

**University of Hertfordshire**

Department of Physics, Astronomy and Mathematics

MSc Data Science FINAL PROJECT REPORT

Pneumonia Detection using CNN

Project report: 7PAM2002 – Data Science Project

Supervisor: Dr.Vandana das

Name: Mahesh Kumar Konda

Student Id: 21059972

Pneumonia Detection using convolutional neural networks and transfer learning models

**MSc Final Project Declaration**

This report is submitted in partial fulfilment of the requirement for the degree of Master of Data Science at the University of Hertfordshire (UH).

It is my work except where indicated in the report.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

**Abstract**

Pneumonia continues to be a major global health concern, especially among children under the age of five, where it accounts for more than 15% of all deaths. The difficulty of making fast and correct diagnoses is exacerbated by the limits of traditional diagnostic approaches, such as clinical evaluations and chest X-ray interpretations, which frequently rely on radiologists' knowledge and can be both time-consuming and error-prone. To overcome these challenges, this work investigates the use of Convolutional Neural Networks (CNNs) and transfer learning models (ResNet50V2 and VGG16) in improving pneumonia detection from chest X-ray images. The ResNet50V2 model has a recall of 0.96 for pneumonia detection, which means it correctly identifies 96% of real pneumonia cases, with an accuracy of 0.87 and an F1-score of 0.90, demonstrating a strong balance between precision and recall. In comparison, the VGG16 model has a recall of 0.92, an accuracy of 0.85, and an F1-score of 0.88, indicating high performance but slightly below ResNet50V2. However, the simple CNN model has a lower recall of 0.78, indicating that it is less effective at recognizing pneumonia cases than more complex designs. These findings highlight the ability of fine-tuned CNN models to greatly improve diagnosis accuracy and support radiologists in practical situations.

**Table of** **Contents**

[**Introduction** 3](#_Toc173881242)

[**Literature Review** 3](#_Toc173881243)

[**Methodology** 5](#_Toc173881244)

[Dataset 5](#_Toc173881245)

[Data Collection 5](#_Toc173881246)

[Data Preparation 6](#_Toc173881247)

[Data Augmentation and Preprocessing 6](#_Toc173881248)

[**Convolutional Neural Network Model** 7](#_Toc173881249)

[CNN Layers and Their Function 8](#_Toc173881250)

[**Transfer Learning Models** 8](#_Toc173881251)

[ResNet50V2 Structure 9](#_Toc173881252)

[VGG16 Structure 9](#_Toc173881253)

[**Convolutional Neural Network Architecture** 9](#_Toc173881254)

[CNN Model Training 10](#_Toc173881255)

[**ResNet50V2 Architecture** 10](#_Toc173881256)

[ResNet50V2 Model Training 11](#_Toc173881257)

[**VGG16 Architecture** 11](#_Toc173881258)

[VGG16 Model Training 12](#_Toc173881259)

[**Evaluation Metrics** 12](#_Toc173881260)

[**Results** 13](#_Toc173881261)

[**Discussion** 15](#_Toc173881262)

[**Limitations** 15](#_Toc173881263)

[**Conclusion** 15](#_Toc173881264)

[**References** 16](#_Toc173881265)

# 

# **Introduction:**

Pneumonia is a significant global health problem, and inflammation in the lungs along with alveolar fluid accumulation represent aspects leading to considerable morbidity and mortality. The World Health Organization (WHO) has already stated that pneumonia is the leading killer of children under five, accounting for more than 15% of all deaths within this age bracket. To provide successful treatment and improved patient outcomes, it is necessary to diagnose pneumonia promptly and accurately. However, the ability of diagnostic techniques based on clinical symptoms and chest X-ray interpretation is fundamentally limited. Chest X-ray is also vital for telling whether someone has pneumonia, and so it somewhat pertains to the diagnosis. It is less useful than clinical examination or basic laboratory tests but much more widely available. This is the main catch as much we appreciate how chest X-rays have their benefits, interpreting them will always rely more on radiologists. This method of manual analysis can be both time-intensive and error-prone in a setting with high patient volumes or limited access to trained radiologists. These constraints also underscore a pressing need for automated diagnostic tools to aid healthcare workers in diagnosing pneumonia.

Medical imaging has been transformed by artificial intelligence, and in particular deep learning. Deep learning uses neural networks, where the nodes that make up each layer are 'trained' to recognize patterns and features within data. Several layers of such neural nets are known as deep networks. Convolutional Neural Networks (CNNs), a type of deep learning model, excel at image categorization by automatically extracting key structural features from input images and recognizing them using algorithms on the resulting feature map. This property makes CNNs particularly suited to medical image handling because small visual differences can mean big variations in diagnostic accuracy. In recent years, CNNs have come to dominate pneumonia detection from chest X-ray images. As the news day writes, they produce results with such levels of accuracy that at times this can exceed human radiologists. Numerous reports verify that CNNs achieve results of this kind. Developing good Convolutional Neural Network (CNN) models from scratch requires large amounts of labeled data and substantial computational resources. To solve these problems, researchers are increasingly turning to transfer learning. As the name implies, this method transfers large datasets that have already experienced numerous iterations before handing them over to a new task with the same settings. In the process, it requires little new information or additional processing power. Transfer learning obviates the need for large amounts of labeled data and high computational resources by replacing the parameters of pre-trained models.

This report is primarily concerned with constructing and evaluating Convolutional Neural Networks (CNNs) as well as fine-tuning ResNet50V2 and VGG16 architectures, to increase the recall for the Pneumonia class. The concern for recall stems from a desperate requirement to accurately identify pneumonia cases, where high recall is needed to reduce false negatives substantially and make sure all true pneumonia cases are picked up.

# **Literature Review**

The existing studies have taken advantage of Convolutional Neural Networks (CNNs) and transfer learning towards identifying pneumonia in x-ray images of the chest. This section simply summarizes the key findings of well-known works in this field. Liang and Zheng (2018) used the CNN technique integrating residual connections and dilated convolutions to find pneumonia. Residual connections, as proposed by He et al. (2016), prevent vanishing gradients by transmitting gradients straight through shortcut links. In this way, deeper networks can learn better. Dilated convolutions meanwhile increase the receptive field of the network without increasing its parameters. This means that more context can be captured by the network without adding layers. They found that transfer learning (TL) played a key role in making CNN performance better. If the pre-trained models change, then the performance of the model can be improved greatly and the work involved in training a new problem-related model reduced considerably. This paper gives important insights into how advanced CNN architectures and transfer learning, together with pneumonia identification systems, can exist in harmony.

L. Mao, T. Yumeng, and C. Lina (2020) investigated pneumonia detection in chest X-rays using a deep learning approach that combines ensemble Retina Net and Mask R-CNN models. Their study, presented at the 2020 International Conference on Advanced Cloud and Big Data, proposed a novel model that integrates improved versions of RetinaNet and Mask R-CNN, utilizing ResNet-50 and ResNet-101 as backbone networks. The model was validated on a dataset of 26,684 chest radiographs from Kaggle, achieving a recall of 0.813.

Shagun Sharma and Kalpna Guleria (2023) developed a deep-learning model for pneumonia detection from chest X-ray images using VGG-16 and neural networks. Their study, published in Procedia Computer Science, addresses the challenge of quickly identifying pneumonia, particularly in resource-limited settings. The proposed model, leveraging VGG-16 combined with neural networks, achieved impressive results: 92.15% accuracy, 0.9308 recall, 0.9428 precision, and 0.937 F1-score on one dataset, and 95.4% accuracy, 0.954 recall, 0.954 precision, and 0.954 F1-score on another dataset.

Rajpurkar et al. build a deep learning model called "CheXNet" a kind of 121-layer CNN, to detect pneumonia in chest X-ray images by using the ChestX-ray14 dataset. CheXNet is designed to provide higher accuracy than the best radiologist for distinguishing chest X-ray images. Their model not only proved effective at diagnosing pneumonia; it could also differentiate 14 other diseases. A demonstration of deep learning's contribution to medical diagnosis. CheXNet's results served to emphasize the significance of deep learning model in improving and assisting professionals' diagnosis skills.

Patrik Szepesi and László Szilágyi (2022) tackled the challenge of automated pneumonia detection using convolutional neural networks and deep learning. Trained and tested on a dataset of 5,856 labeled chest X-ray images from a Kaggle medical imaging challenge, the model achieved remarkable performance metrics: 97.2% accuracy, 97.3% recall, and 97.4% precision.

Ranjan, A., Kumar, C., Gupta, R.K., and Misra, R. (2022) developed a transfer learning-based approach for pneumonia detection using a customized VGG16 deep learning model. The proposed model was trained on 5,856 chest X-ray images, achieving an accuracy of 98.28%, precision of 0.98, recall of 0.97, and an F1 score of 0.976.

S. Kalgutkar et al. (2021) explored the use of transfer learning for pneumonia detection from chest X-rays. Their study developed and trained Convolutional Neural Network models such as VGG16, ResNet-50, and InceptionV3. The models achieved testing accuracies of 94%, 93.9%, and 93.5%, respectively. The study emphasized the role of advanced computing technology in revolutionizing healthcare by providing quick and accurate diagnostic tools, which are crucial for timely treatment and saving lives, especially in the context of COVID-19.

**Maquen-Niño et al. (2024)** utilized transfer learning with convolutional neural networks (CNNs) to detect pneumonia in chest X-ray images. Using the CRISP-DM data processing methodology, they analyzed 5,856 anteroposterior chest X-rays from Kaggle, dividing the data into training, validation, and testing sets. The study compared the performance of DenseNet, VGG19, and ResNet50V2 models, with ResNet50V2 achieving the highest accuracy of 0.91, followed by DenseNet at 0.87 and VGG19 at 0.86. This comparative analysis highlights the effectiveness of transfer learning models in pneumonia detection.

Zhang et al. (2021) developed a VGG-based model with fewer layers for pneumonia detection from chest X-ray images, addressing the global challenge of limited professional radiologists. The proposed model significantly reduces parameters compared to VGG-16, ResNet-50, Xception, and DenseNet121, while maintaining high performance. It achieved an accuracy of 96.068%, AUC of 0.99107, precision of 94.408%, recall of 90.823%, and an F1 score of 92.851%, outperforming the mentioned models in several metrics.

Sharma et al. (2020) investigated pneumonia detection using CNN architectures with and without dropout layers. Their approach involved two CNN models, each comprising convolutional and max-pooling layers. The first model included two convolution layers with 32 units each, and the second featured layers with 64 and 128 units. Both models used ReLU activation to introduce nonlinearity. Testing accuracy results for these models were 90.68%, 89.3%, 79.8%, and 74.9%.

# **Methodology**

## Dataset

The dataset for this study consists of 5,863 chest X-ray images obtained from Kaggle, a well-known site for data science and machine learning competitions. These images have been meticulously sorted into three main folders: 'train', 'test', and 'val', each with further subfolders categorizing images as 'Pneumonia' or 'Normal'. This categorization makes model training and assessment more efficient. The chest X-ray images were obtained from a retrospective investigation carried out at Guangzhou Women and Children's Medical Center. The investigation focused on children ages one to five years old, and the X-rays were obtained as part of standard clinical care. This provides a dependable and useful dataset for detecting pneumonia with machine-learning algorithms. The training set contains 5,216 chest X-ray images. Out of these, 1,341 images are classified as 'Normal', while 3,875 images are classified as 'Pneumonia'. The substantial number of images in this set ensures that the model has substantial data to learn from and generalize. The validation set includes a total of 16 images, equally separated between the 'Normal' and 'Pneumonia' categories, with 8 images in each classification. This set is utilized during the training process to fine-tune the model and prevent overfitting by supplying an unbiased evaluation of the model's performance on unseen data. The test set consists of 624 images, with 234 images labeled as 'Normal' and 390 images labeled as 'Pneumonia'. This set is used to assess the final performance of the trained models, confirming that the results are robust and reliable.

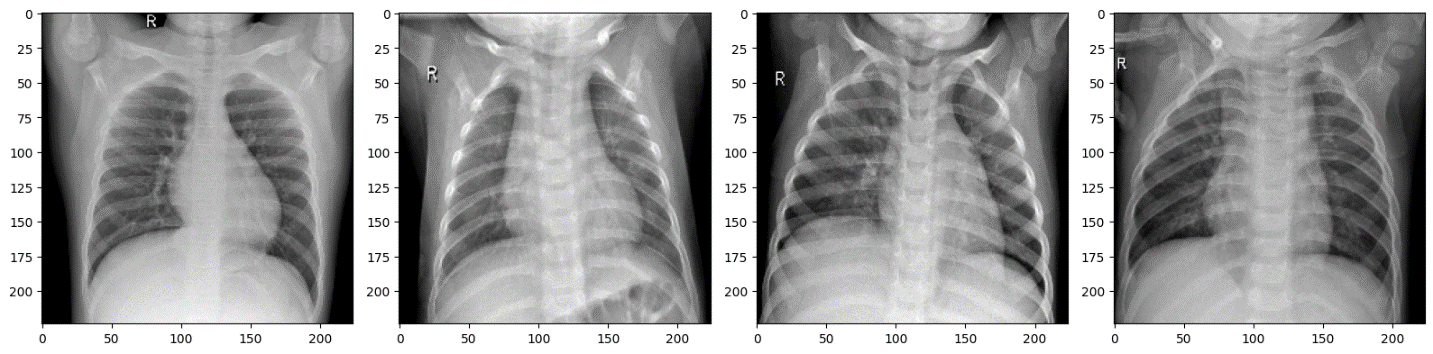
## Data Collection

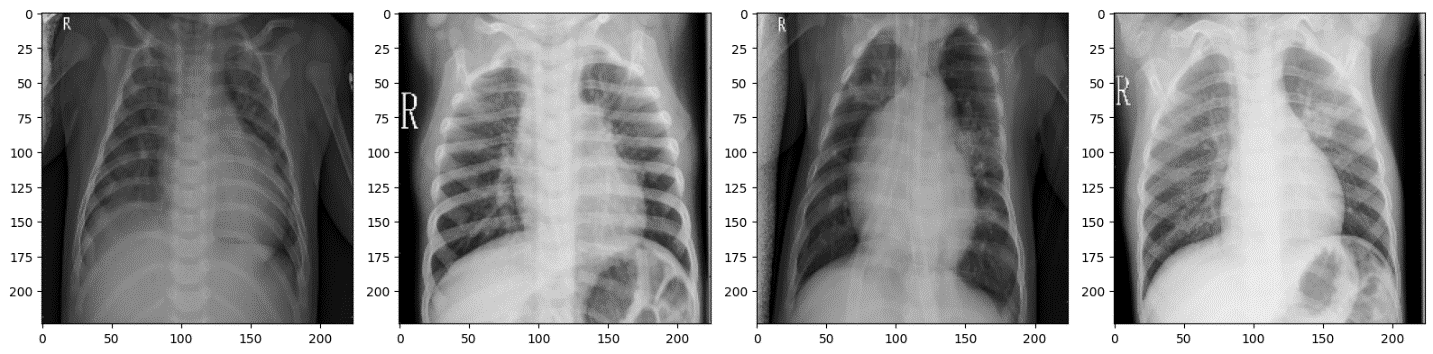
The dataset for the study was carefully divided into folders for training, validation, and testing. Each of these primary folders featured subfolders called 'NORMAL' and 'PNEUMONIA,' corresponding to the health problems depicted in the chest X-rays. images for training were saved in the 'train' folder, images for validation in the 'val' folder, and test images in the 'test' folder. Each image was sorted and categorized according to its category. For example, the training images were labeled as either 'NORMAL' or 'PNEUMONIA,' and the labeling process was carefully executed to ensure accuracy. The same method was used for the validation and test sets to keep everything well-organized and consistent.

## Data Preparation

Each category was divided into three primary groups: training, validation, and testing. Within each of these categories, images were divided separately into two groups: 'NORMAL' and 'PNEUMONIA'. The images were screened by their file extension to ensure only the predesignated image formats appeared in the dataset. For the training set, a list of file paths for 'NORMAL' images was created by iterating through the directory contents and filtering files based on their extensions. A similar method was adopted to acquire images labeled 'PNEUMONIA'. The two lists were then joined, and each image was assigned its appropriate class label ('Normal' or 'Pneumonia'), resulting in a comprehensive dataset.

The same process was applied to the validation and test sets. In the validation set, file paths derived from the 'NORMAL' and 'PNEUMONIA' directories were arranged in an orderly manner before being labeled. This resulted in another dataset containing image paths and classification labels for the validation set. The test set was conducted similarly, eventually producing a third data Frame indexed with these images. Each data frame consisted of two columns: one for the image file paths and another for the class labels. This systematic organization enabled subsequent data processing and model training to proceed smoothly, with each image accurately classified and readily accessible for processing.





## Data Augmentation and Preprocessing

Data augmentation and preprocessing were critical for improving the model's resilience and ability to be generalized. Data augmentation is a means to artificially expand large and diverse datasets by applying various perturbations to the data. In the context of image data, data augmentation is making minor changes to the images to create new, distinct versions of them. Generating many more variations of each image, effectively enlarges the training set, with potential benefits for ML model performance. Augmented data introduces extra variability and diversity in the training set so that the model does not simply memorize its training samples but may genuinely generalize to ensure good performance on novel data. This reduces the risk of overfitting. Training on augmented data allows the model to learn to recognize images in all sorts of positions, sizes, and lighting conditions. It is thus more robust to real-world variations.

The Image Generator class from the Keras library was used to synthetically increase the dataset and add variability. A training data generator object was configured with various augmentation techniques including rescaling pixel values to 1/255, horizontal and vertical shifts within 0.1 ranges, shear transformations up to 0.1, zooming in or out within 0.1, and random horizontal flipping. The "nearest" fill mode handled empty pixels from transformations. Validation and testing datasets were not augmented to maintain the original distribution, though pixel values were rescaled to 1/255 for standardization. Accordingly, validation and test data generators only included this rescaling step.

The dataset was systematically prepared for training, validation, and testing by creating batches of images with appropriate labels extracted from the data frames. The training dataset consisted of images scaled to 224x224 pixels and classified as 'Normal' or 'Pneumonia.' The images were divided into 16 batches, and class labels were treated as binary outcomes. To maintain consistency, the dataset was not shuffled, therefore the images remained in sequential order throughout the training procedure. Similarly, the validation dataset was processed to ensure uniformity in input size and format. Images were resized to 224x224 pixels, labeled accordingly, and organized into batches of 16. A binary class mode was employed to differentiate between the two categories. As with the training data, the validation dataset was not shuffled, providing a consistent reference for assessing model performance during training. The test dataset was configured slightly differently. Images were scaled to 224x224 pixels and associated with their labels. However, the batch size was set to one to allow for an independent examination of each image. This configuration allowed for an in-depth evaluation of the model's accuracy and performance metrics. The test data sequence was fixed and not shuffled, resulting in a consistent and trustworthy evaluation process.

## **Convolutional Neural Network Model**

A Convolutional Neural Network is a specific type of feedforward artificial neural network that is primarily used to analyze visual data. They work by passing input images through a series of convolutional layers that use filters one after the other to discern elements such as edges, textures, and patterns. On top of this, pooling layers reduce dimensions too; decrease computational hours, and provide focus to more important aspects of the image. As the input travels through layers of a network, more complex parts are combined to detect higher-level structures and patterns. In the end, fully connected layers just combine each of these parts to provide a prediction. CNNs are great for tasks such as image classification, object detection, and segmentation because they can learn hierarchical spatial information directly from input images.

### **CNN Layers and Their Function**

The main element of convolutional neural networks (CNN) is a convolutional layer that operates on input data, for example, images. This layer sweeps a set of learnable filters or kernels over the input, sliding window style to produce feature maps which are selective highlighting that emphasize different features, e.g., edges and textures. The operation: the filter is multiplied element-wise with a region of data space and all is then summed up. Convolutional layers take an image as input and reduce the spatial dimensions of this image while not losing much information, extracting useful features and making it easier for later layers to learn and perform complex tasks like classification and detection.

The batch normalization layer scales the activation of previous layers by normalizing their output. Subtraction of the mean and division by the standard deviation of a batch is done to reduce internal covariate shift-seeking stabilization in the distribution of the input layer. Batch normalization normalizes the inputs, which improves learning by stabilizing it and allowing for higher learning rates that ultimately lead to faster convergence. This also serves as a regularizer and can reduce the overhead of Dropout, improving model generalization. The resulting normalized output is then scaled and shifted using learnable parameters, making it possible for the network to tune how each normalization affects input activations.

An activation layer is where the non-linearity comes from, because as we remember earlier layers are linear transformations on top of filters. Activation functions like ReLU, sigmoid, and tanh are utilized to scale the data into 0-1 or -1 to +1 ranges so that we can easily compute its nonlinear relationships. For instance, ReLU rectifies the weights and kills a neuron if it receives an input less than zero, which tackles vanishing gradient issues and greatly improves the training time. The model's performance and convergence are heavily influenced by the activation function used.

The pooling layer is used to reduce the feature map's spatial dimensions. To do this, we can apply pooling operations like max-pooling or average-pooling that select the maximum and averaging value within non-overlapping parts of our feature map. The above process will not only lead to a smaller number of parameters, thus needing lesser computational burden; but also helps in providing translational invariance which makes the network robust against small translations on the input data. Pooling layers help in capturing the most prominent features while minimizing overfitting.

The dropout layer is a regularization technique used to prevent overfitting; it is done by randomly setting a fraction rate of input neurons to zero in each training iteration. By prohibiting the network from activating certain neurons, this technique ensures that it learns more robust and varied features. The dropout rate specifies the fraction of neurons to drop, we often set this value (a hyperparameter) between 0 and 1. During inference, dropout is turned off, and all neurons are combined with their corresponding scaled weights to make sure the model uses everything it has learned for optimal performance.

## **Transfer Learning Models**

Transfer learning works on a pre-trained network like ResNet50V2 and VGG16 which are trained previously with a large dataset such as ImageNet. These models built upon the learned features and patterns of the original dataset will perform better on a new but typically smaller one. A pre-trained model further could be fine-tuned by transfer learning. More commonly, you replace the last layer(s) and re-train them with examples from your new dataset while keeping earlier layers fixed or only minorly adjusted (Fine-Tuning). This regime yields much faster training, and better performing models than those from scratch. The model does not need to start from scratch because it already "knows" certain generic features such as edges, textures, and so on--which may just need to be adapted for the specific task at hand, such as taking X-rays to look for pneumonia.

### ResNet50V2 Structure

ResNet50V2, an advanced architecture from the Residual Networks family, builds on top of what worked for its predecessor ResNet50, and enforces batch normalization within its residual blocks. This design improvement is made to increase the overall performance and stability of learning in a network. ResNet50V2 is similar to ResNet but has an alternative architecture that leverages residual learning to develop meaningful and accurate deep neural networks while keeping entries small. The network has 50 layers with a series of residual blocks, including convolutional layers and shortcuts within each stage. These shortcut connections are critical in allowing gradients to bypass specific layers, preserving signal integrity, and encouraging the learning of identity mappings. The key difference between ResNet50V2 and the original version is the use of a pre-activation residual block rather than the post-activation utilized in typical ResNet blocks. This adjustment raises the network's representational capacity and improves training stability. A typical architecture would look like this: begin with the initial convolutional layer, followed by groups of residual blocks at every stage, then down sample through pool layers at each. The network's final layers include global average pooling and a fully connected layer.

### **VGG16 Structure**

VGG16 is a well-known deep convolutional neural network architecture that is both simple and remarkably effective in design. There are a total of 16 weight layers (13 convolutional and 3 fully connected) systematically organized into five different blocks in this architecture. Convolutional layers are cycled several times in each block followed by a max-pool layer responsible for the "spatial dimension reduction" phase. The convolutional layers use small 3x3 receptive fields which allows many more hidden units in the network, thereby able to capture finer-grained information in input images. These layers are activated using the Rectified Linear Unit (ReLU) function, which introduces the required non-linearity for learning complex patterns and representations. Max-pooling layers were placed at the end of each transition block to carry out down-sampling operations more effectively as well, directing the network's attention primarily on salient features while progressively increasing depths in feature maps and decreasing spatial sizes. After the last convolutional block, we flatten these feature maps and send them through three fully connected layers. The first two fully connected layers are followed by ReLU activations and dropout for regularization, while the last layer uses an activation function that allows classification outputs so it produces probabilities on each class.

## **Convolutional Neural Network Architecture**

To detect pneumonia from chest X-ray images, a convolutional neural network (CNN) was built using the Keras Sequential API. The model's design included many convolutional layers, followed by batch normalization, activation functions, max-pooling layers, and dropout for regularization. The model began with a convolutional layer consisting of eight filters, each having a kernel size of 3x3. The 'valid' padding technique was used, which means no padding was applied to the input image. This initial layer was given a 224x224-pixel input shape with three color channels, which corresponded to the resized input images. Following convolution, batch normalization was used to stabilize and speed up the training process. The Rectified Linear Unit (ReLU) activation function was then employed to add nonlinearity to the model. Following that, a max-pooling operation was done to reduce the spatial dimensions and therefore minimize the computational load. To reduce overfitting, a 0.1 dropout rate was used after this layer, which involved randomly setting a fraction of input units to zero during training. The second convolutional block contained 16 filters with identical kernel sizes and padding schemes. Similar to the first block, batch normalization was used after convolution, followed by a ReLU activation function, max-pooling, and a 0.1 dropout rate. The third convolutional block increases the number of filters to 32, maintaining the same kernel size, padding, and following layers. As the model's complexity rose, the dropout rate was adjusted to 0.2 to improve regularization even further. Following the convolutional and pooling layers, the network had a flattening layer that converted the three-dimensional output to a one-dimensional feature vector. This was followed by a dense layer of 32 neurons using the ReLU activation function. To further prevent overfitting, a 0.3 dropout rate was used. The final layer consisted of a dense output layer with a single neuron that generated a binary classification probability score using the sigmoid activation function. The model was built with the binary cross-entropy loss function, which is suitable for binary classification applications. The Adam optimizer, known for its efficiency in dealing with sparse gradients, was used with a learning rate of 3e-5. The model's performance was evaluated using the accuracy metric, which measures the proportion of correctly classified samples.

### **CNN Model Training**

The model was trained for 20 epochs on the prepared training dataset. The data is iterated over in batches during each epoch; the number of iterations needed to cover the entire dataset, depending on batch size, is set by the step count of 163. Using a different validation dataset, real-time validation was also incorporated into the training methodology. By using this technique, it was possible to track the model's performance on unidentified data continuously and get an idea of its generalization abilities as it was being trained. Specific strategies were used to maximize training while avoiding overfitting. The early stopping technique tracked validation loss and stopped training when improvements plateaued, specifically when there was no further drop in validation loss for three consecutive epochs. Furthermore, the learning rate modification technique dynamically reduced the learning rate when the validation loss did not improve significantly after two consecutive epochs. This adaptive adjustment sought to improve the model's convergence behavior by allowing for smaller weight updates, which are critical for fine-tuning the model's parameters in later stages of training.

## **ResNet50V2 Architecture**

A transfer learning approach was used with the ResNet50V2 architecture, which was initialized with weights from the ImageNet dataset. This pre-trained model served as the architecture's core, exploiting its existing feature extraction capabilities. The input shape for the model was set to 224x224 pixels with three color channels. The final completely connected levels of the ResNet50V2 architecture were removed, allowing for the inclusion of new categorization layers tailored to the individual task at hand. To fine-tune the model for the goal of detecting pneumonia, the pre-trained base's layers were frozen, preventing them from being modified during the training process. This decision was made to maintain the general image features learned from the large ImageNet dataset, which were believed to be useful for the task. The pre-trained base was used as the beginning component to create a new sequential model. This was followed by a flattening layer that reduced the convolutional base's multidimensional output to a one-dimensional vector. Following that, a set of completely connected layers were introduced. The first of these thick layers included 128 units and a corrected linear activation function. This layer also featured L2 regularization with a 0.001 penalty coefficient, which was designed to reduce overfitting by penalizing large weights. To avoid overfitting, a dropout layer with a 50% dropout rate was introduced, which randomly set half of the input units to zero at each training update. This pattern was repeated with a dense layer of 64 units, followed by another dropout layer. Another dense layer of 32 units was added, which included L2 regularization and dropout. The model's last layer was a single unit with a sigmoid activation function, which was chosen due to its suitability in binary classification tasks. The ResNet50V2 model was compiled utilizing the Adam optimizer at a learning rate of 0.0001 and binary cross-entropy as the loss function for binary classification. Accuracy was used to measure the proportion of correctly classified cases.

### ResNet50V2 Model Training

The model was trained on the dataset using the fit technique across 20 epochs.  A validation step was included in the training procedure to monitor the model's performance on unseen data. The training procedure comprised 163 steps per epoch, representing the number of iterations over batches within each epoch. To enhance the training process, two callbacks were implemented: early stopping and learning rate plateau reduction.

## **VGG16 Architecture**

The VGG16 architecture, a well-known convolutional neural network model pre-trained on the ImageNet dataset, served as the foundation for transfer learning. The pre-trained weights from ImageNet were used to initialize the model, guaranteeing that the network has a strong base in feature extraction abilities. To enable network customization for the particular task of pneumonia detection, the upper layers of the model were eliminated. The input shape was adjusted to fit the RGB format of the input images, measuring 224 by 224 pixels with three color channels. The pre-trained VGG16 base was combined with a global max pooling layer, which effectively condensed the spatial dimensions of the feature maps into a single vector by calculating the maximum value across each feature channel. This method preserved the most important properties while lowering the data's dimensionality. All layers in the VGG16 base were made non-trainable, which preserved the learnt weights and ensured that the generalized features derived from the ImageNet dataset remained intact. A custom sequential model was created, with the frozen VGG16 base followed by more fully connected layers. The first dense layer included 128 units and used a rectified linear activation function, which is well-known for its ability to introduce nonlinearity into models. This was followed by a second dense layer of 64 units that used the rectified linear activation function. A third dense layer of 32 units used the same activation technique, gradually reducing the number of units and therefore the dimensionality of the feature space. The model's last layer was a single-unit dense layer with a sigmoid activation function, which was chosen because it is appropriate for binary classification tasks. This activation function returns a probability value between 0 and 1, representing the model's confidence in classifying an input image as 'Normal' or 'Pneumonia'. The same approach as ResNet50V2 was used in compiling the VGG16 model.

### **VGG16 Model Training**

The VGG model was trained on the dataset over 20 epochs, using both training and validation phases. The training utilized batches of data, with 163 steps per epoch, that is the number of batch iterations within each epoch. Early stopping and learning rate plateau reduction callbacks were implemented to avoid overfitting.

# **Evaluation Metrics**

Several metrics were utilized to evaluate the effectiveness of our models in pneumonia detection, including accuracy, precision, recall, and F1-score. These measures offer a thorough grasp of how well our models distinguish between pneumonia and normal cases.

Accuracy: Accuracy is measured as the ratio of correctly predicted instances, including both pneumonia and normal cases, to the total instances. Mathematically, it is expressed as:

Accuracy in the context of pneumonia detection indicates how well the model performs overall in distinguishing between pneumonia and normal instances. However, accuracy by itself could not be an adequate measure of performance if the dataset is imbalanced and one class occurs substantially more frequently than the other. Because it does not fully reflect the model's ability to recognize the less common class, high accuracy can often be misleading.

Precision: Precision is defined as the ratio of accurately predicted positive instances (pneumonia) to the total predicted positive instances. It measures the accuracy of the positive predictions. The formula for precision is:

In the context of pneumonia detection, precision reveals how many of the cases predicted as pneumonia are pneumonia. High precision indicates that the model makes fewer false positive errors, ensuring that patients diagnosed with pneumonia are more likely to truly have the condition.

Recall: Recall, also known as sensitivity or the true positive rate, measures the model's ability to detect all relevant events. It is defined as the ratio of correctly predicted positive instances (pneumonia) to the total number of positive instances and is given by:

In pneumonia detection, recall is important because it indicates how well the model identifies patients with pneumonia. High recall means that the model successfully detects most of the actual pneumonia cases. This is vital for ensuring that patients receive timely and appropriate treatment, which can significantly impact patient outcomes.

F1-score: The F1-score is the harmonic mean of precision and recall, providing a single metric that balances both aspects. It is calculated as:

F1-score provides a balanced evaluation of the model's ability to identify pneumonia patients. It combines precision and recall into a single statistic. A high F1-score implies that the model maintains an appropriate balance between correctly identifying pneumonia cases and minimizing false positives. This balance is critical for delivering a reliable assessment of the model's diagnostic capabilities.

In medical diagnosis, particularly for diseases like pneumonia, recall is frequently highlighted because missing a positive case (false negative) can have serious consequences for patient health.

# **Results**

Throughout the training period, the CNN model showed a steady decrease in loss and an increase in accuracy. The training loss began at 0.5553 and was greatly decreased to 0.2191 by the last epoch, while training accuracy increased from 76.96% to 90.91%. The validation accuracy also improved significantly, moving from 50.00% to 81.25%. Despite significant volatility, notably in the middle epochs, the model's validation loss and accuracy stabilized near the end, proving the effectiveness of the Adam optimizer and adaptive learning rate technique.

|  |  |
| --- | --- |
| Figure 1-CNN Model Loss | Figure 2-CNN Model Accuracy |

During training, the ResNet50V2 model demonstrated a considerable decrease in both training and validation losses while improving accuracy. The training loss started at 1.2305 and decreased to 0.4828 by the last epoch. Similarly, the training accuracy increased from 62.42% to 85.89%. Although the validation accuracy fluctuated, it eventually stabilized and peaked at 93.75%.

|  |  |
| --- | --- |
| Figure 3-ResNet50V2 Model Loss | Figure 4-ResNet50V2 Model Accuracy |

The VGG16 model demonstrated considerable improvements in both training and validation accuracy, as well as a loss reduction. The training loss began at 0.5177 and fell to 0.1686 by the last epoch, while the training accuracy improved from 77.34% to 93.06%. The validation accuracy, while varying, peaked at 81.25%, confirming the model's capacity to generalize well.

|  |  |
| --- | --- |
| Figure 5-VGG16 Model Loss | Figure 6-VGG16 Model Accuracy |

The table below displays the results of all models:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Recall for Class 1 | F1-Score for Class 1 |
| CNN | 0.81 | 0.78 | 0.84 |
| ResNet50V2 | 0.87 | 0.96 | 0.90 |
| VGG16 | 0.85 | 0.92 | 0.88 |

The model performance is evaluated based on Recall. The recall for class 1 (pneumonia) is important in this context since it indicates the model's ability to properly identify positive cases. High recall is especially crucial in medical diagnostics, where failing to detect a disease like pneumonia can have severe consequences.

# **Discussion**

The results based on Recall for class 1 show that ResNet50V2 outperforms with a recall of 0.96, suggesting that it correctly identifies 96% of pneumonia cases. This excellent recall is critical for medical diagnosis because it reduces the danger of false negatives, which could lead to a pneumonia case being missed. The model also has the highest accuracy of 0.87 and an F1-score of 0.90 in class 1, indicating a balanced performance in terms of precision and recall. These data indicate that ResNet50V2 not only identifies the majority of pneumonia cases but also achieves a solid balance between detecting true positives and avoiding false positives.

VGG16 has a recall of 0.92 for class 1, which is slightly lower than ResNet50V2 but still high, indicating great performance in detecting pneumonia cases. The model's accuracy of 0.85 and F1-score of 0.88 for class 1 demonstrate that it is effective in this domain. Although VGG16 has a little lower recall than ResNet50V2. In contrast, the CNN model has a recall of 0.78 for class 1, which is much lower than ResNet50V2 and VGG16. This inferior recall means that the CNN model misses more pneumonia cases than the other two models.

# **Limitations**

The study's shortcomings are mostly related to the dataset and the computing constraints of training complicated models. A big concern is the dataset's class imbalance. Within both the training and test sets, there are much more 'Pneumonia' images than 'Normal' images. This imbalance may induce bias into the model, causing it to preferentially forecast the majority class. As a result, the model's sensitivity and accuracy in detecting the minority class may be lowered, thus compromising its ability to identify less frequent episodes of pneumonia. Additionally, the tiny size of the validation set exacerbates the situation. With only 16 images, the validation set is insufficient to provide reliable and meaningful performance measurements. The limited amount of validation samples limits the ability to accurately assess the model's performance. While the test set is larger, it may still fail to capture the complete range of pneumonia cases and normal variations, reducing the model's generalizability. Furthermore, training deep learning models like ResNet50V2 and VGG16 poses significant computing challenges. Because of their complex topologies, these models require a large amount of computational resources, resulting in lengthy training times. This computational intensity severely limits efficiency and practicality, especially when model retraining occurs frequently or computational resources are restricted.

# **Conclusion**

The broad impact of this research lies in improving the early detection and diagnosis of pneumonia through efficient and accurate deep learning models. This work expands on previous research by improving pneumonia detection recall for class 1 using advanced CNNs and transfer learning techniques. According to the study's findings, the ResNet50V2 model had the highest accuracy (0.87), recall (0.96), and F1-Score (0.90) for detecting pneumonia. The VGG16 model was close behind, with an accuracy of 0.85, recall of 0.92, and F1-Score of 0.88. The baseline CNN model was effective, but had a lower accuracy of 0.81, a recall of 0.78, and an F1-Score of 0.84. Overall, the findings are encouraging; nevertheless, further research is needed to address these constraints and investigate approaches to increase the applicability and scalability of these models for effective clinical deployment.

# **References**

* Jain R, Nagrath P, Kataria G, Kaushik VS, Hemanth DJ. Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning. Measurement. 2020; 165:108046. ISSN 0263-2241. https://doi.org/10.1016/j.measurement.2020.108046
* Varshni D, Thakral K, Agarwal L, Nijhawan R, Mittal A. Pneumonia Detection Using CNN based Feature Extraction. In: 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-7. https://doi.org/10.1109/ICECCT.2019.8869364
* Pant A, Jain A, Nayak KC, Gandhi D, Prasad BG. Pneumonia Detection: An Efficient Approach Using Deep Learning. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-6. https://doi.org/10.1109/ICCCNT49239.2020.9225543
* Rajpurkar P, Irvin J, Ball RL, Zhu K, Yang B, Mehta H, Duan T, Ding D, Bagul A, Langlotz CP, Patel BN, Yeom KW, Shpanskaya K, Blankenberg FG, Seekins J, Amrhein TJ, Mong DA, Halabi SS, Zucker EJ, Ng AY, Lungren MP. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. PLoS Medicine. 2018; 15(11) https://doi.org/10.1371/journal.pmed.1002686
* Elshennawy NM, Ibrahim DM. Deep-pneumonia framework using deep learning models based on chest X-ray images. Diagnostics. 2020; 10(9):649. https://doi.org/10.3390/diagnostics10090649
* Alapat DJ, Menon MV, Ashok S. A Review on Detection of Pneumonia in Chest X-ray Images Using Neural Networks. Journal of Biomedical Physics and Engineering. 2022; 12(6):551-558. https://doi.org/10.31661/jbpe.v0i0.2202-1461
* Zhang D, Ren F, Li Y, Na L, Ma Y. Pneumonia Detection from Chest X-ray Images Based on Convolutional Neural Network. Electronics. 2021; 10(13):1512. https://doi.org/10.3390/electronics10131512
* Maquen-Niño GL, Nuñez-Fernandez JG, Taquila-Calderon FY, Adrianzén-Olano I, De-La-Cruz-VdV P, Carrión-Barco G. Classification Model Using Transfer Learning for the Detection of Pneumonia in Chest X-Ray Images. International Journal of Online and Biomedical Engineering (iJOE). 2024; 20(05):150-161. https://doi.org/10.3991/ijoe.v20i05.45277
* Kalgutkar S, et al. Pneumonia Detection from Chest X-ray using Transfer Learning. In: 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, 2021, pp. 1-6. https://doi.org/10.1109/I2CT51068.2021.9417872
* Ranjan A, Kumar C, Gupta RK, Misra R. Transfer Learning Based Approach for Pneumonia Detection Using Customized VGG16 Deep Learning Model. In: Misra R, Kesswani N, Rajarajan M, Veeravalli B, Patel A, editors. Internet of Things and Connected Technologies. ICIoTCT 2021. Lecture Notes in Networks and Systems, vol 340. Springer, Cham; 2022. https://doi.org/10.1007/978-3-030-94507-7\_2
* Szepesi P, Szilágyi L. Detection of pneumonia using convolutional neural networks and deep learning. Biocybernetics and Biomedical Engineering. 2022; 42(3):1012-1022. https://doi.org/10.1016/j.bbe.2022.08.001
* Mao L, Yumeng T, Lina C. Pneumonia Detection in chest X-rays: a deep learning approach based on ensemble RetinaNet and Mask R-CNN. In: 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 2020, pp. 213-218. https://doi.org/10.1109/CBD51900.2020.00046
* Sharma S, Guleria K. A Deep Learning based model for the Detection of Pneumonia from Chest X-ray images using VGG-16 and Neural Networks. Procedia Computer Science. 2023; 218:357-366. <https://doi.org/10.1016/j.procs.2023.01.018>
* Sharma H, Jain JS, Bansal P, Gupta S. Feature extraction and classification of chest X-ray images using CNN to detect pneumonia. In: 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 227-231. doi: 10.1109/Confluence47617.2020.9057809.

**Code Data:**

from google.colab import drive

drive.mount('/content/drive')

import pandas as pd

import numpy as np

import cv2

import os

import glob

import seaborn as sns

import matplotlib.pyplot as plt

import os

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras import layers, models

from tensorflow.keras.layers import BatchNormalization, Conv2D, MaxPool2D, Flatten, Dense, Dropout

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import callbacks

from sklearn import metrics

import warnings

warnings.filterwarnings('ignore')

**Data Path**

path = '/content/drive/MyDrive/chest\_xray'

**Train, Valid, and Test Path**

train\_path = os.path.join(path, "train")

valid\_path = os.path.join(path, "val")

test\_path = os.path.join(path, "test")

**Data Collection**

train\_normal = glob.glob(os.path.join(train\_path, "NORMAL", "\*.jpeg"))

train\_pneumonia = glob.glob(os.path.join(train\_path, "PNEUMONIA", "\*.jpeg"))

valid\_normal = glob.glob(os.path.join(valid\_path, "NORMAL", "\*.jpeg"))

valid\_pneumonia = glob.glob(os.path.join(valid\_path, "PNEUMONIA", "\*.jpeg"))

test\_normal = glob.glob(os.path.join(test\_path, "NORMAL", "\*.jpeg"))

test\_pneumonia = glob.glob(os.path.join(test\_path, "PNEUMONIA", "\*.jpeg"))

**Data Preparation**

train\_normal = [os.path.join(train\_path, "NORMAL", img) for img in os.listdir(os.path.join(train\_path, "NORMAL")) if img.endswith('.jpeg')]

train\_pneumonia = [os.path.join(train\_path, "PNEUMONIA", img) for img in os.listdir(os.path.join(train\_path, "PNEUMONIA")) if img.endswith('.jpeg')]

train = pd.DataFrame({'image': train\_normal + train\_pneumonia, 'class': ['Normal'] \* len(train\_normal) + ['Pneumonia'] \* len(train\_pneumonia)})

valid\_normal = [os.path.join(valid\_path, "NORMAL", img) for img in os.listdir(os.path.join(valid\_path, "NORMAL")) if img.endswith('.jpeg')]

valid\_pneumonia = [os.path.join(valid\_path, "PNEUMONIA", img) for img in os.listdir(os.path.join(valid\_path, "PNEUMONIA")) if img.endswith('.jpeg')]

valid = pd.DataFrame({'image': valid\_normal + valid\_pneumonia, 'class': ['Normal'] \* len(valid\_normal) + ['Pneumonia'] \* len(valid\_pneumonia)})

test\_normal = [os.path.join(test\_path, "NORMAL", img) for img in os.listdir(os.path.join(test\_path, "NORMAL")) if img.endswith('.jpeg')]

test\_pneumonia = [os.path.join(test\_path, "PNEUMONIA", img) for img in os.listdir(os.path.join(test\_path, "PNEUMONIA")) if img.endswith('.jpeg')]

test = pd.DataFrame({'image': test\_normal + test\_pneumonia, 'class': ['Normal'] \* len(test\_normal) + ['Pneumonia'] \* len(test\_pneumonia)})

**Train Data Distribution**

plt.figure(figsize=(5,4))

sns.countplot(x='class', data = train, palette="viridis")

plt.xlabel("Class")

plt.ylabel("Number of images")

plt.xticks([0, 1], ['NORMAL', 'PNEUMONIA'])

plt.show()

**Valid Data Distribution**

plt.figure(figsize=(5,4))

sns.countplot(x='class', data = valid, palette="viridis")

plt.xlabel("Class")

plt.ylabel("Number of images")

plt.xticks([0, 1], ['NORMAL', 'PNEUMONIA'])

plt.show()

**Test Data Distribution**

plt.figure(figsize=(5,4))

sns.countplot(x='class', data = test, palette="viridis")

plt.xlabel("Class")

plt.ylabel("Number of images")

plt.xticks([0, 1], ['NORMAL', 'PNEUMONIA'])

plt.show()

**Train data - Normal Images**

plt.figure(figsize=(16, 9))

for i in range(4):

img = cv2.imread(train\_normal[i])

img = cv2.resize(img, (224, 224))

plt.subplot(1, 4, i + 1)

plt.imshow(img)

plt.tight\_layout()

plt.show()

**Train data - Pneumonia images**

plt.figure(figsize=(16, 9))

for i in range(4):

img = cv2.imread(train\_pneumonia[i])

img = cv2.resize(img, (224, 224))

plt.subplot(1, 4, i + 1)

plt.imshow(img)

plt.tight\_layout()

plt.show()

**Data Augmentation**

train\_datagen = ImageDataGenerator(

rescale = 1./255,

width\_shift\_range = 0.1,

height\_shift\_range = 0.1,

shear\_range = 0.1,

zoom\_range = 0.1,

horizontal\_flip = True,

fill\_mode = 'nearest'

)

val\_datagen = ImageDataGenerator(rescale = 1./255)

test\_datagen = ImageDataGenerator(rescale = 1./255)

train\_data = train\_datagen.flow\_from\_dataframe(train,

x\_col = 'image',

y\_col = 'class',

target\_size = (224, 224),

class\_mode = 'binary',

batch\_size = 16,

shuffle = False)

valid\_data = val\_datagen.flow\_from\_dataframe(valid,

x\_col = 'image',

y\_col = 'class',

target\_size = (224, 224),

class\_mode = 'binary',

batch\_size = 16,

shuffle = False)

test\_data = test\_datagen.flow\_from\_dataframe(test,

x\_col = 'image',

y\_col = 'class',

target\_size = (224, 224),

class\_mode = 'binary',

batch\_size = 1,

shuffle = False)

**CNN Model Architecture**

model = models.Sequential()

model.add(layers.Conv2D(filters=16, kernel\_size=3, padding='valid', input\_shape=(224, 224, 3)))

model.add(layers.BatchNormalization())

model.add(layers.Activation('relu'))

model.add(layers.MaxPool2D())

model.add(layers.Dropout(0.2))

model.add(layers.Conv2D(filters=32, kernel\_size=3, padding='valid'))

model.add(layers.BatchNormalization())

model.add(layers.Activation('relu'))

model.add(layers.MaxPool2D())

model.add(layers.Dropout(0.2))

model.add(layers.Conv2D(filters=64, kernel\_size=3, padding='valid'))

model.add(layers.Conv2D(filters=64, kernel\_size=3, padding='valid'))

model.add(layers.BatchNormalization())

model.add(layers.Activation('relu'))

model.add(layers.MaxPool2D())

model.add(layers.Dropout(0.4))

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(1, activation='sigmoid'))

**Model Compilation**

model.compile(

loss='binary\_crossentropy',

optimizer=Adam(learning\_rate=3e-5),

metrics=['accuracy']

)

**Training Callbacks**

early\_stopping = callbacks.EarlyStopping(

monitor='val\_loss',

patience=2,

min\_delta=1e-7,

restore\_best\_weights=True,

)

plateau = callbacks.ReduceLROnPlateau(

monitor='val\_loss',

factor = 0.2,

patience = 2,

min\_delt = 1e-7,

cooldown = 0,

verbose = 1

)

**Model Training**

mdl = model.fit(train\_data, epochs = 10, validation\_data=valid\_data, callbacks=[early\_stopping, plateau])

**Model Loss Visualization**

plt.plot(mdl.history['loss'])

plt.plot(mdl.history['val\_loss'])

plt.title('Model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper right')

plt.show()

**Model Accuracy Visualization**

plt.plot(mdl.history['accuracy'])

plt.plot(mdl.history['val\_accuracy'])

plt.title('Model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()

**Model Evaluation**

score = model.evaluate(test\_data, steps = len(test), verbose = 0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

**Label Encoding**

num\_label = {'Normal': 0, 'Pneumonia' : 1}

Y\_test = test['class'].copy().map(num\_label).astype('int')

predictions = model.predict(test\_data, steps=len(test\_data), verbose=0)

pred\_labels= np.where(predictions>0.5, 1, 0)

from sklearn import metrics

**Classification Report**

print(metrics.classification\_report(Y\_test, pred\_labels, labels = [0, 1]))

**Confusion Matrix Heatmap**

confusion\_matrix = metrics.confusion\_matrix(Y\_test, pred\_labels)

sns.heatmap(confusion\_matrix, annot=True, fmt="d")

plt.xlabel("Predicted Label", fontsize= 12)

plt.ylabel("True Label", fontsize= 12)

plt.show()