Imaging and Radiation Research 2024, 7(1), 6257.
https://doi.org/10.24294/irr6257

Article

Classification of X-ray images and model evaluation

Aya Naser[1], Şafak Bera Şafak[1], Emrah Utkutağ[1], Simge İnci Sin[1], Sena Sude Taşkin[1], İrem Koca[1], Refika Sultan Doğan[1,2,*]

[1] Department of Bioengineering, Faculty of Life and Natural Sciences, Abdullah Gül University, Kayseri 38080, Turkey
[2] Biomedical Instrumentation and Signal Analysis Laboratory, Faculty of Engineering, Abdullah Gül University, Kayseri 38080, Turkey
*** Corresponding author:** Refika Sultan Doğan, refikasultan.dogan@agu.edu.tr

CITATION

Naser A, Şafak SB, Utkutağ E, et al. Classification of X-ray images and model evaluation. Imaging and Radiation Research. 2024; 7(1): 6257. https://doi.org/10.24294/irr6257

ARTICLE INFO

Received: 7 May 2024
Accepted: 13 June 2024
Available online: 21 November 2024

Abstract: Inflammation of the lungs, called pneumonia, is a disease characterized by inflammation of the air sacs that interfere with the exchange of oxygen and carbon dioxide. It is caused by a variety of infectious organisms, including viruses, bacteria, fungus, and parasites. Pneumonia is more common in people who have pre-existing lung diseases or compromised immune systems, and it primarily affects small children and the elderly. Diagnosis of pneumonia can be difficult, especially when relying on medical imaging, because symptoms may not be immediately apparent. Convolutional neural networks (CNNs) have recently shown potential in medical imaging applications. A CNN-based deep learning model is being built as part of ongoing research to aid in the detection of pneumonia using chest X-ray images. The dataset used for training and evaluation includes images of people with normal lung conditions as well as photos of people with pneumonia. Various preprocessing procedures, such as data augmentation, normalization, and scaling, were used to improve the accuracy of pneumonia diagnosis and extract significant features. In this study, a framework for deep learning with four pre-trained CNN models—InceptionNet, ResNet, VGG16, and DenseNet—was used. To take use of its key advantages, transfer learning utilizing DenseNet was used. During training, the loss function was minimized using the Adam optimizer. The suggested approach seeks to improve early diagnosis and enable fast intervention for pneumonia cases by leveraging the advantages of several CNN models. The outcomes show that CNN-based deep learning models may successfully diagnose pneumonia in chest X-ray pictures.

Keywords: convolutional neural networks; image classification; image processing; medical imaging; artificial intelligence

1. Introduction

Pneumonia is an infection-related inflammation of the lungs' air sacs (alveoli). Alveoli is found at the ends of the respiratory bronchioles, allowing the exchange of oxygen and carbon dioxide gas. It may be referred to as bronchopneumonia if the airways are also affected. Pneumonia occurs when these air sacs are filled with fluid or pus, hindering the gas exchange process, resulting in difficulty breathing and a cough reflex. It can affect the lung in one or more sites (sometimes known as "double" or "multilobar" pneumonia). pneumonia can be caused by a variety of factors, the majority of which are infectious [1].

Pneumonia is typically caused by virus or bacterial infection from the environment or from another person. Infection can be spread from person to person by direct touch (typically through the hands) or through inhaling droplets in the atmosphere from coughing or sneezing. Secondary infection from bacteria like Staphylococcus aureus can occasionally occur in a person who has a viral illness, such as the influenza virus, while they are ill. Pneumonia can also be caused by a parasite,

fungus, or yeast. Aspiration pneumonia is brought on by a foreign substance entering the lungs through the throat. Typically, this material is food or vomit, which causes irritation to the airways and lung tissue and raises the risk of bacterial infection [1].

Pneumonia can occur at any age. However, it seems to affect small children and the elderly more frequently. Pneumonia can be more serious for some people due to pre-existing lung conditions, poor nutrition, swallowing issues, other chronic health issues, or immune system issues. Pneumonia is more likely to occur among smokers and those who are around tobacco smoke. People who have not had the annual influenza vaccination or the pneumococcal vaccines Prevnar13® and/or Pneumovax®23 are also at a higher risk of developing lung infections [2].

People who have pneumonia frequently experience coughs, fever or chills, respiratory problems, low energy, and poor appetite. Chest pain, nausea, and/or diarrhea can all occur frequently. Without a cough or fever, pneumonia is possible. Symptoms may appear suddenly or develop gradually over time. An individual who has a viral upper respiratory illness (cold) may occasionally experience a new fever and deterioration, which indicates the beginning of the secondary bacterial infection [2].

The medical professional will take the symptoms into account and do a physical assessment. Pneumonia can cause the noises in the lung to be diminished or irregular. Blood tests may be performed to check the white blood count and other measurements that may be off related to a disease. A chest x-ray is frequently taken in order to identify the site or regions affected by pneumonia. Sometimes a CT scan—often referred to as a "cat" scan—is performed for more precise computerized x-rays [3]. Sputum, also known as phlegm or mucus, is excreted during coughing, and may be tested and cultured to determine the presence of any germs or viruses. More frequently, tests are performed on patients who are ill enough to be hospitalized for the most likely viruses and bacteria. A procedure known as flexible bronchoscopy may be used to extract a sample of mucus from the lung through the airways if a patient is not improving, has a serious infection, or is at a high risk of developing a rare infection [3]. It can be difficult to determine what type of infection (for example, which bacterium) is causing pneumonia. This could be a result of the tests being insufficiently precise or because you might have undergone treatment prior to the testing. However, the healthcare provider will work with the patient to choose a course of action based on the most likely cause determined by the patient's information, the types of infections that are prevalent in the patient's community, and the types of infections that the patient may be more susceptible to if they already have a health issue [3]. The prognosis for pneumonia and the seriousness of the patient's condition both influence treatment. Antibiotics that are efficient against the most likely microorganisms causing the infection are typically given. The patient may require medications to address more resistant germs if the pneumonia occurred while they were a patient in a hospital or another healthcare facility such as a nursing home [3]. Because the symptoms of the illness are not readily apparent on CT or X-ray scans, pneumonia objectively and automatically detecting poses a significant issue in medical imaging. Chest X-rays (CXR) or computed tomography (CT) scans are frequently used to detect pneumonia; the former is the most used method because it is more affordable and widely available worldwide. Due to its great speed and objective, reproducible judgment, the computer can assist the human expert in making the

diagnosis of pneumonia because the symptoms of pneumonia in X-ray images are not always obvious or readable to the human eye [4]. For the purposes of computer vision, researchers have suggested various CNN-based deep networks for image classification, picture segmentation, object recognition, and localization. CNNs have proven to be extremely effective and successful at resolving medical issues as well, including the diagnosis of Alzheimer's disease, the classification of skin lesions, the identification of breast cancer, and the segmentation of brain tumors [5].

Machine learning and deep learning methods have recently been developed in numerical computing for medical picture analysis. Since they offer great accuracy and amazing outcomes when compared to other models, convolutional neural networks (CNNs) are the most favored and well-liked deep learning models with superior accomplishments in the medical imaging sector. On the basis of the CNNs, numerous research using various methods to perform chest X-rays was undertaken to identify pneumonia. A CNN model, for instance, was proposed by Stephen et al. and trained to categorize pneumonia using chest X-rays. The proposed model's accuracy, according to the authors, is 95.31% [6]. Convolutional neural networks (CNNs) were used in picture classification tasks, and their use of Deep Learning (DL) models proved their potential for doing so. This feature-extraction approach necessitates transfer learning techniques, in which pre-trained CNN models first learn the generic features on massive datasets like ImageNet, then transfer those features to the desired job. The process of important feature extraction is greatly facilitated by the availability of pre-trained CNN models like AlexNet, VGGNet, Xception, ResNet, and DenseNet. Additionally, the classification of photos when using highly-rich extracted features performs better. The datasets used, which include 112,120 anterior chest X-ray pictures from 30,085 patients, are also freely accessible on the Kaggle platform [7].

Since deep CNN models like ResNet, Xception, or DenseNet have millions of trainable parameters, training them from the start takes a large amount of data because the model wouldn't be sufficiently generalized with a small dataset. A TL approach can be used to reuse these models with their pre-trained weights. A pre-trained CNN model is reused in TL, a helpful machine learning technique, to use its weights as initialization for a new CNN model that will be used for a different task. The two main approaches to use the TL from a model are to either reuse the model to do Fine-Tuning (FT) or to reuse the model as a feature extractor and use an entirely new classifier. FT is a strategy that modifies the new Fully Connected (FC) layers of the classifier as well as particular layers of the CNN, such as convolutional layers, somewhat [8].

2. Materials and method

2.1. Dataset

A dataset of chest X-ray images labeled as Pneumonia and normal was used for detection with CNNs (**Figure 1** and **2**). The "Chest X-Ray Images (Pneumonia)" dataset was utilized in the code. The dataset consists of a series of chest X-ray images divided into 4273 "PNEUMONIA" images and 1583 "NORMAL" images categories in total. X-ray images of patients with pneumonia are included in the "PNEUMONIA" category, whereas images of healthy people without pneumonia are included in the "NORMAL" category.

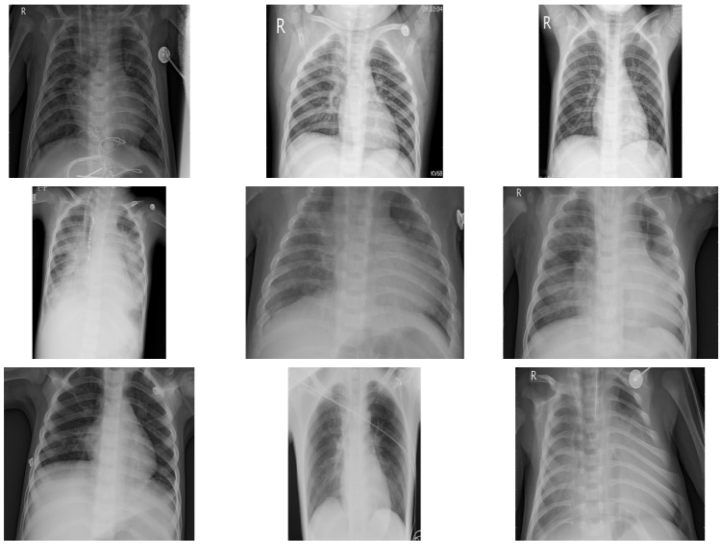

Figure 1. Chest X-Ray images of pneumonia patients.

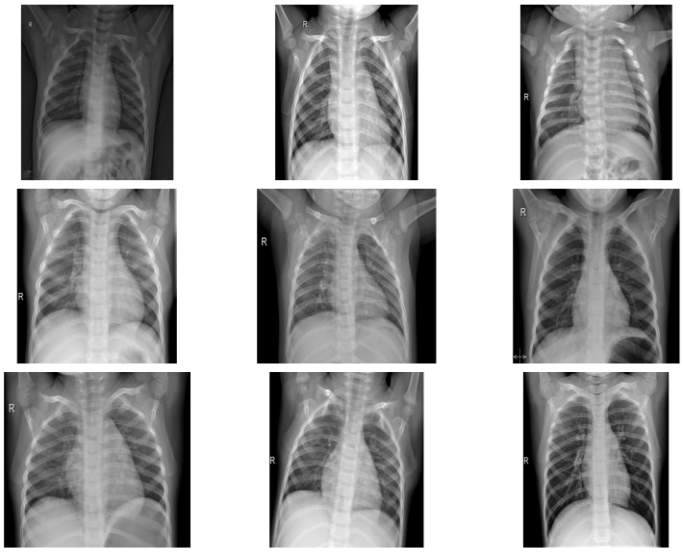

Figure 2. Chest X-Ray images of normal (non-pneumonia) patients.

2.1.1. Data pre-processing

The necessary pre-processing steps were applied on the X-ray images as follows: Before training, the images are modified for better training of a convolutional neural network. The Keras ImageDataGenerator function was used to prepare X-ray images for training a convolutional neural network (CNN). Image data augmentation is a method for enhancing the variety and variability of training data for deep learning models. The ImageDataGenerator provides random variations to the images during training by changing transformation parameters including rotation range, width shift range, shear range, and zoom range. Furthermore, the choice of samplewise centering and standardization helps in separately normalizing the pixel intensity of each image. However, this ImageDataGenerator combines these techniques to generate enhanced image batches that may be used to train deep learning models, enhancing generalization and robustness by offering a more diverse and variable training set.

2.1.2. Data split

The dataset was divided into three subsets: Train, validation, and test sets (**Table 1**).

Table 1. The number of input images with normal and pneumonia chest X-ray images.

	Normal	Pneumonia
Train	1341	3875
Test	234	390
Val	8	8

2.1.3. Data augmentation (normalization and resizing)

To standardize the input distribution and facilitate model training, the pixel values in each set of images were fixed in a range from 0 to 1, and the images were resized to a consistent size. This analysis provided valuable information about the intensity and spread of pixel values across the dataset, aiding our understanding of the data characteristics.

2.2. Channel conversion

Our chosen pre-trained neural network requires three-channel input. Hence, we utilized the generator to convert the single-channel grayscale X-ray images into a three-channel format. This conversion was achieved by replicating the pixel values across all three channels.

2.2.1. Convolutional neural networks (CNNs) model building

CNNs are deep learning architectures that excel at extracting features from images before classifying them. By employing convolution filters with different dimensions or values, various features can be extracted from the images. Features are detected using ReLu activation at each pixel and enhanced with MaxPool layers. The stride parameter determines the distance between each filter, while the padding parameter determines whether the mesh should consider boundary pixels. Zero padding applied to the neural network to provide information from image borders. The outputs from these operations were combined and passed through Dense layers. A sigmoid activation function was used to determine the final layer of the network, the class to which the image belongs.

2.2.2. CNNs model training

Training images were fed to CNN. Used a weighted loss function, such as binary cross-entropy, to balance the contribution of several classes to the loss calculation.

A sequential CNN framework for identifying pneumonia (pna) in clinical images is shown in **Figure 3**. Four pre-trained neural network (CNN) models are included in the framework: InceptionNet, ResNet, VGG16, and DenseNet. These pre-trained models are effective tools for feature extraction and picture segmentation tasks since they have been refined and optimized on huge datasets. The framework can benefit from the information and representation acquired from various datasets thanks to the use of pre-trained CNN models. The system gains from the capacity to extract useful

information from input photos and generate precise predictions based on these retrieved features by adding these models. A CNN architecture called InceptionNet introduces the idea of inception modules, which successfully capture characteristics at various spatial scales. ResNet, on the other hand, makes use of residual connections to help deep network training without experiencing vanishing gradient problems. A well-liked CNN architecture noted for its ease of use and effectiveness in image classification tasks is VGG16. Finally, DenseNet uses densely interconnected blocks to enhance information flow across layers, making layer use simpler and lowering the possibility of overpopulation. The inclusion of these four pre-trained CNN models in **Figure 3** shows that the design takes advantage of their unique strengths and different design methods to improve the accuracy and reliability of pneumonia diagnosis. By combining the capabilities of these models, the design aims to capture a wide range of image features and provide powerful predictions for the detection of pneumonia, thus facilitating early diagnosis and timely medical intervention.

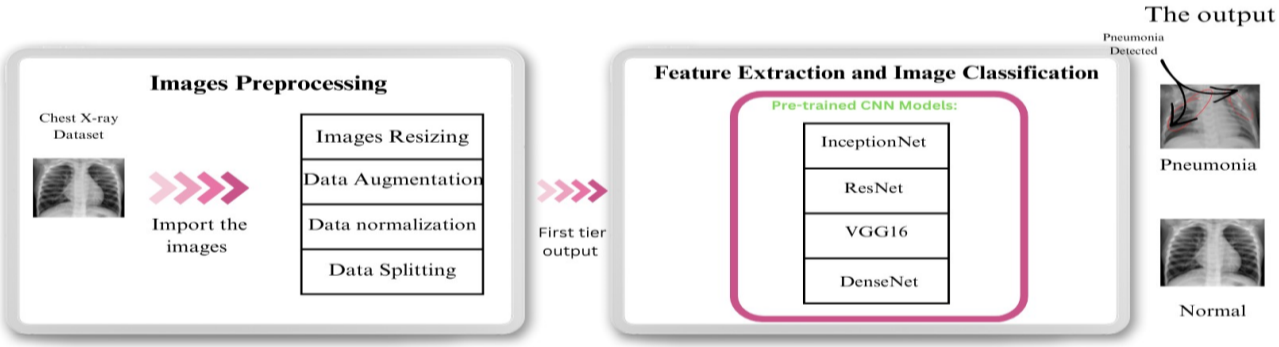

Figure 3. The deep learning framework that is suggested for diagnosing pna.

$$L^{W}_{cross_{entropy}}(x) = -\left(w_p y \log\left(f(x)\right) + w_n (1-y) \log\left(1-f(x)\right)\right) \quad (1)$$

Loss function formula explained in Equation (1). Class weights were applied for class 0 and class 1, which were applied to tolerate the imbalance condition. Weight for class 0: 0.74, weight for class 1: 0.26 were found. The Adam optimizer was used to update the model weights and minimize the loss value. Transfer Learning was employed using DenseNet, a convolutional neural network architecture. DenseNet is characterized by dense connections between layers, where each layer is connected to all the subsequent layers in the network. This connectivity pattern facilitates the flow of gradients throughout the network, enabling efficient feature propagation.

2.2.3. Transfer learning models training

InceptionNet

Google created the convolutional neural network (CNN) architecture known as InceptionNet to outperform earlier CNNs. It is well-known for using inception modules, which are layers' building blocks that learn a mix of local and global features from the input data. There are 22 layers in the InceptionNet architecture, including fully connected, pooling, and convolutional layers. The use of "Inception modules", parallel convolutional blocks with various filter sizes (1 × 1, 3 × 3, and 5 × 5) and pooling operations, is one of its key innovations. The network can learn spatial and

temporal features from the input data using these modules, which are made up of smaller convolutional and pooling layers that are combined. Other deep convolutional neural networks can take longer to train than InceptionNet because of the way it was designed. It serves as the foundation for well-known neural network architectures like Inception-v4 and Inception-ResNet and has been used in image classification, object detection, and face recognition. The InceptionNet model has roughly 5 million parameters total. But over time, a few variations, and more advanced versions of InceptionNet, including InceptionV2, InceptionV3, InceptionV4, and Inception-ResNet, each with a different number of parameters, were developed. Depending on the depth and complexity of the architecture, these variants have anywhere between tens of millions and hundreds of millions of parameters. The complexity of the computations is typically increased in the deeper versions in exchange for performance gains [9].

Conventional convolutional neural networks often use convolutional and pooling layers to extract features from the input data. This problem is resolved by Inception blocks' modular nature, which allows the network to learn various feature maps at varied scales. These feature maps are then concatenated to produce a more comprehensive representation of the input data. To help with tasks like picture categorization, this enables the network to collect a variety of features, both high-level and low-level. Using inception blocks allows the InceptionNet architecture to learn a larger range of features from the input data, which improves the network's performance on tasks like picture categorization. The Inception network is made up of convolutional design configurations in the form of recurring patterns known as Inception modules [10].

- Input layer
- 1 × 1 convolution layer
- 3 × 3 convolution layer
- 5 × 5 convolution layer
- Max pooling layer
- Concatenation layer

The Inception modules are a key part of the InceptionNet convolutional neural network architecture. These built-in blocks, known as layers, are designed to extract a combination of local and global properties from the incoming data. By combining smaller convolutional and pooling layers from inception modules, the network may extract spatial and temporal properties from the input data. The objective of the inception module is to learn many feature maps at different scales, then integrate them to produce a more comprehensive representation of the input data. The network may consequently gather a wide range of low-level and high-level information that can be useful for tasks like picture categorization. Depending on the desired level of complexity and the volume of input data, inception modules can be added to the network at different points. They can also be altered by altering the convolutional and pooling layers' size and number, as well as the nonlinear activation function's type.

The necessary TensorFlow libraries and classes are imported. The input shape and number of classes for images are specified, including pneumonia and normal. To customize the pre-trained InceptionNet model for the classification task, additional custom layers are loaded. The model is put together using the Adam optimizer and

categorical cross-entropy loss after the pre-trained layers have been filled in. To perform real-time data augmentation during training, ImageDataGenerator is set up. The heap size and training period count are then set, along with the indexes for the train, validation, and test sets of data. Train loads and preprocesses validation and test data using flow_from_directory. Tags are categorically encoded, and images are resized. The model is trained by passing the train and validation data, the number of steps per period, and the validation steps through the fit function. Using the evaluate and print test loss and accuracy option, the model is evaluated on the test data after the training.

ResNet

Using the layer inputs as a guide, the weight layers of a residual neural network develop residual functions. Identifier mappings are carried out by skip connections in a residual network, which is added to the layer outputs. The strategy behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping [11]. Thus, let the network fit instead of using, say, the initial mapping of $H(x)$,

$$\mathrm{F}(x) = H(x) - x \; which \; gives \; H(x) = F(x) + x$$

In this section, we'll go through how to classify images in Keras using ResNet50. Import libraries: Import Keras and other necessary libraries. Install ResNet50 and use Keras to initialize the ResNet50 model. The intended shape of the input photos is specified by the input shape argument. Set include top=False to prevent ResNet50's fully linked layers from being included in the model. For certain classification jobs, this enables customization. Use pre-trained weights: To start the model with pre-trained weights from the ImageNet dataset, set weights = 'imagenet'. These weights offer a place to start for accurate categorization.

An effective tool for picture categorization in Keras is ResNet50. By loading the model, setting it up, and using weights that have already been trained, accurate results can be obtained with minimal work. The model must be assembled, and the dataset must be ready, for the implementation to be complete. Using the Keras API of TensorFlow, additional code constructs a neural network model based on the ResNet architecture. The model is made up of numerous fully connected layers that are placed on top of a pre-trained ResNet basic model. The model is intended to solve a binary classification problem in which one of two classes is to be determined for each input. For enhancing model performance and avoiding overfitting, the design contains additional layers for feature extraction and editing. Accuracy, precision, and recall are just a few of the criteria that have been established to gauge how well the model is working. These metrics shed light on several facets of the categorization performance of the model. Overall, the code builds a binary classification model based on ResNet, compiles it using Adam optimizer and binary cross-entropy loss, and sets evaluation metrics to track the model's performance during training and evaluation.

VGG16

The VGG model, commonly known as VGGNet, is referred to as VGG16. It is a 16-layer convolutional neural network (CNN) model. When using numerous smaller layers rather than a single large layer, the decision functions are improved, and the

network can converge more quickly because there are more non-linear activation layers present [12]. In the top five tests, the model performs 92.7% accurately in ImageNet, a dataset of over 14 million images divided into 1000 classes [8].

VGG16 operating principle is based on the idea that stacking many smaller layers increases the network's decision-making power. The model can capture and learn complicated characteristics from input photos by employing multiple convolutional layers with non-linear activation functions. The input image for VGG16 is typically 224 × 224 pixels in size and is processed through several convolutional layers. Each convolutional layer extracts various features at various degrees of abstraction from the input by applying a set of learnable filters to the input [11]. These filters have been trained to recognize forms, edges, and other visual patterns. A non-linear activation function, such as the Rectified Linear Unit (ReLU), is used after each convolutional layer. By introducing non-linearity, the activation function enables the model to learn intricate connections between the retrieved data. The max-pooling layers in VGG16 also help to decrease the spatial dimensions of the feature maps by choosing the maximum value within a constrained area. Using methods like backpropagation and gradient descent, VGG16 learns to modify the weights and biases of its layers during training. To reduce the discrepancy between the model's anticipated outputs and the ground truth labels provided in the training data, the model is trained.

DenseNet

Each layer's features and gradients are strengthened by DenseNet by using the top classifier to oversee the other layers through feature connection. The efficiency of features from each hidden layer is less enhanced or verified by the top classifier, which is more likely to evaluate the effectiveness of the sum of input features for the final layer [13].

It uses convolutional layers, pooling, and dense blocks to obtain significant representations from input images. The input images' sizes are specified by the input shape option, and the final classification layers are disregarded if include top is set to False. In addition, pre-trained weights from the ImageNet dataset are loaded by setting weights to "imagenet", assisting with model initialization. After the convolutional layers, the pooling operation is determined by the pooling parameter. The base_model.summary() method displays the architecture's layers and parameter counts. The efficacy of DenseNet in many computer vision applications is mostly a result of its dense connections and configurable parameters [13].

2.3. Performance metrics

To measure performance, it is necessary to evaluate the trained model using the validation dataset. Accuracy, precision, recall, and F1-score are evaluation metrics for binary classification issues. The effectiveness of the classification models can be assessed in several different ways [14]. To assess the performance for categorizing colon polyps, we employed accuracy (Equation (2)), precision (Equation (3)), recall (Equation (4)), and f-measure (Equation (5)) measures.

$$Accuracy = \frac{(t_p + t_n)}{(t_p + f_p + f_n + t_n)} \quad (2)$$

$$Precision = \frac{t_p}{(t_p + f_p)} \tag{3}$$

$$Recall = \frac{t_p}{(t_p + f_n)} \tag{4}$$

$$f - measure = \frac{2}{(\frac{1}{Precision} + \frac{1}{Recall})} \tag{5}$$

All measures distinguish the correct classification of labels within different classes. Recall is a function of its correctly classified examples and its misclassified examples.

2.4. Model fine-tuning

Hyperparameters can be tuned, architecture changed, or other techniques such as normalization can be used to improve the performance of the model. Architectures such as VGG16, ResNet, InceptionNet can be used (**Figure 4**).

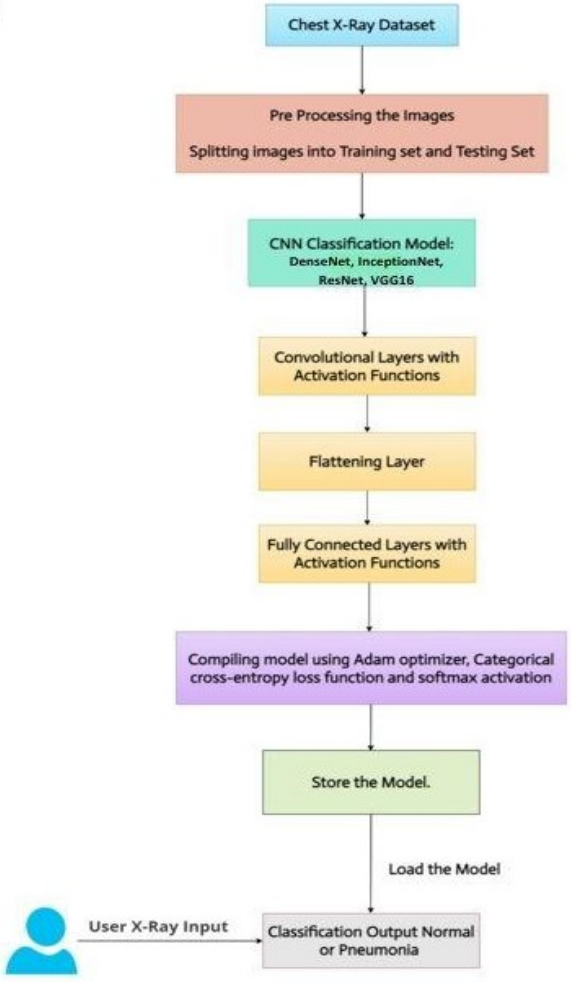

Figure 4. Model of pneumonia detection using convolutional neural networks from chest X-ray images.

3. Results and discussion

Table 2 shows a summary of the distribution of X-ray images for the training, testing, and validation datasets. The table has two categories: "Pneumonia" and "normal lung", which represent the presence or absence of pneumonia on X-ray images.

Table 2. Length of the input files with normal and pneumonia chest X-ray images.

	Normal	**Pneumonia**
Train	1341	3875
Test	234	390
Val	8	8

Table 3. Breakdown of the numbers in each dataset.

	Train	**Test**	**Validation (Val)**
Pneumonia	3875	390	8
Normal Lungs	1341	234	8

In the training list, there are 3875 X-ray images labeled as pneumonia and 1341 X-ray images labeled as normal lung. These images may be used to train a machine learning model to distinguish between pneumonia and normal lung X-ray images. The test dataset contains 390 pneumonia X-ray images and 234 normal lung X-ray images (**Table 3**). This dataset is often used to test the performance of a model trained on unobserved data. Validation data, with only 8 images for each class, is usually used to adjust or validate the model after the training and testing phases. The small size of the validation dataset indicates that only a small subset of the data is used for final model testing or hyperparameter tuning. These numbers provide an overview of how X-ray images are distributed in different datasets for the task of diagnosing pneumonia.

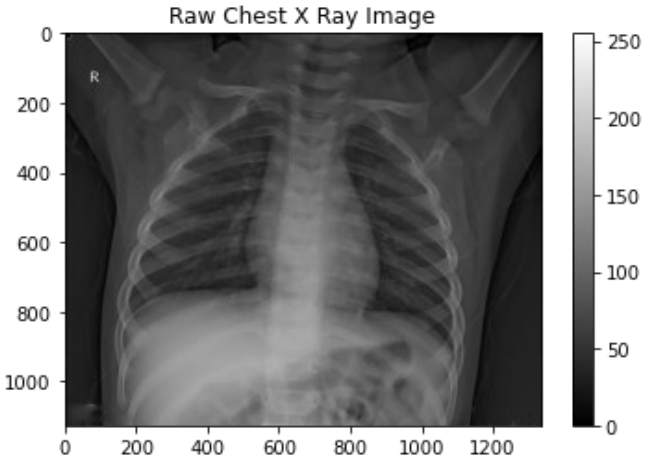

Figure 5. Pixel distribution and values.

According to **Figure 5**, the X-ray images in this study have a resolution of 1128 pixels and a height of 1336 pixels, with one color channel The pixel density ranges

from a minimum value of 0.00000 to a value of a maximum of 255.00000, representing the darkest and lightest values of the pixel, respectively. Graph 1 offers a pixel density distribution plot, which indicates the frequency of different pixel intensities inside the snap shots. The graph offers records for the X-ray pix. The measured pixel density is calculated as 73.2978, this means that a mean pixel density. However, the same old deviation of 38.1653 shows the distinction in pixel depth of many of the photographs. This statistical information is important for know-how the houses of X-ray pix and can help to develop picture processing techniques for medical applications.

3.1. Image preprocessing

In the context of image processing using the Python programming language, a set of parameters is used to add and manipulate images. These parameters include rotation range, width_shift_range, shear range, zoom range, and samplewise_center. The rotation range parameter allows random rotation of images up to 20 degrees, introducing contrast and improving the diversity of the dataset. The width_shift_range parameter enables the shift of image pixels by 10% of the image width, to produce a small change that simulates a different view or position. The shear range parameter introduces shear changes to images, enabling image shapes to be distorted by up to 10% of the image width. The zoom range parameter allows one to randomly zoom in or out of images by up to 10%, providing an additional level of flexibility and scale. Finally, the samplewise_center parameter sets the pixel values of each image by normalizing the images and subtracting the median value of the dataset. Together these parameters contribute to data optimization and manipulation techniques, facilitating the training and evaluation of machine learning models by expanding the data and improving its diversity.

3.2. Separate generator for valid and test sets

The reason why we cannot use a single generator for validation and analysis of the training data is because of the general operation performed by the generator. In training, the generator normalizes each image using batch statistics. This means that the mean and standard deviation used for normalization are calculated based on the images in the batch being processed. However, when it comes to validation and testing, we want to simulate a real situation where we deal with images individually and not in groups. In this case, the model should not have any knowledge about the test data beforehand. If we were to use a single generator with “batch normalization” for validation and analysis, it would provide information about the test data implicitly by allowing it to compute several batches. To avoid this issue, we need to use a separate generator for validation and test data. This generator needs to simplify the incoming check information and the usage of statistics taken from the training middle. Using popular facts used while training, we ensure that the version obtains regular and independent information during validation and assessment. This approach helps to hold the integrity of the assessment procedure, because the model is tested on random statistics without prior know-how or advantage received from the test institution. It permits us to accurately look at the version’s performance and determine how properly it generalizes to actual-international information. By following this method, we

expand a robust and independent evaluation framework for the overall performance of our version.

The photo in **Figure 6** is an X-ray image, that is represented in a virtual layout. The dimensions of this X-ray picture are precise as 180 pixels in width and 180 pixels in top. This method consists of a grid of one hundred eighty pixels horizontally and one hundred eighty pixels vertically. Additionally, it is cited that the X-ray picture consists of an unmarried color channel. In the context of grayscale pix, inclusive of X-rays, an unmarried coloration channel represents the depth or brightness values of each pixel. This indicates that the photograph is represented in shades of gray instead of containing coloration facts. By having these dimensions and a single-color channel, the X-ray photo in **Figure 7** can be processed and analyzed using numerous laptop imaginative and prescient techniques. Understanding the characteristics of the photo, which include its size and color channel, is critical for in addition evaluation, interpretation, and capability application of photograph processing algorithms or device mastering models within the scientific area. The pixel density graph in **Figure 8** shows the distribution of pixel intensity in the X-ray image shown in 6. The graph shows the frequency or number of pixels at different intensity levels. The observed statistics provide some details of the image, where the maximum and minimum pixel values of 2.5969 and −2.4856 respectively indicate the brightest and darkest pixels. With a total value of 0.0000, the overall strength appears to be around the neutral level. Furthermore, a standard deviation of 1.0000 suggests a wide range of pixel intensities throughout the image. These statistical studies contribute to a better understanding of image characteristics and can facilitate image processing techniques and medical applications.

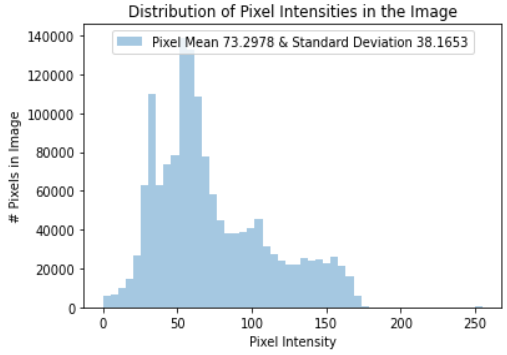

Figure 6. Pixel intensity distribution graph.

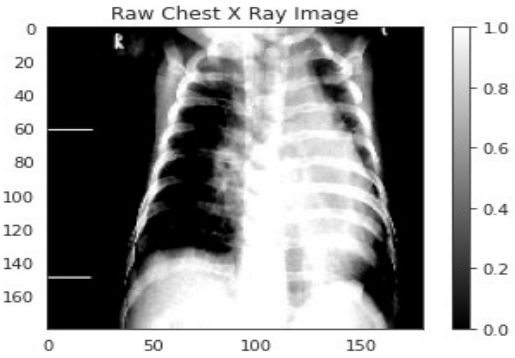

Figure 7. Raw chest X-ray image.

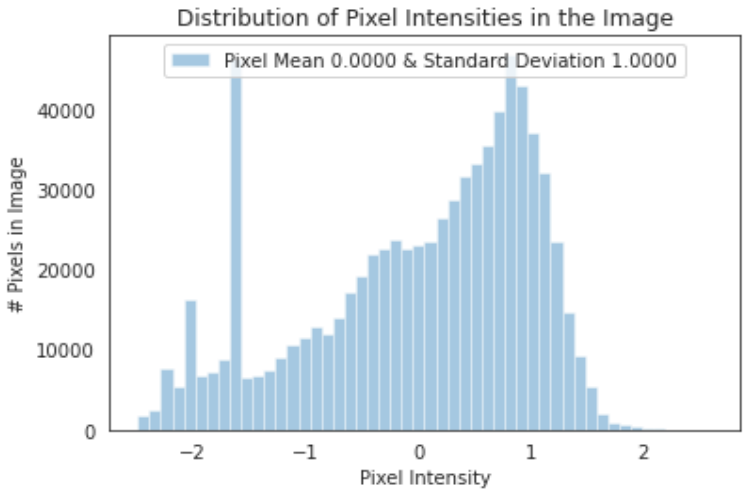

Figure 8. Pixel intensity graph.

3.3. Building a CNN Model

Table 4. Parameter types and numbers.

Parameter Types	Number of Parameters
Total Params	6,203,681
Trainable Params	6,202,785
Non-trainable Params	896

According to **Table 4**, the model has a total of 6,203,681 parameters. Of these, 6,202,785 parameters are trainable, meaning they can be changed and improved during the model training process to improve its performance in each task. On the other hand, there are 896 untrained parameters that are predicted and remain constant throughout the training period. These untrained layers usually include feature sets or pre-trained features obtained from previous examples or external sources. By separating trained and untrained phases, the model benefits from a balance between flexibility and stability. This separation allows the model to learn specific information while using pre-existing knowledge or fixed features of the structure. Controlling and understanding these parameters is important for effective model training, optimization, and obtaining optimal results in various machine learning applications. During version schooling, several parameters are frequently defined to govern and optimize the mastering procedure. The parameter "epochs" shows that the education process entails repeating the complete dataset 10 instances, allowing the model to regularly analyze from the facts. The "validation records" parameter shows that a separate validation dataset is used to check the model's performance and examine its typical capacity at some point of schooling. The "class weight" parameter suggests the mission of various weights to extraordinary instructions, allowing for example to prioritize certain instructions or to cope with an unbalanced distribution of lessons. The "steps per epoch" parameter determines the number of steps or batches to be processed in each education length, which impacts the granularity and performance of the training method. Finally, the "validation steps" parameter specifies the range of steps or agencies to be processed at some point of the validation step, controlling the frequency of the

evaluation and the computational resources used. By cautiously tuning these parameters, the version's education approach may be satisfactory tuned to obtain better overall performance, enhance connectivity, and deal with the challenges or demands of a given task.

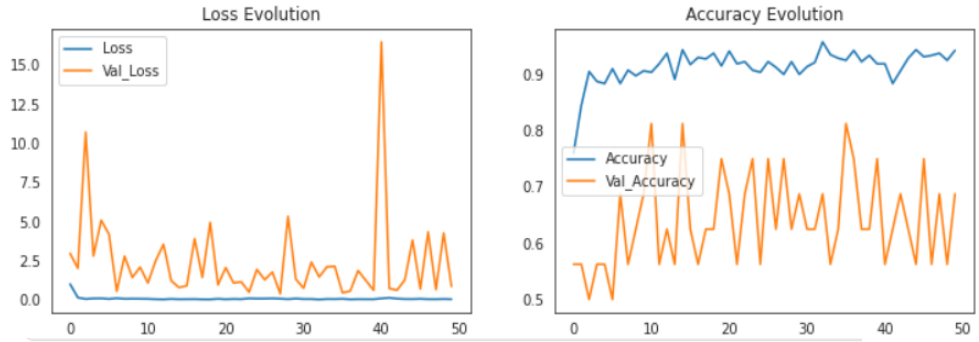

Figure 9. Loss and accuracy values of CNN Model for test and train input.

The given statistics represent the performance result of the train model after training **(Figure 9)**. The first line means that the loss executed on the check dataset is 0.4553, which shows how well the model's predictions healthy the actual values, with a decrease value indicating higher overall performance. The accuracy of the corresponding look at is 83.49%, which represents the percentage of correctly anticipated labels as compared to the total range of samples inside the test dataset. The second line refers to the decrease lack of 0.1417 at the education dataset, suggesting a development in performance during training. The accuracy of the train is said to be 95.32%, which shows the share of efficaciously anticipated labels inside the training set. These results advocate that the model has carried out excessive accuracy at the train information, however there is a mild lower in performance at the check dataset, which may additionally suggest the want for similarly improvement to improve the capability to generalize the model and enhance its overall performance on unobserved information.

		Actual class	
		P	**N**
Predicted class	**P**	**91**	143
	N	21	**369**

Figure 10. Confusion matrix of CNN model.

The Confusion Matrix is a commonly used tool in machine learning to evaluate the performance and accuracy of feature classification (**Figure 10**). It provides a comprehensive overview of model predictions by comparing them with actual labels. The main purpose of using the confusion matrix is to get information about the performance of the model in different groups and to identify the sources of errors. By

analyzing these figures, we calculated various parameters such as accuracy, precision, recall, and F1 scores, which provide a clearer understanding of the model's performance.

Table 5. Evaluations obtained from the confusion matrix of CNN model.

	0	1	Accuracy	Macro Avg.	Weighted Avg.
Precision	0.95	0.81	0.84	0.88	0.86
Recall	0.62	0.98	0.84	0.8	0.84
F1-score	0.75	0.89	0.84	0.82	0.83
Support	234	390	0.84	624	624

The results of the perturbation analysis of the CNN model show that the model performed well in general (**Table 5**). The accuracy of predicting the "Normal" category (0) was excellent, at 0.95, meaning that when the model described an event as "Normal", it was 95% accurate. On the other hand, the recall for the "Normal" group was 0.62, suggesting that the model missed many "Normal" events as only 62% were correctly identified. The F1-score for the "Normal" group was 0.75, indicating a well-balanced performance in terms of precision and recall. The number of incidents in the "Normal" group was recorded as 234, which shows the support of that group. Further, the accuracy of the category "pneumonia" (1) was 0.81, suggesting that when the model predicted the event as "pneumonia", it was correct 81% of the time. The recall for the group "pneumonia" was strong, at 0.98, which means that the model correctly identified 98% of cases of "pneumonia". The "pneumonia" cluster has an F1-score of 0.89, indicating a strong combination of precision and recall. The number of cases in the "pneumonia" group was reported as 390, indicating support for that group. Accuracy measurements provide a comprehensive assessment of the model's performance. Both precision and recall were 0.84, suggesting that the model was 84% accurate. The accuracy of the F1-score was also 0.84, indicating that the entire dataset performs well in terms of accuracy and recall. Moderate support for all groups is indicated by positive support of 0.84. When statistical significance was considered, the accuracy was 0.88, suggesting acceptable overall accuracy across groups. The model has some problems in treating cases from both groups equally, as seen by the average score of 0.80. The average F1-score was 0.82, indicating a good level of accuracy and recall for all groups. The average support measured across all groups was reported as 624 with a total of major support. Finally, the weighted values provide a general assessment that accounts for group imbalance. The mean accuracy was 0.86, suggesting that the overall accuracy was satisfactory. The weighted average recall was 0.84, indicating that the model can capture samples from both groups. The weighted average F1-score was 0.83, indicating a balanced performance for all groups when precision and recall were considered. All the heavy support in all groups is indicated by the heavy support of 624. In general, the model worked well, with excellent accuracy and recall for the "pneumonia" group but significant problems collecting cases from the "Normal" group. These findings indicate that the model may require further development to accurately identify "Normal" cases while maintaining high accuracy for "pneumonia" cases.

3.3.1. DenseNet

Table 6. Parameter count of DenseNet.

Total params:	7,037,504
Trainable params:	6,953,856
Non-trainable params:	83,648

In a traditional CNN, each layer is only connected to the next layer. However, in Densenet, every layer is connected to every deep layer in the network. This dense connectivity model facilitates reuse and promotes the propagation of gradients, which can improve information flow and improve network performance. Dense connectivity in Densenet is achieved through a specific structure called a "dense block". A dense domain has many layers, where each layer takes all the previous maps as input. This design ensures that information from earlier stages is directly accessible to subsequent stages, allowing for the extraction of more efficient features. Additionally, Densenet includes a transition layer between dense blocks to control the number of feature maps. The transformation layer includes batch normalization, followed by a 1 × 1 convolutional layer and averaging. This reduces the size of feature maps, which helps reduce network complexity. In the results, the total number of parameters of the Densenet model is 7,037,504. Of these, 6,953,856 parameters are trainable, meaning they are learned during training (**Table 6**). The remaining 83,648 parameters are not configurable, which usually include batch normalization parameters or other fixed parameters. Overall, Densenet has shown remarkable performance in various computer vision tasks, such as image segmentation, object recognition, and segmentation. Its dense network has been shown to improve gradient flow, reduce the vanishing gradient problem, and promote segmentation. These characteristics make Densenet an efficient framework for deep learning tasks, leading to superior results on benchmark datasets.

```
624/624 [==============================] - 13s 21ms/step - loss: 1.3580 -
Test Accuracy: 70.99%
652/652 [==============================] - 91s 139ms/step - loss: 0.2226
Train Accuracy: 92.48%
```

Figure 11. Loss and accuracy values of DenseNet for test and train input.

In the results you provided, the Densenet model achieved an analysis loss of 1.3588 and a train loss of 0.2226 (**Figure 11**). The loss value shows how well the model works in reducing the difference between the predicted and actual values. Low loss rates usually indicate good model performance. A test accuracy of 70.99% means that the model predicted the class labels for 70.99% of the test data samples. Similarly, the training accuracy of 92.48% indicates that the model has achieved an accuracy of 92.48% on the training data. Accuracy is a measure of how well the model performs overall, with higher values indicating better performance. A confusion matrix is a useful tool for analyzing the performance of a cluster model. It shows the number of correct and incorrect guesses for each category. In the confusion matrix:

		Actual class	
		P	**N**
Predicted class	**P**	**87**	147
	N	3	**387**

Figure 12. Confusion matrix of DenseNet.

The rows represent the actual groups, while the columns represent the predicted groups. The diagonal elements of the matrix represent the correct estimates, where the predicted group matches the true group. In this case, the model correctly described 224 events of the first class and 260 of the second class. The off-diagonal elements represent misclassifications. According to the confusion matrix (**Figure 12**), it can be noted that the model did not distinguish 10 times the first group as the second group and 130 times the second group as the first group. To comment on the results, it is important to have some context about the specific task or dataset of the Densenet model that was trained and tested. However, based on the data provided, it seems that this model has achieved good accuracy in the training and test groups, although the test accuracy is slightly higher. The wrong choices shown in the confusion matrix indicate that the model makes some mistakes in distinguishing between the two groups. Further research, such as examining poorly structured samples or considering other evaluation metrics, will be necessary to understand the nature of these errors and improve the performance of the model.

Table 7. Evaluation metrics of DenseNet.

	0	**1**	**Accuracy**	**Macro Avg.**	**Weighted Avg.**
Precision	0.98	0.69	0.71	0.84	0.8
Recall	0.24	0.99	0.71	0.62	0.71
F1-score	0.39	0.81	0.71	0.6	0.66
Support	234	390	0.71	624	624

Based on the generated confusion and evaluation measures, the DenseNet model (at time 50) shows mixed performance (**Table 7)**. The accuracy of the "Normal" (0) category is as high as 0.98, suggesting that when the model predicts an image as "Normal", it is true 98% of the time. The recall for the "Normal" category, on the other hand, is very low at 0.24, which means that the model detects only 24% of "Normal" events. The accuracy of the category "pneumonia" (1) is 0.69, showing an accuracy of 69% in predicting cases of pneumonia. The recall for the "pneumonia" category is high at 0.99, which means that the model correctly identifies 99% of true pneumonia cases. The recall for the category "pneumonia" is strong, at 0.99, indicating that the model correctly detects 99% of true cases of pneumonia. The F1-score for the "Normal" group is 0.39, while it is 0.81 for the "pneumonia" group, indicating a high level of precision and recall. The overall accuracy of the model is 0.71, and the main and average metrics show good capability. According to the confusion matrix, the model correctly identified 57 cases as "Normal" in fact "Normal", but incorrectly predicted a large number of cases (177) as "Normal" is actually "pneumonia". Except

for one case (389), it was well taken care of "pneumonia". These findings suggest that the DenseNet model has a large false positive rate for "Normal" cases, indicating a potential limitation in its ability to correctly detect non-pneumonia images. More work may be needed to improve the performance of the model to correctly identify "Normal" patients while maintaining its good performance in pneumonia discrimination.

3.3.2. VGG16

Table 8. Parameter count of VGG16.

Total params:	14,714,688
Trainable params:	14,714,688
Non-trainable params:	0

```
624/624 [==============================] - 11s 18ms/step - loss: 0.3491 -
5
Test Accuracy: 84.78%
652/652 [==============================] - 92s 141ms/step - loss: 0.2014 -
13
Train Accuracy: 93.08%
```

Figure 13. Loss and accuracy values of VGG16 for test and train input.

VGG16 is a convolutional neural network (CNN) architecture that was introduced by means of the Visual Geometry Group (VGG) at the University of Oxford in 2014 (**Table 8**). It is known for its simplicity and classical design, which consists of many convolutional layers with 3 × 3 small filters, followed by max-pooling layering. The architecture also includes a dense community with hidden layers, every composed of 4096 nodes, and an output layer with 1000 nodes (Kaggle). The VGG16 architecture won reputation due to its deep layer structure, which has proven progressed overall performance in a whole lot of computer vision obligations, which include image processing, item detection, and segmentation. The use of smaller filters (3 × 3) allows a deeper community with fewer layers compared to the use of larger filters. The VGG16 model achieved a test loss of 0.3491 and a train loss of 0.2014 (**Figure 13**). Low loss rates indicate good performance, as model predictions are in good agreement with the ground truth value. It is important to note that the loss of training is lower than the loss of the test, it suggests a general level where the model works well with the training data compared to the unknown test data. The test accuracy of 84.78% indicates that the model predicted the class labels for 84.78% of the test data samples. Similarly, the training accuracy of 93.08% highlights the model's performance of 93.08% accuracy on the training data. These positive data indicate that the model's performance is good, but there is still room for improvement. To enhance the performance of the model, methods such as weight control can be considered to reduce excess weight. In addition, data augmentation techniques can be used to increase the diversity of the training data, thereby improving the overall ability of the model to generalize and predict accurately.

Table 9. Evaluation metrics of VGG16.

	0	1	Accuracy	Macro Avg.	Weighted Avg.
Precision	0.88	0.85	0.86	0.86	0.86
Recall	0.72	0.94	0.86	0.83	0.86
F1-score	0.79	0.89	0.86	0.84	0.85
Support	234	390	0.86	624	624

Precision is defined as the proportion of correctly expected cases in relation to the total number of correctly expected cases (**Table 9**). The precision of the "Normal" (0) category is 0.88, suggesting that when the model predicts an image as "Normal", it is true 88% of the time. The precision of the "pneumonia" category (1) is 0.85, showing an accuracy of 85% in predicting cases of pneumonia. The number of correctly expected cases out of the total number of positive events is measured by recall, also known as sensitivity or true positive rate. The recall for the "Normal" class is 0.72, which means that the model correctly detects 72% of "Normal" events. The recall for the "pneumonia" category is 0.94, which means that the model correctly detects 94% of all pneumonia cases. The F1-score is a balanced evaluation of model performance as it is a proportional measure of precision and recall. The F1-score for the "Normal" group is 0.79, while the F1-score for the "pneumonia" group is 0.89. An increased F1 score means better balance and recall. The number of occurrences of each group in the dataset is represented by the support. The "Normal" group has 234 supporters, while the "pneumonia" group has 390. The accuracy of the model predictions across all groups was measured. This VGG model has an accuracy of 0.86, suggesting that it predicts correctly 86% of the time. A macro average aggregates average performance across groups without considering group imbalances, but a weighted average does. The overall precision, recall, and F1 scores are respectively 0.86, 0.83, and 0.84. The weighted average precision, recall, and F1 scores are respectively 0.86, 0.86, and 0.85. A confusion matrix is a table that compares model predictions with actual labels. In this situation, the model predicted 169 cases as "Normal" actually as "Normal", and 65 cases as "pneumonia" actually as "pneumonia". It misdiagnosed 24 cases as "pneumonia" when they were actually "Normal", and correctly identified 366 cases as "pneumonia". In general, the model works very well for both groups, with high accuracy and recall, suggesting its ability to distinguish between "Normal" and "pneumonia". The effectiveness of the model is also supported by the F1 and precision scores. It is important to emphasize, however, that a comprehensive study will require more information about the specific data, the problem being addressed, and any specific methods or limitations.

3.3.3. ResNet

Table 10. Parameter count of ResNet.

Total params:	23,587,712
Trainable params:	23,534,592
Non-trainable params:	53,120

```
624/624 [==============================] - 13s 20ms/step - loss: 1.0157 - a
0
Test Accuracy: 66.67%
652/652 [==============================] - 91s 140ms/step - loss: 0.4419 -
69
Train Accuracy: 81.19%
```

Figure 14. Loss and accuracy values of ResNet for test and train input.

ResNet, short for Residual Network (**Table 10**), is a deep convolutional neural network architecture proposed by He et al. [11]. Introduced the concept of residual connections, which helped solve the problem of vanishing gradients in deep neural networks. In traditional deep systems, information flows through a series of layers, and each layer learns to extract elements from the input. However, as the network gets deeper, the gradients can become very small, making it difficult for the network to learn effectively. ResNet solves this problem by introducing skip links, also known as shortcut links or data maps.

In the results we provided for the ResNet model, the test loss of 1.0157 and the train loss of 0.4419 show how well the model works to reduce the difference between the predicted and actual values (**Figure 14**). Lower loss valuesare generally desirable, suggesting better model performance. A test accuracy of 66.67% means that the model correctly predicted the class letters for 66.67% of test data samples. Similarly, the training accuracy of 81.19% indicates that the model has achieved an accuracy of 81.19% on the training data. Higher levels of accuracy indicate better overall performance.

Comparing the accuracy of the train with the accuracy of the test, it seems that this model has a certain degree of overfitting. Overfitting occurs when a model performs well on the training data but does not fit well on the unobserved test data. Regular methods such as dropping out of school or losing weight can be used to reduce excess and improve overall.

Table 11. Evaluation metrics of ResNet.

	0	**1**	**Accuracy**	**Macro Avg.**	**Weighted Avg.**
Precision	1	0.64	0.65	0.82	0.78
Recall	0.08	1	0.65	0.54	0.65
F1-score	0.14	0.78	0.65	0.46	0.54
Support	234	390	0.65	624	624

The ResNet example (time 50) (**Table 11**) shows the performance based on the given confusion matrix and evaluation metrics. Accuracy for the “Normal” category (0) is 1, indicating that when the model predicts an image as “Normal”, it is correct 100% of the time. However, the recall for the “Normal” group is very low at 0.08, suggesting that the sample only identifies 8% of “Normal” cases. For the group “pneumonia” (1), the accuracy is 0.64, showing an accuracy of 64% in predicting cases of pneumonia. The recall for the category “pneumonia” is higher than 1, which means that the model correctly identifies all cases of pneumonia. However, the very low recall for the “Normal” category raises concerns about the model’s ability to correctly

identify “Normal” cases. The F1-score for the “Normal” group is 0.14, which shows the imbalance between precision and recall. The overall accuracy of the model is 0.65, and the main and limited metrics suggest subpar performance compared to previous models. The confusion matrix reveals that the model predicted 18 events as “Normal” in fact as “Normal”, but misclassified many events (216) as “Normal” when they were. and “pneumonia”. It correctly classified all cases (390) as “pneumonia”. These results suggest that the ResNet model struggles to accurately identify “Normal” cases and may have a high false positive rate, which may be a cause for concern in clinical settings. Further research and fine-tuning may be needed to improve the performance of the model.

3.3.4. InceptionNet

```
624/624 [==============================] - 15s 23ms/step - loss: 0.3736 - accu
racy: 0.8558 - precision: 0.8676 - recall: 0.9077
Test Accuracy: 85.58%
652/652 [==============================] - 95s 145ms/step - loss: 0.2899 - acc
uracy: 0.9043 - precision: 0.9913 - recall: 0.8790
Train Accuracy: 90.43%
```

Figure 15. Loss and accuracy values of InceptionNet for test and train input.

InceptionNet, also known as GoogLeNet, is a deep neural network architecture developed by Szegedy et al. [10]. It was designed to tackle the problems of deep network training by introducing the concept of “starting modules” and reducing the computational cost of convolutions. A key innovation in InceptionNet is the initialization module, which consists of multiple convolutional layers with different filter sizes. The purpose of the startup module is to capture information at different spatial scales by applying filters of different sizes within the same scale. This allows the network to efficiently extract local and global components.

Regarding the results we provided for the InceptionNet model, the test loss of 0.3736 and the train loss of 0.2899 show that the model has succeeded in reducing the difference between the predicted and actual values (**Figure 15**). Lower loss values are generally desirable as they indicate better performance. A test accuracy of 85.58% means that the model correctly predicted the class letters for 85.58% of the test data samples. Similarly, a train accuracy of 90.43% indicates that the model has achieved an accuracy of 90.43% on the training dataset. These accuracy scores indicate that the performance of the model is good but still leaves room for improvement.

It is vital to be aware that the outcomes we supplied show a big overall performance gap between training and test accuracy, which indicates over-performance. Overfitting occurs when a version learns to carry out properly on training data but fails to generalize to unobserved look at records. Standard methods along with dropout, weighting, or growing the size of the training dataset can be used to further reduce and improve generalization.

Table 12. Evaluation metrics of InceptionNet.

	0	1	Accuracy	Macro Avg.	Weighted Avg.
Precision	0.83	0.85	0.84	0.84	0.84
Recall	0.73	0.9	0.84	0.82	0.84
F1-score	0.77	0.88	0.84	0.82	0.84
Support	234	390	0.84	624	624

Based on the specified confusion matrix and evaluation criteria, the InceptionNet model (at 50 times) performs very well (**Table 12**). The accuracy of the "Normal" (0) category is 0.83, suggesting that when the model predicts an image as "Normal"， it is correct 83% of the time. The recall for the "Normal" class is 0.73, which means that the model correctly detects 73% of all "Normal" events. The accuracy of the "pneumonia" category (1) is 0.85, which means an accuracy of 85% in predicting cases of pneumonia. The recall for the group "pneumonia" is strong, at 0.99, which means that the model correctly detects 99% of cases of real pneumonia. Both groups have good F1 scores, with 0.77 for "Normal" and 0.88 for "pneumonia". The overall accuracy of the model is 0.84, and large and heavy metrics support this figure. According to the confusion matrix, the model predicted 170 cases as "Normal" as "Normal"， and 64 cases as "Normal" as "pneumonia". It misdiagnosed 36 cases as "pneumonia" when they were actually "Normal", and correctly identified 354 cases as "pneumonia". These findings show that the InceptionNet model performs well in classifying "pneumonia" cases but outperforms in detecting "Normal" cases compared to VGG16.

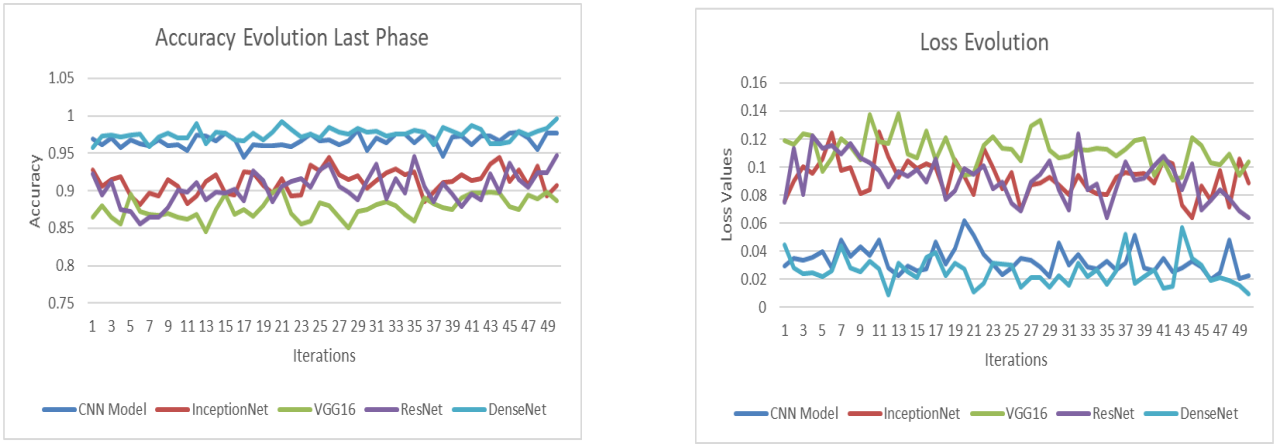

Figure 16. Final accuracy and loss evolution after 250 epochs.

DenseNet's high accuracy can be attributed to its unique dense connectivity method **(Figure 16)**. By connecting each layer to each layer in a feedforward manner, DenseNet promotes the use of layers, which can help improve gradient flow and reduce the vanishing gradient problem. This dense network allows DenseNet to accurately capture and distribute information across the network, leading to powerful performance. ResNet, with its new residual networks, has shown great accuracy in solving the vanishing edge problem. By learning residual maps instead of generating direct maps, ResNet allows deep neural networks to be trained efficiently. These

remaining connections enable the gradient to flow smoothly, simplifying the training of very deep networks and contributing to high accuracy. The VGG16, although relatively simple compared to other models, has achieved competitive accuracy. Its parallel architecture, with small convolutional filters (3 × 3) and max-pooling layering, allows capturing local features at multiple scales. However, the depth of VGG16 is relatively low compared to DenseNet and ResNet, which reduces its ability to learn complex representations and contributes to low accuracy. InceptionNet, particularly Inception V3, has slightly lower accuracy compared to the others. The Inception architecture, with its inception modules, captures information at different spatial scales using parallel convolutional filters. While this design enables computational efficiency, it introduces increased complexity, leading to challenges in training and potentially impacting accuracy.

4. Conclusion

Through multiple training phases with the same input, the models were able to achieve increased accuracy. This iterative process allowed the models to learn and refine their predictions, resulting in improved performance over time. Despite the limitation of limited computing power and server space, the models underwent a maximum of 250 training iterations (epochs). This constraint was necessary to balance the computational resources available while still achieving notable progress. Additionally, it is worth noting that reviews from the literature support the notion that these models are robust and generalizable. Therefore, even when tested with different X-ray image inputs, the models are expected to exhibit strong performance, indicating their reliability and effectiveness across various scenarios.

The difference in accuracy among the models is attributed to the fundamental design variations between them.

Author contributions: Conceptualization, AN and RSD; methodology, SBS; writing—original draft preparation, EU and SIS; writing—review and editing, IK, SIS and EU; supervision, RSD and SST. All authors have read and agreed to the published version of the manuscript.

Conflict of interest: The authors declare no conflict of interest.

References

1. Baque-Juston M, Pellegrin A, Leroy S, et al. Organizing pneumonia: What is it? A conceptual approach and pictorial review. In Diagnostic and Interventional Imaging. Elsevier Masson SAS. 95(9): 771–777. doi: 10.1016/j.diii.2014.01.004
2. Ayan E, Ünver HM. Diagnosis of Pneumonia from Chest X-Ray Images using Deep Learning. In: Proceedings of the 2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT); 24−26 April 2019; Istanbul, Turkey.
3. Jaiswal AK, Tiwari P, Kumar S, et al. Identifying pneumonia in chest X-rays: A deep learning approach. Measurement: Journal of the International Measurement Confederation. 2019; 145: 511–518. doi: 10.1016/j.measurement.2019.05.076
4. Szepesi P, Szilágyi L. Detection of pneumonia using convolutional neural networks and deep learning. Biocybernetics and Biomedical Engineering. 2022; 42(3): 1012–1022. doi: 10.1016/j.bbe.2022.08.001
5. Elshennawy NM, Ibrahim DM. Deep-Pneumonia Framework Using Deep Learning Models Based on Chest X-Ray Images. Diagnostics. 2020; 10(9). doi: 10.3390/diagnostics10090649

6. Mohammed Ahmed A, Alhadi Babikir G, Mohammed Osman S. Classification of Pneumonia Using Deep Convolutional Neural Network. American Journal of Computer Science and Technology. 2022; 5(2): 26. doi: 10.11648/j.ajcst.20220502.11
7. Varshni D, Nijhawan R, Thakral K, et al. Pneumonia Detection Using CNN based Feature Extraction. In: Proceedings of the 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT); 20−22 February 2019; Coimbatore, India.
8. Hassan M. ul. VGG16—Convolutional Network for Classification and Detection. Available online: https://neurohive.io/en/popular-networks/vgg16/ (accessed on 9 May 2024)
9. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. Computer Science. 2017.
10. Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions. Computer Science. 2014.
11. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. Computer Science. 2015.
12. Desai P, Pujari J, Sujatha C, et al. Hybrid Approach for Content-Based Image Retrieval using VGG16 Layered Architecture and SVM: An Application of Deep Learning. SN Computer Science. 2021; 2(3). doi: 10.1007/s42979-021-00529-4
13. Tao Y, Xu M, Lu Z, Zhong Y. Dense net-based depth-width double reinforced deep learning neural network for high-resolution remote sensing image per-pixel classification. Remote Sensing. 2018; 10(5). doi: 10.3390/rs10050779
14. Tiwari A. Supervised learning: From theory to applications. Artificial Intelligence and Machine Learning for EDGE Computing. 2022; 23–32. doi: 10.1016/B978-0-12-824054-0.00026-5