

The objective of a fully connected layer is to take the results from the convolutional layer and use them to classify the image to a label as in the case of a simple binary classification problem. The resulting feature maps from the final convolution layer or pooling layer is flattened into a one-dimensional array of numbers and then connected to one or more fully connected layers known as dense layers. Each input in the dense layer is connected to every output by a learnable weight before passing through an activation function.

The activation function applied to the last fully connected layer is selected according to the task

of the network. For binary classification, the sigmoid activation function is typically used. Softmax and sigmoid functions are also used for multiclass single-class and multiclass multiclass classifications respectively. For tasks that involve regression to continuous values, the identity activation function can be used.

Materials and Methods

Dataset

The dataset used were chest x-ray images (Anteroposterior view) of pediatric patients of age one to five years old from the Guangzhou Women and Children's Medical Center in Guangzhou, China [3]. The dataset was already organized into three different folders (train, test, and val), each with two sub-folders (Normal and Pneumonia). A total of 5,216 images, 1341 of which are normal chest x-ray images and 3,875 are classified as pneumonia chest x-ray images, were used as the training set. For the test set, a total of 624 chest x-ray images were used in which 234 are pre-classified as normal and 390 as pneumonia infected. A validation set was used to measure validation loss in the model and was composed of 8 normal and 8 pneumonia x-ray images.

Model

The model used in this project is a Sequential model generated using Tensorflow. Sequential models were used for plain stack of layers where each layers has exactly one input tensor and one output tensor.

The model generated was composed of an input layer, two hidden layers using the ReLu activation function and its corresponding pooling layers using maxpooling operations, and two dense layers using 'ReLu' for the first dense layer and a 'Sigmoid' activation function for the last dense layer as shown by the model summary in figure 8. A dropout layer was added between the dense layers to regularize the model and reduce overfitting. The dropout layer temporarily removed random nodes in the dense layers. Standard backpropagation learning creates co-adaptations that work for the training data but does not generalize to new inputs. Dropping random nodes breaks these co-adaptations by thinning the neural network to force the nodes within a layer to take on more or less

responsibility for the inputs [6].

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 512, 512, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 256, 256, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 256, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 64)                8388672
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 8,408,129
Trainable params: 8,408,129
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: Model Summary of the CNN used.

The model was compiled using the model function of Keras and optimized using the *Adam* optimization function. ADAM, derived from *adaptive moment estimation*, is an extension of stochastic gradient descent. It has the advantage of adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp). The loss function used is the cross entropy loss function which is the default loss function for binary classification problems and can be expressed as

$$H_p(q) = -\frac{1}{N}\sum_{n=1}^{N} y_i \log(p(y_i)) + (1 - y_i)\log(1 - p(y_i))$$

where y is the label (1 and 0 for pneumonia and normal, respectively) and p(y) is the predicted probability of the data being 1 for all N images. Cross entropy calculates a score that summarizes the average difference between the probability distributions of the actual and the predicted values.

Training the Model

The available data was split into three sets: training, validation, and test set. The model is trained using a set of pre-classified chest x-ray images. Figure 9 shows an example of a normal and pneumonia

infected x-ray images of lungs. Image pixels were normalized to ensure similar data distribution. Due to the limited number of training set, data augmentation was performed. Data augmentation artificially inflates the training set while preserving the original label of the images [7]. Table 1 shows the different geometric transformations performed.

A validation set was used to evaluate the model during the training process to facilitate the fine tuning of hyperparameters and selection of model. Data augmentation was not performed for the validation and test sets since the goal was to perform classifications using minimally manipulated images.

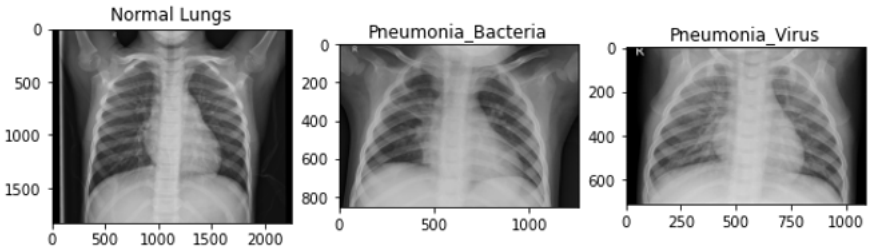


Figure 9: Left: Normal Chest X-ray. Middel: Chest X-ray of bacterial-induced Pneumonia. Right: Chest X-ray of viral-induced Pneumonia.

Due to the limited computational resource available, the hyperparameters selected were limited to the image size, number of epochs, number of batch size, size of the feature maps, and the number of channels of images. For the image size, CNN learning is performed for 64x64, 128x128, 256x256, and 512x512 pixels dimension for both the rgb (channel = 3) and grayscale (channel = 1) mode. The number of iterations was initially set at 50 epochs. The feature map dimension was set at 32x32 and the batch size was set at 16 images.

The metric used to measure the predictive power of the model was the accuracy metric. This was done using the *evaluate generator* functionality of keras. This generates an accuracy score based on the images from the test set.

Once the model has been trained, images from the test set were used to evaluate its predicting capability. This was performed using the *predict* and *predict classes* functions. These functions requires input images from the test set to feed it to the trained model. The resulting output yields an array of 1's and 0's which corresponds to 'Normal' and 'Pneumonia' lungs for the predict classes

function, and an array of the predicted score for the predict function. The resulting model with the best accuracy will be used to evaluate the effects of the other hyperparameters.

Geometric Transformation	Range of Values
Shear Transformation	0.2
Random Zoom	0.2
Random Rotation	30^o
Horizontal Flip	True

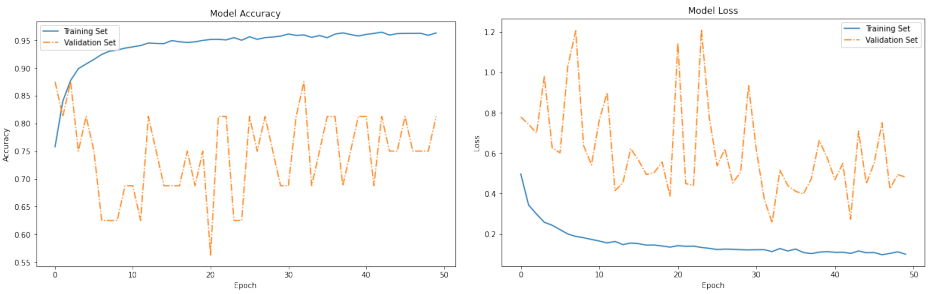
Table 1: Geometric Transformation and corresponding range of values for data augmentation.

Results and Discussion

Figure 10 shows an example of a batch of images processed in the network. It can be observed that some of the images were already skewed, rotated, and flipped as a result of the image augmentation process. The additional images were generated based on the original test set to reduce the possibility of the model to overfit due to the limited number of test images.

Figure 10: Sample images of a batch in the training set.

The accuracy metric was used to evaluate the performance of the model. Figure 11a shows the training and validation set accuracy across all the epochs. It can be observed that the accuracy of the model increases over the number of iterations which indicates that the model is updating its nodes' weights to improve detection capability. The large variations in the validation set accuracy was possibly due to the small number of images. Nevertheless, the slight increase in the validation set can still be observed from the graph. Figure 11b shows the loss for both the validation and training sets. The training set loss decreases as the epoch progresses. Similarly, the validation set



(a) Accuracy (b) Loss

Figure 11: Training and Validation set accuracy and loss over epoch.

loss also exhibits a decreasing trend.

Table 2 shows the accuracy comparison of models with different target image dimensions for both RGB and grayscale channels. For the RGB images (channels = 3), an accuracy of 91.987% was obtained for dimensions = 512x512, 90.064% for 256x256, 90.705% for 128x128, and 91.827% for target image dimensions = 64x64. For grayscale images, an accuracy of 92.628% was obtained for image dimensions = 64x64, 91.346% for dimension = 128x128, 91.026% for 256x256, and 89.583% for 512x512 as shown in table 3. The image dimensions were selected arbitrarily and was limited by the computational resources available. Any dimensions higher than 512 x 512 corresponds to several days of processing time.

Table 2: Accuracy and Precision Comparison for Models using different target image dimensions for images in RGB channel.

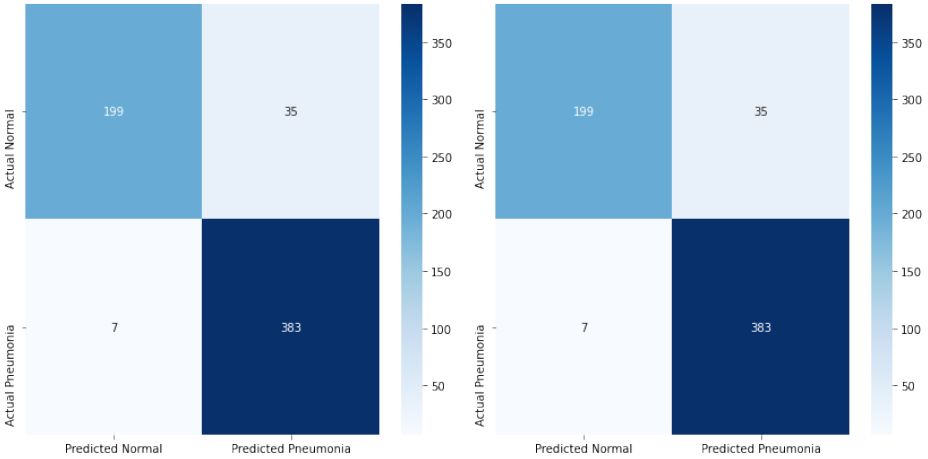
Model	Dimension	Accuracy	Precision	
			Normal	Pneumonia
A	64 x 64	91.827	0.91	0.92
B	128 x 128	90.702	0.94	0.89
C	256 x 256	90.064	0.97	0.87
D	512 x 512	91.987	0.93	0.92

Table 2 and table 3 also shows the precision of the model to predict normal and pneumonia lungs. Precision is defined as the accuracy of positive prediction. For the RGB channel, model D shows a better overall average precision as compared to the other models with a 93% precision to detect normal lungs and 92% precision to detect pneumonia lungs. For grayscale channel, model E exhibits the best precision with 96% precision for normal lungs and 91% precision for pneumonia

Table 3: Accuracy and Precision Comparison for Models using different target image dimensions for images in grayscale channel.

Model	Dimension	Accuracy	Precision	
			Normal	Pneumonia
E	64 x 64	93.269	0.97	0.92
F	128 x 128	91.346	0.95	0.90
G	256 x 256	91.026	0.95	0.89
H	512 x 512	89.583	0.96	0.87

lungs. The confusion matrices for model D and model E is shown in figure 12.



(a) Model D (b) Model E

Figure 12: Confusion Matrix of Model D and Model E.

Furthermore, another run was performed using a new model with an extra hidden convolution layer to check if an additional layer can improve the accuracy of the model. The model was applied to RGB images and results show an accuracy of 91.506 % and a precision of 96 % to detect normal lungs and 89 % to detect pneumonia lungs. This result is similar to that of model D with two hidden convolutional layers.

Conclusion and Recommendations

In this project, a deep learning algorithm is applied to a chest x-ray dataset to perform classification of normal and pneumonia-infected lungs. A standard convolutional neural network with two hidden

convolutional layer was trained using a test dataset and was used to classify the test set. Result shows a range of accuracies from 90% to 93% using different hyperparameter settings.

This project demonstrates the capacity of a deep learning algorithm to extract feature information for image classification. The accuracy obtained in this project is enough to perform x-ray image classification. However, further improvements can be performed by using optimizing further the hyperparameters, and increasing the number of validation and test sets. Furthermore, transfer learning or utilizing existing pre-trained CNN models such as VGG91, ResNet, and Mobile Net [11] can be implemented since these algorithms are already 'learned'.

References

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[2] Morten Hjorth-Jensen. Lecture notes on machine learning, February 2021.

[3] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[4] Mary C Potter, Brad Wyble, Carl Erick Hagmann, and Emily S McCourt. Detecting meaning in rsvp at 13 ms per picture. *Attention, Perception, & Psychophysics*, 76(2):270–279, 2014.

[5] RadiologyInfo. Pneumonia. `https://www.radiologyinfo.org/en/info.cfm?pg=pneumonia`. Accessed: 2021-02-22.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[7] L Taylor and G Nitschke. Improving deep learning using generic data augmentation. arxiv 2017. *arXiv preprint arXiv:1708.06020*.

[8] Eric W. Weisstein. Convolution. https://mathworld.wolfram.com/Convolution.html. Accessed: 2021-02-23.

[9] WHO. Who factsheet: Pneumonia. https://www.who.int/news-room/fact-sheets/detail/pneumonia. Accessed: 2021-02-21.

[10] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[11] Zhenjia Yue, Liangping Ma, and Runfeng Zhang. Comparison and validation of deep learning models for the diagnosis of pneumonia. *Computational Intelligence and Neuroscience*, 2020, 2020.