Optimized Deep CNN Model for Enhanced COVID-19 Detection

A Thesis Submitted to the Committee on Graduate Studies

in Partial Fulfillment of the Requirements for the Degree of Master of Science

in the Faculty of Arts and Science

Trent University

Peterborough, Ontario, Canada

Applied Modelling and Quantitative Methods M.Sc. Graduate Program

September 2024

ABSTRACT

Optimized Deep CNN Model for Enhanced COVID-19 Detection

Sayed Mansour Hashemipoor

This research presented an AI-driven methodology for precise COVID-19 detection in medical diagnostic imaging using chest X-ray images. The primary focus was on developing an optimized model using convolutional neural networks (CNNs) and leveraging transfer learning as a low-level feature extractor method. Significant attention was also so given to data transformation and enhancement techniques to improve the information content and distinguishability of non-linearities. The proposed methodology enabled the flexibility to apply multiple models within the framework and identify the most suitable model for the specific task at hand. By emphasizing state-of-the-art CNN models and employing a strategic exploration-exploitation trade-off, this study identified a robust model with heightened accuracy. The results demonstrated a model accuracy of 90.11%, a sensitivity of 91.16%, and a precision of 89.19%, highlighting the model's effectiveness in accurately identifying both true positive and true negative test cases.

Keywords: Machine Learning, Deep Learning, Convolution Neural Networks, Transfer Learning, Automatic Bayesian Optimization, Hyperparameter tuning, COVID-19, X-ray Images, Medical Diagnostic Imaging.

Acknowledgements

I would like to dedicate my master's thesis work to the victims of Flight PS752, as a tribute to all 176 innocent passengers who tragically lost their lives in the downing of the Ukraine International Airlines Flight 752 on January 8^{th}, 2020.

Among these victims were brilliant minds and dedicated students who contributed substantially to Canadian educational institutions. I extend my deepest gratitude to the team at Global Affairs Canada for their unwavering dedication to honoring the victims of the flight. This horrific event occurred over the city of Tehran, when two missiles struck the plane just three minutes after takeoff.

I would also love to express my heartfelt gratitude to my sister. Her unwavering support and sacrifices have been a constant source of strength throughout my academic endeavors in Canada and absolutely my entire life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the field of AI has experienced rapid growth, incorporating a diverse array of techniques and approaches that empower machines to execute tasks traditionally associated with human intelligence. The escalating accessibility of data and advancements in computing power have elevated AI to an indispensable tool for businesses and organizations. Its role extends beyond mere automation, encompassing the unlocking of insights and the augmentation of decision-making capabilities [34].

Artificial intelligence (AI) and classical programming epitomize two distinct paradigms in problem-solving with machines. To begin, classical programming entails explicitly formulating a set of rules and instructions that guide the machine's actions. These rules derive from the programmer's comprehension of the problem and the desired results. In contrast to AI, which leverages learning algorithms and data to enable machines to dynamically adapt and evolve, classical programming relies on predetermined instructions, reflecting the programmer's explicit knowledge and problem-solving strategies [36].

While classical programming proves effective in solving well-defined problems, its efficacy diminishes when faced with complex tasks characterized by a high degree of

uncertainty or variability. In contrast, AI involves training machines to learn from data and adapt their behavior based on experience, rendering it a more flexible and versatile approach to problem-solving [63]. AI systems possess the capability to analyze extensive datasets, discern patterns and correlations, and utilize this information to make predictions or take actions without the need for explicit programming [55]. This makes AI particularly well-suited for tasks that involve complex decision-making, such as natural language processing, image recognition, and autonomous driving.

Another pivotal distinction between AI and classical programming lies in the level of human intervention required. In classical programming, programmers must meticulously design and write the program to achieve the desired outcome, a process that often demands substantial time and effort. In contrast, AI systems can learn and enhance their performance autonomously, diminishing the need for extensive human intervention and enabling machines to undertake tasks previously deemed beyond their capabilities [62].

Artificial Intelligence (AI) encompasses machine learning (ML) as one of its subsets. ML, in turn, is a specialized domain within AI, focusing on training machines using extensive datasets to recognize patterns and formulate predictions [77]. Within the realm of ML, deep learning (DL) emerges as a potent subclass. Deep learning represents a sophisticated branch of machine learning that aims to mimic the learning process of the human brain. At its core lies the development of artificial neural networks, which seek to replicate the electrochemical interactions observed in biological neurons through the application of mathematical functions [66].

Deep learning involves training neural networks with multiple layers, demonstrating significant accomplishments in diverse applications, notably excelling in areas like

image and speech recognition [85, 118]. This hierarchical relationship, Figure 1.1, illustrates the broader scope of AI, the specialized domain of ML within it, and the advanced capabilities offered by DL within the ML framework.

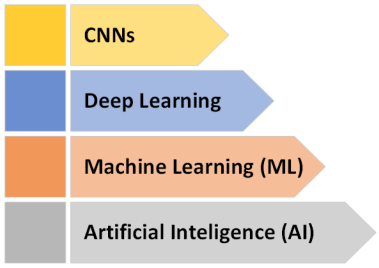

Figure 1.1: Hierarchy of the relationships

The objective of machine learning is to discover patterns and relationships in data and to put those discoveries to use. This discovery process is achieved through modeling techniques developed over the past 30 years in statistics, computer science, and applied mathematics. These various approaches can range from simple to tremendously complex, but all share a common goal to estimate the functional relationship between the input features and the target variable.

Convolutional neural networks (CNNs) stand out as a prominent, and widely adopted category of machine learning models, finding applications across diverse domains such as image recognition [41], speech recognition [51], and natural language processing (NLP) [38]. Notably, CNNs have demonstrated exceptional efficacy in tasks related to image recognition, consistently achieving state-of-the-art results across various benchmarks. A crucial category of neural networks within the realm of deep learning is the Convolutional Neural Network, which is specifically designed for image recognition and classification. CNNs employ multiple layers to extract information from images and determine their respective classes. Take the ability of a

CNN to discern whether an X-ray image features a healthy or COVID-19 case, based on its training with a set of training images. The key to their success lies in their hierarchical architecture, featuring layers that conduct convolution operations. This design enables CNNs to adeptly capture spatial features and patterns within images, operating at different levels of abstraction [23, 110].

Presently, algorithms, heuristics, and artificial intelligence models are being applied in various fields of knowledge. In the healthcare sector, there has been significant advancement in using computer technology to save lives and enhance the quality of life. Particularly, visual information is frequently employed for disease detection and evaluation. This has led to the integration of computer vision and medical diagnostic imaging. Effective healthcare system improvements require the integration of these technologies, along with data analysis from multiple sources and real-time processing capabilities.

1.1 Problem Overview

Combating COVID-19 proves to be a formidable challenge, given its swift transmission rate among citizens globally. Furthermore, the virus has the potential to mutate, creating additional strains. Consequently, early detection emerges as a crucial strategy in the fight against COVID-19. Various approaches to early detection include blood tests, viral tests, and the analysis of different imaging modalities, such as X-ray and Computed Tomography (CT) scans [12].

On one hand, efficient detection of COVID-19 through medical diagnostic imaging plays a crucial role in controlling the problems caused by the disease and reducing the mortality rate. Currently, chest X-rays (CXR) and CT scans are commonly used

for assessing and monitoring the severity of the disease in patients. On the other hand, additional research is necessary to ascertain the duration of protection and effectiveness of vaccines, particularly concerning newly identified variants. Moreover, initiatives are in progress to create efficient screening protocols for identifying and isolating cases, as well as to improve treatment strategies and testing capacities through versatile testing methods.

Diagnosing COVID-19 through radiographic imaging necessitates patient data as the primary input, which is typically accessible only through diagnostic centers. Consequently, patients must physically visit these centers to confirm the presence of COVID-19 in their bodies. Many households in underdeveloped nations lack access to private transportation, and individuals in rural areas often have to travel long distances to reach a diagnostic center. As a result, individuals must rely on public transportation for COVID-19 testing, which increases the risk of virus transmission and other related issues. In the initial stages of the pandemic, the availability of training data posed a significant constraint, despite efforts to expand the dataset. Addressing this limitation necessitated the aggregation of data from diverse sources to construct a comprehensive dataset for this investigation. To overcome the challenges in medical diagnostics, Computer-Aided Diagnostics (CAD) is increasingly employed to augment precision and speed in medical diagnoses. Artificial Intelligence (AI) algorithms, particularly those embedded in CAD systems, are harnessed for their ability to efficiently process extensive datasets [59].

1.2 Motivation

The medical field has witnessed extensive integration of AI methods, notably in disease diagnosis using chest scans, including conditions such as Viral Pneumonia, COVID-19, Lung Opacity, etc. Deep Learning (DL) algorithms as a subset of AI, have emerged as a particularly potent tool for image classification, proving highly effective in detecting diseases like COVID-19, characterized by multi-layer neural networks, exhibit proficiency in identifying intricate patterns within images without requiring extensive preprocessing. Over time, advancements in convolutional neural networks have substantially minimized error rates in image categorization competitions.

Within this domain, the analysis of medical images often involves the application of deep learning techniques, such as Transfer Learning (TL) in employing pre-trained convolutional neural network models either as a feature extractor or fine-tuned approaches.

Most prior state-of-the-art studies have typically focused on training and evaluating a specific model in an effort to improve its accuracy. The objective of this study is not only to achieve high classification accuracy through model evaluation, but also to accomplish it through the training of automated end-to-end optimized deep learning CNN models and choose the best one based on the objective function and the ability to generalize to any image classification problems through medical diagnostic imaging. This can be achieved through the combination of either pre-trained models or any single state-of-the-art models. The framework also aims to be adaptable to new variations and mutations of the disease as well as the ability to detect any lung-related diseases in the future.

This research was motivated by the publication by Loey et al. [72], which introduced a new approach to chest X-ray image classification using a Bayesian optimization-based convolutional neural network (CNN) model. Their proposed CNN model is designed for the extraction and learning of deep features. Furthermore, the Bayesian-based optimizer is utilized to adjust the CNN hyperparameters based on the objective function.

1.3 Thesis Contributions

This study applied the utilization of different pre-trained models specifically designed for the nuanced task of COVID-19 detection in medical images. Leveraging the low-level features extracted from extensive datasets proves instrumental in enhancing model performance. The central aim of the research is to meticulously identify the most optimal hyperparameters within a judicious range of values, as these parameters wield a direct influence on the efficacy of each model.

To achieve this objective, this study employs the Bayesian optimization technique, delving deeply into the nuanced impact of critical factors. These factors include batch size, learning rate, momentum, gradient optimizers, and L2-regularization for each model. By conducting a comprehensive exploration of these parameters, the study seeks to pinpoint their optimal values, thereby bolstering the overall effectiveness of COVID-19 detection models in medical diagnostic imaging applications.

The contributions of the thesis research are as follows:

- Model Selection: The work focuses on the selection of the most suitable model amongst others within the framework, ranging from the combination of pre-trained models with new state-of-the-art models to the incorporation of any

newly developed models.

- Optimized CNN Models: The research broadens the domain of search space to automatically optimize CNN models' trainable hyperparameters by embracing a strategic exploration-exploitation trade-off, allowing for a more comprehensive exploration of the model configurations.

- Data Transformation and Enrichment: The thesis delves into image enhancement techniques for data transformation and enrichment, with the aim of improving information content and reducing data volume through domain transformations, sampling, and image visual enhancement.

- COVID-19 Application for Image Classification: The thesis applies the developed framework to the specific domain of COVID-19 image classification, contributing to the detection of any related lung-diseases rather than COVID-19 or its mutations, and thereby bolstering the advancement of medical diagnostic imaging.

These contributions collectively aim to enhance the effectiveness and adaptability of deep learning models for image classification, particularly in the context of medical diagnostic imaging and the specific challenges posed by COVID-19 disease.

1.4 Thesis Outline

The remainder of my thesis is organized as follows:

In Chapter 2, the basics of ANN, ML, and DL, including the fundamentals of CNN, are explored, along with a review of related works to provide context. Chapter 3 meticulously details the process of model development. Chapter 4 delves into the

proposed methodology and outlines the model components, covering data preparation, model training, and model evaluation. Chapter 5 elucidates the experimental setup and results, discussing the intricacies of model design, the proposed dataset, implementation considerations, and the examination of the experiment results. Finally, the last chapter encompasses conclusions and future work, providing insights into the findings and outlining potential avenues for future research endeavors.

The Appendices include mathematical formulas, structures of the pre-trained CNN models, and sample X-ray images with their enhanced output results.

Chapter 2

Background

The incorporation of Artificial Intelligence (AI) into medical imaging has the potential to bring about significant changes. Proficiency in the principles and use of radiomics, artificial neural networks, machine learning, and deep learning forms a crucial groundwork for developing design solutions that adhere to ethical and regulatory standards [86]. This proficiency is also essential for constructing AI-based algorithms aimed at improving outcomes, quality, and efficiency. Additionally, a comprehensive understanding of applications, opportunities, and challenges from a programmatic standpoint is vital for fostering an ethical and sustainable implementation of AI solutions in the field of medical imaging.

Within the realm of medical imaging, artificial neural network (ANN) stands as the cornerstone of both machine learning and deep learning (DL). Comprising layers of interconnected nodes, an ANN functions as an analytical algorithm, with inputs ranging from radiomic features extracted from image files to the images themselves when utilizing a convolutional neural network. The integration of AI into medical imaging marks an exhilarating era, offering a reengineered and reimagined landscape for clinical and research capabilities.

A significant catalyst for the rise of AI in medical imaging has been its ability to enhance visual recognition in radiology, yielding lower error rates than human observers since 2015 [65, 74]. The versatile capabilities of ML in medical imaging encompass, among other functions, lesion detection and classification, automated image segmentation, data analysis, extraction of radiomic features, prioritization of reporting and study triage, and image reconstruction [68, 74]. Furthermore, an ANN can complement traditional statistical analysis, providing deeper insights into research data [26].

2.1 Basics of Machine Learning Models

An artificial neural network (ANN) is composed of nodes, which can range from hundreds to millions, arranged in multiple layers, a configuration known as depth [108]. Deep learning employs an ANN with numerous layers, often exceeding six, and is recognized as a more advanced implementation of machine learning. DL is capable of conducting more intricate analyses, integrating larger datasets, and representing higher levels of abstraction. In this network, each node receives information from other nodes, and the outputs are weighted. The primary objective of the ANN is to maximize correct answers in comparison to a reference (ground truth) by adjusting the weights on each node, based on the error calculated during each forward propagation [74, 108]. Across each iteration or epoch, the mathematical solution gradually converges toward a more accurate outcome. This iterative process bears resemblance to the approach described for iterative reconstruction [27].

Optimal outcomes during the training phase are achieved with a substantial dataset. With each iteration, improvements in results become progressively smaller

[108]. A second, typically smaller, dataset is commonly employed to validate inferences, representing a significant portion of published work at present. In the domain of medical imaging, big data plays a crucial role in supplying large and reliable training datasets for ML and DL algorithms to learn from. DL specifically refers to the depth or number of layers in the ANN, often associated with convolutional neural networks, which are adept at identifying and extracting features directly from images [119].

Figure 2.1 indicates the lifecycle of the Machine Learning models comprising several distinct steps [63].

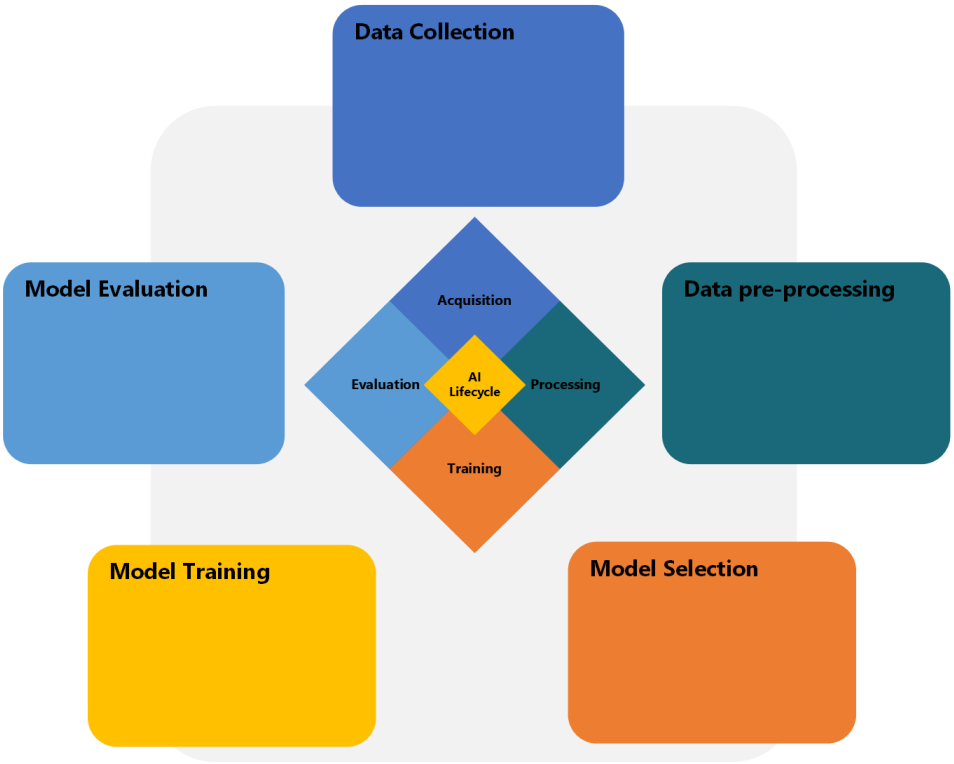

Figure 2.1: The Implementation Phase of Machine Learning Models

To begin with, Data Collection is the initial step involves gathering data from

diverse sources for utilization in the learning process. The quality of the data significantly influences the outcomes produced by the algorithms. Given that avoiding data quality issues is often impractical in most applications, data mining concentrates on two primary aspects: (1) identifying and rectifying data quality problems, and (2) employing algorithms that can accommodate suboptimal data quality.

Data preprocessing is a comprehensive area encompassing various interrelated strategies and techniques. In this phase, raw data collected in the previous step undergoes cleaning, organization, and transformation to achieve a format suitable for analysis. The tasks involved in data preprocessing, include removing missing values, handling outliers, and scaling the data for normalization and consistency.

The next step in the machine-learning workflow is to use that data to begin exploring and uncovering the relationships that exist between the input features and the target. In machine learning, this process is done by building statistical models based on the data. A suitable model is chosen based on the nature of the problem. The selected model is then trained using the training data from the previous phase, where the objective is to extract features and patterns from the data to enable accurate predictions. The training process entails adjusting the model parameters iteratively to minimize the error between predicted and actual values. This iterative adjustment allows the model to learn from the training data and improve its performance over time. The ultimate goal is to achieve a trained model that can effectively generalize its learning to make accurate predictions on new, unseen data.

Following the tuning of a machine-learning model, the subsequent crucial step is to assess its accuracy. Accuracy assessment is vital in understanding how well the

model will predict outcomes on new, unseen data. A thorough evaluation of predictive performance is essential before deploying the model in a production environment to analyze fresh data. If the assessment reveals satisfactory predictive performance, it instills confidence in deploying the model for practical use. Conversely, if the predictive performance falls short of the desired level for the intended task, it necessitates a reevaluation of both the data and the model. This reassessment aims to identify areas for improvement and optimization to enhance the model's accuracy and efficacy. Properly assessing the predictive performance of an ML model is a nontrivial task and various metrics can be used to evaluate the model's performance. Upon successful evaluation and validation, the model can be deployed for its designated task.

2.2 Convolutional Neural Networks

A crucial category of neural networks within the realm of deep learning is the Convolutional Neural Network. Specifically designed for image recognition and classification, CNNs employ multiple layers to extract information from images and determine their respective classes. Take, for instance, the ability of a CNN to discern whether an image features a cat, based on its training with a set of cat images. This section will delve into the architecture and functioning of CNNs.

From a program's perspective, any image is essentially a set of RGB numbers presented in a vector format. If a neural network can grasp the underlying patterns, it can be structured as a CNN to effectively detect images. Regular neural networks function as universal mathematical approximators, processing input through a series of functions to derive output. However, these regular neural networks encounter scalability issues when applied to image analysis. For instance, a $32 * 32$ pixel RGB

image would require $32 * 32 * 3 = 3072$ weights in the hidden layer. While regular neural nets may perform adequately in such cases, scaling up to a $224 * 224$ pixel RGB image results in a considerable increase to $224 * 224 * 3 = 150,528$ weights, leading to performance issues.

CNNs offer a solution to this scalability problem. In CNNs, the layers are organized in three dimensions—height, width, and depth. The CNN functions as a sequence of neural net layers, where each layer transforms one volume of activations to another through a differentiable function. Three fundamental types of layers construct the CNN including the Convolutional layer, the Pooling layer, and the Fully connected layer.

CNNs exhibit remarkable efficacy in tasks revolving around visual data analysis, encompassing image classification, object detection, and segmentation. Their proficiency stems from the capacity to assimilate hierarchical representations of visual information. By constructing higher-level features atop lower-level features, CNNs adeptly decipher intricate patterns in data, facilitating precise predictions.

Moreover, CNNs excel in tasks entailing sequential data analysis, notably in domains like speech recognition and Natural Language Processing (NLP). The inherent spatial or temporal structure in sequential data proves advantageous for Convolutional Layers. In speech recognition, the input waveform unfolds as a sequence of frames, allowing Convolutional Layers to extract features capturing phonetic nuances. In NLP, these layers extract features from textual inputs, encapsulating word meanings and their contextual relationships, exemplifying the versatility of CNNs across diverse applications.

The effectiveness of CNNs hinges on the availability of substantial labeled training

data. These models, laden with millions of parameters, necessitate extensive learning from data, demanding a diverse dataset to mitigate the risk of overfitting. Generally, a surplus of training data correlates with improved model performance. Nevertheless, amassing sizable labeled datasets can be resource-intensive, particularly for tasks involving intricate classification or segmentation. In such scenarios, transfer learning emerges as a valuable strategy. By commencing with a pre-trained model and refining it on a smaller dataset, the need for an extensive labeled dataset diminishes, contributing to enhanced model performance.

Furthermore, it's crucial to recognize that the quality of training data significantly influences CNN performance. Noisy or biased data may lead the model to learn artifacts rather than genuine patterns. Hence, meticulous curation and preprocessing of training data are imperative to ensure its representativeness for the targeted task.

2.2.1 Traditional CNN Architecture

Nowadays, Convolutional Neural Network architectures have demonstrated proficiency in achieving expert-level performance comparable to human capabilities across intricate visual tasks, notably in domains like medical image analysis and pathology detection. The evolution of CNNs dates back to the success of the first CNN in 1998, known as LeNet, primarily utilized for handwritten digit recognition and devised by Yann LeCun. LeNet, consisting of three convolutional, two average pooling, and two fully connected layers, marked the inception of CNN applications. While LeNet is considered shallow by modern standards, numerous CNN architectures have since been introduced in the literature. CNNs excel in feature extraction, a pivotal aspect illustrated in Figure 3.1, offering a schematic representation of a typical CNN tailored

for COVID-19 prediction, which contains Convolution layer, Pooling layer, and Fully connected layer [105].

2.2.2 Pre-trained CNN Models

A pre-trained model has undergone training on a large benchmark dataset, addressing problems similar to the one at hand. Due to the computational expense of training such models, a common practice is to import and utilize models from existing literature, leveraging the concept of transfer learning (TL). In TL, knowledge acquired by a deep learning model trained on a large dataset is applied to a task with a smaller dataset, reducing the need for extensive data and prolonged training periods required for DL algorithms taught from scratch [105].

2.2.3 Fundamentals of CNN

Convolutional Neural Networks are a potent form of neural network extensively utilized for image recognition tasks. Comprising convolutional and pooling layers, they extract pertinent features from input images. Subsequent fully connected layers employ these features for predictions. Prior to deployment, CNNs necessitate training on extensive datasets, enabling them to learn associations between extracted features and corresponding labels. During training, backpropagation and optimization refine this association process [33, 102].Once trained, CNNs predict labels for new images by passing them through the network, choosing the label with the highest predicted probability. Figure 2.2 illustrates a simplified CNN for image classification [10]. For instance, given an input image of a car, the network progressively extracts features

through hidden layers, culminating in a final layer producing a probability distribution over different classes (e.g., cat, bird, dog). The weight adjustments in each layer are learned via backpropagation to minimize prediction errors.

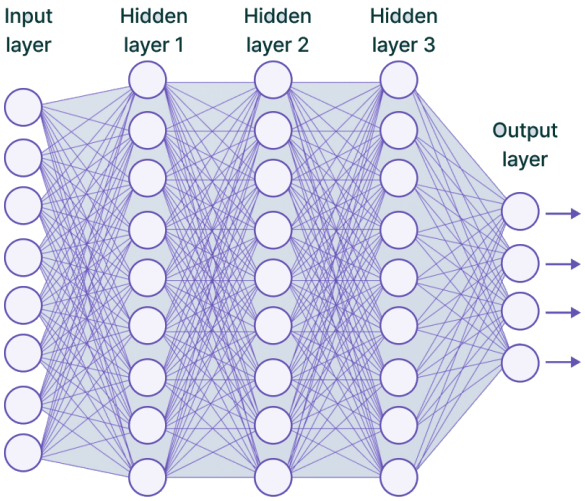

Figure 2.2: A simplified illustration of CNN network

Advantages

CNNs boast several advantages, rendering them a formidable tool for the analysis of both visual and sequential data. One primary strength lies in their capacity to acquire hierarchical representations of input data. By constructing higher-level features upon lower-level ones, CNNs exhibit an adeptness for comprehending intricate patterns in data, facilitating accurate predictions.

The capability to handle inputs of diverse sizes and aspect ratios stands out as another advantage of CNNs. Convolutional layers are proficient in learning translation-invariant features, enabling the recognition of patterns regardless of their location in the input. The incorporation of max-pooling layers further allows for downsampling, accommodating inputs with varying sizes and aspect ratios.

Generalization to new data represents a significant strength of CNNs. These models can provide accurate predictions for previously unseen data by learning task-relevant features rather than memorizing specific instances from the training set. Moreover, the application of transfer learning enhances their adaptability, allowing knowledge from pre-trained models to be transferred to new tasks, thereby improving accuracy and reducing the demand for extensive training data [63].

Limitations

While CNNs offer numerous advantages, it's essential to acknowledge certain limitations when utilizing them. One notable constraint is their demand for a substantial amount of labeled training data to achieve optimal performance. The vast number of parameters in these models requires extensive data variation to prevent overfitting, making the collection and labeling of ample training data a time-consuming and costly endeavor, particularly for tasks involving fine-grained classification or segmentation.

Another limitation stems from CNNs' sensitivity to the quality of training data. Noisy or biased data may cause the model to learn artifacts rather than the genuine underlying patterns. Consequently, meticulous curation and preprocessing of training data are crucial to ensuring its representativeness for the intended task.

Furthermore, CNNs can be computationally intensive during both training and evaluation, especially for large models with numerous layers. The memory and processing power requirements of CNNs can be substantial, but advancements in hardware and software, along with techniques like model pruning, quantization, and compression, have mitigated these challenges.

It's important to recognize that CNNs might not be the optimal choice for every

task. Tasks involving data with different structures, such as graphs or trees, may benefit more from other neural network types like graph neural networks or recursive neural networks. Similarly, tasks requiring reasoning over symbolic or logical representations might find better-suited models in rule-based or logic-based systems. Therefore, careful consideration of the task's specific requirements is essential when selecting an appropriate model architecture [63].

Layers of CNNs

A Convolutional Neural Network typically comprises, more specifically, several specialized layers, each contributing uniquely to the network's functionality:

- **Convolutional Layer**: The convolutional layer (ConvLayer) serves as the fundamental unit in a CNN [83], constituting its core building block. In operation, this layer conducts a convolution operation on the input image utilizing a set of learnable filters. These filters, represented by small matrices, traverse the input image, performing a dot product with a small region of the input at each step. The outcome of this process is a collection of feature maps that effectively represent distinct features present in the input image. By leveraging these learnable filters, the convolutional layer excels at capturing intricate patterns and features, making it a pivotal element in the success of Convolutional Neural Networks for image-related tasks.

- **Pooling Layer**: The pooling layer (PoolLayer) plays a crucial role in diminishing the spatial dimensions of the feature maps generated by the convolutional layer [39]. Operating independently on each feature map, this layer achieves

downsampling by either extracting the maximum or average value from non-overlapping regions. The primary objectives of pooling are twofold: first, it contributes to lowering the computational complexity of the network, and second, it enhances the network's resilience to minor translations in the input image. By strategically reducing the spatial resolution while retaining essential features, the pooling layer aids in creating a more computationally efficient and robust Convolutional Neural Network, particularly beneficial for image-related tasks.

- **Activation Layer**: The activation layer serves a critical role by applying a non-linear activation function to the output of the preceding layer [81]. This introduction of non-linearity is pivotal as it empowers the network to learn and represent more complex features. Without non-linear activation functions, the network's ability to capture intricate patterns and relationships within the data would be severely limited. By incorporating non-linearities, such as Rectified Linear Unit [1] or sigmoid functions, the activation layer enables the Convolutional Neural Network to surpass the representational capacity of linear models. This non-linear transformation is fundamental for the network's capability to discern and comprehend intricate structures and patterns in the input data, essential for tasks like image recognition and classification.

- **Batch Normalization Layer**: The batch normalization layer plays a crucial role in the normalization of the output from the preceding layer [113]. This normalization process involves subtracting the mean and dividing by the standard deviation of the batch. By doing so, the batch normalization layer addresses

[1] ReLU

the issue of internal covariate shift, contributing to enhanced convergence in the network. Internal covariate shift refers to the undesirable fluctuation in the distribution of internal activations during the training process, which can impede the convergence of the model. The batch normalization layer mitigates this problem by ensuring a more stable distribution of activations, thereby facilitating more efficient and faster training. Overall, batch normalization is a pivotal technique in improving the training dynamics and performance of Convolutional Neural Networks.

- **Dropout Layer**: The dropout layer is a crucial element in preventing overfitting during the training of a neural network [31]. Its primary function is to randomly deactivate a percentage of neurons in the preceding layer during the training process. By doing so, the dropout layer introduces an element of randomness and variability into the network, preventing the model from becoming overly reliant on specific neurons. This stochastic dropout mechanism forces the network to learn more robust and generalized features, enhancing its ability to make accurate predictions on unseen data. Overfitting occurs when a model learns the training data too well, including its noise and fluctuations, leading to reduced performance on new, unseen data. The dropout layer acts as a regularization technique, promoting the creation of a more resilient and adaptable neural network, ultimately improving its overall generalization capabilities.

- **Fully Connected Layer**: The fully connected layer is a fundamental component of a neural network, representing a traditional layer where each neuron is connected to every neuron in the preceding and subsequent layers [120]. Also known as a dense layer, its purpose is to consolidate and integrate the features

learned in the earlier layers to generate a final output. Typically situated at the end of the network, the fully connected layer transforms the high-dimensional feature representations into a probability vector or a set of scores corresponding to different classes in the dataset. During training, the network adjusts the weights associated with each connection through backpropagation, optimizing them to minimize the difference between predicted and actual outputs. This process enables the fully connected layer to capture complex relationships within the data and produce a meaningful and contextually relevant final prediction.

The various layers within a CNN collectively work to acquire and extract features from input images, culminating in the production of precise predictions or classifications. A nuanced comprehension of each layer's role empowers researchers and practitioners to devise more potent CNN architectures tailored to a broad spectrum of computer vision applications [70].

Convolution Operation

The convolution operation (ConvOpt) is a prevalent mathematical procedure extensively applied in signal processing, image processing, and computer vision [73]. Its primary function is to amalgamate two signals or functions, generating a third signal that signifies the impact of one signal upon the other, proportionally weighted by the shape of the latter. In the realm of computer vision, convolution serves the purpose of feature extraction from images through the utilization of Convolutional Neural Networks.

In the domain of computer vision, the convolution operation is instrumental for

feature extraction from images through the utilization of Convolutional Neural Networks. The ConvOpt, a foundational process in CNNs, entails sliding a filter or kernel across an input image, computing the dot product at each position, and thereby generating an output feature map.

The ConvOpt is executed independently for each channel of the input image, and the resultant feature maps are amalgamated to constitute the output. The filter size is a pivotal parameter determining the receptive field's dimensions, influencing the output at a given position. The stride dictates the spacing between adjacent filter positions, and the degree of zero padding around the input image can be tuned to regulate the output feature map's size and resolution.

Several methods, including the direct method, Fourier method, and fast Fourier transform (FFT) method, can perform the ConvOpt [60]. While the direct method is the most straightforward, it may be computationally demanding for extensive input volumes and filters. In contrast, the Fourier methods leverage the convolution theorem, indicating that spatial convolution is equivalent to frequency domain multiplication, thereby accelerating the operation. In summary, convolution is a mathematical process merging two signals or functions to produce a third, representing the influence of one signal on the other. In computer vision, convolution leverages CNNs for feature extraction, involving filter sliding and dot product computation to generate an output feature map. Adjustable parameters like filter size, stride, and zero padding impact the output's size and resolution. The convolution operation can be conducted through various methods, and the choice of activation function holds significance.

Feature Maps

In the realm of Convolutional Neural Networks, a feature map (FeatureMap) stands as a pivotal element, signifying the output of a convolutional layer [56]. The FeatureMap is a two-dimensional array that mirrors the extent to which local regions of the input image align with the filters applied to them. Each element within the FeatureMap corresponds to the activation of a neuron in the layer, encapsulating a specific facet of the input image.

The computation of the FeatureMap involves convolving a set of filters with the input image. Each filter generates a single-channel FeatureMap that accentuates a distinct feature or pattern within the input image. Collating the FeatureMap produced by various filters in the same layer results in a multi-channel FeatureMap, encompassing diverse aspects of the input image.

The computation of a Feature Map entails the systematic movement of a filter across the input image, calculating the dot product between the filter and the corresponding local region of the input at each position. The accumulated values undergo summation and pass through an activation function, ultimately yielding the conclusive values of the FeatureMap. This iterative process is replicated for each filter within the layer, resulting in a collection of FeatureMap that collectively encapsulates diverse aspects of the input image.

In essence, the Feature Map proves to be a potent mechanism within Convolutional Neural Networks, empowering them to discern and encapsulate pivotal features from the input image. This capability positions CNNs to proficiently execute an extensive array of visual recognition tasks.

Activation Functions

Activation functions (ActivFuncs) play a vital role in Convolutional Neural Networks by introducing non-linearity into the model [30, 43]. This non-linearity is essential as it allows the model to capture the complex, non-linear behavior exhibited by many real-world phenomena. A model relying solely on linear operations may fall short in accurately representing such phenomena. Therefore, ActivFuncs enhance the expressiveness of CNNs, enabling them to model a broader spectrum of functions.

Figure 2.3 indicates commonly employed ActivFuncs in CNNs, encompassing the rectified linear unit (ReLU), the sigmoid function, and the hyperbolic tangent (tanh) function. Notably, ReLU stands out as the most prevalent ActivFunc in contemporary CNNs due to its simplicity and efficacy in mitigating the vanishing gradient problem during training.

Mathematically, the ReLU ActivFunc is defined as:

$$Relu(z) = max(0, z) \tag{2.1}$$

where z is the input to the neuron. This function returns the maximum of 0 and z, effectively "turning off" any negative values and leaving positive values unchanged. This simple nonlinear function allows for better learning of complex features in CNNs and helps to prevent the saturation of neurons during training.

The sigmoid function, on the other hand, is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

where z is the input to the neuron. The sigmoid function has a characteristic S-shape

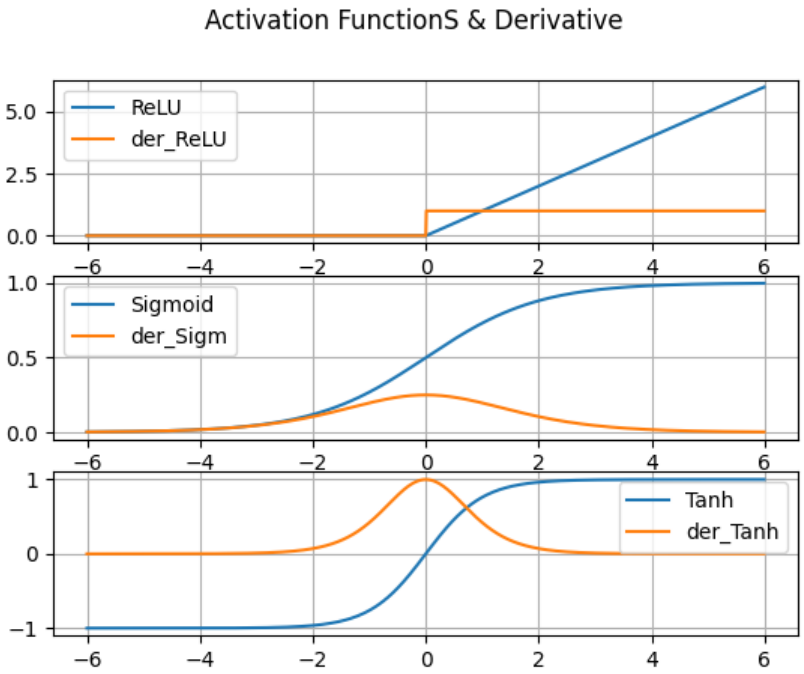

Figure 2.3: Commonly Used CNN Activation Functions and Derivatives

and maps any real number to a value between 0 and 1, making it a good choice for binary classification problems. However, it suffers from the vanishing gradient problem when used in deep neural networks due to its saturating nature, which can slow down the training process.

The hyperbolic tangent (tanh) function is also similar to the sigmoid function, but maps any real number to a value between -1 and 1:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{1 - e^{-2z}}{1 + e^{-2z}} \tag{2.3}$$

It is also a non-linear function that is useful for introducing non-linearity into the model. However, like the sigmoid function, it can suffer from the vanishing gradient problem in deep neural networks.

Beyond the commonly used Activation Functions, additional functions, like the softmax function, play critical roles in Convolutional Neural Networks. The softmax function, specifically employed in the output layer of CNNs for multi-class classification problems, transforms the last layer's output into a probability distribution across classes. This characteristic makes it particularly suitable for tasks involving probability-based classification.

Softmax function is a multi-class single-label classification:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2.4}$$

The selection of ActivFuncs in CNNs significantly influences model performance and convergence. Each function possesses distinct strengths and weaknesses. Consequently, choosing the appropriate function tailored to the specific task is a crucial aspect of designing a successful CNN.

Padding, Stride, and Filters

In Convolutional Neural Networks, the parameters of padding, stride, and filters play a pivotal role in shaping the size and resolution of the output feature maps. These parameters are instrumental in controlling the information retained in the FeatureMaps and hold significant sway over the network's performance.

Padding entails the addition of extra rows and columns of zeros around the input image before applying filters. This step helps maintain the spatial dimensions of the image as it traverses through the convolutional layers. Padding is often utilized to ensure that the output FeatureMaps possess the same spatial dimensions as the input image or to regulate the size of the output FeatureMaps.

Stride represents another crucial parameter in CNNs, determining the distance between adjacent filter positions as the filter slides over the input image. The adjustment of stride allows control over the resolution of the output FeatureMaps. A larger stride yields a lower-resolution output FeatureMap, whereas a smaller stride produces a higher-resolution output FeatureMap.

Filters, essential in Convolutional Neural Networks, are small matrices applied to the input image to generate the FeatureMap. These filters undergo learning during the training process, with their values updated through backpropagation. The size of the filter stands out as a crucial parameter influencing the size and intricacy of the features it can capture. Larger filters possess the capability to capture more complex features but come with increased computational demands and a higher susceptibility to overfitting.

The dimensions of the output FeatureMap hinge on various factors, including the size of the input image, the size of the filter, the stride, and the amount of padding. The calculation of the output FeatureMap size follows a specific formula:

$$\frac{W - F + 2P}{S} + 1 \tag{2.5}$$

where W is the width (or height) of the input image, F is the width (or height) of the filter, P is the amount of padding, and S is the stride. This formula gives the number of positions where the filter can be applied to the input image to produce the output FeatureMap.

Tailoring the padding, stride, and filter size is a pivotal aspect of crafting a CNN architecture capable of adeptly capturing input image features and generating high-quality FeatureMaps. These parameters offer a level of adaptability in designing

convolutional layers, enabling customization based on the specific task at hand. For instance, a network geared towards image classification might employ larger filters and smaller strides to capture intricate features, while a network designed for object detection could opt for smaller filters and larger strides to streamline computation and expedite the detection process [63].

In essence, comprehending the significance of padding, stride, and filters in CNNs is imperative for devising neural network architectures that effectively discern and categorize intricate visual patterns in images and other signal data.

2.3 COVID-19 Detection-Related Work

In this section, literature reviews are provided about COVID-19 detection, particularly Binary classification problem, either through CT or X-ray medical images, by applying AI techniques.

This section is structured into two primary categories: COVID-19 detection through X-ray images and COVID-19 detection through CT images. Notably, two additional categories, COVID-19 detection through Ultrasound images and multi-modal images, are not addressed in this research.

The organization of the review is further delineated based on two key factors: the type of classification employed and the corresponding accuracy percentage. Papers are categorized based on whether they undertake a binary classification resulting in an accuracy of more than 90% or a binary classification yielding an accuracy of less than 90%. This classification scheme provides a systematic approach to evaluating and comparing the efficacy of different methodologies in the context of COVID-19 detection using X-ray and CT images [59].

2.3.1 COVID-19 detection in X-Ray Images

Accuracy less than 90%

Hemdan et al. [46] developed the COVIDX-Net framework for COVID-19 detection, employing seven CNN architectures, including VGG19 and DenseNet-201, which demonstrated the highest accuracy in classification with F1-scores of 0.89 and 0.91 for normal and COVID-19 cases, respectively. Their dataset comprised 50 X-ray images evenly split between normal and COVID-19 cases.

Catak et al. [98] introduced five deep CNN techniques, including VGG16, VGG19, ResNet, DenseNet, and InceptionV3, for identifying COVID-19 from X-ray images. VGG16 achieved the highest accuracy at 80%, utilizing a dataset with 50 COVID-19 patients and 50 non-COVID-19 patients in the training phase, and 20 cases each of COVID-19 and non-COVID-19 in the testing phase.

Horry et al. [47] developed pre-trained models, including Xception, VGG16, VGG19, Inception v3, and ResNet-50, achieving the highest precision of 83% with VGG19 for COVID-19 detection from a dataset of 115 COVID-19 images.

Haghanifar et al. [42] created the CheXNet model based on Xception, DenseNet, EfficientNet-B7, and ResNet, achieving an overall accuracy of 87.88% for classifying X-ray images into COVID-19, CAP, and normal categories. Their extensive dataset included 1326 COVID-19 images, 5000 normal images, and 4600 CAP images.

Accuracy more than 90%

Ozturk et al. [82] developed an automatic COVID-19 detection model from X-ray images, utilizing a CNN with 17 convolution layers and incorporating both binary (COVID vs. no findings) and multi-class classification (COVID vs. no findings vs.

pneumonia). Employing the YOLO (You Only Look Once) object detection system, the model achieved an impressive accuracy of 98.08% for binary classification and 87.02% for multi-classification.

Apostolopoulos & Mpesiana [15] proposed the use of VGG19 for COVID-19 detection in chest X-ray images, achieving an accuracy of 93.48% for binary classification and 98.75% for multi-classification. The dataset encompassed 700 pneumonia, 504 normal, and 224 COVID-19 X-ray images.

Another article, Minaee et al. [75] introduced the DeepCOVID model, utilizing 5071 X-ray images to distinguish COVID-19 from other lung pneumonia. Trained through four CNNs, including ResNet-50, DenseNet-191, and SqueezeNet, the model generated a heatmap to identify COVID-19-infected regions. SqueezeNet demonstrated the highest performance, achieving 95.6% specificity and 100% sensitivity. Moving on, Narin et al. [78] developed CNN models for three datasets, with ResNet-50 achieving the highest accuracy of 96.1%, 99.5%, and 99.7% for respective datasets. Singh et al. [104] devised a CNN and Multi-Objective Differential Evaluation (MODE) classifier-based detection model, achieving an accuracy of 94.65% by extracting essential information through clustering and statistical approaches. Pandit et al. [84] proposed a DCNN model for COVID-19 detection, achieving an accuracy of 93%.

In the other prior art, Zhang et al. [122] developed a deep anomaly model for COVID-19 screening through X-ray images, achieving 96% accuracy for COVID-19 cases and 70.65% for non-COVID-19 cases. However, the study acknowledged limitations in false positive rates and missed COVID-19 cases. Transitioning to Alqudah et al. [13], they introduced a COVID-19 identification model based on Support Vector Machine (SVM), Random Forest (RF), and CNN. While SVM exhibited less time

consumption, CNN achieved 95.2% accuracy in the test stage. Finally, Hossain et al. [48] applied ResNet50 with various pre-trained weights to 7262 X-ray images, achieving impressive scores of 99.17%, 99.31%, and 99.03% for accuracy, precision, and sensitivity, respectively.

In conclusion, the extensive review of literature on COVID-19 detection models in chest X-ray images underscores the critical advancements and implications for medical diagnostic imaging and the ongoing battle against the pandemic. The reviewed studies have demonstrated the potential of deep learning techniques, especially convolutional neural networks (CNNs), in accurately identifying COVID-19 from X-ray images. The research on binary classification models for COVID-19 detection in chest X-ray images has yielded diverse outcomes. Several studies, such as those by Hemdan et al., Catak et al., Horry et al., and Haghanifar et al., have demonstrated the development and application of various CNN architectures for COVID-19 detection, achieving accuracies below 90%. Conversely, research by Ozturk et al., Apostolopoulos & Mpesiana, Minaee et al., Narin et al., Singh et al., Pandit et al., Zhang et al., Alqudah et al., and Hossain et al. has showcased the successful implementation of CNN models with accuracies exceeding 90%. These findings underscore the potential of deep learning techniques in accurately identifying COVID-19 from X-ray images, paving the way for enhanced diagnostic capabilities in the context of the ongoing pandemic [59].

2.3.2 COVID-19 detection in CT-Scan Images

Accuracy less than 90%

Shah et al. [99] developed the CT-Net10 model for classifying CT-scan images into COVID-19 or non-COVID-19 categories, achieving an overall accuracy of 82.1%. Employing various models such as VGG16, ResNet-50, Inception V3, DenseNet, and VGG19, VGG19 emerged as the superior model, attaining an accuracy of 95.52%. The dataset comprised 738 CT-scan images, including 349 COVID-19 images from 216 patients.

Shuai Wang et al. [112] employed a DL model to investigate radiographic changes in CT images, utilizing the adjusted inception transfer learning technique on 1065 CT images. The model yielded an accuracy of 89.5% and a sensitivity of 0.87. However, challenges such as poor signal-to-noise ratio, complicated data integration, a small training dataset, and the presence of numerous variable objects affected the efficiency of DL. Future work aims to enhance diagnosis through multi-modeling analysis of clinical information and genetic features linked with CT image hierarchical features.

Following my exploration, Amyar et al. [14] developed a multitask DL model for COVID-19 detection from chest CT images, determining disease severity through segmentation of the infected region and making a reconstruction. The dataset, sourced from multiple hospitals, included 1369 CT images and achieved an accuracy of 86% and an Area Under the Curve (AUC) of 0.93. Xiong et al. [19] applied an AI-based system to distinguish COVID-19 from other pneumonia, segmenting chest CT images based on HU with a 320-thresholding value. The segmented region was then input to the EfficientNet B4 deep neural network, achieving an accuracy of 87% and an AUC of 0.90 on a dataset comprising 512 COVID-19 CT images and 665 non-COVID-19

pneumonia images.

Accuracy more than 90%

Wang et al. [114] developed the Decov Net framework based on Unet and CNN to detect COVID-19 from CT images. The model utilized Unet for lung segmentation and DNN to predict COVID-19 infection probability, achieving 90.9% accuracy and 95.9% AUC (Area Under the Curve) on a dataset containing 219 non-COVID and 313 COVID-19 CT images. Despite some limitations, including imperfect Unet training and a single-hospital dataset, the study demonstrated promising results.

Do and Vu [32] explored various transfer learning models for COVID-19 detection in CT-scan images, with DenseNet-201 exhibiting the highest accuracy of 85% and recall of 91%. Future work aims to investigate stacking multiple architectures and integrating different imaging modalities into a single model. In a similar vein, Attallah et al. [18] developed a CAD system based on multiple CNNs, achieving 94.7% accuracy and 0.98 AUC on a dataset comprising 347 COVID and 347 non-COVID CT images. Challenges included the need for a larger training dataset and limited support for segmentation techniques beyond lung differentiation.

Meanwhile, Gozes et al. [40] introduced an automated analysis tool for COVID-19 progress tracking, achieving 98.2% sensitivity and 92.2% specificity on 157 patients' CT images. Shan et al. [101] proposed VB-Net for quantification and segmentation of COVID-19-infected regions, achieving a 91.6% Dice similarity coefficient and 0.3% mean Point Of Interest (POI) estimation error. However, limitations included a single-center dataset, and the system only quantified COVID-19 infection. Chen et al. [22] utilized UNet++ and pre-trained ResNet-50 to distinguish COVID-19 from

other pneumonia, achieving 100% sensitivity and 95.2% accuracy on a large dataset of 20,886 CT images. Jin et al. [57] developed an AI system based on ResNet-50 and 3D Unet++ for COVID-19 detection, achieving 94.8% accuracy and 97.4% sensitivity on a dataset from five different centers.

Similarly, Abbasian Ardakani et al. [7] created COVIDag, a CAD system using various classifiers, achieving a 96.5% AUC. Future work involves developing a model to estimate the severity of COVID-19 infection. Afify et al. [9] developed a CAD system based on 200 CT-scan images, utilizing lung segmentation and genetic algorithm-based feature selection. KNN achieved 100% accuracy, while the decision tree achieved 95%. Saeedi et al. [97] designed an online CAD system using DenseNet-121, achieving 90.61% accuracy and 90.80% recall on a dataset of 349 COVID-19 and 397 non-COVID-19 patients.

Ardakani et al. [16] created a CAD system based on ten pre-trained convolution layers, achieving 100% sensitivity for ResNet-101 and 98.04% for Xception. ACAR et al. [8] developed a CAD system using CT images with a low dose and CNN methods, reaching 99.5% accuracy and sensitivity for COVID-19 detection. Swapnarekha et al. [106] applied ResNet-50 V2 and DenseNet-201, achieving high accuracy, specificity, and sensitivity. Mete et al. [80] employed various deep-learning approaches, with SVM and ResNet-50 showing the highest performance with 96.29% accuracy, 95.86% F1-score, and 0.9821 AUC. Kogilavan et al. [61] proposed different deep learning models, with VGG16 achieving the best performance with 97.68% accuracy on 3873 CT images.

In conclusion, the comprehensive analysis of the literature on binary classification models for COVID-19 detection in CT-scan images reveals significant advancements

and implications for the field of medical imaging and disease diagnosis. The reviewed studies have demonstrated the potential of deep learning techniques, particularly convolutional neural networks (CNNs), in accurately distinguishing COVID-19 from non-COVID-19 cases in CT-scan images. The findings underscore the potential of deep learning techniques in accurately identifying COVID-19 from CT-scan images, paving the way for enhanced diagnostic capabilities and improved disease surveillance in the context of the ongoing pandemic [59].

Chapter 3

Model Development

This chapter introduces the components of developing the computational model, including the CNN architectures for model selection, as well as optimizing the trainable hyperparameters in each model using the automatic optimization method.

Achieving optimal model performance necessitates a comprehensive grasp of the problem, insights into data transformation and enrichment, and understanding of the development steps. This knowledge is pivotal for determining the most effective preprocessing techniques. The design of proposing any optimal computational model involves different considerations that need careful consideration at each step to achieve better classification accuracy on average.

In various studies, researchers leverage DL algorithms, particularly Convolutional Neural Networks, a subset of Artificial Neural Networks (ANNs). Comprising four types of layers—convolution, pooling, fully linked, and non-linearity—CNNs enhance pattern recognition and image classification [11]. Researchers exploring novel coronavirus detection have employed diverse CNN architectures, including Visual Geometry Group (VGG), Residual Convolution Neural Network (ResNet), and Dense Convolution Network (DenseNet). However, the choice of CNN architectures is tailored to

the dataset's size and their characteristics.

Figure 3.1 visually illustrates the overall procedure of obtaining an optimized model for Image classification grounded in Deep Learning (DL), emphasizing the integration of advanced neural network architectures in the diagnostic framework. In this figure, the model can be optimized by fine-tuning the hyperparameters during the model evaluation through a certain number of iterations.

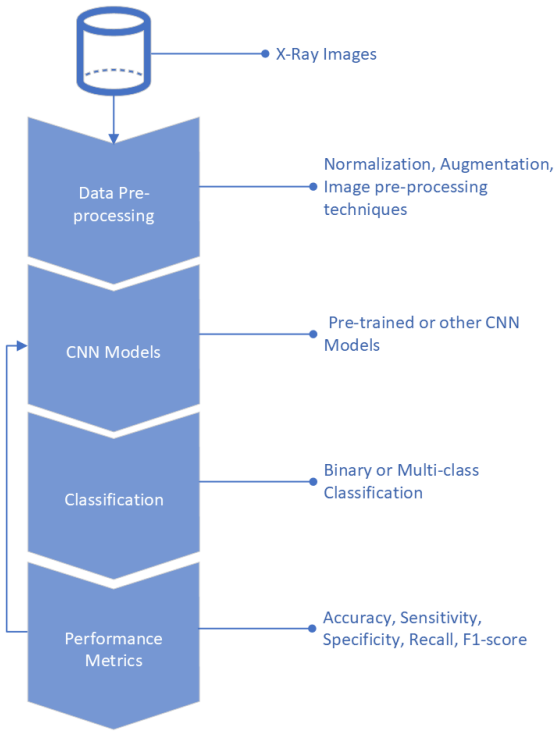

Figure 3.1: The basic architecture of model development for image classification task

3.1 Convolutional Neural Networks

Deep Learning algorithms surpass classical machine learning algorithms in some aspects, particularly when handling extensive datasets and raw images. DL excels at extracting knowledge and information even without the necessity for preprocessing,

enhancement, or segmentation of images. This capability extends to image analysis, where DL algorithms demonstrate notable enhancements [69]. Their application extends to disease detection, notably in areas like identifying COVID-19 and diagnosing retinal diseases affecting the iris and delicate nerves, potentially leading to blindness [25]. The proficiency of DL algorithms in processing vast and unprocessed data underscores their efficacy in complex tasks, making them pivotal in advancing medical diagnostics and image-based analyses.

3.1.1 Selection of CNN Models

In recent times, CNN architectures have shown superior performance in complex tasks, particularly in medical image analysis. The most popular models include the following, with further details of their structure provided in Appendix A, Chapter A.2.

LeNet

LeNet, created by Yann L.C. et al. in 1998, stands as one of the initial CNN designs, specifically devised for handwritten digit recognition. The LeNet structure comprises two convolution layers succeeded by two fully connected layers. These convolution layers employ diminutive filters and pooling layers to extract features from the input image. The fully connected layers, utilizing softmax activation, produce a probability distribution across potential classes [58].

AlexNet

AlexNet, created by Alex Krizhevsky, represents an evolution from LeNet, featuring increased depth with additional filters, stacked convolution layers, dropout, and max pooling. In 2012, AlexNet achieved a remarkable 17% top-five error rate and secured victory in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) contest, an annual competition held from 2010 to 2017 [64]. Researchers commonly leverage AlexNet in COVID-19 detection efforts to address the challenge of overfitting. Overfitting occurs when deep learning models exhibit higher accuracy on training data compared to testing data, and AlexNet proves effective in mitigating this issue.

GoogLeNet

GoogLeNet surpasses AlexNet in depth, boasting 22 or 27 layers when considering pooling layers. In 2014, GoogLeNet achieved a remarkable victory in the ILSVRC contest with a 6.67% top-five error rate. A pivotal element of GoogLeNet is the inception module (IM), functioning as a miniature network capable of learning spatial and cross-channel correlations. The IM offers several advantages, including enabling the training of significantly deeper models with ten times fewer learnable parameters. The IM's output features fewer maps than its input, effectively reducing dimensionality. Moreover, the IM excels in capturing intricate patterns at various scales within both spatial and depth dimensions [107].

VGGNet

VGGNet, with its 19 convolution layers, achieved a notable 7.3% top-five error rate in the 2014 ILSVRC contest. Unlike AlexNet, VGGNet embraces architectural simplicity, featuring three fully connected layers. Developed by the Visual Geometry Group at Oxford University, VGGNet's straightforward design contributes to its widespread adoption in various domains. Despite its simplicity, VGGNet employs three times as many parameters as AlexNet. The architecture serves as a foundation for advanced object identification models and outperforms baselines across diverse tasks and datasets beyond ImageNet. VGGNet's enduring popularity persists as one of the most utilized image recognition architectures [103].

ResNet

ResNet, the winner of the 2015 ILSVRC contest with 152 layers, introduced a revolutionary residual module featuring a standard layer and a skip connection. This unique architecture achieved a remarkable 3.6% top-five error rate. The skip connection, linking the input signal of a layer to its output, facilitates the traversal of the input signal across the network. Residual Units (RUs) empowered the training of an exceptionally deep 152-layer model. A block is formed by connecting layer activations to subsequent layers, and these building blocks are stacked to construct ResNets. The skip link's advantage lies in regularization, as it skips any layer detrimental to architecture performance, enabling the training of extremely deep neural networks without grappling with issues like vanishing or exploding gradients [44].

Inception

Inception, initially introduced by Google in 2014 as Inception V1, revolutionized image model blocks by simulating an optimal local sparse structure in CNNs. This architecture strategically employs numerous filters of different sizes on the same level to mitigate data overfitting and address computational expense issues. Unlike traditional approaches using a single filter size, Inception combines multiple filter sizes into a unified image block before passing it to the subsequent layer. This innovative design enhances the model's ability to capture diverse features within the input data, contributing to its effectiveness in various computer vision tasks [88].

Xception

Xception, a creation of the Google team, is designed with depth-wise separable convolutions, earning its name from "extreme Inception." Considered as an interpretation of Inception modules, Xception comprises three main structures: entry flow, middle flow, and exit flow. These structures consist of 14 modules each, totaling 36 convolution layers. The entry flow and the middle flow (repeated eight times) guide the data through the initial processing steps. Notably, batch normalization is applied after both the convolution and separable convolution layers in Xception, reflecting its emphasis on optimizing network performance [88].

MobileNet

MobileNet stands out as a widely embraced CNN-based model for image classification, particularly for its suitability in resource-constrained environments. The architecture's key advantage lies in its significantly reduced computational demands

compared to conventional CNN models, making it particularly well-suited for deployment on mobile devices and computers with limited processing power. Featuring depth-wise separable convolution layers and ReLU non-linearity, MobileNet concludes with a fully connected layer followed by the SoftMax classification layer. This design introduces a valuable trade-off between latency and precision, allowing model builders to select an appropriate model size based on their application needs while accommodating computational constraints [49, 71].

DenseNet

DenseNet emerges as a distinctive convolutional neural network, boasting a noteworthy top-five error rate of 6.12%, all the while being more computationally efficient than other leading CNN architectures like ResNet. The key innovation lies in the incorporation of Dense Blocks, establishing direct connections between all layers when their FeatureMap sizes align. This novel approach enables dense connections, wherein each layer receives additional inputs from all preceding layers and forwards its own FeatureMaps to subsequent layers. In contrast to conventional CNN architectures with L connections between L layers, DenseNet boasts $L(L+1)/2$-layer connections. Leveraging the FeatureMaps of all preceding layers as inputs and propagating its own FeatureMaps to subsequent layers, DenseNet overcomes challenges such as vanishing gradients, enhances feature propagation, encourages feature reuse, and achieves notable parameter reduction [50].

WideResNet

Wide Residual Networks (WideResNet) are a variation of the original Residual Networks (ResNet) that address the problem of diminishing feature reuse in deep networks. Introduced by Sergey Zagoruyko and Nikos Komodakis in their 2016 paper "Wide Residual Networks" [121]. WideResNet modifies the original ResNet architecture by increasing the width of residual blocks, which means adding more filters in each convolutional layer, while simultaneously reducing the depth (number of layers).

SqueezeNet

SqueezeNet, proposed by Iandola et al. [52], is a compact CNN architecture that achieves AlexNet-level accuracy on ImageNet while using 50 times fewer parameters. By employing model compression techniques, the authors successfully reduced SqueezeNet's size to less than 0.5MB, making it highly popular for applications requiring lightweight models.

3.1.2 The Choice of Hyperparameters

The key trainable hyperparameters in CNN models have selected based on the different concepts to be fine-tuned during the model evaluation, including regularization, batch normalization, backpropagation, and gradient optimizers, with detaled mathematical definition provided in Appendix A, Chapter A.1.

Regularization Techniques

To prevent overfitting in neural networks, regularization techniques are employed. Overfitting occurs when a model performs well on the training data but poorly on

new, unseen data. These techniques involve adding a penalty term to the LossFunc, encouraging the model to possess simpler weights or reduce the magnitude of the weights [111].

Various regularization techniques are commonly used, and some of them include:

- L1 regularization: adds a penalty to the LossFunc that is proportional to the absolute value of the weights. This encourages the model to have sparse weights and can potentially lead to feature selection [115].

- L2 regularization: adds a penalty to the LossFunc that is proportional to the square of the weights. This encourages the model to have small weights and aids in preventing overfitting [94].

- Dropout: is a regularization technique that involves randomly dropping out some neurons in the network during training. This prevents the model from relying too heavily on any specific feature and can enhance generalization performance [123].

- Early stopping: is a regularization technique that halts the training process early based on a performance metric evaluated on a validation set. This helps prevent the model from overfitting to the training data. The concept involves monitoring the validation loss or accuracy throughout training and terminating the training process when the validation performance ceases to improve [67].

Table 3.1 summarizes commonly utilized regularization techniques for preventing models from overfitting in neural networks [63].

Regularization Technique	**Description**
L1 regularization	Imposes a penalty on the loss function proportional to the absolute value of the weights
L2 regularization	Imposes a penalty on the loss function proportional to the square of the weights
Dropout	Randomly drops some neurons during training to avoid over-reliance on any single feature and enhance generalization
Early stopping	Terminates the training process early based on validation set performance metrics to prevent overfitting to the training data

Table 3.1: Comparison of Regularization Techniques

Batch Normalization

Normalization layers are crucial in deep network training, and among the prominent normalization techniques, batch normalization (BN) has proven effective in enhancing model training speed and generalization capability [54]. The success of BN is attributed to various factors, including the reduction of internal covariate shift, enabling the use of larger learning rates, and smoothing the optimization landscape.

In Deep Neural Network (DNN) training, a crucial aspect is normalizing the training data and intermediate features. The normalization of input data is known to expedite training. Batch normalization, a widely employed technique, extends this concept to intermediate layers within a deep network by normalizing samples in a mini-batch during training. BN has demonstrated its ability to accelerate training speed, facilitate the use of larger learning rates, and enhance model generalization accuracy. As a result, BN has become a fundamental component in popular network architectures like ResNet and DenseNet.

In addition to its role in accelerating convergence, batch normalization offers a

regularization capability. Due to the updating of sample mean and variance on mini-batches during training, these values are not precise. Consequently, BN introduces a degree of noise, akin to the effect of dropout. This noise contributes to enhancing the generalization capability of the trained model.

While batch normalization has demonstrated significant improvements in terms of enhancing training speed and model generalization across various applications, it faces challenges with small batch sizes. In such cases, the sample mean and variance might deviate significantly from those of the entire population, impacting BN's effectiveness. To mitigate this issue, batch renormalization (BReN) was introduced [53], constraining the range of estimated mean and variance within a batch. However, the practical application of BReN is hindered by the difficulty in tuning its hyperparameters, limiting its versatility across different tasks.

Back Propagation

Backpropagation is an algorithm that effectively calculates the LossFunc's gradients concerning a neural network's weights and biases. This method involves propagating the error from the OutLayer back through the network and determining the derivative of the LossFunc with regard to each parameter using the chain rule of calculus. Backpropagation is typically combined with an optimization algorithm to iteratively update the weights and biases of the network during the training process [28, 91].

Mathematically, the backpropagation algorithm can be represented as following steps:

1. Compute the output of the network for a given input.
2. Compute the error between the predicted output and the true output.

3. Compute the gradient of the LossFunc concerning the network output.
4. Use the chain rule to compute the gradient of the LossFunc concerning the weights and biases of each layer in the network:

$$\begin{aligned} \frac{\partial L}{\partial w_{i,j}^{(l)}} &= \frac{\partial L}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial w_{i,j}^{(l)}} \\ \frac{\partial L}{\partial b_i^{(l)}} &= \frac{\partial L}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial b_i^{(l)}} \end{aligned} \tag{3.1}$$

   where L is the LossFunc, $w_{i,j}^{(l)}$ is the weight connecting neuron i in layer l − 1 to neuron j in layer l, $b_i^{(l)}$ is the bias of neuron i in layer l, and $z_i^{(l)}$ is the weighted sum of the inputs to neuron i in layer l.
5. Update the weights and biases of the network using an optimization algorithm.
6. Repeat steps 1–5 for each training example.
7. Repeat steps 1–6 for a fixed number of epochs or until convergence.

Optimization Algorithms

Optimization algorithms play a crucial role in adjusting the weights and biases of a neural network during training, aiming to minimize the LossFunc [79,117]. Among the widely used optimization algorithms, stochastic gradient descent (SGD) is prevalent. SGD updates the weights and biases incrementally, taking small steps determined by the gradient of the LossFunc concerning these parameters.

Other optimization algorithms are as follows:

- Momentum: This algorithm adds a momentum term to the gradient update, which helps to smooth out the updates and accelerate convergence.

- AdaGrad: is an optimization algorithm that dynamically adjusts the learning rate for each weight based on the magnitude of the gradients observed during training. This adaptive learning rate facilitates faster convergence on flat directions and slower convergence on steep directions.

- RMSProp: is an optimization algorithm that incorporates a moving average of the squared gradients to dynamically adjust the learning rate for each weight. This adaptation aids in preventing the learning rate from becoming excessively large.

- Adam: is an optimization algorithm that integrates concepts from both momentum and AdaGrad, resulting in an adaptive learning rate approach suitable for a broad spectrum of neural networks.

Table 3.2 summarizes the optimization algorithms that are commonly used to update the weights and biases of neural networks during training in order to minimize the loss function [63].

Optimization Algorithm	**Description**
SGD	Adjusts weights and biases according to the gradient of the loss function
Momentum	Incorporates a momentum term to smooth updates and speed up convergence
AdaGrad	Modifies the learning rate for each weight based on the magnitude of past gradients
RMSProp	Utilizes a moving average of squared gradients to adjust the learning rate for each weight
Adam	Merges concepts from momentum and AdaGrad to form an adaptive learning rate algorithm

Table 3.2: Comparison of Optimization Algorithms

3.2 Transfer Learning

Transfer Learning involves the transfer of knowledge from one network or model to another, as illustrated in Figure 3.2. Convolutional neural networks, known for autonomously learning to interpret images, excel at creating high-quality features. Hence, pre-trained convolutional neural networks often serve as ideal feature extractors in many scenarios.

Leveraging pre-trained models with ImageNet weights, as demonstrated in studies like [92], yielded acceptable heatmaps; however, the visualization was limited to only a few images. Recognizing the constraints of small dataset sizes, it was suggested to fine-tune models that were previously trained on similar data.

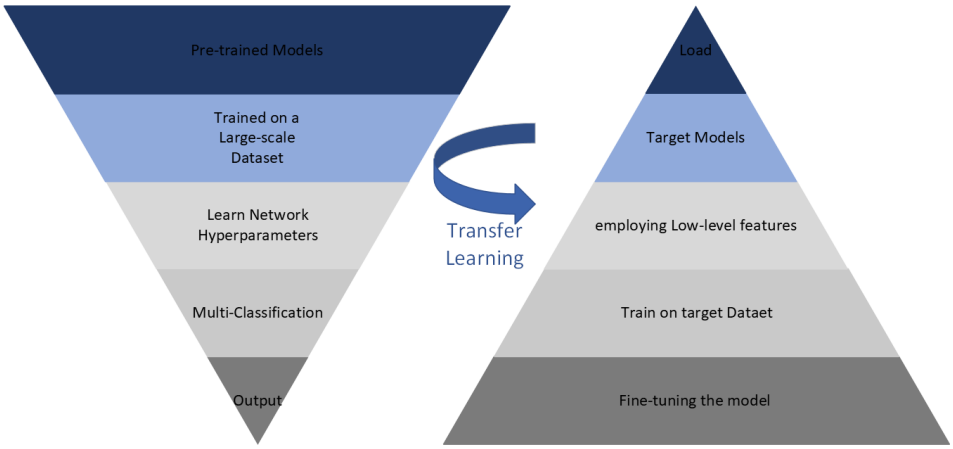

Figure 3.2: Transfer Learning Workflow

3.3 Bayes Optimization

Finding the best set of hyperparameters to improve or degrade the fitness function is one of the most challenging aspects of any Deep Learning project. This is particularly difficult due to the wide range of available neural network topologies. Creating a

fixed search space becomes even more challenging when considering the numerous criteria involved. The objective of this study was to address this issue by constructing a fitness function and a search space domain encompassing key hyperparameters such as gradient optimization functions, the choice of model, batch normalization, regularization, momentum, and learning rate. Noteworthy models like SqueezeNet, COVIDNet, ResNet, DenseNet, and VGG16 have been pivotal in related studies, showcasing the synergy between pre-trained models and hyperparameter tuning in different dataset sizes [75, 76].

Bayesian optimization is a method employed for optimizing an objective function, denoted as $f(x)$ so called underlying or latent [37, 100]. This approach is handy in situations where observations are costly, subject to noise, and there is no expression for $f(x)$. In such cases, the goal is to determine values that not only yield substantial information but also minimize (or maximize) the objective function with the fewest possible number of observations. In my case, I want to obtain the best parameter configuration for the CNN network, given by a vector x, that obtains the smallest error in objective $f(x)$.

3.3.1 The Necessity

In artificial intelligence scenarios, one can consider situations like synthesizing a molecule with specific characteristics, where each evaluation necessitates a costly real experiment in terms of time and money. In such instances, the objective is to acquire the optimal parameter configuration, represented by a vector x, that minimizes the error in $f(x)$ [116].

Therefore, the corresponding mathematical expression would be as follows:

$$x^* = \min f(x)$$

where x is the input value, which minimizes the underlying function $f(x)$, and X is the feature space where it is being optimized. If X is chosen wrong, $f(x)$ is not optimized correctly. To effect this minimization, a trade-off will be made between exploiting promising solutions and exploring unknown areas of the entry space.

Constraints:

- f is a black box for which no closed form is known;
  - gradients $\frac{df}{dx}$ are not available.
- f is expensive to evaluate;
- uncertainty on observations y_i of f

Goal:

find x^*, while minimizing the number of evaluations $f(x)$.

3.3.2 The Pseudocode

To strike a balance in the exploration-exploitation trade-off, the acquisition function utilizes the mean $\mu(x_{t+1})$ and the covariance $\sigma^2(x_{t+1})$. This enables the calculation of the expected utility associated with observing a specific point $x_t + 1$. Hence, Bayesian optimization is a technique that relies on predictions derived from the belief one holds about the model. This approach yields superior results compared to simple random

search or grid search strategies, as neither of these methods leverages the model to guide the minimization process [116].

The Bayesian optimization pseudocode for $t = 1 : T$ sequence of trials is shown as follows:

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
    - Integrate all possible true functions, using Gaussian Process regression and fit a model.
2. Re-compute minimizing the utility function u based on the posterior distribution for sampling the next point.
    - Exploit uncertainty to balance exploration against exploitation.

$$x_{t+1} = \arg\min_{x} u(x)$$

3. Sample the next observation using utility function y_{t+1} at x_{t+1}.

Ultimately, the concept behind Bayesian optimization is to employ the Gaussian Process (GP) model to compute the next best point for evaluation, determined by the acquisition function [93]. Consequently, the problem transforms into optimizing the acquisition function in each evaluation. The associated cost is considered negligible compared to evaluating the objective function, as the acquisition function lacks noise and is explicitly counted, rendering it easier to optimize.

3.3.3 Acquisition function

Acquisition or utility function u(x) specifies which sample x should be tried next using one of the concepts, including Probability of improvement, Expected improvement, and Lower Confidence Bound [116]. The mathematical concept of choice of acquisition functions provided in Appendix A, Chapter A.1.

Finally, the acquisition function that uses the lowest confidence limit (LCB) (or the highest, upper confidence limit (UCB) if it is being maximized) is optimistic about the variance $\sigma(x)$. The mathematical expression of this strategy is as follows:

$$LCB(x) = \mu(x) - \kappa\sigma(x) \tag{3.2}$$

where kappa, κ, is a constant value ≥ 0, as I would like to minimize the objective function.

Chapter 4

Methodology

Transfer learning involves leveraging the knowledge gained from a pre-trained model in a new classification task. Certain pre-trained models undergo training on millions of images over numerous epochs, achieving high accuracy in a broad task. In my experimentation, I specifically selected pre-trained architectures and fine-tuning, including DenseNet, ResNet, WideResNet, and SqueezNet which were initially trained on the ImageNet large-scale dataset [29, 95, 96]. The utilization of pre-trained models involves fine-tuning—training on target datasets for a limited number of epochs rather than undertaking an extensive retraining process. It is important to note that since ImageNet images and labels differ from my dataset, consideration should be given to pre-trained models on similar data types for optimal performance.

The proposed methodology consists of three primary components: Data and Image pre-processing, building CNN architectures, and the Bayesian optimizer. Within the Bayesian optimizer methodology, three fundamental steps are involved: hyperparameter selection, fitness function computation, and hyperparameter tuning. The CNN architecture plays a crucial role in transfer learning by performing deep low-level feature extraction followed by fine-tuning, and then a classifier is applied. Additionally,

the model employs data pre-processing techniques and enhancement of large-scale radiography X-ray images, resulted in making non-linearities more distinguishable, before feeding them into the model's training.

The training and validation sets play a crucial role during the training and tuning processes. Also, the testing data will be employed to evaluate the optimized model which has been trained as the best model. It is important to select test data that have the same distribution as the validation data. Following the fine-tuning of CNN hyperparameters using the Bayesian optimizer, the optimizer selects the optimal hyperparameters for utilization in the testing phase.

The Bayesian function determines whether the next set of hyperparameters is generated randomly or using the fitness model. The CNN architecture is updated to align with the acquired hyperparameters, followed by training and validation loss calculation. The Bayesian process is then updated to provide a more accurate estimation of the objective function. This process is repeated for a total of 13 iterations and 30 epochs per each iteration, and the model with the lowest loss is selected. This outlines my procedural steps for tuning hyperparameters using the Bayesian-optimized CNN model. The choice of 13 iterations and 30 epochs per iteration ensures a balance between computational efficiency and the thoroughness of the hyperparameter search process. Thirteen iterations allow for a sufficient number of updates to the Bayesian model, enabling it to refine its predictions and identify optimal hyperparameters effectively. Meanwhile, 30 epochs per iteration ensure that each set of hyperparameters is adequately evaluated, allowing the model to converge and providing a reliable measure of its performance.

As depicted in Figure 4.1, my architecture comprises two primary components:

Training and Test phases. Each of these components is further divided into several sub-components. The Training component encompasses a data separation structure, followed by data and image pre-processing and the sub-component of automatic hyperparameter optimization. Collectively, these elements constitute the training flow and the search for the best model, considering various configurations.

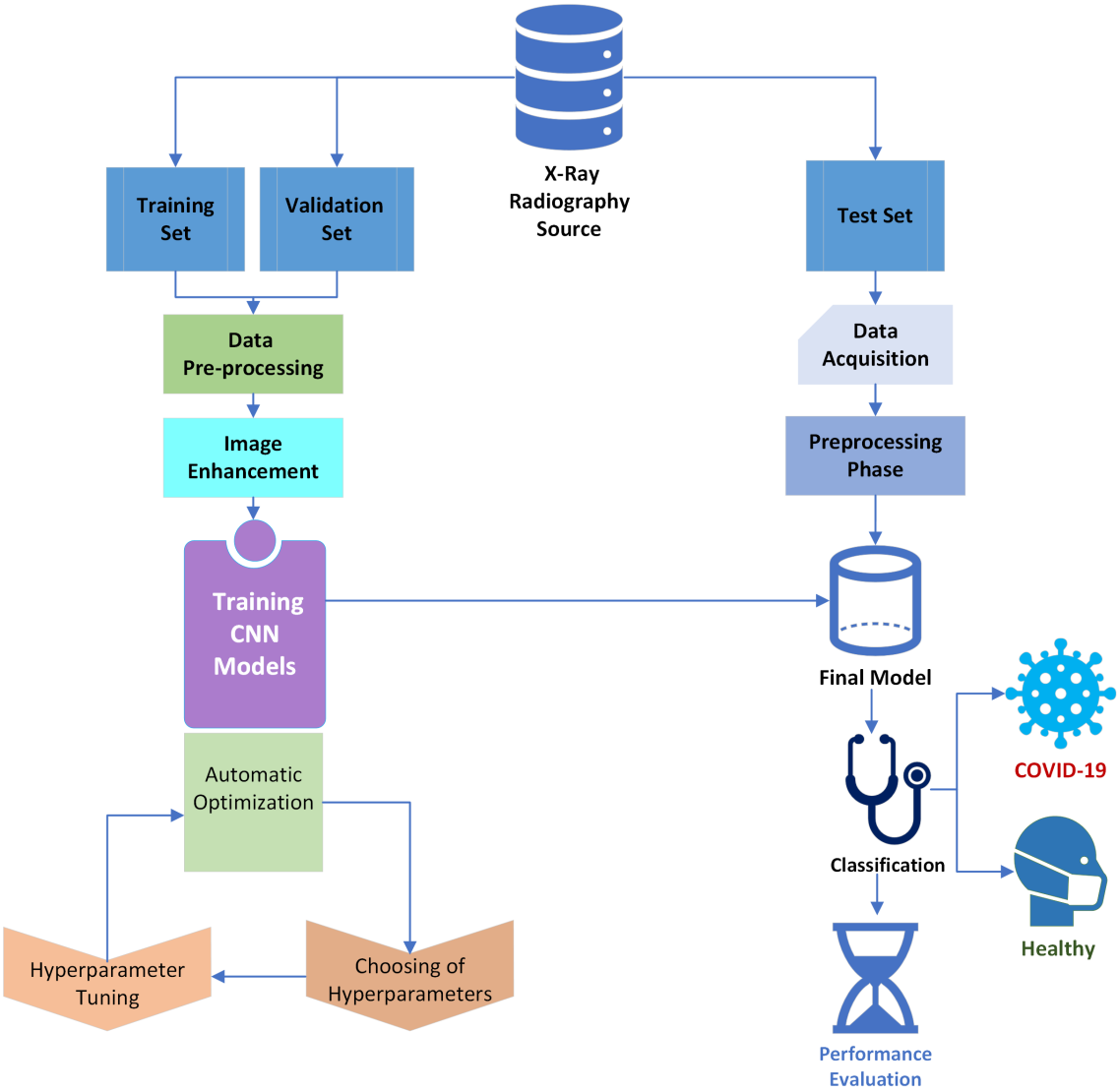

Figure 4.1: The Proposed Methodology

4.1 Training Components

4.1.1 Training Phase

The selected radiography X-ray dataset [24, 89] was split into 80% for training, 10% for validation, and 10% for test datasets. The allocation of validation data serves the purpose of preventing overfitting in the models. Also, the test data was employed to assess the model's performance in real-world scenarios, thereby evaluating its generalizability beyond the training and validation phases. Generally, regularization techniques such as L1 or L2 regularization, dropout, or batch normalization can also be applied to prevent overfitting of the model. Regularization can help to improve the generalization of the model and prevent it from memorizing the training data. When using regularization techniques, it is important to consider the specific requirements of the task and the architecture of the model. For example, if the model is deep, it may be necessary to use dropout to prevent overfitting, while if the model is shallow, L2 regularization may be more appropriate. In my work, I chose L2 Regularization and early-stopping strategy, if there is no decrease in validation, to avoid models overfitting.

4.2 Data Pre-processing

Data preprocessing is a crucial step in CNN development, involving the conversion of raw input data into a format compatible with the CNN architecture. This process includes tasks such as image resizing, pixel value normalization, and data augmentation techniques like flipping or rotation. Effective preprocessing can significantly enhance

model accuracy and generalization. It is essential to tailor preprocessing to the specific characteristics of the dataset and the requirements of the model. For instance, resizing images to a consistent size may be necessary for datasets with varying image dimensions. Similarly, normalizing pixel values to a smaller range might be required if the original values span a large range, ensuring optimal learning by the model. The preprocessing steps undertaken for the entire datasets in my work are as follows:

4.2.1 Data Resizing

The input data underwent preparation according to the pre-processing pattern employed in training the pre-trained models utilized in this thesis. Specifically, the images were resized and normalized to dimensions of $224 * 224$ pixels using the bilinear interpolation technique, aligning with the standard size commonly employed by popular convolutional neural network models.

4.2.2 Data Normalization

The input images were first loaded within a range of [0, 1], and subsequently normalized using the mean values of [0.485, 0.456, 0.406] and standard deviation values of [0.229, 0.224, 0.225], which are based on the statistics of the ImageNet dataset.

4.2.3 Data Center Cropping

For the validation image dataset, the process involved cropping at the center and resizing the images to dimensions of $224 * 224$.

4.3 Data Enhancement

Surprisingly, there is still room to incorporate more effective image preprocessing techniques to further increase the accuracy of these systems, like histogram equalization (HE) method [109], contrast limited adaptive histogram equalization (CLAHE) method [35], gamma correction (GC) method [90], and CLAHE, and bi-histogram equalization with adaptive sigmoid function (BEASF) methods [42].

The importance of using image enhancement methods based on histogram equalization relies on enhancing image features to make image non-linearities more distinguishable. Radiologists, for example, use manual contrast improvement to better diagnose masses and nodules. Chest X-ray images can exhibit variations in image contrast or brightness owing to differences in patient body size or X-ray dose. To mitigate the potential negative impact of such variations, a histogram equalization method is employed to normalize the image. This application of histogram equalization aids in contrast modification, enhancing the visualization of lung tissue patterns and characteristics associated with COVID-19 infection [45].

4.3.1 Contrast Limited Adaptive Histogram Equalization

Contrast Limited Adaptive Histogram Equalization (CLAHE) is an improved version of Adaptive Histogram Equalization (AHE) that overcomes the limitations of standard histogram equalization. CLAHE partitions an image into contextual regions and applies histogram equalization to each region, resulting in a more even distribution of grey values and making hidden features of the image more visible. This technique utilizes the full grey spectrum to express the image [35].

CLAHE differs from ordinary adaptive histogram equalization in its contrast-limiting feature. This feature can also be applied to global histogram equalization, giving rise to contrast limited histogram equalization (CLHE), although CLHE is rarely used in practice. In the case of CLAHE, the contrast limiting procedure is applied for each neighborhood from which a transformation function is derived.

CLAHE has been applied in medical imaging, computer vision, and other domains where precise visual perception is critical. CLAHE is known for its effectiveness in enhancing contrast and improving the discernibility of details within images, especially those with inadequate contrast or inconsistent illumination circumstances.

The CLAHE algorithm relies on two essential hyperparameters: the number of tiles and the clip limit. Improper hyperparameter selection may lead to a decrease in image quality. To address this, a learning-based hyperparameter selection method has been proposed to efficiently determine these hyperparameters for the CLAHE technique.

Contrast Limited Adaptive Histogram Equalization (CLAHE) has better revealed nodular-shaped opacity related to COVID-19 pneumonia. CLAHE is one of the most popular enhancement methods in different image types [87].

4.3.2 Bi-Histogram Equalization with Adaptive Sigmoid Function

Another histogram equalization-based algorithm is Bi-histogram Equalization with Adaptive Sigmoid Function. Bi-Histogram Equalization with Adaptive Sigmoid Function (BEASF) is a method used for image enhancement that overcomes the limitations of traditional histogram equalization techniques. The BEASF technique involves splitting the image histogram into two sub-histograms using the mean as a threshold

and replacing their cumulative distribution functions with two smooth sigmoids with their origins placed on the median of the sub-histograms. Subjective and objective assessments have shown the superiority of the BEASF method over existing image enhancement techniques [42].

The BEASF technique has been applied in various domains, including caries detection using multidimensional projection and neural networks, as well as in color image enhancement applications. The BEASF algorithm involves a multi-phase process, including the introduction of bi-histogram equalization with adaptive sigmoid functions to enhance image quality, followed by other enhancement models like grey thresholding and active contour. The features are then extracted using multilinear principal component analysis (MPCA) [17].

In my work, I enhanced the input data by concatenating the output results of CLAHE and BEASF methods with the original image. This arrangement was used to create an improved input image. Figure 4.2 shows the image pre-processing step to generate three images fed into RGB channels of the CNN models. Three images namely, the original chest X-ray image, the contrasted limited adaptive histogram equalization of the original image, and the bi-histogram equalization with adaptive sigmoid function of the original image using the specific hyperparameter, are fed into three input RGB channels of the CNN-based pre-trained models, respectively.

BEASF adaptively improves image contrast based on the global mean value of the pixels. It has a hyperparameter γ to define the sigmoid function slope. Although BEASF did not improve opacity detection in all images, it could complement the CLAHE method. I chose a BEASF-enhanced image method with a $\gamma = 1.5$ to be concatenated with CLAHE-enhanced and the main images, generating 3 images fed

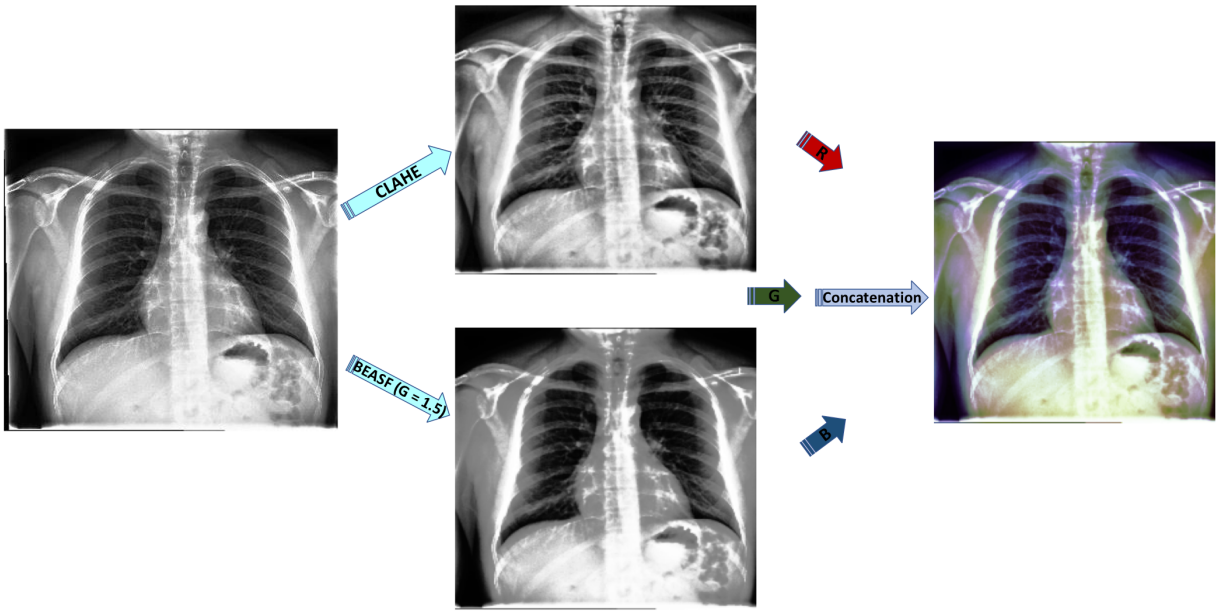

Figure 4.2: The process of the concatenation of three Original, CLAHE, and BEASF ($\gamma = 1.5$) Images into RGB channel

into three channels of the model as the final step of my data preparation. The figure 4.3 illustrate the comparison between the original image and its pre-processing results, including the application of CLAHE and BEASF with varying γ hyperparameters, ranging from 0.5 to 2 [42].

The figures 4.4, and 4.5 displays the sample COVID-19 and normal images, along with their corresponding RGB enhanced images. Moreover, other X-ray images from the proposed dataset with their corresponding RGB enhanced images provided in Appendix B.

Figure 4.6 shows examples of Normal and COVID-19 cases. To aid in better understanding the interpretation of these images, it is crucial to recognize the visual differences between COVID-19 and normal X-ray images. In COVID-19 X-ray images, specific patterns such as ground-glass opacities, consolidations, and bilateral

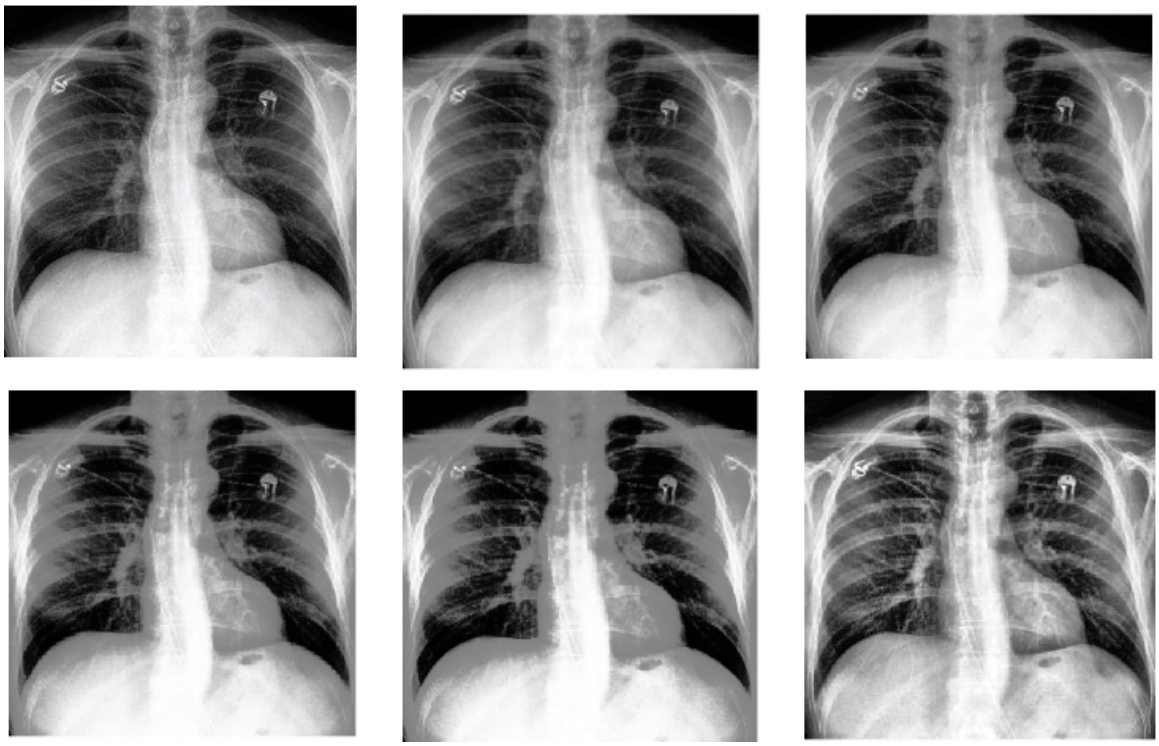

Figure 4.3: BEASF with different γ compared with Original image and CLAHE: (Original Image, BEASF ($\gamma = 0.5$), BEASF ($\gamma = 1.0$)), (BEASF ($\gamma = 1.5$), BEASF ($\gamma = 2.0$), CLAHE Image)

infiltrates are commonly observed in the lung regions. These patterns appear as hazy areas that reduce the clarity of the lung structures. In contrast, normal X-ray images typically show clear lung fields without such opacities or consolidations, allowing for a distinct and unobstructed view of the lung anatomy.

4.4 Pre-trained Models

Certain pre-trained models have been trained on vast image datasets for numerous epochs, achieving high accuracy on general tasks. In my experiments, I specifically chose the DenseNet, WideResNet, and SqueezNet CNN architectures that were previously pre-trained on the ImageNet large-scale dataset. These pre-trained models

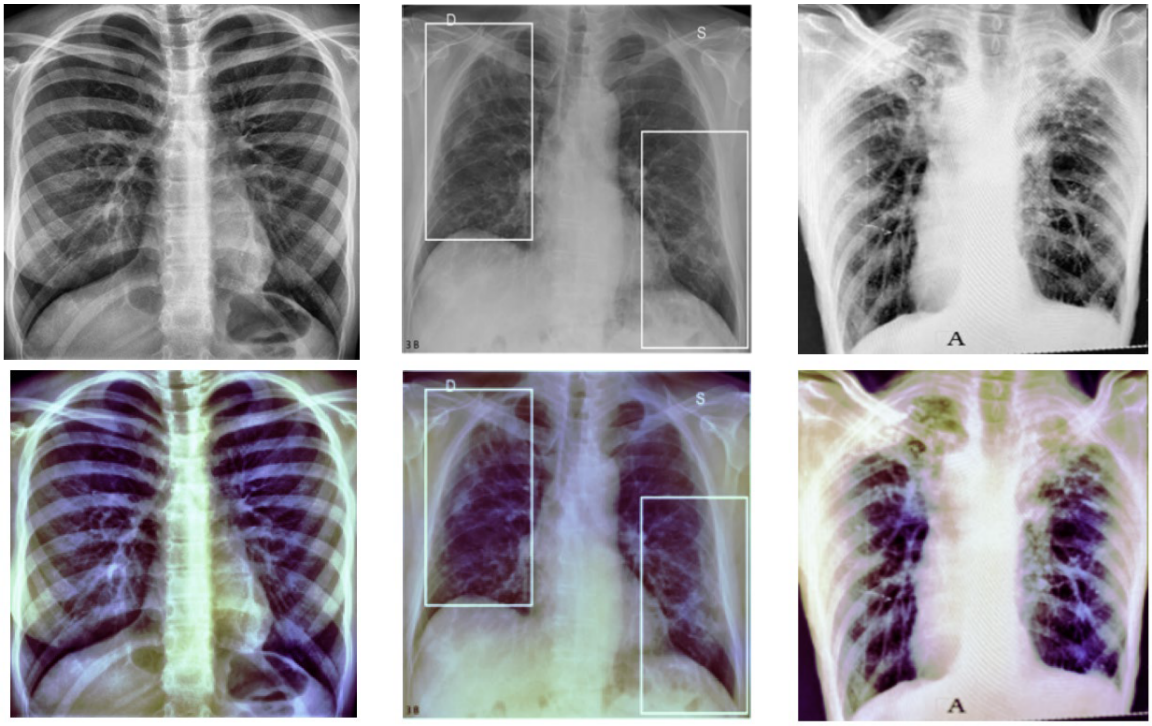

Figure 4.4: Sample COVID-19 and corresponding RGB Enhanced Images

were then fine-tuned by the last layer, meaning they will train on target datasets for a 30 number of epochs instead of undergoing full model training.

4.5 Automatic Hyperparameters Optimization

Bayesian optimization is a powerful method for efficiently tuning hyperparameters in machine learning models. It is particularly effective for optimizing hyperparameters in various classification and regression machine-learning algorithms. This method is versatile and efficient in time and memory capacity for tuning many hyperparameters, making it well-suited for the complex task of hyperparameter tuning in neural networks. Bayesian Optimization offers significant benefits in terms of efficiency and

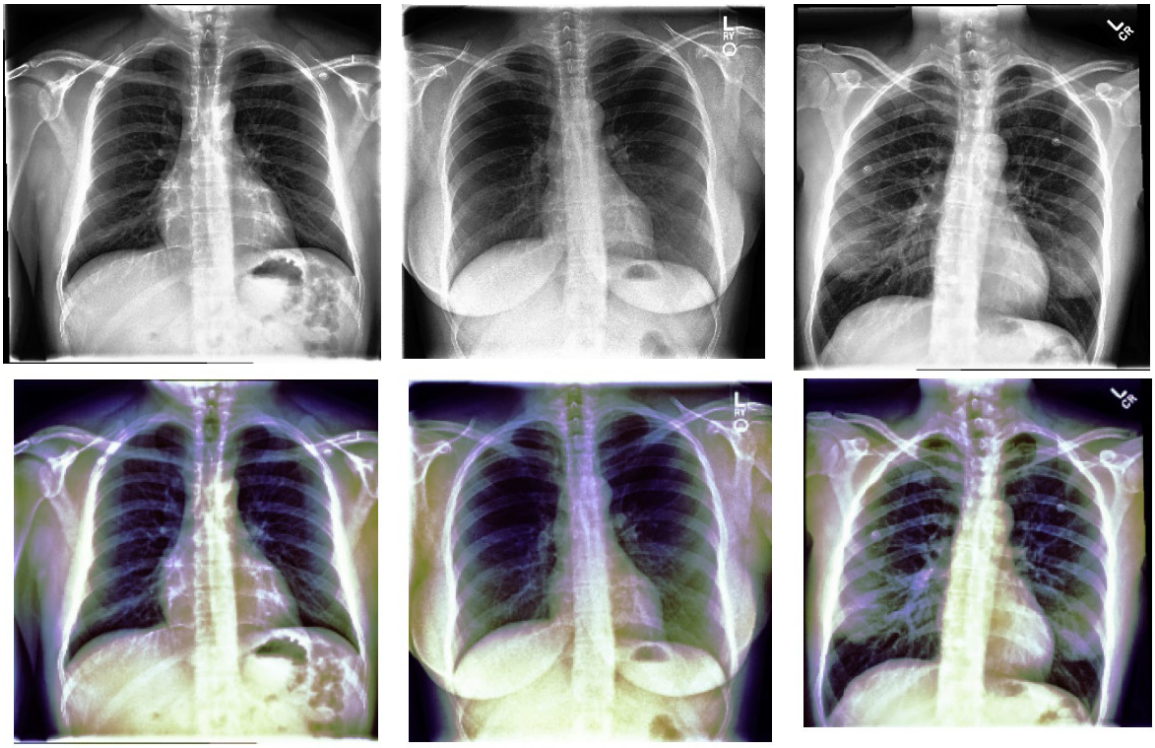

Figure 4.5: Sample NORMAL and corresponding RGB Enhanced Images

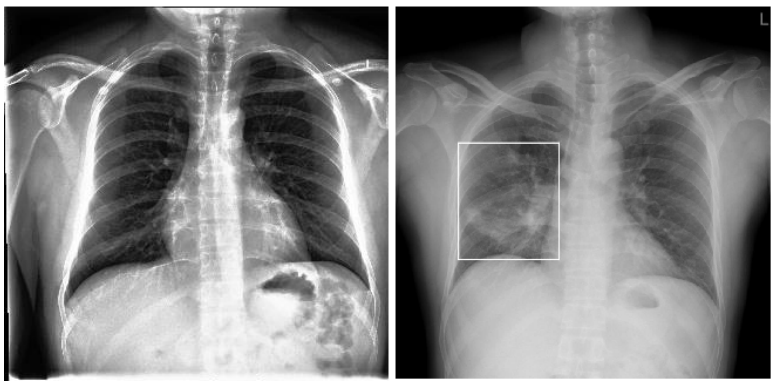

Figure 4.6: Sample NORMAL and COVID-19 X-ray Images:
(NORMAL, COVID-19)

effectiveness, especially when dealing with expensive function evaluations, like training deep learning models. While grid and random search optimization techniques are simpler and can be effective for smaller problems or when computational resources are abundant, Bayesian Optimization is generally more powerful for complex and large-scale hyperparameter tuning tasks.

The validation procedure was employed to determine the optimal set of hyperparameters for the selected CNN model. Initially, the the CNN pre-trained models having the trainable parameter values obtained from a large-scale dataset was loaded. Subsequently, the hyperparameters of the CNN model were adjusted to minimize the objective function based on the value of the loss function, which serves as the fitness function. A new set of hyperparameters was then obtained using the projected improvement acquisition function.

Bayesian Optimization is a probabilistically principled method for global optimization, particularly well-suited for tuning the hyperparameters of algorithms in machine learning. It provides a more efficient and targeted exploration of the hyperparameter space, ultimately leading to improved model performance.

Finding the best hyperparameter configuration for a given function shouldn't rely solely on intuition or individual experience. Instead, it's crucial to use methods that ensure optimal results. Various approaches exist for this purpose, including exhaustive searches like Grid Search and Random Search, as well as optimization paradigms such as Genetic Algorithms and Bayesian Optimization [21, 116].

In my work, I utilized a Python library for automating the configurable search domain for the optimal hyperparameter, empowering both CPU and GPU computations. It operates on the principles of Bayesian Optimization and is backed by the

SMBO (Sequential Model-Based Global Optimization) methodology [20].

4.5.1 Bayesian Search Space Domain

Selecting the right value of the hyperparameters, including the learning rate, batch size, and number of epochs, can significantly impact model performance. Techniques like grid search or random search can be employed to tune these hyperparameters. Strategies such as early stopping or learning rate decay are crucial to prevent overfitting. The dataset characteristics and model complexity should guide hyperparameter selection. For instance, a smaller learning rate may be needed for effective convergence with large datasets. Likewise, complex models might require smaller batch sizes to avoid overfitting during training.

Defining the search space in Bayesian Optimization is crucial for determining the justified configuration for each model, including the key trainable hyperparameters targeted for the optimization of the entire network.

My selected hyperparameters targeted for optimization encompass the following:

- Learning Rate:

  This hyperparameter sets the rate at which the model's weights are updated during training.

- Gradient Optimizers:

  These include SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, and Adamw, which are different optimization algorithms used to update the model's weights during model training.

- Momentum:

  This hyperparameter contributes to the optimization process by adding a fraction of the update vector from the previous iteration.

- L2 Regularization:

  It adds a penalty term to the loss function, preventing the weights from becoming too large.

- Batch Size:

  It determines the number of training examples to combine to find the gradients for a single step in gradient descent.

- Models:

  These models are often used as a starting point in finding the optimized model, leveraging the knowledge gained from training on a large dataset. My model includes the possibility of building and training any CNN models as one of the hyperparameters.

Bayesian Optimization is particularly useful for optimizing complex functions that are expensive to evaluate, making it well-suited for hyperparameter tuning in machine learning models. It works by building a probabilistic model of the function being optimized and using that model to make intelligent decisions about where to evaluate the function next.

In practical implementations, the search space is defined to include these hyperparameters and Bayesian Optimization algorithms are used to efficiently explore this space and find the best-performing hyperparameters for machine learning models. This approach is especially valuable when training a model is expensive and

time-consuming, as it allows for a more targeted and efficient exploration of the hyperparameter space, ultimately leading to improved model performance.

In this subcomponent, a search space was defined, encompassing values exempt from optimization, along with hyperparameters tuning for optimization. Following the execution of each iteration, statistical data was recorded, comprising the chosen values for each hyperparameter by the Bayesian optimizer, the loss of each model, and the accuracy during the model validation.

Table 4.1 indicates my search space domain, including the range, function, and data type of the hyperparameters for training the model.

Subsequently, the best configuration was determined based on the lowest loss achieved. Using the recovered hyperparameter configurations, the final model was trained and evaluated with previously unseen test data in the testing component.

Hyperparameter	**Range**	**Function**	**Data Type**
Learning Rate	[0.001, 1]	Logarithmic	Real number
Optimizers	SGD, RMSProp, AdaGrad, AdaDelta, Adam, AdaMax, AdamW	None	-
Momentum	[0.8, 1]	None	Real number
L2-Regularization	$[10^{-10}, 0.001]$	Logarithmic	Real number
Batch Size	[5, 100]	None	Real number
Models	WideResNet50, DenseNet-161, SqueezeNet	None	-

Table 4.1: The Model Hyperparameters Configuration

4.6 Model Evaluation

After determining the optimized hyperparameters for each pre-trained model, the trained model will undergo evaluation using the validation set to attain detection accuracy. The statistics of all settings, encompassing the optimized hyperparameter values, detection accuracy, training time, and loss function per iteration, will be exported. This facilitated the job of selecting the best model upon completion of all iterations.

4.7 Test Component

4.7.1 Test Phase

During the testing phase, all X-ray images from the test dataset underwent the same data preprocessing step for classification. The best model attained during the training phase was used in the inference component. As a result, each test image went through the same data preprocessing steps, ultimately leading to the model classifying the input image as either a COVID-19 case or a healthy case.

Chapter 5

Experiments

5.1 Experimental Setups

5.1.1 Dataset

Dataset Overview

COVID-19 patients are expected to undergo extensive data collection procedures, which involve not only the structure of samples within a dataset but also their distribution across different classes. This distribution significantly affects the performance of the resulting model. Factors such as color, geometry, and pattern play crucial roles in the effectiveness of intelligent computer-aided prototypes. Furthermore, to ensure a consistent and robust model, it is essential to have an equal number of samples representing all possible scenarios or occurrences for each class.

Dataset Source

COVID-19 data samples were gathered from various publicly accessible datasets, online sources, and published papers without applying any augmentation techniques to create more COVID-19 images, Table 5.1.

Data Source Name	Number of Images	Reference
PadChest	2473	[4]
Medical School in Germany	183	[3]
SIRM	559	[6]
GitHub Repository	400	[1]

Table 5.1: Summary of Data Sources for COVID-19 CXR Images

Additionally, the Normal data samples were collected from two different datasets, Table 5.2.

Data Source Name	Number of Images	Reference
RSNA	8851	[5]
Kaggle	1341	[2]

Table 5.2: Summary of Data Sources for Normal CXR Images

By integrating the COVID-19 radiography dataset [24, 89], we developed a new dataset. To achieve the highest degree of model efficiency, a balanced selection of two classes and an equal proportion of data from different sources are applied. The sample X-ray images of both classes from the proposed dataset provided in Appendix B.

All the data source images were in Portable Network Graphics (PNG) file format with a resolution of 299*299 pixels. Among these images, a total of 3,617 images from both classes were equally selected to be considered as the dataset, which was then split into 80% for the training set, 10% for the validation set, and 10% for the test set, as indicated in the Figure 5.1, which means exactly 2,893, 362, and 362 images respectively [72].

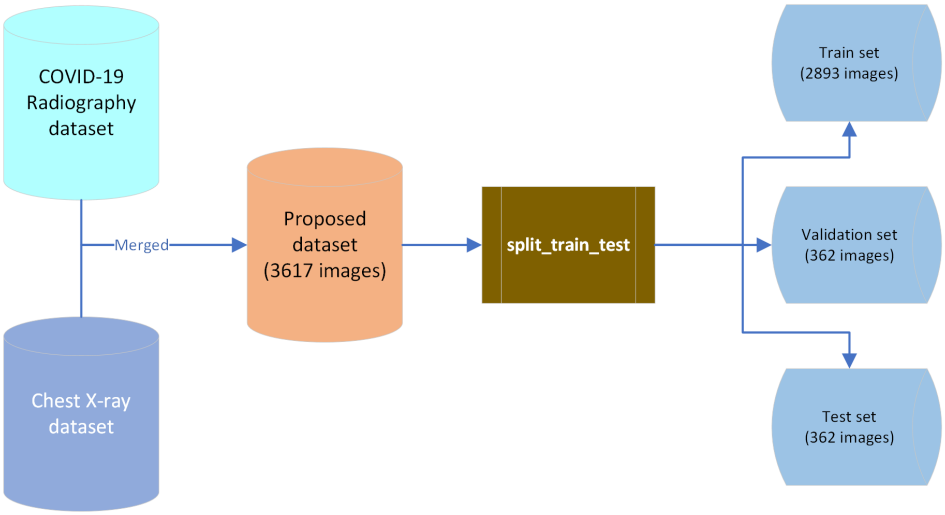

Figure 5.1: The Proposed X-ray Dataset

5.1.2 Bayesian Model-Based Optimization

Models Configuration

A search space defines the range of possible values for the hyperparameters that an optimization algorithm explores to find the best-performing model configuration. It is important because it determines the boundaries within which the optimal hyperparameters are sought, significantly influencing the efficiency and outcome of the optimization process.

Defining a search space involves structuring nested function expressions encompassing stochastic expressions, representing the hyperparameters. These stochastic expressions serve as the variables subject to optimization. Random search algorithms sample from this nested stochastic program, guiding the search for optimal hyperparameters. However, hyperparameter optimization algorithms go beyond standard sampling methods. Instead, they employ adaptive exploration strategies, which deviate from traditional sampling approaches and focus on optimizing the search process

without directly sampling from the specified distributions within the search space.

Optimization Algorithm

Hyperparameter optimization's primary objective is to identify the optimal value of a scalar-valued function, which may exhibit stochastic behavior, across a defined set of potential arguments. Unlike many optimization tools that assume inputs are drawn from a vector space, hyperparameter optimization emphasizes a detailed description of the search space. By providing comprehensive information about the function's domain and the regions likely to yield optimal values, hyperparameter optimization enables algorithms to conduct more effective searches. This approach enhances the efficiency of the optimization process by leveraging a deeper understanding of the function's behavior and potential optimal areas. The selected choice for a hyperparameter optimization algorithm is the Tree Parzen Estimator [20].

Definition of Objective Function

Hyperparameter optimization provides a few levels of increasing flexibility and complexity when it comes to specifying an objective function to minimize. I chose the loss function as an objective function, aiming to minimize it.

5.1.3 Hardware Implementation

Hardware considerations, such as the availability of Graphics Processing Unit (GPU) or Tensor Processing Unit (TPU), profoundly impact the performance and efficiency of CNN models. Given the computational demands, it is crucial to select hardware capable of handling the workload efficiently. Distributed training across multiple

machines can be employed to train large models in parallel, enhancing performance. Optimizing memory usage and batch sizes further boosts efficiency. Striking a balance between model complexity and available resources is essential. For instance, if memory is limited, smaller batch sizes may be necessary to ensure the model fits within memory constraints. Similarly, when processing power is constrained, simpler model architectures or longer training times may be required. While distributed training accelerates model training, it introduces complexity and necessitates specialized infrastructure and expertise. The exact hardware specification was the use of a T4 GPU hardware accelerator, 15.0 GB GPU RAM, and 12.7 GB system RAM.

5.2 Experimental Results

Evaluating the performance of a CNN model is a crucial step in its development process. This typically involves partitioning the dataset into training, validation, and testing sets, and then employing various metrics such as accuracy, precision, recall, and F1-score to assess how well the model performs. Additionally, visualization techniques like confusion matrices can provide further insights into the model's behavior.

When evaluating model performance, it is essential to tailor the assessment to the specific requirements of the task at hand. For instance, in medical diagnosis, maximizing the sensitivity of the model to detect true positives might be prioritized, even if it leads to a higher rate of false positives. Furthermore, it is imperative to consider potential biases and ethical implications associated with the model's use. CNNs find application across various domains, including facial recognition and surveillance, where biases and ethical concerns must be carefully addressed. For example, facial recognition systems have exhibited bias against certain demographic groups, highlighting

the need to identify and rectify such biases to ensure fairness and transparency in model deployment.

5.2.1 Evaluation Metrics

To evaluate the learning performance of the CNN model, a confusion matrix serves as a conventional evaluation technique. In the realm of image categorization, the confusion matrix allows the assessment of predicted results against actual values, thereby measuring the model's performance. By utilizing the confusion matrix, not only accurate and inaccurate predictions can be identified, but insights into specific types of errors made can also be gained. The calculation of the confusion matrix necessitates a set of test data and validation data containing the obtained results' values. This traditional approach facilitates the identification of six types of welding flaws, as depicted in Figure 5.2.

The results are categorized into four groups:

- True positive (TP): This category in the confusion matrix signifies the instances where the model accurately predicts the positive class or event out of all the true positive instances in the dataset.
- True negative (TN): In the confusion matrix, "true negative" denotes the instances where the model correctly predicts the negative class out of all the true negative instances in the dataset.
- False positive (FP): "False positive" represents the model's error when it inaccurately predicts the presence of a specific condition or event that is not actually present. This error type, also known as a type I error, can lead to incorrect decisions if not appropriately managed.

- False negative (FN): A "false negative" in the confusion matrix occurs when the model inaccurately predicts the absence of an event among all actual positive instances in the data. It measures the model's tendency to overlook positive cases or make type II errors.

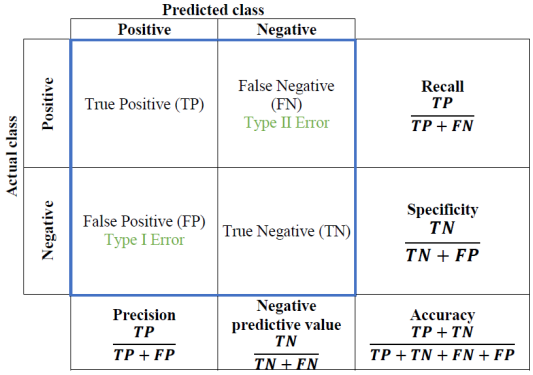

Figure 5.2: Confusion matrix

5.2.2 Results

I have used the Keras and PyTorch machine learning frameworks, along with the Python version 3.8 programming language for implementing the model. The average time for training the models was about 4 hours and 38.3 minutes using the NVIDIA Telsa T4 Cloud GPU, provided by Google Colab.

Table 5.3 shows the performance metrics obtained during the test phase and for the evaluation of the model with the unseen data.

Based on the confusion matrix and the corresponding performance metrics, the explanation of each metric is as follows:

1. Accuracy (ACC): Accuracy measures the proportion of true results among the

Evaluation Metric	**Value**
False Negatives (FN)	16
False Positives (FP)	20
True Negatives (TN)	161
True Positives (TP)	165
Accuracy (ACC)	**0.9011**
False Negative Rate (FNR) or Miss Rate	0.0889
False Positive Rate (FPR) or Fall-Out	0.1105
True Positive Rate (TPR) or Sensitivity (SEN) or Recall	0.9116
Specificity (SPC) or True Negative Rate (TNR)	0.8890
Precision (PRC) or Positive Predictive Value (PPV)	0.8919
The harmonic mean of precision and recall or F1-Score	0.9017

Table 5.3: Model performance metrics during the Test phase

total number of cases examined. The overall accuracy of the model is approximately 90.11%. This means that the model correctly classified about 90.11% of all cases.

2. False Negative Rate (FNR) or Miss Rate: The proportion of actual positive cases that got predicted as negative. The model incorrectly identified about 8.89% of actual positive cases as negative. This is a relatively low rate, indicating that the model is quite good at identifying positive cases.

3. False Positive Rate (FPR) or Fall-Out: The proportion of actual negative cases that got predicted as positive. The model incorrectly identified about 11.05% of actual negative cases as positive. This is also a relatively low rate, suggesting that the model does a decent job of distinguishing between positive and negative cases.

4. True Positive Rate (TPR) or Sensitivity (SEN) or Recall: The proportion of

actual positive cases that got predicted as positive. The model correctly identified about 91.16% of actual positive cases. This is a high rate, indicating that the model is very effective at identifying positive cases.

5. Specificity (SPC) or True Negative Rate (TNR): The proportion of actual negative cases that got predicted as negative. The model correctly identified about 88.90% of actual negative cases. This is also a high rate, showing that the model is effective at identifying negative cases.

6. Precision (PRC) or Positive Predictive Value (PPV): The proportion of positive identifications that were actually correct. The model correctly identified about 89.19% of cases it predicted as positive. This is a high precision, indicating that when the model predicts a case as positive, it is likely to be correct.

7. F1-Score: The harmonic mean of precision and recall. The F1-Score is approximately 0.9017, which is the harmonic mean of precision and recall. This score is very close to the accuracy, indicating that both precision and recall are high, and the model performs well across these metrics.

Table 5.4 shows the details of the hyperparameter tuning during the training phase consisting of 30 epochs and 13 iterations, indicating the values of each parameter defined in the search space. These results suggest that the Bayes optimizer may have identified Densenet as the best pre-trained model for this problem.

Additionally, table 5.5 presents the details of the hyperparameter tuning during the training phase consisting of 30 epochs and 13 iterations, by eliminating the data enrichment phase of the data transformation to compare with the situation including it. Figure 5.4 is the confusion matrix for the best model that is trained without

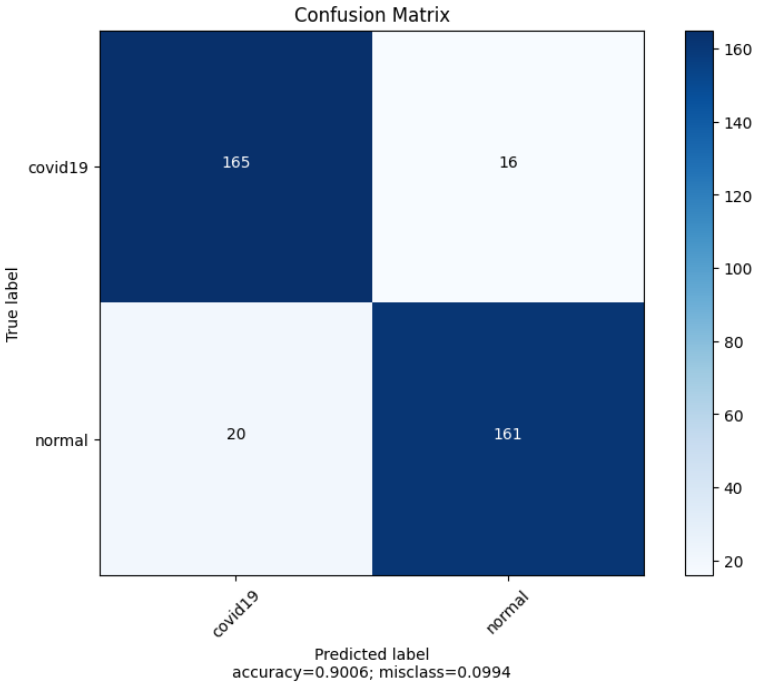

Figure 5.3: The confusion matrix of the test dataset

Loss function	Batch size	Learning rate	Momentum	Optimizer	Model Selection	L2-regularization	Accuracy%	Training Time
0.113	90.0	0.014	0.3	Adagrad	densenet-161	1.525e-05	88.7	16m 23s
0.138	60.0	0.018	0.4	Adam	squeezenet	8.179e-05	86.2	11m 40s
0.108	25.0	0.012	0.6	Adam	squeezenet	0.000178	89.2	10m 29s
0.135	25.0	0.013	0.7	SGD	densenet-161	7.438e-07	86.5	11m 30s
0.110	70.0	0.011	1.0	RMSprop	densenet-161	0.000921	89.0	11m 34s
0.086	60.0	0.012	0.4	RMSprop	densenet-161	1.066e-06	**91.4**	11m 34s
0.395	15.0	0.020	0.6	Adadelta	wideresnet50	1.154e-06	60.5	12m 42s
0.180	70.0	0.015	0.8	Adamax	wideresnet50	0.000271	82.0	12m 47s
0.119	15.0	0.016	0.7	RMSprop	densenet-161	2.798e-07	88.1	12m 21s
0.138	5.0	0.013	0.2	Adamax	wideresnet50	1.621e-06	86.2	15m 7s
0.403	50.0	0.011	0.0	Adam	wideresnet50	2.249e-06	59.7	12m 21s
0.458	10.0	0.090	0.9	RMSprop	squeezenet	6.053e-10	54.2	11m 10s
0.403	15.0	0.014	0.2	Adadelta	densenet-161	6.707e-05	59.7	11m 33s

Table 5.4: Optimized hyperparameters obtained during the model training

including the enhancement techniques. It is evident that applying image enhancement techniques can be beneficial.

The figure 5.3 highlights an accuracy of 90% during the evaluation of the test dataset, by utilizing the best-selected model obtained in the training phase, indicating the model's generalization as it is close to the best model accuracy during model evaluation, investigated in Table 5.4.

Loss function	Batch size	Learning rate	Momentum	Optimizer	Model Selection	L2-regularization	Accuracy%
0.146	90.0	0.014	0.3	Adagrad	densenet-161	1.525e-05	85.4
0.141	60.0	0.018	0.4	Adam	densenet-161	8.179e-05	85.9
0.130	25.0	0.012	0.6	Adam	squeezenet	0.000178	87.0
0.169	25.0	0.013	0.7	SGD	squeezenet	7.438e-07	83.1
0.130	70.0	0.011	1.0	RMSprop	densenet-161	0.000921	87.0
0.127	60.0	0.012	0.4	RMSprop	densenet-161	1.066e-06	**87.3**
0.384	15.0	0.020	0.6	Adadelta	wideresnet50	1.154e-06	61.6
0.188	70.0	0.015	0.8	Adamax	wideresnet50	0.000271	81.2
0.141	15.0	0.016	0.7	RMSprop	densenet-161	2.798e-07	85.9
0.138	5.0	0.013	0.2	Adamax	wideresnet50	1.621e-06	86.2
0.409	15.0	0.014	0.2	Adadelta	densenet-161	6.707e-05	59.1
0.519	60.0	0.810	0.9	AdamW	squeezenet	0.0004745	48.1
0.130	50.0	0.012	0.0	Adam	wideresnet50	2.249e-06	87.0

Table 5.5: Optimized hyperparameters obtained during the model training by excluding the image enhancement component

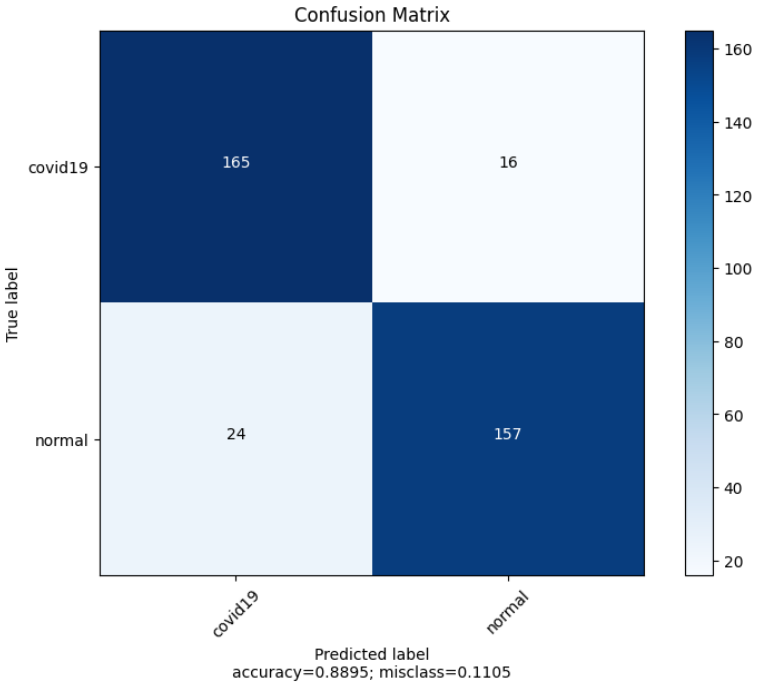

Figure 5.4: The confusion matrix of the test dataset without considering the image enhancement component

5.2.3 Discussion

The high accuracy achieved by the model can be attributed to its training with a substantial amount of data, enabling it to learn and generalize the distinctive features associated with a COVID-19 diagnosis in images. The test results suggest that the Bayesian optimizer may have determined or modeled that Densenet is the most suitable pre-trained model for this problem. This inference is supported by the consistent performance of Densenet across various experiments and the optimized hyperparameters obtained through Bayesian optimization. Therefore, it is likely that the optimizer identified Densenet as the optimal choice based on its ability to generalize well to medical image classification tasks, particularly for detecting COVID-19 cases from chest X-ray images.

DenseNet's pre-trained model achieved better results than SqueezeNet and WideResNet models in the study due to several key factors:

- Dense Connectivity: DenseNet's architecture allows each layer to connect to every other layer in a feed-forward manner, enhancing feature propagation and reuse, which is beneficial for complex tasks like medical image classification.

- Efficient Parameter Usage: DenseNet is more parameter-efficient than ResNet. It achieves high performance with fewer parameters, reducing the risk of overfitting and improving generalization.

- Gradient Flow: The dense connections in DenseNet alleviate the vanishing gradient problem, ensuring that gradients can flow more easily through the network during training.

- Pre-trained Feature Transfer: DenseNet, pre-trained on large datasets like ImageNet, benefits from a rich feature set that can be effectively transferred to the task of COVID-19 detection. This pre-training helps DenseNet to start with a strong base of learned features, which is crucial for transfer learning.

- Balanced Architecture: DenseNet balances depth and computational efficiency well. While ResNet is also deep and powerful, DenseNet's dense connections provide an edge in terms of feature reuse and efficiency.

Notably, while SqueezeNet is compact and efficient, it lacks the depth and dense connectivity of DenseNet, which may limit its performance on complex tasks. Meanwhile, WideResNet's residual connections help with gradient flow, but DenseNet's dense connections offer even more comprehensive feature reuse and gradient propagation.

Chapter 6

Conclusions

6.1 Summary

This study encompassed a thorough analysis and the development of a workflow for the automatic fine-tuning of trainable hyperparameters in each model to classify COVID-19 images using the PyTorch library. Multiple iterations were carried out to optimize hyperparameters for each model, taking into account various data pre-processing techniques, including data transformation and enrichment.

Pre-trained models were also incorporated as hyperparameters to identify the most suitable models for medical image classification. The findings revealed that the performance of pre-trained models varied based on the data enrichment techniques used. Notably, the Densenet-161 model consistently delivered accurate and robust results in both validation and test datasets, demonstrating the model's generalization capability.

The proposed methodology offers several advantages, including the application of feature and model selection methods and modeling and determining the optimal parameters of the entire network with metaheuristic algorithms.

As a result of the research, the study successfully identified the best hyperparameters for the top-performing models, which achieved an accuracy of 91% for the best model during the test phase. These optimized hyperparameters significantly enhanced the inferencing model, enabling the rapid and sensitive detection of COVID-19 cases using chest X-ray images. The study concludes that Bayesian optimization proves to be an effective strategy for enhancing transfer learning and fine-tuning applications, particularly in the field of medical image diagnostic tasks.

6.2 Future Work

Factors such as choosing the wrong algorithm, improper parameter configuration, limited availability of training data, selecting inappropriate features, and inadequate model generalization contribute to the presence of uncertainty in the algorithm's prediction and analysis.

In future research, the focus should be on exploring the effects of various data preprocessing and enrichment, such as data transformation and enhancement, to achieve better model performance. Additionally, there should be a deeper exploration of the potential of diverse machine learning classifiers.

Moreover, other intriguing focus areas could involve implementing semantic segmentation by integrating an attention mechanism into the U-Net model. This integration would allow the capture of long-range dependencies between input data, enabling the model to focus on important parts of the input and understand the relationships between pixels. The creation of regions of interest and lung masks, involvement in transformers, and consideration of image content criteria could offer significant benefits. Specifically, these techniques could assist in extracting particular features from

the data, such as lung pixels from body and background images, edges, and other visual content within medical images.

Bibliography

[1] GitHub repository. https://github.com/armiro/COVID-CXNet/.

[2] Kaggle dataset. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[3] medical school in Germany. https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png.

[4] PadChest dataset. https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/#1590858128006-9e640421-6711.

[5] RSNA dataset. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data.

[6] SIRM platforms. https://sirm.org/category/senza-categoria/covid-19/.

[7] A. Abbasian Ardakani, U.R. Acharya, S. Habibollahi, and A. Mohammadi. Covidiag: A clinical cad system to diagnose covid-19 pneumonia based on ct findings. *European Radiology*, 31(1):121–130, 2021.

[8] E. Acar, B. Oztoprak, M. Reşorlu, M. Das, I. Yılmaz, and I. Oztoprak. Efficiency of artificial intelligence in detecting covid-19 pneumonia and other pneumonia causes by quantum fourier transform method. *medRxiv*, 2021.

[9] H.M. Afify, A. Darwish, K.K. Mohammed, and A.E. Hassanien. An automated cad system of ct chest images for covid-19 based on genetic algorithm and k-nearest neighbor classifier. *Ingenierie des Systemes d Inf.*, 25(5):589–594, 2020.

[10] N. A. Alam, M. Ahsan, M. A. Based, and J. Haider. Intelligent system for vehicles number plate detection and recognition using convolutional neural networks. *Technologies*, 9:9, 2021.

[11] S. Albawi, T.A. Mohammed, and S. Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. IEEE, 2017.

[12] Hussam Alghamdi, Ghaida Amoudi, Sami Elhag, Kamran Saeedi, and Jamal Nasser. Deep learning approaches for detecting covid-19 from chest x-ray images: A survey. *IEEE Access*, 2021.

[13] A.M. Alqudah, S. Qazan, H. Alquran, I.A. Qasmieh, and A. Alqudah. Covid-19 detection from x-ray images using different artificial intelligence hybrid models. *Jordan Journal of Electrical Engineering*, 6(2):168–178, 2020.

[14] A. Amyar, R. Modzelewski, H. Li, and S. Ruan. Multi-task deep learning based ct imaging analysis for covid-19 pneumonia: Classification and segmentation. *Computers in Biology and Medicine*, 126:104037, 2020.

[15] I.D. Apostolopoulos and T.A. Mpesiana. Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Physics and Engineering in Medicine*, 43(2):635–640, 2020.

[16] A.A. Ardakani, A.R. Kanafi, U.R. Acharya, N. Khadem, and A. Mohammadi. Application of deep learning technique to manage covid-19 in routine clinical practice using ct images: Results of 10 convolutional neural networks. *Comput. Biol. Med.*, 121:103795, 2020.

[17] Eduardo F. Arriaga-Garcia, Raul E. Sanchez-Yanez, and Miguel Garcia-Hernandez. Image enhancement using bihistogram equalization with adaptive sigmoid functions. In *2014 International Conference on Electronics, Communications, and Computers (CONIELECOMP)*, pages 28–34. IEEE, 2014.

[18] O. Attallah, D.A. Ragab, and M. Sharkas. Multi-deep: A novel cad system for coronavirus (covid-19) diagnosis from ct images using multiple convolutional neural networks. *PeerJ*, 8:10086, 2020.

[19] H. Bai, R. Wang, Z. Xiong, B. Hsieh, K. Chang, K. Halsey, T. Tran, J. Choi, D. Wang, L. Shi, and et al. Erratum: Artificial intelligence augmentation of radiologist performance in distinguishing covid-19 from pneumonia of other origin at chest ct (radiology (2020) 296 3 (e156–e165)). *Radiology*, pages 225–225, 2021.

[20] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, volume 24. 2011.

[21] James Bergstra, Daniel Yamins, and David D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, pages I–115–I–123, June 2013.

[22] J. Chen, L. Wu, J. Zhang, L. Zhang, D. Gong, Y. Zhao, Q. Chen, S. Huang, M. Yang, and X. Yang. Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography. *Sci. Rep.*, 10(1):1–11, 2020.

[23] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.*, 54:6232–6251, 2016.

[24] MEH Chowdhury, T Rahman, A Khandakar, R Mazhar, MA Kadir, ZB Mahbub, KR Islam, MS Khan, A Iqbal, N Al-Emadi, MBI Reaz, and MT Islam. Can ai help in screening viral and covid-19 pneumonia? *IEEE Access*, 8:132665–132676, 2020.

[25] J. Cunha-Vaz. The blood-retinal barrier in the management of retinal disease: Euretina award lecture. *Ophthalmologica*, 237(1):1–10, 2017.

[26] Gillian Currie. Intelligent imaging: radiomics and artificial neural networks in heart failure. *J Med Imaging Radiat Sci*, 50(4):571–574, 2019.

[27] Gillian Currie, Joanne Hewis, and Stewart Bushong. Tomographic reconstruction; a non-mathematical overview. *J Med Imaging Radiat Sci*, 46(4):403–412, 2015.

[28] D. Dang, S.V.R. Chittamuru, S. Pasricha, R. Mahapatra, and D. Sahoo. Bplight-cnn: A photonics-based backpropagation accelerator for deep learning. *ACM J. Emerg. Technol. Comput. Syst. (JETC)*, 17:1–26, 2021.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[30] C.K. Dewa and Afiahayati. Suitable cnn weight initialization and activation function for javanese vowels classification. *Procedia Comput. Sci.*, 144:124–132, 2018.

[31] P. Dileep, D. Das, and P. K. Bora. Dense layer dropout based cnn architecture for automatic modulation classification. In *2020 National Conference on Communications (NCC)*, pages 1–5, 2020.

[32] C. Do and L. Vu. An approach for recognizing covid-19 cases using convolutional neural networks applied to ct scan images. In *Applications of Digital Image Processing XLIII*, volume 11510, page 1151034. International Society for Optics and Photonics, 2020.

[33] E. M. Dogo, O. Afolabi, N. Nwulu, B. Twala, and C. Aigbavboa. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In *Proceedings of the 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pages 92–99, 2018.

[34] Y. Duan, J.S. Edwards, and Y.K. Dwivedi. Artificial intelligence for decision making in the era of big data-evolution, challenges and research agenda. *International Journal of Information Management*, 48:63–71, 2019.

[35] Khalid El Asnaoui and Youssef Chawki. Using x-ray images and deep learning for automated detection of coronavirus disease. *Journal of Biomolecular Structure and Dynamics*, 2020:1–12, 2020.

[36] Wolfgang Ertel. *Introduction to Artificial Intelligence*. Springer, Berlin/Heidelberg, Germany, 2018.

[37] Daniel Fernandez Sánchez. Creating a bayesian optimization tool in python, 2019.

[38] A. Fesseha, S. Xiong, E.D. Emiru, M. Diallo, and A. Dahou. Text classification based on convolutional neural networks and word embedding for low-resource languages: Tigrinya. *Information*, 12:52, 2021.

[39] N. I. Galanis, P. Vafiadis, K. G. Mirzaev, and G. A. Papakostas. Convolutional neural networks: A roundup and benchmark of their pooling layer variants. *Algorithms*, 15:391, 2022.

[40] O. Gozes, M. Frid-Adar, H. Greenspan, P.D. Browning, H. Zhang, W. Ji, A. Bernheim, and E. Siegel. Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection and patient monitoring using deep learning ct image analysis. *arXiv preprint arXiv:2003.05037*, 2020.

[41] T. Guo, J. Dong, H. Li, and Y. Gao. Simple convolutional neural network on image classification. In *Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pages 721–724, 2017.

[42] A. Haghanifar, M.M. Majdabadi, Y. Choi, S. Deivalakshmi, and S. Ko. Covid-cxnet: Detecting covid-19 in frontal chest x-ray images using deep learning. *Multimedia Tools and Applications*, pages 1–31, 2022.

[43] W. Hao, W. Yizhou, L. Yaqin, and S. Zhili. The role of activation function in cnn. In *Proceedings of the 2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pages 429–432, Guangzhou, China, Dec 2020.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[45] M. Heidari, S. Mirniaharikandehei, A. Z. Khuzani, G. Danala, Y. Qiu, and B. Zheng. Improving performance of cnn to predict likelihood of covid-19 using chest x-ray images with preprocessing algorithms. 2020.

[46] E.E.-D. Hemdan, M.A. Shouman, and M.E. Karar. Covidx-net: A framework of deep learning classifiers to diagnose covid-19 in x-ray images. *arXiv preprint arXiv:2003.11055*, 2020.

[47] M. J. Horry, S. Chakraborty, M. Paul, A. Ulhaq, B. Pradhan, M. Saha, and N. Shukla. X-ray image based covid-19 detection using pre-trained deep learning models. *Engineering Archive*, 2020.

[48] M.B. Hossain, S.H.S. Iqbal, M.M. Islam, M.N. Akhtar, and I.H. Sarker. Transfer learning with fine-tuned deep cnn resnet50 model for classifying covid-19 from chest x-ray images. *Information Medicine Unlocked*, 30:100916, 2022.

[49] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[50] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

[51] J.T. Huang, J. Li, and Y. Gong. An analysis of convolutional neural networks for speech recognition. In *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4989–4993, 2015.

[52] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[53] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *Advances in Neural Information Processing Systems*, pages 1945–1953, 2017.

[54] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[55] Philip C. Jackson. *Introduction to Artificial Intelligence*. Courier Dover Publications, Mineola, NY, USA, 2019.

[56] P. Jiang, Y. Xue, and F. Neri. Convolutional neural network pruning based on multi-objective feature map selection for image classification. *Appl. Soft Comput.*, 139:110229, 2023.

[57] S. Jin, B. Wang, H. Xu, C. Luo, L. Wei, W. Zhao, X. Hou, W. Ma, Z. Xu, and Z. Zheng. Ai-assisted ct imaging analysis for covid-19 screening: Building and deploying a medical ai system in four weeks. *MedRxiv*, 2020.

[58] Mohammed Kayed, Ahmed Anter, and Hossam Mohamed. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture. In *Proceedings of the 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 238–243, Aswan, Egypt, February 2020.

[59] Rana Khattab, Islam R. Abdelmaksoud, and Samir Abdelrazek. Deep convolutional neural networks for detecting covid-19 using medical images: A survey. *New Generation Computing*, 41:343–400, 2023.

[60] G. Kiran Kumar, E. Parimalasundar, D. Elangovan, P. Sanjeevikumar, F. Lannuzzo, and J.B. Holm-Nielsen. Fault investigation in cascaded h-bridge multilevel inverter through fast fourier transform and artificial neural network approach. *Energies*, 13:1299, 2020.

[61] S. Kogilavani, J. Prabhu, R. Sandhiya, M. S. Kumar, U. Subramaniam, A. Karthick, M. Muhibbullah, and S. B. S. Imam. Covid-19 detection based on

lung ct scan using deep learning techniques. *Computational and Mathematical Methods in Medicine*, 2022.

[62] M. Krichen, A. Mihoub, M.Y. Alzahrani, W.Y.H. Adoni, and T. Nahhal. Are formal methods applicable to machine learning and artificial intelligence? In *Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 48–53, Riyadh, Saudi Arabia, 2022.

[63] Moez Krichen. Convolutional neural networks: A survey. *Computers*, 12(8), 2023.

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25*, 2012.

[65] Curtis Langlotz, Bruce Allen, and Bradley Erickson. A roadmap for foundational research on artificial intelligence in medical imaging: from the 2018 nih/rsna/acr/the academy workshop. *Radiology*, page 190613, 2019.

[66] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[67] Mengxue Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 4313–4324, Naha, Okinawa, Japan, April 16-18 2020. PMLR.

[68] Clarissa Liew. The future of radiology augmented with artificial intelligence: a strategy for success. *Eur J Radiol*, 102:152–156, 2018.

[69] G. Litjens, T. Kooi, B.E. Bejnordi, A.A.A. Setio, F. Ciompi, M. Ghafoorian, J.A. Van Der Laak, B. Van Ginneken, and C.I. Sánchez. A survey on deep learning in medical image analysis. *Med. Image Anal.*, 42:60–88, 2017.

[70] H. Liu, C. Zhang, Y. Deng, B. Xie, T. Liu, Z. Zhang, and Y.F. Li. Transifc: Invariant cues-aware feature concentration learning for efficient fine-grained bird image classification. *IEEE Trans. Multimed.*, pages 1–14, 2023.

[71] Xun Liu, Zhongwei Jia, Xiaochuan Hou, Mingyuan Fu, Lin Ma, and Qian Sun. Real-time marine animal images classification by embedded system based on mobilenet and transfer learning. In *OCEANS 2019-Marseille*, pages 1–5. IEEE, 2019.

[72] Mostafa Loey, Shaker El-Sappagh, and Seyedali Mirjalili. Bayesian-based optimized deep learning model to detect covid-19 patients using chest x-ray image data. *Computers in Biology and Medicine*, 142:105213, Mar 2022.

[73] Y. Ma, Y. Cao, S. Vrudhula, and J.S. Seo. Optimizing the convolution operation to accelerate deep neural networks on fpga. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 26:1354–1367, 2018.

[74] Morgan P McBee, Omer A Awan, and Andrew T. Colucci. Deep learning in radiology. *Acad Radiol*, 25(11):1472–1480, 2018.

[75] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G.J. Soufi. Deepcovid: Predicting covid-19 from chest x-ray images using deep transfer learning. *Medical Image Analysis*, 65:101794, 2020.

[76] Miguel Miranda, Kid Valeriano, and Jos´e Sulla-Torres. A detailed study on the choice of hyperparameters for transfer learning in covid-19 image datasets using bayesian optimization. *International Journal of Advanced Computer Science and Applications*, 12(4), 2021.

[77] Vincent C Muller and Nick Bostrom. Future progress in artificial intelligence: A survey of expert opinion. *Fundamental Issues of Artificial Intelligence*, 376:555–572, 2016.

[78] A. Narin, C. Kaya, and Z. Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, 24(3):1207–1220, 2021.

[79] Philomene S. B. Ndong, Wazir Y. H. Adoni, Tahani Nahhal, Calisto Kimpolo, Mohamed Krichen, A. El Byed, Ibtissam Assayad, and Francky K. Mutombo. A face-mask detection system based on deep learning convolutional neural networks. In *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*, pages 273–283, Berlin/Heidelberg, Germany, 2021. Springer.

[80] Caglar Oguz and Murat Yaganoglu. Detection of covid-19 using deep learning techniques and classification methods. *Information Processing and Management*, 59(5):103025, 2022.

[81] W. Ouyang, B. Xu, J. Hou, and X. Yuan. Fabric defect detection using activation layer embedded convolutional neural network. *IEEE Access*, 7:70130–70140, 2019.

[82] T. Ozturk, M. Talo, E.A. Yildirim, U.B. Baloglu, O. Yildirim, and U.R. Acharya. Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in Biology and Medicine*, 121:103792, 2020.

[83] J. A. Pandian, K. Kanchanadevi, V. D. Kumar, E. Jasi´nska, R. Go˘no, Z. Leonowicz, and M. Jasi´nski. A five convolutional layer deep convolutional neural network for plant leaf disease detection. *Electronics*, 11:1266, 2022.

[84] M.K. Pandit, S.A. Banday, R. Naaz, and M.A. Chishti. Automatic detection of covid-19 from chest radiographs using deep learning. *Radiography*, 27(2):483–489, 2021.

[85] Angkoon Phinyomark and Erik Scheme. Emg pattern recognition in the era of big data and deep learning. *Big Data and Cognitive Computing*, 2:21, 2018.

[86] Luís Pinto-Coelho. How artificial intelligence is shaping medical imaging technology: A survey of innovations and applications. *Bioengineering*, 10(12), 2023.

[87] Stephen M. Pizer, Ellis P. Amburn, Jon D. Austin, Robert Cromartie, Arnold Geselowitz, Thomas Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

[88] Ozkan Polat. Detection of covid-19 from chest ct images using exception architecture: A deep transfer learning-based approach. *Sakarya Univ. J. Sci.*, 25(3):813–823, 2021.

[89] T Rahman, A Khandakar, Y Qiblawey, A Tahir, S Kiranyaz, SBA Kashem, MT Islam, SA Maadeed, SM Zughaier, MS Khan, and ME Chowdhury. Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images. *arXiv preprint arXiv:2012.02238*, 2020.

[90] Tanzila Rahman, Abu Khandakar, Yousuf Qiblawey, Ammar Tahir, Serkan Kiranyaz, Sifat Bin Abul Kashem, Mohammad Tariqul Islam, Somaya Al Maadeed, Susu M Zughaier, Muhammad Shoyaib Khan, and Mohammad E Chowdhury. Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images. *Computers in Biology and Medicine*, 132:104319, 2021.

[91] V. Raj, R. Kumar, and N.S. Kumar. An scrupulous framework to forecast the weather using cnn with back propagation method. In *Proceedings of the 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 177–181, 2022.

[92] S. Rajaraman, J. Siegelman, P.O. Alderson, L.S. Folio, L.R. Folio, and S. Antani. Iteratively pruned deep learning ensembles for covid-19 detection in chest x-rays. *IEEE Access*, 8:115041–115050, 2020.

[93] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[94] Sourav S. Roy, Vasundhara Goti, Anshul Sood, Himanshu Roy, Traian Gavrila, Dorin Floroian, Nicolae Paraschiv, and Behnam Mohammadi-Ivatloo. L2 regularized deep convolutional neural networks for fire detection. *Journal of Intelligent & Fuzzy Systems*, 43:1799–1810, 2022.

[95] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[96] Olga Russakovsky and Li Fei-Fei. Attribute learning in large-scale datasets. In *European Conference of Computer Vision (ECCV), International Workshop on Parts and Attributes*, Crete, Greece, September 2010.

[97] A. Saeedi, M. Saeedi, and A. Maghsoudi. A novel and reliable deep learning web-based tool to detect covid-19 infection from chest ct-scan. *arXiv preprint arXiv:2006.14419*, 2020.

[98] K. Sahinbas and F.O. Catak. Transfer learning-based convolutional neural network for covid-19 detection with x-ray images. In *Data Science for COVID-19*, pages 451–466. Elsevier, 2021.

[99] V. Shah, R. Keniya, A. Shridharani, M. Punjabi, J. Shah, and N. Mehendale. Diagnosis of covid-19 using ct scan images and deep learning techniques. *Emerging Radiology*, 28(3):497–505, 2021.

[100] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[101] F. Shan, Y. Gao, J. Wang, W. Shi, N. Shi, M. Han, Z. Xue, D. Shen, and Y. Shi. Lung infection quantification of covid-19 in ct images with deep learning. *arXiv preprint arXiv:2003.04655*, 2020.

[102] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *Proceedings of the Computer Vision–ECCV 2016: 14th European Conference*, pages 467–482. Springer, 2016.

[103] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[104] D. Singh, V. Kumar, V. Yadav, and M. Kaur. Deep neural network-based screening model for covid-19-infected patients using chest x-ray images. *International Journal of Pattern Recognition and Artificial Intelligence*, 35(03):2151004, 2021.

[105] K. Subramaniam, N. Palanisamy, R.A. Sinnaswamy, et al. A comprehensive review of analyzing the chest x-ray images to detect covid-19 infections using deep learning techniques. *Soft Computing*, 27:14219–14240, 2023.

[106] H. Swapnarekha, H. S. Behera, J. Nayak, and B. Naik. Deep densenet and resnet approach for covid-19 prognosis: Experiments on real ct images. In *Computational Intelligence in Pattern Recognition*, pages 731–747. Springer, 2022.

[107] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[108] An Tang, Rebecca Tam, and Alexandre Cadrin-Chenevert. Canadian association of radiologists white paper on artificial intelligence in radiology. *Can Assoc Radiol J*, 69(2):120–135, 2018.

[109] Enzo Tartaglione, Carlo Alberto Barbano, Chiara Berzovini, Mirko Calandri, and Marco Grangetto. Unveiling covid-19 from chest x-ray with deep learning: A hurdles race with small data. *International Journal of Environmental Research and Public Health*, 17(18):6933, 2020.

[110] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei. Deep convolutional neural network architecture with reconfigurable computation patterns. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 25:2220–2233, 2017.

[111] Fang Wang, Heng Huang, and Jie Liu. Variational-based mixed noise removal with cnn deep learning regularization. *IEEE Transactions on Image Processing*, 29:1246–1258, 2019.

[112] S. Wang, B. Kang, J. Ma, X. Zeng, M. Xiao, J. Guo, M. Cai, J. Yang, Y. Li, and X. Meng. A deep learning algorithm using ct images to screen for coronavirus disease (covid-19). *European Radiology*, 31(8):6096–6104, 2021.

[113] S. H. Wang, J. Hong, and M. Yang. Sensorineural hearing loss identification via nine-layer convolutional neural network with batch normalization and dropout. *Multimed. Tools Appl.*, 79:15135–15150, 2020.

[114] X. Wang, X. Deng, Q. Fu, Q. Zhou, J. Feng, H. Ma, W. Liu, and C. Zheng. A weakly-supervised framework for covid-19 classification and lesion localization from chest ct. *IEEE Transactions on Medical Imaging*, 39(8):2615–2625, 2020.

[115] Qiang Wei and Bin Wang. Research on image retrieval using deep convolutional neural network combining l1 regularization and prelu activation function. *IOP Conference Series: Earth and Environmental Science*, 69:012156, 2017.

[116] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[117] Nan Xie, Xuelong Li, Ke Li, Yang Yang, and Heng Tao Shen. Statistical karyotype analysis using cnn and geometric optimization. *IEEE Access*, 7:179445–179453, 2019.

[118] Jie Xiong, Dianguang Yu, Shuai Liu, Lei Shu, Xiaoyun Wang, and Zhen Liu. A review of plant phenotypic image recognition technology based on deep learning. *Electronics*, 10:81, 2021.

[119] Rikiya Yamashita, Mizuho Nishio, Ryan K. G. Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights Imaging*, 9(4):611–629, 2018.

[120] C. Yang, Z. Yang, J. Hou, and Y. Su. A lightweight full homomorphic encryption scheme on fully-connected layer for cnn hardware accelerator achieving security inference. In *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 1–4, 2021.

[121] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[122] J. Zhang, Y. Xie, G. Pang, Z. Liao, J. Verjans, W. Li, Z. Sun, J. He, Y. Li, and C. Shen. Viral pneumonia screening on chest x-rays using confidence-aware anomaly detection. *IEEE Transactions on Medical Imaging*, 40(3):879–890, 2020.

[123] Yu Zheng, Zhiwei Gao, Yu Wang, and Qiang Fu. Mooc dropout prediction using fwts-cnn model based on fused feature weighting and time series. *IEEE Access*, 8:225324–225335, 2020.

Appendix A

Appendices

A.1 Mathematical definition

A.1.1 Optimization Algorithms

Optimization algorithms play a crucial role in adjusting the weights and biases of a neural network during training, aiming to minimize the LossFunc [79,117]. Among the widely used optimization algorithms, stochastic gradient descent (SGD) is prevalent. SGD updates the weights and biases incrementally, taking small steps determined by the gradient of the LossFunc concerning these parameters. By using the SGD optimizer, the network weights will be updated using the following equation:

$$\begin{aligned} w_{t+1} &= w_t - \eta \frac{\partial L}{\partial w_t} \\ \frac{\partial L}{\partial w_t} &= \nabla_W C(w_t; x^{(B)}; y^{(B)}) \end{aligned} \tag{A.1}$$

where η is the learning rate, x are the sample images used, y are the image labels, and w are the weights being updated.

Other optimization algorithms are as follows:

- Momentum: This algorithm adds a momentum term to the gradient update, which helps to smooth out the updates and accelerate convergence. The update rule for momentum can be written as:

$$\begin{aligned} \nu_t &= \beta\nu_{t-1} + (1-\beta)\Delta_w L(w_{t-1}) \\ w_t &= w_{t-1} - \alpha\nu_t \end{aligned} \tag{A.2}$$

  where ν_t is the momentum at time t, β is the momentum coefficient, $\Delta_w L(w_{t-1})$ is the gradient of the LossFunc with respect to the weights at time $t-1$, α is the learning rate, and w_t is the updated weights.

- AdaGrad: is an optimization algorithm that dynamically adjusts the learning rate for each weight based on the magnitude of the gradients observed during training. This adaptive learning rate facilitates faster convergence on flat directions and slower convergence on steep directions. The update rule for AdaGrad can be expressed as follows:

$$\begin{aligned} G_t &= G_{t-1} + (\Delta_w L(w_{t-1}))^2 \\ w_t &= w_{t-1} - \frac{\alpha}{\sqrt{G_t + \epsilon}}\Delta_w L(w_{t-1}) \end{aligned} \tag{A.3}$$

  where G_t is the diagonal matrix of sums of squares of past gradients up to time t, ϵ is a small constant to avoid division by zero, and all other variables are as previously defined.

- RMSProp: is an optimization algorithm that incorporates a moving average of the squared gradients to dynamically adjust the learning rate for each weight.

This adaptation aids in preventing the learning rate from becoming excessively large. The update rule for RMSProp can be formulated as:

$$\begin{aligned} G_t &= \beta G_{t-1} + (1-\beta)(\Delta_w L(w_{t-1}))^2 \\ w_t &= w_{t-1} - \frac{\alpha}{\sqrt{G_t + \epsilon}} \Delta_w L(w_{t-1}) \end{aligned} \tag{A.4}$$

where G_t is the moving average of the squared gradients up to time t, β is the decay rate, and all other variables are as previously defined.

- Adam: is an optimization algorithm that integrates concepts from both momentum and AdaGrad, resulting in an adaptive learning rate approach suitable for a broad spectrum of neural networks. The update rule for Adam can be expressed as:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\Delta_w L(w_{t-1}) \\ \nu_t &= \beta_2 \nu_{t-1} + (1-\beta_2)(\Delta_w L(w_{t-1}))^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{\nu}_t &= \frac{\nu_t}{1-\beta_2^t} \\ w_t &= w_{t-1} - \frac{\alpha}{\sqrt{\hat{\nu}_t + \epsilon}} \hat{m}_t \end{aligned} \tag{A.5}$$

where m_t and ν_t are the first and second-moment estimates of the gradients, β_1 and β_2 are the decay rates for the first and second moments, $\hat{m}_t$ and $\hat{\nu}_t$ are bias-corrected estimates of the first and second moments, ϵ is a small constant to avoid division by zero, and all other variables are as previously defined.

A.1.2 Regularization Techniques

To prevent overfitting in neural networks, regularization techniques are employed. Overfitting occurs when a model performs well on the training data but poorly on new, unseen data. These techniques involve adding a penalty term to the LossFunc, encouraging the model to possess simpler weights or reduce the magnitude of the weights [111].

Various regularization techniques are commonly used, and some of them include:

- L1 regularization: adds a penalty to the LossFunc that is proportional to the absolute value of the weights. This encourages the model to have sparse weights and can potentially lead to feature selection [115]. The regularized LossFunc for L1 regularization can be expressed as:

$$\tilde{L}(w) = L(w) + \lambda \sum_{i=1}^{n} |w_i| \tag{A.6}$$

  where $L(w)$ is the original LossFunc, w_i is the ith weight in the network, n is the total number of weights, and λ is the regularization strength.

- L2 regularization: adds a penalty to the LossFunc that is proportional to the square of the weights. This encourages the model to have small weights and aids in preventing overfitting [94]. The regularized LossFunc for L2 regularization can be expressed as:

$$\tilde{L}(w) = L(w) + \frac{\lambda}{2} \sum_{i=1}^{n} w_i^2 \tag{A.7}$$

- Dropout: is a regularization technique that involves randomly dropping out some neurons in the network during training. This prevents the model from

relying too heavily on any specific feature and can enhance generalization performance [123]. The dropout regularization is implemented by randomly setting the output of each neuron to zero with a certain probability p during training. The dropout regularization can be expressed as:

$$\begin{aligned} \tilde{y} &= r \odot y \\ P(r_i = 1) &= 1 - p \\ P(r_i = 0) &= p \end{aligned} \tag{A.8}$$

where y is the output of a layer, $\tilde{y}$ is the regularized output, r is a binary mask that is randomly generated for each training example, and $\odot$ denotes element-wise multiplication. During testing, the dropout is turned off and the output is scaled by $1 - \mathrm{p}$ to compensate for the effect of dropout.

- Early stopping: is a regularization technique that halts the training process early based on a performance metric evaluated on a validation set. This helps prevent the model from overfitting to the training data. The concept involves monitoring the validation loss or accuracy throughout training and terminating the training process when the validation performance ceases to improve [67].

A.1.3 Batch Normalization

Normalization layers are crucial in deep network training, and among the prominent normalization techniques, batch normalization (BN) has proven effective in enhancing model training speed and generalization capability [54]. The success of BN is attributed to various factors, including the reduction of internal covariate shift, enabling the use of larger learning rates, and smoothing the optimization landscape.

For a layer with $d-$dimensional input

$$X = (x^{(1)}, x^{(2)}, ..., x^{(d)})$$

in a mini-batch χ_B with size m, BN normalizes each dimension of the input samples as:

$$\begin{aligned} \hat{x}_k &= \frac{x^{(k)} - \mu_\beta^k}{\sqrt{(\sigma_\beta^k{}^2 + \epsilon)}} \\ \mu_\beta^k &= \frac{1}{m}\sum_{i=1}^{m} x_i^{(k)} \\ \sigma_\beta^k &= \frac{1}{m}\sum_{i=1}^{N}(x_i^{(k)} - \mu_\beta^k)^2 \end{aligned} \tag{A.9}$$

A.1.4 Acquisition function in Bayesian Optimization

Acquisition or utility function u(x) specify which sample x should be tried next using one of the concepts, including Probability of improvement, Expected improvement, and Lower Confidence Bound.

- Probability of improvement: $\mathrm{PI}(x) = P(f(x) \leq f(x_t^+) - \epsilon) = \Phi(\frac{f(x_t^+)+\epsilon-\mu(x)}{\sigma(x)})$;
- Improvement: $\mathrm{I}(x) = max\{0, [f(x_t^+) - f(x_t + 1)]\}$;
- Expected improvement: $\mathrm{EI}(x) = \mathbb{E}[f(x_t^+) - f(x_t+1)] = (f(x_t^+)+\epsilon-\mu(x))\Phi(Z)+(\sigma(x))\phi(Z)$ where Φ is the CDF and ϕ is the PDF of a standard Gaussian distribution and $Z = \frac{f(x_t^+)+\epsilon-\mu(x)}{\sigma(x)}$;
- Lower Confidence Bound: $\mathrm{LCB}(x) = \mu_{GP}(x) - \kappa\sigma_{GP}(x)$;

where x_t^+ is the best-observed value thus far. In both abovementioned equations, the regulation constant ϵ, which is added to the $f(x_t^+)$, is used to indicate that there could be a value higher than the best-found value so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (*exploitation*)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (*exploration*)

In the end, we need to plug everything together ($t = 0$) and repeat until convergence:

$$x_{t+1} = \arg\min_x \mathrm{LCB}(x)$$

A.2 Pre-trained CNN Models Architecture

Deep Learning algorithms surpass classical machine learning algorithms in accuracy, particularly when handling extensive datasets and raw images. DL excels at extracting knowledge and information without the necessity for preprocessing, enhancement, or segmentation of images. This capability extends to image analysis, where DL algorithms demonstrate notable enhancements [69]. Their application extends to disease detection, notably in areas like identifying COVID-19 and diagnosing retinal diseases affecting the iris and delicate nerves, potentially leading to blindness [25]. The proficiency of DL algorithms in processing vast and unprocessed data underscores their efficacy in complex tasks, making them pivotal in advancing medical diagnostics and image-based analyses.

Moreover, DL algorithms find application in classification tasks, notably in analyzing Magnetic Resonance Imaging (MRI) images. MRI, a medical imaging technology,

produces detailed images of the body's organs and tissues through the combination of a magnetic field and computer-generated radio waves. In various studies, researchers leverage DL algorithms, particularly Convolutional Neural Networks, a subset of Artificial Neural Networks (ANNs). Comprising four layers—convolution, pooling, fully linked, and non-linearity—CNNs excel in enhancing pattern recognition and image classification [11]. Researchers exploring novel coronavirus detection have employed diverse CNN architectures, including Visual Geometry Group (VGG), Residual Convolution Neural Network (ResNet), and Dense Convolution Network (DenseNet). The choice of CNN architectures is tailored to the data's size and characteristics.

In recent times, CNN architectures have demonstrated superior performance in complex tasks, notably in medical image analysis and disease detection. Yann LeCun pioneered the development of effective CNNs with the creation of LeNet in 1998. Initially designed for handwritten digit detection, LeNet comprised three convolution layers, two pooling layers, and two fully connected layers. Subsequently, this section explores some of the widely recognized CNN architecture.

A.2.1 LeNet

LeNet, created by Yann L.C. et al. in 1998, stands as one of the initial CNN designs, specifically devised for handwritten digit recognition. The LeNet structure comprises two convolution layers succeeded by two fully connected layers. These convolution layers employ diminutive filters and pooling layers to extract features from the input image. The fully connected layers, utilizing softmax activation, produce a probability distribution across potential classes [58].

Mathematically, the LeNet architecture can be represented as:

1. ConvLayer with 6 filters of size 5×5, followed by a 2×2 max PoolLayer.
2. ConvLayer with 16 filters of size 5×5, followed by a 2×2 max PoolLayer.
3. Fully connected layer with 120 units and a sigmoid ActivFunc.
4. Fully connected layer with 84 units and a sigmoid ActivFunc.
5. Output layer with 10 units and a softmax ActivFunc.

A.2.2 AlexNet

AlexNet, created by Alex Krizhevsky, represents an evolution from LeNet, featuring increased depth with additional filters, stacked convolution layers, dropout, and max pooling. In 2012, AlexNet achieved a remarkable 17% top-five error rate and secured victory in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) contest, an annual competition held from 2010 to 2017 [64]. Researchers commonly leverage AlexNet in COVID-19 detection efforts to address the challenge of overfitting. Overfitting occurs when deep learning models exhibit higher accuracy on training data compared to testing data, and AlexNet proves effective in mitigating this issue.

Mathematically, the AlexNet architecture can be represented as:

1. ConvLayer with 96 filters of size 11×11, followed by a 2×2 max PoolLayer.
2. ConvLayer with 256 filters of size 5×5, followed by a 2×2 max PoolLayer.
3. ConvLayer with 384 filters of size 3×3.
4. ConvLayer with 384 filters of size 3×3.
5. ConvLayer with 256 filters of size 3×3, followed by a 2×2 max PoolLayer.
6. Fully connected layer with 4096 units and a ReLU ActivFunc.
7. Fully connected layer with 4096 units and a ReLU ActivFunc.
8. Output layer with 1000 units and a softmax ActivFunc

A.2.3 GoogLeNet

GoogLeNet surpasses AlexNet in depth, boasting 22 or 27 layers when considering pooling layers. In 2014, GoogLeNet achieved a remarkable victory in the ILSVRC contest with a 6.67% top-five error rate. A pivotal element of GoogLeNet is the inception module (IM), functioning as a miniature network capable of learning spatial and cross-channel correlations. The IM offers several advantages, including enabling the training of significantly deeper models with ten times fewer learnable parameters. The IM's output features fewer maps than its input, effectively reducing dimensionality. Moreover, the IM excels in capturing intricate patterns at various scales within both spatial and depth dimensions [107].

A.2.4 VGGNet

VGGNet, with its 19 convolution layers, achieved a notable 7.3% top-five error rate in the 2014 ILSVRC contest. Unlike AlexNet, VGGNet embraces architectural simplicity, featuring three fully connected layers. Developed by the Visual Geometry Group at Oxford University, VGGNet's straightforward design contributes to its widespread adoption in various domains. Despite its simplicity, VGGNet employs three times as many parameters as AlexNet. The architecture serves as a foundation for advanced object identification models and outperforms baselines across diverse tasks and datasets beyond ImageNet. VGGNet's enduring popularity persists as one of the most utilized image recognition architectures [103].

Mathematically, the VGG architecture can be represented as:

1. ConvLayer with 64 filters of size 3×3, followed by a 2×2 max PoolLayer.
2. ConvLayer with 128 filters of size 3×3, followed by a 2×2 max PoolLayer.

3. ConvLayer with 256 filters of size 3×3, followed by a 2×2 max PoolLayer.
4. ConvLayer with 512 filters of size 3×3, followed by a 2×2 max PoolLayer.
5. ConvLayer with 512 filters of size 3×3, followed by a 2×2 max PoolLayer.
6. Fully connected layer with 4096 units and a ReLU ActivFunc, followed by dropout regularization.
7. Fully connected layer with 4096 units and a ReLU ActivFunc, followed by dropout regularization.
8. Output layer with 1000 units and a softmax ActivFunc.

A.2.5 ResNet

ResNet, the winner of the 2015 ILSVRC contest with 152 layers, introduced a revolutionary residual module featuring a standard layer and a skip connection. This unique architecture achieved a remarkable 3.6% top-five error rate. The skip connection, linking the input signal of a layer to its output, facilitates the traversal of the input signal across the network. Residual Units (RUs) empowered the training of an exceptionally deep 152-layer model. A block is formed by connecting layer activations to subsequent layers, and these building blocks are stacked to construct ResNets. The skip link's advantage lies in regularization, as it skips any layer detrimental to architecture performance, enabling the training of extremely deep neural networks without grappling with issues like vanishing or exploding gradients [44].

Mathematically, the ResNet architecture can be represented as:

1. convolution layer with 64 filters of size 7×7, followed by a 3×3 max PoolLayer.
2. Multiple residual blocks, each of which consists of:
   (a) ConvLayer with 64 filters of size 3×3.

(b) ConvLayer with 64 filters of size 3×3.

(c) Shortcut connection that adds the original input to the output of the block.

3. Multiple residual blocks, each of which consists of:

   (a) ConvLayer with 128 filters of size 3×3, followed by a 2×2 max PoolLayer.

   (b) ConvLayer with 128 filters of size 3×3.

   (c) ConvLayer with 128 filters of size 3×3.

   (d) Shortcut connection that adds the original input to the output of the block.

4. Multiple residual blocks, each of which consists of:

   (a) ConvLayer with 256 filters of size 3×3, followed by a 2×2 max PoolLayer.

   (b) ConvLayer with 256 filters of size 3×3.

   (c) ConvLayer with 256 filters of size 3×3.

   (d) Shortcut connection that adds the original input to the output of the block.

5. Multiple residual blocks, each of which consists of:

   (a) ConvLayer with 512 filters of size 3×3, followed by a 2×2 max PoolLayer.

   (b) ConvLayer with 512 filters of size 3×3.

   (c) ConvLayer with 512 filters of size 3×3.

   (d) Shortcut connection that adds the original input to the output of the block.

6. Fully connected layer with 1000 units and a softmax ActivFunc.

A.2.6 Inception

Inception, initially introduced by Google in 2014 as Inception V1, revolutionized image model blocks by simulating an optimal local sparse structure in CNNs. This architecture strategically employs numerous filters of different sizes on the same level

to mitigate data overfitting and address computational expense issues. Unlike traditional approaches using a single filter size, Inception combines multiple filter sizes into a unified image block before passing it to the subsequent layer. This innovative design enhances the model's ability to capture diverse features within the input data, contributing to its effectiveness in various computer vision tasks [88].

Mathematically, the InceptionNet architecture can be represented as:

1. ConvLayer with 64 filters of size 7×7, followed by a 2×2 max pooling layer.
2. ConvLayer with 64 filters of size 1×1, followed by a ConvLayer with 192 filters of size 3×3, and a 2×2 max pooling layer.
3. Multiple inception modules, each of which consists of:
   (a) ConvLayer with 64 filters of size 1×1.
   (b) ConvLayer with 96 filters of size 3×3.
   (c) ConvLayer with 128 filters of size 5×5.
   (d) Max pooling layer with a 3×3 filter and stride 1.
   (e) Concatenation of the outputs of the previous layers.
4. Multiple inception modules, each of which consists of:
   (a) ConvLayer with 192 filters of size 1×1.
   (b) ConvLayer with 96 filters of size 3×3.
   (c) ConvLayer with 208 filters of size 5×5.
   (d) Max PoolLay with a 3×3 filter and stride 1.
   (e) Concatenation of the outputs of the previous layers.
5. Multiple inception modules, each of which consists of:
   (a) ConvLayer with 160 filters of size 1×1.
   (b) ConvLayer with 112 filters of size 3×3.

(c) ConvLayer with 224 filters of size 5×5.
(d) Max pooling layer with a 3×3 filter and stride 1.
(e) Concatenation of the outputs of the previous layers.

A.2.7 DenseNet

Dense Convolutional Network (DenseNet) is another popular architecture that won the 2017 ImageNet competition. In DenseNet, each layer receives additional inputs from all preceding layers and passes on its own feature maps to all subsequent layers. This structure enables each layer to gain collective knowledge from all previous layers. Because each layer aggregates feature maps from all preceding layers, the network can be designed to be thinner and more compact, resulting in higher computational and memory efficiency [106].

1. ConvLayer with 96 filters of size 7×7, followed by a 3×3 max PoolLayer with stride 2.
2. Dense Block 1:

   (a) 6 Dense layers, each containing:
   (b) BatchNorm (BN)
   (c) ReLU Activation Function
   (d) ConvLayer with 1×1 filters (Bottleneck Layer)
   (e) BatchNorm (BN)
   (f) ReLU Activation Function
   (g) ConvLayer with 3×3 filters
   (h) Transition Layer after Dense Block 1:

      i. BatchNorm (BN)
      ii. ConvLayer with 1×1 filters

iii. 2×2 average PoolLayer with stride 2

3. Dense Block 2:

   (a) 12 Dense layers, each containing:
   (b) BatchNorm (BN)
   (c) ReLU Activation Function
   (d) ConvLayer with 1×1 filters (Bottleneck Layer)
   (e) BatchNorm (BN)
   (f) ReLU Activation Function
   (g) ConvLayer with 3×3 filters
   (h) Transition Layer after Dense Block 2:

      i. BatchNorm (BN)
      ii. ConvLayer with 1×1 filters
      iii. 2×2 average PoolLayer with stride 2

4. Dense Block 3:

   (a) 36 Dense layers, each containing:
   (b) BatchNorm (BN)
   (c) ReLU Activation Function
   (d) ConvLayer with 1×1 filters (Bottleneck Layer)
   (e) BatchNorm (BN)
   (f) ReLU Activation Function
   (g) ConvLayer with 3×3 filters
   (h) Transition Layer after Dense Block 3:

      i. BatchNorm (BN)
      ii. ConvLayer with 1×1 filters
      iii. 2×2 average PoolLayer with stride 2

5. Dense Block 4:

   (a) 24 Dense layers, each containing:
   (b) BatchNorm (BN)
   (c) ReLU Activation Function
   (d) ConvLayer with 1×1 filters (Bottleneck Layer)
   (e) BatchNorm (BN)
   (f) ReLU Activation Function
   (g) ConvLayer with 3×3 filters

6. Final Layers:

   (a) BatchNorm (BN)
   (b) ReLU Activation Function
   (c) 7×7 global average PoolLayer
   (d) Fully connected layer with 1000 units
   (e) Softmax Activation Function

A.2.8 SqueezeNet

SqueezeNet, proposed by Iandola et al. [52], is a compact CNN architecture that achieves AlexNet-level accuracy on ImageNet while using 50 times fewer parameters. By employing model compression techniques, the authors successfully reduced SqueezeNet's size to less than 0.5MB, making it highly popular for applications requiring lightweight models.

1. ConvLayer with 96 filters of size 7×7, followed by a 3×3 max PoolLayer with stride 2.
2. Fire Module with:

   (a) Squeeze: 1×1 ConvLayer with 16 filters.

(b) Expand: 1 × 1 ConvLayer with 64 filters and 3 × 3 ConvLayer with 64 filters.

3. Fire Module with:

   (a) Squeeze: 1 × 1 ConvLayer with 16 filters.
   (b) Expand: 1 × 1 ConvLayer with 64 filters and 3 × 3 ConvLayer with 64 filters.

4. Fire Module with:

   (a) Squeeze: 1 × 1 ConvLayer with 32 filters.
   (b) Expand: 1 × 1 ConvLayer with 128 filters and 3 × 3 ConvLayer with 128 filters.

5. 3 × 3 max PoolLayer with stride 2.
6. Fire Module with:

   (a) Squeeze: 1 × 1 ConvLayer with 32 filters.
   (b) Expand: 1 × 1 ConvLayer with 128 filters and 3 × 3 ConvLayer with 128 filters.

7. Fire Module with:

   (a) Squeeze: 1 × 1 ConvLayer with 48 filters.
   (b) Expand: 1 × 1 ConvLayer with 192 filters and 3 × 3 ConvLayer with 192 filters.

8. Fire Module with:

   (a) Squeeze: 1 × 1 ConvLayer with 48 filters.
   (b) Expand: 1 × 1 ConvLayer with 192 filters and 3 × 3 ConvLayer with 192 filters.

9. Fire Module with:
    (a) Squeeze: 1×1 ConvLayer with 64 filters.
    (b) Expand: 1×1 ConvLayer with 256 filters and 3×3 ConvLayer with 256 filters.
10. 3×3 max PoolLayer with stride 2.
11. Fire Module with:
    (a) Squeeze: 1×1 ConvLayer with 64 filters.
    (b) Expand: 1×1 ConvLayer with 256 filters and 3×3 ConvLayer with 256 filters.
12. ConvLayer with 1000 filters of size 1×1, followed by a global average PoolLayer.
13. Output layer with 1000 units and a softmax Activation Function.

A.2.9 WideResNet

Wide Residual Networks (WideResNet) are a variation of the original Residual Networks (ResNet) that address the problem of diminishing feature reuse in deep networks. Introduced by Sergey Zagoruyko and Nikos Komodakis in their 2016 paper "Wide Residual Networks" [121]. WideResNet modifies the original ResNet architecture by increasing the width of residual blocks, which means adding more filters in each convolutional layer, while simultaneously reducing the depth (number of layers).

1. ConvLayer with 64 filters of size 7×7, followed by a 3×3 max PoolLayer with stride 2.
2. Residual Block 1:
    (a) Block 1.1:

i. ConvLayer with 64 filters of size 1×1
ii. BatchNorm (BN)
iii. ReLU Activation Function
iv. ConvLayer with 64 filters of size 3×3
v. BatchNorm (BN)
vi. ReLU Activation Function
vii. ConvLayer with 256 filters of size 1×1
viii. BatchNorm (BN)
ix. Skip Connection: ConvLayer with 256 filters of size 1×1

(b) Block 1.2 and 1.3:

i. Similar to Block 1.1 but without the initial ConvLayer in the skip connection

3. Residual Block 2 :

(a) Block 2.1:

i. ConvLayer with 128 filters of size 1×1
ii. BatchNorm (BN)
iii. ReLU Activation Function
iv. ConvLayer with 128 filters of size 3×3
v. BatchNorm (BN)
vi. ReLU Activation Function
vii. ConvLayer with 512 filters of size 1×1
viii. BatchNorm (BN)
ix. Skip Connection: ConvLayer with 512 filters of size 1×1

(b) Block 2.2, 2.3, and 2.4:

i. Similar to Block 2.1 but without the initial ConvLayer in the skip connection

4. Residual Block 3:

(a) Block 3.1:

i. ConvLayer with 256 filters of size 1×1
ii. BatchNorm (BN)
iii. ReLU Activation Function
iv. ConvLayer with 256 filters of size 3×3
v. BatchNorm (BN)
vi. ReLU Activation Function
vii. ConvLayer with 1024 filters of size 1×1
viii. BatchNorm (BN)
ix. Skip Connection: ConvLayer with 1024 filters of size 1×1

(b) Block 3.2 through 3.6:

i. Similar to Block 3.1 but without the initial ConvLayer in the skip connection

5. Residual Block 4:

(a) Block 4.1:

i. ConvLayer with 512 filters of size 1×1
ii. BatchNorm (BN)
iii. ReLU Activation Function
iv. ConvLayer with 512 filters of size 3×3
v. BatchNorm (BN)
vi. ReLU Activation Function
vii. ConvLayer with 2048 filters of size 1×1
viii. BatchNorm (BN)
ix. Skip Connection: ConvLayer with 2048 filters of size 1×1

(b) Block 4.2 and 4.3:

i. Similar to Block 4.1 but without the initial ConvLayer in the skip connection

6. Final Layers :

   (a) BatchNorm (BN)
   (b) ReLU Activation Function
   (c) 7×7 global average PoolLayer
   (d) Fully connected layer with 1000 units
   (e) Softmax Activation Function

Appendix B

Appendices

B.1 X-ray Images from Proposed Dataset

The sample images from both healthy and COVID-19 cases, along with their corresponding RGB enhanced images depicted in the following figures. These figures showcase a diverse range of X-ray images from various domains, exhibiting different qualities and positions.

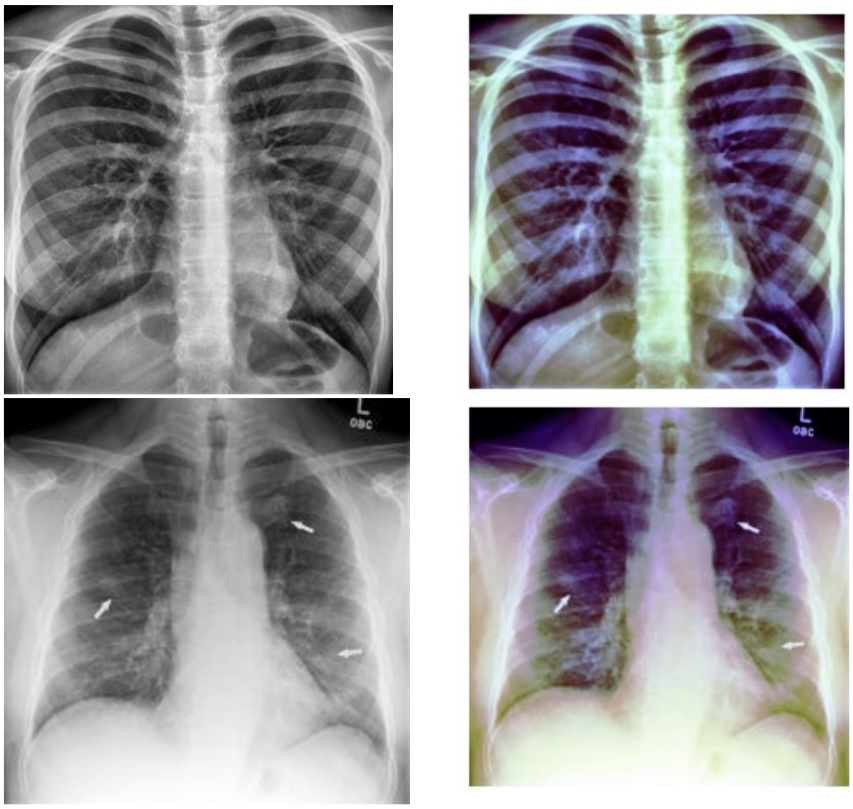

Figure B.1: Sample COVID-19 images and enhanced counterpart

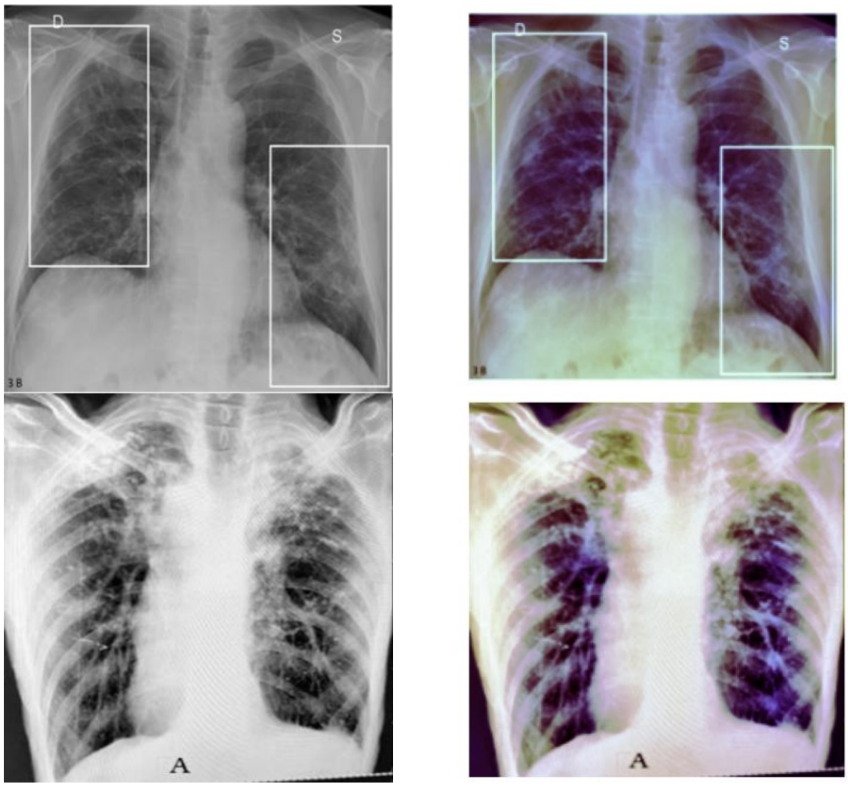

Figure B.2: Sample COVID-19 images and enhanced counterpart

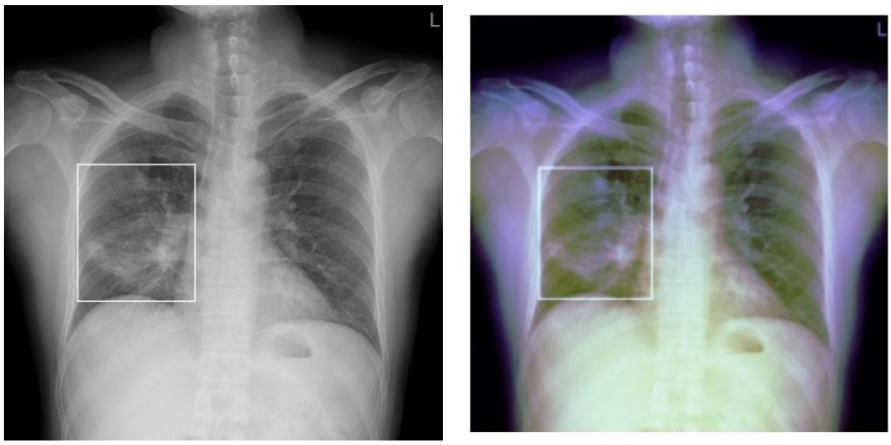

Figure B.3: Sample COVID-19 images and enhanced counterpart

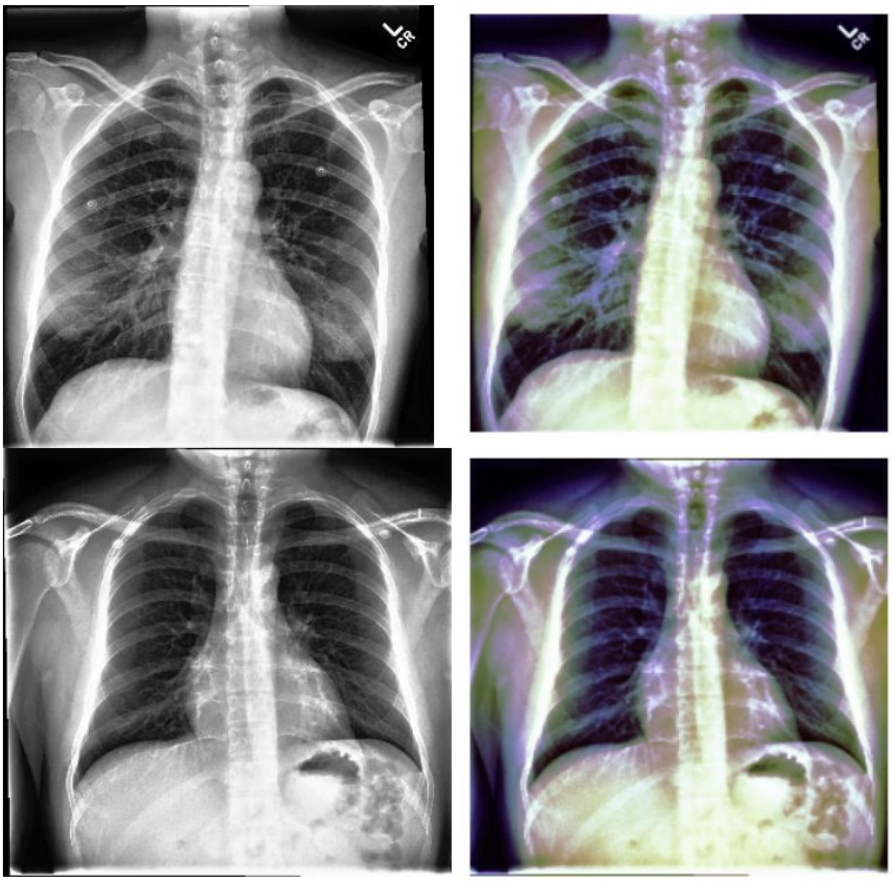

Figure B.4: Sample healthy images and enhanced counterpart

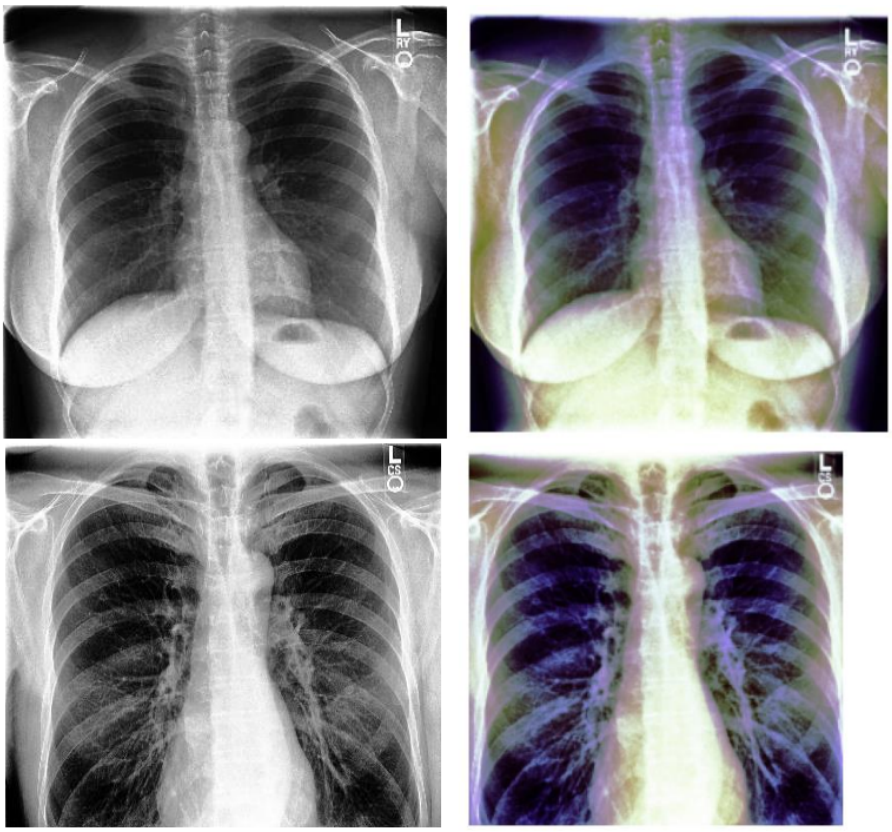

Figure B.5: Sample healthy images and enhanced counterpart

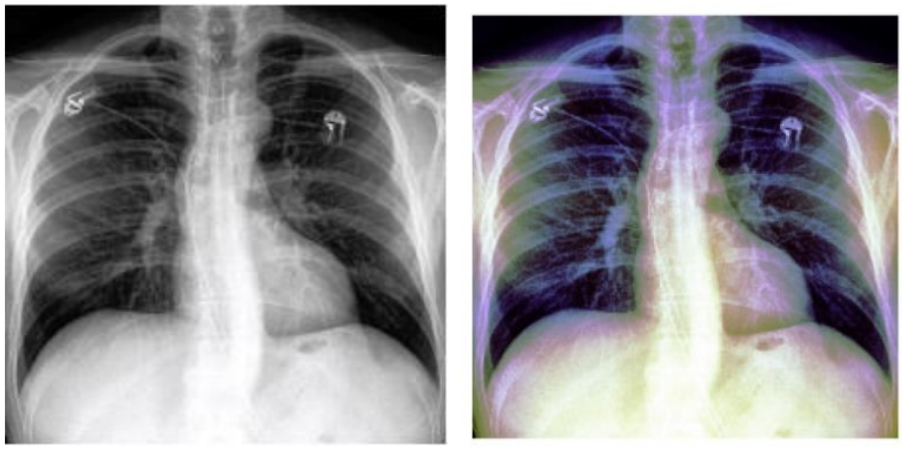

Figure B.6: Sample healthy images and enhanced counterpart

Figure B.7: Sample COVID-19 X-ray images

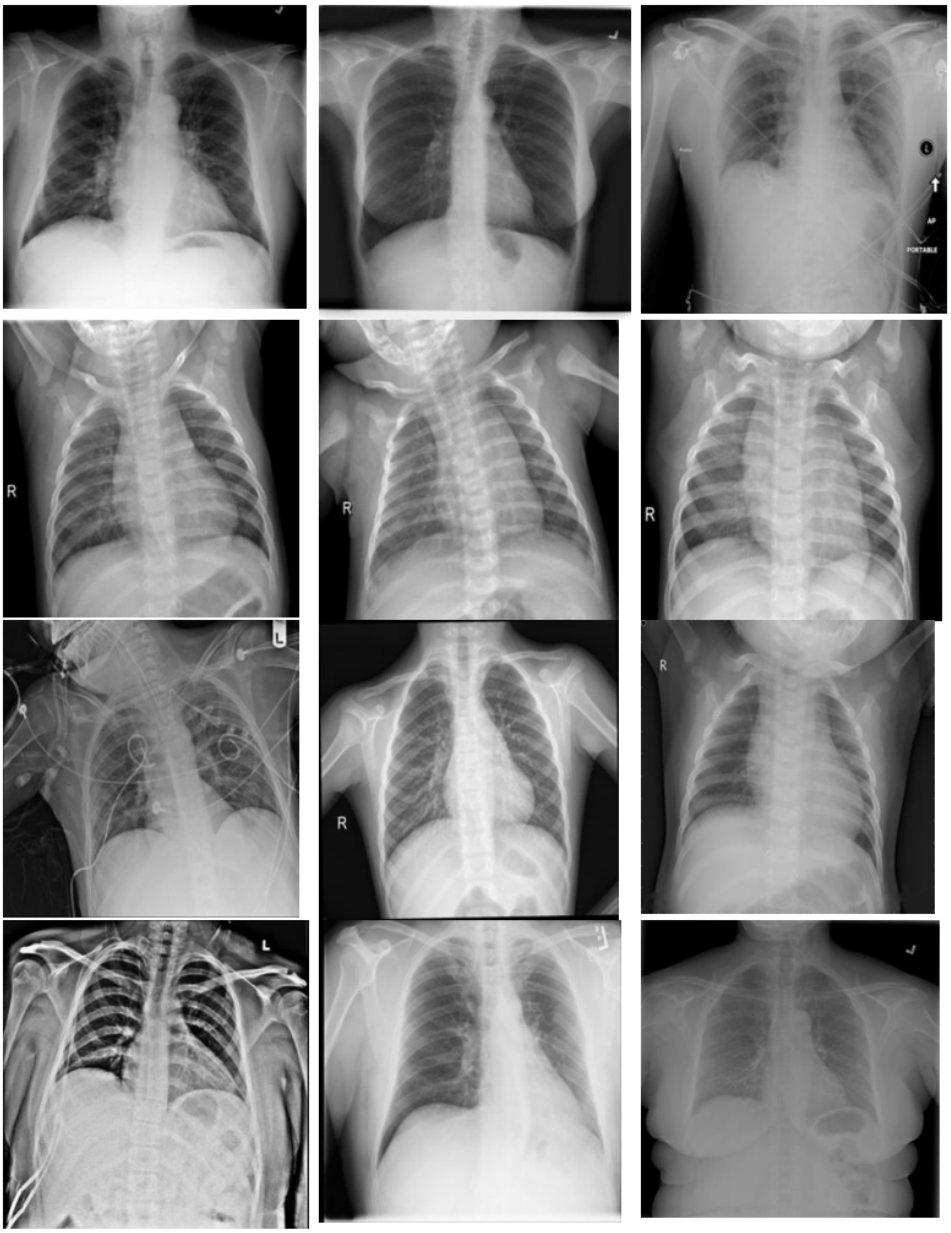

Figure B.8: Sample healthy X-ray images