Università degli Studi di Milano
Facoltà di Scienze e Tecnologie
Corso di Laurea Magistrale in Informatica

Advanced Analysis of Convolutional Neural Network Classification: Understanding the Decision-Making Process in Chest X-ray Diagnosis

Relatore: Prof. Nunzio Alberto BORGHESE

Tesi di:
Francesco PINESCHI
Matricola: 969658

Anno Accademico 2023-2024

Abstract

This thesis aims to study the decision-making process of a convolutional neural network (**CNN**), with a focus on analyzing the distinctive patterns extracted during the classification of radiographic images (**X-rays**) into three classes: *COVID-19*, *viral pneumonia*, and non-patological images. The main objectives include analyzing the network's **decision making** process, visually exploring the relevant areas in the images for each class, and examining classification errors. To achieve these goals, three explainability frameworks — **GradCAM**, **LIME**, and **SHAP** — were employed and compared. These tools provide a visual analysis of the areas considered relevant by the network, an in-depth examination of the network's internal workings, layer by layer and a statistical method to calculate class similarity. The results of this work will contribute to enhance the understanding of the model's decisions and to improve transparency in medical diagnostics.

Contents

Chapter 1

Introduction

Convolutional Neural Networks (**CNNs**) represent one of the most advanced technologies in the field of artificial intelligence applied to computer vision. Due to their ability to automatically extract and analyze complex features from data, CNNs have been applied in numerous sectors, such as biometric security systems, autonomous driving, and medical diagnostics. The field of radiology is one sector that can benefit significantly from the application of deep learning models. A concrete example was provided during the COVID-19 pandemic. The emergency generated a large number of radiographic images, creating an extensive dataset that was used to train deep learning models during and after the pandemic. These tools have not only supported doctors in diagnosing respiratory diseases but have also stimulated progress in AI, promoting the development of increasingly precise and efficient models. However, the use of CNNs presents significant challenges in terms of interpretability, as they are often considered "black-box" models, meaning systems whose decisions are not immediately understandable.

This thesis addresses the issue of interpretability applied to CNNs, with the aim of understanding and analyzing the decision-making process of a specific network, **VGG16**, applied to the classification of radiographic images into three classes: *COVID-19*, *viral pneumonia*, and non-pathological images.

1.1 Research Objectives

The main objectives of this research can be summarized as follows:

- Analyze the **decision-making process** of the VGG16 network, investigating how the different layers of the network contribute to the final predictions.
- Identify the **distinctive patterns** extracted by the network for each class, with

particular focus on the areas of the images considered relevant for the model's decisions.

- Perform an **analysis of classification errors**, examining the cases where the model fails in order to identify potential limitations or biases in the network's behavior.

To achieve these objectives, three explainability frameworks were chosen: **Grad-CAM**, **LIME**, and **SHAP**. These tools were selected because they represent the most advanced methodology for CNNs in particular: GradCAM provides visualizations of relevant areas in the images, LIME offers local explanations based on interpretable models, while SHAP combines visual and numerical explanations, allowing for the calculation of similarity between classes.

1.2 Research Context

The COVID-19 pandemic has highlighted the importance of rapid and effective diagnostic tools that can support doctors in managing a large number of patients. Doctors use radiographic images (**X-rays**) and computed tomography (**CT**) for diagnoses. In this study, X-ray images were analyzed, as CT images involve greater complexity in the training process, being composed of sections from three planes that together reconstruct a three-dimensional image.

In COVID-19 diagnostic research, certain distinctive features have been identified, such as "***ground-glass opacities***", typical signs of SARS-CoV-2 infection. However, distinguishing between *COVID-19*, viral pneumonia, and normal conditions is not always immediate, even for experienced radiologists. The network used in this study will focus specifically on distinguishing these patterns and their spatial distribution.

The **VGG16** network, used in this work, is a well-established architecture in the field of image processing. Despite its relative simplicity compared to more recent models, VGG16 is particularly suited for interpretative analysis due to its modular structure and the presence of numerous convolutional layers.

1.3 Methodology

The work presented in this thesis is divided into several phases, each aimed at addressing the objectives outlined above:

1. **Dataset preparation**: Radiographic images from three classes (*COVID-19*, *viral pneumonia*, non-pathological images) were collected and preprocessed to ensure data uniformity and quality.

2. **Model training**: The VGG16 network was trained on the dataset using established techniques to optimize its performance.

3. **Prediction analysis**: Using GradCAM, LIME, and SHAP, the network's decisions were analyzed, with particular attention to the areas of the image considered relevant.

4. **Comparison of explainability frameworks**: The three tools were evaluated based on their ability to provide visual and numerical explanations, as well as their usefulness in error analysis.

1.4 Structure of the Thesis

The thesis is organized into seven chapters, each addressing a specific aspect of the work:

- **Chapter 2**: Introduces artificial neural networks, focusing on CNNs, describing their architecture, functioning, and training techniques.

- **Chapter 3**: Examines the clinical context of *COVID-19*, with particular attention to the radiographic features that distinguish this condition from viral pneumonia and normal conditions.

- **Chapter 4**: Presents the VGG16 network and describes the explainability frameworks GradCAM, LIME, and SHAP, highlighting their theoretical principles and practical applications.

- **Chapter 5**: Details the experimental phase, from dataset preparation to model training and performance evaluation.

- **Chapter 6**: Delves into the analysis of the network's predictions using the explainability frameworks, focusing on the identification of distinctive patterns and error analysis.

- **Chapter 7**: Summarizes the main results of the thesis, discussing the implications of the findings and proposing potential future developments.

Chapter 2

Neural Networks

In order to properly understand the contents of this paper, it is important to review what machine learning is and what neural networks are.

2.1 What is an Artificial Neural Network?

Deep learning is a branch of artificial intelligence that uses deep neural networks to analyze complex data. These networks, composed of many layers of *neurons*, autonomously learn to recognize patterns and make predictions, making them particularly effective in tasks like image recognition and natural language processing. A neural network is a set of sequential layers of neurons, where:

- Each neuron receives numerical inputs, applies a function (often nonlinear), and passes the result to the neurons in the next layer.
- The network consists of *weights* (model parameters), which are numbers associated with the connections between neurons. During training, these weights are updated to minimize the error between the network's output and the real data.

A neural network is therefore an architecture that autonomously learns to solve complex problems, such as image recognition, natural language processing, or robot control.

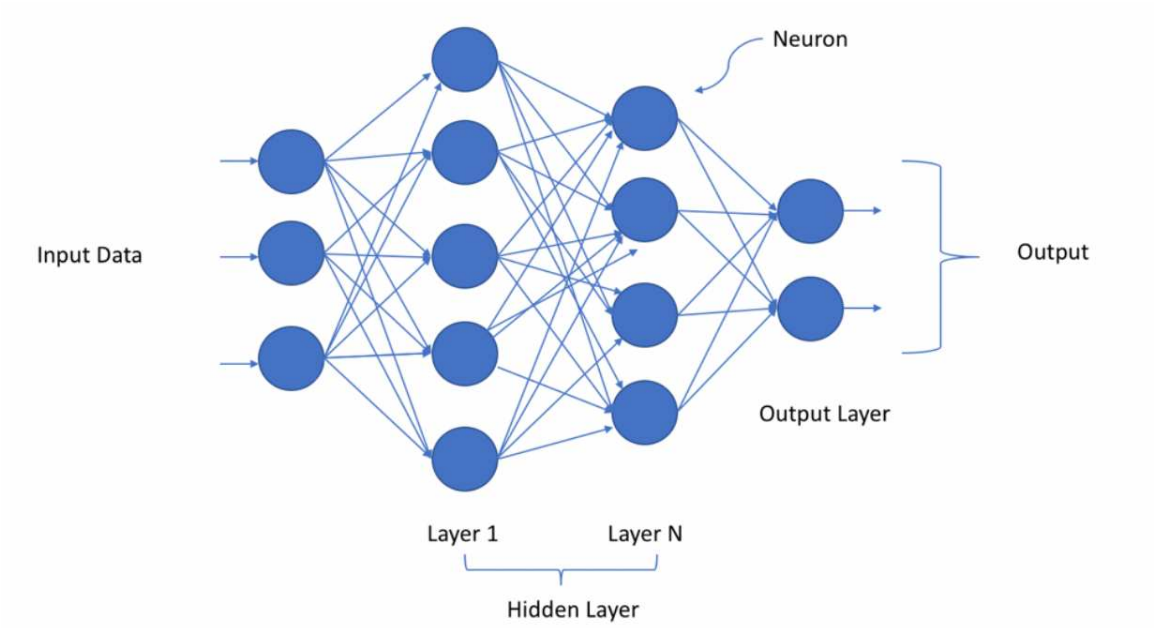

Figure 1: Generic structure of a deep neural network.

2.1.1 Structure of Layers in a Neural Network

- **Input Layer:** This layer receives the raw input data (for example, the pixels of an image or numerical parameters). It does not perform calculations but distributes the data to the successive layers.
- **Hidden Layers:** These are intermediate layers that process the input data and extract relevant features. These layers apply complex transformations to the data using nonlinear activation functions (such as *ReLU* or *Sigmoid*), allowing the network to learn complex patterns. The more hidden layers there are, the "deeper" the network, hence the term *deep learning*.
- **Output Layer:** This is the final layer of the network that produces the final result, such as a predicted class (e.g., "cat" or "dog" in an image classifier). The activation function used in the output depends on the type of problem:
  - *Softmax* for multi-class classification.
  - *Sigmoid* for binary classification.
  - No activation for regression problems.

2.1.2 How Layers Work

Features are numerical representations that describe relevant aspects of data, such as an image or text. In a neural network, each layer processes the features received from the previous layer, progressively extracting more abstract and complex representations.

This process involves taking the output data from the previous layer (in the case of hidden layers) or directly from the input layer, multiplying it by weights (parameters learned during training), adding a bias, and applying an activation function.

However, more layers can increase computational complexity and the risk of *overfitting*, which can be controlled with techniques like *dropout* or regularization.
In summary, the layers in deep learning neural networks are fundamental units that process and transform data, allowing the network to learn complex patterns and make accurate predictions.

2.2 Training a Neural Network

The **training** process of a neural network involves updating the weights of the connections between neurons to minimize the error between the network's predictions and the expected values. Training is based on two fundamental concepts: *forward propagation* and *backpropagation*, using optimization techniques such as *gradient descent*.

2.2.1 Forward Propagation

During *forward propagation*, the input passes through the network's various layers, moving through neurons that apply activation functions to their weighted inputs.
The output of a neuron j in layer l is given by:

$$a_j^{(l)} = f\left(\sum_i w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)}\right) \tag{1}$$

where:

- $a_i^{(l-1)}$ is the activation of neuron i in the previous layer $(l-1)$,
- $w_{ij}^{(l)}$ is the weight between neuron i in layer $l-1$ and neuron j in layer l,
- $b_j^{(l)}$ is the bias of neuron j in layer l,

- $f(\cdot)$ is the activation function (such as ReLU, Sigmoid, or Tanh).

2.2.2 Cost Function

The **cost function** quantifies the error between the network's predictions and the desired (target) values. For classification problems, a common cost function is *cross-entropy*:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c=1}^{C} y_{i,c} \log(h_\theta(x_i)_c) \tag{2}$$

where:

- m is the number of training examples,
- C is the number of classes,
- $y_{i,c}$ is the target value for example i and class c,
- θ represents the model parameters (weights),
- $h_\theta(x_i)_c$ is the probability predicted by the network for example i belonging to class c.

2.2.3 Backpropagation and Gradient Descent

To minimize the cost function, the network uses **backpropagation**, a technique to calculate the gradients of the cost function with respect to the network's weights. The gradient of the cost function $J(\theta)$ with respect to a weight $w_{ij}^{(l)}$ is given by:

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)} \tag{3}$$

where:

- $\delta_j^{(l)}$ is the error for neuron j in layer l,
- $a_i^{(l-1)}$ is the activation of neuron i in the previous layer.

The error $\delta_j^{(l)}$ is calculated by propagating the error from the next layer:

$$\delta_j^{(l)} = \left(\sum_k w_{jk}^{(l+1)} \delta_k^{(l+1)} \right) f'(z_j^{(l)}) \tag{4}$$

where $f'(z_j^{(l)})$ is the derivative of the activation function with respect to the weighted input $z_j^{(l)}$.

2.2.4 Weight Update

Weights are updated using **gradient descent**. The update rule is:

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{\partial J}{\partial w_{ij}^{(l)}} \tag{5}$$

where η is the *learning rate*, which controls the speed at which weights are updated. This process repeats iteratively for each batch of training data until the model converges to an optimal solution, minimizing the network's error.

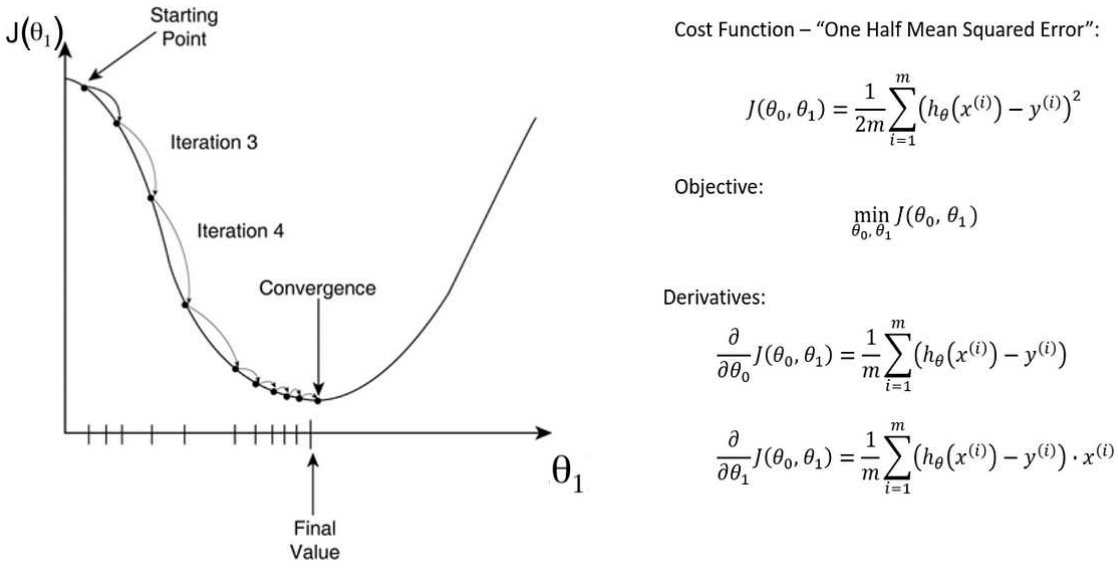

Figure 2: Representation of gradient descent

- **Overfitting**: When a model learns too much from the training data, including noise and details that do not generalize well, leading to high performance on the training set but poor performance on unseen data.
- **Underfitting**: When a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and test sets.

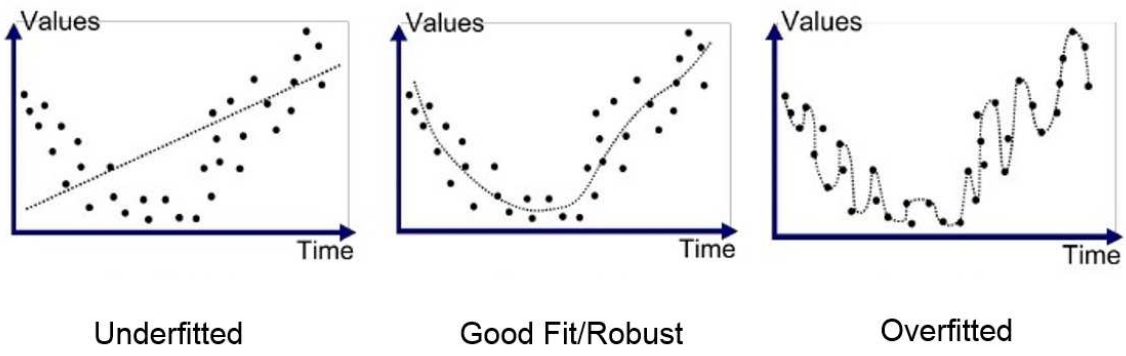

Figure 3: examples of Underfitting, Good fit and Overfitting

The image illustrates three examples of model performance in fitting data points:

- **Underfitting (Left):** The curve fails to capture the underlying pattern of the data. It is overly simplistic and does not approximate the distribution of the points, resulting in poor performance on both the training set and unseen data.
- **Robust Fit (Center):** The curve demonstrates a well-balanced fit, maintaining an optimal distance from most data points. It effectively captures the underlying pattern while avoiding overcomplication, ensuring good generalization to unseen data.
- **Overfitting (Right):** The curve closely follows the data points, capturing noise and irregularities in the dataset. This results in poor generalization, as the model is overly tailored to the specific dataset and performs poorly on new, unseen data.

2.3 Characteristics of Deep Networks

2.3.1 Linear Units and Data Filtering

Each layer in a deep neural network can be seen as a filter or a linear transformation of the data. The fundamental computation in each neuron is given by:

$$z = W \cdot x + b \tag{6}$$

Where:

- z: The linear output of the neuron before applying the activation function.

- W: The weight vector, which determines how the input x is combined.
- x: The input data (e.g., numerical values or features).
- b: The bias term, which allows for a shift in the activation function, increasing flexibility.

The depth of the network enables the construction of **hierarchical representations**:

- **Early layers:** Extract low-level features such as edges and textures.
- **Later layers:** Combine these features to detect more complex structures like shapes or complete objects.

2.3.2 Activation Function and ReLU

The activation function introduces **non-linearity** into the network, enabling it to learn complex, non-linear relationships within the data. Without activation functions, each layer would simply perform a linear transformation, limiting the model's capacity.
One widely used activation function is the **ReLU (Rectified Linear Unit)**, defined as:

$$f(x) = \max(0, x) \tag{7}$$

Purpose of ReLU:

- Introduces non-linearity while maintaining computational simplicity.
- Reduces the issue of **vanishing gradient**, common with functions like Sigmoid and Tanh.

Vanishing Gradient: This problem occurs when gradients (which guide weight updates during training) become very small in deeper layers. It is often caused by activation functions with derivatives close to zero for extreme input values, such as Sigmoid and Tanh. ReLU mitigates this by ensuring gradients remain constant for positive inputs, accelerating learning.

2.3.3 Sub-sampling (Pooling)

Sub-sampling, or **pooling**, is used to reduce data dimensionality while retaining relevant information.
Types of pooling:

- **Max Pooling:** Selects the maximum value in a specific region (e.g., a 2×2 subset of the input matrix). It preserves dominant features such as edges and contours.
- **Average Pooling:** Computes the average value in a region, providing a more generalized feature extraction.

Logical purpose of pooling:

- **Dimensionality reduction:** Decreases the number of parameters and computations, reducing overfitting.
- **Translation invariance:** Makes the model robust to small shifts or variations in the object's position within the input (e.g., an object slightly shifted in an image).

2.3.4 Batch Normalization

Batch normalization normalizes the outputs of a layer within a batch during training. This ensures that the activations in the batch have a distribution with zero mean and unit variance, adjusted by learned parameters.
The formula for batch normalization is:

$$\text{BN}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta \tag{8}$$

Where:

- μ: Mean of the batch.
- σ: Standard deviation of the batch.
- γ and β: Trainable parameters that scale and shift the normalized values.

Purpose of batch normalization:

- Improves training stability and speed by reducing oscillations in optimization.
- Mitigates the problem of **covariate shift**, where data distributions change during training, destabilizing gradients.

2.3.5 Uniform Spacing (Equispacing)

Uniform spacing in convolutional layers refers to the regular application of operations (convolution, pooling) across the input matrix, adhering to a grid defined by consistent strides and kernel sizes.
Motivations for uniform spacing:

- **Consistency in computation:** Ensures that all input positions are processed evenly, avoiding spatial bias.
- **Computational efficiency:** Uniform operations are optimized for modern hardware such as GPUs.
- **Ease of design:** Regular structures are easier to analyze and implement.

Example: VGG16 The **VGG16** architecture exemplifies uniform spacing:

- Each convolutional stack uses 3×3 kernels with consistent stride, ensuring uniform processing.
- After two or three convolutions, a **max pooling** operation with a 2×2 kernel and stride 2 reduces dimensionality.

This repetitive and regular structure makes VGG16 simple yet powerful, promoting a consistent hierarchy of feature extraction.
In conclusion, deep networks leverage linear transformations (filtering) and non-linearities (activations) to model complex relationships. Techniques such as pooling, batch normalization, and uniform spacing contribute to their robustness, efficiency, and scalability, as evidenced by architectures like VGG16.

2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks represent a class of deep learning models specifically used for processing visual data. They have demonstrated remarkable performance in various computer vision tasks, ranging from image classification to object detection and segmentation.

An example of CNN application in the real world includes image classification tasks, where CNNs can accurately categorize images into predefined classes. For instance, in medical imaging, CNNs have been employed to diagnose diseases based on X-ray or MRI scans, showcasing their potential in critical healthcare applications.

Key components of CNNs include convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply learnable filters to input data, capturing local patterns and features. Pooling layers reduce spatial dimensions, preserving important features while reducing computational complexity. Fully connected layers integrate extracted features for classification or regression tasks. By leveraging these components, CNNs can effectively learn representations of image's input data, giving accurate predictions across various domains.

2.4.1 Convolutional Layers

Convolutional layers are fundamental components of Convolutional Neural Networks. They play a crucial role in feature extraction by applying convolution operations to input images using filters.

- **Filters**: Filters, also known as kernels, are small matrices applied to input images during convolution. Each filter extracts specific features from the input by performing element-wise multiplication and summation operations.
- **Kernel Size**: The kernel size refers to the spatial dimensions of the filter. It determines the receptive field of the filter and influences the types of features extracted. Common kernel sizes include 3x3 and 5x5.
- **Strides**: Strides determine the step size of the filter as it traverses the input image during convolution. A stride of 1 means the filter moves one pixel at a time, while larger strides result in downsampling of the output feature map.
- **Padding**: Padding is the process of adding additional border pixels to the input image before convolution. It helps preserve spatial information and prevent loss of information at the image boundaries. Common padding types include 'valid', which applies no padding, and 'same', which pads the input to ensure the output feature map has the same spatial dimensions as the input.
- **Activation Function**: The activation function introduces non-linearity into the convolutional layer's output. Common activation functions include ReLU (Rectified Linear Unit), which introduces sparsity and addresses the vanishing gradient problem.

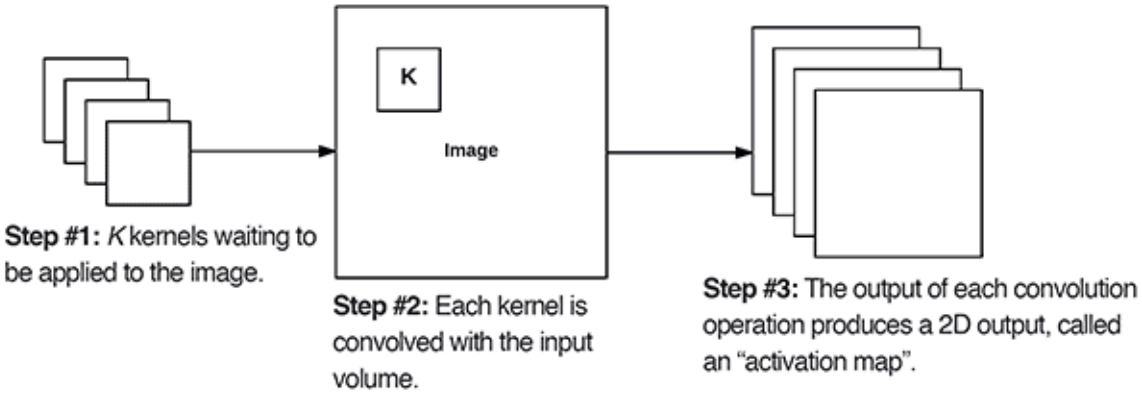

Figure 4: Illustration of a Convolutional Neural Network architecture.

2.4.2 Pooling Layers

Pooling layers are essential components in Convolutional Neural Networks used for down-sampling and feature reduction. They help reduce the spatial dimensions of the input feature maps, making the network more computationally efficient and reducing overfitting.

Pooling operations typically fall into three main categories:

- **Max Pooling**: Max pooling selects the maximum value from each subregion of the input feature map. It identifies the most significant features while discarding less important ones. Max pooling is the most common type of pooling operation used in CNNs due to its simplicity and effectiveness.
- **Average Pooling**: Average pooling computes the average value from each subregion of the input feature map. It provides a smoothed representation of the features and is less prone to overfitting compared to max pooling.
- **Global Pooling**: Global pooling computes a single value for each feature map by applying a pooling operation across the entire map. It reduces the spatial dimensions to a single value per feature map, often used as the input to the final classification layer of the network.

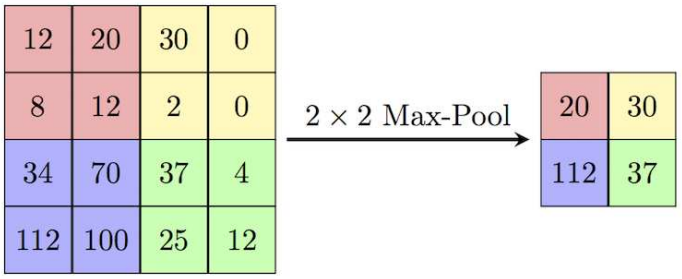

Figure 5: Illustration of Max Pooling (2x2) performed on a matrix (4x4) of integers.

2.4.3 Fully Connected Layers

Fully connected layers, also known as dense layers, integrate the features extracted by convolutional and pooling layers for final classification or regression tasks. Dense layers connect every neuron in one layer to every neuron in the next layer, enabling global information propagation through the network.
In Keras, fully connected layers are implemented using the `Dense` layer. These layers require input data to be in the form of a one-dimensional vector. However, the output of convolutional and pooling layers is typically a three-dimensional tensor. Therefore, before passing the output to the dense layers, the tensor is flattened into a one-dimensional vector using the `Flatten` layer.
The `Flatten` layer serves the purpose of reshaping the output of the preceding convolutional and pooling layers into a format suitable for input to the dense layers. It collapses the spatial dimensions of the feature maps into a single vector while preserving the relationships between the features.

2.5 *Chapter 2: Summary and Insights*

Artificial Neural Networks *(ANNs) are made up of layers of neurons that transform data linearly and then apply non-linear functions (such as ReLU, Sigmoid, or Tanh). The cost function calculates how much the network's predictions differ from the correct values, and techniques like backpropagation and gradient descent are used to minimize this error.*

Deep networks *learn to recognize increasingly complex information: the first layers detect simple elements (e.g., edges), while the subsequent layers combine them into more complex structures. Tools like pooling (e.g., max pooling) reduce the amount of data to process, while still keeping important information, and batch normalization*

makes training more stable and faster.

Convolutional Neural Networks *(CNNs) are perfect for working with images. Convolutional layers use small filters to detect local details, while pooling layers simplify the data without losing too much detail. Fully connected layers combine everything to produce the final output. Architectures like VGG16 show how powerful and useful these networks are, for example in medical diagnosis or object recognition.*

Having a basic understanding of these models is essential to fully grasp the concepts explained in the thesis and to follow the practical examples more easily.

Chapter 3

COVID-19

3.1 Use of Machine Learning during the COVID-19 Pandemic

COVID-19 was an unprecedented global event that quickly mobilized scientific research, including advanced techniques like *machine learning* (ML) to tackle the challenges posed by the virus. Even in the early phases of the pandemic, ML models were employed to track virus spread, forecast peak infection rates, identify mutations, and develop rapid diagnostic tools. Below are described the main stages of ML application and the types of data used.

3.1.1 Stages of Machine Learning Utilization

- **Early Pandemic**:
  - *Tracking and Spread Prediction*: ML models monitored virus spread and predicted infection peaks, using epidemiological and mobility data to help governments and health organizations plan containment measures.
  - *Symptom Analysis and Screening*: ML models were developed to differentiate COVID-19 symptoms from other respiratory illnesses, improving the screening process.
  - *Image-based Diagnosis*: Imaging data, like X-rays and computed tomography (CT) scans, were used to train models capable of identifying COVID-19 pneumonia, distinguishing it from other pathologies. Convolutional Neural Networks (CNNs) helped radiologists recognize signs of COVID in patients, reducing diagnostic time.
- **Later Stages of the Pandemic**:

  - *Vaccine and Drug Development*: ML models accelerated drug discovery and vaccine design by analyzing interactions between viral proteins and drugs, identifying effective therapeutic candidates.
  - *Genomic Sequence Analysis*: As the virus evolved, ML enabled the analysis of large genomic datasets, identifying critical mutations and predicting the spread of new variants.
  - *Optimization of Healthcare Resources*: Predictive models were used to manage hospital resources, forecasting the need for intensive care and other essential equipment based on patient clinical data.

3.1.2 Types of Data Used

- **Epidemiological Data**: Includes numbers of confirmed cases, recoveries, and deaths, collected globally. These data allowed the construction of predictive models on the pandemic's progression.
- **Clinical Data from Patients**: Medical records and demographic data, such as age, gender, pre-existing health conditions, and observed symptoms, helped understand risk factors and develop tools for assessing disease severity.
- **Medical Imaging Data**: Patient X-rays and CT scans were used to train algorithms capable of supporting doctors in diagnosing COVID-19 pneumonia.
- **Genomic Data**: Genomic sequences of SARS-CoV-2 were analyzed to identify variants and understand virus evolution.
- **Mobility and Social Media Data**: These data were used to monitor population behavior, understand virus spread dynamics, and detect early outbreak hotspots.

The use of machine learning during and after the pandemic demonstrated the ability of these tools to rapidly analyze large amounts of data, supporting the medical community and institutions in clinical and public health decisions.

3.2 X-rays in the Context of COVID-19

X-rays, or radiographs, are diagnostic imaging techniques based on the use of electromagnetic radiation, usually X-rays, to visualize the body's internal structures. In respiratory contexts, chest X-rays allow observation of the lungs, heart, and thoracic structures to detect abnormalities such as infections, inflammations, or lesions. However, while useful for preliminary diagnosis, X-rays do not always provide a clear

visualization of lung pathologies like COVID-19 or other pneumonias, as the information obtained is limited compared to other imaging techniques such as computed tomography (CT).

3.2.1 Examples of X-rays for COVID-19, Viral Pneumonia, and Normal Cases

Below are examples of X-rays for the three cases of interest: normal lungs, lungs affected by viral pneumonia, and lungs affected by COVID-19. Differentiating these conditions based solely on X-rays is challenging, as the images provide a two-dimensional view and are less detailed than CT scans.

Figure 6: Examples of X-rays: (a) Normal case, (b) Viral pneumonia, (c) COVID-19

3.2.2 Characteristics of X-rays for COVID-19, Viral Pneumonia, and Normal Cases

- **Normal Case**: In normal chest X-rays, the lungs appear clear and homogeneous, without signs of opacity or consolidations. Lung structures, such as bronchi and blood vessels, are visible but show no significant abnormalities.
- **Viral Pneumonia**: In viral pneumonia X-rays, consolidations, opacities, and inflammation are often seen in specific lung areas. However, the appearance of viral pneumonia can vary depending on the infection's severity and location, making it difficult to differentiate from COVID-19 using only X-rays.
- **COVID-19**: X-rays of COVID-19 patients often show bilateral "ground-glass" opacities, typical of the disease, along with extensive lung inflammation. However, these opacities can also appear in cases of other lung infections, making it challenging for doctors to distinguish COVID-19 from other pathologies using only X-rays.

3.2.3 Limitations of X-rays and Advantages of CT Scans

The evaluation of X-rays in the case of lung infections, such as COVID-19, provides only an estimate. X-rays offer a two-dimensional image that, while allowing a preliminary diagnosis, does not provide a sufficient level of detail to ensure an accurate distinction between pneumonias of different origins.
In contrast, **computed tomography** (CT) is a more advanced imaging technique that provides three-dimensional images and detailed body sections. CT uses X-rays to produce high-resolution images, allowing physicians to view successive layers of the lungs, which enables more precise identification and localization of lesions.

3.3 Clinical Research During the Pandemic Emergency

During the COVID-19 pandemic emergency, medical experts conducted numerous studies to identify the distinctive features of COVID-19 pathology in radiological images, such as computed tomography (CT) scans and, to a lesser extent, chest X-rays (CXR). These studies highlighted a series of key characteristics that can help differentiate the disease from other pulmonary conditions. Below, we outline the main findings.

3.3.1 Main Characteristics

The most common radiological abnormalities observed in CT scans of patients with COVID-19 include concentrations of inflammation or lacerations of the following types:

- **Bilateral**: Alterations are often present in both lungs, suggesting a symmetrical spread of the disease.
- **Subpleural or peripheral distribution**: Lesions tend to localize in the peripheral areas of the lung, near the pleura.
- **Multiple locations**: Radiological abnormalities manifest in several areas of the pulmonary parenchyma.
- **Posterior regions**: Lesions are frequently observed in the posterior regions of the lungs.
- **Multilobar with a preference for the lower lobes**: The disease typically affects multiple lung lobes, with a preference for the lower ones.

- **Symmetrical distribution**: The symmetry of the lesions is a frequent characteristic in the images of COVID-19 patients.

3.3.2 Specific Radiological Signs

Among the most specific signs observed in the radiological images of COVID-19 patients, we find:

- **Ground-glass opacities (GG)**: Zones of partial opacity that do not completely obscure the underlying blood vessels and bronchial structures. These indicate widespread pulmonary inflammation.
- **Crazy-paving (CP)**: A combination of ground-glass opacities and fibrotic streaks that form an appearance similar to an irregular pavement. This sign is typical of inflammation progression.
- **Parenchymal consolidations**: Pulmonary areas where the air has been replaced by fluid, inflammatory cells, or other materials.

For the work conducted, it is important to bear in mind some of these recurring patterns, which are often identified during the neural network classification experiments.

Although ground-glass opacities and crazy-paving patterns are much more significant in CT scans compared to CXR, the patterns of **bilateral** lacerations, **symmetrical** distribution, and multiple scattered locations are extremely important for providing a visual confirmation of the behavior of the predictive model.

3.4 *Chapter 3: Summary and Insights*

*This work focuses on classifying chest X-ray images into three categories: **Normal**, **COVID-19**, and **Viral Pneumonia**. To perform this task accurately, it is essential to understand the key features that distinguish COVID-19 from other conditions. For example, COVID-19 often shows specific signs such as **ground-glass opacities** and crazy-paving patterns in the lungs. Another key distinction is the spatial distribution of inflammations. In the case of COVID, they are often **bilateral** with a central concentration, whereas in viral pneumonia, they are often scattered, asymmetric, and exhibit irregular patterns. These characteristics help differentiate COVID-19 from other types of pneumonia, which can look similar on X-rays. Recognizing these patterns is crucial for the machine learning model to make correct predictions. Understanding these features is fundamental to analyzing the model's predictions and identifying any mistakes.*

Chapter 4

Model Explainability frameworks

As machine learning models become more complex, understanding their decision-making process is increasingly important. Model explainability tools like Grad-CAM, LIME and SHAP offer ways to interpret these "black-box" models. Grad-CAM provides visual explanations by highlighting image regions that influence predictions, while LIME and SHAP create simplified, interpretable models to explain individual predictions in a local space. VGG16 is a well-regarded convolutional neural network (CNN) architecture for image classification tasks and serves as the chosen model to drive the experiments in this work.

4.1 VGG16: Architecture and Use

VGG16 is a convolutional neural network (CNN) architecture developed by the *Visual Geometry Group* (VGG), the engineering sciences department at the University of Oxford. The network was presented at the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* in 2014. VGG16 is a widely recognized and utilized model for Object Recognition tasks due to its efficiency and relatively shallow depth (only 13 convolutional layers).

Imagenet: ImageNet is a large-scale dataset of images primarily used for training and evaluating computer vision models. It was created to support research in machine learning and artificial intelligence, particularly in image classification and object recognition. In 2012, the AlexNet model, trained on ImageNet, marked a significant breakthrough by drastically reducing classification errors, thus launching a new era for deep learning applications in computer vision.

4.1.1 Structure of VGG16

VGG16 is named as such because it has 16 layers in depth, including 13 convolutional layers and 3 fully connected layers. The structure follows a simple and modular design, with fixed 3x3 convolutional filters applied at different input resolutions. Each convolutional layer is followed by a max-pooling operation, which reduces the spatial size of the activation maps while preserving relevant features for classification.

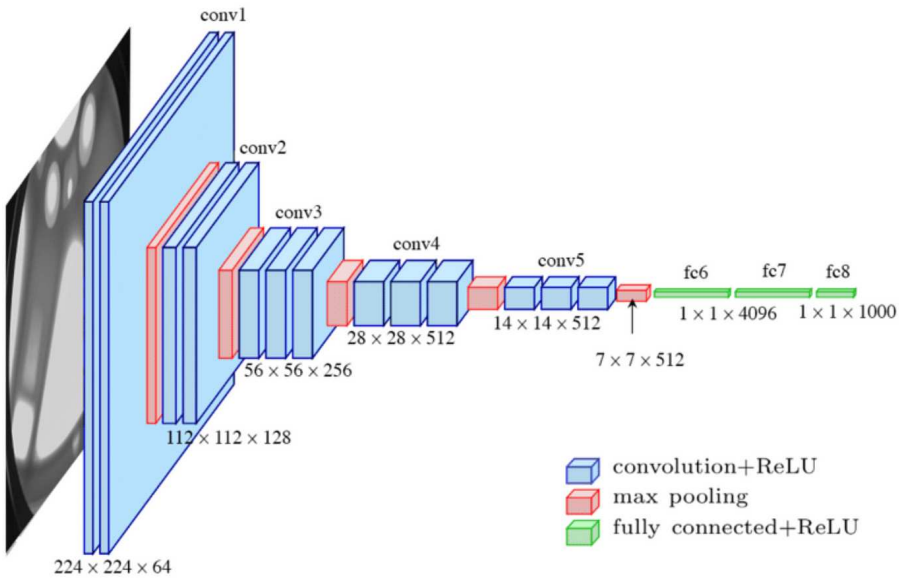

Figure 7: VGG16 Architecture

1. **First Stack:**
   - 64 two-dimensional convolutional filters of size 3x3 are applied, with stride 1 and padding 1.
   - The output, with dimensions 224x224x64, is then processed by a second identical convolutional filter.
   - The output dimensions remain unchanged, but are reduced via a two-dimensional *max pooling layer*, resulting in a final volume of 112x112x64.
2. **Second Stack:**
   - The number of convolutional filters doubles: 128 two-dimensional filters of size 3x3 are applied, resulting in an initial volume of 112x112x128.
   - A second convolutional filter of the same size is then applied.

- Finally, a *max pooling layer* reduces the final volume to 56x56x128.

3. **Third Stack:**
   - On the input from the previous stack, 256 two-dimensional convolutional filters of size 3x3 are applied.
   - This operation is repeated three times.
   - A *max pooling layer* is finally used to reduce the volume to 28x28x256.
4. **Fourth Stack:**
   - 512 two-dimensional convolutional filters of size 3x3 are applied three times in succession.
   - A *max pooling layer* then reduces the final volume to 14x14x512.
5. **Fifth Stack:**
   - Similar to the fourth stack: three convolutional layers and a *max pooling layer* reduce the output to 7x7x512.
6. **Sixth Stack:**
   - In the final part, the first *fully connected layer* flattens the 7x7x512 volume into a vector of 25,088 neurons.
   - This is connected to a *dense layer* of 4096 neurons, followed by a *dropout layer* with the same number of neurons.
7. **Seventh Stack:**
   - There is a second *dense layer* of 4096 neurons, followed by another *dropout layer*.
8. **Eighth Stack:**
   - The architecture is completed with an *output layer* of type *dense*, consisting of 1000 neurons for the final classification.

4.1.2 Use of VGG16

VGG16 is widely used for image classification, object recognition, and segmentation tasks. Its deep yet simple architecture makes it one of the easiest neural networks to understand and implement, while still being very effective for various deep learning applications. Despite the increased complexity of modern networks, VGG16 remains a benchmark model, often used as a base for transfer learning in various computer vision problems.

4.2 Grad-CAM: Gradient-Weighted Class Activation Mapping

Grad-CAM (Gradient-weighted Class Activation Mapping) is a visualization technique used to interpret and explain the decisions of a convolutional neural network (CNN). Grad-CAM relies on calculating the output class gradients with respect to the feature maps of a convolutional layer in order to generate activation maps that highlight the relevant regions of the image for a specific prediction.
Consider an example of applying Grad-CAM on a model trained to recognize images of animals.

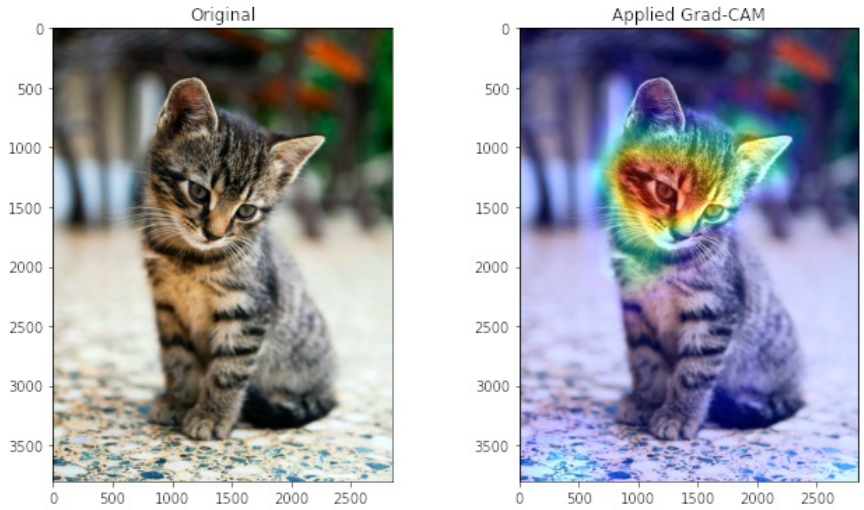

Figure 8: Left: input image (cat). Right: Grad-CAM activation map overlaid.

The model correctly predicts "cat" with a 90% probability. The Grad-CAM activation map primarily highlights the cat's head, indicating that this region contributed the most to the prediction.

4.2.1 How Grad-CAM Works

The Grad-CAM method utilizes the gradients of the feature maps with respect to the prediction of a target class to understand which regions of the image most influence the neural network's final decision. The process can be broken down into the following steps:

1. **Forward the input image through the CNN:** The input image is fed into the model (forward pass), and the model calculates the prediction. The series of

convolutional layers, regularized by the ReLU activation function, each produce a Rectified Convolutional Feature Map A. The feature map generated will be processed by the Fully Connected (FC) or Dense Layers, which will output a classification score y for the respective class c.

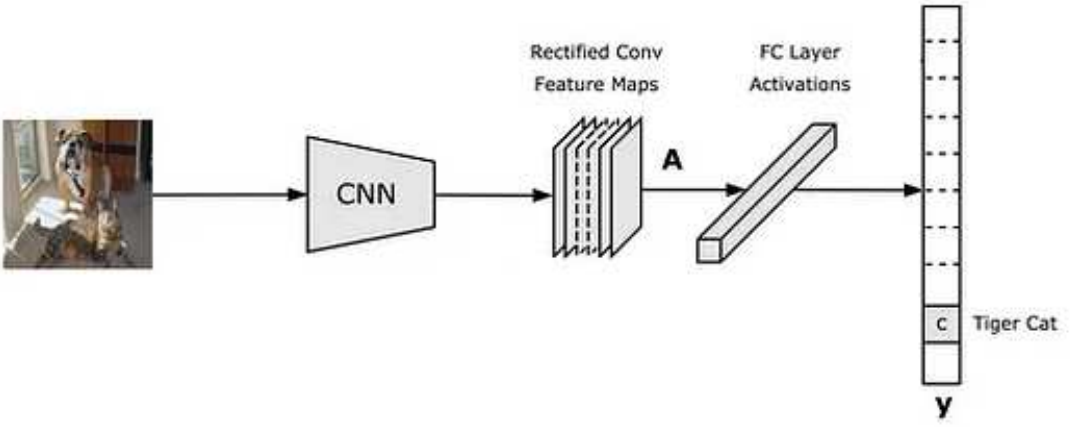

Figure 9: Forward Pass of the CNN. A represents the rectified convolutional feature maps matrix, FC Layer activations are the outputs of the set of fully connected layers, and y is the classifier's output corresponding to a specific class c.

2. **Calculate the gradients:** Gradients of the target class prediction are calculated as the partial derivative of the classification score y with respect to each feature map A^k of an intermediate convolutional layer L. These gradients $\frac{\partial y^c}{\partial A^k}$ represent how much the output of that class would change if a specific feature map was slightly modified. A higher gradient means greater importance of that feature map for the prediction.

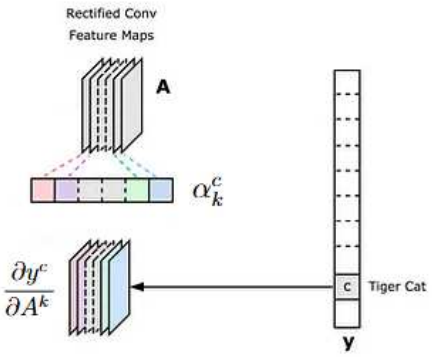

Figure 10: Backward Pass of the CNN. y is the classification score related to the class c. $\frac{\partial y^c}{\partial A^k}$ is the partial derivative of y with respect to the feature map A^k

3. **Weight the feature maps:** Once obtained the gradients $\frac{\partial y^c}{\partial A^k}$, the feature maps are weighted according to their calculated importance based on the averaged gradients across each map. This operation generates the coefficients α_k^c, which represent the weight of the k-th feature map A^k for the target class c.

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{9}$$

4. **Create the activation map:** The weighted feature maps are summed to obtain a combined activation map. This map visually represents the regions of the image that most influenced the target class prediction. Only positive regions are retained by applying the ReLU function, which removes negative values, preserving only activations that contributed positively.

$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{10}$$

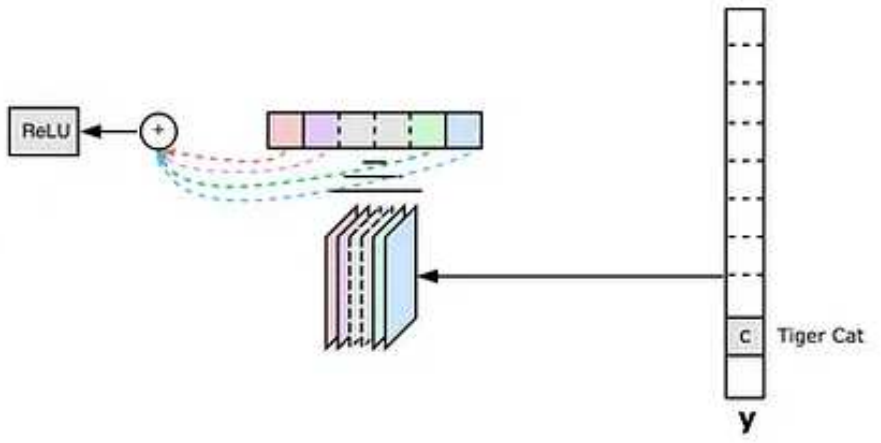

Figure 11: Overall image of the backpropagation: y is the classification score related to the class c. $\frac{\partial y^c}{\partial A^k}$ is the partial derivative of y with respect to the feature map A^k

5. **Visualize the Grad-CAM map:** The activation map is overlaid on the original image to produce an interpretable visualization. The regions highlighted in red correspond to the parts of the image that most influenced the model's prediction.

Algorithm 1 Grad-CAM Algorithm

Data: Input image x, trained CNN model y, target class c, convolutional layer L
Result: Grad-CAM activation map $L^c_{\text{Grad-CAM}}$ for class c
1. Perform a forward pass to obtain the output $y(x)$ and intermediate activations of layer L
2. Compute the gradients $\frac{\partial y^c}{\partial A^k}$ of the class score $y^c(x)$ with respect to the feature maps A^k of layer L
3. Calculate the importance weights $\alpha_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}}$ with respect to activation A^k at position (i, j)
4. Combine the feature maps weighted by α_k^c to produce $L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$
5. Overlay $L^c_{\text{Grad-CAM}}$ on the original image x for visualization

---

4.2.2 Grad-CAM Formula

The mathematical formula of Grad-CAM is : (10)

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha^c_k A^k\right)$$

Where:

- $L^c_{\text{Grad-CAM}}$ is the weighted activation map for class c.
- A^k is the k-th feature map of the selected convolutional layer.
- α^c_k is the coefficient measuring the importance of feature map A^k for class c.

The coefficient α^c_k is calculated as: (9)

$$\alpha^c_k = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}}$$

Where:

- $\frac{\partial y^c}{\partial A^k_{ij}}$ is the gradient of the class c prediction with respect to activation A^k at position (i, j).
- Z is the total number of elements in the feature map A^k, i.e., the product of height and width: $Z = H \times W$.

This formula indicates that the contribution of each feature map A^k is weighted based on the spatially averaged gradients relative to class c. Finally, the ReLU function is applied to remove negative values, leaving only regions with positive influence.

4.2.3 Example of Grad-CAM

Activation Map and Predictions: Let's assume we have an activation map A generated by a convolutional layer with dimensions $2 \times 2 \times 2$ (height, width, channels). The network predicts the class *dog* with a probability of 0.85 and the class *cat* with a probability of 0.15.
The activation map is as follows:

$$A = \begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix} \quad \begin{bmatrix} 0.5 & 1.5 \\ 2.5 & 3.5 \end{bmatrix}$$

where the two blocks represent the activations in channels 1 and 2. Thus, we have an activation map of size $2 \times 2 \times 2$:

- $A_{11} = 1.0, A_{12} = 2.0, A_{21} = 3.0, A_{22} = 4.0$ (channel 1)
- $A_{11} = 0.5, A_{12} = 1.5, A_{21} = 2.5, A_{22} = 3.5$ (channel 2)

Gradient of the Prediction with Respect to the Activations: Suppose the gradient of the *dog* prediction with respect to the activation map is as follows:

$$\frac{\partial p_{\text{dog}}}{\partial A} = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} \quad \begin{bmatrix} 0.05 & 0.15 \\ 0.25 & 0.35 \end{bmatrix}$$

Each value represents the derivative of the *dog* probability with respect to a specific activation in the map.

Calculate the Global Weights α_c: Next, we calculate the global weights α_c with the formula (9) for each channel by averaging the gradient for each channel.
For channel 1:

$$\alpha_1 = \frac{1}{4}(0.1 + 0.2 + 0.3 + 0.4) = \frac{1}{4} \times 1.0 = 0.25$$

For channel 2:

$$\alpha_2 = \frac{1}{4}(0.05 + 0.15 + 0.25 + 0.35) = \frac{1}{4} \times 0.8 = 0.2$$

Calculate the Weighted Activation Map: Now, we multiply each value in the activation map by the corresponding weight.
For channel 1:

$$L^1 = \begin{bmatrix} 0.25 \times 1.0 & 0.25 \times 2.0 \\ 0.25 \times 3.0 & 0.25 \times 4.0 \end{bmatrix} = \begin{bmatrix} 0.25 & 0.5 \\ 0.75 & 1.0 \end{bmatrix}$$

For channel 2:

$$L^2 = \begin{bmatrix} 0.2 \times 0.5 & 0.2 \times 1.5 \\ 0.2 \times 2.5 & 0.2 \times 3.5 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.3 \\ 0.5 & 0.7 \end{bmatrix}$$

Sum the Weighted Activation Maps: We sum the weighted maps according to (10) to obtain the final map:

$$L = L^1 + L^2 = \begin{bmatrix} 0.25 + 0.1 & 0.5 + 0.3 \\ 0.75 + 0.5 & 1.0 + 0.7 \end{bmatrix} = \begin{bmatrix} 0.35 & 0.8 \\ 1.25 & 1.7 \end{bmatrix}$$

Apply the ReLU Function: Finally, we apply the ReLU (7) function to remove negative values and obtain the final activation map:

$$L' = \max(0, L) = \begin{bmatrix} 0.35 & 0.8 \\ 1.25 & 1.7 \end{bmatrix}$$

Visualize the Activation Map: The final map L' represents the importance of each pixel for the *dog* prediction. The higher the value in the map, the more that region of the image contributed to the network's prediction for the *dog* class.
In a practical application, this map would be resized to match the original image, and then visualized to highlight the areas of highest importance for the prediction.

4.2.4 Difference Between Gradients in Grad-CAM and Training

The gradients calculated in Grad-CAM and those used during the training of a convolutional neural network serve different purposes and should not be confused.

1. **Gradients during training:** During the training of a neural network, gradients are used to update the model's weights through backpropagation. These gradients indicate how much each weight of the network should be adjusted to reduce the total error, i.e., the difference between the model's predictions and the correct labels. The weights are updated in the opposite direction of the gradients to minimize the cost function.

2. **Gradients in Grad-CAM:** In Grad-CAM, gradients are not used to update the model's weights but to identify the importance of the *feature maps* relative to the prediction of a specific class. The gradients are calculated with respect to the activations of a certain convolutional layer and the output of a class. They serve to visually localize the areas of the image that most influence the model's prediction for that class.

4.3 Grad-CAM Intralayer

In the context of Grad-CAM, the term Grad-CAM "Intralayer" refers to applying the Grad-CAM method not only to the last convolutional layer of a neural network but also to a subset of intermediate convolutional layers within the network. This approach allows for a more detailed analysis of how the network processes information at different depths, offering insights into the features and patterns captured by various internal layers. By visualizing activations at these intermediate layers, it becomes possible to trace the progression of decision-making and understand the role of specific features in the classification process.

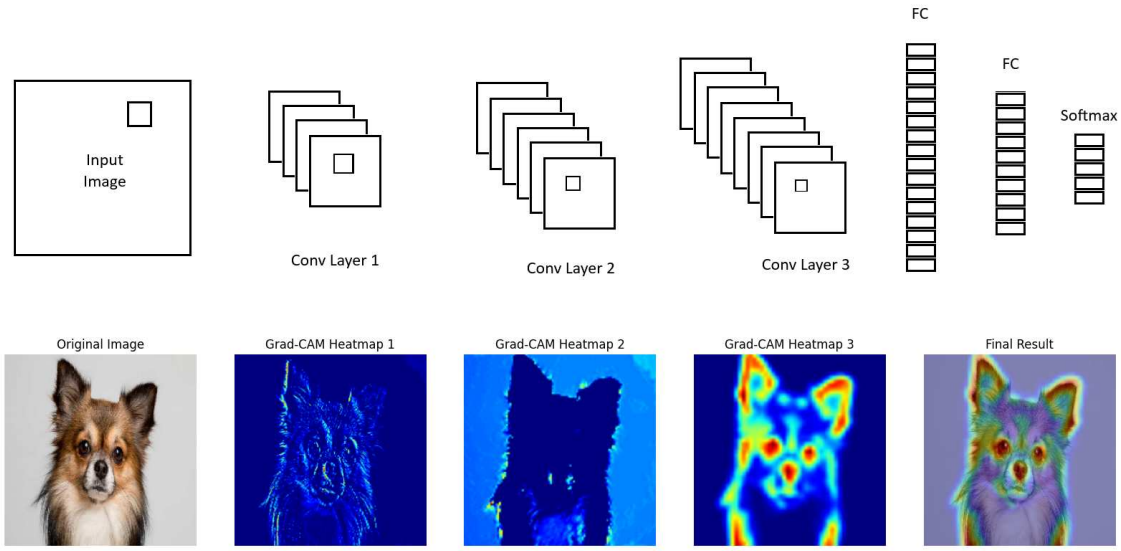

Figure 12: A CNN with three convolutional layers, each producing a Grad-CAM heatmap. The heatmaps highlight key regions of the input image: broad features at Layer 1, intermediate textures at Layer 2, and class-specific details at Layer 3.

To compute Grad-CAM for all convolutional layers of a neural network and return the set of resulting heatmaps, we define the following formula:

$$\mathcal{L}_{\text{Grad-CAM}} = \left\{ \text{ReLU}\left(\sum_k \alpha_k^{c,l} A^{k,l} \right) \,\middle|\, l \in \{1, 2, \ldots, L\} \right\} \tag{11}$$

Where:

- $\mathcal{L}_{\text{Grad-CAM}}$ is the set of Grad-CAM heatmaps generated for all convolutional layers.
- L is the total number of convolutional layers in the neural network.
- $A^{k,l}$ represents the k-th activation map of the l-th layer.
- $\alpha_k^{c,l}$ is the weight for the k-th activation map in the l-th layer, calculated as (9)
- ReLU is the rectified linear unit function, defined as (7).

The result, $\mathcal{L}_{\text{Grad-CAM}}$, is a collection of heatmaps generated for each convolutional layer of the network. These heatmaps can be overlaid on the input image to visualize important regions at different layers of the network.

4.3.1 Advantages of Grad-CAM Intralayer

- **Greater interpretability**: Grad-CAM Intralayer offers a detailed view of activations at various levels of the network, enabling a more granular understanding of the model's behavior.
- **In-depth diagnosis of the classification process**: This method allows for visualizing not only where the model made the final prediction but also at which intermediate layers it began to make decisions. This is particularly useful for identifying where the model may start making errors. For example, if activations in deeper layers begin focusing on irrelevant areas of the image, we can identify where the model "goes wrong" in the classification process.
- **Analysis of intermediate features**: It allows for visualizing the importance of features extracted in intermediate layers, which may capture relevant intermediate representations such as edges, textures, or shapes.

4.4 LIME Method

The LIME (Local Interpretable Model-agnostic Explanations) method was introduced in 2016 by Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. It was designed as a tool to improve the understanding and interpretability of complex machine learning models, particularly "black-box" models such as deep neural networks. The goal of LIME is to provide interpretable explanations for predictions, allowing users to understand which features influenced a specific decision.

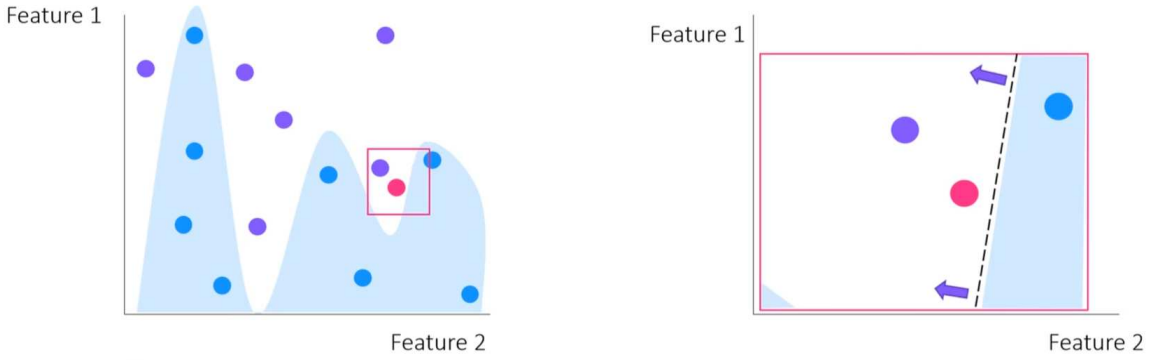

Figure 13: Complex Model (left): A nonlinear decision boundary separates blue and violet points based on two features. Linear Model (right): In the local region around the instance, points are classified, and a linear decision boundary approximates the complex model's behavior for that instance.

4.4.1 LIME Formula

The mathematical formula of LIME is

$$\xi(x) = \arg\min_{g \in G} \left(L(f, g, \pi_x) + \Omega(g) \right) \tag{12}$$

- $\xi(x)$: The explanation for the specific instance x. It is the interpretable model g that best approximates the behavior of the complex model f locally.
- $\arg\min_{g \in G}$: Indicates that we are searching for the interpretable model g within the set of interpretable models G (e.g., linear regression or decision trees) that minimizes the objective function.
- $L(f, g, \pi_x)$: A loss function that measures how well the interpretable model g approximates the predictions of the complex model f in the local region around x, weighted by π_x. The kernel π_x assigns higher weights to points closer to x.
- $\Omega(g)$: A regularization term that ensures the simplicity and interpretability of g. For instance, it may limit the number of features used or constrain the structure of the model.
- G: The set of interpretable models, such as linear models or shallow decision trees.

Objective: The formula selects an interpretable model g that balances two goals:

1. Minimize the difference between the complex model f and the interpretable model g in the local region around x.
2. Ensure that g remains simple and easy to understand.

4.4.2 Explanation of the LIME Process

The LIME method involves the following steps to generate an interpretable explanation for a complex model's prediction:

1. **Generation of new data points in the neighborhood:**
   For a given input data point x, we generate new perturbed data points in its neighborhood:
   $$z_i \sim \mathcal{N}(\mu_x, \sigma_x), \quad \forall i \in [1, N] \tag{13}$$
   where z_i are the perturbed points, N is the number of generated points, $\mathcal{N}$ represents a normal (or Gaussian) distribution and μ_x, σ_x are the mean and

standard deviation for each feature of x. A proximity kernel π_x assigns higher weights to points closer to x.

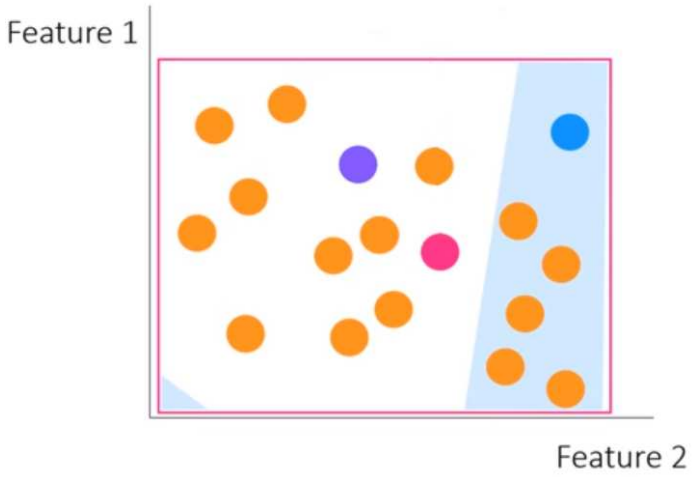

Figure 14: Linear function in the approximated local space. Blue and violet points are classified in two different classes. The pink point represent our input data. The orange ones are the points generated by perturbations

2. **Classification of perturbed points with the complex model f:**
   The generated points z_i are passed through the complex model f to obtain predictions (labels). - Points on one side of the linear separator are labeled as "No", while points on the other side are labeled as "Yes". - This results in a new training dataset (Z, Y), where:

$$Z = \{z_i\}_{i=1}^{N}, \quad Y = \{f(z_i)\}_{i=1}^{N}$$

3. **Training an interpretable model g:**
   Using the new dataset (Z, Y), an interpretable model g (e.g., linear regression or decision tree) is trained by minimizing a combined loss function: (12)

$$\xi(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

   where:

- $L(f, g, \pi_x)$ represents the local fidelity loss:

$$L(f, g, \pi_x) = \sum_{i=1}^{N} \pi_x(z_i) \cdot \ell(f(z_i), g(z_i)) \tag{14}$$

  Here, $\ell(\cdot, \cdot)$ is an error metric, such as mean squared error or cross-entropy.
- $\Omega(g)$ is a regularization term ensuring g remains simple and interpretable.

4. **Final result:**
   The trained interpretable model g provides a localized explanation of f's behavior around x. It highlights the most influential features that impacted $f(x)$, enabling a better understanding of the complex model.

Algorithm 2 LIME Algorithm

Data: Complex model f, instance of interest x, number of perturbations N
Result: Interpretable model g for explaining $f(x)$
1. Generate N perturbations $z_i{}_{i=1}^{N}$ around x with (13);
2. Use the complex model f to calculate predictions $f(z_i)_{i=1}^{N}$;
3. Weight the perturbed instances based on their similarity to the instance of interest x with (14);
4. Train an interpretable model g (e.g., linear regression) on the perturbed instances with (12);
5. Return g, representing the local interpretation of f's prediction on x;

4.4.3 Example of LIME

Below is an example of LIME's functionality in a CNN classification case where we aim to determine if the image in question is a frog.

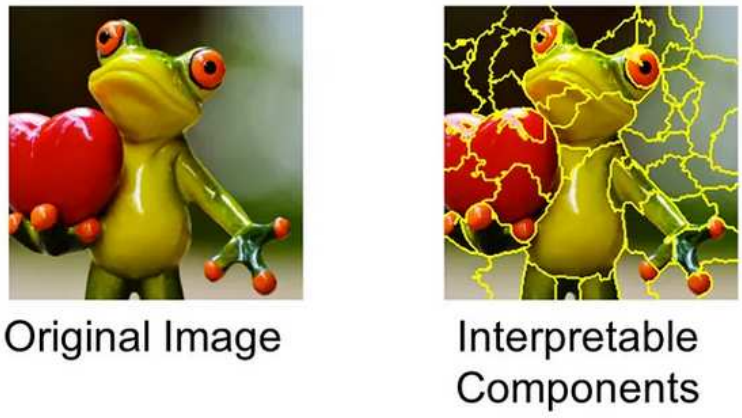

Figure 15: Partitioning the image into interpretable components.

In the left image, the original image (a frog on a colorful background) is divided into interpretable components, named **superpixels** as shown in the table on the right.

In LIME, **image segmentation** can be achieved in various ways, ranging from more systematic approaches, such as grid-based segmentation, to more sophisticated methods. The default method used by LIME for segmentation is *QuickShift*, which is a clustering-based approach. *QuickShift* operates by treating each pixel as a point in a feature space and clustering pixels based on their spatial proximity and color similarity. It iteratively shifts each point towards the mean of its neighboring points until convergence, forming regions that are similar to each other. These regions are then used as segments in the image.

Once the image is segmented into a set of super-pixels, LIME generates **perturbations** by randomly removing some of the super-pixels from the image. This removal can be done in two main ways:

- **Masking:** The selected super-pixels are darkened (e.g., by replacing the pixels with a fixed value, such as the mean value of the image or black pixels).
- **Elimination:** The super-pixels are completely removed, reducing the image size.

LIME then calculates the probability of classifying the image as "tree frog" for each perturbed version with the local fidelity loss formula(14).

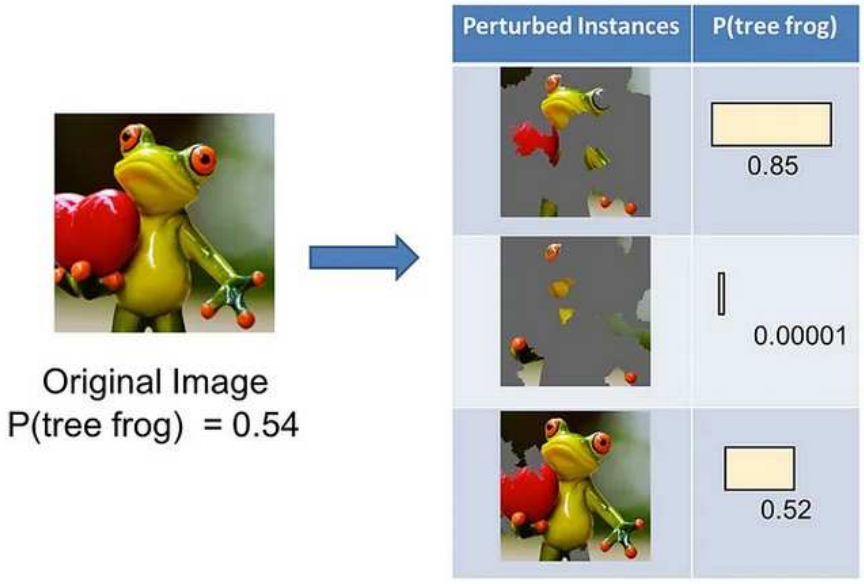

Figure 16: Examples of perturbed images with classification probabilities.

As shown, the probability varies depending on the masked parts. For example:

- When the frog and the fruit are visible, the probability of being classified as "tree frog" is high (0.85).
- When most of the frog and fruit are obscured, the probability drops drastically (0.00001).
- With only small modifications, the probability remains similar to the original classification (0.52).

LIME then builds a local interpretable model to determine which components most influence the classification. In this case, LIME finds that the presence of the frog (head and body) is a decisive factor for the high probability of classification as "tree frog."
Each segment of the image is weighted locally, and the local regression model assigns each segment a weight based on its influence on the classification probability of the "tree frog" class. The weighting and local regression process allow for visually identifying the most relevant areas of the image for classification.

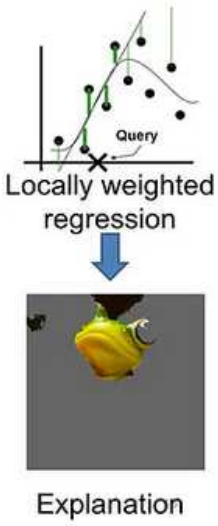

Figure 17: The final LIME result, highlighting the areas contributing most to the classification as "tree frog".

As shown, the final result of LIME is an image highlighting the critical areas for classification. In this case, LIME determined that the frog's head is the most influential part in the model's output, highlighting it as the main portion of the image that drives the "tree frog" prediction.

4.5 SHAP Method (SHapley Additive exPlanations)

The SHAP method (SHapley Additive exPlanations) is an interpretability technique proposed in 2017 by Scott Lundberg and Su-In Lee. SHAP is inspired by Shapley values from game theory, introduced by Lloyd Shapley, which allocate the "payout" among participants in a cooperative game based on their contribution to the final outcome. In machine learning, SHAP aims to quantify the contribution of each feature to a model's prediction, using a mathematical approach.
Shapley values are based on the idea that, for a given prediction, one can calculate the marginal effect of each feature. This involves calculating the average contribution of each feature to the prediction, considering all possible combinations of the other features. SHAP implements an efficient version of this calculation which would otherwise be computationally prohibitive with many features.

4.5.1 Shapley value Formula

The Shapley value formula can be written as:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \underbrace{\frac{|S|!\,(|N| - |S| - 1)!}{|N|!}}_{\text{Weight}} \cdot \underbrace{[v(S \cup \{i\}) - v(S)]}_{\text{Marginal Contribution}} \tag{15}$$

Explanation

1. **Weight**:

   - The fraction $\frac{|S|!\,(|N|-|S|-1)!}{|N|!}$ represents the weight assigned to the marginal contribution of player i to coalition S.
   - This weight is determined by the number of permutations of players:
     - $|S|!$: The number of ways to order the players in coalition S.
     - $(|N| - |S| - 1)!$: The number of ways to order the remaining players not in S and not including i.
     - $|N|!$: The total number of permutations of all players in N.

2. **Marginal Contribution**:

   - The term $v(S \cup \{i\}) - v(S)$ represents the marginal contribution of player i to coalition S.
   - It measures the increase in the value of the coalition S when player i joins it.

This distinction clarifies how the Shapley value assigns weights to marginal contributions by considering all possible coalitions and permutations of players.
The SHAP algorithm follows these steps:

Example: Shapley Value Calculation

Consider a cooperative game with three players: $N = \{1, 2, 3\}$. The characteristic function v is defined as follows:

$$\begin{aligned} v(\{1\}) &= 0, \\ v(\{2\}) &= 0, \\ v(\{3\}) &= 0, \\ v(\{1,2\}) &= 4, \\ v(\{1,3\}) &= 4, \\ v(\{2,3\}) &= 6, \\ v(\{1,2,3\}) &= 20. \end{aligned}$$

We calculate the Shapley value for player 1 ($\phi_1(v)$) by considering all subsets S of $N \setminus \{1\}$:

Step-by-Step Calculation

1. For $S = \emptyset$:
   - Marginal contribution: $v(\{1\}) - v(\emptyset) = 0 - 0 = 0$
   - Weight: $\frac{0!\cdot 2!}{3!} = \frac{1\cdot 2}{6} = \frac{1}{3}$
   - Term: $\frac{1}{3} \cdot 0 = 0$
2. For $S = \{2\}$:
   - Marginal contribution: $v(\{1,2\}) - v(\{2\}) = 4 - 0 = 4$
   - Weight: $\frac{1!\cdot 1!}{3!} = \frac{1\cdot 1}{6} = \frac{1}{6}$
   - Term: $\frac{1}{6} \cdot 4 = \frac{2}{3}$
3. For $S = \{3\}$:
   - Marginal contribution: $v(\{1,3\}) - v(\{3\}) = 4 - 0 = 4$
   - Weight: $\frac{1!\cdot 1!}{3!} = \frac{1}{6}$

- Term: $\frac{1}{6} \cdot 4 = \frac{2}{3}$

4. For $S = \{2, 3\}$:
   - Marginal contribution: $v(\{1, 2, 3\}) - v(\{2, 3\}) = 20 - 6 = 14$
   - Weight: $\frac{2!\cdot 0!}{3!} = \frac{2\cdot 1}{6} = \frac{1}{3}$
   - Term: $\frac{1}{3} \cdot 14 = \frac{14}{3}$

Final Calculation for $\phi_1(v)$ is obtained summing all terms:

$$\phi_1(v) = 0 + \frac{2}{3} + \frac{2}{3} + \frac{14}{3} = \frac{18}{3} = 6$$

Similarly, we compute the Shapley values for the other players:

$$\phi_2(v) = 7, \quad \phi_3(v) = 7$$

The total value of $v(N) = 20$ is fairly distributed among the three players as:

$$\phi(v) = (6, 7, 7)$$

4.5.2 Shap framework

The Shapley value formula used in the explainability of complex models is the same as the one used in game theory, adapted for the context of machine learning predictions. In this context, the model's features (input variables) are the "players" that contribute to the model's output.

The formula is the same of (15):

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left[f(S \cup \{x_i\}) - f(S) \right]$$

but with some variations:

1. $f(S)$: It is the model's output calculated considering only the features present in S. Missing features are treated as unavailable (often replaced with reference values such as the dataset's mean).
2. $f(S \cup \{x_i\}) - f(S)$: It is the marginal contribution of the feature x_i to the model's output when added to the subset S.

Algorithm 3 SHAP Algorithm

Data: Complex model f, instance of interest x, set of features F
Result: SHAP values for each feature of x
1. For each feature i, calculate the marginal contribution of the feature across all possible combinations of the other features
2. Average all marginal contributions to obtain the Shapley value of feature i
3. Repeat the process for all features of x
4. Return the SHAP values, which represent each feature's contribution to the prediction of f on x

4.5.3 SHAP on Regression

The following image shows SHAP values for various features of a regression model applied to a housing dataset. Each point in the diagram represents a single observation in the dataset. The labels on the left indicate different features used by the model, such as "working class," "number of rooms," and "NOX concentration." The horizontal axis shows the SHAP value, representing each feature's impact on the model's prediction.

- **Colors**: Points are colored based on the feature value itself: red points represent high feature values, while blue points represent low feature values.
- **Position**: The horizontal position of each point reflects the feature's effect on the model's prediction. Points to the right of zero indicate an increase in the prediction due to the feature, while points to the left indicate a decrease.

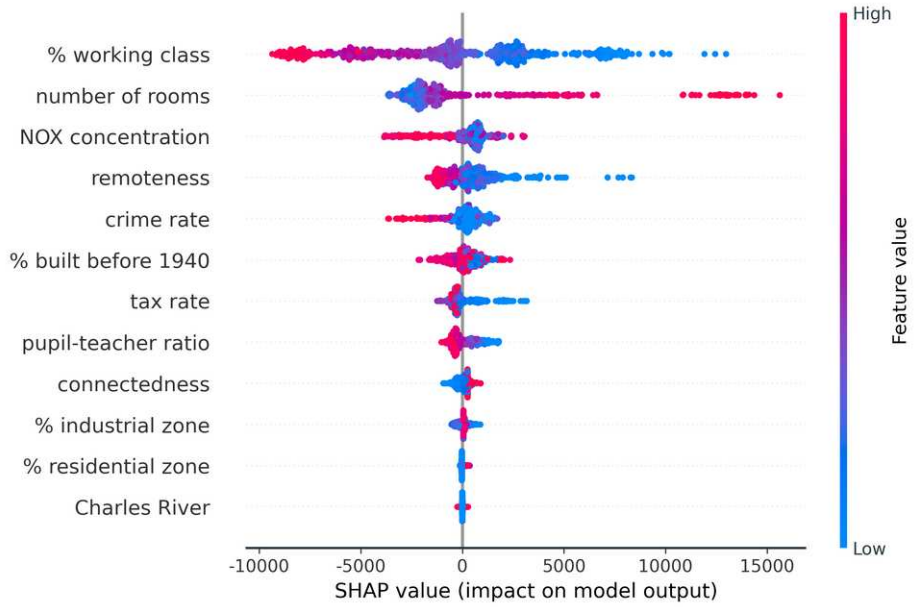

Figure 18: SHAP value plot for various features in a regression model.

For example, in the "working class" feature, high values (in red) tend to reduce the prediction, suggesting a negative effect on property value. Conversely, for the "number of rooms," higher values have a positive impact.

4.5.4 SHAP on Image Classification

The second image shows the application of SHAP on an image classification model. The images on the left represent the original inputs to the model (an American egret and a speedboat), while the images on the right show the contributions of various pixels for different output classes (e.g., "American egret" or "speedboat").

- **Colored Pixels**: Colors represent SHAP values for each pixel or region of the image. Red pixels have a positive effect on the probability of the indicated class (e.g., supporting classification as "American egret"), while blue pixels have a negative effect (reducing the probability for that class).
- **Mask Analysis**: Each class has an activation mask highlighting the relevant areas for classification. In this example, for the class "American egret," the pixels containing the bird's head and neck are colored red, indicating these areas contributed most to the classification as an egret. For the class "speedboat," the red is concentrated on the body of the boat, signaling that this area is crucial for identifying the object.

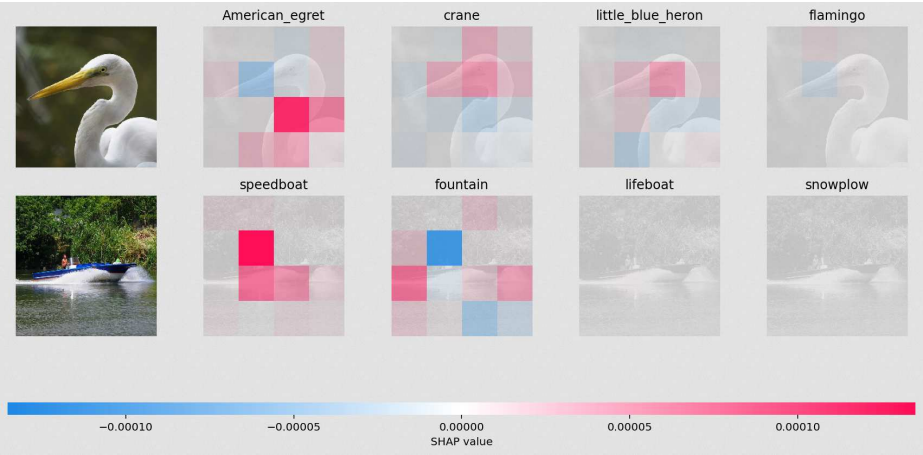

Figure 19: Visualization of SHAP applied for various image and various image classes.

This type of visualization provides an intuitive way to understand which parts of the image most influence the model's prediction, making the neural network's decision-making process more transparent and allowing users to verify the validity of the model's decisions.

SHAP is generally preferred over LIME as it provides more rigorous and consistent explanations; Shapley values ensure fair distribution of importance across features. However, LIME is faster to compute and more flexible, especially for complex models or when a quick, localized explanation is needed, making it useful in time-constrained scenarios.

4.6 *Chapter 4: Summary and Insights*

*The **VGG16** network was chosen for this project due to the simplicity of its structure and its effectiveness in image classification. It is characterized by small convolutional filters and a hierarchical layer design, offering a good balance between computational efficiency and performance. VGG16 is supported by extensive documentation and numerous studies validating its reliability. It provides a solid foundation for adapting to the specific task of medical image classification.*

*In the context of radiological images, frameworks for prediction analysis are essential to ensure transparency and explainability. Convolutional neural networks, despite their high predictive capabilities, often operate as a "**black box**," making their decision-making processes challenging to understand. For this reason, frameworks*

such as Grad-CAM, LIME, and SHAP were employed to provide visual and quantitative explanations of the model's behavior. This chapter explains the mathematical and logical theory behind the algorithms that power these frameworks. The results will be presented in Chapter 6.

Grad-CAM *(Gradient-weighted Class Activation Mapping) was used to visually identify the regions of greatest influence within the images. By calculating gradients in the final convolutional layer, it generates activation maps that highlight areas important for a specific class. In this project, Grad-CAM clearly revealed which regions of the lungs the model focuses on to differentiate between COVID-19, viral pneumonia, and normal images. Additionally, applying Grad-CAM to intermediate layers of VGG16 provided further insights into the classification process, offering a deeper understanding of the model's decision-making.*

LIME *(Local Interpretable Model-agnostic Explanations) was used to provide local explanations for predictions. This method analyzes the model's behavior by focusing on the importance of individual superpixels in the classification process. LIME was applied in this study to explore its potential in complementing the insights provided by Grad-CAM.*

SHAP *(SHapley Additive exPlanations) provides a comprehensive analysis by quantifying the positive and negative contributions of each superpixel to the model's output. The Shapley values represent the impact of each superpixel in the final classification output. Applied to images, SHAP generates visual maps where red indicates positive contributions, blue negative contributions, and white regions with no influence. In addition to these visual insights, SHAP also offers a numerical analysis, enabling a detailed comparison between COVID-19, viral pneumonia, and normal classes. This highlights differences and similarities in radiological patterns and ranks predictions based on the relevance of each feature.*

Chapter 5

Model Training

This chapter provides a detailed description of the setup employed for conducting the practical experiments. It outlines the dataset split, input normalization, as well as the types of callbacks and precautions established prior to training.

5.1 Dataset Modeling

The dataset used for image classification includes three main classes: **COVID-19**, **Viral Pneumonia**, and **Normal**. Each class represents a different pulmonary health condition. The image proportions within the dataset were chosen to reflect a balanced distribution among the different classes, ensuring that the model could effectively learn from each category.

1. **Normal**
2. **Viral Pneumonia**
3. **Covid**

To optimize the model training, the dataset was divided into three subsets:

- **Training Set:** This set is used to train the model. It comprises about 80% of the data, allowing the model to learn the distinctive characteristics of the classes.
- **Validation Set:** Representing 10% of the dataset, this set is used during training to monitor the model's performance. It allows for parameter adjustments and helps prevent overfitting.
- **Test Set:** Consisting of the remaining 10% of the data, the test set is used to evaluate the model's final performance on data it has not seen during training.

The choice of these proportions is based on standard practices in machine learning, promoting a balance between model learning and performance evaluation.
During preprocessing, image normalization was applied by scaling pixel values to the range [0, 1]. This was accomplished using the `ImageDataGenerator` class in Keras, setting the rescaling parameter to 1/255. Normalization is important as it helps reduce data variance and facilitates the model's learning process, improving convergence and stability during training.

5.2 Model Selection

Initially, a convolutional neural network model consisting of four convolutional layers was employed. Despite the inclusion of dropout layers after each max-pooling layer and data augmentation, the model exhibited clear signs of underfitting, with a final accuracy of less than 70%. Furthermore, the analysis of Grad-CAM results on the final layer revealed that the network barely identified the ribcage, failing to extract any meaningful features related to lung inflammation.

This analysis highlighted the insufficient level of detail in the feature extraction process. As a result, it was decided to increase the number of convolutional layers and switch to the VGG16 network.

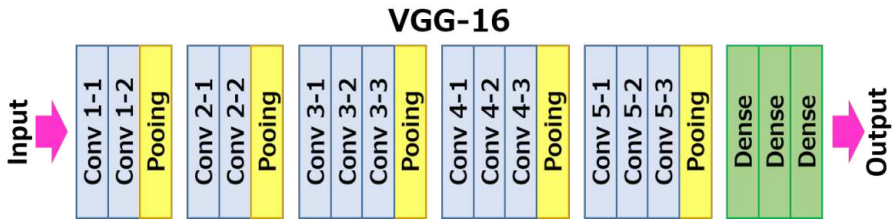

Figure 20: VGG16 architecture, showing the composition of each stack.

5.3 Callbacks Used During Training

During the model training process, several callbacks were implemented to monitor performance and manage training more effectively. The main callbacks used include:

- **ModelCheckpoint**: This callback saves the model weights during training. It is configured to store only the model with the best validation performance, using validation accuracy as the monitoring criterion. This is useful for resuming training later or avoiding loss of the best model during training.

- **EarlyStopping**: This callback monitors validation loss and stops training if no improvements are observed for a specified number of consecutive epochs (patience). This helps prevent overfitting by stopping training when the model no longer improves.

- **CSV Logger Callback**: This is a custom callback developed to log training metrics in a CSV file. The CSV Logger allows saving accuracy and loss data at the end of each epoch, enabling an in-depth analysis of the model's performance over time.

  This callback is useful if the training process is particularly slow or if there is insufficient computational power to run multiple epochs in reasonable time.

  Data saved at each epoch include:

  - Epoch number
  - Training accuracy
  - Training loss
  - Validation accuracy
  - Validation loss

  This enables tracking the model's performance trends systematically and identifying potential issues such as overfitting or underfitting. It also allows generating detailed graphs and reports on the model's performance.

- **ReduceLROnPlateau**: This callback monitors a specified metric (e.g., validation loss) and reduces the learning rate when the monitored metric stops improving for a defined number of epochs (patience).

  Reducing the learning rate can help the model converge to a better local minimum by taking smaller optimization steps when performance plateaus.

  This callback is particularly useful when the model's progress stagnates, as it allows continued fine-tuning without overshooting or terminating training prematurely.

5.4 Transfer learning from ImageNet

In this study, a VGG16 model was used to classify images of COVID-19, viral pneumonia, and normal cases. During training, pre-trained weights from ImageNet were employed, leveraging the benefits of *transfer learning* to improve performance and address common issues in deep models, such as the *vanishing gradient* problem.

In this study, a VGG16 model was used to classify images of Covid-19, viral pneumonia, and normal cases. During training, pre-trained weights from ImageNet were adopted, leveraging the advantages of *transfer learning* to enhance performance and address common issues in deep models, such as the *vanishing gradient*.

5.4.1 Benefits of Transfer Learning and Pre-trained Weights

Transfer learning allows reusing knowledge gained from a model trained on a large, generic dataset, such as ImageNet, to solve specific tasks. Pre-trained weights capture general features useful for describing images (e.g., edges, textures, and shapes), which can be easily adapted to specific domains.
For the VGG16 model, the use of pre-trained weights from ImageNet led to the following observations:

- **High initial performance**: The model achieved an initial accuracy of **95%** in the early stages of training, which could be improved to **98%** with a brief fine-tuning process. This demonstrates that pre-trained weights provide an optimal foundation for distinguishing the classes in the dataset.
- **Reduction in critical misclassifications**: Experiments showed that using pre-trained weights completely eliminated misclassification errors, such as classifying *Covid-19* as *viral pneumonia*, significantly improving system reliability.

5.4.2 Limitations of Training with Zero-initialized Weights

Before adopting the transfer learning approach, an experiment was conducted by training the model from scratch. In these tests, the model with zero-initialized weights did not exceed a maximum accuracy of **60%**, clearly highlighting the inadequacy of this strategy for this case study. The stagnation in performance is likely due to:

- **Lack of initial generalization**: The absence of a consolidated knowledge base forces the model to learn fundamental features such as edges and textures from scratch, resulting in slower training and stagnant performance.
- ***Vanishing Gradient***: In deep models like VGG16, gradients in the initial layers decrease significantly during *backpropagation*, making weight updates ineffective. Without optimal initialization, the model cannot overcome this limitation.

5.5 Results and Performance

The accuracy graph (21) and the loss graph (22) provide an overall view of the model's (VGG16) performance during training.

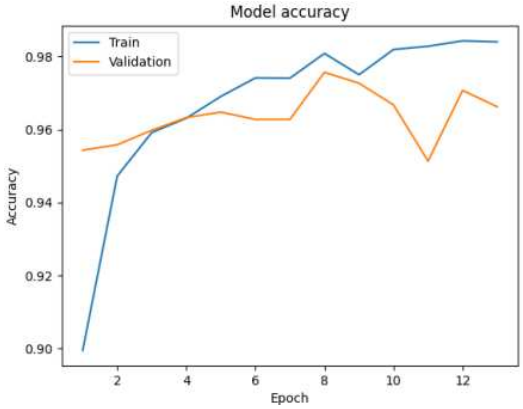

Figure 21: Model accuracy during training. The blue curve represents the training accuracy, the orange one represents the validation accuracy

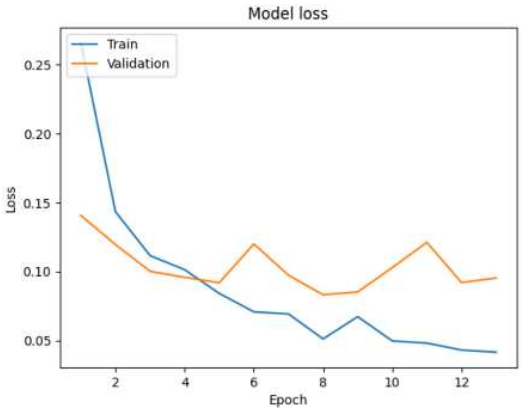

Figure 22: Model loss during training. The blue curve represents the training loss, the orange one represents the validation loss

In both graphs, the training curve shows a parabolic increase (or decrease in the case of loss), while the validation curve oscillates within a maximum range of 2%.

"If the *training accuracy* increases while the *validation accuracy* remains constant, it means that the model is improving its ability to fit the training data, but is not generalizing as well to the validation data. This scenario can indicate two possible cases:

- **Overfitting**: The model might be learning too many specific details of the training set, including noise or anomalies, without being able to generalize to unseen data. In this case, the *training accuracy* increases because the model is "memorizing" the training set, but does not improve on the validation.
- **Learning Saturation**: The model might have reached a point where it is only improving on the specific distribution of the training data, without better adapting to the validation data.

Generally, small fluctuations in the validation accuracy during training are common in many stages of the training process. They could be caused by the stochastic nature

of training, variation in the validation data, or a nonlinear response to changes in the model weights. Additionally, since the model's accuracy is very high and the loss is equally low, it can be hypothesized that the model is in the second scenario, meaning it is unable to achieve better performance than this.

5.6 Confusion Matrix

The confusion matrix, shown in Figure 23, is a key tool for evaluating the classification performance of the trained neural network on the test set. It provides a detailed breakdown of the model's predictions across the three target classes: **COVID-19**, **Normal**, and **Viral Pneumonia**.
From the confusion matrix, we can derive the following details:

- **COVID-19:**
  - True Covid: 1666 (correctly classified as COVID-19).
  - False Covid: 20 (misclassified as Normal).
  - False Covid: 0 (misclassified as Viral Pneumonia).
- **Normal:**
  - True Normal: 2448 (correctly classified as Normal).
  - False Normal: 23 (misclassified as Viral Pneumonia).
  - False Normal: 77 (misclassified as COVID-19).
- **Viral Pneumonia:**
  - True Viral Pneumonia: 1394 (correctly classified as Viral Pneumonia).
  - False Viral Pneumonia: 1 (misclassified as COVID-19).
  - False Viral Pneumonia: 11 (misclassified as Normal).

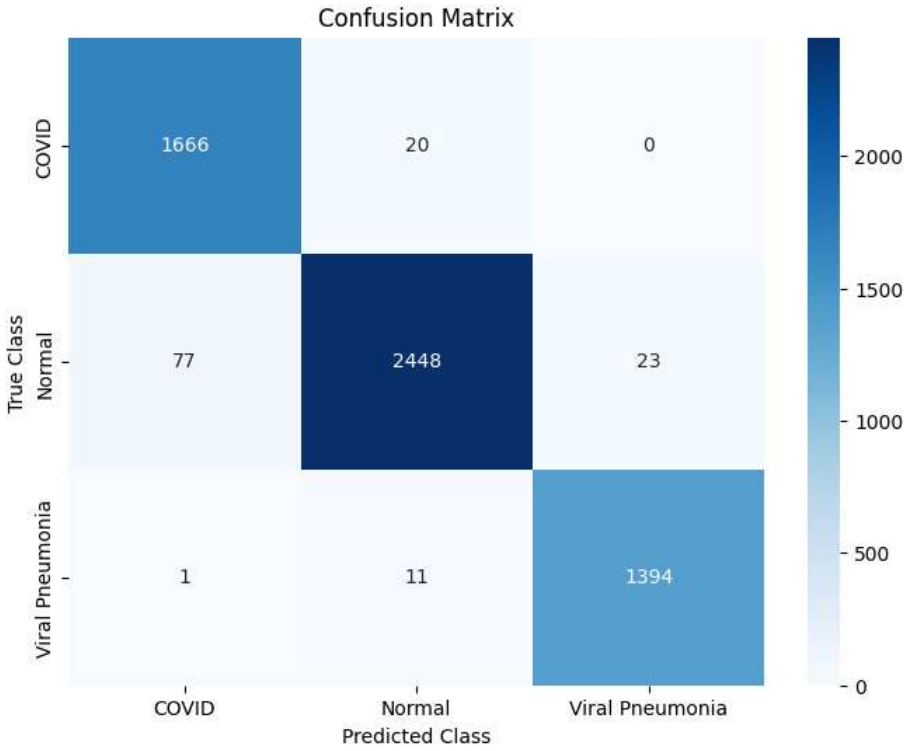

Figure 23: Confusion Matrix for the test set.

As explained earlier, the performance achieved by using pre-defined weights is very high. It is particularly notable how the model almost perfectly distinguishes between viral pneumonia and Covid.

5.7 Classification Report

Table 1 summarizes the performance metrics of the model in terms of precision, recall, F1-score, and support for each class. These metrics provide a detailed evaluation of how well the model performs for each category.

Class	Precision	Recall	F1-score	Support
COVID-19	0.96	0.99	0.97	1686
Normal	0.99	0.96	0.97	2548
Viral Pneumonia	0.98	0.99	0.99	1406
Accuracy		0.98		5640
Macro Avg	0.98	0.98	0.98	5640
Weighted Avg	0.98	0.98	0.98	5640

Table 1: Classification Report for the test set.

The metrics indicate the following:

- The model achieves a high level of precision, recall, and F1-score across all three classes, with particularly strong performance for the *Normal* and *Viral Pneumonia* classes.
- The overall accuracy of the model is 97.29%, demonstrating its effectiveness in classifying chest X-rays into the correct categories.

5.8 ROC Curve and AUC

The ROC curve (Receiver Operating Characteristic curve), shown in Figure 24, evaluates the model's ability to distinguish between the three classes (**COVID-19**, **Normal**, and **Viral Pneumonia**) using a one-vs-rest approach. The Area Under the Curve (AUC) is used as a summary metric of the model's discriminative performance.

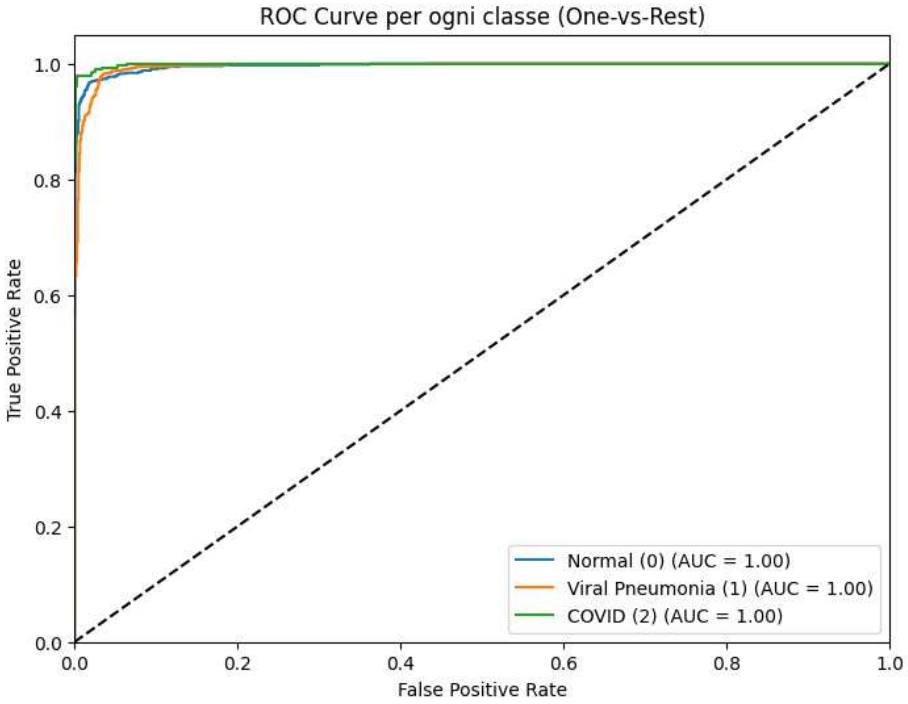

Figure 24: ROC Curve and AUC for the test set.

Key observations from the ROC curve and AUC values:

- **COVID-19:** The ROC curve for the COVID-19 class demonstrates excellent separability, with an AUC of 1.00, indicating the model's high effectiveness in distinguishing COVID-19 cases from others.
- **Normal:** Similarly, the AUC for the Normal class is 1.00, confirming the network's exceptional performance for this category.
- **Viral Pneumonia:** The AUC for Viral Pneumonia is also 1.00, showcasing the model's ability to accurately differentiate this class from others.

The AUC values of 1.00 across all classes suggest that the network achieves near-perfect discrimination among the target classes, making it an ideal tool for identifying COVID-19, normal cases, and viral pneumonia based on chest X-rays.

5.9 *Chapter 5: Summary and Insights*

In this chapter, the experimental part of the project began, where a neural network model was trained to classify images of COVID-19, Viral Pneumonia, and Normal conditions.

***Use of VGG16**: The VGG16 network was chosen for its excellent balance between complexity and performance. After the initial CNN model showed signs of underfitting, it was decided to switch to VGG16 to improve feature extraction. Its depth and proven architecture allowed for better performance in recognizing relevant patterns, such as lung inflammation patterns.*

***Dataset Size and Partitions**: The dataset was divided into three sets: 80% for training, 10% for validation, and 10% for testing. This division allowed the model to learn effectively from the data, monitor and prevent overfitting using the validation set, and ensure an accurate performance evaluation on the test set.*

***Callbacks and Normalization**: Several callbacks were implemented during training to enhance efficiency. The ModelCheckpoint saved the best weights, while EarlyStopping stopped training when no further improvements were observed. Additionally, ReduceLROnPlateau reduced the learning rate when performance stagnated. Image normalization was applied by scaling pixel values to the range [0, 1], improving model stability and convergence speed.*

***Transfer Learning from ImageNet**: Transfer learning was employed to leverage pre-trained weights from ImageNet, allowing the model to learn general image features such as edges and textures without starting from scratch. This technique enabled faster and more accurate learning, addressing common issues in deep models like vanishing gradient.*

***High Model Performance**: After adopting VGG16 and pre-trained weights, the model achieved exceptional performance, with a final accuracy of 98%. The accuracy on the test data, along with metrics derived from the confusion matrix and classification report, demonstrated that the model can accurately distinguish between COVID-19, viral pneumonia, and normal cases, with a high level of reliability.*

Chapter 6

Analysis of Classification Results

This chapter delves into a detailed analysis of the classification results by comparing the three interpretability frameworks introduced earlier: Grad-CAM, SHAP, and LIME. The goal is to analyze, where possible, the functioning and reasons behind incorrect and ambiguous classifications in a model with very high accuracy (98%).

Grad-CAM is utilized for a visual analysis of the most influential regions of an image, highlighting the areas that contribute most to the model's predictions. This method helps in recognizing distinctive patterns and features that drive the classification decisions. A more in-depth application of Grad-CAM within the hidden layers reveals how the activation masks of feature maps evolve from layer to layer, offering a clear view of where the gradient descent process occurs during feature extraction.

SHAP, on the other hand, assigns a quantitative score to each class of the classifier, enabling a comparison of superpixel activations across different classes. This approach allows for understanding not only which regions influence predictions but also how much a prediction for one class differs from another, offering a robust metric for interpretability.

LIME, by contrast, does not produce significant results in this context. The local interpretable predictors generated by LIME fail to provide consistent outcomes compared to Grad-CAM and SHAP, regardless of the image partitioning algorithm used. This inconsistency underscores the challenges of using LIME for complex visual classification tasks, where the relationship between features and predictions may not align well with its localized interpretability framework.

6.1 Grad-CAM Analysis

The analysis of the true predictions for each class highlights recurring patterns in the analyzed images. Below, Grad-CAM visualizations are presented for representative images of each class (COVID-19, normal, and viral pneumonia) to highlight the activated regions of interest.

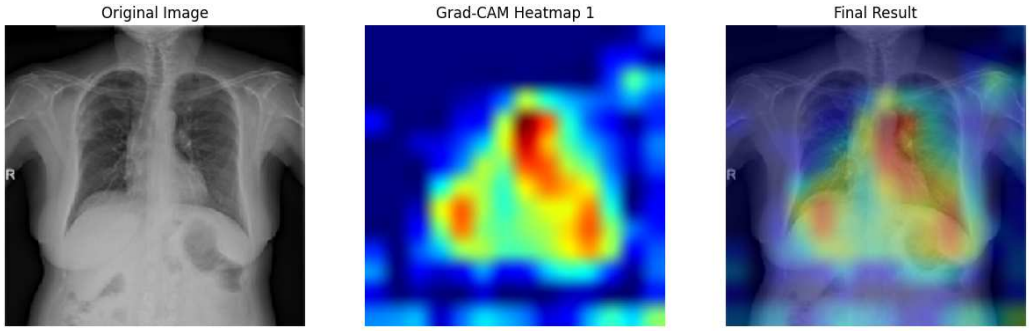

Figure 25: Grad-CAM visualization for a COVID-19 image.

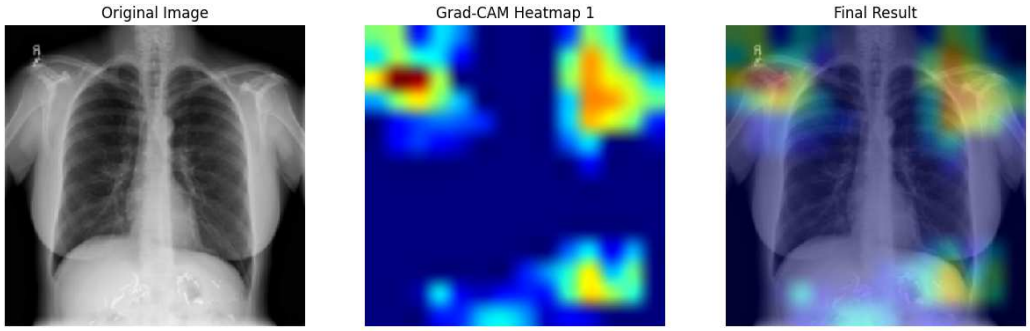

Figure 26: Grad-CAM visualization for a normal chest X-ray.

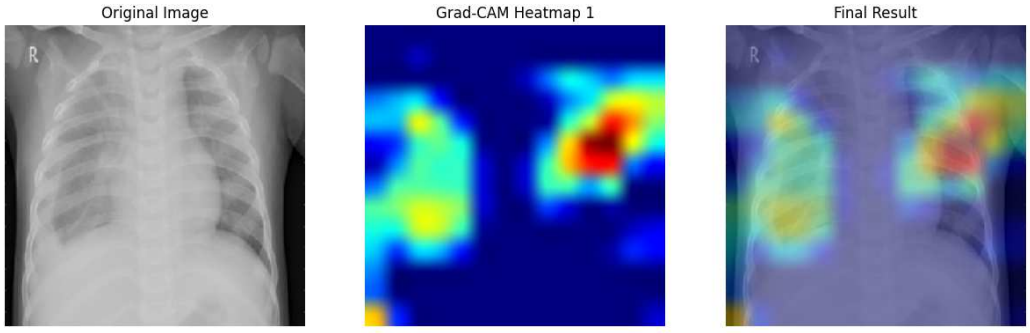

Figure 27: Grad-CAM visualization for a viral pneumonia image.

The dataset consists of 5640 images, making it challenging to generalize all possible

cases. The examples presented in this chapter have been selected to either showcase generalized patterns or highlight specific cases that provide valuable insights for the analysis.

COVID-19: The Grad-CAM analysis highlights several distinctive features commonly observed in the model's predictions for COVID-19 cases:

1. The activations are primarily **centralized** and **continuous** within the lung regions, indicating a focused response in the pulmonary area.
2. The activations are **bilateral**, originating from a central region of the chest and spreading symmetrically to both the right and left sides.
3. The activations tend to expand over **darker gray areas** in the X-rays, avoiding overly bright regions. These areas often display distinguishable patterns of striations, referred to in radiology as ***ground-glass opacities***.

Normal: The normal predictions exhibit a consistent and well-defined pattern that can be summarized as follows:

1. It is evident that the activations do not focus on the lungs but are primarily concentrated along the **edges** of the image, with the shoulders often highlighted as relevant areas.
2. Occasionally, regions near the stomach or liver are marked as relevant, although these activations are generally subtle and not strongly pronounced.
3. The network might rely on the **absence** of anomalous activations in key areas rather than actively analyzing the lung tissue, suggesting a more implicit decision-making process for these cases.

Viral pneumonia: The predictions for the viral pneumonia class exhibit the following distinctive traits:

1. One of the most noticeable characteristics is that, in many cases, only one side of the chest exhibits strong activations (often the left side). This results in a pronounced **asymmetry** in the distribution of relevant features.
2. Alongside the asymmetry, there is frequently an absence (or weak presence) of activations in the central sternum region. This contrasts with the typical pattern observed in COVID-19 cases.

3. Unlike the other two classes, viral pneumonia presents **less regular** patterns that are challenging to generalize, making its distinguishing features less consistent and harder to define.

The following three paragraphs provide an overview of a representative sample of three images selected from each class.

6.1.1 Grad-CAM Analysis of Images Classified as True COVID-19

Observing the heatmaps and overlays of images classified as *True COVID*, conclusions can be drawn about the model's behavior. A detailed analysis of each image is provided below.

First Image

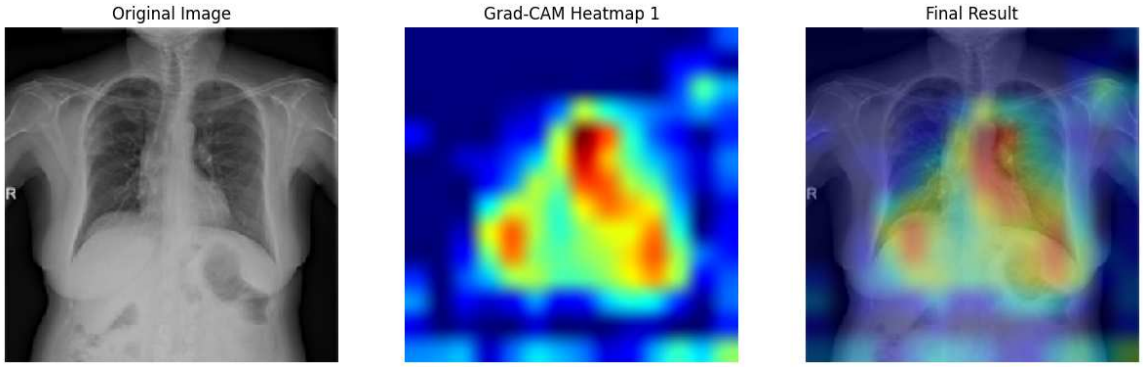

Figure 28: First True COVID image analyzed with Grad-CAM.

- The activations are distributed in the central lung area and symmetrically across the lower parts of both lungs.
- Zooming into the regions corresponding to the red-highlighted areas reveals the typical textures of *ground-glass opacities* (Figure 32).

The model correctly identifies abnormal zones, suggesting that it is using information related to lung tissue density to differentiate COVID-19 from other conditions. Central activations indicate that the model recognizes patterns of inflammation or consolidation compatible with COVID-19 pneumonia.

Second Image

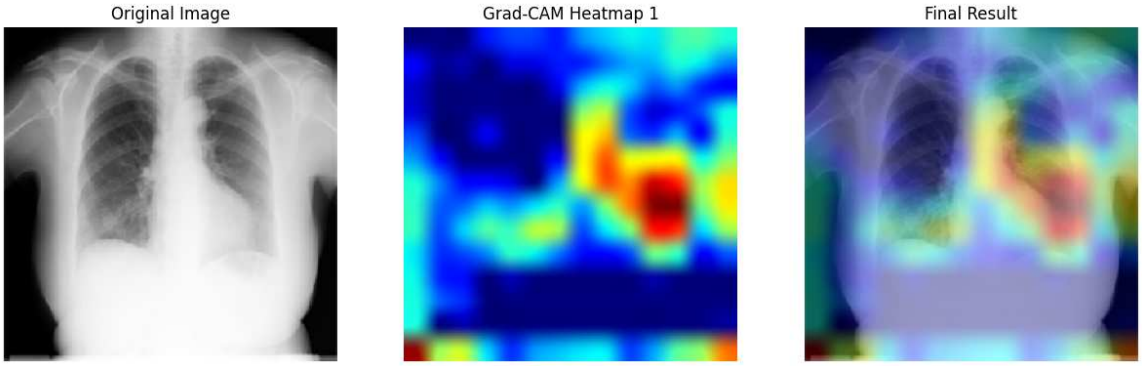

Figure 29: Second True COVID image analyzed with Grad-CAM.

- The activations in the lower area of the left lung are significantly more pronounced compared to the right lung.
- The heatmap exhibits a noticeable asymmetry resembling the pattern of viral pneumonia. However, the strong activation in the central lung region suggests otherwise.

The model focuses its attention on the lower area of the left lung, which exhibits a concentration of opacities. From this image, it is possible to clearly observe the precise color gradient the model uses to identify critical points for classification.

Third Image

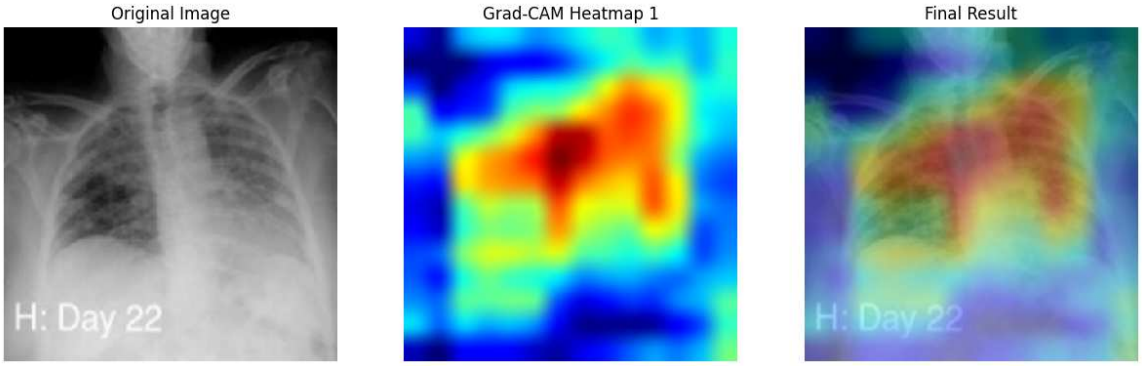

Figure 30: Third True COVID image analyzed with Grad-CAM.

- The activations cover the entire thoracic area; unlike the previous examples, this time the focus is on the upper regions.
- As in the first example, the ground glass opacities are clearly visible, the same areas that the model highlighted in red.

The model shows a clear focus on bilateral pathological patterns, such as consolidations and ground-glass opacities. Although the patient's torso is slightly tilted, the model successfully identifies the relevant anatomical points with ease.

Distinctive Patterns

- **Bilateral Opacities:** Bilateral opacities indicate that the inflammation expands across areas of both lungs. In X-rays, inflammation is represented by a grayish opacity resembling a smoke cloud that obscures the clarity of the organs and thoracic cavity. Bilateral opacities are a distinctive pattern the model uses to differentiate COVID-19 from Viral Pneumonia.

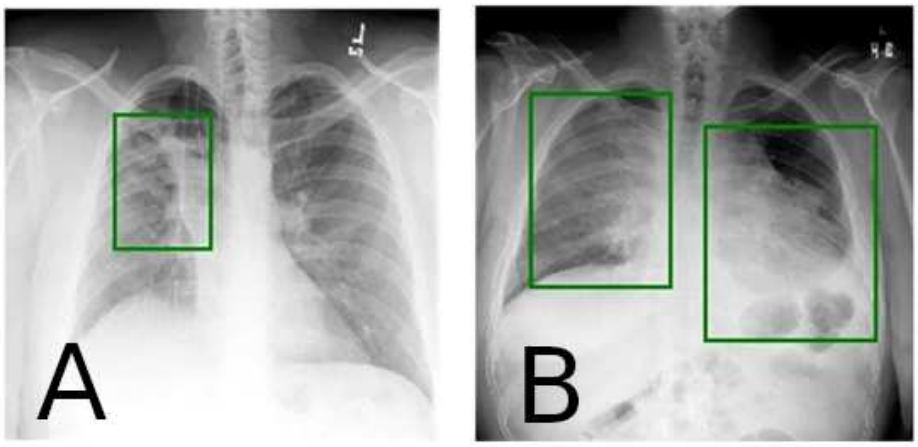

Figure 31: **A** shows unilateral inflammation. **B** represents bilateral inflammation.

- **Ground-Glass Opacities (GGO):** Ground-glass opacities (GGO) are areas of lung opacity visible in X-ray and CT images, characterized by increased lung density that does not completely obscure anatomical details such as blood vessels and bronchi. These opacities are frequently associated with pathological conditions, particularly COVID-19.

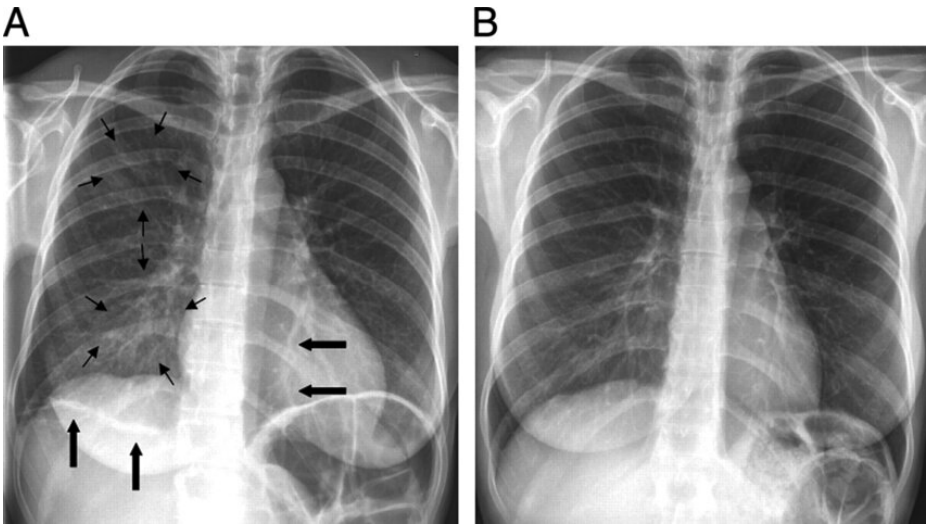

Figure 32: **A** illustrates the typical ground-glass opacity pattern, showing both evident striations in the texture and a generally grayer appearance in the affected lung area. **B** represents an X-ray without signs of inflammation.

6.1.2 Grad-CAM Analysis of Images Classified as True Normal

From observing the heatmaps of the three True Normal images, it is evident that the model's activations do not focus on the pulmonary areas, raising considerations about the classification process. Below is a detailed analysis.

First Image

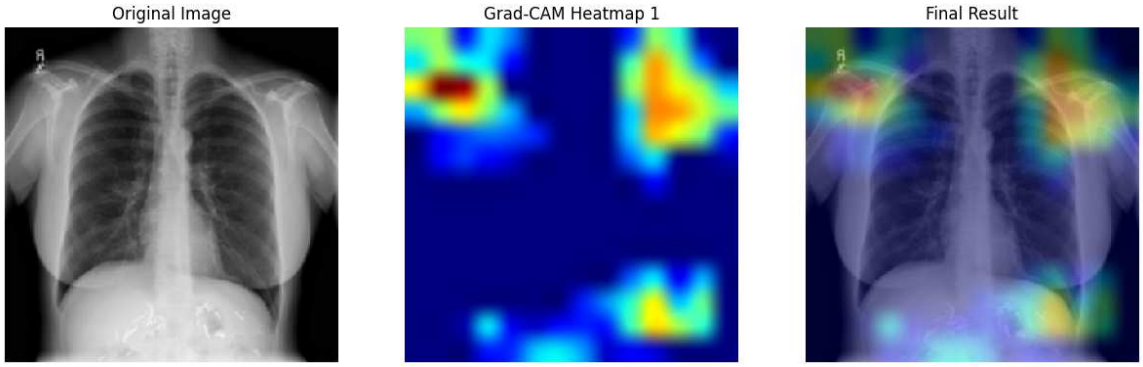

Figure 33: First True Normal image analyzed with Grad-CAM.

- The interior of the lungs is almost completely inactive (blue), indicating that the network is not analyzing these regions.
- The activations are symmetrically concentrated on the shoulders. This behavior is recurrent across the analyzed examples.

The model appears to base the classification on general characteristics of the thoracic structure (such as symmetry and edges) rather than on lung specifics. To classify "Normal," the network might rely on the absence of anomalous activations in key areas rather than actively analyzing lung tissue.

Second Image

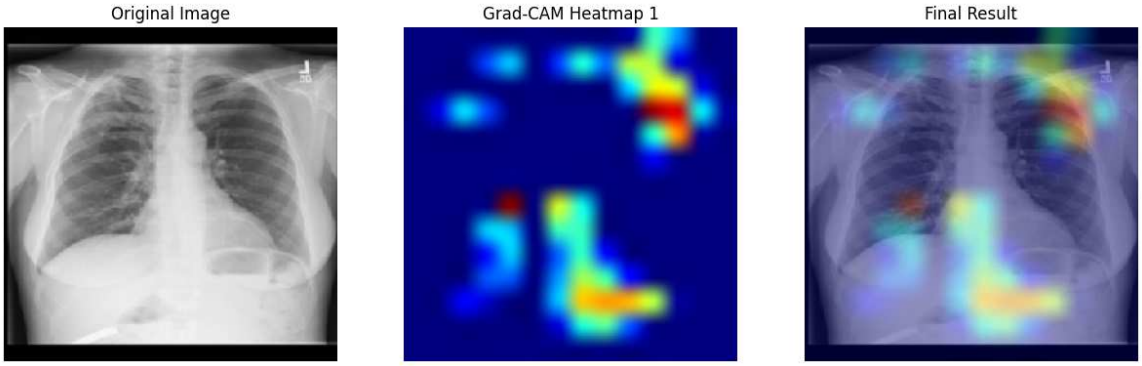

Figure 34: Second True Normal image analyzed with Grad-CAM.

- Activations are concentrated in certain peripheral regions, such as the left shoulder, the lower central area, and a portion of the stomach.
- The central lung regions remain inactive.

Here too, the classification relies on the absence of visible anomalies rather than on actively extracting pulmonary features. Activations along the edges might represent patterns the network associates with a normal structure.

Third Image

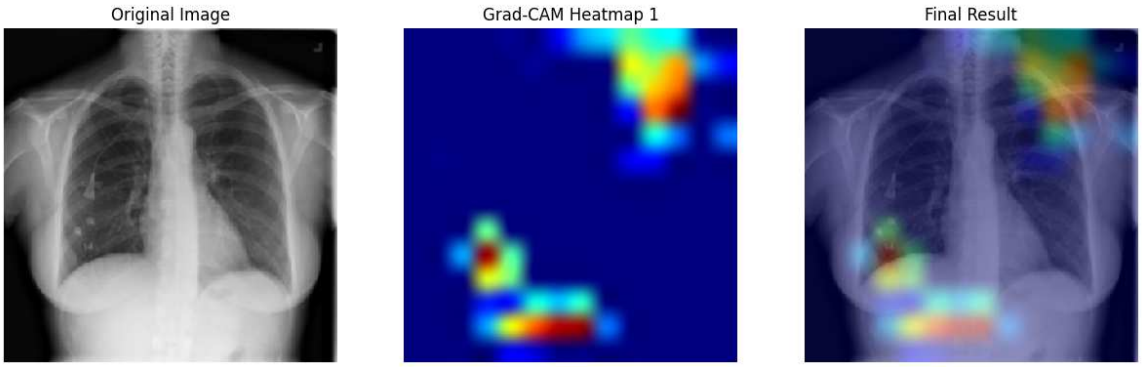

Figure 35: Third True Normal image analyzed with Grad-CAM.

- The left shoulder is consistently active, as in the previous case.
- The stomach is active, but this time it's the right side.

The network seems to completely overlook lung details. By comparing this case with the previous ones, it seems that the model tends to identify other anatomical points, such as the shoulders and the stomach, almost as if it seeks a reference point for classification.

Distinctive Patterns

- **Lack of Pulmonary Focus:** In all three images, activations in pulmonary areas are practically absent. This indicates that the network is not actively analyzing lung tissue to confirm the "Normal" classification.
- **Basis for Classification:** The network appears to rely on indirect signals, such as global symmetry, rib cage edges, and the absence of anomalous patterns that could indicate pathology.

Potential Limitations:

- The lack of activations in central pulmonary regions could represent a weakness of the model, as it does not ensure that the lungs are actively analyzed to rule out pathologies.
- This approach might work well to distinguish "Normal" from severe conditions but could be less effective in identifying subtler anomalies.

6.1.3 Grad-CAM Analysis of Images Classified as True Viral Pneumonia

The results of the analysis for classifications of the type True Viral Pneumonia are similar to those of COVID but with some differences. Unlike True Normal classifications, the most influential areas are the lungs. However, the following analyses show that the areas where opacities are concentrated are scattered and asymmetrical.

First Image

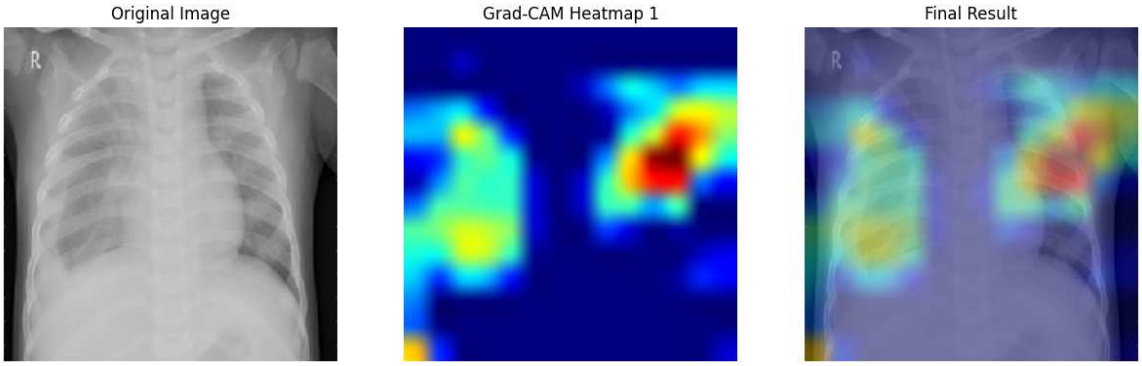

Figure 36: First True Viral Pneumonia image analyzed with Grad-CAM.

- Both the right and left lungs generate activations, but the left lung (in most cases) shows much more intense activation.
- The central area of the chest (sternum/center of the lungs) is completely inactive.

This analysis represents the most common and generalizable case in the Grad-CAM analysis. Overall, the heatmap shows asymmetry and a lack of activations in the central area of the sternum. It differs from the true COVID classification because the relevant area is not central and not uniform. It also differs from the true normal classification because the focus of the relevant areas is still associated with the thoracic cage and not with surrounding regions.

Second Image

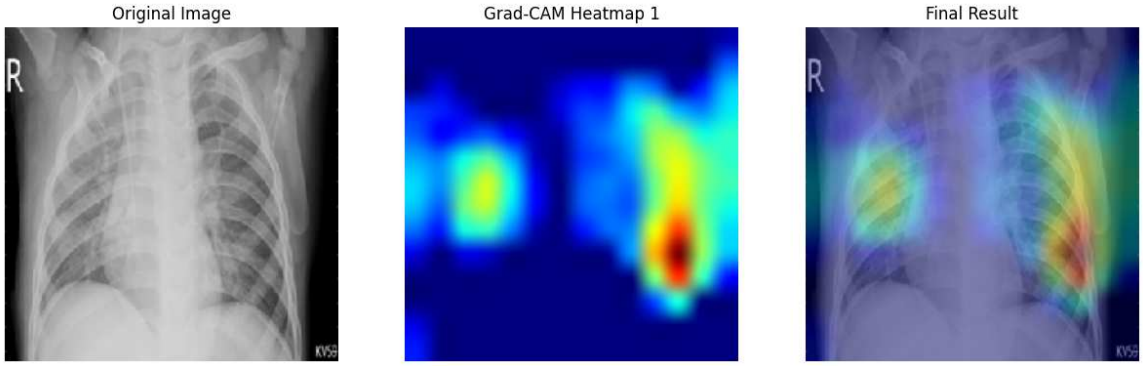

Figure 37: Second True Viral Pneumonia image analyzed with Grad-CAM.

- The case closely resembles the previous one, with no central activations. Activations are present on both the right and left sides, but the left lung shows significantly more activity.
- The area highlighted in red represents the region where there seems to be no significant concentration of opacities.

This pattern appears to reflect a case of diffuse interstitial pneumonia. The network distinguished the chest area with fewer opacities, showing asymmetry in pulmonary inflammation, which is more typical of viral pneumonia.

Third Image

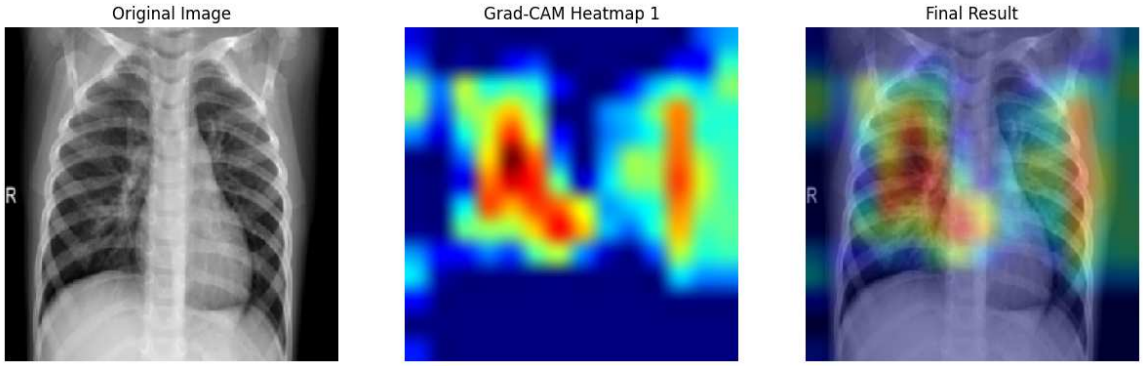

Figure 38: Third True Viral Pneumonia image analyzed with Grad-CAM.

- The areas highlighted in red are present in both lungs, but with greater intensity in the right lung.
- Analyzing the red area on the right, patterns resembling the *ground-glass opacities* typical of COVID-19 can be seen.

This suggests that *ground-glass opacities* themselves may not be the determining factor for classification, but rather, their distribution plays a crucial role. The overlay demonstrates that the network focused on key regions but lacks a symmetrical distribution typical of bilateral viral infections.

Distinctive Patterns

- **Variability:** Viral pneumonia shows a more variable distribution, with activations that do not follow a fixed pattern. This might indicate greater variability in the viral pneumonia dataset compared to COVID.
- **Importance of Non-Opaque Areas:** As seen in the examples, the model tends to focus on the thoracic cage (unlike true normal classifications) but highlights areas where inflammation/opacities are absent.
- **Asymmetry:** The detection of non-inflamed areas results in an underlying asymmetry in the heatmap of the image.
- **COVID vs Pneumonia:** The main differences between COVID-19 and viral pneumonia lie in the symmetry of activations and the localization of relevant

areas. COVID images show well-defined and bilateral activations, while viral pneumonia activations are more scattered and less intense.

6.2 Grad-CAM Intralayer Analysis

The intralayer analysis provides a detailed examination of the Grad-CAM activations of any convolutional layer of the VGG16 network. This approach allows us to reconstruct the logical process of feature extraction layer by layer, offering insights into how the model progressively refines its understanding of image features.

Process Overview: Grad-CAM intralayer analysis involves applying Grad-CAM to the final convolutional layer of each of the five convolutional stacks in the VGG16 architecture. This includes layers from shallow, general feature extraction to deep, task-specific features. By visualizing the activations across these layers, we gain an understanding of how the network processes images at different levels of abstraction. The heatmaps are related to layers in the picture: conv1, conv2, conv3, conv4, conv5.

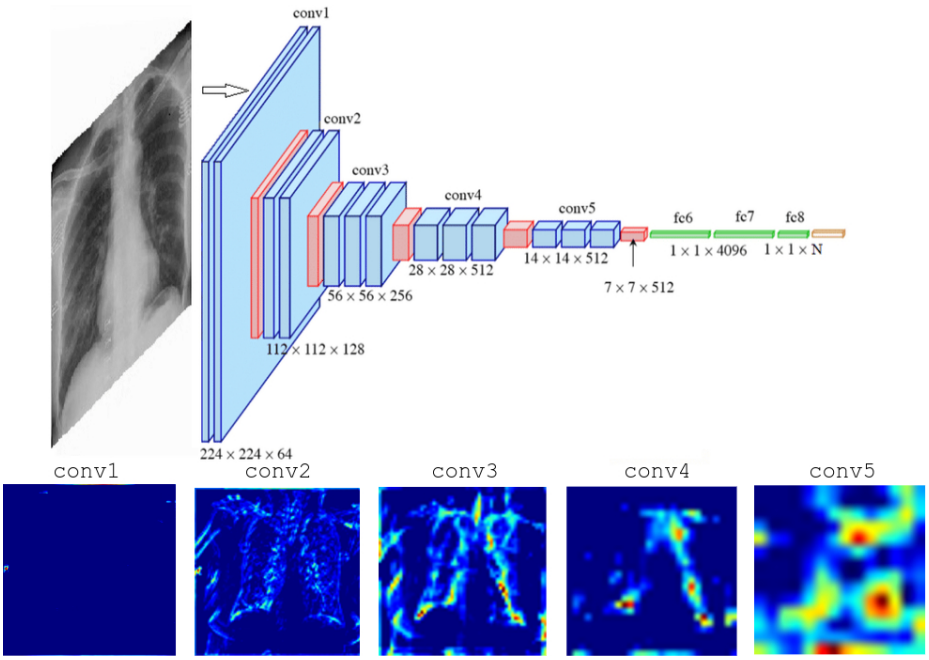

Figure 39: Structure of the VGG16 network. The highlighted layers represent the final convolutional layers of each stack analyzed during the intralayer Grad-CAM process.

6.2.1 Vanishing Gradient

The *vanishing gradient* is a problem that occurs during the training of deep neural networks, especially when using activation functions like the sigmoid or hyperbolic tangent. This phenomenon is related to the propagation of gradients through the network layers during the *backpropagation* process.

Origin of the Problem

During *backpropagation*, the gradient of the loss function is computed with respect to the network weights and propagated backward by repeatedly multiplying with the derivatives of the activation functions at each layer. When these derivatives are very small (close to zero), the gradient diminishes exponentially as it moves toward the initial layers of the network. This results in negligible updates to the weights of the earlier layers, making it difficult for the network to learn low-level features.

Effects of the Vanishing Gradient

The *vanishing gradient* problem leads to:

- Poorly trained initial layers, with weights remaining almost unchanged during training.
- Difficulty in learning fundamental low-level features, such as edges and textures in images.
- A general reduction in the speed and effectiveness of the training process.

Solutions to the Problem

Several techniques and approaches have been introduced to mitigate the *vanishing gradient* problem:

1. **Alternative activation functions:** Replace functions like sigmoid or hyperbolic tangent with functions like *ReLU* (Rectified Linear Unit) or its variants (*Leaky ReLU*, *Parametric ReLU*, *ELU*).

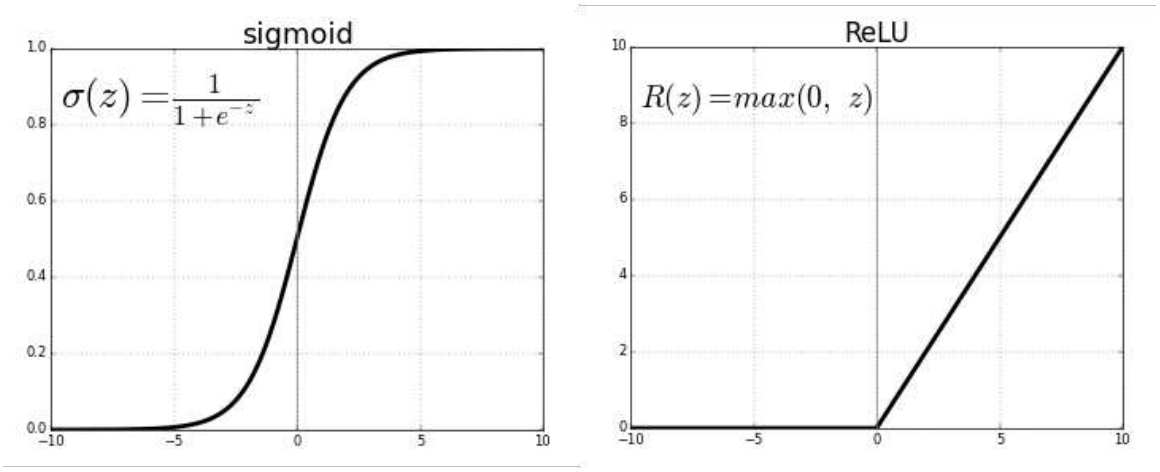

Figure 40: **ReLU** ($f(x) = \max(0, x)$) and **sigmoid** ($f(x) = \frac{1}{1+e^{-x}}$) differ mainly in gradient behavior and efficiency. ReLU maintains a constant gradient for $x > 0$, avoiding the *vanishing gradient* problem common in sigmoid, where gradients flatten for extreme inputs. Additionally, ReLU is computationally simpler and produces sparse activations ($x \leq 0 \rightarrow f(x) = 0$), improving model efficiency. Conversely, sigmoid activates all neurons and requires more complex calculations. The graphs highlight ReLU's linear behavior compared to sigmoid's curvature.

2. **Advanced weight initialization:** Use techniques such as Transfer Learning (e.g., weights pre-trained on ImageNet) or methods like *Xavier initialization* or *He initialization* to prevent gradients from becoming too small or too large.

3. **Normalization:** Introduce *Batch Normalization* to stabilize intermediate values and improve gradient propagation.

4. **Dynamic optimizers:** Adopt optimizers like *Adam* or *RMSProp*, which dynamically adjust the learning rate.

6.2.2 Example - True prediction of Viral pneumonia

This example shows the classification process of a correctly classified case of viral pneumonia by the VGG16 model. By visually analyzing each convolutional layer, one can observe the criteria used for classification to varying degrees of clarity. As previously explained, the network consists of 5 stacks, each separated by a max-pooling layer.

First Stack

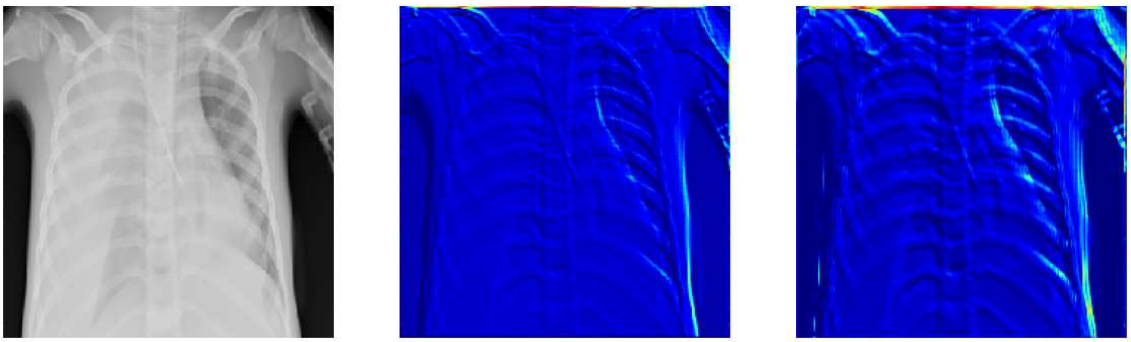

Figure 41: The figure shows an X-ray image labeled as viral pneumonia and the analysis of the first stack of VGG16.

In the first convolutional layer, the network correctly extracts edges, contours, and simple textures. Due to the extensive opacities on the left side of the chest, the right side shows a more defined pattern. In the second convolutional layer, the information remains the same, with a slight enhancement of the previously highlighted area but no additional insights.

Second Stack

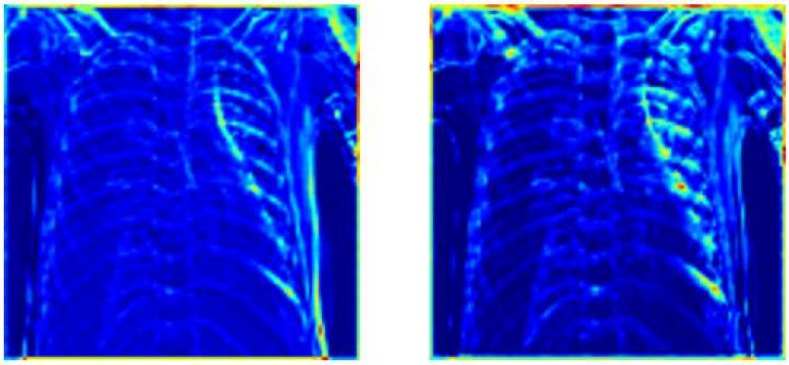

Figure 42: The figure shows the outputs of the two convolutional layers in the second stack of VGG16.

The second stack performs a deeper analysis of the structural details of the skeleton, highlighting the contours of the bones. In the first convolutional layer, all previously

highlighted areas, including those on the left side, are amplified with lighter colors (indicating stronger gradient activations). In the second layer, certain central points in the right lung area show stronger activations (yellow/red).
The first two stacks, comprising the first four convolutional layers, successfully extract the general structural details of the image.

Third Stack

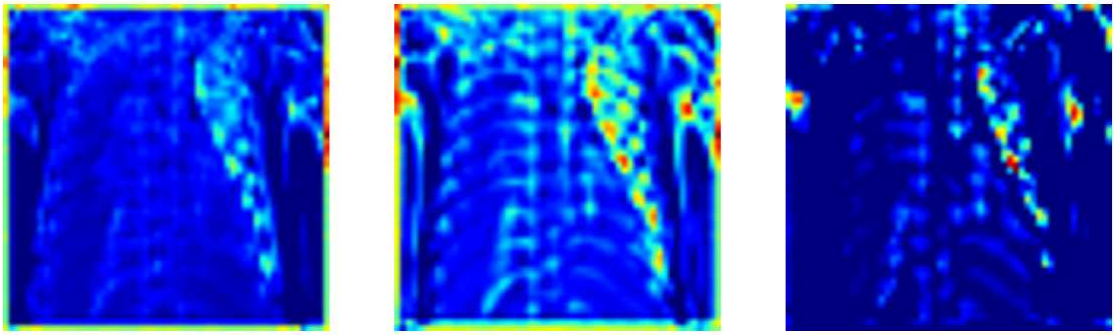

Figure 43: In the third stack, structural details, edges, contours, and simple textures are lost. The network begins to highlight more specific and singular details.

In the first layer of the third stack, the heatmap appears more blurred compared to the previous one. At this point in the analysis, the network begins to focus on feature maps not strictly related to the image's edges. In the second convolutional layer, red highlights appear between the ribs near the sternum. Referring back to the original image (41), these points represent the least blurred area of the X-ray. In the third convolutional layer, all activations appear reduced, leaving only the described points clearly active.
In summary, the third stack emphasizes the previously identified relevant points, while the last convolutional layer reduces the importance and activations of all other areas.

Fourth Stack

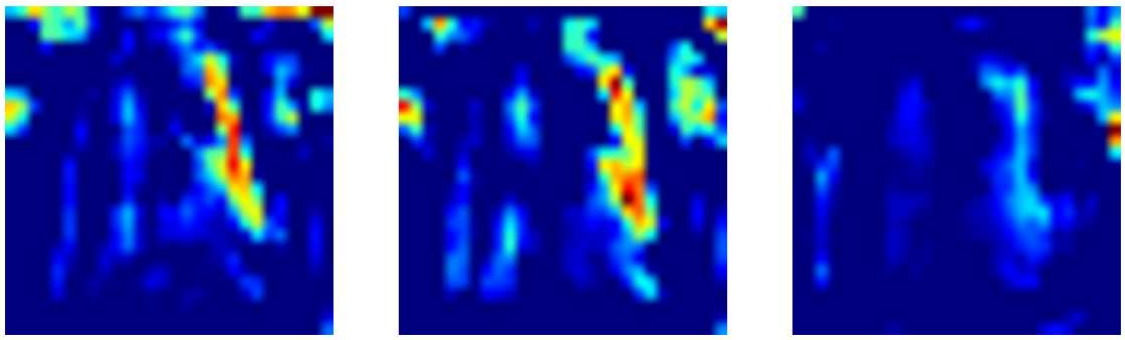

Figure 44: In the fourth stack, activations related to the skeleton's shape are almost entirely lost, and the main Grad-CAM activation area is defined.

In the first layer, the red-highlighted points from the previous stack are amplified and extended, forming a continuous area of activations. The same occurs in the second layer, but with additional focus on scattered points in the image, which appear unrelated to the classification. In the third layer, however, the behavior diverges: all activations significantly decrease, leaving only the area related to the right arm strongly active. Although the intent of this layer aligns with the third layer of the previous stack, its behavior seems anomalous. Nonetheless, the subsequent analysis does not appear to be affected by this anomaly.

Fifth Stack

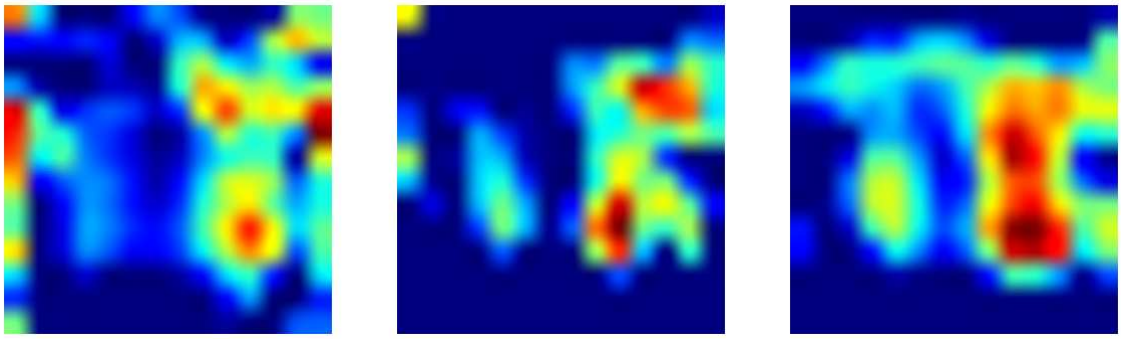

Figure 45: In the final stack, activations seem to exponentially increase across all convolutional layers, fully highlighting the right lung area.

In all three convolutional layers of the final stack, feature maps appear to exponentially increase in all areas previously identified as important. The entire right lung area becomes widely active, while the lower left area shows only slight activations. The central sternum area remains inactive, effectively separating the right and left hemispheres of the chest.

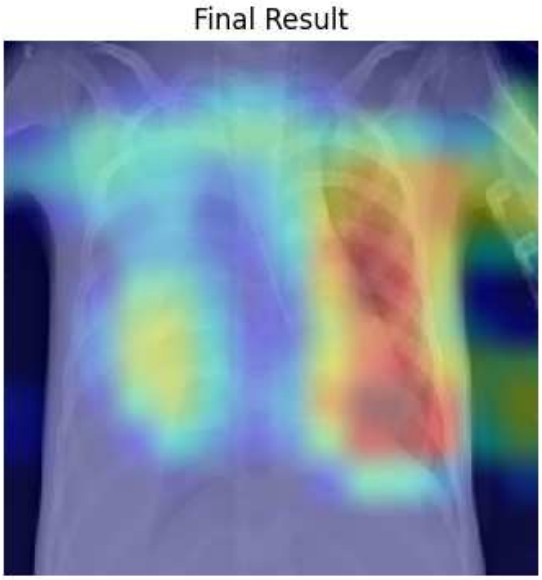

Figure 46: Original image overlaid with the final Grad-CAM result.

From the final result, it can be observed that the activation map matches the distinctive patterns associated with viral pneumonia:

1. Activations are concentrated on the lungs and not elsewhere.
2. There is a strong asymmetry between the right and left lungs.
3. The sternum remains entirely inactive, and the activation maps of the lungs are disconnected.

Except for the third convolutional layer in the fourth stack, no anomalies in the extraction of relevant features were observed.

6.2.3 Example - True prediction of a Normal case with image displacement

This example demonstrates how the model correctly classifies a normal case, even though the image is resized and slightly tilted. This displacement is not a result of data augmentation but rather due to inherent variability in the dataset, which includes images with suboptimal quality.

First Stack

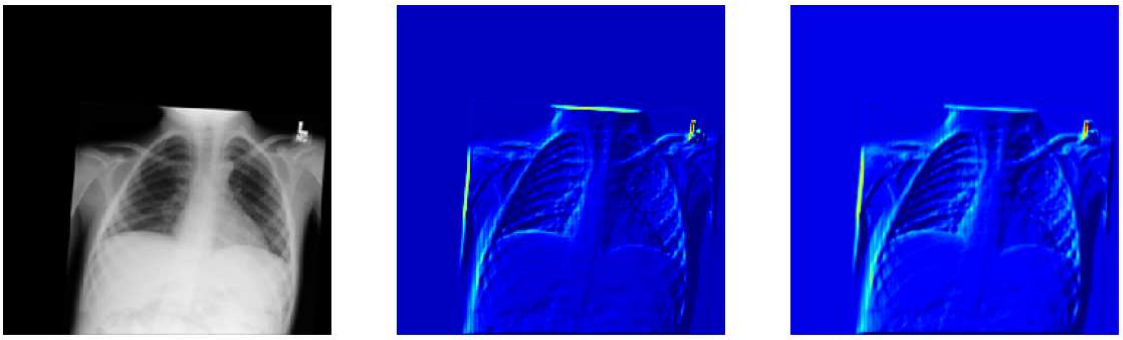

Figure 47: X-ray image labeled as normal, analyzed using the first stack of VGG16.

As in previous examples, both convolutional layers of the first stack successfully extract structural details from the image, such as edges and contours of the skeleton. The second convolutional layer slightly emphasizes the feature maps detected in the first layer. Particular attention is given to the edges where the neck and left arm terminate.

Second Stack

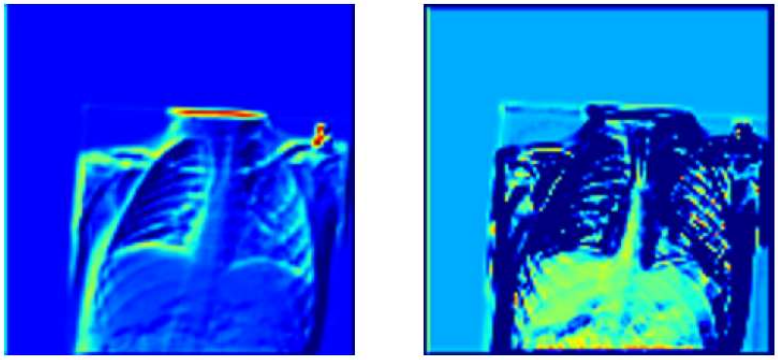

Figure 48: Output of the two convolutional layers in the second stack of VGG16, highlighting the Feature Polarity Inversion phenomenon.

In the second stack, the thoracic cavity structure is highlighted in detail, with no significant opacity or blurriness observed. A striking difference is evident between the activation maps of the two convolutional layers. This phenomenon, referred to as **Feature Polarity Inversion** (also known as **Phase Inversion** or **Sign Reversal**), occurs when successive filters process feature maps in complementary ways. Specifically, filters in subsequent layers may emphasize or suppress patterns detected earlier by computing opposing gradients or highlighting inverted contours, resulting in an effect visually resembling a photographic negative.

Third Stack

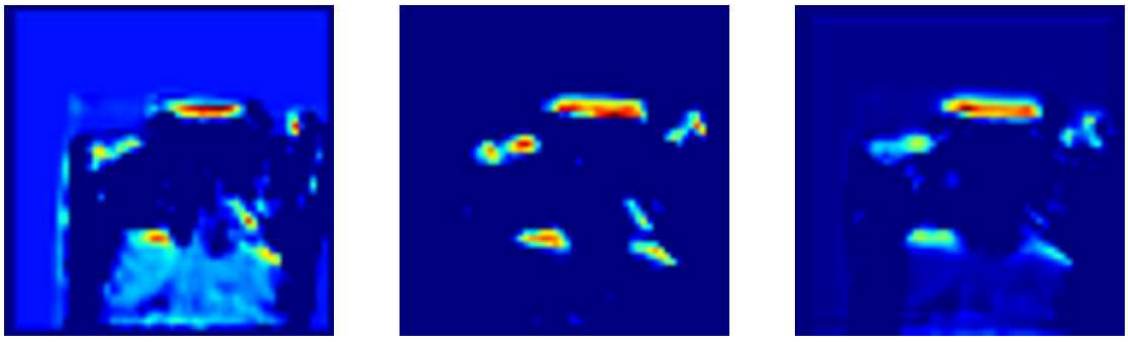

Figure 49: Emergence of critical feature maps for classification in the third stack.

Compared to the previous convolutional layer, the polarity of the feature maps in the third stack is restored, but certain regions, such as the base of the neck and the lower thoracic cavity, remain prominent. In the second convolutional layer, low-importance activations (blue regions) are further reduced to inactive (dark blue), while high-importance regions (red) are retained. The edge of the sub-image, which was visible in the previous stack, is no longer evident.
The model successfully isolates features that will be crucial for the final classification.

Fourth Stack

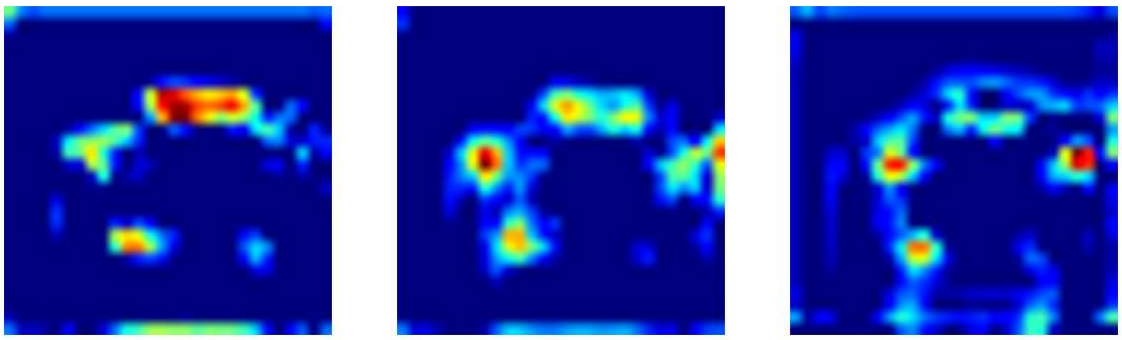

Figure 50: Feature map variations in the fourth stack, with no drastic changes.

In the first convolutional layer, the regions with high activations (red) are further accentuated, similar to previous examples. In the second layer, the model places more importance on the shoulders, a key feature for distinguishing normal classifications. This trend continues in the third convolutional layer, where the most prominent activations remain focused on the shoulders.

Fifth Stack

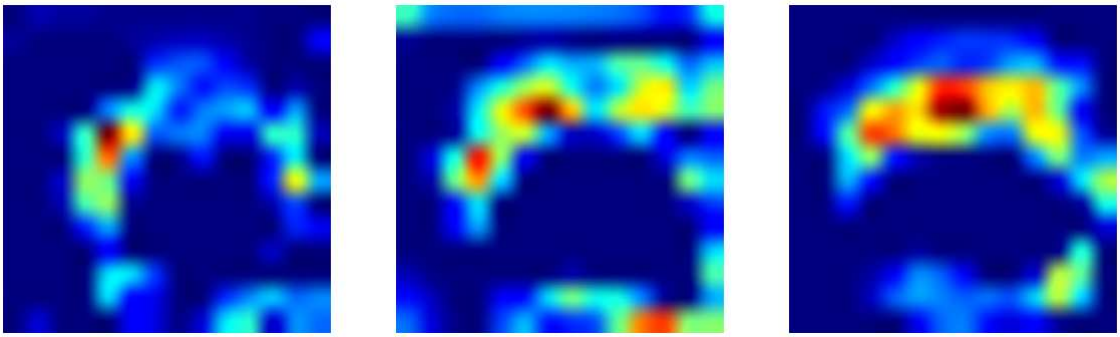

Figure 51: Final stack activation maps focusing on the neck and shoulders, delineating the upper boundary of the skeleton.

In the first convolutional layer, the activation map related to the shoulders expands slightly toward the neck region, which appears less active than in the previous stack. In the second layer, activations in the neck region regain prominence, further intensifying in the third convolutional layer.

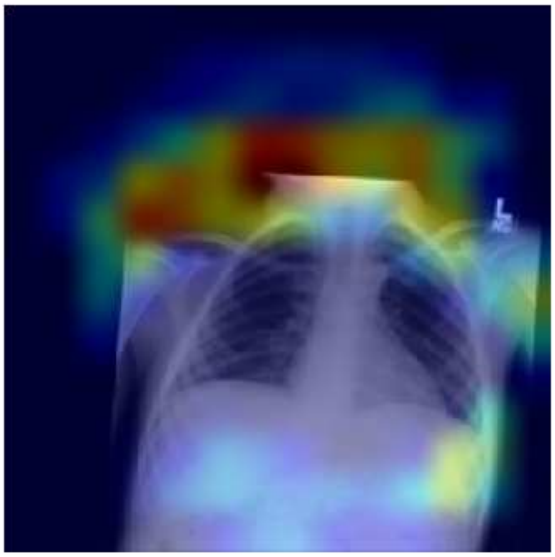

Figure 52: Original image overlaid with the final Grad-CAM result.

Analyzing the behavior of activation maps, we observe that in the initial layers, the model effectively captures the thoracic cavity and lungs, which are central to relevant

activations. From the third stack onward, the model no longer focuses on lung details but shifts attention to key points, such as the neck and shoulders. In the final stack, the union of activation maps highlights an upper boundary of the lungs, typical of normal classification.
Notably, this example was chosen specifically to include displacement, demonstrating that the model's identification of relevant features is not strictly tied to the image's geometry or spatial layout.

6.2.4 Example - True Prediction of Covid

This example presents a correct prediction of a Covid case. In this prediction, as in many other cases classified as Covid, the model struggles to extract features in the initial convolutional layers. Despite this, by progressing to higher-level feature extraction, it successfully identifies the relevant regions.

First Stack

Figure 53: X-ray image labeled as Covid, analyzed using the first stack of VGG16.

Observing the original image, it is noticeable that the X-ray appears excessively bright, with shadows outlining the contours of the image poorly defined. The first stack of VGG16, which we know focuses on light activations along edges, extracts very little information, resulting in activation maps that are mostly inactive.

Second Stack

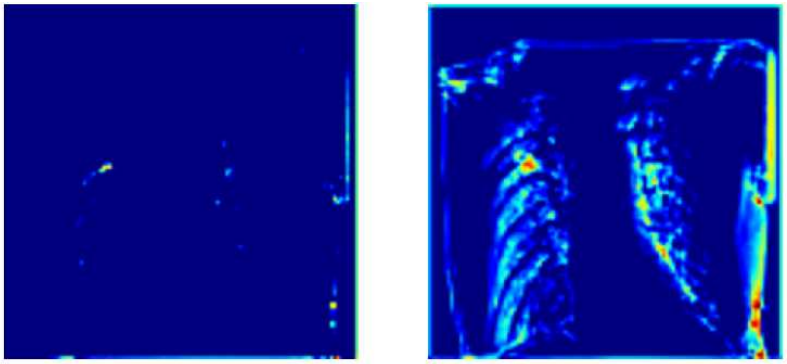

Figure 54: In the second stack of VGG16, important features related to the thoracic cavity are identified.

The first convolutional layer of the second stack seems to lose the little information extracted by the previous layers, completely failing to retain the anatomical structure of the thoracic cavity. However, the second convolutional layer clearly and sharply outlines the image's contours, a task that in other examples was achieved as early as the first stack.

Third Stack

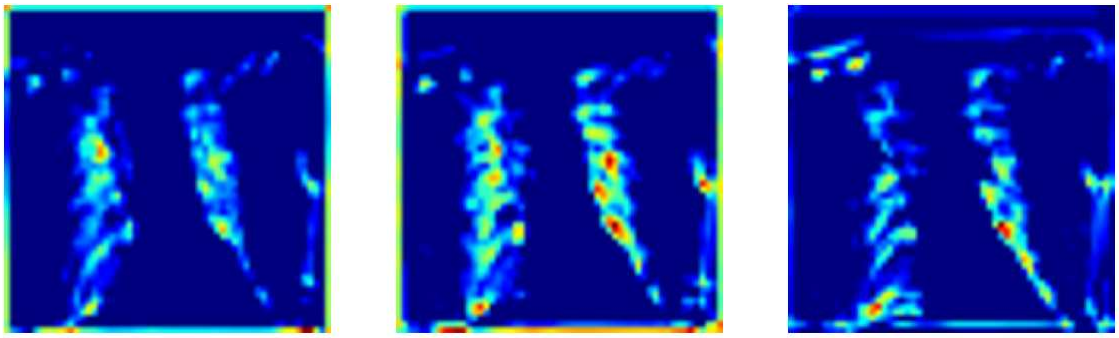

Figure 55: In the third stack, relevant features are extracted in the areas previously identified by the prior layer.

In the third stack, all three convolutional layers extract sparse but accentuated features in the areas corresponding to both lungs. This is significant because it highlights the symmetry typical of Covid cases.

Fourth Stack

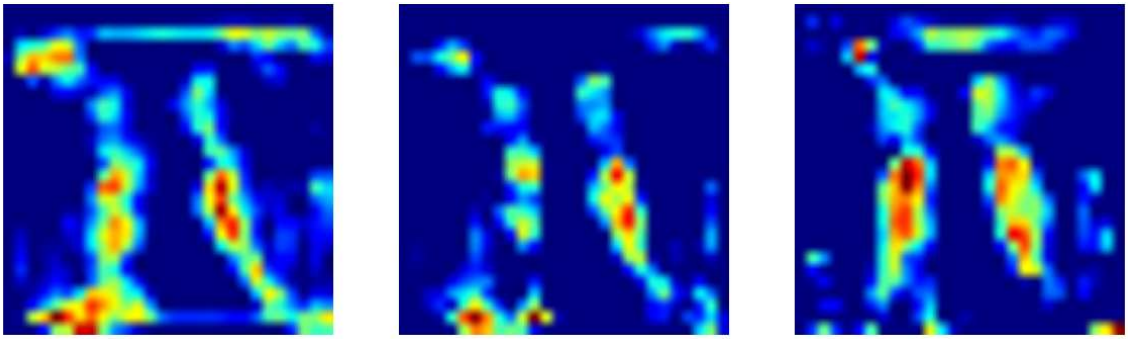

Figure 56: In the fourth stack, activations intensify vertically across both lungs.

As seen in previous examples, after identifying specific points with intense activations, the network tends to expand the area of interest, transforming disconnected points into a more coherent activation map. This happens symmetrically across both lungs.

Fifth Stack

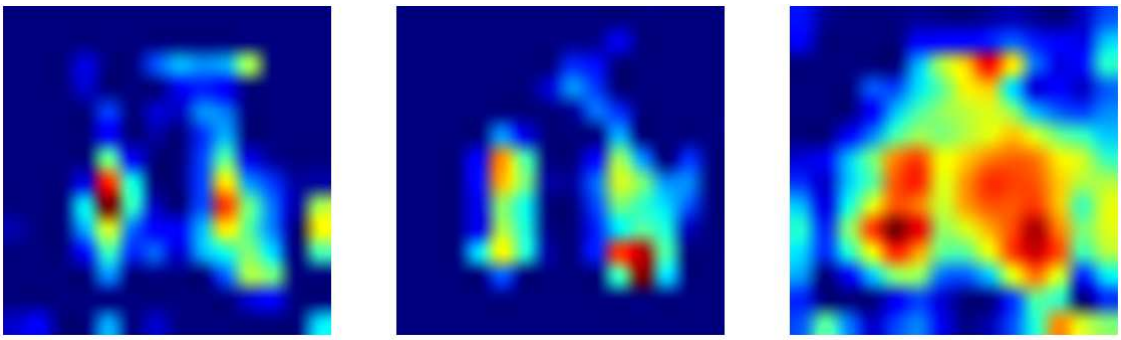

Figure 57: In the final stack, activations spread widely across both lungs and the central area.

In the fourth stack, the activation map changes further from layer to layer. In the first layer, activations reduce, suppressing some sparse, irrelevant areas outside the lungs. In the second layer, activations along the image's edges are further flattened, while lung-specific activations expand. Finally, in the last layer, the activation map significantly amplifies and spreads across the entire lung regions and toward the central thorax, with notable intensity.

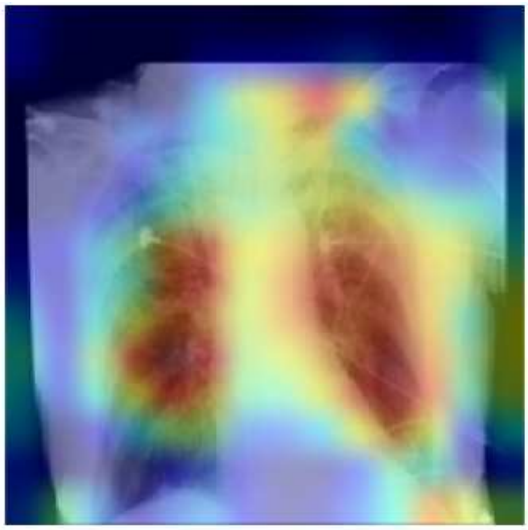

Figure 58: Original image overlaid with the final Grad-CAM result.

Examining the original image overlaid with the red activation areas, it is easy to identify the patterns of ground-glass opacities (32). This explains the high intensity of activations in this example. Additionally, note the bilateral and central positioning of the activations, indicating a pattern typical of Covid cases.

6.2.5 Results of the Grad-CAM Analysis

The Grad-CAM analysis conducted on various samples highlighted the main patterns that the network associates with the classification of different classes. Below, the results for each class are summarized:

Normal Class: The heatmaps for the "Normal" class show that the network focuses its activations mainly on the peripheral areas of the image, avoiding a direct analysis of the lungs and the thoracic cavity. This suggests that the "Normal" classification relies on the absence of visible anomalies rather than a detailed examination of the pulmonary structures.

Covid Class: For the "Covid" class, the heatmaps reveal activations concentrated in the central lung areas and on specific opacities present in the thoracic cavity. This indicates that the network associates this class with patterns of abnormal density typical of COVID-19 infections.

Viral Pneumonia Class: The heatmaps for the "Viral Pneumonia" class exhibit an asymmetric pattern, with greater focus on the thoracic cavity, particularly the unaffected region. This suggests that the network uses asymmetry as a key indicator to distinguish this class.

General Considerations: Compared to other datasets, chest X-ray images have very similar characteristics, which introduces two main challenges and peculiarities:

1. **Difficulty in visual distinction:** For an experienced radiologist, distinguishing between the presence and absence of pathology may be relatively straightforward. However, in cases of confirmed pathology, even an expert may find it challenging to determine with certainty whether it is a viral lung infection or COVID-19. In such situations, a computed tomography (CT) scan is often required for a more precise diagnosis. In these cases, the model could serve as a valuable tool to guide the type of diagnostic follow-up needed.

2. **Importance of geometric localization:** Since the size and offset of chest X-rays are generally standardized, the model "knows" the exact coordinates of the anatomical elements in the image. This makes the concept of geometric localization and the distinction between symmetry and asymmetry particularly significant in classification.

6.3 SHAP Analysis

In the context of image classification, SHAP (SHapley Additive exPlanations) offers a unique approach to interpretability compared to other methods such as Grad-CAM and LIME. While Grad-CAM provides visual heatmaps that highlight regions of the input image contributing to the model's decision, and LIME generates local approximations of the decision boundary, SHAP assigns a numerical score to quantify the contribution of each pixel towards the membership of the image in every possible class.

One notable advantage of SHAP over LIME lies in its grid-based masking approach. This ensures that every superpixel contributes to the classification, avoiding the non-regular segmentation and random perturbations of LIME, which introduce non-deterministic and potentially unstable factors into the explanations.

When applied to image classification, SHAP analysis goes beyond visual explanations by providing a detailed numerical breakdown of how strongly the input image belongs to each class. This makes SHAP particularly valuable for tasks requiring precise interpretability. For each input image, SHAP calculates a contribution score for all

classes. These scores are aggregated to provide a global understanding of the model's decision-making process while preserving the pixel-level granularity of explanations.

While Grad-CAM focuses solely on positive contributions, SHAP considers both positive and negative contributions, providing richer data and insights that can be analyzed to enhance explainability.

Unlike Grad-CAM and LIME, which focus predominantly on the most likely class, SHAP ensures that every class is considered, offering a comprehensive perspective on the model's behavior.

6.3.1 SHAP Analysis for a True COVID Prediction

In this section, we analyze the results of the SHAP analysis applied to an image that was correctly classified as COVID by the model. The model's prediction for this image shows a class ranking where the highest probability is assigned to COVID, followed by *Viral Pneumonia*, and finally *Normal*. Below, we present the original image along with the SHAP analysis and the corresponding Grad-CAM analysis.

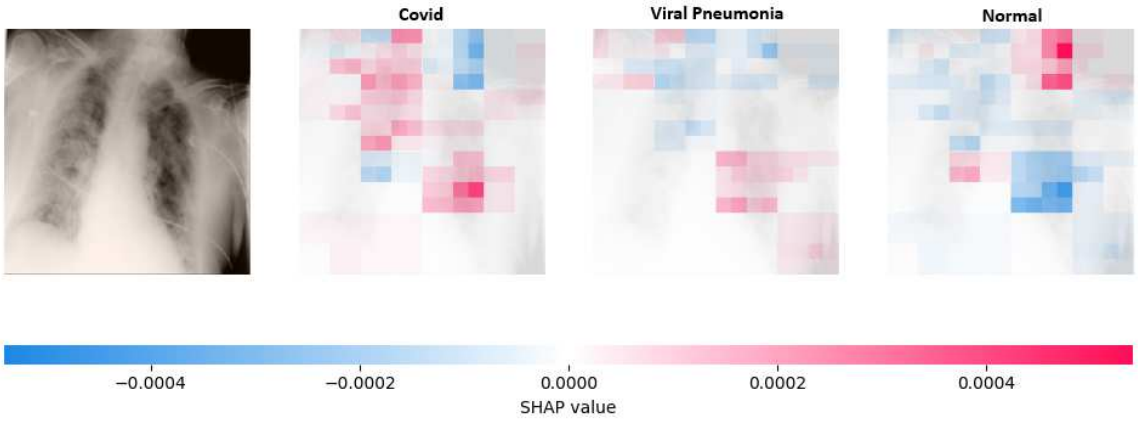

Figure 59: Example image classified as COVID with SHAP. The red values indicate a significant positive contribution to the classification of the specific class. The blue values indicate a negative contribution. White areas have no influence.

The SHAP analysis for the COVID class shows a strong correspondence with the associated Grad-CAM analysis. The areas highlighted by both techniques focus on the same superpixels, indicating that the model primarily based its prediction on

these regions. This result strengthens confidence in the model, as the two independent techniques converge on the same relevant features.

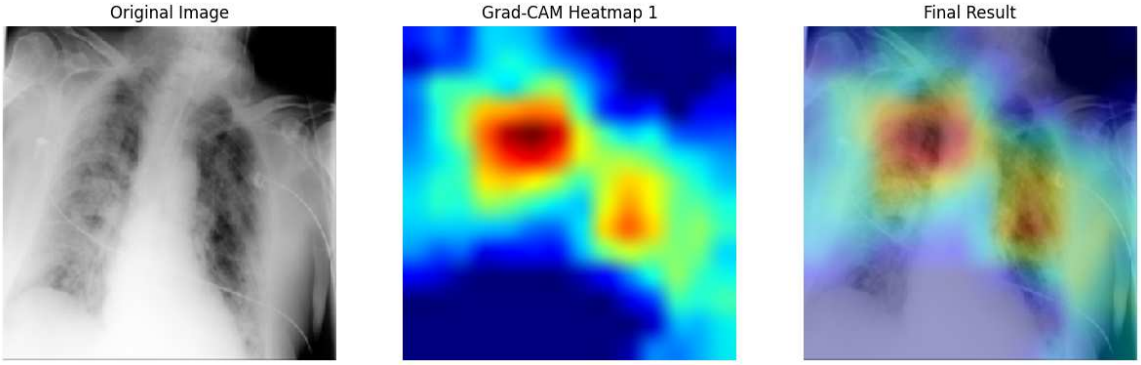

Figure 60: Grad-CAM applied to the same image correctly predicted as COVID. The correspondence of the red areas with the SHAP analysis in the first image of the previous figure can be observed.

A further examination reveals an interesting behavior across different classes. Observing the superpixels highlighted by SHAP for the COVID and *Normal* classes, we note that the blue and red regions are almost perfectly opposite to each other. This implies that the superpixels that positively contribute to the classification as COVID are the ones that most strongly discourage the classification as *Normal*, and vice versa.

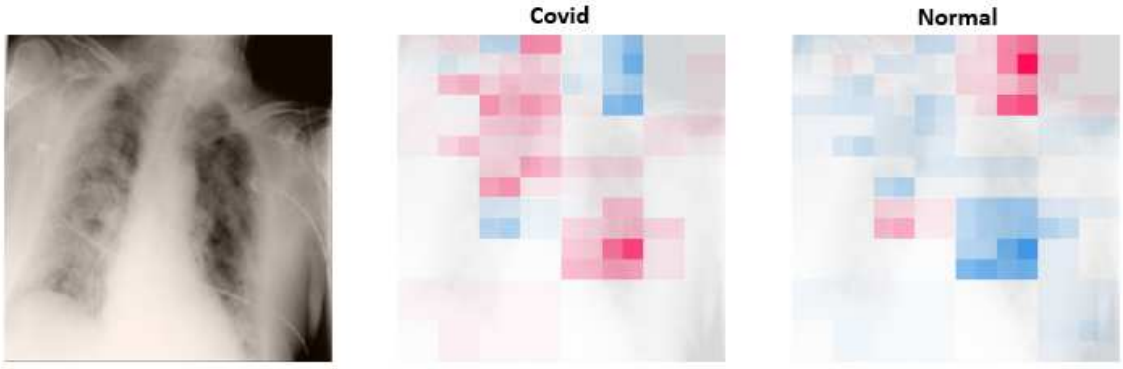

Figure 61: The image shows the positive (red) and negative (blue) contributions of the superpixels. In the two masks, the superpixels are almost exactly opposite.

On the other hand, the superpixels highlighted for the COVID and *Viral Pneumonia* classes show greater similarity. However, in the case of the *Viral Pneumonia* class, the superpixels in the left lung area lose intensity, creating an asymmetry between the two sides of the lungs.

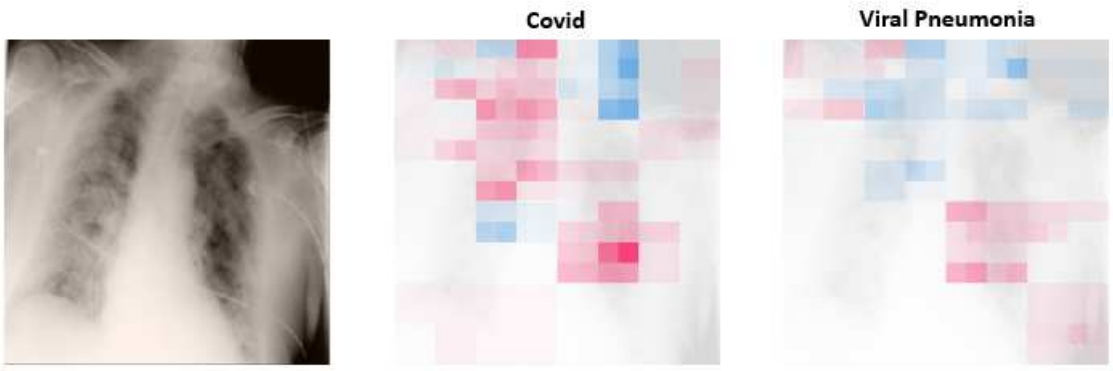

Figure 62: In the mask corresponding to Viral Pneumonia, the entire upper-left area appears blue, indicating that it is not a decisive region for the Viral Pneumonia classification.

6.3.2 SHAP Analysis for a True Normal Prediction

In this section, we analyze the SHAP results for an image that was correctly classified as Normal by the model. Similar to the previous case, the relationship between the Normal and COVID classes is consistent, with opposing red and blue regions. The ranking of the class probabilities for this prediction is Normal, followed by Viral Pneumonia, and finally COVID. Below, we present the SHAP analysis and the corresponding Grad-CAM analysis for this prediction.

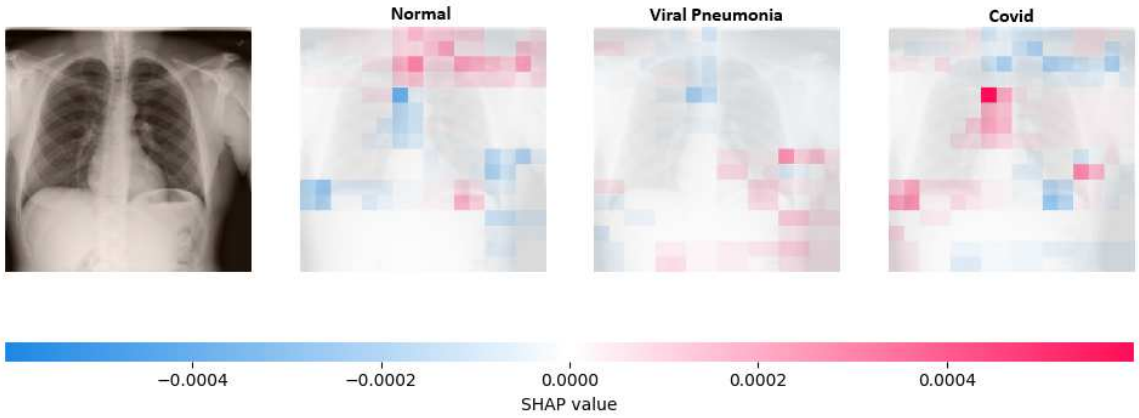

Figure 63: Example image classified as Normal with SHAP. The red regions contribute positively to the classification as Normal, while the blue regions discourage it. White regions indicate no significant influence.

Similar to the SHAP analysis, Grad-CAM highlights the same key regions, further validating the reliability of the model's decision-making process.

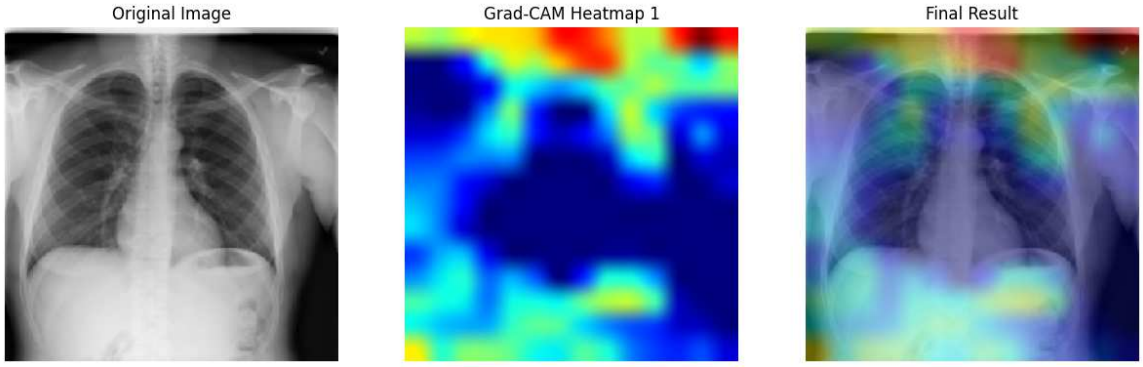

Figure 64: Grad-CAM applied to the same image correctly predicted as Normal. The highlighted areas in Grad-CAM correspond well to the SHAP analysis in the previous figure.

In the case of the Normal prediction, the SHAP values associated with individual superpixels are much less pronounced compared to the previous example. This indicates that the majority of the superpixels do not contribute significantly to the classification, suggesting a less distinct pattern in the image compared to other classes.

Finally, the comparison between the COVID and Viral Pneumonia classes is entirely different and does not exhibit any notable similarities. This further emphasizes the distinct features that the model relies on when differentiating between these two classes.

SHAP Analysis for a True Viral Pneumonia Prediction

In this example, as in the previous cases, SHAP was used to analyze the correct prediction of a Viral Pneumonia case. The red pixels in the SHAP map represent positive contributions to the classification, while the blue pixels indicate negative contributions. Unlike the cases analyzed for COVID and Normal, this example does not show a strong dualism between the two primary classes. The distribution of points is more scattered, reflecting a less pronounced pattern in activations compared to the other examples.

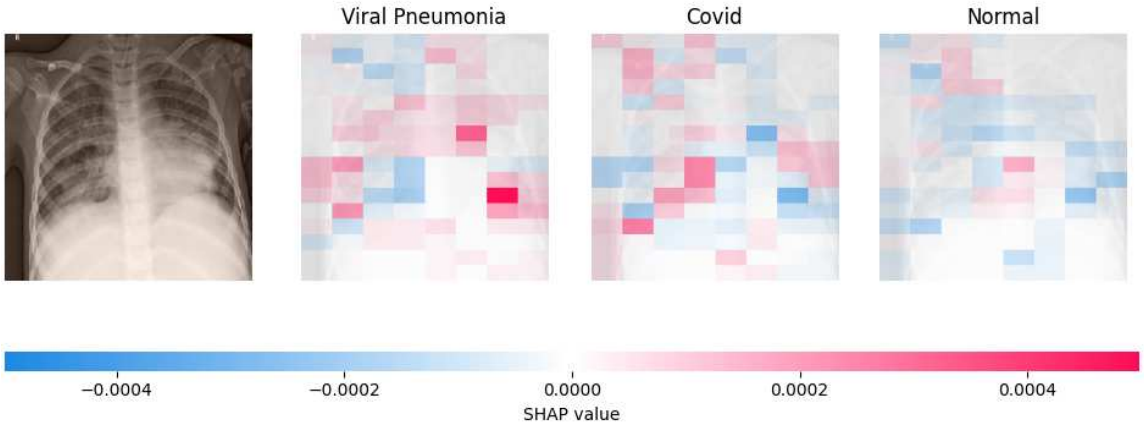

Figure 65: Example image classified as Viral Pneumonia with SHAP.

Despite this characteristic, certain points with significant positive contributions stand out clearly in the SHAP map. These points are particularly important as they highlight the areas the model relied on to recognize the Viral Pneumonia case. As observed, these points correspond to the red areas highlighted by Grad-CAM in the following figure. This correspondence suggests that both methods focus on crucial regions, confirming the robustness of the model's predictions.

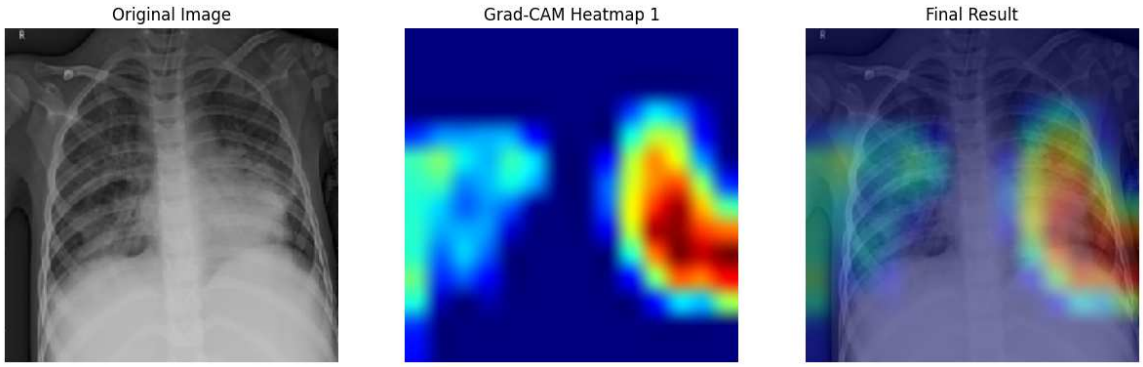

Figure 66: Grad-CAM applied to the same image correctly predicted as Viral Pneumonia. The highlighted areas in Grad-CAM correspond well to the SHAP analysis in the previous figure.

The Grad-CAM applied to the same image further strengthens the interpretation: the red areas highlighted correspond perfectly to the SHAP points with positive contributions. This result indicates that both methods converge in recognizing specific regions, such as opacities and irregular lung contours, that characterize Viral Pneumonia cases.

6.3.3 Shapley Values in Image Analysis

As mentioned earlier, each Shapley value is associated with a numerical scale, reflecting the contribution of individual pixels to the model's prediction. By examining the relationships between these values for each class, a detailed numerical analysis can be performed, helping to identify which regions of an image have the most influence on the classification.

The sum of all Shapley values associated with a prediction of a class provides the ranking of that class in comparison to the others. In the analyses performed, both positive and negative contributions are considered to understand the level of balancing associated with each class and to better understand the "distance" between one prediction and another.

Example

In the example shown in Figure 67, as in previous cases, the superpixels are overlaid on the original image, with each one assigned a numerical value displayed on the scale beneath the image.

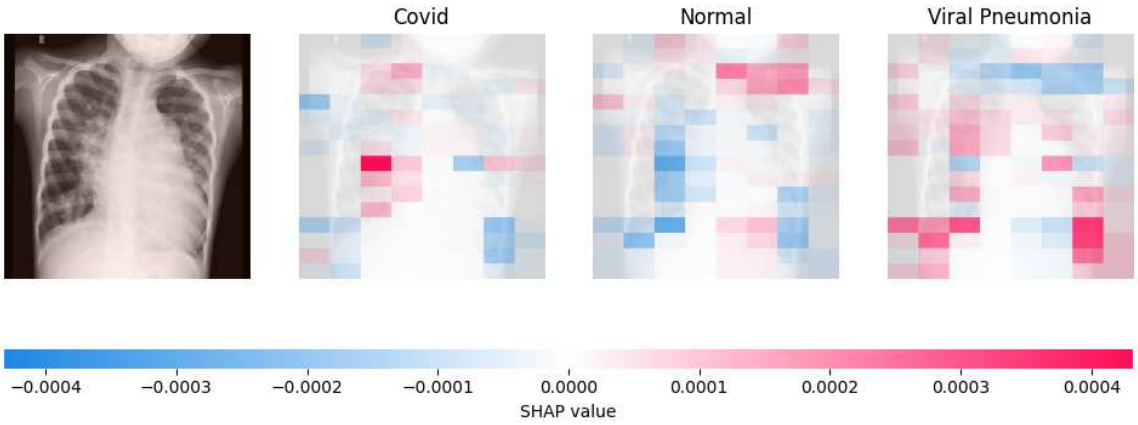

Figure 67: Example of SHAP analysis on a chest X-ray case of true viral pneumonia.

The superpixels are organized into a 16x16 grid, and each cell is assigned a value quantized along a gradient scale that transitions from blue to magenta, passing through white. For the purpose of numerical analysis and data representation, white pixels and those close to white were isolated. To achieve this, a *data clipping* process was performed on colors with hexadecimal codes whose chromatic components fell below a certain threshold.

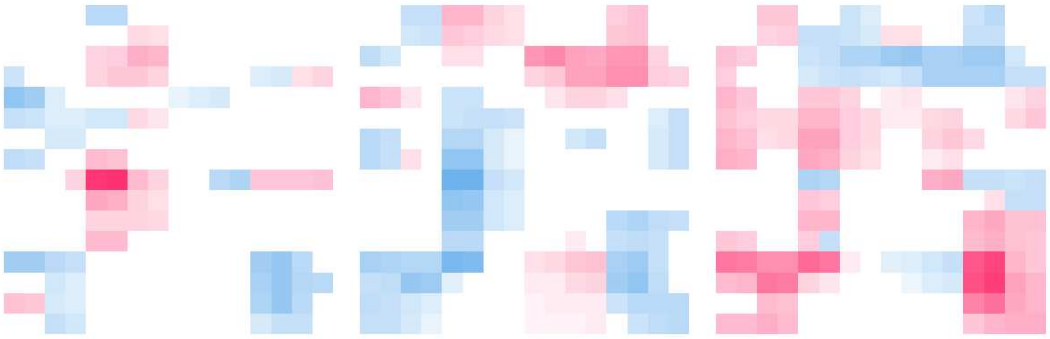

Figure 68: Shapley value maps with *data clipping* filtering for: COVID, normal, and viral pneumonia, respectively.

In Figure 69, the Shapley values are shown in a histogram with real values divided by class.

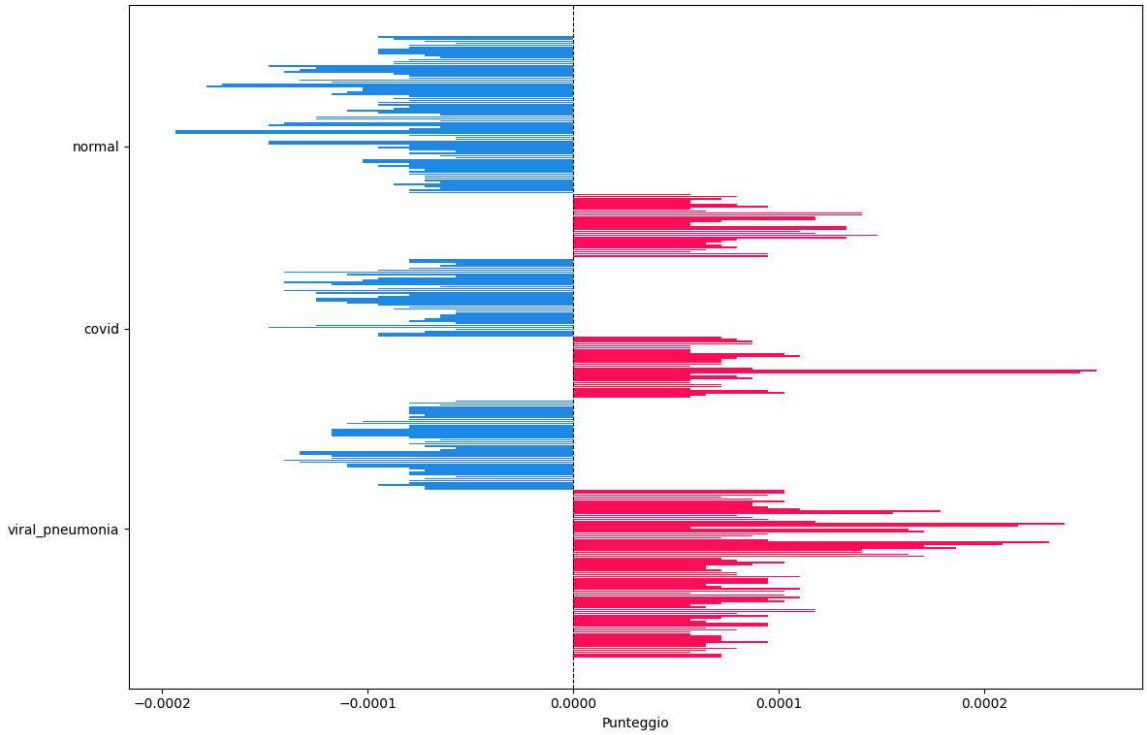

Figure 69: Representation of the positive and negative Shapley values for each class.

From the graph, several important insights about the distribution of the Shapley values can be immediately observed. The key takeaways relate to the quantity of features/Shapley values per class and the high intensity of certain contributions. Here are some considerations:

- The positive values for the class *viral pneumonia* are significantly more numerous than for the other classes and greatly outnumber the negative ones. At first glance, it is clear that *viral pneumonia* is the prediction with the highest score.
- In the *COVID* class, there are some values that are remarkably high, reaching the maximum value of the numerical scale. This intensity in the Shapley values can be explained by examining the original image, where the reddest pixels likely correspond to areas of opacity, potentially classified as ground-glass opacities.

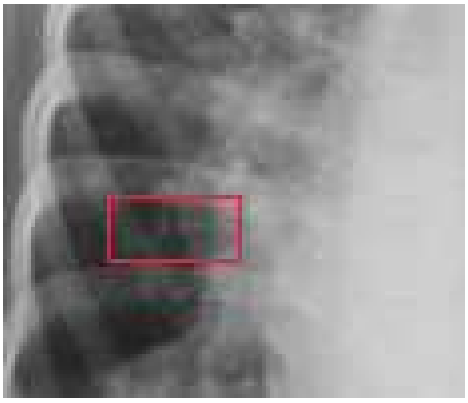

Figure 70: A zoomed-in view of the area in the original image corresponding to the two Shapley values with the maximum values on the scale for the *COVID* prediction. It indeed appears to be a ground-glass opacity.

- Despite the two high positive values in the *COVID* class, there are numerous negative values that balance out the final result when aggregating the contributions. In the subsequent two graphs, the sum of the Shapley values for each class is displayed.
- For the *normal* class, negative contributions are more numerous and more intense than the positive ones, clearly indicating that this class has the lowest score overall.

Once the individual Shapley values are analyzed, they must be summed to obtain the contribution (both positive and negative) they have for each class.

In Figure 71, we clearly observe that the positive values for the viral pneumonia class far outweigh the negative ones, giving it a higher ranking among the three classes. The normal class, on the other hand, also has negative values that significantly exceed the positive ones. More interesting is the data related to the COVID class, which, despite containing the features with the highest value, has a total of negative contributions that outnumber the positive ones.

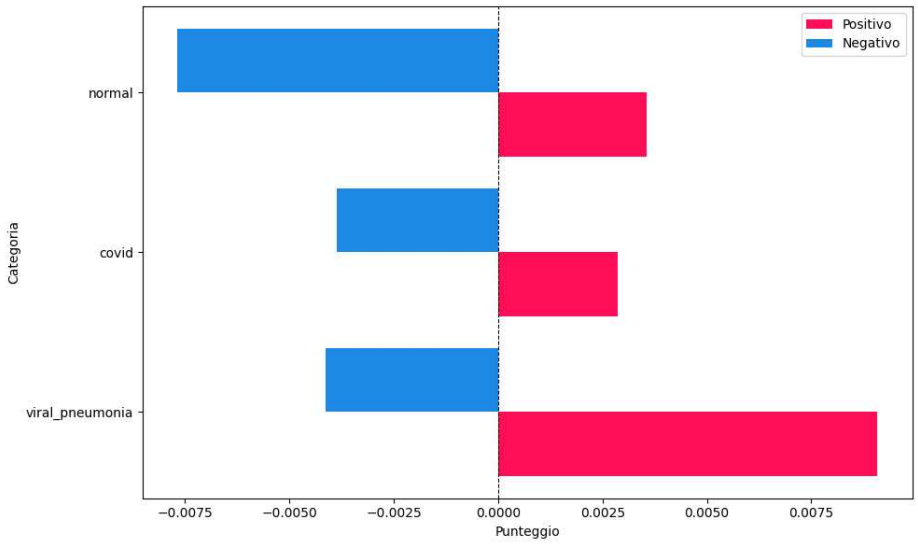

Figure 71: A histogram showing the sum of positive and negative values.

In the last Figure 72, the total contribution of the Shapley values for each class is shown, calculated simply as the sum of both positive and negative values.

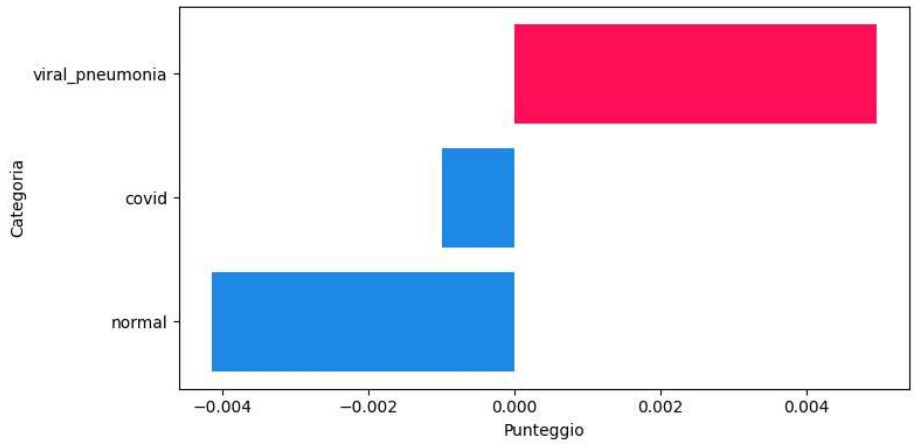

Figure 72: Overall sum of the Shapley values for each class.

The final scores are:
1) `viral_pneumonia: 0.004955696202531647`
2) `covid: -0.0009911392405063293`
3) `normal: -0.004135443037974682`

In the SHAP analysis, it is observed that predictions with a ranking score close to zero are the result of the sum of positive and negative contributions that cancel each other out.

6.3.4 Statistical Analysis of Shapley Values

Now that the SHAP value rankings for an image in the test set have been defined and displayed, it is possible to perform a statistical analysis on the distributions of these values across the test set.

To perform this analysis, a sample of 280 images with true predictions was taken. For these images, the rankings of the sum of the Shapley values were calculated. In Table 2, the mean score and the standard deviation for each class are shown, while in Figure 73, the normal distribution of the data is presented.

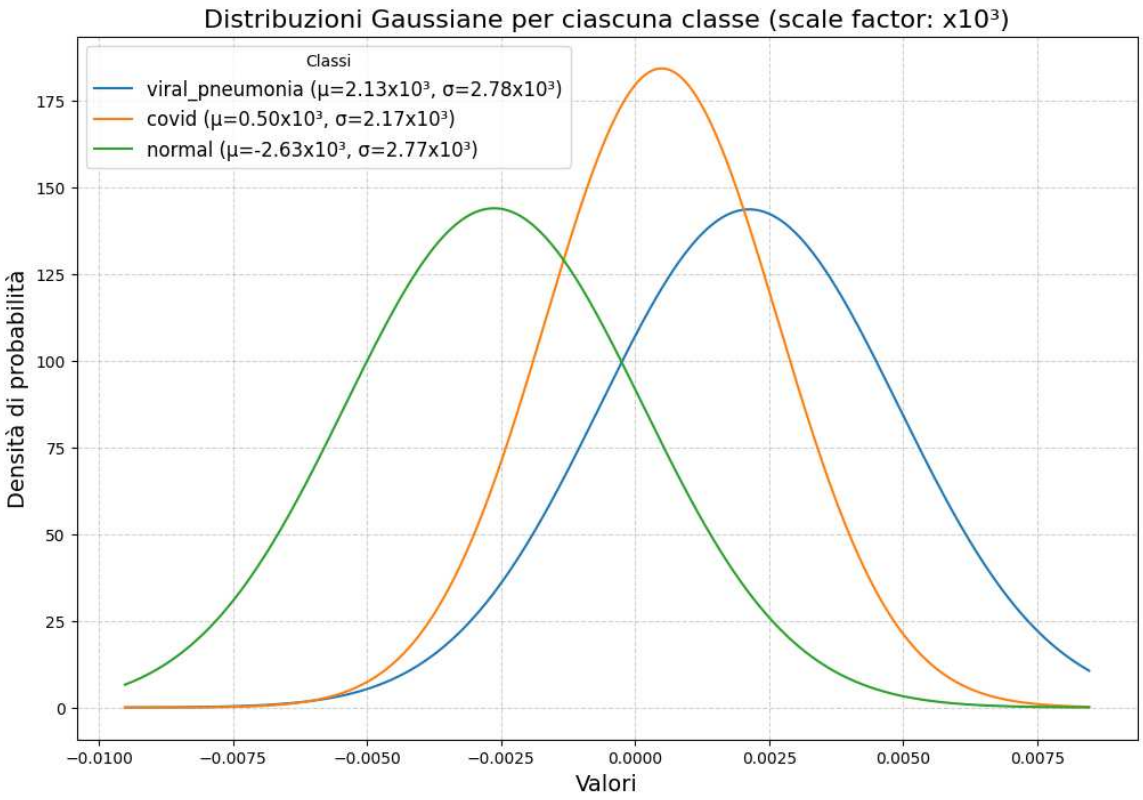

Figure 73: Representation of the distribution of Shapley values for each class. The mean values is on the x-axis, the probability density on the y-axis.

Class	Mean Score	Standard Deviation
Viral Pneumonia	0.0021	0.0028
Covid	0.0005	0.0022
Normal	-0.0026	0.0028

Table 2: General statistics of the scores for Viral Pneumonia, Covid, and Normal.

By observing the graph, it can be seen that the Covid class has a higher probability concentration towards the positive values near zero. As already seen in the previous example, this result can be justified by the presence of both positive and negative contributions of the Shapley values in almost equal measure. In the previous example, the negative contributions were greater, whereas in this statistical analysis, on average, the positive contributions prevail.

The Shapley values for Viral Pneumonia and Normal, on the other hand, have similar probability distributions, but the mean scores are positive and negative, respectively. The importance of negative contributions is significant in these graphs. The distributions of the curves are closely related to the amount of positive Shapley values compared to the negative ones.

Absolute Differences between Mean Values of Classes

The absolute difference between the mean values of each class indicates the level of ambiguity in the predictions. A larger difference suggests that the model is able to distinguish the two classes with greater confidence, while a smaller difference implies that the model struggles to differentiate between the classes.

- **Viral Pneumonia and Covid**: Difference = 0.0016 (most similar).
- **Covid and Normal**: Difference = 0.0031.
- **Viral Pneumonia and Normal**: Difference = 0.0048 (most distinct).

As expected, and intuitively, the greatest ambiguity lies in distinguishing between Viral Pneumonia and COVID, as both are pathological cases and exhibit similar patterns. The Normal predictions are quite distant from both of the other classes.

Correlations between Classes

The correlation coefficient measures the strength and direction of the linear relationship between two variables. It ranges from:

- -1: perfect negative correlation (when one variable increases, the other decreases in constant proportion).
- 0: no linear correlation (the two variables are independent or not linearly correlated).
- 1: perfect positive correlation (when one variable increases, the other also increases in constant proportion).

The most commonly used correlation coefficient is Pearson's correlation coefficient, which is calculated with the formula:

$$r(X, Y) = \frac{Cov(X, Y)}{\sigma_X \cdot \sigma_Y} \tag{16}$$

Where:

- $Cov(X, Y)$ is the covariance between X and Y, which measures how they vary together.
- σ_X and σ_Y are the standard deviations of the variables X and Y.

By calculating the Pearson correlation coefficients between the classes, we obtain:

- **Viral Pneumonia and Covid**: Correlation = -0.424 (moderate and negative).
- **Viral Pneumonia and Normal**: Correlation = -0.660 (strong and negative).
- **Covid and Normal**: Correlation = -0.363 (moderate and negative).

The negative correlations indicate that as one class increases, the other decreases. The correlation between **viral pneumonia and COVID-19** (-0.424) is moderate, suggesting an inverse relationship, but not very strong: as the probability of belonging to one of the two classes increases, the probability of the other decreases. The correlation between **viral pneumonia and normal** (-0.660) is stronger, indicating a clear distinction: as the probability of belonging to viral pneumonia increases, the probability of being in the normal class decreases markedly. Finally, **COVID-19 and normal** (-0.363) shows an inverse but weaker relationship, with visual overlaps between the two.

6.4 LIME Analysis

The analyses conducted with LIME are the least successful experiment in this work. In most cases, the areas or superpixels identified as influential or non-influential for classification correspond to skeletal parts of the image, which are decidedly irrelevant for the classification task. Let us briefly revisit the steps of the LIME algorithm (12).

1. **Perturbation of the Input Data:** The algorithm generates multiple perturbed versions of the input sample by making slight modifications (e.g., masking portions of the image or altering feature values).

2. **Model Predictions:** Each perturbed sample is passed through the original model to obtain its predictions, which are then recorded.

3. **Weight Assignment:** A proximity metric is used to assign weights to the perturbed samples based on their similarity to the original input. Samples closer to the original are given higher weights.

4. **Training a Simple Model:** A simpler, interpretable model (e.g., linear regression) is trained locally on the perturbed samples, using their assigned weights and the predictions of the original model as the target output.

5. **Result Interpretation:** The coefficients or contributions of features in the simpler model are analyzed to understand which features or regions are most important for the original model's decision.

The likely reason for the method's ineffectiveness is related to point 1, specifically the criterion the algorithm uses to segment the image into superpixels. LIME uses the Quickshift segmentation algorithm by default, which clusters pixels based on color similarity and spatial proximity.

This algorithm is not very effective for the images in our dataset, as it creates too many samples (superpixels), with segmentation primarily influenced by the skeletal geometry rather than other features. This results in insufficient focus on lung opacities.

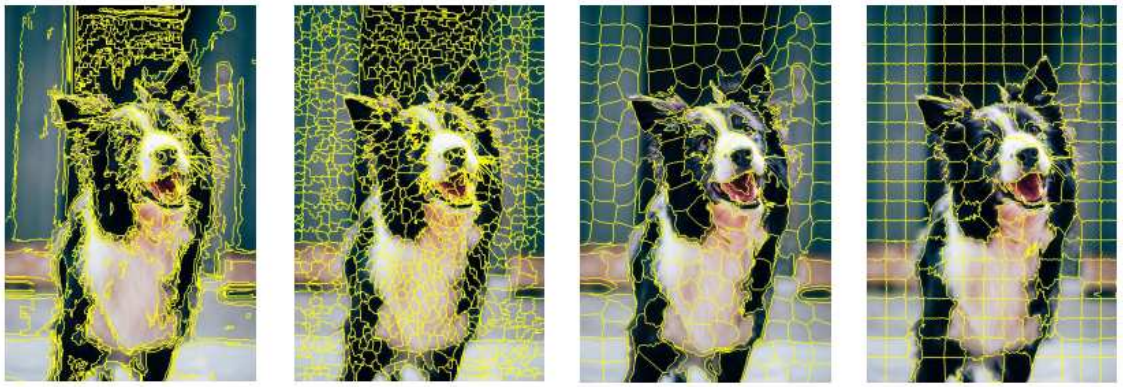

Figure 74: Algorithms of image segmentation

However, there are other segmentation algorithms that can be used. The image shows examples of Felzenszwalb (a), Quickshift (b), SLIC (c), and Compact-Watershed (d). These algorithms use different segmentation criteria, which may be more or less related to the geometric composition of the image compared to an absolute grid-based arrangement. We can see that the Felzenszwalb algorithm relies almost exclusively on the geometric composition of the image, while Compact-Watershed favors grid-based segmentation.
It is worth noting that the SHAP algorithm uses grid-based segmentation, making it more similar to the image (d) shown in the figure. In the experiments conducted in this work, I used LIME with its default partitioning algorithm, Quickshift (b), and compared it with SLIC (Simple Linear Iterative Clustering) (c).

6.4.1 Quickshift

Quickshift is an unsupervised clustering algorithm. It is based on the idea of estimating the local density of points in the feature space and identifying clusters by following the local maxima of the density.

Theoretical Basis

The algorithm uses *Kernel Density Estimation* (KDE) to estimate the density $p(x)$ of each point x in the feature space. The density is calculated as:

$$p(x) = \sum_{x_i \in \mathcal{N}(x)} K(x, x_i) \tag{17}$$

where:

- x is the point of interest.
- $\mathcal{N}(x)$ represents the set of points near x within a maximum radius max_dist.
- $K(x, x_i)$ is a Gaussian kernel defined as:

$$K(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{18}$$

  with $\|x - x_i\|$ representing the Euclidean distance between points x and x_i, and σ controlling the width of the kernel.

Algorithm Description

Quickshift operates by shifting each point x toward a neighbor x' with a higher density. This process iterates until all points converge to a local density maximum. The update rule is given by:

$$x' = \arg\max_{x_i \in \mathcal{N}(x)} p(x_i), \tag{19}$$

where $p(x_i)$ represents the density estimated at point x_i.

Algorithm 4 Quickshift Algorithm

Data: Dataset $D = \{x_1, x_2, \ldots, x_n\}$, parameters σ, max_dist
Result: Cluster labels for each point
1. Compute the density $p(x)$ for each point x_i using the Kernel Density Estimation (KDE) formula:
$p(x_i) = \sum_{x_j \in \mathcal{N}(x_i)} \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$
2. For each point x_i, find the neighbor x_j with the highest density:
$x' = \arg\max_{x_j \in \mathcal{N}(x_i)} p(x_j)$
3. Move point x_i toward the neighbor x' with a higher density:
$x_i \leftarrow x'$
4. Repeat Step 2 until each point x_i converges to a local density maximum;
5. Assign points that converge to the same local maximum to the same cluster;

Key Parameters

The main parameters of Quickshift are:

- σ: The width of the Gaussian kernel, which controls sensitivity to local density variations.

- max_dist: The maximum radius to consider neighboring points. It limits connections between distant points and reduces noise.

The choice of these parameters greatly influences the clustering results.

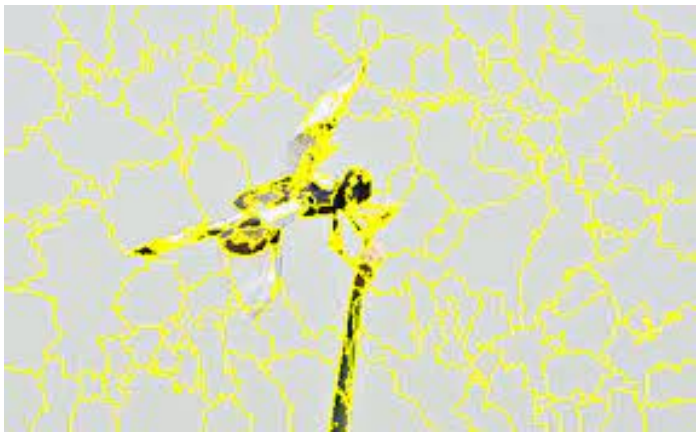

Figure 75: Segmentation of an image with quickshift with default hyperparameters ($\sigma = 5$, max_dist $= 10$). The superpixels are marked in yellow thick lines. The area inside the red rectangle is split up in 56 superpixels despite being quite homogeneous.

Quickshift on LIME

The analyzed image, shown in Figure 76, belongs to the COVID class, and true COVID prediction. The image shows the results of the analysis of an image obtained using LIME (Quickshift) and Grad-CAM analysis.

It is easy to see that the regions of the image highlighted as important by the two methods do not match at all. LIME highlights well-defined areas near the skeletal structure (shoulders and sternum), whereas Grad-CAM focuses on the lung area filled with opacities, which, based on previous observations, represents the correct behavior of the model.

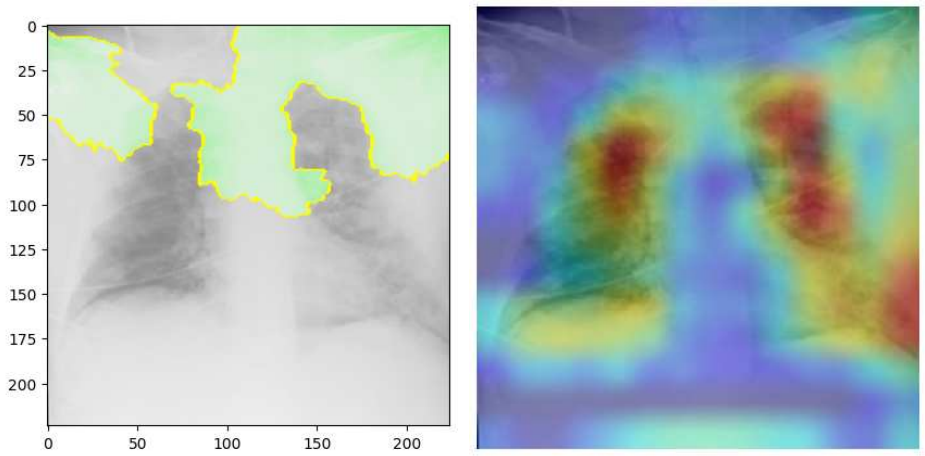

Figure 76: LIME Quickshift applied to a COVID image. The green areas correspond to the top 5 most influential superpixels for the model's classification.

It is clear that the areas related to the superpixels are defined by the borders and the geometrical structure of the image. The Quickshift algorithm outlines the perimeters of the skeletal structure and may not focus on the blurred areas related to the opacities, relevants for the classification.

6.4.2 SLIC

The SLIC (*Simple Linear Iterative Clustering*) algorithm is a clustering method widely used for image segmentation. It is based on a variation of the *k-means* algorithm, designed to operate in the spatial and color domains, creating *superpixels*: connected regions of pixels with similar characteristics.

Theoretical Basis

SLIC divides an image into segments or clusters based on:

- **Color:** pixels with similar colors tend to be grouped together.
- **Space:** pixels close to each other in space are grouped together.

The algorithm operates in the *Lab* color space (or another perceptually uniform color space, such as *RGB*) combined with spatial coordinates (x, y).
The *Lab* color space is a perceptually uniform space that represents colors based on three main components:

- L: Relative lightness of the color, ranging from 0 (black) to 100 (white).

- a: Green-red component, where negative values indicate green and positive values indicate red.
- b: Blue-yellow component, where negative values indicate blue and positive values indicate yellow.

The distance d_{lab} measures the difference between two colors in the Lab space. Given two pixels with coordinates (L_1, a_1, b_1) and (L_2, a_2, b_2), the distance is defined as:

$$d_{\text{lab}} = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2}. \tag{20}$$

Algorithm Workflow

1. **Initialization:**

   The image is divided into a regular grid with size S, which represents the desired size of the superpixels. The initial cluster centers are chosen as the central pixels of each region.

2. **Refinement of centroids:**

   Each centroid is moved to the pixel with the lowest intensity value within a small 3×3 window, to improve accuracy.

3. **Pixel assignment to clusters:**

   Each pixel is assigned to the nearest cluster based on a combined distance (Lab + spatial distance). The distance used is called the **combined distance**, defined as:

   $$D = \sqrt{d_{\text{lab}}^2 + \left(\frac{d_{\text{xy}}}{S}\right)^2 \cdot m^2}, \tag{21}$$

   where:

   - d_{lab} is the Euclidean distance in the Lab color space.
   - d_{xy} is the Euclidean spatial distance.
   - S is the desired superpixel size.
   - m is a parameter controlling the relative importance of color and spatial proximity. Higher values of m give more emphasis to spatial proximity.

4. **Centroid update:**

   For each cluster, the centroid is recalculated as the mean of the assigned pixels (in both Lab and spatial domains).

5. **Iterations:**

   Steps 3 and 4 are repeated until convergence, i.e., when the clusters do not change significantly between iterations.

6. **Superpixel connectivity:**

   After assignment, SLIC applies a process to connect disconnected pixels within a superpixel, ensuring that each region is connected.

Key Parameters

- S: The initial size of the superpixels, determining the total number of clusters. Smaller values of S result in smaller and more numerous superpixels.
- m: The compactness factor that balances the importance of color and space. Higher values produce more compact and spatially homogeneous clusters.

SLIC on LIME

The Simple Linear Iterative Clustering (SLIC) algorithm segments an image into superpixels using an optimized version of the K-Means algorithm.
How it works:

1. **Initialization:** The image is divided into a regular grid, and an initial seed is calculated for each superpixel.
2. **Assignment:** Each pixel is assigned to the nearest seed based on a combined distance that considers both color space (chromatic similarity) and Euclidean space (spatial proximity).
3. **Update:** The seeds are recalculated as the centroid of the assigned pixels.
4. **Iteration:** The assignment and update process is repeated until convergence.

The superpixels are then modified (e.g., obscured) to evaluate their impact on the model's prediction, determining which areas of the image are most influential for classification.

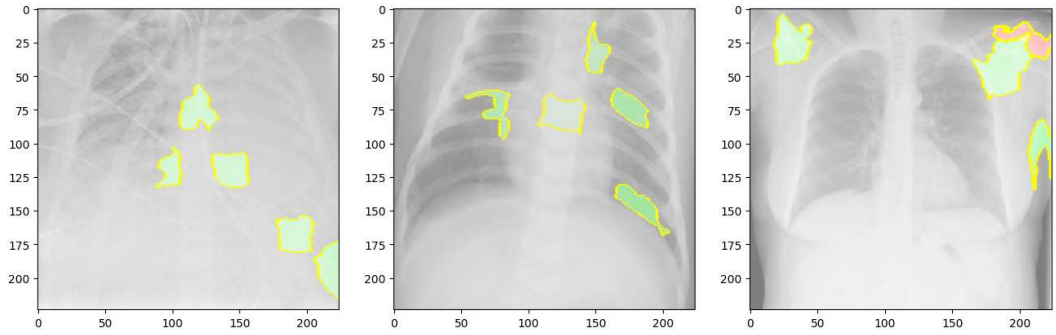

Figure 77: The areas highlighted in green correspond roughly to the patterns identified in the Grad-CAM chapter. From left to right, the first image is a true covid prediction, the second one is a true viral pneumonia prediction and the third one is a true normal prediction

SLIC is generally more suitable than Quickshift in cases where the segmentation process needs to focus on **compactness**, **regularity**, and **spatial structure**. Here are the scenarios where SLIC is more appropriate:

1. **Control Over Compactness and Regularity:**
   SLIC provides a parameter to control the **compactness** of superpixels. This allows the creation of uniformly shaped and sized superpixels, making it ideal for tasks requiring structured segmentation, such as medical imaging or satellite image analysis. Quickshift, by contrast, generates superpixels based on density and local gradients, which can lead to irregular and uneven shapes.

2. **Focus on Spatial Relationships**
   SLIC incorporates **spatial proximity** into its distance measure, making it effective for tasks that require **clear spatial boundaries**, such as detecting objects or regions based on geometric properties.

6.4.3 Limitations of LIME in Model Analysis

Regardless of the algorithm used, the results obtained for analyzing the performance of the image classification model (COVID-19, viral pneumonia, and normal) are neither accurate nor reliable. In fact, when comparing the Grad-CAM activation maps (which represent the general behavior of the model) with the superpixels identified as most relevant for classification, there are few overlaps, and in some cases, no correspondence at all.

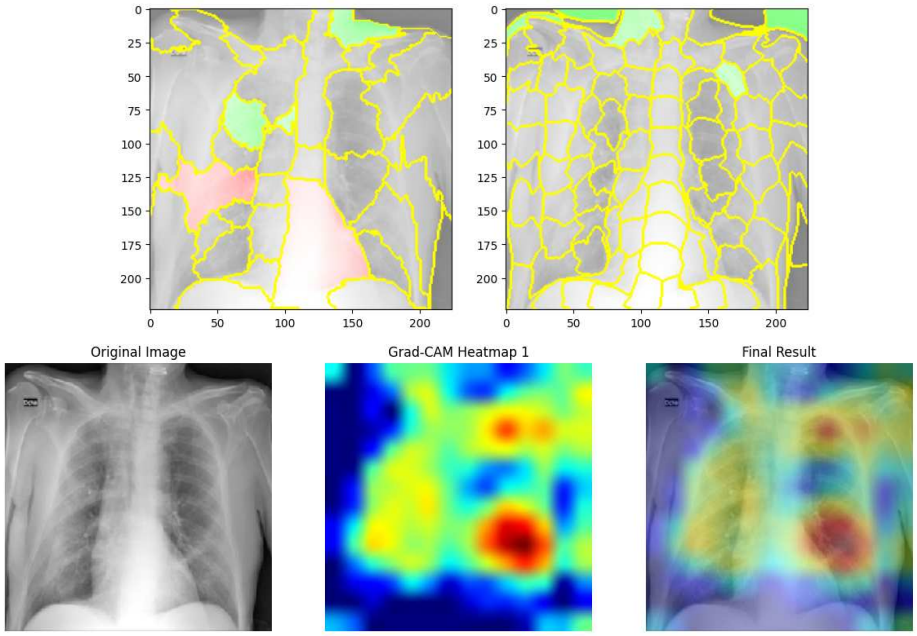

Figure 78: The first two images represent the Quickshift and SLIC analysis of the X-ray. The third image is the original X-ray, while the last two images show the Grad-CAM analysis and the overlay of the Grad-CAM result on the original image, respectively.

From Figure 78, it is evident that there is almost no correspondence between the areas with significant Grad-CAM activations and the positive superpixels identified by both Quickshift and SLIC.

When comparing Quickshift and Grad-CAM, only one superpixel (approximately at coordinates 75 on both axes) coincides with a weak activation in the Grad-CAM map.

Similarly, when comparing SLIC and Grad-CAM, only one superpixel (approximately at vertical coordinate 50 and horizontal coordinate 175) aligns with a strong activation in the Grad-CAM map.

In both cases, the vast majority of the Grad-CAM heatmap activations do not overlap with the superpixels derived from LIME.

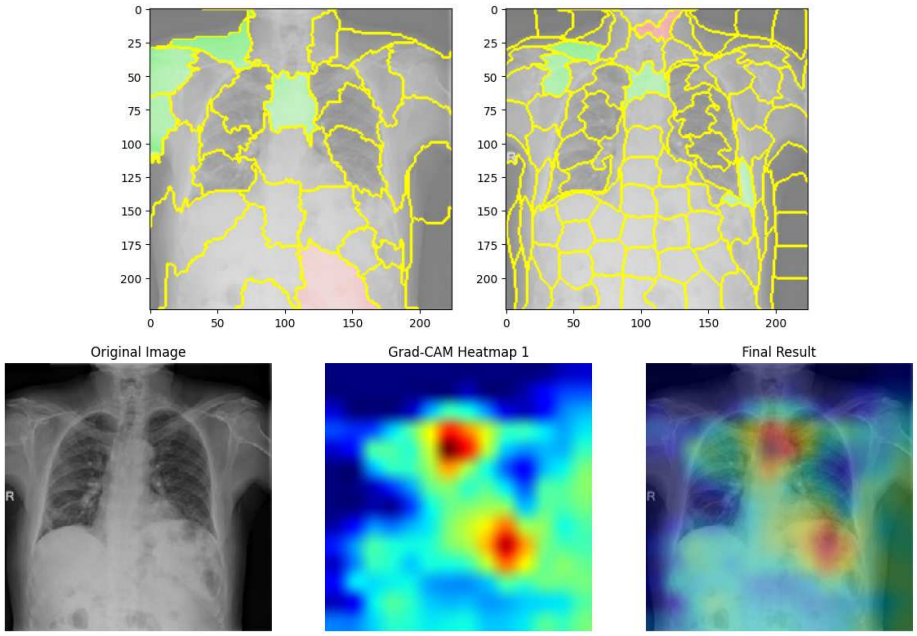

Figure 79: As the previous example (Figure 78), the first two images represent the Quickshift and SLIC analysis. The third image is the original X-ray, while the last two images show the Grad-CAM analysis and the overlay of the Grad-CAM result on the original image, respectively.

Here's another example (Figure 79). In both LIME analyses, a superpixel (located at vertical coordinate 50/75 and horizontal coordinate 100) can be observed, which corresponds to a region with high activation in the Grad-CAM analysis. However, the other positive superpixels identified in both Quickshift and SLIC tests point to regions of the image that are completely inactive in the Grad-CAM analysis.

The comparison of these two methods shows that LIME, regardless of the internal segmentation parameters, is capable of capturing some image patterns that are useful for classification. However, the randomness associated with generating perturbations likely introduces a fundamental unreliability in the explanations provided by the framework, particularly in this specific context.

Reasons for LIME's Unreliability

The unreliability of LIME in this specific context can be explained by several factors:

1. **Sensitivity to Perturbations:** LIME relies on local perturbations by obscuring or altering superpixels to evaluate their impact on the outcome. However,

many features relevant for classification might be distributed over large or non-localized regions, making it difficult for LIME to accurately identify significant areas.

2. **Dependence on Segmentation:** The choice of segmentation parameters (size and number of superpixels) strongly influences the results. Suboptimal parameters can fragment or overly aggregate important regions, reducing the quality of the explanations. In the examples analyzed, both Quickshift and SLIC performed poorly, failing to generalize sufficiently across all superpixels containing the most relevant features. Thus, it can be hypothesized that segmentation is not the primary issue.

3. **Nonlinear Interactions:** LIME assumes that classification is primarily influenced by local linear interactions. However, a model like VGG16 captures highly nonlinear interactions between features, which cannot be adequately explained with local perturbations. It can therefore be assumed that LIME's linear models are insufficient for extracting the desired features.

4. **Noise Introduced by Perturbations:** The process of obscuring or altering superpixels might introduce artificial noise that unnaturally alters the model's activations, leading to explanations that do not accurately reflect the model's behavior. These perturbations introduce randomness that can easily disrupt the symmetry of the image, which we observed to be crucial in the classification process.

These combined factors make LIME a less reliable method for analyzing a classification model in our context.

6.5 GradCAM, LIME, and SHAP Comparison

This section summarizes and compares the results obtained from the use of GradCAM, LIME, and SHAP. The results of each method are compared individually for an image related to a true COVID prediction.

GradCAM

GradCAM highlights that the image shows distinctive patterns in the central area of the chest, with less pronounced activations in the lower left section. The texture of the *ground glass opacities* can be recognized as the red area to the right of the sternum.

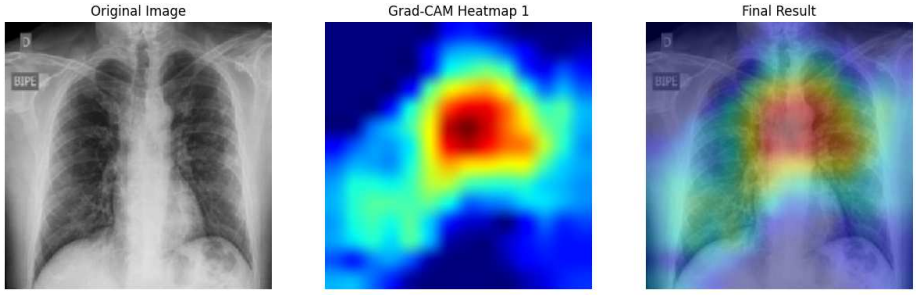

Figure 80: GradCAM Analysis

A detailed intralayer analysis for each VGG16 stack shows that the first stack fails to process relevant information, with the map completely inactive. The second stack effectively highlights the image's borders, outlining the geometric structure of the chest. The third stack shifts the focus from geometry to detecting pulmonary opacities. The fourth stack identifies symmetrical areas of influence in both lungs. Finally, in the last stack, the central area of the chest is activated, including the previously detected opacities.

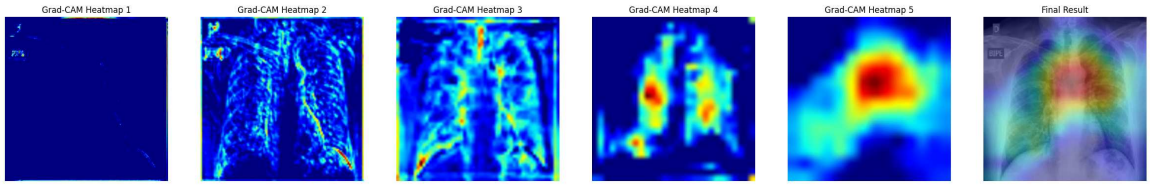

Figure 81: Intralayer Analysis with GradCAM

LIME

The LIME analysis shows uncertain results, likely due to the randomness in generating perturbations and the segmentation algorithms used.

The following images compare the analysis performed with two segmentation algorithms, Quickshift and SLIC. In this specific case, Quickshift produces more promising results, while SLIC completely diverges.

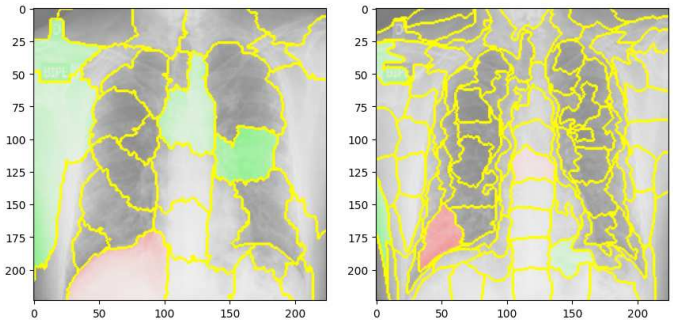

Figure 82: LIME Analysis, on the left segmentation with Quickshift, on the right with SLIC

This result highlights the unreliability of the method in this specific context, or at least in my case study.

SHAP

The SHAP analysis highlights the superpixels with positive relevance, which roughly correspond to those of GradCAM, though with a slightly offset distribution.

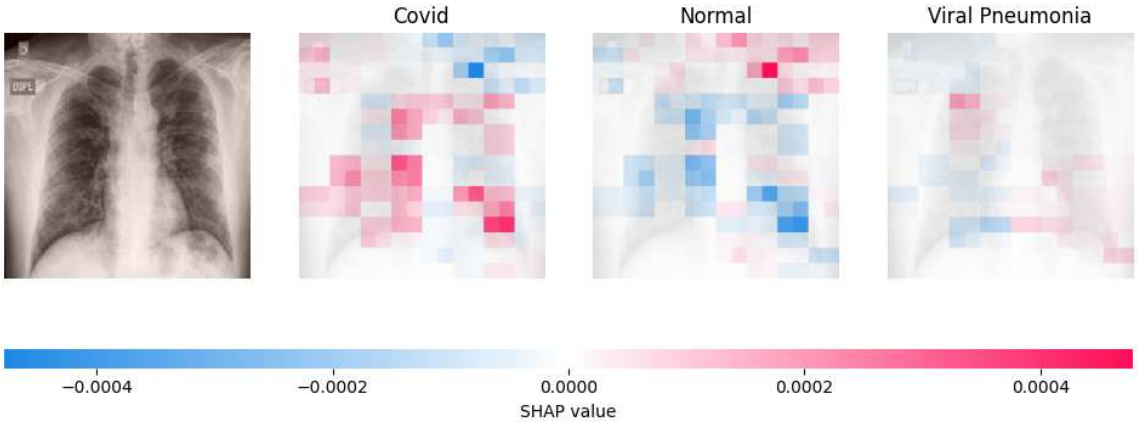

Figure 83: SHAP Analysis

By showing the Shapley values in a histogram, it is observed that positive values prevail for the COVID class. Therefore, the prediction is justified.

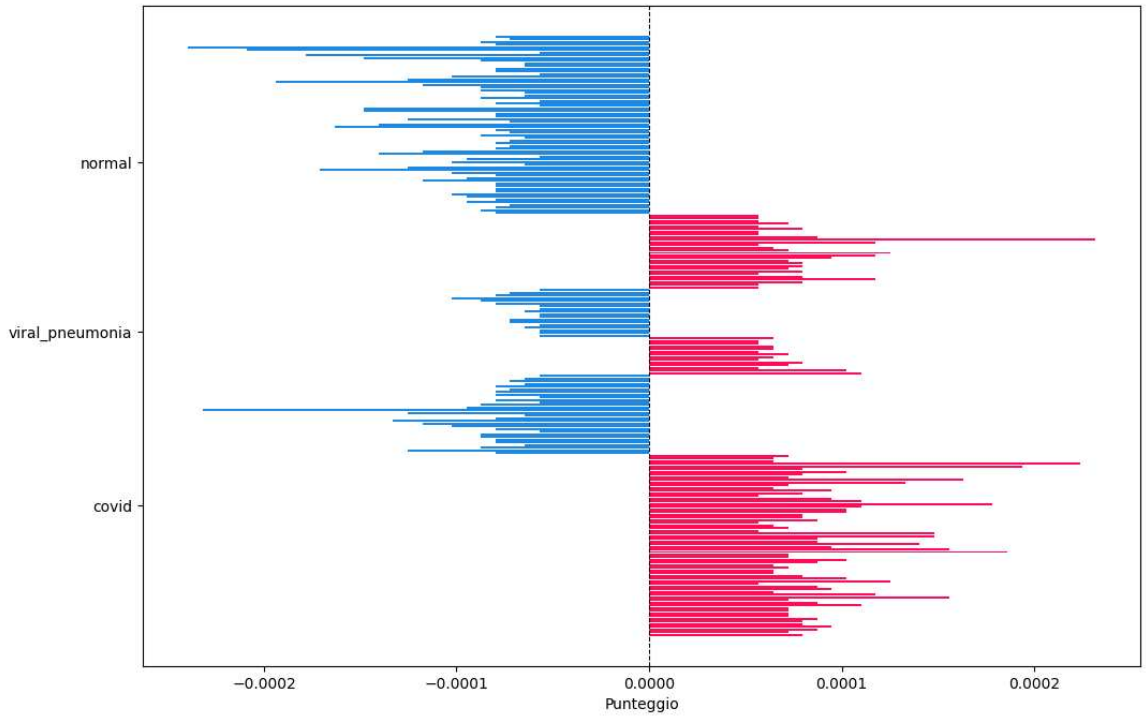

Figure 84: Distribution of Positive and Negative Shapley values by class

Summing these values yields the final score rankings.

1) `covid: 0.004086075949367087`
2) `viral_pneumonia: -8.734177215189901e-05`
3) `normal: -0.004154430379746837`

By calculating the distance between the individual score values for each class, the similarity of the individual prediction is obtained:

- **Viral Pneumonia and Normal**: Difference = 0.0040 (most similar).
- **Viral Pneumonia and COVID**: Difference = 0.0041.
- **COVID and Normal**: Difference = 0.0082 (most distinct).

6.6 *Chapter 6: Summary and Insights*

This section provides a detailed analysis of the classification results using interpretability methods: Grad-CAM, SHAP, and LIME. Each framework's strengths, limitations, and contributions to understanding the model's decision-making are discussed. Overall, the results from Grad-CAM and SHAP are consistent in most cases, allowing us to draw conclusions about the distinctive patterns of each class.

COVID-19 Cases: *Grad-CAM shows centralized and bilateral activations focused on the lung regions, particularly highlighting ground-glass opacities, a hallmark of COVID-19. These patterns are symmetrical and consistent across true predictions.*

Normal Cases: *Activations are concentrated along peripheral structures like shoulders, avoiding lung regions. This indicates that the model relies on the absence of anomalies to classify Normal cases.*

Viral Pneumonia Cases: *Unlike COVID-19, Grad-CAM for Viral Pneumonia shows asymmetrical and scattered activations, focusing on one lung or avoiding central thoracic regions. This reflects the diversity in viral pneumonia presentations.*

Grad-CAM Analysis: *Grad-CAM visualizes influential regions in input images, highlighting class-specific activation patterns across convolutional layers. Key observations include:*

- ***Distinctive patterns:*** *As already mentioned and listed earlier, Grad-CAM analysis generally revealed recurring patterns mostly in the COVID and normal classes, while for viral pneumonia, the distributions were not uniform.*
- ***Visualization of the decision-making process:*** *The Grad-CAM analysis applied to the internal layers of each VGG16 stack revealed interesting insights into how the network behaves during the decision-making process. In some cases, the early layers fail to extract low-level features, such as edges and contours in the image, resulting in completely inactive maps. On the other hand, the sequence of heatmaps shows how activations change and grow from layer to layer, starting from geometric lines and gradually highlighting entire areas of interest.*

***SHAP Analysis*:** *SHAP provides a quantitative breakdown of how individual pixels contribute to the classification for each class. Compared to Grad-CAM, SHAP evaluates contributions for all possible classes, offering a more granular understanding:*

- ***Distinctive Patterns:*** *The result of the SHAP analysis was also crucial for validating the results of Grad-CAM. Both methods converge on the same areas of interest, mutually confirming their effectiveness.*
- ***Importance of the negative contribution:*** *Unlike Grad-CAM, SHAP also uses negative values for classification, showing which areas of the image work against the classification into a specific class. The visual analysis of these areas reveals that, in many cases, two specific classes exhibit opposite activations, where one is positive and the other is negative, while the third class shows areas with lower intensity.*
- ***Ranking of prediction scores:*** *By analyzing the Shapley values, several analyses can be performed to easily show the ranking of the classification scores for each class. This way, it is possible to show how similar the prediction of one class is to another, based on the classification score calculated as the sum of the Shapley values.*

***LIME Analysis*:** *LIME approximates the decision boundary locally by perturbing input images and analyzing model responses. However, it underperforms compared to Grad-CAM and SHAP due to several limitations:*

- ***Perturbation noise:*** *The random perturbation of superpixels may introduce artifacts that do not align with the natural behavior of the model.*
- ***Sensitivity to segmentation parameters:*** *The choice of segmentation method and its parameters can significantly influence the results, potentially fragmenting or oversimplifying important image regions.*
- ***Over-reliance on linearity assumptions:*** *LIME assumes a linear relationship between input features and model output, which may not hold for complex, non-linear deep learning models.*

LIME's reliance on segmentation limits its effectiveness in medical imaging, making it less consistent than SHAP or Grad-CAM.

The analysis demonstrates that Grad-CAM and SHAP effectively interpret a high-performing classification model by highlighting its reliance on features such as ground-glass opacities and asymmetry. Conversely, LIME proves less effective due to factors like image segmentation and the randomness introduced by perturbations. Together, these methods provide valuable tools for interpreting deep learning models in complex medical imaging tasks.

Chapter 7

Conclusions and Future Work

This chapter concludes the journey undertaken in this thesis, which aimed to contribute to the topic of explainability. It used advanced analysis tools applied to complex technologies, such as convolutional neural networks, in a specific but real-world context: the classification of diagnostic clinical images. After discussing the theoretical aspects, the application context and the experimental results, this chapter provides a summary of the work done, followed by some final reflections and suggestions for possible future developments.

7.1 Summary of Contents

This thesis analyzes the functioning of convolutional neural networks (**CNNs**), with a specific focus on the classification process and the interpretability of the results. The primary goal is to explore the mechanisms driving the decisions of a **VGG16** neural network applied to a dataset of radiographic images (**X-rays**) categorized into three classes: *COVID-19*, *viral pneumonia*, and healthy images.

After a theoretical introduction to deep learning and its fundamental mathematical concepts, the study delves into the epidemiological and radiological context of **COVID-19**, highlighting its diagnostic differences compared to other respiratory diseases. Following this, the structure of the VGG16 network is examined, and the explainability frameworks employed — **GradCAM**, **LIME**, and **SHAP** — are thoroughly described.

In the experimental phase, the VGG16 network was trained using transfer learning from ImageNet. The performance analysis revealed a high classification accuracy, prompting a deeper investigation, not into classification errors, but into the network's **decision-making process** itself.

Finally, a comparative analysis of the explainability frameworks was conducted. GradCAM proved particularly effective in visualizing the network's areas of interest, identifying specific patterns for each class. SHAP confirmed GradCAM's findings while providing a numerical ranking of predictions and further insights into the similarities between COVID-19 and viral pneumonia. Conversely, LIME was less suited to this context, demonstrating significant limitations in representing areas of influence.

7.2 Analysis of Thesis Outcomes

To provide a thorough evaluation of the obtained results, it is necessary to revisit the objectives defined in the introduction of the research and compare them with the achieved outcomes. Although the primary goal of this thesis was to implement and evaluate advanced interpretability solutions for convolutional neural networks, the work also included an analysis and preliminary tasks that do not strictly align with the stated objectives. Therefore, the section related to the training of the VGG16 model will not be included in the objectives.

Visual Identification of Distinctive Patterns

The dataset used, related to pulmonary diseases, stands out for its complexity in extracting distinctive patterns. While the classification of images such as animals, road signs or, more generally, ImageNet multiclassification tasks, is easily verifiable by the human eye, the analysis of an X-ray requires the intervention of an experienced eye, capable of identifying subtle details and often imperceptible differences.

Moreover, in the radiological field, X-rays (RX) provide only a preliminary and indicative diagnosis of the condition. To identify truly discriminative features for classification purposes, a specialized doctor is often required to use computed tomography (CT), which offers a more detailed and precise view of pulmonary abnormalities.

For this specific problem, **GradCAM** proved to be a particularly useful tool, as it is able to identify the relevant areas for each classification with sufficient detail, highlighting the distinctive patterns necessary for an accurate diagnosis. The same areas of influence are displayed in **SHAP** as Shapley values with a positive contribution. The combination of these two methods provides a precise and reliable visual analysis of the distinctive patterns that led to a classification.

Analysis of the Decision-Making Process

The issue of explainability in deep neural networks concerns the difficulty of understanding and interpreting the decision-making process of complex models, often referred to as "black boxes." In the medical field, a system that autonomously makes diagnoses tends to raise concerns among the public, especially when the criteria behind the diagnoses are unclear. One of the primary goals of this thesis is to provide detailed explanations of the network's decision-making process.

The method that was fundamental for this experiment was the application of **GradCAM** in the intermediate layers of the neural network, with the goal of analyzing the decision-making process and feature extraction in more depth. This approach allows us to understand how the sequence of layers in the network, acting as filters, contribute to the extraction of relevant features, providing a more detailed view of the model's classification process.

However, in this thesis, GradCAM Intralayer is limited to providing a visual analysis, without conducting in-depth numerical quantitative analyses on the activation maps of the feature maps. This is certainly a limitation of the present work, which could be addressed and further explored in future studies.

Analysis of Classification Errors

The visual analysis of classification errors was not particularly exhaustive, as only 132 out of a total of 5640 images were misclassified. This small number does not allow for the extraction of generalizable concepts, but rather specific observations for each individual case. Additionally, the examination of these few errors revealed patterns indistinguishable from those of the other cases, without providing any indicative data that could clearly justify the errors.

Certainly, the nature of the dataset used may have influenced the less than satisfactory outcome of the error analysis. Previously, I conducted similar experiments on a neural network trained to distinguish between images of chihuahuas and muffins, inspired by a viral social media trend that humorously pointed out the resemblance between a dog's face and a muffin. The experiments provided insights into the types of errors.

In image 85, by comparing the final GradCAM result for the two images, the similarities between the patterns extracted from both classes become clearly evident. The network identifies as relevant features both the eyes and nose of the dog, as well as

the chocolate pieces of the muffin, thus creating an ambiguity that compromises the accuracy of the prediction.

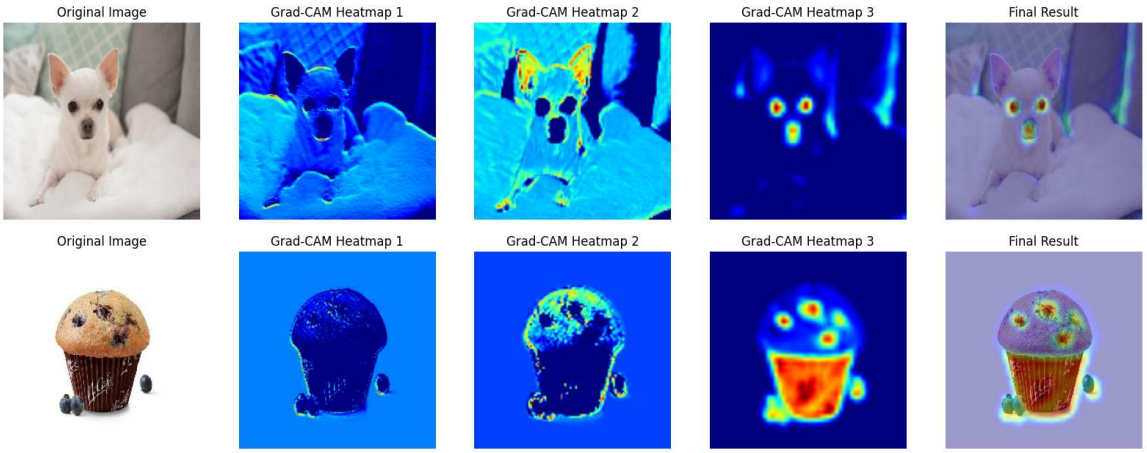

Figure 85: GradCAM Intralayer applied to a network with three convolutional layers trained to distinguish between muffins and chihuahuas.

Another common case of classification error involves the interference of the image background. In image 86, it is evident that the leaves in the background confuse the classification process, as clearly shown by the analysis performed using GradCAM.

Figure 86: An example of how the background influences classification, resulting in an error.

Although it proved challenging to conduct this type of analysis in the context of COVID, the numerical approach adopted with **SHAP** yielded interesting results. By

comparing the individual contributions (positive or negative) of each image region or their cumulative effect for each class, it is possible to quantify the similarity between classes in predictions.

In the experiment, the mean difference in rankings between predictions of the three classes was calculated using a dataset sample. The similarities between the predictions are as follows:

- **Viral Pneumonia and COVID**: Difference = 0.0016 (most similar).
- **COVID and Normal**: Difference = 0.0031.
- **Viral Pneumonia and Normal**: Difference = 0.0048 (most distinct).

In this way, the model does not indicate *where* it made an error, but rather *how much* it deviated. The analysis of class similarities proves to be a valuable tool for understanding how a model generates generalizations and associates patterns during its decision-making process. Instead of focusing solely on the final output, this analysis leverages the ranking scores assigned to each class, offering a deeper insight into the internal functioning of the network.

7.3 Future Developments

7.3.1 SHAP Intralayer - Intermediate Numerical Analysis

As previously explained, the experiments conducted on the decision-making process primarily provided results of a visual nature. A potential future development of this work could involve integrating a more detailed numerical analysis of the outputs at each convolutional layer within the network.

The proposed idea is to calculate a ranking score for each prediction at the level of each intermediate convolutional layer. In other words, this would involve developing an approach similar to SHAP but applied to every layer of the network—essentially a SHAP Intralayer. This numerical analysis would enable the observation of how the prediction rankings for each class evolve across the network's layers.

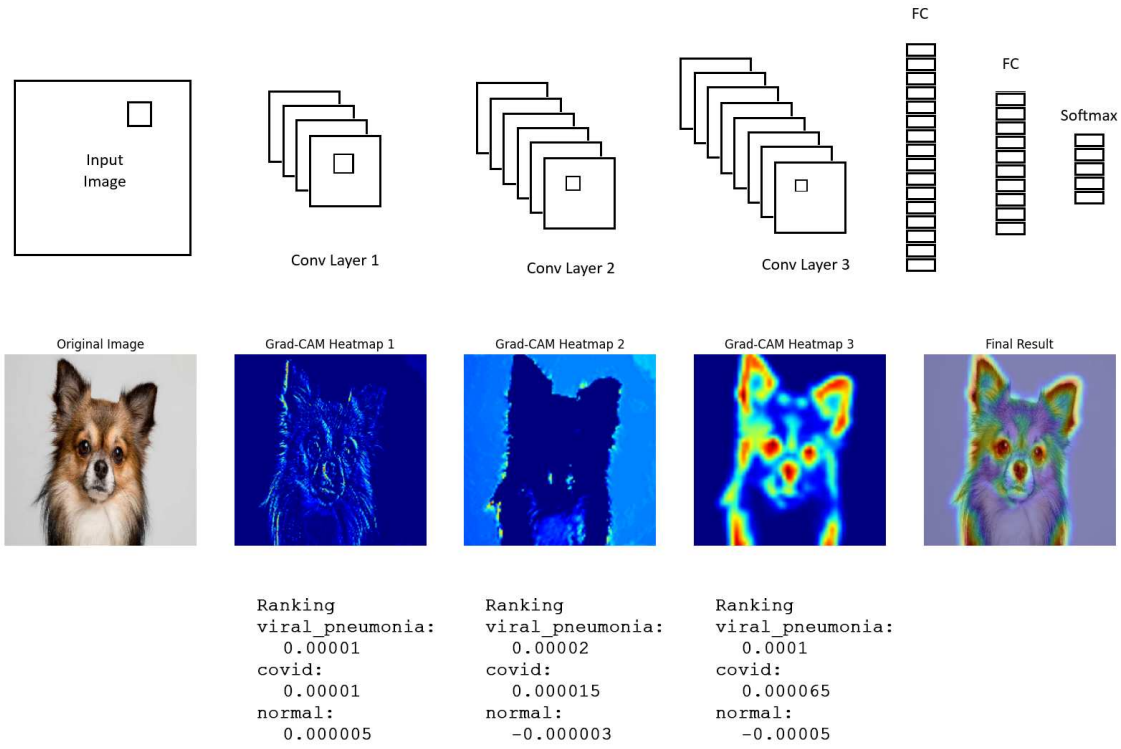

Figure 87: The image illustrates a hypothetical SHAP Intralayer analysis, where, after visualizing GradCAM Intralayer heatmaps, a ranking is computed at the end of each stack of the VGG16 network.

As a first observation, one could examine whether the ranking of the correct prediction increases or decreases at an intermediate layer compared to the final layer. This analysis can be useful for understanding if a deep model has too many layers. If the ranking of the correct class decreases in the later layers, it indicates that the network might be losing important information, focusing instead on less useful details in the subsequent layers. This could suggest that additional layers do not enhance the model but rather make it more confused. Reducing the number of layers could simplify the model, making it both more accurate and efficient.

7.3.2 Class Similarities on ImageNet - Multiclass Ranking Difference

In the results obtained in this work, it emerged that, on average, predictions related to *COVID* and *Viral Pneumonia* show strong similarities, while the *Normal* class

appears significantly more distinct. From a clinical perspective, this result is understandable, if not obvious, as it clearly highlights the sharp distinction between a pathological condition and a non-pathological one.

Given that, SHAP has proven to be a consistent tool for evaluating class similarities in predictions. It is particularly useful in classifiers with a large number of classes. For this reason, it might be interesting to conduct an in-depth analysis on ImageNet (4.1) using well-known models such as AlexNet and ResNet. The objective would be to compare the similarities between the classes in the dataset. Through this analysis, it would be possible to identify the classes with the greatest and the least similarity.

In this example, a numerical analysis using SHAP is applied to the ResNet50 network, with weights imported from the ImageNet dataset. The input image represents a frog, and the classifier produces the following outputs:

Figure 88: SHAP analysis of an input image of a frog. Below are the classes with the most significant contributions from the Shapley values.

The ranking of Shapley value contributions for each class, listed in order of relevance (excluding null values), is as follows:

1. Tailed Frog: 0.0088
2. Tree Frog: 0.0059
3. Frilled Lizard: 0.0005
4. Boa Constrictor: -0.0002

The similarity coefficients derived from the rankings are presented below, ordered from highest to lowest similarity:

- Boa Constrictor and Frilled Lizard: 0.0007
- Tailed Frog and Tree Frog: 0.0029
- Frilled Lizard and Tree Frog: 0.0054
- Boa Constrictor and Tree Frog: 0.0061
- Tailed Frog and Frilled Lizard: 0.0083
- Tailed Frog and Boa Constrictor: 0.0090

From this analysis, it can be understood that, although the predicted class is *Tailed Frog* due to its highest ranking and almost exclusively positive contributions, the classes with the greatest similarities are *Boa Constrictor* and *Frilled Lizard*.

7.3.3 Advanced Classification of Pulmonary Pathologies

The experiments conducted in this thesis focused on a generalized approach, limiting classification to three categories: absence of pathology, COVID-19, and viral pneumonia. While this approach proved effective for the specific objectives of the study, it is not entirely precise from a clinical perspective. In the medical field, there are various types of pneumonia and other pulmonary pathologies that could be classified more distinctly.

Pneumonias can primarily be divided into two major categories: **viral** and **bacterial**. The viral category includes subtypes such as COVID-19, influenza, respiratory syncytial virus (RSV), and varicella pneumonia, among others. The bacterial category includes pneumonias caused by bacteria like *Streptococcus pneumoniae* and *Haemophilus influenzae*. Additionally, other types of pulmonary conditions, such as lymphocytic pneumonia or pneumonia caused by *Pneumocystis jirovecii*, are often associated with immunocompromised conditions.

A more detailed analysis, such as one based on comparing similarities between classes in predictions, could be useful for better distinguishing among these types of pathologies. For instance, the approach recently discussed for ImageNet to calculate class similarities could be applied to classifiers with a larger number of categories. This would allow the identification of shared patterns or significant differences among classes, thereby improving diagnostic precision.

Moreover, this type of numerical analysis could have significant implications for medical research. Platforms like **Radiopaedia**, which collect and organize diagnostic images for specific conditions, could benefit from a model that not only classifies images but also provides insights into the relationships between different conditions. This approach could support the development of advanced diagnostic systems and assist radiologists in understanding shared or distinctive features among diverse pathologies.

7.3.4 COVIDNet-CT Analysis

As previously explained, from a diagnostic perspective, computed tomography (CT) scans are significantly more effective and useful than traditional X-rays (RX). However, training a convolutional neural network (CNN) on CT scan images presents several challenges.
In particular:

- **Data size**: CT images are three-dimensional (3D), with information distributed across multiple *slices* or sections. This significantly increases the amount of data to be processed compared to two-dimensional (2D) X-rays, requiring greater memory, computational power, and a more complex model design.
- **Dataset annotation**: CT images, being rich in detail, require precise and multidimensional annotations. This process is time-consuming and demands specific expertise from medical professionals, increasing the risk of errors or incomplete annotations.
- **Computational cost**: Training a CNN on CT images is significantly more computationally expensive compared to X-rays due to the higher complexity and data size.
- **Data representation**: The CNN must be designed to process 3D images or convert CT scans into 2D images without losing relevant information. This necessitates a complex architecture and meticulous preprocessing.

For these reasons, and due to evident time constraints, the focus of the current thesis has been solely on X-ray images.

Nevertheless, there is a project called **COVIDNet-CT**, developed in September 2020 by Hayden Gunraj, Linda Wang, and Alexander Wong, which leverages a deep convolutional neural network to detect COVID-19 cases using CT chest images. The network utilizes a dataset of 2D images, approximated from a larger set of 3D images, aiming to classify COVID-19-related pathologies. The network currently achieves an impressive accuracy of around 98%, suggesting that the dataset approximations are sufficiently robust. The analytical methods described in this thesis could be applied as explainability tools in a broader and more complex project, such as COVIDNet-CT.

Ringraziamenti

Vorrei fare un ringraziamento speciale a mio nonno. Nonno Igino è stato per me, come per molti altri credo, un esempio da seguire sia per la sua dedizione allo studio sia per la sua tenacia e vitalità che lo hanno sempre contraddistinto. Fin da quando ho memoria, Nonno Igino ha sempre avuto aspettative altissime su di me, ed oltre ad una responsabilità, per me questo è stato un grande onore. Credo che lui abbia avuto molta più fiducia in me di quanta io ne abbia mai avuta in me stesso. Probabilmente, è stato proprio questo il contributo più grande che mi ha permesso di andare avanti in questi anni, nonostante tutte le difficoltà. Dato che Nonno non può essere qui in questo momento, vorrei ringraziarlo per tutto questo.

Desidero ringraziare il Professore Nunzio Alberto Borghese. Oltre ad avermi seguito come relatore in questo periodo delicato, il Professor Borghese ha apprezzato con entusiasmo il mio impegno durante gli esami sostenuti con lui. Questo aspetto è stato importante per me, considerando che, non avendo mai avuto la possibilità di frequentare le lezioni, mi è stato impossibile vivere l'università come un luogo di condivisione sociale, sia con i colleghi studenti sia con i docenti.

Ringrazio i miei genitori che, oltre ad aver sempre riposto fiducia in me, mi hanno supportato e, soprattutto, sopportato nei momenti più difficili. Sebbene questa 'odissea' sia durata ben oltre le aspettative, non c'è mai stato un momento in cui abbia avvertito alcuna pressione da parte loro. Per tutto il percorso, mamma e papà hanno condiviso con me successi, soddisfazioni, delusioni e frustrazioni e posso dire con certezza di non essermi mai sentito solo.

Vorrei ringraziare mia nonna che si è sempre interessata ai miei studi in modo attivo. Nonna Franca è riuscita più volte a farmi vedere il lato artistico dell'informatica con articoli di giornale, libri e programmi televisivi. In questo percorso così complicato per me è stato fondamentale avere avuto questi spunti di riflessione e la sua preziosa presenza.

Desidero ringraziare in modo speciale zio Antonello, che non smette mai di insegnarmi e di supportarmi in ogni modo possibile. Zio mi ha aiutato in modo concreto e pragmatico in più situazioni riguardanti il percorso accademico, lavorativo e non solo. Ha contribuito in modo utile e con sincero interesse agli esperimenti condotti in questa tesi. La sua presenza e il suo carattere professionale, spiritoso e irriducibile mi hanno aiutato molto in questo percorso.

Vorrei ringraziare molto Giulia che mi è stata sempre vicino in questa fase finale della mia esperienza universitaria. Anni fa non avrei mai immaginato di arrivare a questo traguardo senza trovarmi in un totale crollo psicologico. Pensandoci ora, il merito va a lei e alla sua capacità di rendere belli e leggeri anche i momenti più difficili nella vita quotidiana.

Un ringraziamento speciale va a Paolo, che è stato per definizione l'amico che mi è stato vicino per tutto il percorso universitario, triennale e magistrale. Aver potuto condividere con lui tutti gli aspetti, positivi e negativi, della vita universitaria è stato impagabile. Penso che ognuno abbia una persona su cui poter contare sempre. Per me è lui.

Ringrazio Laura che invece è stata l'amica con cui ho condiviso tutta l'esperienza universitaria e lavorativa. Con lei mi sono potuto consultare su qualsiasi aspetto tecnico inerente al mio percorso accademico. Laura è stata fondamentale in più occasioni per affrontare ostacoli che, senza di lei, non sarei mai riuscito a superare.

Desidero ringraziare la mia famiglia, zii e cugini De Pascale e Pineschi, che, da sempre, sono stati coprotagonisti della mia vita. Con loro ho condiviso qualsiasi esperienza possibile e immaginabile e, grazie a loro, mi sono sempre sentito parte di una comunità. Ognuno di loro è stato fondamentale, e devo questo traguardo al loro contributo, unico e indispensabile.

Vorrei ringraziare tutto il team Co.Ge.Si. che mi ha dato l'opportunità di intraprendere un percorso lavorativo professionale e stimolante senza avermi mai sottratto tempo allo studio. In particolare ringrazio Antonio che è sempre stato comprensivo e disponibile riguardo a qualsiasi necessità, con me come con ogni suo dipendente.

Vorrei infine ringraziare tutti gli amici e le persone che mi sono state vicine in questi anni. Anche se non hanno contribuito direttamente al mio percorso accademico, il loro supporto si è manifestato nei gesti quotidiani, nei ricordi condivisi e nelle esperienze vissute insieme. Per questo desidero ringraziarli di cuore.

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, MIT Press, 2016.

[2] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, *Deep Learning*, Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[3] Selvaraju, R.R. et al., *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*, International Journal of Computer Vision, 2016.

[4] K. Simonyan, A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, International Conference on Learning Representations, 2015.

[5] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *Why Should I Trust You? Explaining the Predictions of Any Classifier.*

[6] Ludwig Schallner, Johannes Rabold, Oliver Scholz and Ute Schmid (2019) *Effect of Superpixel Aggregation on Explanations in LIME – A Case Study with Biological Data*

[7] Vedaldi, A., Soatto, S.: *Quick shift and kernel methods for mode seeking.* In: European Conference on Computer Vision. pp. 705–718 (2008)

[8] Zhiqiang Xia, Ce Zhu, Zhengtao Wang, Qi Guo, Yipeng Liu. *Every Filter Extracts A Specific Texture In Convolutional Neural Networks* 18 Aug 2016

[9] Lundberg, S. M., & Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions.* In *Advances in Neural Information Processing Systems*, 30.

[10] Ming-Yen Ng, Elaine Y. P. Lee, Jin Yang, Fangfang Yang, Xia Li, Hongxia Wang, Macy Mei-sze Lui, Christine Shing-Yen Lo, Barry Leung, Pek-Lan Khong, Christopher Kim-Ming Hui, Kwok-yung Yuen, Michael D. Kuo. *Imaging Profile of the COVID-19 Infection: Radiologic Findings and Literature Review.* Radiology: Cardiothoracic Imaging, 2(1), e200034, 2020

[11] Hayden Gunraj, Linda Wang, Alexander Wong *COVIDNet-CT: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest CT Images*, 8 Sep 2020

[12] Wynants L, Van Calster B, Collins GS, Riley RD, Heinze G, Schuit E, Bonten MMJ, Dahly DL, Damen JAA, Debray TPA, de Jong VMT, De Vos M, Dhiman P, Haller MC, Harhay MO, Henckaerts L, Heus P, Kammer M, Kreuzberger N, Lohmann A, Luijken K, Ma J, Martin GP, McLernon DJ, Andaur Navarro CL, Reitsma JB, Sergeant JC, Shi C, Skoetz N, Smits LJM, Snell KIE, Sperrin M, Spijker R, Steyerberg EW, Takada T, Tzoulaki I, van Kuijk SMJ, van Bussel B, van der Horst ICC, van Royen FS, Verbakel JY, Wallisch C, Wilkinson J, Wolff R, Hooft L, Moons KGM, van Smeden M. *Prediction models for diagnosis and prognosis of covid-19: systematic review and critical appraisal.* BMJ. 2020 Apr 7;369:m1328. doi: 10.1136/bmj.m1328. Update in: BMJ. 2021 Feb 3;372:n236. doi: 10.1136/bmj.n236. Erratum in: BMJ. 2020 Jun 3;369:m2204. doi: 10.1136/bmj.m2204. PMID: 32265220; PMCID: PMC7222643.

[13] De Pascale A, *"Riflessioni radiologiche sulla infezione da SARS-CoV2" - Incontri clinici di ematologia ed oncologia di "ROMA CENTRO".* 2020

[14] Cozzi D, Cavigli E, Moroni C, Smorchkova O, Zantonelli G, Pradella S, Miele V. *Ground-glass opacity (GGO): a review of the differential diagnosis in the era of COVID-19.* Jpn J Radiol. 2021 Aug;39(8):721-732. doi: 10.1007/s11604-021-01120-w. Epub 2021 Apr 26. PMID: 33900542; PMCID: PMC8071755.

[15] Baptiste Broto, François Bachoc, Laura Clouvel, Jean-Marc Martinez *Block-diagonal covariance estimation and application to the Shapley effects in sensitivity analysis* 13 Feb 2020