# UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

**Escuela de Ciencias Biológicas e Ingeniería**

# TÍTULO: Using artificial intelligence and X-ray images to train and predict COVID-19 and Pneumonia: Tool for diagnosis and treatment

Trabajo de integración curricular presentado como requisito
del título de INGENIERO BIOMÉDICO

**Autor:**

Juárez González Bryan Patricio

**Tutor:**

PhD - Almeida Galárraga Diego Alfonso

Urcuquí, Diciembre 2024

# Autoría

Yo, **BRYAN PATRICIO JUÁREZ GONZÁLEZ**, con cédula de identidad 1723700256, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.
Urcuquí, Diciembre 2024.

Bryan Patricio Juárez González
CI: 1723700256

# Autorización de publicación

Yo, **BRYAN PATRICIO JUÁREZ GONZÁLEZ**, con cédula de identidad **1723700256**, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urcuquí, Diciembre 2024.

Bryan Patricio Juárez González
CI: 1723700256

# Dedication

*“To my parents, Patricio and Janeth, my brothers, and my grandparents for all their support and patience.”*

Bryan Patricio Juárez González

# Acknowledgments

Thanks to the valuable support given by my family. To my friends in Yachay who helped me overcome difficult times, giving me motivation to keep going. Thanks to all the professors in Yachay who taught me with patience and diligence valuable and long-lasting knowledge. Being a student at this exceptional university has forged my character, both professionally and academically. Last but not least, I want to give a special thanks to my advisor, Diego Alfonso Almeida Galarraga, Ph.D., for giving me important insights and sharing his academic experience with me throughout this work.

Bryan Patricio Juárez González

# Resumen

Ecuador presentó un pico de casos de COVID-19 en enero de 2022, con 890,541 casos confirmados y 35,658 muertes. Además, la influenza y la neumonía, no relacionadas con el COVID-19, estuvieron entre las cinco principales causas de muerte en Ecuador de 2019 a 2020. Chamorro y su equipo mencionaron que los métodos de radiografía de tórax y tomografía computarizada de tórax se utilizan para diagnosticar el COVID-19. Sin embargo, Wong et al. dicen que las radiografías de tórax no son tan sensibles como las tomografías computarizadas de tórax para detectar el COVID-19. Esto es cierto solo cuando no se utiliza inteligencia artificial. En el artículo de Tahir et al., informan que los modelos que utilizan U-net y redes neuronales obtienen sensibilidades por encima del 99% y una especificidad perfecta del 100% utilizando U-Net ++ y ResNet18 con 33,920 imágenes de radiografías de tórax. Un sistema de detección rápida de los síntomas podría haber ayudado a priorizar a los pacientes que necesitaban unidades de cuidados intensivos, reduciendo un porcentaje significativo de muertes. En nuestro trabajo, hemos demostrado que la inteligencia artificial combinada con radiografías de tórax puede tener una precisión binaria por encima del 98% de predicción utilizando transfer learning con las redes neuronales Xception, VGG16 y VGG19. Utilizamos un total de 27,052 imágenes de radiografías de tórax, divididas en las clases COVID, Opacidad Pulmonar, Neumonía Viral y Normal. Balanceamos el conjunto de datos y utilizamos otras técnicas de optimización. Utilizamos las plataformas Google Colab y Kaggle para compilar los códigos con GPUs T4. Obtuvimos un excelente puntaje F1 del 98.53% para el modelo de transferencia de aprendizaje VGG19 utilizando las clases COVID y Normal. VGG19 fue entrenado y probado en clasificación multiclase, obteniendo un puntaje F1 por encima del 89%. Hemos demostrado que VGG19 y otros modelos pueden ser herramientas muy útiles para hospitales públicos y privados para ayudar en el diagnóstico del COVID-19. El trabajo futuro podría incluir la mejora de nuestros resultados de rendimiento utilizando un conjunto de datos más grande y la segmentación U-net.

***Palabras Clave***: Inteligencia artificial, COVID-19, redes convolucionales, visión por computadora, neumonía, rayos-X de pecho, transferencia de aprendizaje, Xception, VGG16, VGG19, U-net.

# Abstract

Ecuador presented a peak of COVID-19 cases in January 2022, with 890,541 confirmed cases and 35,658 deaths. Also, influenza and pneumonia, not related to COVID-19, were among the top 5 causes of death in Ecuador from 2019 to 2020. Chamorro and his team mentioned that chest X-rays and chest CT scan methods are used to diagnose COVID-19. However, Wong et al. say that chest X-rays are not as sensitive as chest CT scans for detecting COVID-19. This is true only when AI is not used. In the article by Tahir et al., they report models using U-net and neural networks obtaining sensitivities above 99% and a perfect specificity of 100% using U-Net ++ and ResNet18 using 33,920 chest X-ray images. A rapid detection system of the symptoms might have helped prioritize patients who needed intensive care units, reducing a significant percentage of deaths. In our work, we have demonstrated that AI combined with chest X-rays can have a binary accuracy above 98% of prediction using transfer learning Xception, VGG16, and VGG19 neural networks. We used a total of 27,052 chest X-ray images, divided into COVID, Lung Opacity, Viral Pneumonia, and Normal classes. We balanced the dataset and used other optimization techniques. We used the Google Colab and Kaggle platforms to compile the codes with T4 GPUs. We obtained an excellent F1-Score of 98.53% for the VGG19 transfer learning model using COVID and Normal classes. Vgg19 was trained and tested on multiclass classification, obtaining an F1-Score above 89%. We have demonstrated that VGG19 and other models can be very useful tools for public and private hospitals to help with the diagnosis of COVID-19. Future work might include improving our performance results using a larger dataset and U-net segmentation.

***Keywords***: Artificial intelligence, COVID-19, convolutional networks, computer vision, Pneumonia, chest X-ray, transfer learning, Xception, VGG16, VGG19, U-net.

# Contents

**Resumen** vi

**Abstract** vii

**Contents** viii

**List of Tables** x

**List of Figures** xii

**List of Equations** xiv

**Abbreviations** xv

**1 Introduction** 1
- 1.1 Thesis Overview 1
- 1.2 The Pneumonia Outbreak Problem 1
- 1.3 Thesis Justification 2
- 1.4 Contribution to Society 3
- 1.5 Thesis Objectives 3
  - 1.5.1 Work General Objective 3
  - 1.5.2 Thesis Specific Objectives 3

**2 Theoretical Framework** 5
- 2.1 Overview of Artificial Intelligence Technology 5
  - 2.1.1 Machine Learning as a Subset of Artificial Intelligence 6
  - 2.1.2 Applications of Deep Learning Algorithms 6
  - 2.1.3 Computer Vision Applications 6
- 2.2 The use of Artificial Neural Networks 7
  - 2.2.1 The Loss function During Training 9
  - 2.2.2 Activation Functions in Convolutional Neural Networks 10
  - 2.2.3 Convolution Operator Technique Example 11
  - 2.2.4 Receptive Field and Feature Map in Artificial Intelligence 13
  - 2.2.5 Example of Padding Method 14
  - 2.2.6 Striding Method Applied to CNN 16
- 2.3 Convolutional Neural Networks Architecture Review 17
  - 2.3.1 Dropout Technique in Training 18
  - 2.3.2 Convolutional Layer Definition 18

2.3.3 The Pooling Layer . . . 19
2.3.4 The Fully Connected Layer . . . 19
2.4 Remarkable Convolutional Neural Networks . . . 19
2.4.1 Visual Geometry Group Neural Networks . . . 19
2.4.2 Inception Neural Network . . . 21
2.4.3 Google Xception . . . 21
2.5 Using the Transfer Learning Technique . . . 22
2.6 Steps in Training a Convolutional Neural Network . . . 24
2.6.1 Overview of Deep Learning Frameworks . . . 24
2.6.2 Using Public and Private Datasets . . . 24
2.6.3 Preparing a Dataset Techniques During Training . . . 26
2.6.4 Hyperparameters in Training a CNN . . . 28
2.6.5 Analyzing Results Tools and Techniques During Training . . . 30

**3 State of the Art** **33**
3.1 Pneumonia Infection . . . 33
3.1.1 Diagnostic Tests for Pneumonia Infection . . . 33
3.1.2 Pneumonia Bacterial Treatment . . . 35
3.2 Covid-19 Virus Review . . . 37
3.2.1 Diagnosing Covid-19 Techniques . . . 37
3.2.2 Covid-19 Treatment During the Pandemic . . . 38
3.3 Historical Perspectives on the Use of X-ray Images . . . 39
3.3.1 Diagnosis of Pneumonia and other diseases . . . 39
3.3.2 Radiotherapy . . . 39
3.4 Existing Methods for COVID-19 Detection Using X-ray Images Review . . . 40
3.4.1 Using Chest X-rays as a Diagnostic Method . . . 40
3.4.2 Diagnosis Using Chest Computerized Tomography Scan . . . 40
3.4.3 The Sensitivity of Chest X-ray vs CT Scan . . . 40
3.5 Artificial Intelligence in Chest X-rays with COVID-19 . . . 41
3.5.1 High Precision Models Using Chest X-rays . . . 42
3.5.2 Segmentation Algorithms . . . 42
3.5.3 Using Contrast Limited Adaptive Histogram Equalization . . . 43
3.5.4 COVID-19 Chest X-ray Public and Private Datasets . . . 45

**4 Methodology** **48**
4.1 Methodology Summary . . . 48
4.2 Public Dataset . . . 51
4.2.1 Balancing and Augmentation Techniques . . . 51
4.2.2 Additional Optimizers . . . 52
4.3 Transfer Learning Architectures . . . 53
4.4 Optimizers and Loss Function . . . 57
4.4.1 Default Adam Values . . . 57
4.4.2 Binary and Categorical Cross-entropy . . . 58
4.4.3 Fine Tuning and Metrics . . . 59
4.5 Software and Hardware Used in Our Experiment . . . 59

**5 Results** **61**
5.1 Results Experiment Introduction . . . . . . . . . . . . . . . . . . . . . . . . 61
5.2 Details of the Dataset Used for Three Stages . . . . . . . . . . . . . . . . . . 61
5.2.1 Stage 1: Binary Classification Normal and Covid . . . . . . . . . . 61
5.2.2 Stage 2: 3-Class Classification Normal, Covid, and Viral Pneumonia 62
5.2.3 Stage 3: 4-Class Classification Normal, Covid, Viral Pneumonia, and Lung-Opacity . . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
5.3 Transfer Learning Models Architectures . . . . . . . . . . . . . . . . . . . . . 64
5.4 Training: Convergence Analysis . . . . . . . . . . . . . . . . . . . . . . . . 65
5.4.1 Stage 1: Binary Classification Xception, VGG16 and VGG19 . . . . 65
5.4.2 Stage 2: 3-Class Classification VGG19 . . . . . . . . . . . . . . . . 68
5.4.3 Stage 3: 4-Class Classification VGG19 . . . . . . . . . . . . . . . . 69
5.5 Stage 1: Binary Classification Performance metrics . . . . . . . . . . . . . 70
5.5.1 Binary Classification Summary . . . . . . . . . . . . . . . . . . . . 73
5.5.2 Stage 2: 3-Class Classification VGG19 . . . . . . . . . . . . . . . . 75
5.5.3 Stage 3: 4-Class Classification for VGG19 . . . . . . . . . . . . . . 76
5.6 Grad-CAM on Our VGG19 Model . . . . . . . . . . . . . . . . . . . . . . . 76
5.7 VGG19 Heat Maps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 79

**6 Discussion** **83**
6.1 VI. Discussion of Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
6.1.1 Results of the Xception Neural Network . . . . . . . . . . . . . . . 83
6.1.2 Results of the VGG16 . . . . . . . . . . . . . . . . . . . . . . . . . 83
6.1.3 Results of the VGG19 . . . . . . . . . . . . . . . . . . . . . . . . . 84
6.1.4 Work Limitations in Our Experiment . . . . . . . . . . . . . . . . . 85

**7 Conclusions** **86**
7.1 Thesis General Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . 86
7.2 Future Work Insights . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86

**Bibliography** **88**

**Appendices** **99**
7.3 Appendix 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
7.4 Appendix 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102

# List of Tables

3.1 Sensitivity and Specificity of Different Tests for Pneumonia Detection. . . . 33
3.2 Sensitivity and Specificity of Different Tests for Pneumonia Detection. . . . 34
3.3 Pneumonia Legionella spp. Detection Tests. . . . . . . . . . . . . . . . . . . 34

3.4 Drugs used to Treat Bacterial Pneumonia. . . . . . . . . . . . . . . . . . . . 36
3.5 COVID-19 most Used Tests . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
3.6 COVID-19 Vaccines Characteristics . . . . . . . . . . . . . . . . . . . . . . . 39
3.7 Accuracy ResNet50, InceptionV3, and Inception-ResNetV2 Models . . . . . 42
3.8 Accuracy Transfer Learning Models . . . . . . . . . . . . . . . . . . . . . . . 44
3.9 Accuracy Transfer Learning Models VGG16 and VGG19 . . . . . . . . . . . 44
3.10 *COVID-QU-Ex Dataset* tested on state-of-the-art neural networks using segmentation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46

4.1 Dataset characteristics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.2 Specifications of the Xception model layers. . . . . . . . . . . . . . . . . . . 54
4.3 Specifications of the VGG19 model layers. . . . . . . . . . . . . . . . . . . . 57

5.1 Binary Classification Arrangement of Datasets . . . . . . . . . . . . . . . . 62
5.2 Three-Class Classification Arrangement of Dataset . . . . . . . . . . . . . . 62
5.3 Four-Class Classification Arrangement of Dataset . . . . . . . . . . . . . . 63
5.4 Transfer learning training parameters . . . . . . . . . . . . . . . . . . . . . 65
5.5 Summary binary classification Xception, VGG16, and VGG19. . . . . . . . 73

6.1 Our VGG19 three and four classes classification . . . . . . . . . . . . . . . 84
6.2 Modified VGG19 CNN . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
6.3 Three classes VGG19 transfer learning review . . . . . . . . . . . . . . . . . 85

# List of Figures

1.1 Hospital saturated of patients in Quito-Ecuador . . . . . . . . . . . . . . . . 2

2.1 Attention UNet Model used in breast cancer segmentation. . . . . . . . . . 7
2.2 Natural and artificial neural network comparison . . . . . . . . . . . . . . . . 8
2.3 Representation of a simple artificial neural network . . . . . . . . . . . . . 8
2.4 VGG neural network architecture . . . . . . . . . . . . . . . . . . . . . . . . 9
2.5 Sigmoid and *tanh* activation functions . . . . . . . . . . . . . . . . . . . . 12
2.6 Hadamard product . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
2.7 Three-dimensional convolution . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.8 Cross-correlation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.9 Cross-correlation for height and width using strides . . . . . . . . . . . . . 16
2.10 Padding and stride example . . . . . . . . . . . . . . . . . . . . . . . . . 17
2.11 A simple Convolutional neural network . . . . . . . . . . . . . . . . . . . . 18
2.12 VGG16 and VGG19 layers . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
2.13 Xception model structure . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
2.14 MNIST dataset for handwritten digit recognition . . . . . . . . . . . . . . 25
2.15 Random images extracted from the CT Medical Images dataset . . . . . . 25
2.16 Confusion matrix presenting two classes. . . . . . . . . . . . . . . . . . . . 32

3.1 Ultrasound Pneumonia . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
3.2 Direct Immunofluorescent Antibody Test . . . . . . . . . . . . . . . . . . . 35
3.3 Mycobacterium Tuberculosis Scanning Electron Micrograph . . . . . . . . 37
3.4 SARS-CoV-2 Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
3.5 Chest X-ray Images from Several Patients with Symptoms of COVID-19 . 41
3.6 U-Net Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
3.7 CLAHE Filter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
3.8 Confusion Matrices for Three Datasets . . . . . . . . . . . . . . . . . . . . 45
3.9 COVID-QU-Ex Dataset Images . . . . . . . . . . . . . . . . . . . . . . . . 46
3.10 *COVID-QU-Ex Dataset* segmentation and prediction . . . . . . . . . . . 47

4.1 VGG19 neural network characteristics extracted in each layer . . . . . . . . 50
4.2 Chest X-ray of a patient diagnosed positive for COVID-19 infection. . . . . 52
4.3 Augmentation images . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
4.4 VGG16 Xray Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55
4.5 VGG16 Xray Model Expanded . . . . . . . . . . . . . . . . . . . . . . . . . 56
4.6 T4 tesla GPU . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60

5.1 Binary Chest X-ray . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

5.2 Three Classes Samples Images . . . . . . . . . . . . . . . . . . . . . . . . . . . 63
5.3 Samples of the Four Classes . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
5.4 Xception Convergence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.5 VGG16 Convergence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
5.6 VGG19 Convergence . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68
5.7 3-class Convergence Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 69
5.8 4-class Convergence Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . 70
5.9 Confusion Matrix Xception . . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.10 Confusion Matrix VGG16 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
5.11 Confusion Matrix VGG19 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72
5.12 ROC Graphs-Binary Classification . . . . . . . . . . . . . . . . . . . . . . . 74
5.13 Confusion Matrix VGG19 3-class . . . . . . . . . . . . . . . . . . . . . . . . 75
5.14 Confusion Matrix VGG19 4-class . . . . . . . . . . . . . . . . . . . . . . . . 76
5.15 Chest X-ray of a Patient with COVID-19 . . . . . . . . . . . . . . . . . . . . 77
5.16 Another Chest X-ray of a Patient with COVID-19 . . . . . . . . . . . . . . . 77
5.17 Female Chest X-ray of a patient with COVID-19 . . . . . . . . . . . . . . . . 78
5.18 Chest X-ray of a Healthy Patient . . . . . . . . . . . . . . . . . . . . . . . . 78
5.19 Covid-19 Heat Maps Obtained from the VGG19 Model . . . . . . . . . . . . 79
5.20 Healthy Patient's Heat Maps Obtained from the VGG19 Model . . . . . . 80
5.21 Viral Pneumonia Class Heat Maps VGG19 model . . . . . . . . . . . . . . . 81
5.22 Lung Opacity Class Heat Maps VGG19 Model . . . . . . . . . . . . . . . . 82

7.1 Screen capture of the model we built in the Kaggle platform. . . . . . . . . 101
7.2 Google Colab code used to train and test the Xception pre-trained model. . 102

# List of Equations

2.1 Images in dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.2 ReLU . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.3 Sigmoid activation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.4 Hyperbolic tangent activation function . . . . . . . . . . . . . . . . . . 11
2.5 Convolution operator . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.6 Convolution two-dimensional images . . . . . . . . . . . . . . . . . . . 11
2.7 Cross-correlation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.8 Element-wise product 2x2 dimension . . . . . . . . . . . . . . . . . . . 12
2.9 Convolution example . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
2.10 Stride . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
2.11 Padding and stride example . . . . . . . . . . . . . . . . . . . . . . . 17
2.12 Accuracy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
2.13 Precision . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
2.14 Recall . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
2.15 F1-Score . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
4.1 Adam update parameters . . . . . . . . . . . . . . . . . . . . . . . . . 57
4.2 Binary cross-entropy . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
4.3 Binary accuracy . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59

# Abbreviations

| | |
|---|---|
| **ML** | Machine learning |
| **ANNs** | Artificial neural networks |
| **CNNs** | Convolutional Neural Networks |
| **DL** | Deep learning |
| **RNNs** | Recurrent neural networks |
| **DBNs** | Deep belief networks |
| **GPUs** | Graphical processing units |
| **LTI** | Linear time-invariant system |
| **AF** | Activation Function |
| **ReLU** | Rectified Linear Unit |
| **FC** | Fully Connected |
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **GPT** | Generative Pre-training Transformer |
| **BAIR** | Berkeley AI Research |
| **SGD** | Stochastic Gradient Descent |
| **Adam** | Adaptive Moment Estimation |
| **Adagrad** | Adaptative Gradient Algorithm |
| **RMSProp** | Root Mean Square Propagation |
| **MSE** | Mean Squared Error |
| **PCR** | Polymerase Chain Reaction |
| **DFA** | Direct Immunofluorescent Antibody |
| **RT-PCR** | Real-Time Polymerase Chain Reaction |
| **FDA** | Federal Drug Administration |
| **ACE2** | Angiotensin-Converting Enzyme 2 |
| **CT** | Computed Tomography |
| **PA** | Posteroanterior |
| **AI** | Artificial Intelligence |
| **DNN** | Deep Neural Network |
| **CLAHE** | Contrast Limited Adaptive Histogram Equalization |
| **TL** | Transfer Learning |
| **CXR** | Chest X-ray |
| **GCP** | Google Cloud Platform |
| **ELISA** | Enzyme-Linked Immunosorbent Assay |
| **IESS** | Ecuadorian Institute of Social Security |
| **ROC** | Receiver Operating Characteristic |

# Chapter 1

# Introduction

## 1.1 Thesis Overview

In this introduction, the problem of a lack of tools and resources in Ecuador and all over the world is shown. Also, the contribution to the community and the world by offering a predictive tool using AI. The general and specific objectives were also defined, and all were completed successfully. In the theoretical framework section, several concepts relating to AI are defined to help the reader understand the tools we used in our work. In the state-of-the-art, COVID and pneumonia tests are discussed. Moreover, the available and most accurate neural networks are discussed. Also, transfer learning models are shown in the state-of-the-art section, with some improvements in accuracy and F1-Score during the last few years. In the methodology, the software we used and the hardware are explained. In the results, metrics, a confusion matrix, and training graphs are presented. In the discussion of results, several comparisons of the neural networks we used are made, making comments on the performance of VGG19 in particular. Finally, in the conclusions, a summary of all the work and the performance of our neural networks is shown, along with recommendations and limitations.

## 1.2 The Pneumonia Outbreak Problem

A pneumonia outbreak of unclear cause was documented in Wuhan, Hubei Province, China, in December 2019. Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) was isolated as a result of respiratory sample inoculation extracted from human airway epithelial cells, Vero E6 and Huh7 cell lines. Genome analysis of the virus revealed it to be a novel coronavirus related to SARS-CoV [1]. The SARS-CoV-2 virus produced a disease called COVID-19, which became highly contagious and spread rapidly throughout the world, causing a global pandemic. The symptoms of the disease include fever, shortness of breath, cough, headaches, and more, which lead to pneumonia and finally acute respiratory distress syndrome [2]. Scientists have been debating the origin of the new coronavirus known as SARS-CoV2 since its discovery. There have been rumors that SARS-CoV-2 is the result of experiments in the lab. Genetic evidence, however, refutes this theory and demonstrates that SARS-CoV-2 did not originate from a previously identified viral backbone [3]. Similar to other respiratory viruses, SARS-CoV-2 primarily spreads through the respiratory system and is very effective and contagious. The main confirmed path of transmission is droplet transfer [4]. Elisa and PCR tests were used to confirm the presence of the virus in the early stages of the disease. PCR test obtained more sensibility compared to Elisa. The Elisa targets were IgM and IgG immunoglobulins; for PCR were specific regions of the SARS-CoV-2 virus genome [5]. Saliva secretions showed a higher presence of the coronavirus than in other regions of the body. Also, ocular secretions were tested, and contagion was found, thus suggesting a possible way of infection [6]. Ecuador presented a peak of COVID-19 cases in January 2022, with 890,541 confirmed cases and 35,658 deaths [7]. Also, influenza and pneumonia, not related to COVID-19, were among the top 5 causes of death in Ecuador during 2019, according to INEC Ecuador government statistics [8]. The hospitals around the country were saturated with patients and improvising tents to attend to infected patients Figure 1.1. A need for prevention measures and rapid diagnostic tests appeared during the pandemic. One of the cheaper available tests was a chest X-ray. It could diagnose the presence of the disease with significant sensibility [9].

Figure 1.1: Hospital saturated of patients in Quito-Ecuador during the pandemic. The IESS hospital, located in the south of the city, had to improvise patient stretchers in a cafeteria due to the increasing of COVID-19 infections [10].

## 1.3 Thesis Justification

So, a rapid detection system of the symptoms might have helped prioritize patients to enter the intensive care unit, reducing a significant percentage of deaths. X-ray images allow the confirmation of pneumonia when the quick test gives a negative result but the patient has symptoms of COVID-19.

In Ecuador, the public health system has many limitations. The lack of investment in new equipment and the lack of qualified doctors to diagnose and treat COVID-19 disease have created the need for alternative diagnostic tools. Artificial intelligence has proven to be a very effective alternative when doctors are not available or an additional medical opinion is needed to accurately confirm the presence of the disease. Computer vision offers a way to create and test computer-trained models using large datasets that are prepared by scientists and medical experts. The models can be deployed on several platforms to predict the disease with a high level of accuracy using chest X-ray images. These models are not limited to X-ray images since they can be trained with any type of medical images helping in other types of disease diagnosis.

## 1.4 Contribution to Society

This work offers a model trained with the most advanced neural network techniques to save time and important computer resources during the training and classification of chest X-ray images. Also, we present the parameters to ensure the reproducibility of the work. Moreover, techniques to optimize the dataset are presented. Moreover, the transfer learning models can be used by other experts to deploy and offer a tool for the rapid diagnosis of COVID-19. The publicly available Kaggle dataset [11] was used in this work.

## 1.5 Thesis Objectives

### 1.5.1 Work General Objective

The need for a tool to predict whether or not a patient had COVID-19 appeared during the pandemic. Neural networks have proven to be very effective at learning complex image patterns. So, chest X-ray images that have complicated structures were used in this work. Our objective is to create and test state-of-the-art neural networks that have been used in the prediction of COVID-19 using chest X-rays and try to replicate and possibly improve the results that have been reported so far. Also, the models will be tested with different classes, and the results will be reported to conclude which model is better. The trained models could be used by Information Technology (IT) developers to create mobile apps and websites that can aid in the diagnosis of COVID-19.

### 1.5.2 Thesis Specific Objectives

- Prepare the dataset to be used during the training process.
- Design and create different neural networks using the transfer learning technique
- Determine the best parameters: epochs, learning rate, and optimizer to get the best results.

- Train and test different Deep learning models to perform the COVID classification task.
- Determine the best neural network and test it in other datasets.
- Create trained models that can be used by other scientists without the need for the training and validation process saving time and computer resources.
- Report and discuss the results.

# Chapter 2

# Theoretical Framework

## 2.1 Overview of Artificial Intelligence Technology

Artificial intelligence (AI) is the most exciting and talked-about technology of the past ten years. According to John McCarthy, it is the process of making intelligent machines, especially by creating intelligent computer programs that mimic human intelligence [12]. AI is currently used in many industries, including business, healthcare, education, the military, and manufacturing [13]. AI has various applications that revolutionize industries and daily life. Machine Learning enables computers to learn and predict without explicit programming [14], powering recommendation systems and predictive analytics [15]. Natural language processing facilitates human-computer interaction, as seen in chat-bots and language translation [16]. Computer vision interprets visual data for tasks like object and facial recognition [17]. Robotics involves developing intelligent machines, impacting areas from industrial automation to robotic surgery [18]. Speech recognition converts spoken language into text, used in voice assistants and speech-to-text systems [19]. Genetic algorithms optimize processes through algorithms inspired by natural selection [20]. Reinforcement learning trains agents for sequential decision-making, as applied in game playing and robotic control [21].

In medical diagnosis, AI is being widely used. In this article [22], searchers have used machine learning techniques to detect early Alzheimer's disease using MRI images and data. Surgical robots can be trained using AI to achieve precise control of medical equipment [22]. In breast cancer screening, AI can detect and classify breast lesions [23]. Using chest tomography scans, machine learning algorithms can predict the probability of developing lung cancer [24]. Real-time detection in combination with deep learning (DL) was used in the diagnosis of basal cell carcinoma [25]. Machine learning and DL are used in the detection of tuberculosis using chest X-ray images [26]. Also, DL is used to detect lung cancer, pneumonia, lung opacity, and COVID-19 [27]. Convolutional neural networks, transfer learning, generative adversarial networks, recurrent neural networks, and attention mechanisms are techniques used in machine learning to detect patterns from images and extract information from data [28]. Convolutional neural networks are widely used in image processing and, recently, transfer learning which will be expanded further.

### 2.1.1 Machine Learning as a Subset of Artificial Intelligence

Machine learning (ML), a subset of artificial intelligence (AI), involves developing algorithms that allow computers to learn and make decisions without being explicitly programmed. This approach enables systems to improve and adapt based on experience or data, finding applications in areas like image recognition, natural language processing, and recommendation systems [29]. Depending on human supervision, ML can be divided into three categories: *unsupervised learning*, *supervised learning*, and *reinforcement learning* [30], [31]. Unsupervised learning is a type of machine learning where the algorithm works with unlabeled data, aiming to discover patterns or structures without explicit guidance. It involves two main tasks: clustering, grouping similar data points without predefined categories, and dimensionality reduction, simplifying data by reducing the number of features. This approach is valuable for exploring data structure, identifying hidden patterns, and preparing data for further analysis. Common applications include clustering customers based on behavior or reducing the complexity of datasets for more efficient processing [32]. In supervised learning, the algorithm is trained on a labeled dataset, which means the input data is paired with corresponding output labels. The term supervised comes from the idea that the process involves a teacher or supervisor providing the algorithm with correct answers to guide its learning [33].

### 2.1.2 Applications of Deep Learning Algorithms

Deep learning is a subset of machine learning that involves artificial neural networks (ANNs) with multiple layers (deep neural networks). Unlike traditional machine learning, deep learning algorithms automatically learn hierarchical representations of data, extracting intricate features from raw input [34]. The architecture, inspired by the human brain, allows these networks to model complex relationships and patterns. Deep learning has proven highly effective in tasks such as image and speech recognition, natural language processing, and playing strategic games. Its ability to handle vast amounts of data and automatically discover intricate patterns has contributed to significant advancements in various fields, making it a key technology in modern AI applications [35]. DL involves the use of ANNs. In deep learning, these neural networks have multiple layers, allowing them to automatically learn hierarchical representations of data, which is a key distinguishing feature from traditional machine learning approaches [36]. Hierarchical representations in deep learning involve the automatic learning of increasingly abstract and complex features at different layers of a neural network, allowing the model to capture nuanced patterns in the input data [37]. A DL model necessitates the incorporation of multiple hidden layers within a neural network. Furthermore, various neural network architectures are available, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep belief networks (DBNs) [38]. On the other hand, a poor feature selection may produce inaccurate results when it comes to class identification. In contrast, DL enables the automation of feature learning, enabling simultaneous learning and categorization [39].

### 2.1.3 Computer Vision Applications

CV is a field within artificial intelligence focused on enabling computers to interpret and understand visual information. It involves developing algorithms and models to analyze images or videos, mimicking human visual perception [40]. Key applications include image recognition, object detection, and facial recognition. With the advent of deep learning, particularly CNNs, computer vision has achieved remarkable progress [41]. CV finds widespread use in various sectors, including healthcare, autonomous vehicles, and augmented reality [42], [43], [44]. As technology evolves, computer vision continues to drive innovation, revolutionizing industries and enhancing capabilities in tasks such as image analysis, security, and immersive experiences [45]. In Figure 2.1, we have breast cancer ultrasound images that have been processed by neural networks to segment and predict if the patient has or does not have cancer.

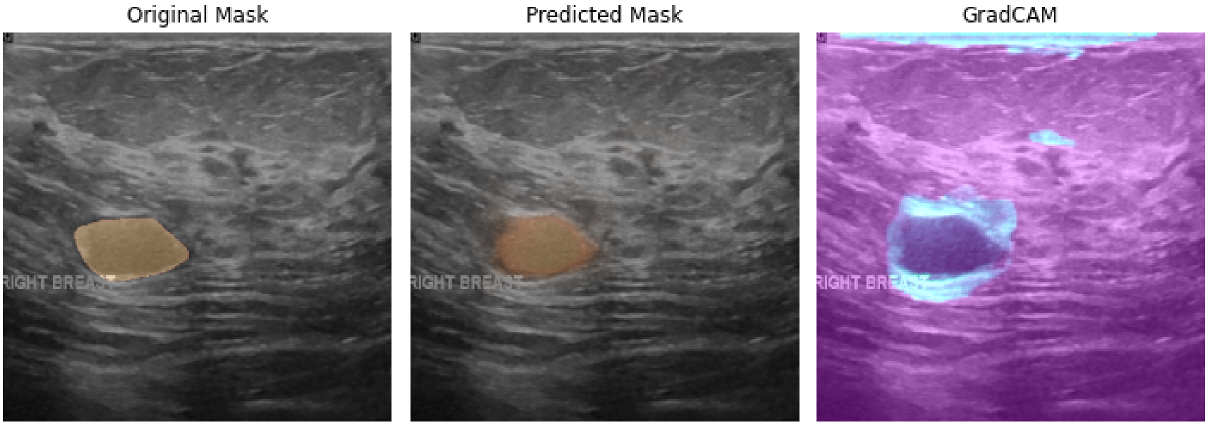
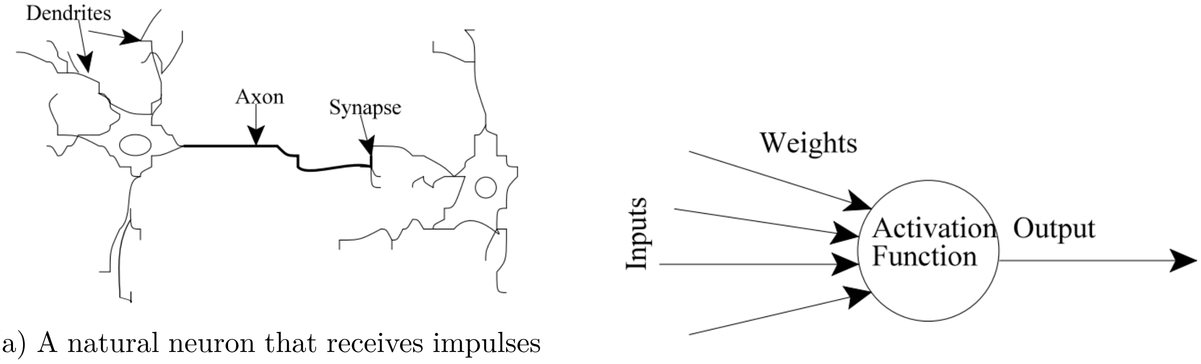


Figure 2.1: Attention UNet Model used in breast cancer segmentation. The Breast Ultrasound Dataset was used in the implementation of this model [46]. Grad-CAM is a technique used for visualizing and understanding the decisions made by convolutional neural networks.

## 2.2 The use of Artificial Neural Networks

ANNs are computational models that draw inspiration from natural neurons Figure 2.2. Natural neurons Figure 2.2a, have synapses on their membranes or dendrites where they receive signals. The neuron is triggered and releases a signal via the axon when the signals it receives are powerful enough to cross a particular threshold. This signal may be transmitted to a different synapse, perhaps causing other neurons to fire [47]. When modeling artificial neurons, the complicated structure of genuine neurons is greatly simplified. In essence, as shown in Figure 2.2b, natural neurons are made up of inputs (such as synapses) multiplied by weights (the strength of the corresponding impulses), which are then calculated by a mathematical function to determine whether the neuron will activate. An additional function, which could be the identity, computes the artificial neuron's output, sometimes based on a threshold. Artificial neurons are combined in ANNs to process information [47]. ANNs consist of interconnected layers of nodes or neurons Figure 2.3. The three main types of layers are the input layer, which receives initial data; the hidden layer, positioned between input and output and responsible for learning complex patterns; and the output layer, which produces the final output or prediction. Hidden layers are termed so because they don't directly interact with the external environment or final output but play a key role in learning and capturing intricate data patterns [48].



(a) A natural neuron that receives impulses at dendrites processes the signal in the neuron's center and transmits the processed impulse to other neurons through the axon. The neurons are connected by a synapse

(b) Artificial neural network that receives inputs and processes the signals with a function and produces an output similar to a natural neuron .

Figure 2.2: A comparison between a natural and an artificial neuron [47].

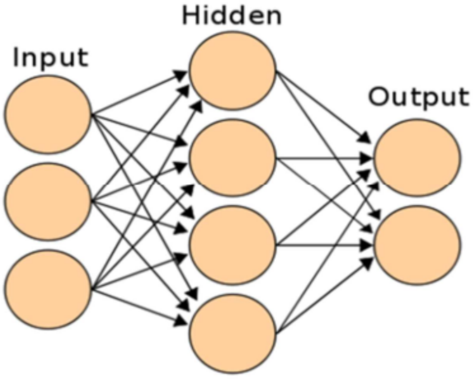


Figure 2.3: A simple ANN composed of an input layer, hidden layers, and an output [49].

The evolution and development of CNNs have a long history. This is a brief overview of their past: Kunihiko Fukushima created the self-organizing neural network model for pattern recognition known as the Neocognitron in 1980. The Neocognitron opened the path for the creation of CNNs by demonstrating its capacity to recognize handwritten characters [50].

CNN architecture AlexNet, created by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, produced ground-breaking outcomes in the 2012 ImageNet Large Scale Visual Recognition Challenge. Then, the University of Oxford's Karen Simonyan and Andrew Zisserman proposed a CNN architecture known as VGGNet in 2014. The goal of VGGNet was to investigate how network depth affected CNN accuracy in image recognition tasks. The multiple convolutional layers of the VGGNet architecture are followed by fully connected layers [51]. VGGNet receives a 224x224 RGB image as input. The network applies a sequence of max-pooling layers with a 2x2 filter and a stride of 2, after which it applies a series of convolutional layers with small 3x3 filters and a stride of 1. Multiple iterations of this convolution and pooling pattern produce a deep network with either 16 or 19 layers, called VGG16 and VGG19, respectively Figure 2.4.

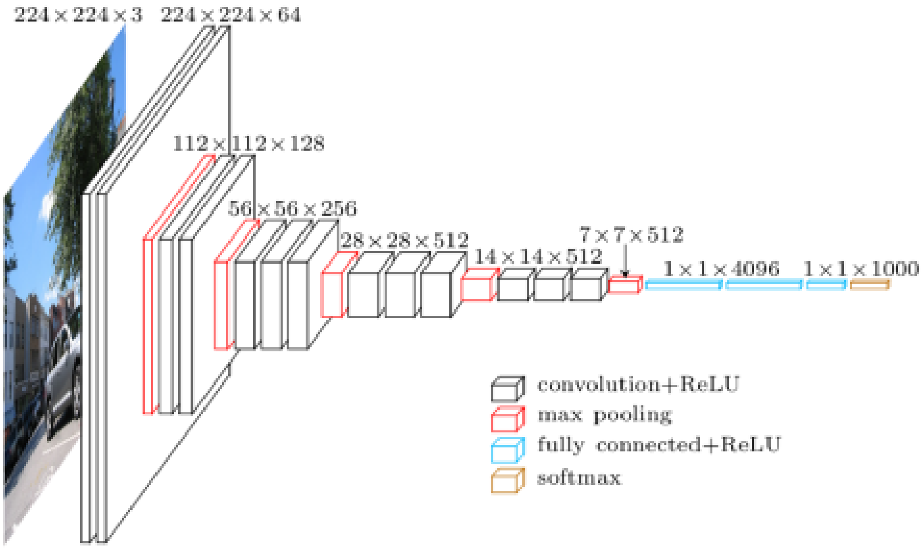


Figure 2.4: VGG Neural Network Architecture with an input of $224 \times 224 \times 3$ [52].

In 2014, in terms of competence, VGGNet performed remarkably well. Its accuracy greatly outperformed earlier models, proving the usefulness of deep CNN architectures. In the competition, VGG16 and VGG19 both had top-5 error rates of 7.3% and 7.5%, respectively. As a foundation for later CNN architectures, the VGGNet architecture has gained popularity as a solution for image recognition applications [53]. It is significant to remember that VGGNet has a deep architecture and that training takes a lot of computer power. Although the network's high parameter count increases accuracy, it also raises computational requirements. VGGNet training can take a long time, and to speed up the process, powerful hardware, like graphical processing units (GPUs), must be available.

### 2.2.1 The Loss function During Training

A loss function, sometimes referred to as an error function, cost function, or discrepancy function, is a mathematical function that quantifies the difference between the model's predicted and actual outputs in the context of neural networks. It measures the model's effectiveness at a certain task.

A loss function's primary function is to indicate how well the model performed during training. The model can minimize the loss and enhance its predictions by adjusting its parameters (weights and biases) through gradient descent and backpropagation after calculating the loss. Some examples:

1. **MSE:** For regression issues, the Mean Squared Error (MSE) loss function is widely

used. The average squared difference between the actual and predicted values is computed. Larger errors are penalized more severely by MSE.

2. **Binary Cross-Entropy:** For binary classification issues, binary cross-entropy is frequently utilized. It calculates the difference between the actual binary labels and the anticipated probabilities.

3. **Categorical Cross-Entropy:** This technique is applied to classification issues involving multiple classes. The average cross-entropy loss between the true class labels and the predicted class probabilities is computed.

One of the more commonly used loss functions is cross-entropy $\mathcal{L}(\boldsymbol{W})$ [54]. Our definition of the *cross-entropy* loss function is $\mathcal{L}(\boldsymbol{W})$.
Through backpropagation of the error through the neural network and convergence to the local minimum value of $\mathcal{L}(\boldsymbol{W})$, the classification network iteratively updates the values of $\boldsymbol{W}$. Images from any dataset can be defined as

$$(\boldsymbol{X}^{(i)}, \boldsymbol{Y}^{(i)}) \quad \text{for} \quad i \in \{1, \ldots, n\}, \tag{2.1}$$

$n$ represents the total number of images in the experimental dataset.

### 2.2.2 Activation Functions in Convolutional Neural Networks

In neural networks, activation functions (AF) are used to calculate the weighted total of input and biases, which determines whether or not a neuron can fire. It uses gradient processing, most commonly gradient descent, to modify the given data before producing an output for the neural network that includes the data's parameters. In some literature, these activation functions are frequently referred to as a transfer function [55]. Since the activation function generates a neuron's output to feed the following layer, it is a crucial component in ANN training. Non-linearity functions are frequently used instead of linearity functions for predicting real phenomena [56]. Next, certain non-linearity activation functions will be explained.

#### Rectified Linear Unit Activation Function

Out of all the AF, one of the most widely used is the so-called rectified linear unit (ReLU) activation function [57]. The function is defined as:

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{if } x_i \geq 0 \\ 0, & \text{if } x_i < 0 \end{cases} \tag{2.2}$$

This function eliminates the vanishing gradient issue seen in previous iterations of activation functions by rectifying input values less than zero and driving them to zero [58]. In the output layers of the network, the ReLU function has been used in the hidden units of deep neural networks with an additional AF. Typical applications of this function include object categorization and speech recognition [59]. There is an issue known as "dying ReLU" that arises when ReLU neurons go dormant and only produce 0 for any given input [60]. The leaky ReLU was suggested as a solution to the dead neuron problems [59].

### Sigmoid Activation Function

In some literature, the sigmoid AF is also known as the logistic function or squashing function [61]. A non-linear AF that is primarily utilized in feedforward neural networks is the sigmoid. It is a real function that is bounded, differentiable, and defined for real input values. It has some smoothness and positive derivatives everywhere [62]. The sigmoid AF is defined by the next relationship:

$$f(x) = \left( \frac{1}{(1 + \exp^{-x})} \right) \tag{2.3}$$

Unfortunately, the sigmoid AF has several significant flaws, such as gradient saturation, and non-zero-centered output, which causes the gradient updates to propagate in various directions [55].

### Hyperbolic Tangent Activation Function (*tanh*)

Another kind of AF utilized in DL is the hyperbolic tangent function, which also has several versions employed in DL applications. The *tanh* function, also referred to as the hyperbolic tangent function, is a smoother zero-centered function with a range of -1 to 1 [34].

$$f(x) = \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right) \tag{2.4}$$

Because it provides higher training performance for multi-layer neural networks, the *tanh* function has replaced the sigmoid function as the favored function [63], [64]. Nevertheless, the sigmoid vanishing gradient issue was also unsolvable by the *tanh* function. The function's primary benefit is that it facilitates back-propagation by producing zero-centered output. One characteristic of the *tanh* function is that it can only achieve a gradient of 1 when the input value is zero or when x is zero (Figure 2.5). As a result, during computing, the *tanh* function produces some dead neurons. The rectified linear unit (ReLU) activation function was created as a result of more research on activation functions in response to the *tanh* function's limitations. The *tanh* function is primarily employed in recurrent neural networks for speech recognition tasks [59] and natural language processing [65].

## 2.2.3 Convolution Operator Technique Example

Convolution is a mathematical technique applied in ML that results in a new output denoting the relationship between x(t) and the reverse-translated version of h(t). The input function x(t) is coupled with a function h(t). Most people refer to the function h(t) as a kernel or filter transformation. This has the following definition in the continuous domain:

$$y(t) = (h \times x)(t) = \int_{-\infty}^{\infty} h(\tau)x(t - \tau)d\tau \tag{2.5}$$

A convolution quantifies the degree of overlap between two functions mathematically [66]. It can be compared to a blending procedure that uses point-wise multiplication with two datasets. The equation for two-dimensional images is defined as:

$$y(i, j) = (h \times x)(i, j) = \sum_n \sum_m h(m, n)x(i - m, i - n) \tag{2.6}$$

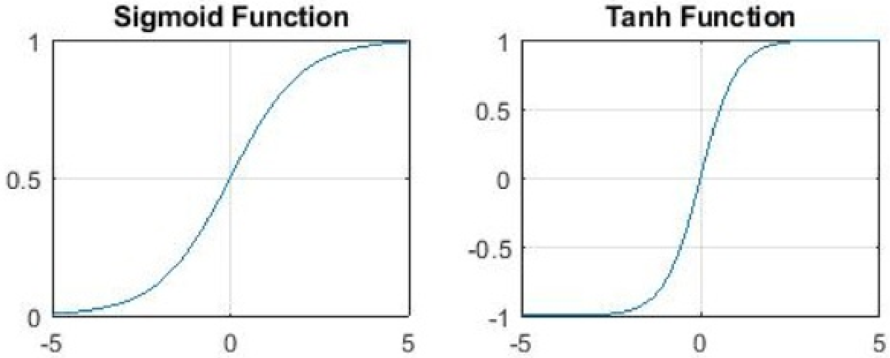


Figure 2.5: Sigmoid and *tanh* activation functions. The ranges of both functions are depicted in the graphs [55].

Cross-correlation can be defined as a mathematical operation that measures the degree of similarity or correlation between two signals, x(t) and ht(t), and it is quite similar to convolution. It is mathematically given by:

$$y(t) = (h \otimes x)(t) = \int_{-\infty}^{\infty} h(\tau)x(t+\tau) \tag{2.7}$$

When it comes to image processing, a convolution filter is just the scalar product of the filter weights with the input pixels within a window. This scalar product is a parallel operation that works well for computing on GPUs and other highly parallel systems.

To give an example with a two-dimensional image, first, let us review element-wise multiplication, also known as the Hadamard product. The Hadamard product of two matrices or vectors involves multiplying corresponding elements together. The general formula for element-wise multiplication of two matrices, A and B, of the same size (having the same number of rows and columns) is as follows: For matrices A and B of the same size, if $A = [a_{ij}]$ and $B = [b_{ij}]$ are two matrices of the same size, then the element-wise multiplication, denoted by $C = A \odot B$, is given by:

$$c_{ij} = a_{ij} \cdot b_{ij}$$

For vectors A and B of the same size, if $A = [a_1, a_2, \ldots, a_n]$ and $B = [b_1, b_2, \ldots, b_n]$ are two vectors of the same size, then the element-wise multiplication, denoted by $C = A \odot B$, is given by:

$$c_i = a_i \cdot b_i$$

As an example, for two matrices of 2x2 dimension, the element-wise product is defined as:

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \odot \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 \cdot b_1 & a_2 \cdot b_2 \\ a_3 \cdot b_3 & a_4 \cdot b_4 \end{bmatrix} \tag{2.8}$$

Let's explore how the Hadamard product works with two-dimensional data and hidden representations, ignoring channels for the time being. The input in Figure 2.6 is a two-dimensional tensor with 3 for height and 3 for width values. We designate the form of the tensors as (3,3) or 3x3. The kernel has two dimensions: height and width. The height and width of the kernel (in this case, 2X2) determine the geometry of the kernel window, also known as the convolution window.

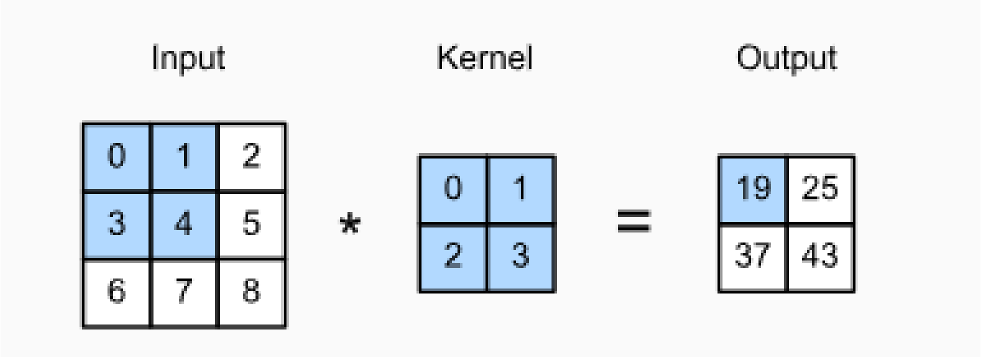


Figure 2.6: Hadamard product, based on cross-correlation operation, applied to 2x2 dimension matrices. The first output element is calculated using the input and kernel tensor elements: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$ [67].

### 2.2.4 Receptive Field and Feature Map in Artificial Intelligence

The output of the convolutional layer in Figure 2.6 is frequently referred to as a feature map since it may be thought of as the learned representations (features) in the spatial dimensions (e.g., width and height) to the next layer. In CNNs, the term "receptive field" describes each element (from all the layers before this one) that could have an impact on the computation during forward propagation. Keep in mind that the receptive field could exceed the input's real size.
Let's continue our explanation of the receptive field using Figure 2.6. The four elements in the shaded part of the input are the receptive field of the shaded output element (calculated value 19) given the 2*2 convolution kernel. Let's now represent the 2*2 output as $y$ and examine a deeper CNN that outputs a single element, $z$, with an extra 2*2 convolutional layer that receives $y$ as input. In this instance, all four of the $y$ elements are included in the receptive field of $z$, whereas all nine input elements of Figure 2.6 are included in the 2*2 receptive field. So, we can construct a deeper network with any image element that requires a larger or smaller receptive field to detect input features over a wider or smaller area.

In Figure 2.7, we have as an input a feature map that contains the RGB channels. Color images typically have the RGB format. So, a filter that can extract the feature and give a convolutional output is needed. The number 3 in $5 \times 5 \times 3$ represents the number of channels that have the feature map. Note that the output does not have this format. The final result is a $3 \times 3$ that is going to be the next input to the next convolutional layer.

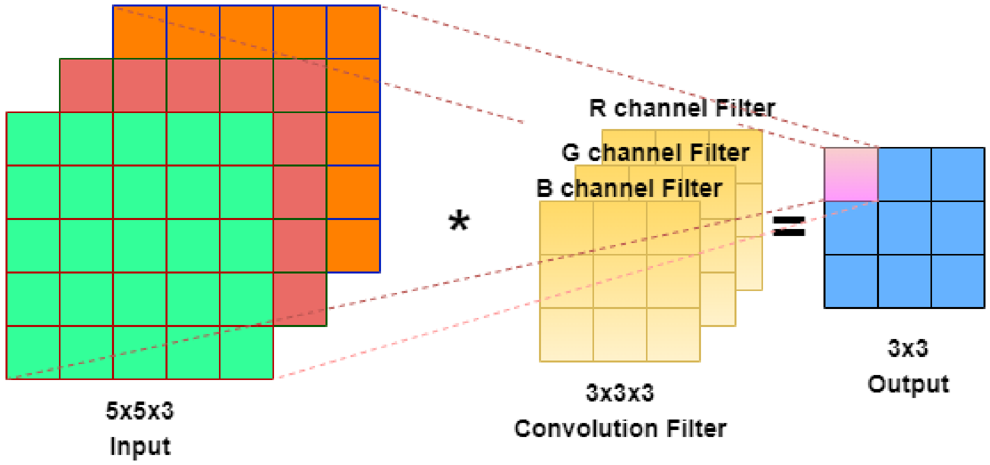


Figure 2.7: 3-dimensional convolution using an input feature map that has 3 channels of color "RGB".

The methods that provide greater control over the output's size are padding and strided convolutions. They will be discussed in the sections that follow. As an introduction, let us examine the fact that after applying numerous successive convolutions, we typically end up with outputs that are significantly smaller than our input because kernels typically have a width and height larger than 1. Ten layers of $5 \times 5$ convolutions take an image that has $240 \times 240$ pixels as a starting point and reduce it to $200 \times 200$ pixels. This process removes 30% of the original image and obliterates any interesting information on the image's edges. The most widely used method for resolving this problem is padding. In other situations, such as when we think the initial input resolution was too complicated, we might want to significantly lower the dimensionality. Stridden convolutions are a well-liked method that can be useful in these situations.

### 2.2.5 Example of Padding Method

For any given convolution, we might only lose a few pixels because we usually use small kernels, but this can add up when we apply many successive convolutional layers. One way to solve this issue is to increase the effective size of the image by adding extra filler pixels to the edges of our input image. Usually, we set the extra pixels' values to zero. In Figure 2.8, a $3 \times 3$ input is padded in to make it $5 \times 5$ in size. Next, a $4 \times 4$ matrix is produced as the corresponding output. The first output element, the input, and the kernel tensor elements used in the output computation are represented by the shaded portions of the equation: $0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$.

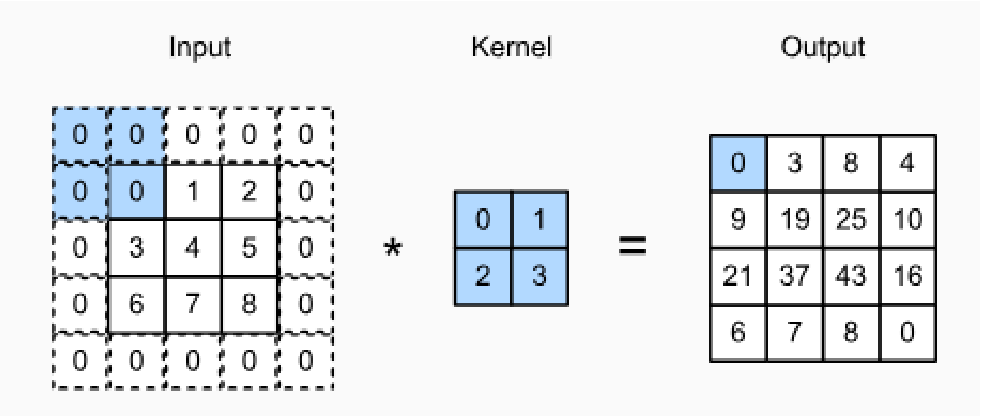


Figure 2.8: Cross-correlation in two dimensions with padding [67].

Consider the convolution example shown in Figure 2.6, where the input was shaped like a $3 \times 3$ matrix and the convolution kernel was shaped like a $2 \times 2$ matrix, resulting in an output representation of dimension $2 \times 2$. The output shape will be $(n_\mathrm{h} - k_\mathrm{h} + 1) \times (n_\mathrm{w} - k_\mathrm{w} + 1)$ assuming that the input shape is $n_\mathrm{h} \times n_\mathrm{w}$ and the convolution kernel shape is $k_\mathrm{h} \times k_\mathrm{w}$: The convolution kernel can only be shifted so far before it runs out of pixels to apply the convolution.

In summary, by adding a $p_\mathrm{h}$ rows of padding and a total of $p_\mathrm{w}$ columns, the output will have the shape:

$$(n_\mathrm{h} - k_\mathrm{h} + p_\mathrm{h} + 1) \times (n_\mathrm{w} - k_\mathrm{w} + p_\mathrm{w} + 1) \tag{2.9}$$

So, the height and width of the output matrix will be increased by $p_\mathrm{h}$ and $p_\mathrm{w}$, respectively.

Convolution kernels with height and width values like 1, 3, 5, or 7 are frequently used in CNNs. The advantage of selecting odd kernel sizes is that we can maintain dimensionality by padding with an equal number of rows on top and bottom and columns on left and right. There are two types of padding: ”valid” and ”same.”

”Valid padding” means that there is no padding, so there are no extra rows and columns. Then, the output in this instance will be $(n_\mathrm{h} - k_\mathrm{h} + 1) \times (n_\mathrm{w} - k_\mathrm{w} + 1)$. To tell a model that all feature map values are valid for convolution, we must specify the valid padding when defining the model in a framework such as Tensorflow or Torch. If we don't, the model won't take into account the more external values.

”Same padding” adds shifting to the convolutional window to maintain the input's size. As a result, equation 2.9 is the formula to determine the padding value. In Figure 2.6, we can observe a convolution with $p_\mathrm{h}$ and $p_\mathrm{w}$ set to a value of 1. Also, the filter size in this image is $2 \times 2$. The values for the filter are typically odd but in this example, the value is even.

### 2.2.6 Striding Method Applied to CNN

To calculate the cross-correlation, we first move the convolution window across all locations to the right and down from the input tensor's upper-left corner. We slid one element at a time by default in the earlier examples. However, there are situations when we shift our window more than one element at a time, omitting the intermediate locations, either for computational speed or because we want to minimize resource consumption. Striding is useful because it captures a large portion of the image; this is especially helpful if the convolution kernel is large.

We name "stride" to the number of rows and columns that are shifted in the convolution window in the input matrix. We have been using the stride of one step in past examples. In Figure 2.9, we observe that the stride for height is 3 and the width is 2. The calculations used to produce the output are $0\times0+0\times1+1\times2+2\times3=8, 0\times0+6\times1+0\times2+0\times3=6$.

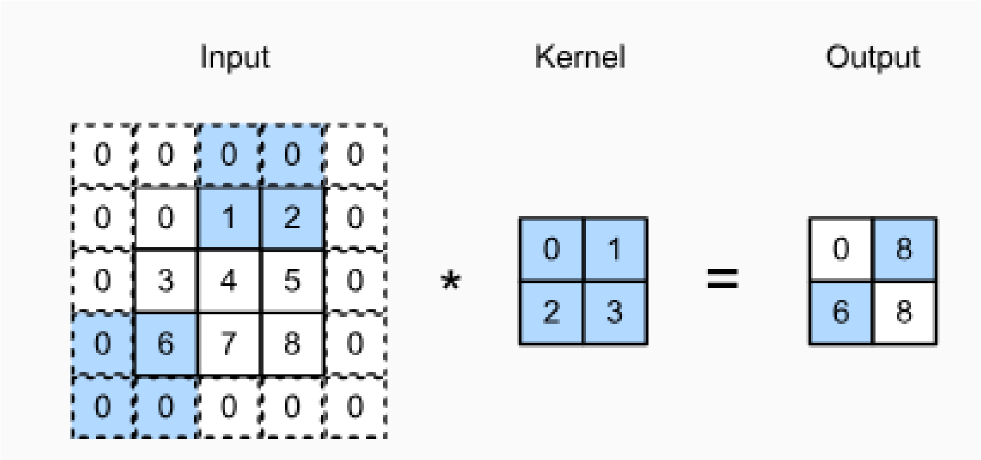


Figure 2.9: Cross-correlation for height and width using strides of 3 and 2, respectively [67].

We can define the stride for the height as $s_\text{h}$ and the stride for the width as $s_\text{w}$, then the output shape is:

$$\lfloor (n_\text{h} - k_\text{h} + p_\text{h} + s_\text{h}) / s_\text{h} \rfloor \times \lfloor (n_\text{w} - k_\text{w} + p_\text{w} + s_\text{w}) / s_\text{w} \rfloor . \tag{2.10}$$

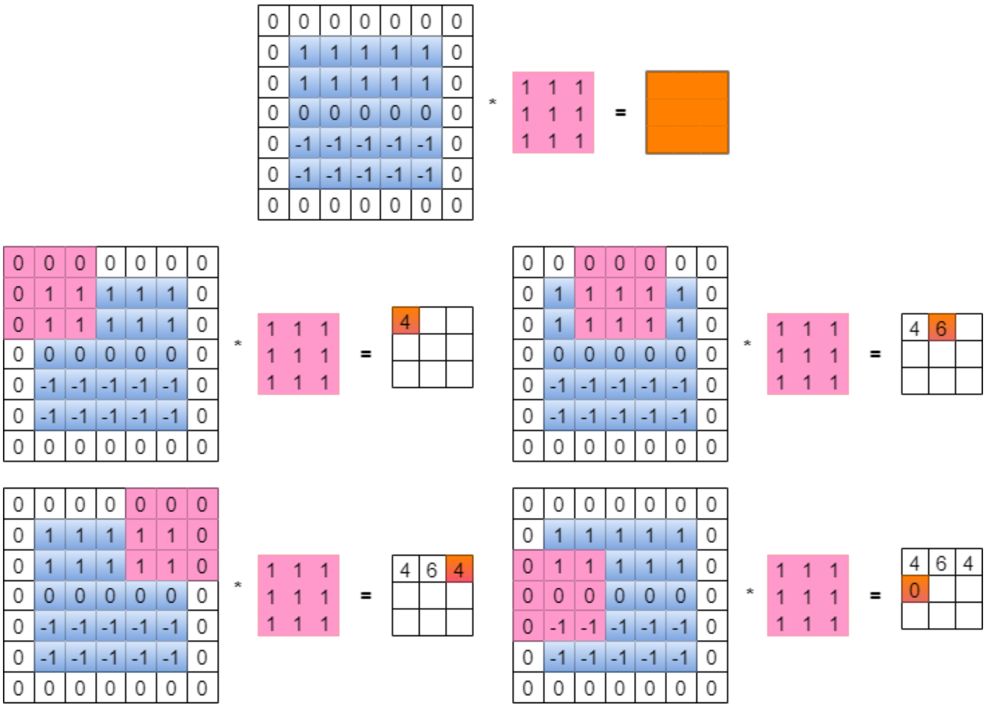


Figure 2.10: An example using both padding and stride with a feature map of 5 × 5 dimension and a 3 × 3 filter. An output of 3 × 3 is produced in the convolution process

In Figure 2.10 we can see a convolution of some of the output matrix. The resulting output shape is given by equation 2.10. The values for $p_{\mathrm{h}} = 2$ and $p_{\mathrm{w}} = 2$ because we are adding 2 rows and 2 columns filled with zeros in Figure 2.10. Also, the values of $s_{\mathrm{h}}$ and $s_{\mathrm{w}}$ are 2 because the shifting of the convolutional window slides two units in each calculation. Replacing the values in equation 2.10 gives an output shape:

$$[\lfloor (5 - 3 + 2 + 2)/2 \rfloor \times \lfloor (5 - 3 + 2 + 2)/2 \rfloor] = 3 \times 3. \tag{2.11}$$

## 2.3 Convolutional Neural Networks Architecture Review

ConvNets, also known as CNNs, are very similar to standard neural networks. They still consist of neurons that can learn new weights from data. After receiving certain inputs, every neuron executes a dot product. The final fully connected layer still has a loss function. A standard neural network goes through several hidden layers after receiving input data as a single vector. Each set of neurons that make up a hidden layer is fully connected to every other neuron in the layer before it. Every neuron in a single layer is independent of the others and does not share any connections. When dealing with an image classification problem, the final fully connected layer—also referred to as the output layer—contains the class scores. Generally, a basic ConvNet consists of three main layers [68]. These three layers are the ”convolution layer”, ”the pooling layer”, and the ”fully connected layer” Figure 2.11.

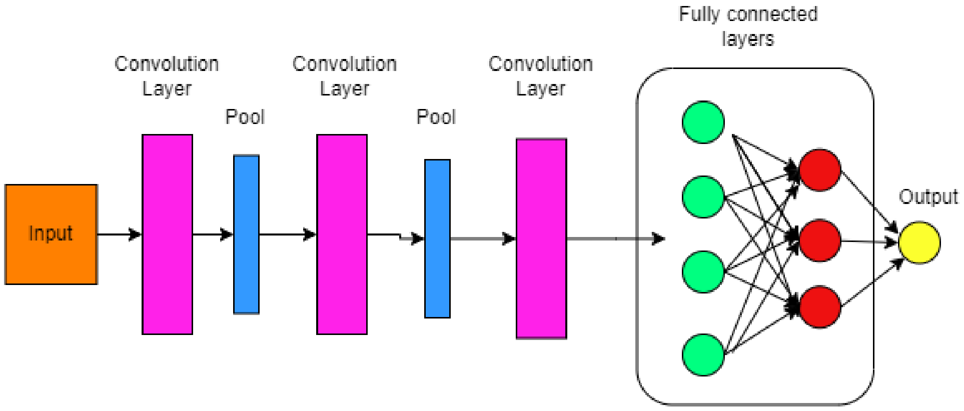


Figure 2.11: A simple CNN with input, several convolution layers, pooling layers, a fully connected layer, and layer output

### 2.3.1 Dropout Technique in Training

You can think of a neural network as a search problem. In a neural network, every node looks for a correlation between the input data and the right output data. Dropout prevents weights from convergent to identical positions by randomly shutting down nodes during forward propagation. Following this, it activates each node individually and backpropagates. Similarly, dropout on a layer can be achieved by randomly setting some of the layer's values to zero during forward propagation. Dropout must be used only during the training process and not during testing.

### 2.3.2 Convolutional Layer Definition

In terms of ConvNet, the primary goal of convolution is to extract features from the input image. In a ConvNet, this layer performs the majority of the computing. The Keras module is typically used to create the layers. The arguments needed are:

- **Filters:** the number of filters used in convolution
- **Kernel size:** this parameter specifies the height and width of the convolution window
- **Stride:** the number to tell how much the convolution window must move and calculate. If this number is not specified, the default stride is set to 1.
- **Padding:** The default is ”valid”. We can specify other padding parameters by using ”same” padding.
- **The ReLU activation function:** It is strongly recommended to add this function to all convolutional layers used in the network. If no activation function is defined, the ReLU is not used as default.

Let us define two examples of creating convolutional layers using keras.

- **First example:** Here we use a $300 \times 300 \times 1$ image which means that the height and width are both 300 and channel 1 means grayscale. The number of filters, in this case 16, is not specific. Typically, the number filters 16,32, and 64 are used. The kernel size is the height and width of the convolutional window. The stride is set to 2, meaning that the convolutional window is going to shift two units to the right. The activation function is ReLU. Here, I present a code used in Python:

$Conv2D(filters = 16, kernel\_size = 2, strides = 2, activation =' relu', input\_shape = (300, 300, 1))$

- **Second example:** After the first example we can create a new layer that is going to be part of the convolutional layer. Now we use 32 filters that will have 3 × 3 dimensions. The padding is set to ”same”, which adds two rows of zeros at the top and bottom and two columns at the left and right of the input. The next code has all the parameters above and also has the ReLU activation function.

  $Conv2D(filters = 32, kernel\_size = 3, padding =' same', activation =' relu')$

### 2.3.3 The Pooling Layer

A convolutional layer, as we've seen, is made up of a stack of feature maps, one for each filter. Convolution becomes more dimensional as more filters are added. More parameters are indicated by higher dimensionality. To minimize the number of parameters and computation, the pooling layer gradually reduces the spatial size of the representation. The convolutional layer is frequently the input source for the pooling layer. The most popular pooling strategy is **max pooling**. Pooling units are capable of completing additional tasks, like **average pooling**, in addition to max pooling. By varying the size and quantity of each filter in a CNN, we can regulate the convolutional layer's behavior. We can increase the number of filters in a convolutional layer to increase the number of nodes, and we can increase the filter's size to increase the pattern's size. So, dimensionality is decreased with the aid of the max pooling layer, which comes after the convolutional layer.

### 2.3.4 The Fully Connected Layer

The necessity of adding Fully Connected (FC) layers appears because the final convolutional layer's receptive field only spans a portion of the input image's spatial dimension. So, shallow CNN models (relatively low convolutional layers) only produce features that correspond to a portion of the image. Therefore, in this case, a small number of FC layers are required. The FC layers make up the majority of the network's parameters in a typical deep neural network. Out of the 60 million parameters in ”AlexNet”, 58 million correspond to the FC layers. In the same way, VGGNet contains 138 million parameters in total, 123 million of which are related to FC layers [69]. The prediction is done in the final layer. A softmax is typically used in the output layer to produce estimated class probabilities.

## 2.4 Remarkable Convolutional Neural Networks

### 2.4.1 Visual Geometry Group Neural Networks

The CNN design was the main topic of Karen Simonyan and Andrew Zisserman's work, ”Very Deep Convolutional Networks for Large-Scale Image Recognition”[51], which was presented at Oxford University in 2014. The major point is to demonstrate that a CNN's depth is essential to achieving higher accuracy. They added more CONV layers to the CNN, deepening its structure. In stacked CONV layers of two and three, they employ 3 × 3 filters to prevent the loss of the spatial size volume on each layer. Unlike AlexNet, which uses 11 × 11 filters, this is not the case. By using 3 × 3 filters in staked CONV layers of two and three CONV layers, we can achieve a behavior that is comparable to that of large receptive fields, such as 5 × 5 and 7 × 7, respectively, but with the benefit of adding depth to the model. Furthermore, we can use one ReLU for every CONV layer. The architecture doubles the number of filters after each max pool layer and is generally homogeneous. Lastly, they display the subsequent models: VGG-19, VGG-11, VGG-13, and VGG-16. In 2014, the architecture emerged as the state-of-the-art after winning the ImageNet challenge with an error rate of 7.3%. A few things to keep in mind with VGG are that the model is 46.6 MB in size (AlexNet is only 1.9 MB) and that there are 140 million more parameters than before. Lastly, a crucial piece of information in the paper is that VGG-19 only slightly improves the model's accuracy; as a result, the accuracy of the model remains saturated after VGG-16 [70]. This finding served as the foundation for subsequent research to provide a solution to the issue of, as of VGG-16, expanding a CNN's depth without compromising accuracy. There are 16 layers in VGG16 and 19 layers in VGG19. In the final three fully connected layers, a sequence of VGGs is identical. A MaxPool is positioned after five sets of convolutional layers in the overall framework Figure 2.12.

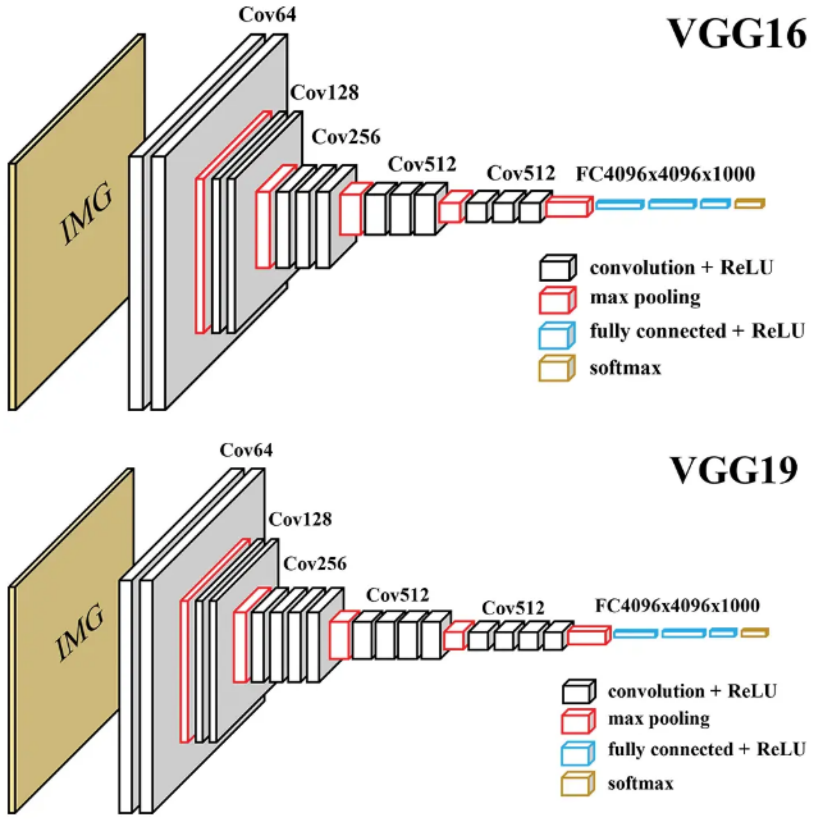


Figure 2.12: VGG16 and VGG19 showing the of layers implemented in the models [71].

### 2.4.2 Inception Neural Network

Researchers at Google presented the Inception neural network, a deep convolutional neural network architecture. Its goal was to provide cutting-edge results in tasks involving image detection and recognition. Inception modules are repeating blocks that make up the Inception network. These modules are inspired by the multi-scale aspect of human visual perception and are designed to extract features at multiple levels. Every Inception module includes max pooling in addition to convolutions using filters of various sizes (1x1, 3x3, and 5x5). Following these operations, the outputs are concatenated and sent to the following layer [72].
With multiple versions, including Inception v1, v2, v3, and so forth, the Inception architecture has changed over time. For instance, Inception v3, which debuted as a component of the GoogLeNet architecture, was utilized in the 2014 ImageNet Large-Scale Visual Recognition Challenge. Inception sought to create deeper networks while preventing an excessive increase in the number of parameters [73]. The Inception network has been extensively employed in many different fields, such as object detection, image analysis, and even life sciences research. You can use deep learning frameworks like TensorFlow or Keras, which offer pre-defined Inception models that you can quickly use in your projects, to implement an Inception network [74].

### 2.4.3 Google Xception

Google researchers created the deep convolutional neural network architecture known as the Xception neural network Figure 2.13. It uses depthwise separable convolutions in place of Inception modules, drawing inspiration from the Inception architecture. Depth-wise Convolution is a kind of convolution in which every input channel receives a single convolutional filter [75]. One way to conceptualize the Xception architecture is as an Inception module with a very large number of towers. Depthwise separable convolutions, a hybrid of depthwise and pointwise convolutions, are used in it [76]. It has been demonstrated that the Xception architecture performs better than Inception V3 on the ImageNet dataset, as well as on a larger image classification dataset with 17,000 classes and 350 million images. It performs better but has the same amount of parameters as Inception V3 [76].

Compared to other networks, the Xception neural network has some benefits. Here are a few main benefits [77], [38]:

- Enhanced Performance: On several image classification tasks, Xception has been demonstrated to perform better than other architectures, including Inception V3. It has demonstrated superior performance, for instance, on the ImageNet dataset and a larger image classification dataset with 17,000 classes and 350 million images.
- Depthwise Separable Convolutions: Compared to traditional convolutions, this method uses fewer connections and computations, which improves efficiency and uses less memory.
- Computational Efficiency: Xception maintains high accuracy while lowering computational complexity through the use of depthwise separable convolutions. Because of

this, it can be used on devices with limited resources or in situations where computational efficiency is essential.

- Transfer Learning: There are pre-trained Xception models that have been trained on massively parallel datasets, such as ImageNet. Even with little training data, users can still achieve good performance by utilizing the learned features from these pre-trained models.
- Versatility: Beyond image classification, Xception can be used for a variety of computer vision tasks, including object detection, semantic segmentation, and image generation. Its features and architecture make it an adaptable option for various visual recognition tasks.

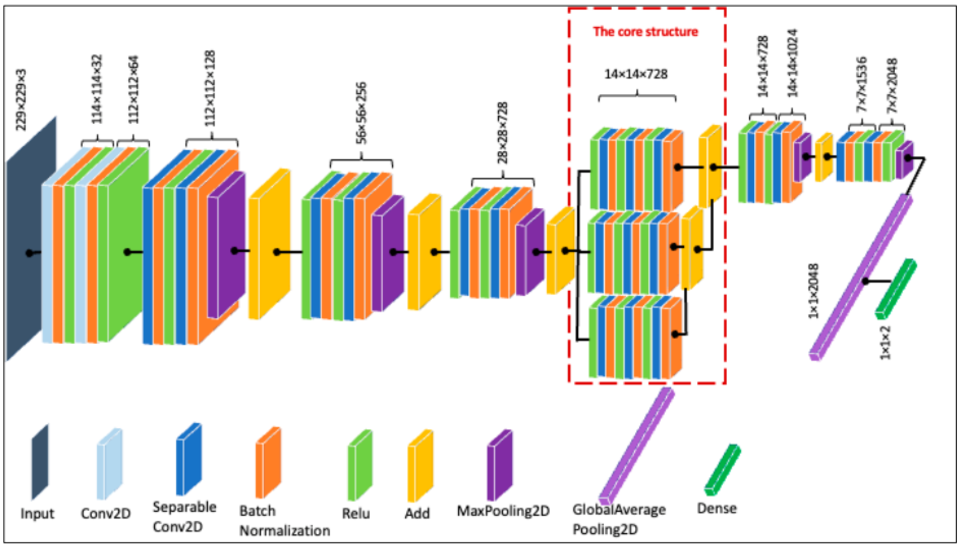


Figure 2.13: Xception model structure [78].

## 2.5 Using the Transfer Learning Technique

In machine learning and deep learning, transfer learning is the process of using a previously trained model as a starting point for a new task or challenge. Transfer learning uses the skills and features acquired from one task to another, rather than starting from scratch when training a model for a new one. Because it enables the training of deep neural networks with relatively little data, transfer learning is popular in the field of deep learning. When dealing with real-world issues where there aren't many labeled data points, this is extremely helpful. Time and computational resources can be saved by fine-tuning a pre-trained model for the new task rather than starting from scratch [79]. Reinforcement learning, natural language processing, image classification, and other fields have all used transfer learning. In tasks like person identification, medical image classification, and disease conversion prediction, it has been used to increase performance and decrease training time.
A few examples of transfer learning are listed below [80], [81], [82]:

- Image Classification: Transfer learning is frequently applied to image classification tasks. A feature extractor could be a pre-trained deep learning model, like ResNet or VGG16, that was trained on a sizable dataset, like ImageNet. The learned features of the pre-trained model are then fed into a new classifier that has been trained on a smaller dataset specifically for the new task.

- Natural Language Processing: Tasks involving natural language processing also make use of transfer learning. For sentiment analysis, for instance, a pre-trained language model such as BERT or GPT can serve as a foundation. Utilizing the language and context knowledge that has already been pre-trained, the model is then improved using a smaller dataset focused on the sentiment analysis task.

- Object Detection: Transfer learning is also applicable to tasks involving object detection. A feature extractor could be a pre-trained model, such as YOLO or Faster R-CNN, trained on a big dataset, such as COCO. The learned features of the pre-trained model are then fed into a new classifier that was trained on a smaller dataset that is particular to the object detection task.

- Reinforcement learning and simulation: Reinforcement learning tasks also make use of transfer learning. Before transferring the knowledge to real-world training, models are trained to perform tasks in various scenarios using simulated environments. For instance, it may be safer and more effective to train a self-driving system first in a simulated setting before moving the model to actual training.

”Freezing” and ”fine-tuning” are terms that are frequently used in transfer learning in the context of machine learning. Let’s examine the components of each of these methods [83]:

The process of taking a pre-trained model and further training it on a new task or dataset is known as fine-tuning. The pre-trained model’s weights are changed during fine-tuning to better suit the new task. Generally, only some of the layers in the pre-trained model are frozen and fine-tuned. Through fine-tuning, the model retains the advantages of the pre-training knowledge while learning task-specific features. A pre-trained model, such as VGG or ResNet, for image classification, for instance, can be improved on a smaller dataset customized to a specific classification task after being trained on a larger dataset, such as ImageNet. The pre-trained model’s early layers, which record basic features like edges and textures, are frequently left untouched, while the later layers are adjusted to pick up on task-specific details.

Freezing is the process of maintaining a pre-trained model’s layers or parameters constant while it is being trained. The weights of a frozen layer are not updated while the layer is being trained. This is frequently done to ensure that the knowledge that the pre-trained model has learned during pre-training is not forgotten. As an example, the embedding layer of a pre-trained language (the layer that has information about how words are represented) model may be frozen during fine-tuning in transfer learning for natural language processing. By doing this, the model can only update the weights of the ensuing layers to adjust to the new task, retaining the semantic information that the pre-trained embeddings had recorded. It is noteworthy that the determination of which layers to fine-tune and which to freeze is dependent upon the particular task, dataset, and resources at hand.

To avoid overfitting, it might be helpful in some situations to freeze more layers; in other situations, it might be required to fine-tune more layers to capture task-specific features.

## 2.6 Steps in Training a Convolutional Neural Network

### 2.6.1 Overview of Deep Learning Frameworks

Software libraries or platforms known as ”deep learning frameworks” offer a range of tools and features for creating, configuring, and implementing deep neural networks. These frameworks provide pre-implemented layers, optimization algorithms, and other utilities, which make the process of creating complex deep-learning models easier [84].

Several well-liked deep learning frameworks consist of:

- **Tensorflow:** It is one of the most popular deep learning frameworks, created by Google. It provides a full environment, supporting both CPU and GPU computations, for creating and implementing machine learning models.
- **Pytorch:** Created by Facebook’s AI Research Lab. Pytorch is an open-source deep-learning framework. It offers dynamic computational graphs, which facilitate the quick definition and modification of models. PyTorch is renowned for being user-friendly and flexible.
- **Keras:** It is a high-level deep learning framework installed on top of backend engines like TensorFlow. Keras is appropriate for novices and quick prototyping because it offers an intuitive API for creating and training neural networks.
- **Caffe:** This framework is used for deep learning and was created by Berkeley AI Research (BAIR). It is renowned for being quick and efficient, especially when it comes to CNNs. Caffe supports a large number of applications and has a large community.
- **MXNet:** It provides a scalable and effective way to train and use deep learning models across multiple platforms, such as cloud servers, GPUs, and CPUs.

### 2.6.2 Using Public and Private Datasets

For deep learning models to be trained and evaluated, datasets are essential. They give the model the required input data along with the labels that go with it. The particular task and area of interest influence the dataset selection.

There are numerous datasets available for diverse applications, including image classification (e.g., ImageNet, CIFAR-10, MNIST), natural language processing (e.g., IMDb, SQuAD), and speech recognition (e.g., LibriSpeech, TIMIT). Public datasets are freely accessible to the general public and have no restrictions on who may use them. Organizations, academic institutions, and governmental bodies frequently produce and distribute these datasets for a range of uses, including benchmarking, analysis, and research. Public datasets offer a variety of data that can enhance model performance and generalization, making them useful tools for training machine learning models. Some examples are the public MNIST dataset for handwritten digit recognition Figure 2.14, the CIFAR-10 dataset for object recognition, and the free-access CT Medical Images dataset Figure 2.15, .

On the other hand, private datasets are not available to the general public and call for special authorization or permissions to access them. These datasets, which are usually owned by people, businesses, or organizations, could include private or sensitive data that has to be kept secure. For internal research, development, or analysis needs, private datasets are frequently utilized. Various types of data, including customer, financial, medical, and proprietary research data, can be found in private datasets. To maintain data security and privacy, authorized people or entities are typically the only ones with access to private datasets.

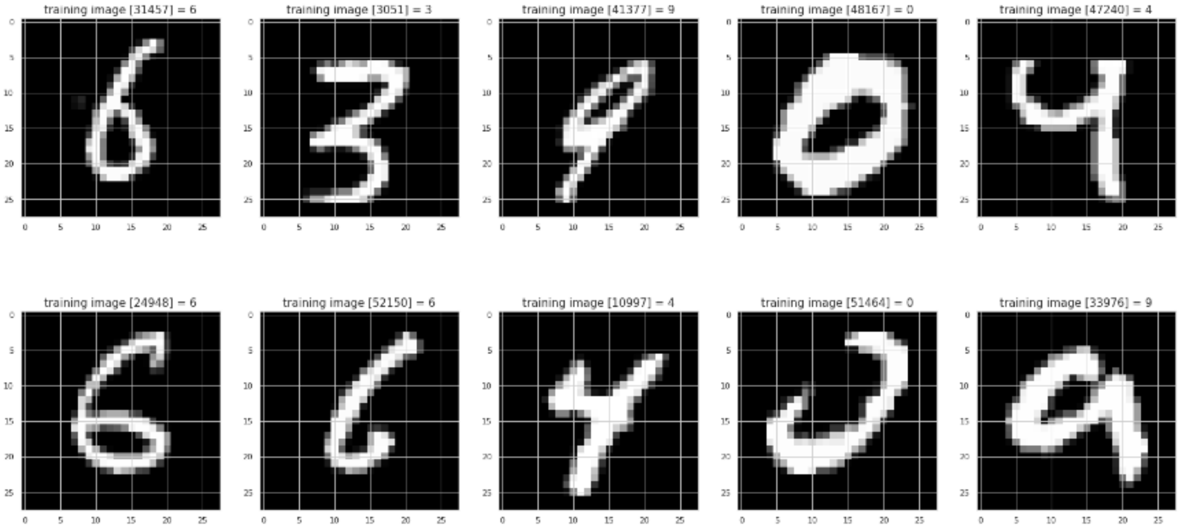

training image [31457] = 6
training image [3051] = 3
training image [41377] = 9
training image [48167] = 0
training image [47240] = 4
training image [24948] = 6
training image [52150] = 6
training image [10997] = 4
training image [51464] = 0
training image [33976] = 9
0
5
10
15
20
25

Figure 2.14: MNIST dataset for handwritten digit recognition

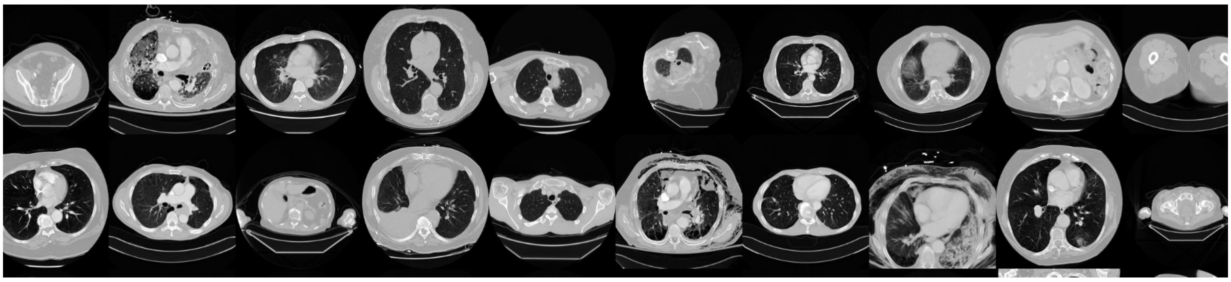


Figure 2.15: Random images extracted from the CT Medical Images dataset that contains images of patients with cancer.

### 2.6.3 Preparing a Dataset Techniques During Training

Preparing a dataset for machine learning is an important step to ensure that the data is clean, complete, and suitable for training a model. Here are some common techniques used in dataset preparation.

#### Labeling Process

In machine learning and data analysis, dataset labeling is an essential step. It entails locating and giving raw data—such as pictures, text files, videos, or other kinds of data—meaningful labels or annotations. Machine learning models are then trained on the labeled data, allowing them to produce precise estimations and predictions. Dataset labeling is a difficult process that calls for careful consideration of many different aspects and techniques. Businesses and researchers must evaluate the task's complexity in addition to its size, scope, and duration to choose the most effective labeling strategy. There are different paths or methods to label data, and each method has its pros and cons. Some common approaches to data labeling include manual labeling, where human annotators manually assign labels to the data, and semi-supervised learning, which combines labeled and unlabeled data to reduce the need for manual labeling.

#### Cleaning a Dataset

The process of locating and fixing mistakes, inconsistencies, inaccuracies, and other problems in a dataset is called dataset cleaning, sometimes referred to as data cleaning or data cleansing. Before model training, this crucial step in data preprocessing must be completed. Cleaning a dataset aims to remove errors like noise, duplicate records, incorrect formatting, missing values, and outliers to increase the data's quality and dependability. Researchers and data analysts can make sure the data is correct, comprehensive, and appropriate for analysis or modeling by cleaning the dataset. The particular procedures and methods used for cleaning a dataset can change based on the type of data and the project's needs. Nonetheless, a few standard methods and approaches are as follows:

- Managing missing values: You can use a variety of advanced techniques, such as regression imputation or multiple imputation, to fill in missing values. Some techniques include mean imputation, median imputation, and mode imputation.

- Eliminating duplicates: To prevent bias and redundancy in the dataset, duplicate records can be found and eliminated.

- Resolving inconsistencies: Data validation techniques, standardizing formats, and fixing typographical errors can all be used to address inconsistent data, which includes conflicting values and formatting errors.

- Managing outliers: If an outlier is an extreme value that differs noticeably from the rest of the data, it can be eliminated if it is incorrect, or it can be handled using the proper statistical techniques.

- Handling noise: Noise in data refers to arbitrary or meaningless variations. Methods such as applying, filtering, or smoothing can reduce the problem of noise in the data.

### Data Augmentation

A common technique in deep learning and machine learning is data augmentation, which creates an artificial increase in the size and diversity of a dataset. To produce new samples that are comparable to the originals but distinct from them, augmentation applies various transformations or modifications to the current data. Models can then be trained using the augmented data, which will increase their performance and generalization.

When used to create new images by applying transformations like rotation, scaling, flipping, cropping, and adding noise or distortions, data augmentation is especially common in computer vision tasks. These modifications strengthen the model's resistance to varying lighting, perspectives, and object orientations, enhancing its capacity to generalize to previously unobserved data. The benefits of data augmentation include:

1. Expanded dataset size: Data augmentation helps get around the limitations of small datasets, which are typical in many categories, by producing new samples.
2. Better generalization of the model: By adding more diversity and variations to the training set, augmented data aids in the model's ability to identify more resilient and all-encompassing patterns.
3. Decreased overfitting: Overfitting occurs when a model performs poorly on untested data because it is too specialized in the training set. Data augmentation can help avoid this.

### Splitting the Dataset

Splitting datasets is an essential step in workflows for deep learning and machine learning. A dataset is divided into distinct subsets for testing, validation, and training. Splitting a dataset is done to avoid overfitting and to evaluate a model's performance and capacity for generalization on untested data. Below is a quick description of every subset:

1. Training Set: The model is trained using the training set. It is the biggest subset of the dataset and is employed to discover the fundamental patterns in the data and optimize the model's parameters.
2. Validation Set: The model is adjusted, and the optimal hyperparameters are chosen using the validation set. It facilitates the assessment of the model's performance throughout training and the decision-making process for model enhancements, like modifying regularization strategies or learning rates.
3. Test Set: The trained model's final performance is evaluated using the test set. It functions as an objective assessment of the model's capacity to generalize to unobserved data. To prevent errors in the evaluation, the test set should not be used during the model development process.

Depending on the size of the dataset and the particular needs of the task, the ratio of division between these subsets may change. Generally speaking, the training set receives 70–80% of the data, the validation set receives 10-15%, and the test set receives the remaining 10-15%. These ratios can be changed to satisfy the particular requirements of the project.

ML frameworks can split datasets using a variety of techniques and libraries. For instance, the $train_test_split$ function from the well-known Python machine learning library scikit-learn makes it simple to randomly divide datasets into training and testing subsets.

### 2.6.4 Hyperparameters in Training a CNN

Hyperparameters are settings made before the start of the training process that govern how the deep learning model behaves. They are manually entered by the user rather than determined from the data. Learning rate, batch size, number of layers, activation functions, and regularization strategies are a few examples of hyperparameters.

Achieving optimal model performance requires selecting the right hyperparameters. It often involves a process of trial and error in which various hyperparameter combinations are examined and tested. Bayesian optimization, random search, and grid search are some of the methods that can be utilized to automate hyperparameter tuning

#### Adjusting the Learning Rate

In machine learning and optimization algorithms, the learning rate is a hyperparameter that sets the step size at each iteration as the algorithm moves toward the minimum of a loss function. It is essential to machine learning models' training process.

A model's learning rate determines how quickly it picks up new information and adjusts to the given situation. Faster convergence is possible with a higher learning rate, but there is a chance that the ideal solution will be missed. Conversely, a lower learning rate could result in slower convergence.

The learning rate can be adjusted using a variety of techniques, including adaptive learning rate approaches, learning rate schedules, and manual tuning. Manual tuning entails choosing a learning rate based on trial and error or past knowledge. Learning rate schedules use cyclical variations or gradual reductions to modify the learning rate during training. Adaptive learning rate techniques dynamically modify the learning rate in response to the optimization process's advancement. Several techniques and approaches can be used to calculate the learning rate $\alpha^{(1)}$. The particular strategy may vary depending on the problem at hand, the optimization framework and algorithm being used, and both. Here are some typical methods:

1. **Fixed Learning Rate:** In certain situations, a fixed learning rate—one that stays constant throughout all training epochs—is employed during the training process. Although this strategy is easy to use, the best outcomes might not always be achieved. $\alpha = 0.001$ is an illustration of a fixed learning rate.
2. **Learning Rate Schedules:** These schedules include changing the rate of learning based on predetermined conditions or at predetermined intervals. When a predetermined condition is met or after a predetermined number of epochs, the learning rate

can be reduced by a fixed factor. Three popular learning rate schedules are exponential, polynomial, and step decay. There may be variations in the formula used in these schedules to calculate the learning rate.

3. **Adaptive Learning Rate Techniques:** These techniques dynamically modify the learning rate in response to the optimization process's advancement. The goal of these techniques is to determine the ideal learning rate for every step. The adaptive learning rate techniques AdaGrad, RMSprop, and Adam are a few examples. Usually, these techniques enhance efficiency when dealing with sparse gradient problems.

It is important to remember that the model's performance and the training process can be greatly impacted by the selection of the learning rate and the updating strategy. Discovering the ideal learning rate for a particular problem and model architecture frequently requires trial and error.

### Number of Epochs in Training

In machine learning, the number of epochs denotes the total number of passes through the training dataset that occurs during the training phase. A batch is a subset of the training data that is processed before the model's parameters are updated, and each epoch is made up of one or more batches.

One hyperparameter that must be set before training starts is the number of epochs. It establishes the number of times the model will run through the training set. Parameters like the size of the dataset, and the model's convergence behavior are some of the variables that influence the number of epochs that are chosen.

### Setting Batch Size

When a neural network is being trained, the term "batch size" describes the number of training examples that are processed collectively in a single forward and backward pass. This hyperparameter holds significant importance as it can affect both the model's performance and the training process. Some factors, such as memory capacity, training speed, and gradient estimation accuracy, influence the choice of batch size.

Accurate gradient estimation and training speed are traded off with batch size. Because each update is based on fewer training examples, smaller batch sizes lead to faster training speeds. Slower convergence and noisy gradient estimates may result from this, though. Larger batch sizes, however, can cause the training process to lag but also yield more accurate gradient estimates.

### Optimizer Algorithm

An optimizer in the context of neural networks is an algorithm or technique that modifies the model's parameters (weights and biases) during training to reduce the loss function and enhance the model's functionality. The optimizer uses the gradients calculated during backpropagation to determine how to update the model's parameters.

There are several commonly used optimizers in neural networks. Here are a few examples:

1. **Stochastic Gradient Descent(SGD):** It's a popular and easy optimizer. By making tiny movements in the direction of the loss function's negative gradient, it modifies the model's parameters. SGD may converge slowly and become trapped in local minima, but it can be computationally efficient.

2. **Adam:** The optimizer Adam (Adaptive Moment Estimation) combines the benefits of RMSProp and AdaGrad. Based on the estimates of the first and second moments of the gradients, it modifies the learning rate for each parameter. Adam is renowned for its quick convergence and strong output on a variety of issues.

3. **Adagrad:** It adjusts the rate of learning for every parameter by utilizing the gradients from the past. Larger updates are provided for parameters with smaller gradients, and smaller updates are provided for parameters with larger gradients. Adagrad can manage various learning rates for various parameters and works well with sparse data.

4. **RMSProp:** Based on the past gradients, the optimizer RMSProp (Root Mean Square Propagation) also modifies the learning rate for every parameter. To normalize the learning rate, a moving average of squared gradients is employed. When it comes to non-stationary objectives, RMSProp works well and can converge more quickly than SGD.

The particular problem at hand, the size of the dataset, and the neural network's architecture all influence the optimizer selection. There isn't a single optimizer that is ideal for every task, so it's usually advised to try out a variety of them to see which one suits your needs.

The optimizer can be specified at the model compilation stage when using well-known deep learning frameworks such as TensorFlow or PyTorch. For instance, you can pass the optimizer as an argument when using TensorFlow's compile() method. You can create an optimizer object in PyTorch and give it the model parameters.

### 2.6.5 Analyzing Results Tools and Techniques During Training

When analyzing the performance of a CNN model, several tools and techniques can be used. Here are some key approaches:

#### Common Metrics in CNN

Several metrics are used to assess a model's performance based on the outcomes found in the test dataset. Metrics allow us to assess a model's overall and per-class performance. Metrics typically make use of the data that the confusion matrix provides. Here are some examples:

1. **Accuracy:** is a commonly used metric for evaluating the performance of classification models. It measures the proportion of correct predictions made by the model out of the total number of predictions. In binary classification, accuracy can be calculated using the formula:

   $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.12}$$

   Where:

   - True Positives represents the number of correctly predicted positive classes.
   - True Negatives represents the number of correctly predicted negative classes.
   - False Positives represents the number of classes that were incorrectly predicted as positive when they were actually negative.
   - False Negatives represent the number of classes that were incorrectly predicted as negative when they were actually positive.

2. **Precision**: It is a metric frequently used to assess a model's performance in classification tasks. Out of all the the cases the model predicts as positive, it calculates the percentage of correctly predicted positive instances. The quality of positive predictions is the main focus of precision. The following formula can be used to determine precision:

   $$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

3. **Recall**: It is a metric that is frequently used in classification tasks to assess a model's performance. Out of all the actual positive cases, it calculates the percentage of correctly predicted positive events. The concept of recall centers on the model's capacity to recognize every positive example. The formula below can be used to calculate recall:

   $$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

4. **F1-score**: A popular metric for assessing a model's performance in classification tasks is the F1 score. It gives a fair assessment of the model's performance by integrating recall and precision into one metric. The accuracy of the model on a dataset is indicated by the F1 score, which is computed as the harmonic mean of precision and recall. When recall and precision are both crucial or when there is an imbalance in the classes, it is especially helpful. The F1 score is computed using the following formula:

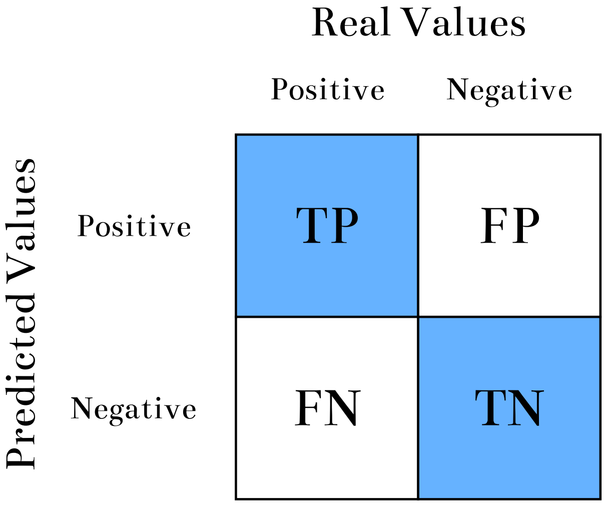
   $$F1\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.15}$$

   Though it can also be applied to multi-class classification by computing the F1 score for each class and then obtaining the average (macro-average or micro-average) based on the particular requirements, it is worth noting that the F1 score is frequently utilized in binary classification tasks.

### Using a Confusion Matrix

Using a confusion matrix, we can visually see the results of our model performance. It is often used to evaluate the accuracy and effectiveness of a machine learning model's predictions. The matrix displays the number of true negatives, true positives, false negatives, and false positives produced by the model on the test data.

Figure 2.16 is an example of a 2 × 2 confusion matrix used in binary classification:



| | | Real Values | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted Values | Positive | TP | FP |
| | Negative | FN | TN |

Figure 2.16: Confusion matrix presenting two classes.

We can compute several evaluation metrics, including accuracy, precision, recall, and F1 score, using the confusion matrix. These measures provide insight into how well the model recognizes positive and negative occurrences.

# Chapter 3

# State of the Art

When the pandemic started, the methods used to diagnose patients with the symptoms of COVID-19 were chest X-rays, Polymerase chain reaction (PCR), and chest CT scans. Wilhelm Röntgen, with the discovery of the X-ray images, opened the door to the diagnosis of several diseases, including pneumonia. Chest X-ray is still being used nowadays and was very helpful during the pandemic. Now, I present a review from the time radiotherapy was used until our days with the use of artificial intelligence in diagnosing diseases, focusing on COVID-19.

## 3.1 Pneumonia Infection

A dangerous lung infection, pneumonia, can be caused by bacteria, viruses, and fungi. A physical examination, diagnostic tests, and the patient's symptoms can all be used to diagnose pneumonia. Chest pain, fever, coughing, shortness of breath, and exhaustion are typical signs of pneumonia. A doctor may identify abnormal lung sounds, like crackles or diminished breath sounds, during a physical examination of the patient [85].

### 3.1.1 Diagnostic Tests for Pneumonia Infection

In this article [86], they compare chest X-ray versus ultrasonography tests Figure3.1. The results are reported in Table 3.1. They used 30 patients with a mean age of 63.8 ± 18.3 years. The value for the specificity of the ultrasonography was reported as not calculable because patients were diagnosed as positive for pneumonia before the test. So, no healthy patients would be found in this article. The same applies for 0% of chest X-ray specificity. In conclusion, an ultrasonography test can detect patients with the disease better than a chest X-ray test.

| Test | Sensitivity [%] | Specificity [%] |
|---|---|---|
| Chest X-ray | 93.1 | 0 |
| Ultrasonography | 100 | not calculable |

Table 3.1: Sensitivity and Specificity of Different Tests for Pneumonia Detection.

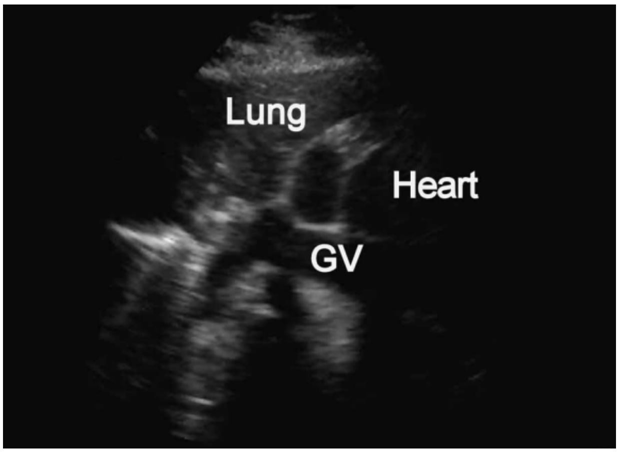


Figure 3.1: A ultrasound with a patient positive for pneumonia. The lung is consolidated or filled with fluid, making the heart visible and the lung looks like a liver [87].

Balk et al. gathered the results of 12 studies that included 1510 patients [88]. In this work, the specificity is reported for both CXR and ultrasound tests Table 3.2.

| Test | Sensitivity [%] | Specificity [%] |
|---|---|---|
| Chest X-ray | 88.75 | 94.23 |
| Lung ultrasound | 86.8 | 98.2 |

Table 3.2: Sensitivity and Specificity of Different Tests for Pneumonia Detection.

In this study [89], statistics are reported in Table 3.3 for Legionnaires' disease, which is produced by the Legionella spp. bacterium. The samples were collected from expectorated sputum, bronchoscopic aspirate, nasopharyngeal swab, nasotracheal suction, and urine samples. The tests they used were direct immunofluorescent antibody (DFA) Figure 3.2, PCR, bacterial culture, and urine antigen. From the results, we observe that PCR and urine antigens have the highest sensitivity but are used to detect specific strains of the bacteria. So, the culture test is better and more affordable in price, offering better specificity and working for all the strains of the bacteria.

| Test | Sensitivity [%] | Specificity [%] |
|---|---|---|
| Culture | 10–80 | 100 |
| DFA stain | 25–70 | 95 |
| PCR | 80–100 | 90 |
| Urine antigen | 70–90 | 99 |

Table 3.3: Pneumonia Legionella spp. Detection Tests.

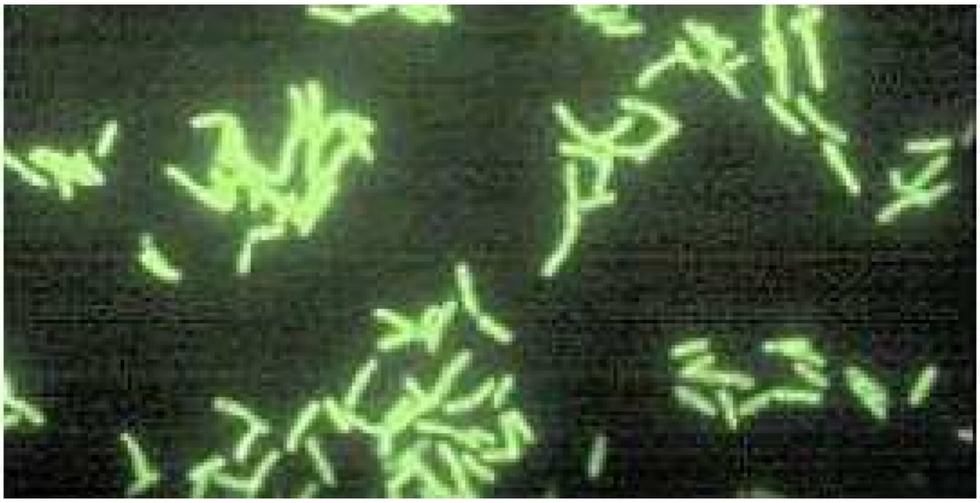


Figure 3.2: Direct immunofluorescent antibody test used to detect the Legionella pneumophila bacterium [90].

### 3.1.2 Pneumonia Bacterial Treatment

The bacteria that produce pneumonia include Streptococcus pneumonia, Staphylococcus aureus, Haemophilus influenza, Mycobacterium tuberculosis, mycoplasma, and chlamydia, among others. The antibiotics used depend on the type of infection that the patient has. I present in Table 3.4 a summary of drugs used to treat this disease [91].

| Drug | Spectrum of Activity | Mechanism of Action |
|---|---|---|
| Ceftaroline | Methicillin-resistant coagulase-negative staphylococci, Streptococcus pneumoniae, vancomycin-resistant Enterococcus faecalis | Inhibition of transpeptidase activity |
| Ceftobiprole | Methicillin-resistant Staphylococcus aureus, Moraxella catarrhalis, Klebsiella pneumoniae, Escherichia coli, Pseudomonas aeruginosa | Inhibition of transpeptidase activity |
| Lefamulin | Gram-positive strains | Inhibition of protein synthesis |
| Telavancin | Gram-positive strains | Inhibition of bacterial wall synthesis and disruption of bacterial membrane function |
| Omadacycline | Penicillin-resistant streptococci, Methicillin-resistant staphylococci, atypical bacterial pathogens, and Gram-negative strains | Action on efflux cell pumps and chemical modifications |

Table 3.4: Drugs used to Treat Bacterial Pneumonia.

In Figure 3.3, we observe a scanning electron micrograph of Mycobacterium tuberculosis, which is the pathogen that causes tuberculosis disease, which attacks the lungs, producing pneumonia, fever, coughing up blood or sputum, persistent cough, and fatigue.

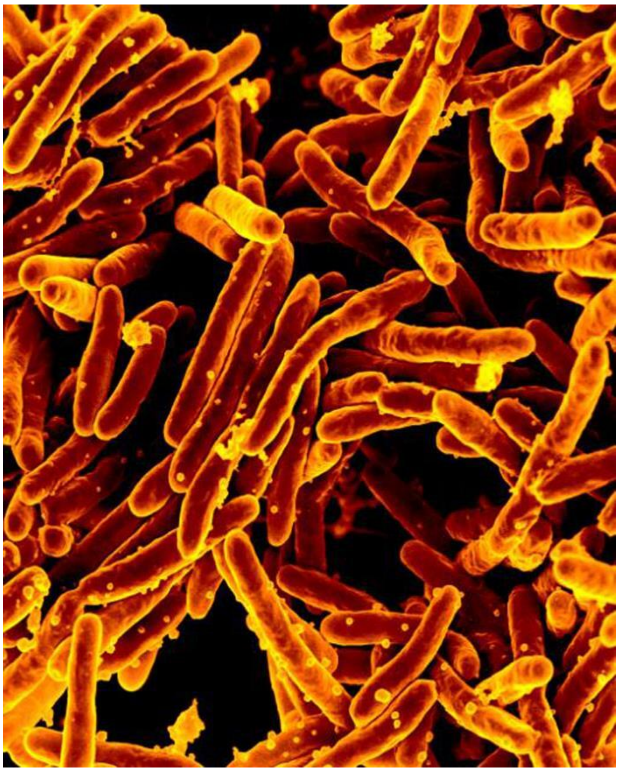


Figure 3.3: Mycobacterium tuberculosis bacteria picture produced by scanning electron micrograph [92].

## 3.2 Covid-19 Virus Review

At the end of 2019, Wuhan, China, was the first city to be contaminated with the highly transmissible SARS-CoV-2 virus seen in Figure 3.4 that causes viral respiratory distress and pneumonia, which was subsequently designated COVID-19. Around 350 million people had tested positive for the virus globally by the end of January 2022, and there had been approximately 5.6 million fatalities [93]. Several methods to diagnose and treat patients will be presented in this work, which were developed during the pandemic and continue to be used nowadays.

### 3.2.1 Diagnosing Covid-19 Techniques

One of the best methods to diagnose the COVID-19 disease is real-time PCR (RT-PCR). Other tests, like rapid IgG and IgM were used before RT-PCR became the gold standard of detection. In Table 3.5 a summary of the most commonly used tests during the pandemic is presented.

| Test | Sensitivity [%] | Specificity [%] | Source |
|---|---|---|---|
| RT-PCR | 98.10 | 98.7 | [93] |
| Ct scan | 97.00 | 56.00 | [94] |
| Chest X-ray | 98.00 | 90.00 | [95] |
| Rapid antigen test | 99.4 | 68.4 | [96] |
| ELISA IgG | 92.00 | 100 | [97] |

Table 3.5: COVID-19 most used tests. The values for specificity and sensibility might vary depending on the laboratory that produced the test and conditions like the concentration of the genetic material extracted from the patient. In the RT-PCR test, there are Federal Drug Administration (FDA) and non-approved FDA, and some of the tests have sensitivity and specificity values of 100% for both FDA and non-FDA.

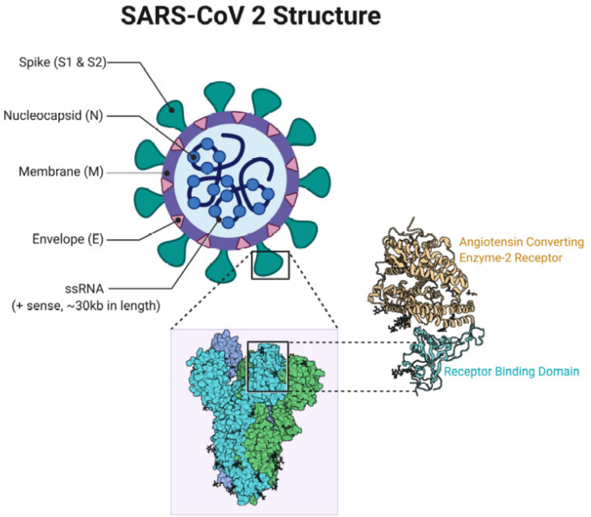


Figure 3.4: SARS-CoV-2 Structure. We can appreciate the spike protein, which is used for binding to the ACE2 (angiotensin-converting enzyme 2) receptor on the surface of human cells, facilitating the entry of the virus into those cells [98].

### 3.2.2 Covid-19 Treatment During the Pandemic

At the beginning of the pandemic, a medicine to treat the disease was not available. So, rest, fluids, fever, and pain medications were the treatments used during the first stages of the disease. When the patient developed pneumonia, antiviral medications were used without success. Monoclonal antibodies were also used in some recovery cases. The final stage required oxygen therapy and assisted ventilation due to the consolidation of the lungs. Finally, several vaccines were developed and distributed around the world [99]. Let us review the vaccines that were developed against the SARS-CoV-2 virus.

| Vaccine | Method | Route of administration | Number of doses |
|---|---|---|---|
| Pfizer/BioNTech | RNA based | IM | 2 |
| Moderna | RNA based | IM | 2 |
| AstraZeneca | Viral vector | IM | 1 or 2 |
| Janssen | Viral vector | IM | 1 or 2 |
| Sinovac | Inactivated virus | IM | 2 |
| Sinopharm | Inactivated virus | IM | 2 |
| CanSino | Viral vector | IM | 1 |

Table 3.6: COVID-19 vaccines characteristics [100].

In Table 3.6, we have a summary of the most common vaccines used around the world. They use different mechanisms to make the immunological body system recognize and rapidly eliminate the antigen before the virus infects and replicates.

## 3.3 Historical Perspectives on the Use of X-ray Images

The first X-ray images were taken by the well-known scientist Wilhelm Conrad Röntgen. He was holding with his fingers a tiny object in a beam that emitted X-rays. Then, Röntgen recognized the image of his fingers on the cardboard screen. So, he captured the first-ever X-ray image. The experiment radically changed medicine and is regarded as the beginning of radiography. He also took a picture of the hand of his wife, which showed the bones of the hand clearly, opening a new field of study called radiology [101].

### 3.3.1 Diagnosis of Pneumonia and other diseases

In the next few years, Professor Morris Manges published an article giving detailed annotations of the diagnosis of X-rays in patients, including bronchopneumonia, acute lobar pneumonia, abscesses, and gangrene [102]. During the Second World War, the use of X-rays to diagnose wounded soldiers became a national interest for the United States. Major Elberth Kenneth Lewis published in 1944 [103]. Kenneth and its medical team examined 6,000 cases, of which 25 percent presented with a type of pneumonia. They presented the X-ray images and the diagnosis of some patients in the publication. In conclusion, they said that an X-ray is not sufficient to accurately diagnose a patient with the aforementioned diseases. They suggested laboratory exams and physical examinations. Along with the discovery of X-rays in diagnosis, some researchers used these rays to treat patients. The appearance of radiotherapy was used before and during Roentgen's discovery [104].

### 3.3.2 Radiotherapy

The treatment of pneumonia using radiotherapy began during the first and second decades of the 20th century. In this paper [105], investigators diagnosed and treated patients in the United States who had several inflammatory conditions, including gangrene, inner ear infections, carbuncles, arthritis, and sinusitis [106]. Musser and Edsall published the first study on the use of X-rays in the treatment of pneumonia patients in 1905 [107]. Musser et al. hypothesized that X-ray therapy might contribute to the metabolism of the congestion produced by pneumonia. However, the use of radiotherapy was discontinued as sulfa drugs demonstrated more efficiency in treating bacterial forms of pneumonia [108].

## 3.4 Existing Methods for COVID-19 Detection Using X-ray Images Review

Nowadays, two main X-ray methods are used to diagnose COVID-19: chest X-rays and chest CT scans [109]. The sensitivity of both methods will be discussed further.

### 3.4.1 Using Chest X-rays as a Diagnostic Method

This method is quick and easy to use to produce an image of the lungs. These pictures can be used to detect fluid, pneumonia, and other abnormalities in the lungs. However, chest X-rays are not as sensitive as chest CT scans for detecting COVID-19 [110]. However, they are cheaper than CT scans. The price of a chest X-ray in Ecuador is around \$ 10 [111]. However, one disadvantage of chest X-rays is their high false-negative rate. The imaging test's immaturity and the absence of pulmonary disease at the time of presentation are two potential causes, particularly for portable X-ray equipment [112]. Also, breast prominence, inadequate inspiration, and improper patient placement can all result in false positive results on chest X-rays [112]. Many studies have examined the sensitivity of portable chest X-rays versus PCRs in the identification of COVID-19 in patients. These did not display particularly high sensitivity numbers at first, but in environments with a severe to moderate condition of the disease, it increased to as much as 89% [113]. In Figure 3.5, we appreciate arrows indicating how COVID-19 affects the lungs, presenting structures that can be recognized by doctors and specialists.

### 3.4.2 Diagnosis Using Chest Computerized Tomography Scan

Chest CT scans offer more image resolution than chest X-rays and can provide more accurate details of the lungs. They are also better at detecting COVID-19. However, chest CT scans are more expensive and require more radiation than chest X-rays [114]. The price of chest CT scans in Ecuador varies from \$ 150 to \$ 300 in the private sector [115]. Now, let us review the sensitivity of a chest X-ray vs. a CT scan.

### 3.4.3 The Sensitivity of Chest X-ray vs CT Scan

The sensitivity of a portable chest X-ray machine is lower than that of a CT scan, with 69% versus 97–98% respectively [110]. However, in this paper [113], they report similar sensitivity values for chest and CT scans. The reason for the sensitivity values to be so close might be the use of 535 patients aged 65±17 years, which is a low number of patients and the age is close to each other. They also used patients with severe symptoms of the disease, which can enhance sensitivity in chest X-rays. Another paper states that, when COVID-19 infection is mild, chest imaging is insensitive [110]. So, the severity of the patient's disease has to be taken into account when chest X-rays are used to diagnose a patient. However, chest images can be used to triage the sick and aid doctors in the diagnosis [116].

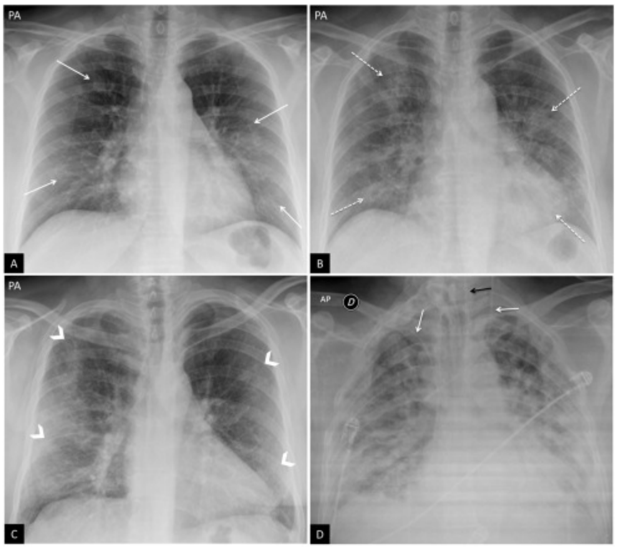


Figure 3.5: Annotations in COVID-19 pneumonia. (A) Woman 47 years old with clear symptoms of COVID-19. Posteroanterior (PA) chest X-ray. The image shows a reticular interstitial pattern peripheral to the lungs (arrows). (B) The same patient is in picture A. Three days later, a PA chest X-ray was taken. SARS-CoV-2 positive PCR result. The X-ray displays faint, rounded bilateral peripheral alveolar opacities (dotted arrows). (C) A male aged 57 who exhibits dyspnea and a positive SARS-CoV-2 PCR has bilateral peripheral opacities (arrow tips) in the middle, lower, and upper fields. (D) A 45-year-old man whose COVID-19 status was verified by PCR along with dyspnea. Anteroposterior chest X-ray demonstrating extensive involvement of both lungs in multiple bilateral diffuse confluent areas of consolidation. Observe the existence of two central venous lines, a gastrointestinal tube (black arrow), and the left and right subclavian jugular veins (white arrows). [109].

## 3.5 Artificial Intelligence in Chest X-rays with COVID-19

Artificial intelligence (AI) has been used to train and predict if a patient has a pulmonary disease using X-ray images [117]. In this paper [118], they used chest and CT scan images. Then they segmented the images using AI and reported high precision in identifying diseased regions in CT and X-ray images. They said that AI can help physicians diagnose patients more quickly and make it easier to determine how serious a patient's infection is afterward. Also, radiologists can receive assistance in screening, diagnosis, and treatment from a machine-learning-based medical assistant platform. Jiao et al. [119] state that chest X-rays may complement clinical information in predicting the likelihood that COVID-19 patients will progress to a critical condition through AI.

### 3.5.1 High Precision Models Using Chest X-rays

Neural networks are being used to accurately predict patterns in new images. Now, I will mention some of these frameworks that have had good results. The Bayesian DNN network has an accuracy of 92.9%. It classifies COVID and other classes not mentioned in the paper [120].

The ResNet50, InceptionV3 and Inception- ResNetV2 [121], accuracies are seen in Table 3.7.

| Model | Binary classification [%] |
|---|---|
| ResNet50 | 98.00 |
| InceptionV3 | 97.00 |
| Inception-ResNetV2 | 87.00 |

Table 3.7: Accuracy ResNet50, InceptionV3, and Inception-ResNetV2 models. All the last-mentioned models classify COVID-19 and normal patients. The number of subjects used in the models is 50 for COVID-19 and 1950 for Normal.

DarkNet (YOLO) [122]. Researchers obtained an accuracy of 98.08% for binary classes (COVID vs. No-Findings) and 87.02% for multi-class (COVID vs. No-Findings vs. Pneumonia). They used 224 COVID-19 images, 700 for bacterial pneumonia, and 504 for normal patients. They used balancing methods to select only 500 images from bacterial pneumonia and 500 from normal. They also mentioned that the number of images that they used for COVID-19 was low due to the lack of positive cases at the time they conducted the study.

### 3.5.2 Segmentation Algorithms

The VB-Net neural network was used in the DL-based segmentation method to identify COVID-19-specific infection regions in CT scans. CT scans from 249 COVID-19 patients were used to train the newly developed DL-based segmentation neural network. An additional 300 COVID-19 CT scans were used to validate the system further [118]. They reported a Dice similarity coefficient of 91.6% which is quite high, indicating that the automatic segmentation results are very similar to the manual segmentations. This suggests that the VB-Net is doing a good job at accurately identifying and delineating the regions of interest in the images. They also obtained the best accuracy of severity prediction of 73.4% ± 1.3%. This accuracy score represents how well the model can predict the severity of lung infections. An accuracy of 73.4% means that the model's predictions match the actual clinical severity in 73.4% of cases, which is a reasonable level of accuracy. Other neural networks.

Liu et al. [123] mentioned that one of the best neural networks for image segmentation is U-Net. In contrast to Deeplab26, SegNet19, and FCN25Figure. Also, Unet performs well when given a few samples of the images and masks. In Figure 3.6, we observe that the network receives an image as input in the format 256*256*3, and the number 3 indicates a color image. Then, the output is 256*256*1, meaning that the resulting image is a mask in black and white. They also found that, when the lung field is obscured by serious diseases, blocked by medical equipment, or severely deformed, the accuracy of lung segmentation is comparatively low. This remark is important when using U-net in COVID-19 datasets. In contrast, in this paper [124], the dice scores for SCAN and Attention-U-Net-U-Net are 97.3 ±0.8% and 96.3 ±0.7% respectively, meaning that the SCAN neural network is slightly better than the Attention-U-net architecture. They also used the Contrast Limited Adaptive Histogram Equalization (CLAHE) filter in Figure 3.7, to improve the visualization of human and computer vision.

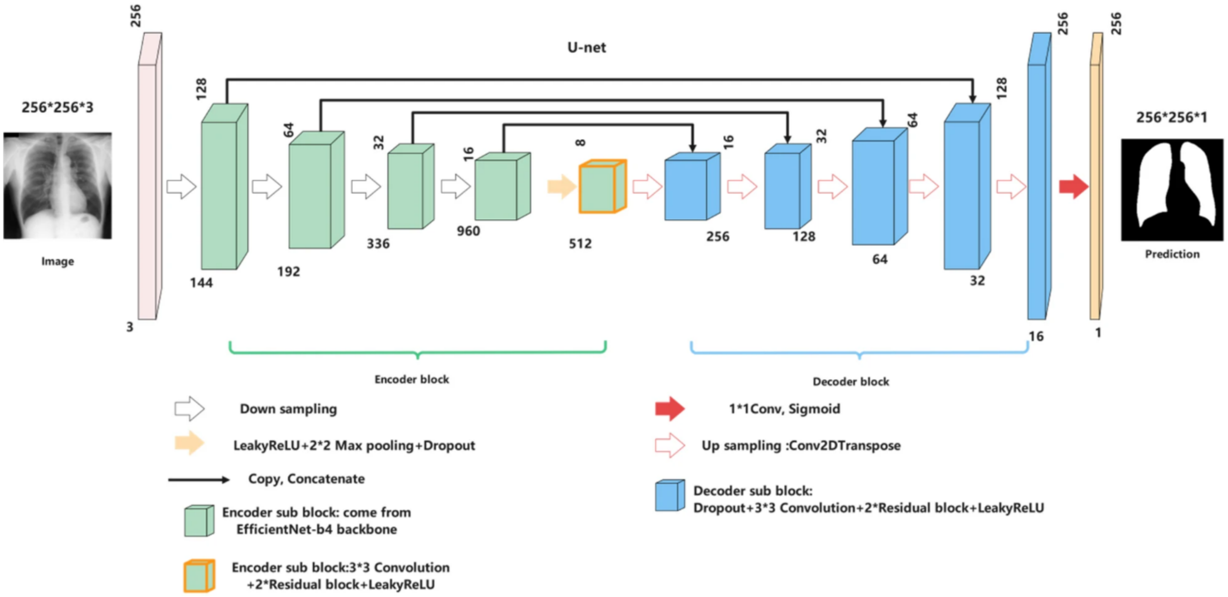

U-net
256*256*3
Image
256
3
128
144
64
192
32
336
16
960
8
512
16
256
32
128
64
64
128
32
256
16
256
1
256*256*1
Prediction
Encoder block
Decoder block
Down sampling
LeakyReLU+2*2 Max pooling+Dropout
Copy, Concatenate
Encoder sub block: come from EfficientNet-b4 backbone
Encoder sub block:3*3 Convolution +2*Residual block+LeakyReLU
1*1Conv, Sigmoid
Up sampling :Conv2DTranspose
Decoder sub block: Dropout+3*3 Convolution+2*Residual block+LeakyReLU

Figure 3.6: U-Net architecture with EfficientNet Encoder [125].

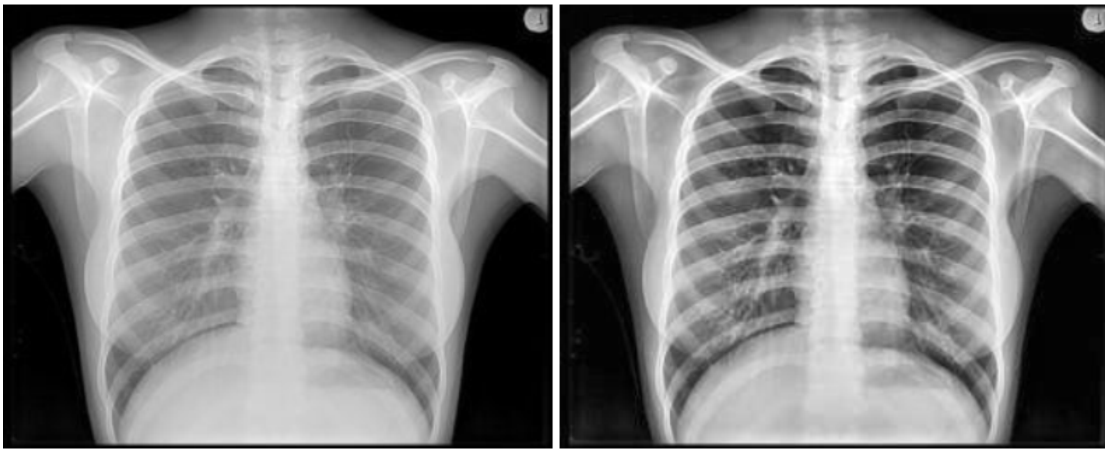


Figure 3.7: CLAHE filter applied in Chest X-ray image before and after [124].

### 3.5.3 Using Contrast Limited Adaptive Histogram Equalization

CLAHE technique uses a pre-trained neural network that has been trained with a huge dataset and transfers that knowledge to a new model, which will receive a dataset of a smaller sample size. The new model is trained using this transferred knowledge and compiles much faster [126]. In this article [127], they used transfer learning models and obtained these results for accuracy in Table 3.8. We can observe that VGG-19 performed better than the other models for both binary and other classes.

| Model | Binary classification [%] | Three classes [%] |
|---|---|---|
| VGG-19 | 98.75 | 93.48 |
| MobileNet | 97.40 | 92.85 |
| Inception | 86.13 | 92.85 |
| Xception | 85.57 | 92.85 |
| Inception ResNet v2 | 84.38 | 92.85 |

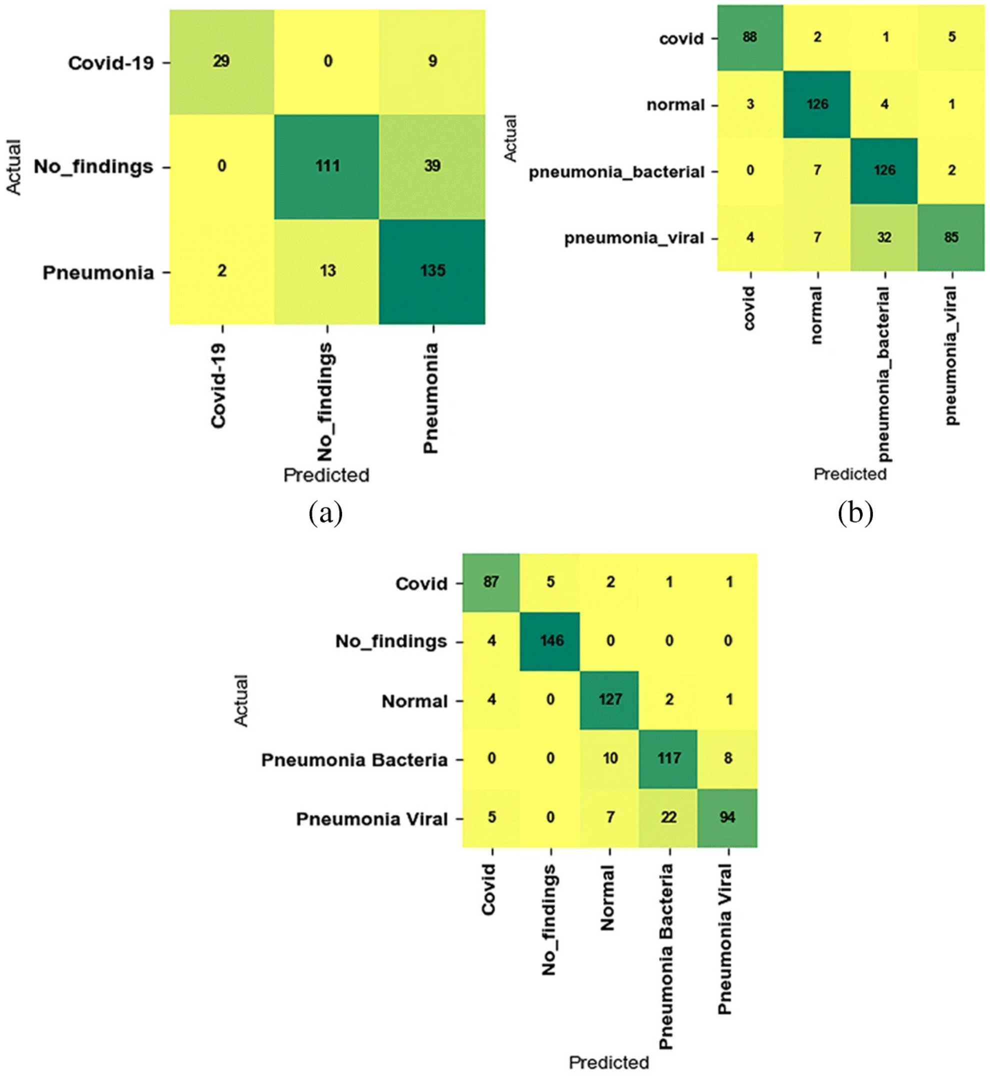
Table 3.8: Several transfer learning model accuracies are reported. The binary classification used the COVID and Normal classes. The three-class classification used COVID, Normal, and Pneumonia

Sitaula et al [128], used the VGG-16 transfer learning model with chest X-ray images. They used three datasets containing an average of 300 images for each class. The first dataset has 3 classes, the second has 4 classes, and the third has 5 classes. Classification accuracy values for datasets 1, 2, and 3 are reported in Table 3.9.

| Model | Three classes [%] | Four classes [%] | Five classes [%] |
|---|---|---|---|
| VGG-16 | 79.58 | 85.43 | 87.49 |
| VGG-19 | 74.84 | 82.83 | 85.00 |

Table 3.9: VGG16 and VGG19 transfer learning model accuracies.

Sitaula and its collaborators found that VGG-16 performed slightly better than VGG-16. In Figure 3.8, we observe the confusion matrix for VGG-16 for the 3 datasets used. It is important to mention that they used an internal testing dataset extracted from the training dataset. The correct method would be using an external testing dataset [129], which simulates better a real-life application with completely new images. This external dataset will ensure a more robust evaluation of the model and its ability to perform on unseen data.



| Actual \ Predicted | Covid-19 | No_findings | Pneumonia |
|---|---|---|---|
| Covid-19 | 29 | 0 | 9 |
| No_findings | 0 | 111 | 39 |
| Pneumonia | 2 | 13 | 135 |

(a)

| Actual \ Predicted | covid | normal | pneumonia_bacterial | pneumonia_viral |
|---|---|---|---|---|
| covid | 88 | 2 | 1 | 5 |
| normal | 3 | 126 | 4 | 1 |
| pneumonia_bacterial | 0 | 7 | 126 | 2 |
| pneumonia_viral | 4 | 7 | 32 | 85 |

(b)

| Actual \ Predicted | Covid | No_findings | Normal | Pneumonia Bacteria | Pneumonia Viral |
|---|---|---|---|---|---|
| Covid | 87 | 5 | 2 | 1 | 1 |
| No_findings | 4 | 146 | 0 | 0 | 0 |
| Normal | 4 | 0 | 127 | 2 | 1 |
| Pneumonia Bacteria | 0 | 0 | 10 | 117 | 8 |
| Pneumonia Viral | 5 | 0 | 7 | 22 | 94 |

(c)

Figure 3.8: Confusion matrix for (a) First dataset with 3 classes, (b) Second dataset 4 classes, and (c) Third dataset that contains 5 classes [128].

### 3.5.4 COVID-19 Chest X-ray Public and Private Datasets

There are public and private COVID-19 datasets that were available during the pandemic to aid experts in the diagnosis of this disease. The Kaggle database, *COVID-19 Radiography Database* [1], has 3616 COVID-19 positive cases along with 10,192 Normal, 6012 Lung Opacity (Non-COVID lung infection), and 1345 Viral Pneumonia. These papers [130], [131], use the "COVID-19 Radiography Database" to construct artificial intelligence models obtaining good prediction results. Another more recent COVID-19 database called *COVID-QU-Ex Dataset* has 33,920 chest X-ray (CXR) [2] and also has masks for segmentation. The classes and number of images are 11,956 for COVID-19, 11,263 for non-COVID

---

[1] `https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database`

[2] `https://www.kaggle.com/datasets/anasmohammedtahir/covidqu/?select=COVID-QU-Ex+dataset.txt`

infections (Viral or Bacterial Pneumonia), and 10,701 for Normal. The authors of the database have constructed the database using images from other databases. In Figure 3.9, we have 3 images obtained from the *COVID-QU-Ex Dataset*. The dataset has also been used in other investigations with successful results [132], [131]. The dataset is also well-balanced, and we might expect very good results by using it in a neural network.

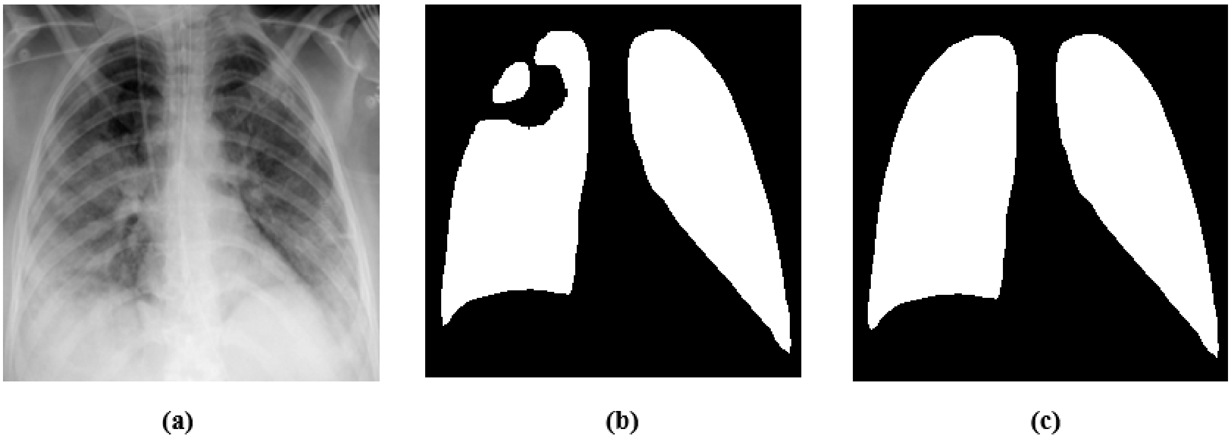


Figure 3.9: Images extracted from the *COVID-QU-Ex Dataset*. (a) A CXR of a patient with COVID-19. (b) A mask of the infection produced by the disease. (c) Lung mask not considering the infection [11].

In this work [132], the *COVID-QU-Ex Dataset* was used to test several neural networks using U-net segmentation. The best results are shown in Table 3.10.

| Model | Encoder | F1 Score[%] | Accuracy [%] | Sensitivity[%] | Specificity [%] |
|---|---|---|---|---|---|
| U-Net | DenseNet121 | 98.81 ± 0.62 | 98.8 ± 0.62 | **99.66 ± 0.33** | 97.94 ± 0.82 |
| U-Net ++ | ResNet18 | 99.22 ± 0.5 | 99.23 ± 0.5 | 98.46 ± 0.71 | **100 ± 0** |
| U-Net ++ | DenseNet121 | 99.22 ± 0.5 | 99.23 ± 0.5 | **99.31 ± 0.48** | 99.14 ± 0.53 |
| FPN | ResNet18 | 98.56 ± 0.68 | 98.54 ± 0.69 | **99.66 ± 0.33** | 97.43 ± 0.91 |

Table 3.10: *COVID-QU-Ex Dataset* tested on state-of-the-art neural networks using segmentation. The U-Net-DenseNet121 and FPN-ResNet18 have the highest sensitivity meaning that they can classify patients with the disease excellently. U-Net++-ResNet18 has the perfect specificity, which means that this neural network classifies, without error, healthy patients. Future work might include replicating this work and trying to achieve similar results using transfer learning with segmentation. They only used three classes, since pneumonia and lung opacity have similar patterns and they merged both classes to get higher performance statistics.

In Figure 3.10, we observe the result of the segmentation performed in real CXR images with COVID-19. Since they are using the highest-performance neural networks, the segmentation of the areas commented in the lungs is highly accurate. This is the final tool we would like to build and deploy in several public hospitals. Tahir et. al [132] have demonstrated that is possible to train and predict AI with the use of CXR images with almost no error.

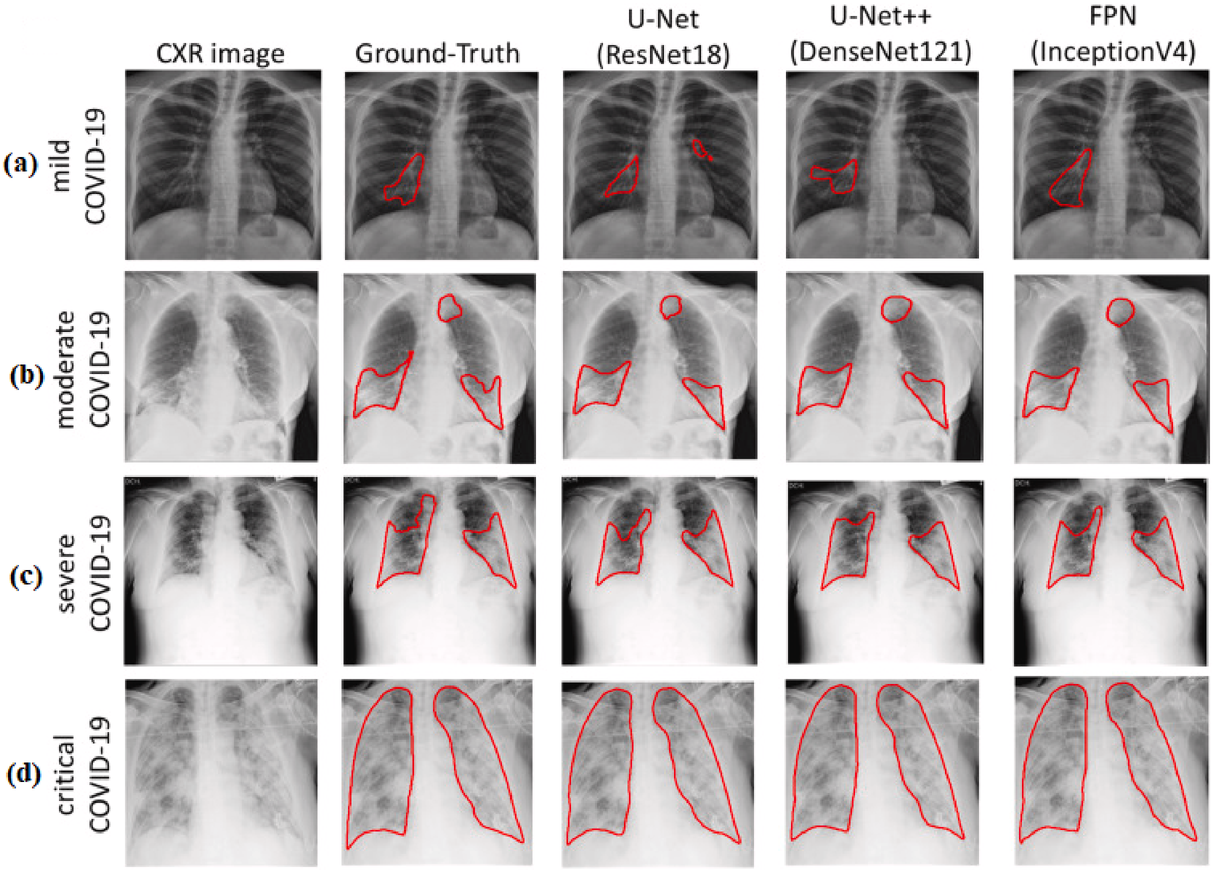


Figure 3.10: *COVID-QU-Ex Dataset* segmentation and prediction. (a) Four images from the 4 stages of COVID-19 are depicted. In mild COVID-19, the CXR has a lot of contrast and the lungs do not present opacities (lighter areas). (b) In a moderate state, opacities are located in the lower areas of the lungs. (c) In severe progression, the lungs have a lot of lighter areas which means that are filled with fluid complicating breathing. The patient in this state is very likely to need assistance to breathe. (d) In the critical phase, the patient has to be intubated and monitored constantly to keep him alive. Observe how the white areas cover all the dark initial space that a healthy lung has [132].

# Chapter 4

# Methodology

## 4.1 Methodology Summary

The model was created using Jupyter Notebooks on the Google Colab and Kaggle platforms. Keras, tensorflow, pandas, numpy, sklearn, matplotlib, PIL, among other libraries were used. The dataset was selected taking into consideration the number of images and the resolution. The dataset was first checked for images that were not part of the Chest-Xray category and other unnecessary archives. Then it was compressed and uploaded to my Google Drive account. Then, the dataset was downloaded in each session of compiling by using the gdown library. So, the downloading only took 6 seconds per session using an average of 208MB/s of speed. Then the dataset was unzipped in the Colab and Kaggle platforms. Finally, the dataset was balanced to obtain the same amount of images for each class. The final size of the dataset is shown further. To train the model we stored the images of the dataset using Keras library. Then, we used grayscale images since color images did not show model accuracy improvement. Image augmentation, autotune, and randomizing were applied. A seed=123 was selected to ensure reproducibility so that the randomization process would be done in a deterministic manner.

The model architectures were chosen due to recent works that showed better performance than Alexnet and other old neural networks. Initially, we tested Alexnet and Inception but the literature shows better results using VGG19 and VGG16 models combining transfer learning techniques. The models used in this work were tested on binary and multiclass datasets that we created from the *COVID-19 RADIOGRAPHY DATABASE* dataset. During the validation process, we used binary and categorical accuracy. Also, binary and categorical cross-entropy were used during the model compiling. The Adam optimizer was used. Heat maps and Grad-Cam were used to visualize the neural network activations. Receiver Operating Characteristic (ROC) was used to visually determine the model performance. ROC is commonly used in binary classification using different threshold values to plot the data. In simpler terms, it's a way to visualize the performance of a binary classification model across different threshold settings. Threshold values determine the boundary in the prediction of classes.

Kaggle provided the *COVID-19 RADIOGRAPHY DATABASE* free dataset. The dataset contains a total of 27.052 images divided into 4 classes. The balancing of the dataset was used to improve the accuracy of the results. Also, data augmentation was applied to increase the number of images. Then, the dataset was divided into training, validation, and testing. A 70% was used for training, 15% for validations, and 15% for testing. Also, prefetching and shuffling were used to ensure the reproducibility of this work. The architecture of one of the neural networks is presented. Finally, the platforms we used and the hardware are shown.

In Figure 4.1 we appreciate an image produced using one of the samples from the COVID-19 RADIOGRAPHY DATABASE. This image allows the reader to understand the basic mechanism of the extraction of patterns performed by a CNN. The yellow areas represent specific characteristics that the model is learning. It is important to mention that this neural network is looking for characteristics unique to each disease and not looking exactly at the lung area.

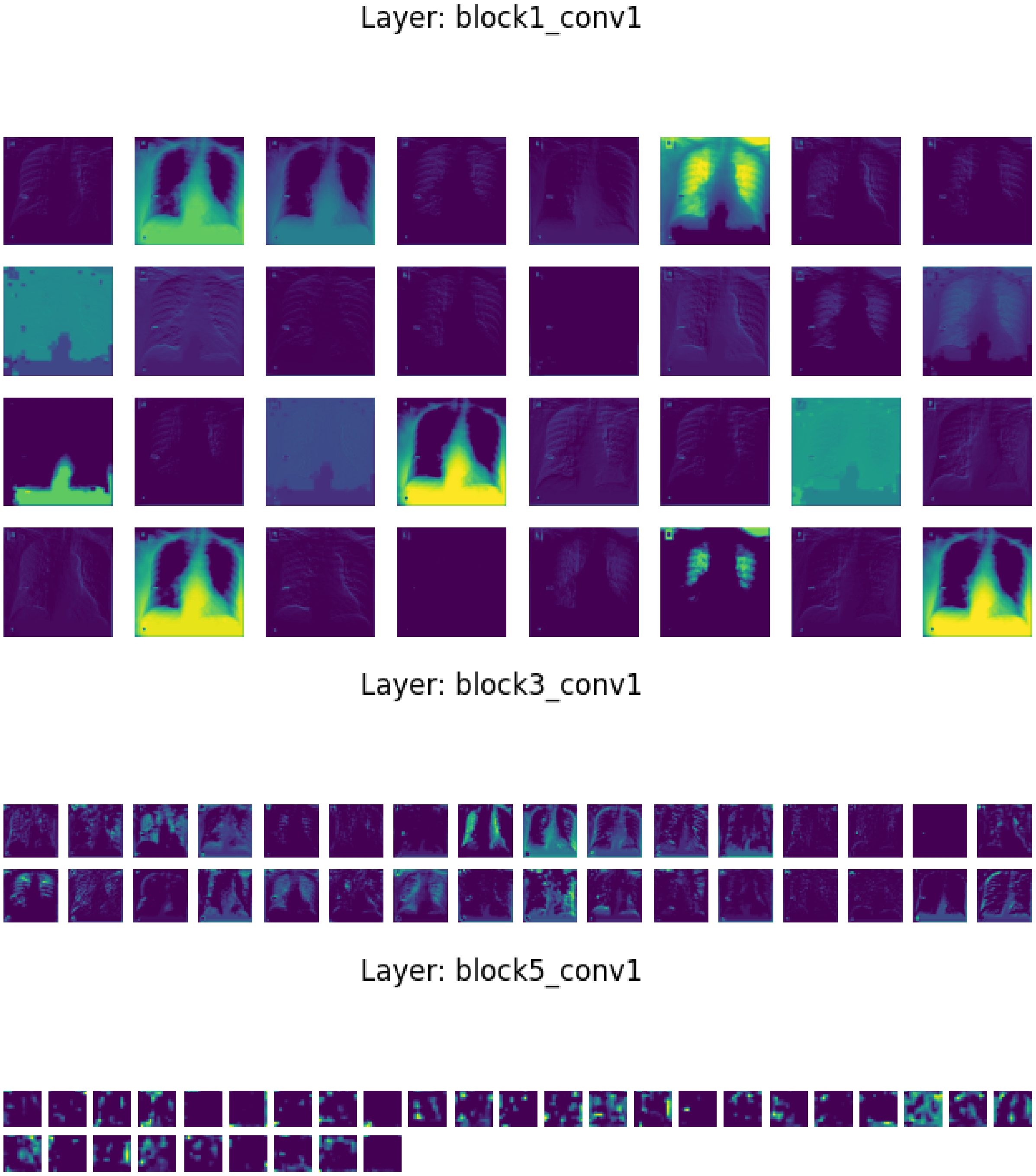


Figure 4.1: VGG19 neural network characteristics were extracted in each layer. In the **block1-conv1** layer, we appreciate how the CNN extracts the characteristics from the same image sixteen times. The **block3-conv1**, which is in the middle of the CNN, extracts more characteristics of the initial chest X-ray image. The **block5-conv1**, which is one of the last layers in the VGG19 CNN, extracts the final patterns. The reduction in size of the initial image is also noted.

## 4.2 Public Dataset

The public ”COVID-19 RADIOGRAPHY DATABASE” [133] was used in this experiment. A database of chest X-ray images for COVID-19-positive cases, along with normal and viral pneumonia images, was created by a team of researchers from Qatar University, Doha, Qatar, and the University of Dhaka, Bangladesh, along with their collaborators from Pakistan and Malaysia, in collaboration with medical professionals. The creators of the database made a first release that contained 219 COVID-19, 1341 normal, and 1345 viral pneumonia chest X-ray (CXR) images. In the second release, they expanded the COVID-19 class to 1200 CXR images. In the third update, they added 10,192 ”Normal”, 6012 ”Lung Opacity” (non-COVID lung infection), 1345 ”Viral Pneumonia” images, and matching lung masks to the database, bringing the total to 3616 COVID-19 positive cases [130], [131].

Table 4.1 contains the characteristics of the database [133] when accessed. In Figure 4.2, a sample from the ”COVID” class was extracted.

| | |
|---|---|
| **Total of images** | 27,052 |
| **Normal class images** | 12157 |
| **COVID class images** | 3815 |
| **Viral Pneumonia images** | 5068 |
| **Lung Opacity images** | 3815 |
| **Format** | .jpg |
| **Images initial size** | 400 × 400 |
| **Shape** | (400,400,3) |

Table 4.1: Dataset characteristics. The number 3 in shape indicates that we have an RGB format for the initial images.

### 4.2.1 Balancing and Augmentation Techniques

The dataset was balanced to improve testing accuracy. The Python code counted the number of images in each folder class. Then the minimum number was 3815 images for the ”COVID” folder. Next, the minimum number was used to randomly select samples from all the other classes. The process was done until all the classes had the same number of images as the minimal number. Then a new balanced dataset was created with a total of 15260 images. Next, the balanced dataset was divided into training, validation, and testing. A 70% was used for training, 15% for validations, and 15% for testing. The percentages can be adjusted in the code as needed. Then, the images were loaded using Keras libraries. The dataset load parameters were: $height = 280$, $width = 280$, $batchsize = 32$, and $seed = 123$ ”to ensure reproducibility when the batch size is randomized”. Color images were used in all the experiments due to the lack of significant changes when using grayscale samples.

Augmentation was done using the Keras package with $random - flip = "horizontal"$ and $random - rotation = 0.1$. In Figure 4.3, we can observe how the flipping and rotation

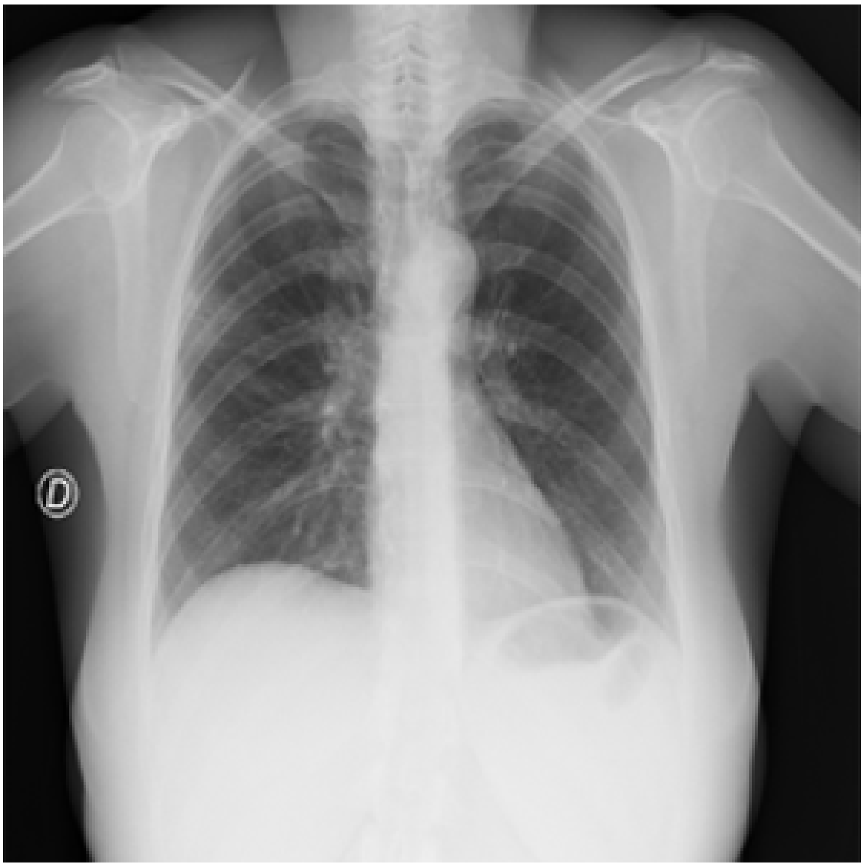


Figure 4.2: Chest X-ray of a patient diagnosed positive for COVID-19 infection.

work. Then the models were constructed using the next structure of classes: COVID and Normal for binary classification, ”COVID” ”Normal” and ”Viral Pneumonia”, and ”COVID” ”Normal” ”Viral Pneumonia” and ”Lung Opacity”.

### 4.2.2 Additional Optimizers

The ”tf.data.Dataset” module provided extra tools to handle our image datasets efficiently. Firstly, the **cache()** method is applied to training, validation, and test datasets. This method efficiently caches dataset elements in memory, significantly reducing data loading times during subsequent iterations and preventing the re-computation of expensive operations.

To introduce randomness and prevent potential biases during model training, we employ the **shuffle()** method. The shuffling operation randomizes the order of elements within the dataset. We used a buffer size of 1000 elements to ensure effective randomization. This is particularly beneficial when the original dataset has any inherent ordering, such as all samples of a specific class being grouped together in the same folder.

Also, we optimized the data loading process by incorporating the **prefetch()** method. Prefetching allows the model to process one batch while the other is processed in the background. The buffer size, specified by the AUTOTUNE variable, can be set to auto or

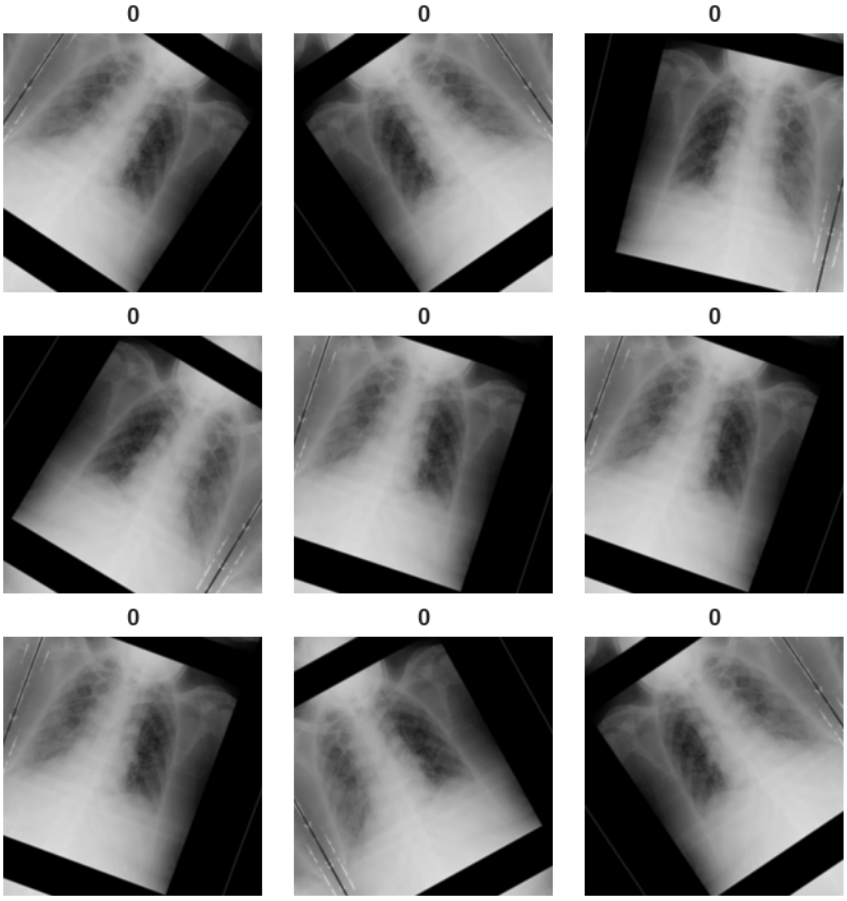


Figure 4.3: Nine randomly selected images from the dataset are shown after using *Augmentation* technique.

a number. We used $AUTOTUNE = 10$, which means that 10 images are ready to be processed in the background while the model is being trained.

These preprocessing techniques contribute to effective data processing, ensuring that our machine-learning models are trained on diverse, randomized, and efficiently loaded datasets. These optimizations are crucial for achieving better model statistics.

## 4.3 Transfer Learning Architectures

The models we used were downloaded using the Keras module. The transfer learning method was used in all the experiments. To summarize ”transfer learning”, we followed the next steps:

1. Using a previously trained model, extract its layers.
2. Freeze the layers to prevent any information from being lost for use in later training cycles.

3. On top of the frozen layers, add a few fresh, trainable layers. They will acquire the ability to forecast using the previous features on a fresh dataset.

4. Utilizing the dataset processed to train the new layers.

5. Fine-tuning, which is retraining the model with a very low learning rate on the new data after unfreezing the complete model you obtained above. This method has the potential to yield significant enhancements by gradually adjusting the pre-trained characteristics to the updated data.

The pre-trained neural networks used are Xception, VGG16, and VGG19. These CNNs can be downloaded using the Keras module and following the steps detailed in the documentation page of the module [134]. In table 4.2, we have the summary of the parameters of Xception.

| Input size | Output size | Layer | Stride | Kernel |
|---|---|---|---|---|
| 299×299×3 | 299×299×32 | Conv2d | 2 | 3×3 |
| 299×299×32 | 149×149×64 | Conv2d | 2 | 3×3 |
| 149×149×64 | 149×149×64 | SeparableConv2d | 1 | 3×3 |
| 149×149×64 | 149×149×128 | SeparableConv2d | 1 | 3×3 |
| 149×149×128 | 74×74×128 | MaxPooling2d | 2 | 3×3 |
| 74×74×128 | 74×74×256 | SeparableConv2d | 1 | 3×3 |
| 74×74×256 | 74×74×256 | SeparableConv2d | 1 | 3×3 |
| 74×74×256 | 37×37×256 | MaxPooling2d | 2 | 3×3 |
| 37×37×256 | 37×37×728 | SeparableConv2d | 1 | 3×3 |
| 37×37×728 | 37×37×728 | SeparableConv2d | 1 | 3×3 |
| 37×37×728 | 37×37×728 | SeparableConv2d | 1 | 3×3 |
| 37×37×728 | 19×19×728 | MaxPooling2d | 2 | 3×3 |
| 19×19×728 | 19×19×1024 | SeparableConv2d | 1 | 3×3 |
| 19×19×1024 | 19×19×1024 | SeparableConv2d | 1 | 3×3 |
| 19×19×1024 | 19×19×2048 | SeparableConv2d | 1 | 3×3 |
| 19×19×2048 | 10×10×2048 | MaxPooling2d | 2 | 3×3 |
| 10×10×2048 | 10×10×2048 | SeparableConv2d | 1 | 3×3 |
| 10×10×2048 | 10×10×2048 | SeparableConv2d | 1 | 3×3 |
| 10×10×2048 | 5×5×2048 | MaxPooling2d | 2 | 3×3 |
| 5×5×2048 | 1×1×2048 | Conv2d | 1 | 1×1 |
| 1×1×2048 | 1×1×1000 | Conv2d | 1 | 1×1 |

Table 4.2: Specifications of the Xception model layers.

In Figure 4.4, we present the final architecture of the VGG16 model. We can appreciate that the pre-trained VGG16 is located within the newly trained model using X-ray images. Also, in Figure 4.5, the VGG16 neural network can be seen within the dashed lines. Both models were plotted using the Keras module.

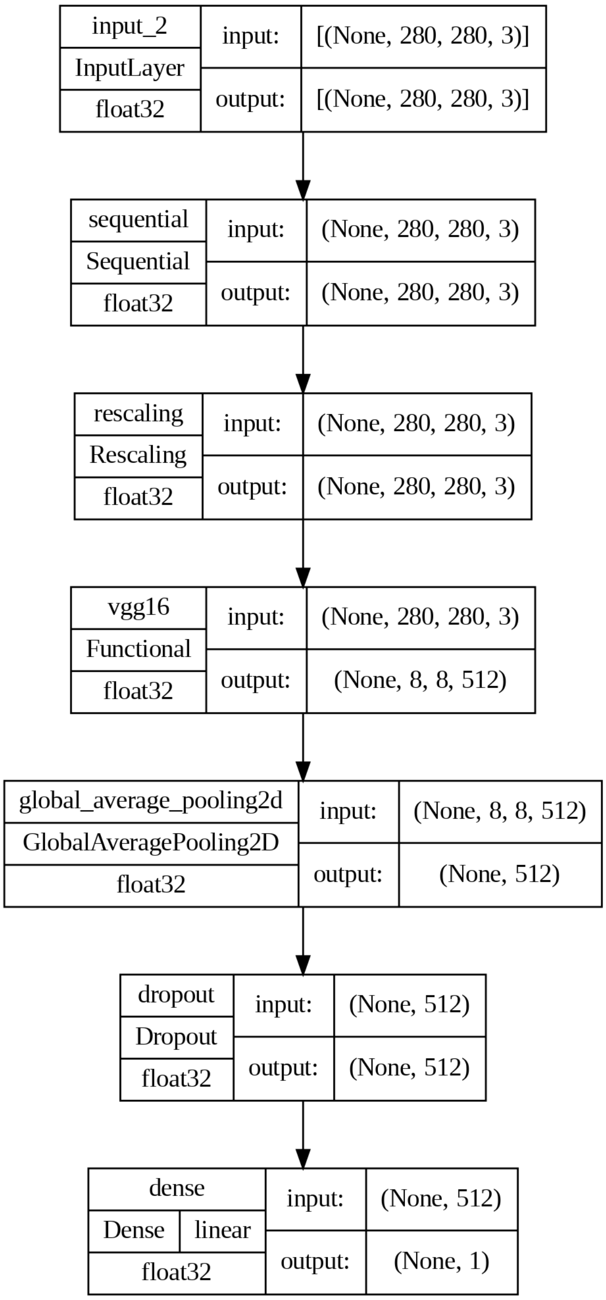


Figure 4.4: The VGG16 final model was trained using the processed chest X-ray dataset. We observe the input layer and the size of the images used during the training process.

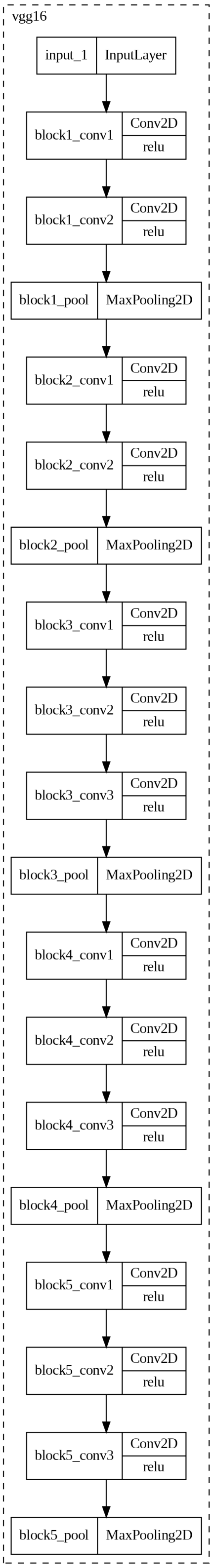


Figure 4.5: Expanded VGG16 model located in Figure 4.4. The pre-trained VGG16 is shown with its complexity of several layers and the ReLU as an activation function in several of its layers.

In Table 4.3, we can observe the detailed parameters for the VGG19 network.

| Input size | Output size | Layer | Stride | Kernel | Activations |
|---|---|---|---|---|---|
| 224×224×3 | 224×224×64 | Conv3-3 | 1 | 3×3 | ReLU |
| 224×224×64 | 224×224×64 | Conv3-3 | 1 | 3×3 | ReLU |
| 224×224×64 | 112×112×64 | MaxPool2 | 2 | 2×2 | None |
| 112×112×64 | 112×112×128 | Conv3-3 | 1 | 3×3 | ReLU |
| 112×112×128 | 112×112×128 | Conv3-3 | 1 | 3×3 | ReLU |
| 112×112×128 | 56×56×128 | MaxPool2 | 2 | 2×2 | None |
| 56×56×128 | 56×56×256 | Conv3-3 | 1 | 3×3 | ReLU |
| 56×56×256 | 56×56×256 | Conv3-3 | 1 | 3×3 | ReLU |
| 56×56×256 | 56×56×256 | Conv3-3 | 1 | 3×3 | ReLU |
| 56×56×256 | 28×28×256 | MaxPool2 | 2 | 2×2 | None |
| 28×28×256 | 28×28×512 | Conv3-3 | 1 | 3×3 | ReLU |
| 28×28×512 | 28×28×512 | Conv3-3 | 1 | 3×3 | ReLU |
| 28×28×512 | 28×28×512 | Conv3-3 | 1 | 3×3 | ReLU |
| 28×28×512 | 14×14×512 | MaxPool2 | 2 | 2×2 | None |

Table 4.3: Specifications of the VGG19 model layers.

## 4.4 Optimizers and Loss Function

Transfer learning is done using two stages of training on the dataset. The first is with some layers frozen to keep the learned patterns, and the second, called fine-tuning, is performed by unfreezing all the layers of the model. Fine-tuning is typically compiled with low epoch numbers. We used 20 epochs for the freezing stage and 10 epochs for fine-tuning.

### 4.4.1 Default Adam Values

The next values correspond to the line code $optimizer = keras.optimizers.Adam()$ for the first stage of training: Learning rate $(\eta) = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$.

**The update equations for Adam using the default values are as follows:**

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{4.1}$$

Where:

- $t$ is the iteration, and the initial values are $m_0 = 0$, and $v_0 = 0$.

- $m_t$ and $v_t$ are the first and second moments, which are estimates of the mean and uncentered variance of the gradients, respectively.
- $g_t$ is the gradient of the objective function concerning the parameters at iteration $t$ and is calculated in each training pass.
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the first and second moments.
- $\beta_1$ and $\beta_2$ are the exponential decay rates for the first and second moments, respectively.
- $\eta$ is the learning rate.
- $\epsilon$ is a small constant and is usually added for numerical stability with a value of $1 \times 10^{-8}$.

The aforementioned formulas collectively allow the Adam optimizer to adapt the learning rates for each parameter during each training iteration. Equation 4.1 inputs the calculated parameters and the learning rate to update the next iteration. Also, the use of moments with bias correction helps to simplify the optimization process.

### 4.4.2 Binary and Categorical Cross-entropy

The Equation 4.2 is used by Keras as a loss function. It is used in binary classification problems, but it is not limited to handling only two classes. We chose binary cross-entropy as suggested by Keras. For the three and four classifications we used categorical cross-entropy because binary gave us compiling errors.

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left[ y_i \cdot \log\left(\hat{y}_i\right) + (1 - y_i) \cdot \log\left(1 - \hat{y}_i\right) \right] \tag{4.2}$$

Where:

- $N$ represents the total number of samples.
- $y_i$ is the real label for the $i$ sample.
- $\hat{y}_i$ is the prediction produced by the model for the $i$ sample.
- log is the natural logarithm function.

Equation 4.2 compares the predicted output $\hat{y}_i$ with the real label $y_i$ and sanctions the differences between them. The part of the equation: $y_i \cdot \log\left(\hat{y}_i\right)$ penalizes the incorrect prediction when the real label has the value 1, and the part: $(1 - y_i) \cdot \log\left(1 - \hat{y}_i\right)$ sanctions the incorrect prediction when the real label is 0. The total loss is the average sum of the penalties for all the samples.

### 4.4.3 Fine Tuning and Metrics

We used a learning rate of $\eta = 1 \times 10^{-5}$ in the Adam optimizer during **fine-tuning**, as suggested by Keras documentation. This means that during training, the model's weights will receive small updates concerning their current values. For binary cross-entropy and binary accuracy, the default values of Keras were used.

For the metrics calculation, we used the binary accuracy described in Equation 4.3. To report the results, we used the equations described previously in introduction section, and finally, to visualize the results, we used the confusion matrix plotting explained in Section 2.6.5.

$$\text{Binary Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}} \tag{4.3}$$

We stored in Google Drive the trained models for all the classes. They can be used in deployments whether online or offline.

## 4.5 Software and Hardware Used in Our Experiment

Python code can be written and run in a collaborative and interactive environment with Google Colab, a free cloud-based platform. The capabilities of the application, which are based on Jupyter Notebooks, include free GPU access for resource-intensive projects, real-time collaborative editing, easy sharing via Google Drive integration, pre-installed libraries for machine learning and data analysis, and support for exporting notebooks in multiple formats. Because of its convenience and accessibility, Colab is widely used in data science and machine learning applications. During the experiments, we used Google Colab, which used the Python 3 version. We also used the 2.14.0 ”Tensorflow” version on the Jupiter Notebooks. We also used the Kaggle plattaform.

Google Cloud Platform (GCP) and Kaggle offer access to strong GPUs, such as T4 GPUs Figure 4.6. Based on the NVIDIA Turing architecture, the T4 GPU is a good choice for workloads that require GPU acceleration, such as machine learning. In the free version, the one we used for this research, GCP offers 1 GPU with 16 GB of RAM of the type GDDR6. Kaggle offered up to 30 GB of RAM, doubling the GPU RAM size of Colab. Both the Colab and Kaggle platforms were used to run our code. The code we created for this work can be publicly accessed on Kaggle [135].

Figure 4.6: T4 tesla GPU offered in GCP for machine learning purposes. It was used to accelerate the training model on the COVID dataset [136].

# Chapter 5

# Results

## 5.1 Results Experiment Introduction

The need for a tool to predict whether or not a patient had COVID-19 appeared during the pandemic. Neural networks have proven to be very effective at learning complex image patterns. So, chest X-ray images that have complicated structures were used in this work. Our objective is to create and test state-of-the-art neural networks that have been used in the prediction of COVID-19 using chest X-rays and try to replicate and possibly improve the results that have been reported so far. Also, the models will be tested with different classes, and the results will be reported to conclude which is better. The trained models could be used by Information Technology (IT) developers to create mobile apps and websites that can aid in the diagnosis of COVID-19. The dataset has been processed using balance, augmentation, shuffling, and prefetching techniques described in the methods section. Several neural networks were tested on the datasets using the transfer learning technique. The Google Colab platform was used to run the Python3 code in the form of Jupyter notebooks. The results are explained and discussed in the next sections.

## 5.2 Details of the Dataset Used for Three Stages

The dataset mentioned in Section 4.2 was used in this experiment, and we divided it into 3 stages to test the performance using different class predictions. The stages are as follows:

### 5.2.1 Stage 1: Binary Classification Normal and Covid

- The original dataset had 27052 images divided into 4 classes: 12157 for Normal, 5068 for Pneumonia, 6012 for Lung Opacity, and 3815 for Covid. After the balancing process, the dataset was divided into two classes: Normal and Covid. The images had a size of 280*280 and a ".jpg" format. A relation of 70%-15%-15% was used to divide the training, validation, and test datasets, respectively. A summary of the structure of the dataset used in binary classification is presented in Table 5.1.

| Class | COVID-19 | Normal | **Total** |
|---|---|---|---|
| Train | 2670 | 2670 | 5340 |
| Validation | 572 | 572 | 1144 |
| Test | 573 | 573 | 1146 |
| **Total** | 3815 | 3815 | 7630 |

Table 5.1: Binary classification arrangement of datasets.

- In Figure 5.1, we can observe several random images obtained from the balanced dataset.

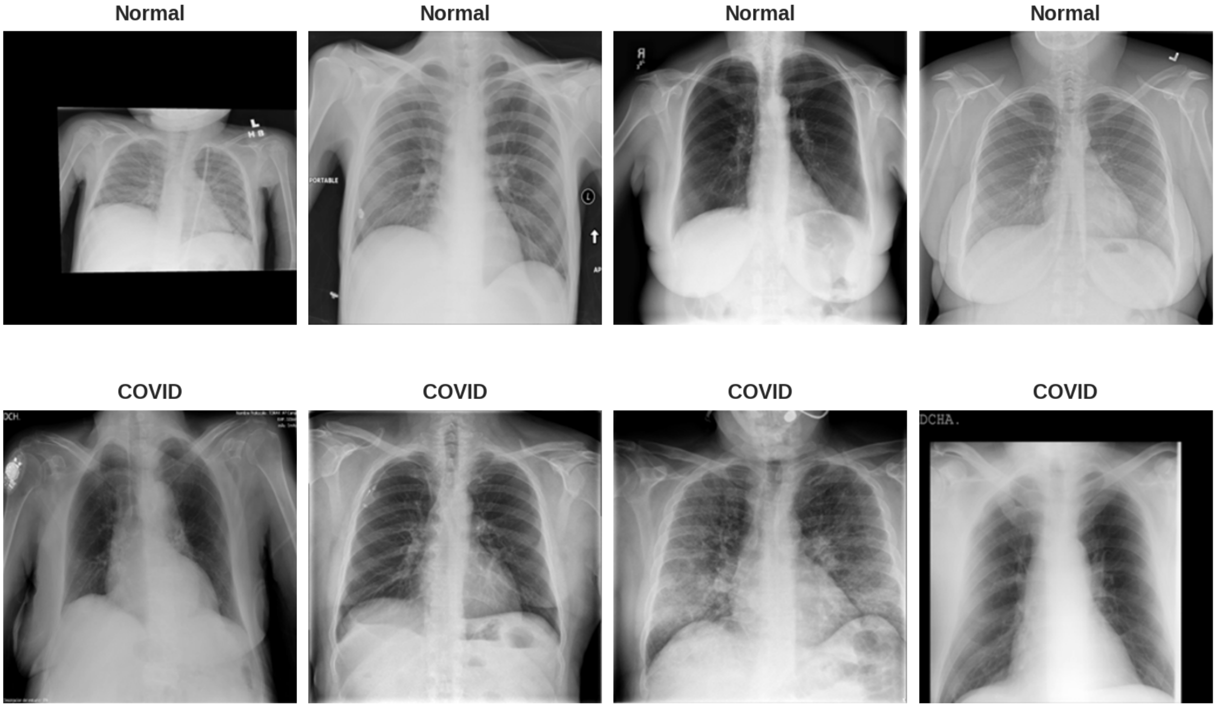


Figure 5.1: Images extracted from the processed and balanced dataset indicating the class for each sample.

### 5.2.2 Stage 2: 3-Class Classification Normal, Covid, and Viral Pneumonia

- A summary of the dataset used in three-class classification is seen in Table 5.2.

| Class | COVID-19 | Normal | Viral pneumonia | **Total** |
|---|---|---|---|---|
| Train | 2670 | 2670 | 2670 | 8010 |
| Validation | 572 | 572 | 572 | 1716 |
| Test | 573 | 573 | 573 | 1719 |
| **Total** | 3815 | 3815 | 3815 | 11445 |

Table 5.2: Three-class classification arrangement of dataset.

- In Figure 5.2, random samples after processing for the 3 classes are shown.

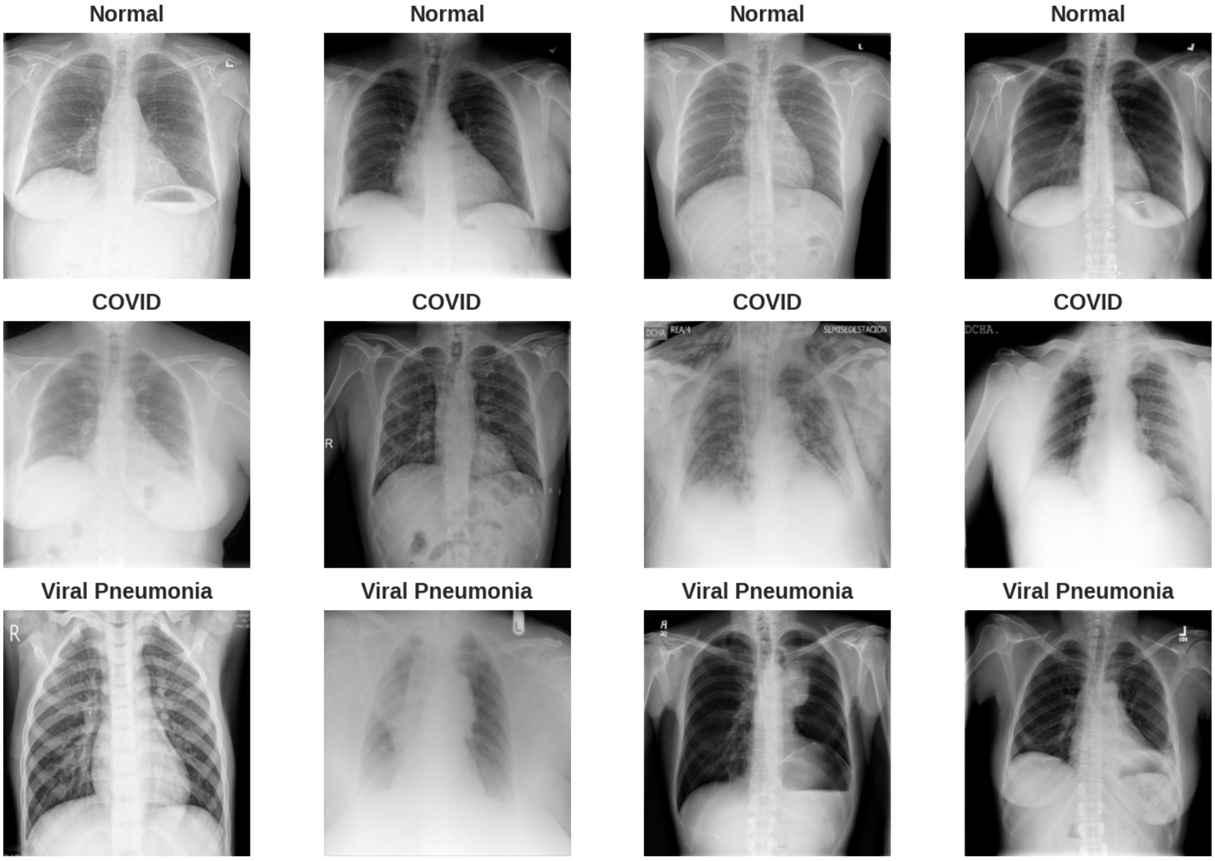


Figure 5.2: Samples obtained after processing and balancing the COVID dataset

### 5.2.3 Stage 3: 4-Class Classification Normal, Covid, Viral Pneumonia, and Lung-Opacity

- In Table 5.3, we have the structure of the dataset used for this stage of classification.

| Class | COVID-19 | Normal | Viral pneumonia | Lung-Opacity | **Total** |
|---|---|---|---|---|---|
| Train | 2670 | 2670 | 2670 | 2670 | 10680 |
| Validation | 572 | 572 | 572 | 572 | 2288 |
| Test | 573 | 573 | 573 | 573 | 2292 |
| **Total** | 3815 | 3815 | 3815 | 3815 | 15260 |

Table 5.3: Four-class classification arrangement of dataset.

- In Figure 5.3, we can appreciate images from all the 4 classes that were used in this work.

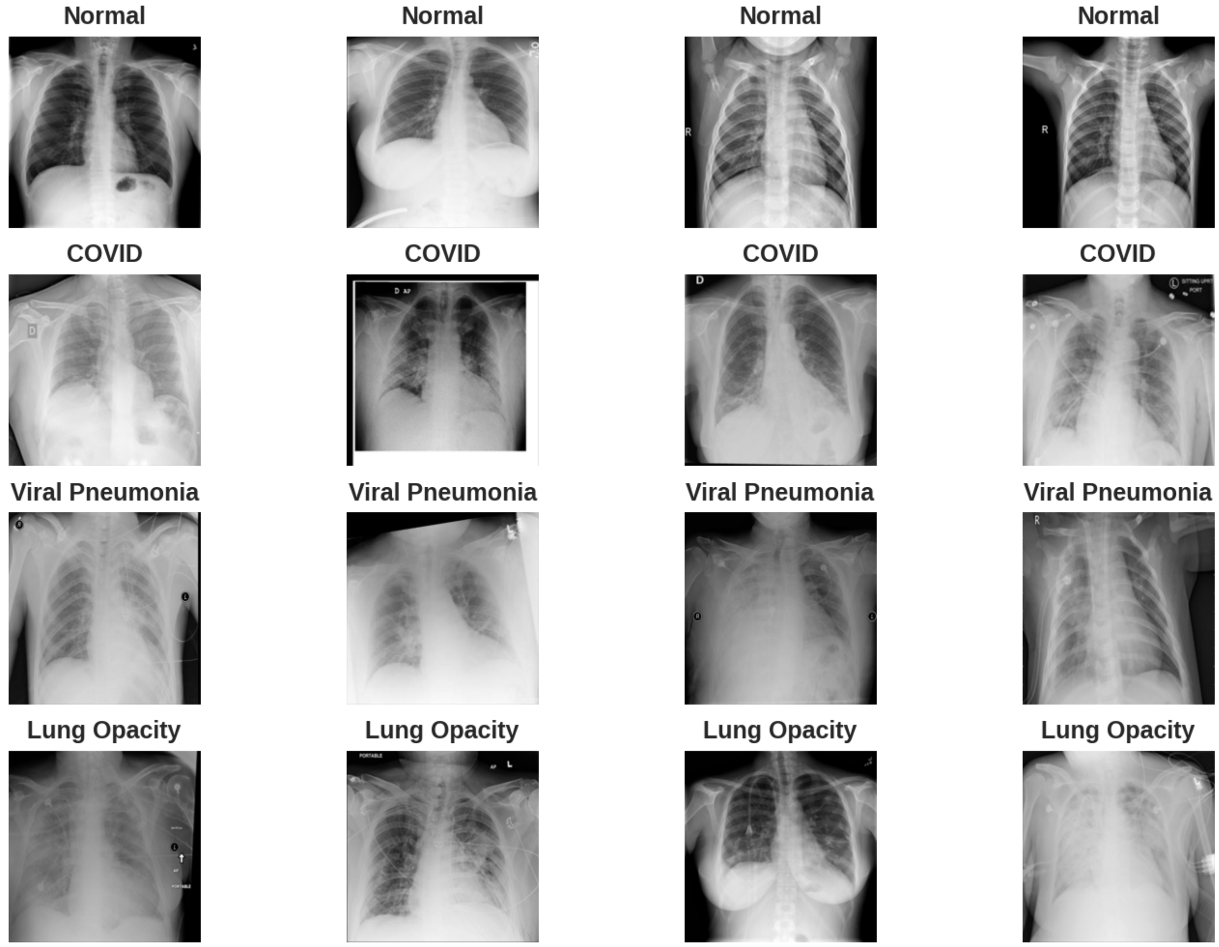


Figure 5.3: Samples from the balanced dataset showing all the classes used in the experiment

## 5.3 Transfer Learning Models Architectures

All the transfer learning models have the same general structure: input, sequential, rescaling, functional, pooling, dropout, and dense layers. The functional layer is where the transfer learning model is located. In all the experiments, the dense layer was set to ”Trainable,” and the functional layer was frozen during the first process of training. The number of parameters is the same for the freezing and the fine-tuning training. Only the functional and dense layers account for parameters. The details of these layers are reported in Table 5.4.

| Model Architecture | Trainable | Non-trainable | **Total** |
|---|---|---|---|
| Xception | 2,049 | 20,861,480 | **20,863,529** |
| VGG16 | 513 | 14,714,688 | **14,715,201** |
| VGG19 | 513 | 20,024,384 | **20,024,897** |

Table 5.4: Transfer learning training parameters. The non-trainable parameters correspond to the functional layer during the freezing process. The functional layer contains the model obtained from the Keras module.

## 5.4 Training: Convergence Analysis

### 5.4.1 Stage 1: Binary Classification Xception, VGG16 and VGG19

**Xception Model**

- Convergence analysis: In Figure 5.4, we can observe the plots produced by our Xception model with an analysis during the freezing and fine-tuning stages of training.
- The freezing stage took 20 minutes and 35 seconds to compile using the free T4 GPU over 20 epochs. It is important to mention that both the RAM and GPU memory were almost at their limit, which means that adding more images or using a larger image size will collapse the compilation due to a lack of resources. Resizing and other methods were used to reduce the use of memory. Reducing the size of images could make them lose important information, making the model have poor accuracy. So, if we want to use larger datasets, a paid version of Google Colab is suggested, which offers more memory for both GPU and RAM.
- The fine-tuning stage took 28 minutes and 2 seconds to compile using the same software and hardware as in the freezing stage over 10 epochs. The compiling time is higher in this stage compared to freezing since more layers are being trained.

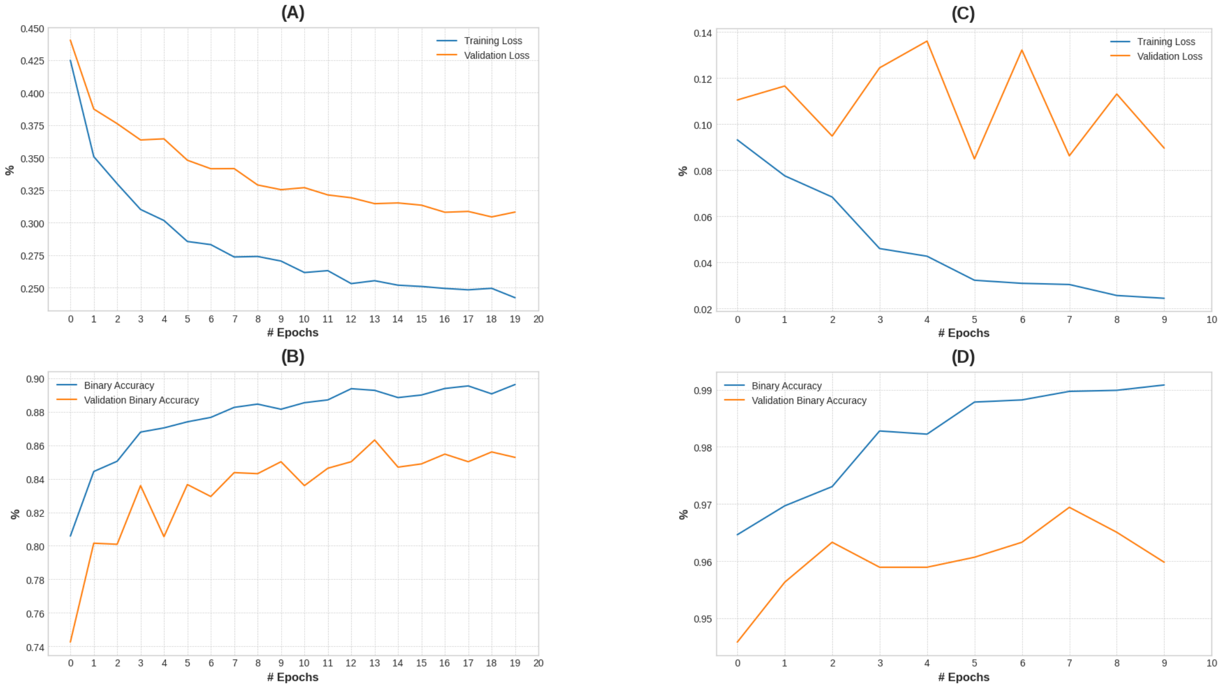


Figure 5.4: Convergence plots for the Xception neural network. **(A) Training loss convergence during the freezing process:** We can appreciate a reduction in the % of both the training and validation losses, which might indicate that the model is capturing the desired patterns and might have good prediction results. **(B) Training accuracy during freezing:** We appreciate that the binary accuracy is around 88% at the end of the training. The validation accuracy is around 80%, which is a good value. **(C) Convergence loss during fine-tuning:** The training loss converges smoothly in contrast to the validation loss, which has high and low-value peaks. **(D) Fine-tuning accuracy convergence:** We appreciate excellent binary accuracy after training the whole neural network with the dataset. Also, we have very good validation accuracy, which might indicate that the model will perform very well when tested with images that the model has not seen.

### VGG16

- Convergence analysis: In Figure 5.5, we can observe the graphs produced by our VGG16 model.
- The freezing stage took 20 minutes and 49 seconds to compile using the T4 GPU over 20 epochs.
- The fine-tuning stage took 22 minutes and 35 seconds to compile over 10 epochs. In general, VGG16 took similar compiling times for both freezing and fine-tuning.

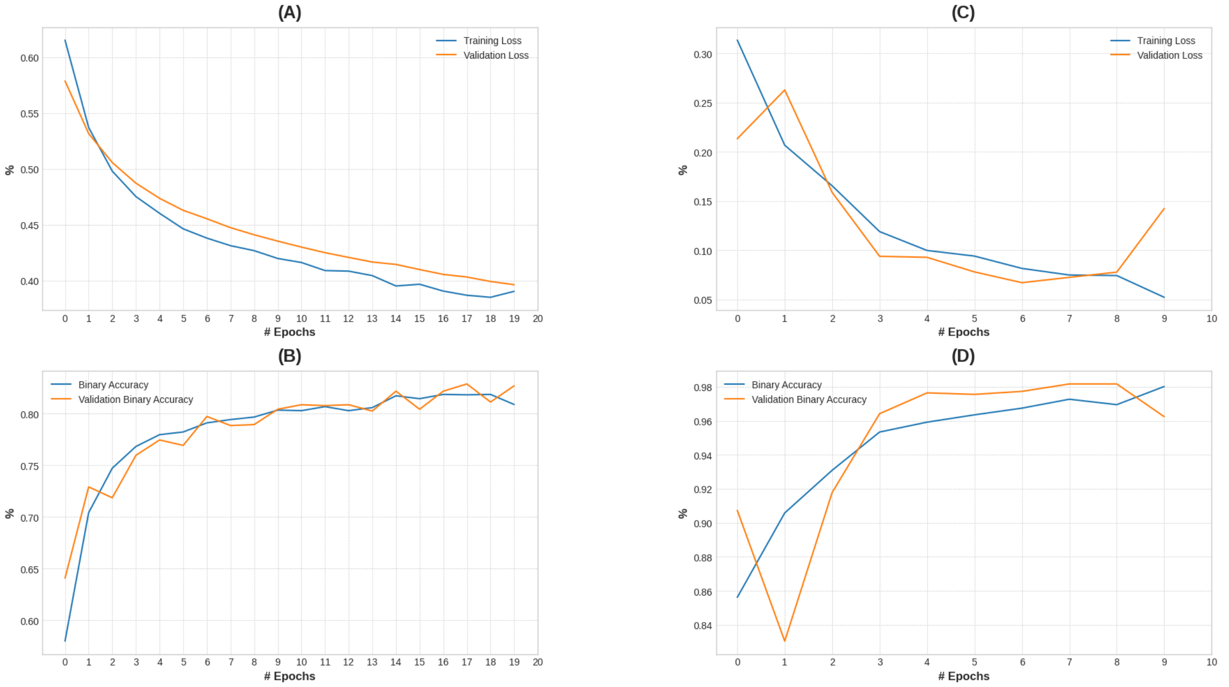


Figure 5.5: Plots VGG16 model. **(A) Loss during the freezing process:** We can appreciate a very smooth graph without noticeable peaks. **(B) Accuracy during freezing:** We appreciate that the binary accuracy is around 80% at the end of the training. Also, the validation accuracy curve is pretty close to binary accuracy in comparison to Figure 5.4:(B), which means that the model is better than Xception in validating the images with the captured patterns during training. Also, the binary accuracy curve has fewer peaks than Xception. **(C) Loss during fine-tuning:** The training loss curve converges very well, but the validation loss has a peak at epoch #1 and a peak at epoch #9. This could be solved by increasing the number of epochs for both freezing and fine-tuning. We would recommend at least 50 epochs for freezing and 20 for fine-tuning. In my actual hardware setup, this would collapse my RAM and GPU memory, so we did not try that solution. **(D) Accuracy fine-tuning:** The binary accuracy curve converges without peaks, but the validation curve has a minimal value at epoch #1 and then converges smoothly. In comparison with Figure 5.4:(C), and (D), both loss and accuracy during fine-tuning get better results than in the Xception model because curves are closer to each other in VGG16. Although the values for binary accuracy are slightly lower than Xception, the closeness of the curves might indicate that VGG16 might perform better than Xception in a new dataset.

### VGG19

- Convergence analysis: In Figure 5.6, plots for loss and accuracy for the VGG19 model.
- The freezing process took 24 minutes and 12 seconds to compile using the free T4 GPU over 20 epochs.

- The fine-tuning stage took 26 minutes and 3 seconds to compile. Xception, VGG16, and VGG19 have very similar values for compiling time during training. So, there is no better model in terms of compiling time. Other parameters like accuracy and F1 score might tell us which model is the better model.

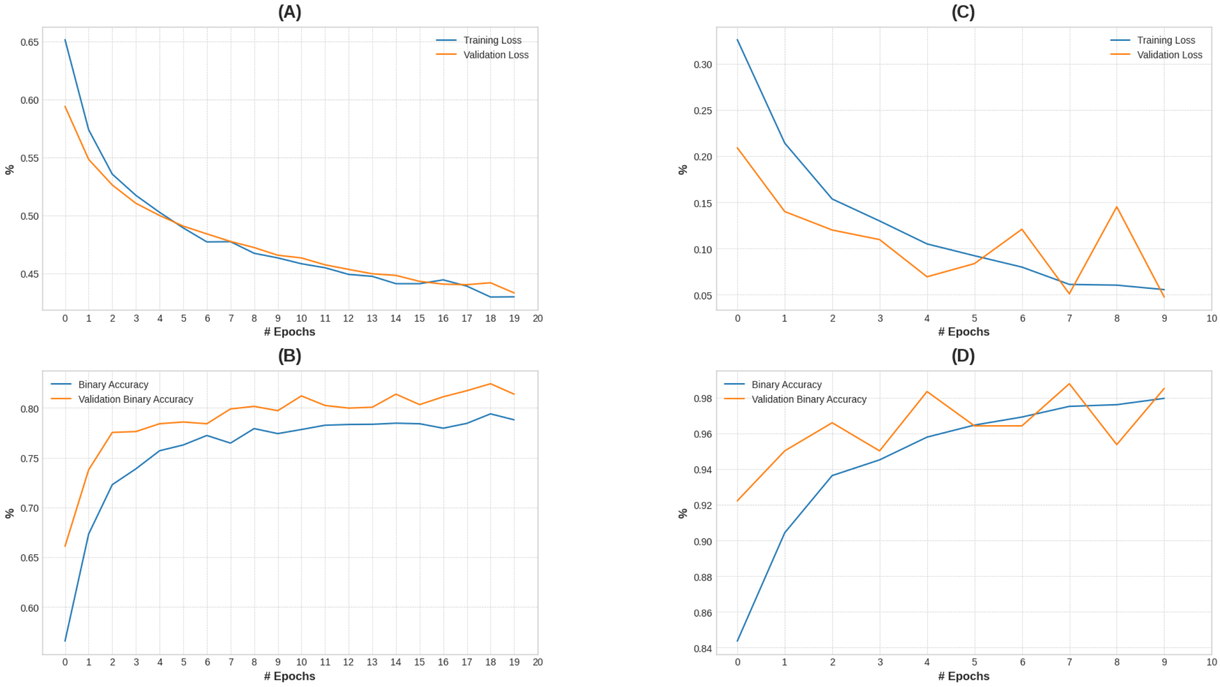


Figure 5.6: VGG19 convergence plots. **(A) Loss during the freezing process:** Both training and validation curves converge smoothly with no significant peaks or minimal values. **(B) Accuracy during freezing:** There is a little separation of the binary accuracy and validation curves in comparison to VGG16 curves. **(C) Loss during fine-tuning:** There are two significant peaks at 6 and 7 epochs. **(D) Accuracy fine-tuning:** The validation accuracy has several peaks during the training. Again more epochs are recommended.

### 5.4.2 Stage 2: 3-Class Classification VGG19

In Figure 5.7, we can observe the plots produced by the VGG19 model. The freezing training took 27 minutes and 43 seconds to compile using T4 GPU over 20 epochs. The fine-tuning stage took 38 minutes and 18 seconds to compile over 10 epochs.

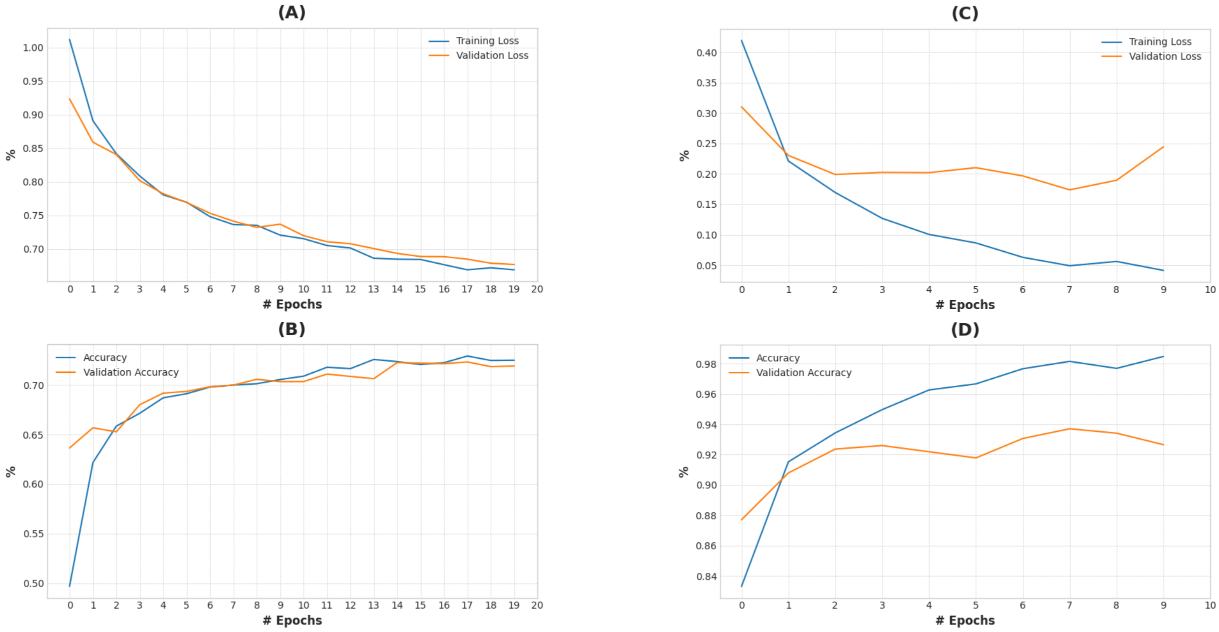


Figure 5.7: Convergence plots for the VGG19 model using three classes: Normal, Covid, and Viral Pneumonia. **(A) Training loss convergence during the freezing process:** We observe that both the training loss and validation loss curves are very similar and do not have significant peaks. **(B) Training accuracy during freezing:** We can appreciate that the convergence is around 70%, and the curves are also close to each other. **(C) Convergence loss during fine-tuning:** In this stage we have reduced the loss significantly compared to the training with freezing. **(D) Fine-tuning accuracy convergence:** Here, we have an excellent value for the accuracy at the end of the tenth epoch but the validation accuracy is not converging close to the training accuracy. This might be caused by the addition of the new Viral Pneumonia class.

### 5.4.3 Stage 3: 4-Class Classification VGG19

In Figure 5.8, we observe the plots produced by the VGG19 model. The freezing training took 32 minutes and 34 seconds to compile using T4 GPU over 20 epochs. The fine-tuning stage took 46 minutes and 16 seconds to compile over 10 epochs.

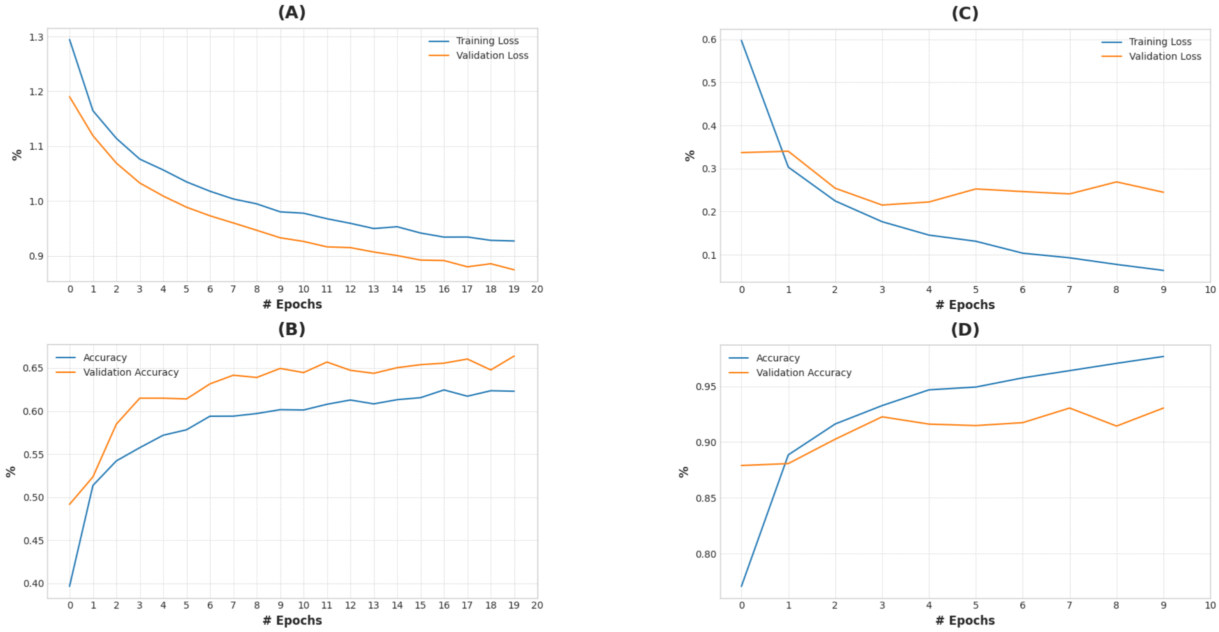


Figure 5.8: Convergence plots for the VGG19 model using all the classes: Normal, Covid, Viral Pneumonia, and Lung Opacity. **(A) Training loss convergence during the freezing process:** Both curves converge smoothly because they never overlap each other. **(B) Training accuracy during freezing:** In this case, the curves present some peaks and again they never overlap, this might indicate that we are going to have more miss-classified images than in the binary and the 3-class classification. **(C) Convergence loss during fine-tuning:** In this stage we have more loss compared to the 3-class and the binary, the curves converge with almost no peaks. **(D) Fine-tuning accuracy convergence:** Finally, the accuracy is above 90% during this stage, which indicates that our model still is very good at classifying the images with the last Lung Opacity class.

## 5.5 Stage 1: Binary Classification Performance metrics

### The Xception Neural Network

Test set performance metrics: In Figure 5.9, we observe the confusion matrix for binary classification for the Xception after being tested in an unseen dataset. The model model correctly identifies 561 cases as COVID-positive, which are True Positives. Also, 40 cases are COVID-positive but the model fails to identify them and predicts them as normal, so these are False Negatives. Moreover, 12 cases are normal but the model incorrectly predicts them as COVID-positive, then these are False Positives. Finally, 533 cases are normal, and the model correctly predicts them as normal, then these are True Negatives. High values for accuracy, sensitivity, specificity, and precision. A higher F1 score indicates a better balance between precision and sensitivity. A high F1 score was obtained, generally indicating good model performance.

The model has high accuracy and is performing well in terms of both precision and sensitivity. A high precision indicates that the model has a low rate of falsely predicting normal cases as COVID-positive. Also, a high sensitivity indicates that the model is effective in capturing most of the COVID-positive cases. A high specificity means that the model is effective in minimizing false positives. The miss-classification rate is relatively low, indicating a good balance between false positives and false negatives.

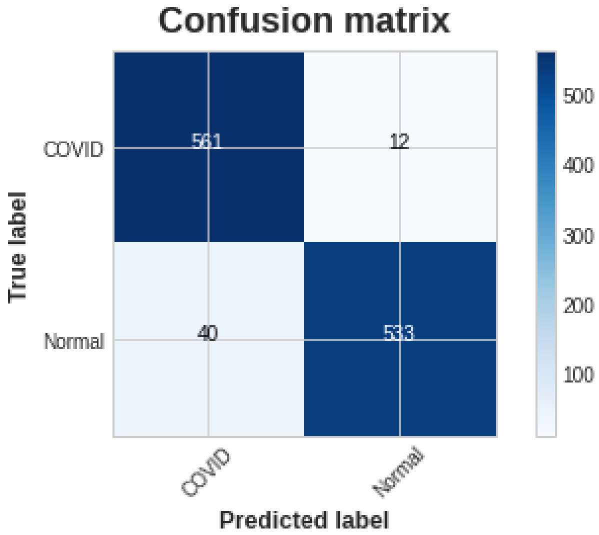


Figure 5.9: In the plot we can observe that Xception has an excellent predictive capability. True Positives: 561, True Negatives: 533, False Positives: 12. False Negatives: 40. Accuracy: 95.46%. Miss-Classification: 4.54%. Sensitivity: 93.34%. Specificity: 97.8%. Precision: 97.91%. F1 Score: 95.57%.

### Results of the VGG16

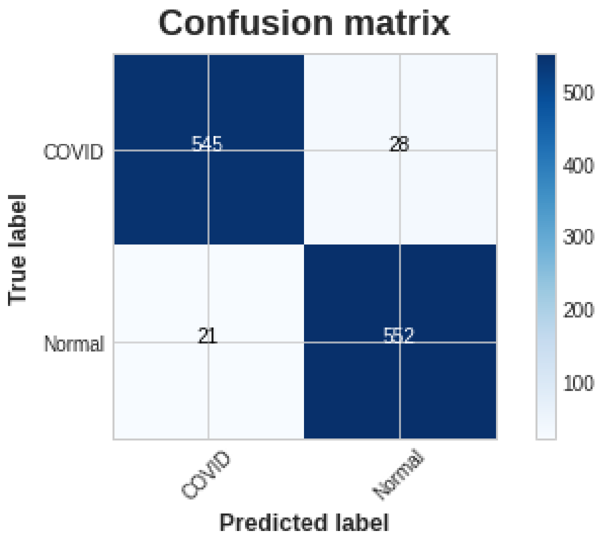


Figure 5.10: In the plot, we can observe that VGG16 also has excellent predictive capability. True Positives: 545, True Negatives: 552, False Positives: 28. False Negatives: 21. Accuracy: 95.72%. Miss-Classification: 4.28%. Sensitivity: 96.29%. Specificity: 95.17%. Precision: 95.11%. F1 Score: 95.7%.

### Neural Network VGG19

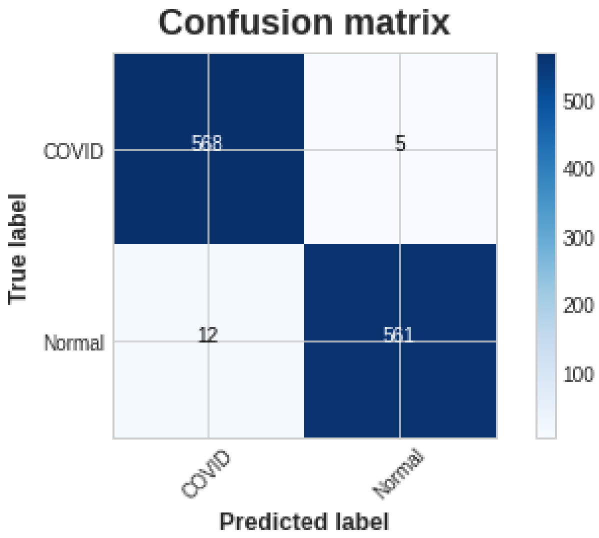


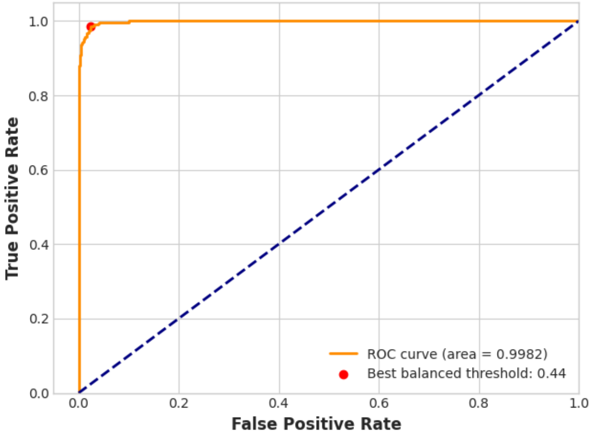
Figure 5.11: VGG19 has excellent predictive capability and is slightly better than Xception and VGG16. True Positives: 568, True Negatives: 561, False Positives: 5. False Negatives: 12. Accuracy: 98.52%. Miss-Classification: 1.48%. Sensitivity: 97.93%. Specificity: 99.12%. Precision: 99.13%. F1 Score: 98.53%.

### 5.5.1 Binary Classification Summary

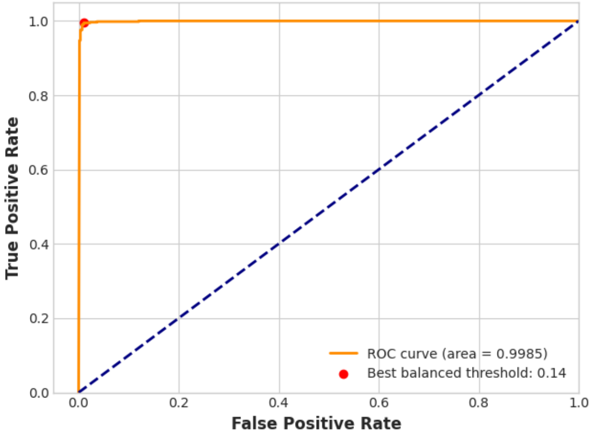
A summary Table 5.5 from Figure 5.9, Figure 5.10, and Figure 5.11 is presented to select the best model to be tested on 3 class and 4 class stages. The VGG19 model has the highest accuracy, sensitivity, specificity, precision, and F1 Score. For the sensitivity, it means that the model is better than the others in predicting patients with COVID-19 and has a minimal amount of mispredicted false negatives. Also, for specificity, it means that VGG19 is excellent in predicting patients that do not have the disease and there is a small amount of false positives. In general, for COVID-19 a test that does not miss the true positives would be chosen to avoid the spread of the disease. The F1 Score takes into account both sensitivity and specificity. VGG19 has an F1 Score of 98.53 % so it is the best model to be tested in the 3 classes and 4 classes classification.

| Model | Accuracy [%] | Sensitivity [%] | Specificity [%] | Precision[%] | F1 Score [%] |
|---|---|---|---|---|---|
| Xception | 95.46 | 93.34 | 97.8 | 97.91 | 95.57 |
| VGG16 | 95.72 | 96.29 | 95.17 | 95.11 | 95.7 |
| VGG19 | 98.52 | 97.93 | 99.12 | 99.13 | **98.53** |

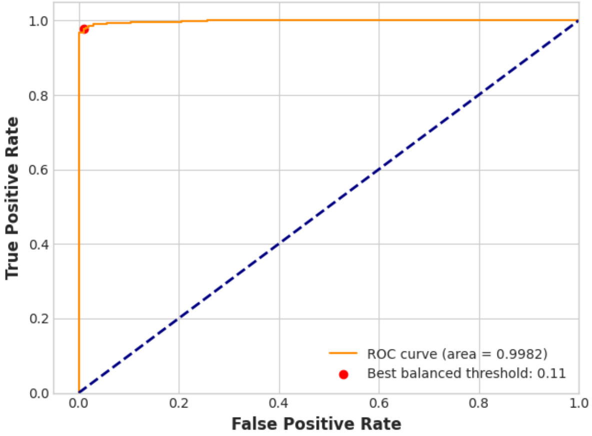
Table 5.5: Metrics summary for the binary classification using the transfer learning models and COVID and Normal classes.



(a) Xception ROC.



(b) VGG16 ROC.



(c) VGG19 ROC.

Figure 5.12: ROC graphs of the models used during binary classification.

Xception (a), VGG16 (b), and VGG19 (c) models are shown in Figure 5.12. We observe that the AUC are very similar, indicating excellent discrimination, meaning the model excellently distinguishes between the positive and negative COVID-19 classes. We used

45 threshold values for all three graphs: min value: -15.115046 and max value: 13.334197 with an average step: 0.6322. The best-balanced threshold indicates the point where the model will effectively classify the cases. We used Youden's J statistic to determine the best-balanced threshold. The VGG16 has the best graph which might indicate a slightly better performance than the other models. However, using only AUC to determine the best model is not recommended. In our particular experiment, detecting patients positive for COVID-19 is the priority. So, the model that has the highest sensitivity is VGG19. Then we chose VGG19 as the best model to be trained with more classes.

### 5.5.2 Stage 2: 3-Class Classification VGG19

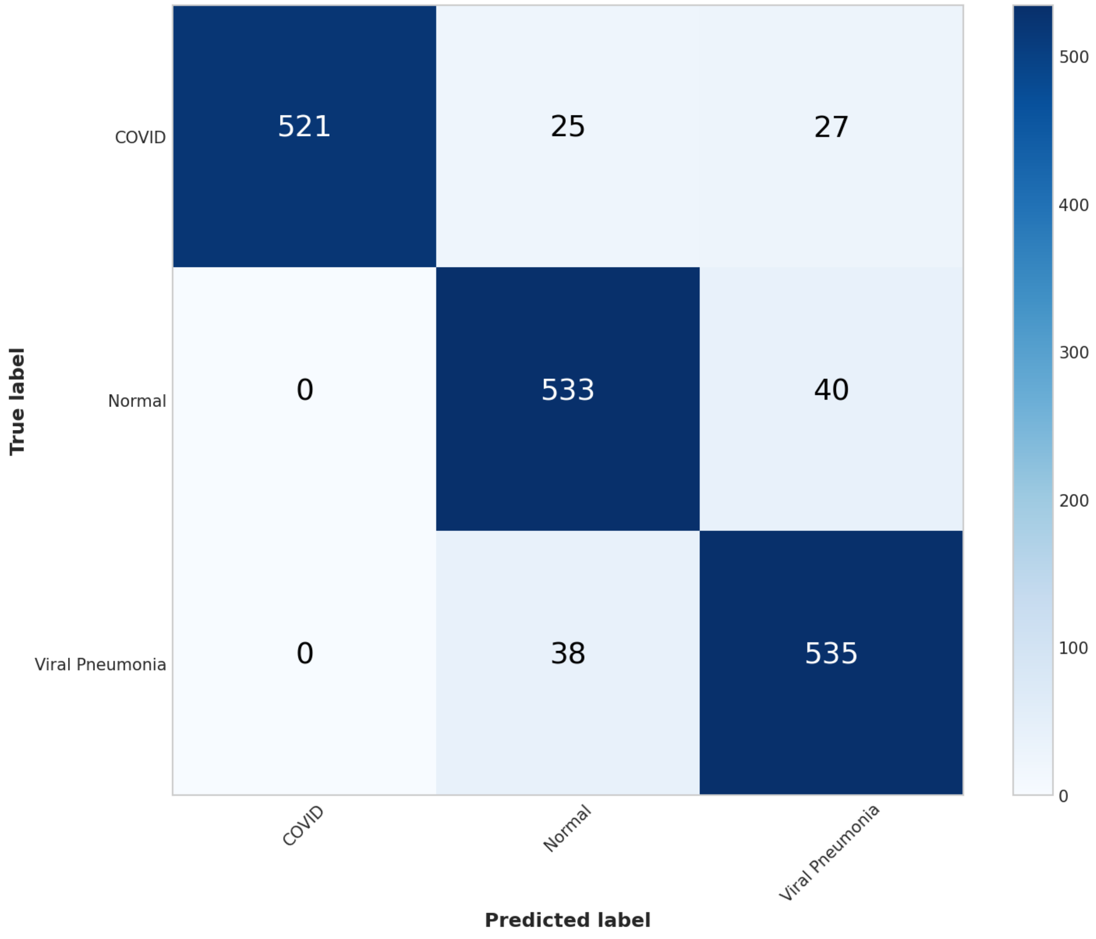
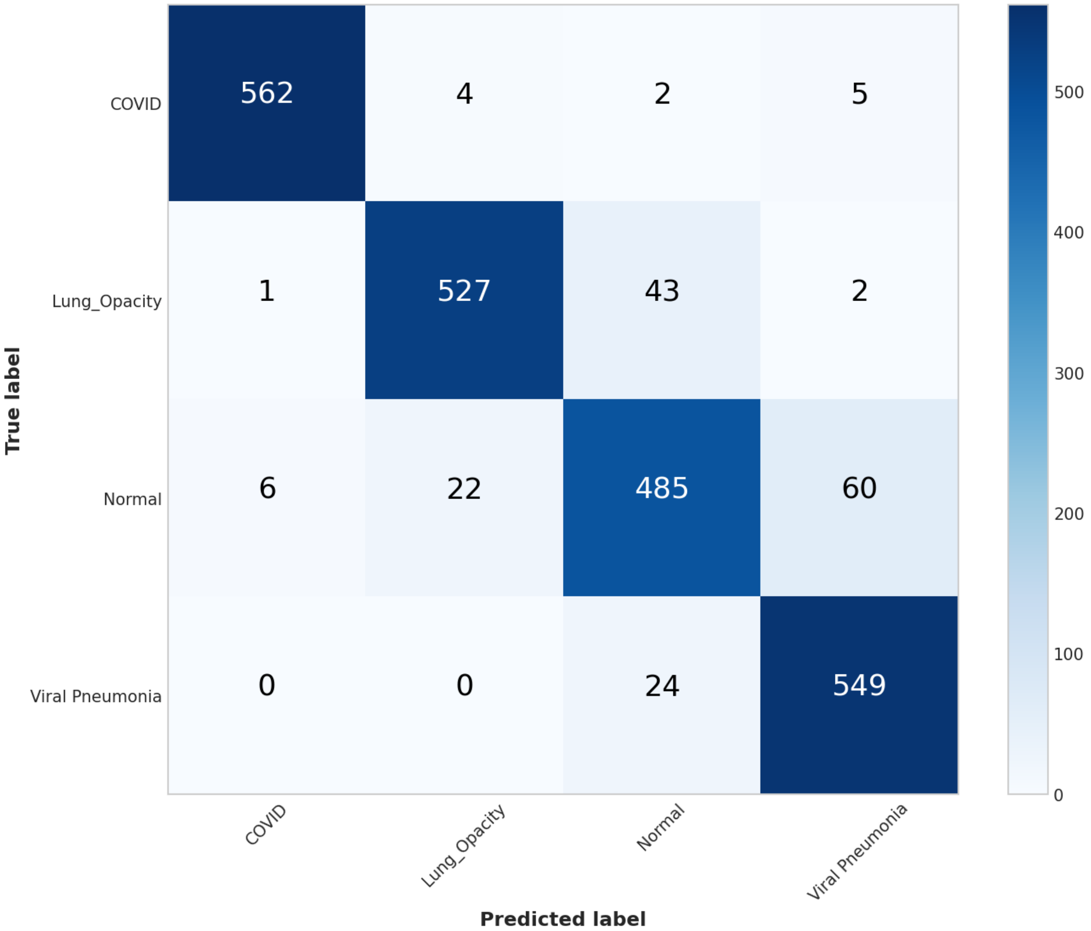


Figure 5.13: VGG19 confusion matrix on Normal, Covid, and Viral Pneumonia classes. Accuracy: 92.44%. Miss-Classification: 7.56%. Weighted average F1 Score: 92.32%.

### 5.5.3 Stage 3: 4-Class Classification for VGG19



| True label \ Predicted label | COVID | Lung_Opacity | Normal | Viral Pneumonia |
|---|---|---|---|---|
| COVID | 562 | 4 | 2 | 5 |
| Lung_Opacity | 1 | 527 | 43 | 2 |
| Normal | 6 | 22 | 485 | 60 |
| Viral Pneumonia | 0 | 0 | 24 | 549 |

Figure 5.14: VGG19 confusion matrix on Normal, Covid, Viral pneumonia, and Lung Opacity classes. Accuracy: 92.63%. Miss-Classification: 7.37%. Weighted average F1 Score: 89.04%.

## 5.6 Grad-CAM on Our VGG19 Model

In Figure 5.15, we observe the Grad-Cam that was obtained using the layer last convolutional layer of our trained VGG19 model named $block5_conv4$. We can appreciate the zones that our model is capturing to predict the class of the image. The red-colored regions are considered highly relevant. Regions highlighted in yellow may still be relevant but to a lesser extent compared to the red regions. Based on the color interpretations, we observe that the left lung has a lot of regions that are considered important. Also, the heart shape is considered as seen in the yellow highlights. These results confirm the annotations mentioned in the state of the art about the reticular interstitial patterns and the appearance of white areas in the lungs called opacities which are areas filled with fluid.

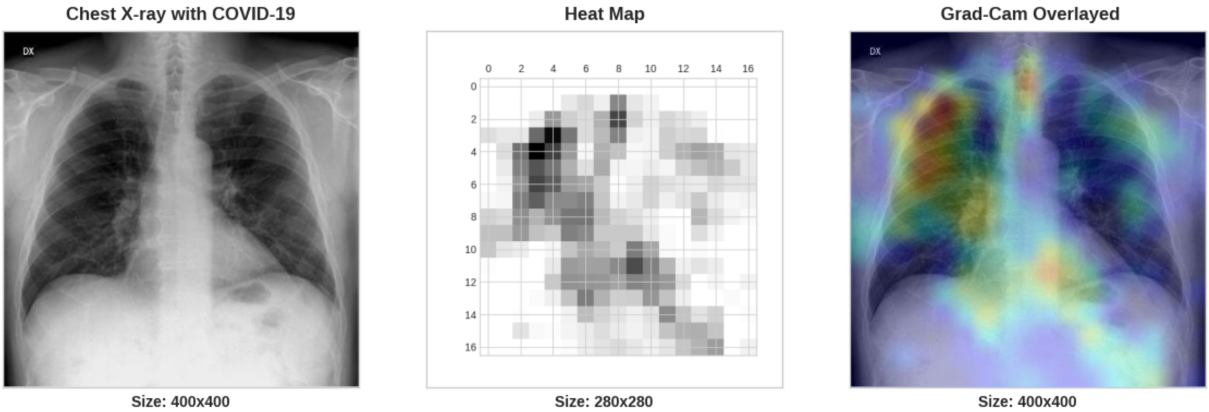


Figure 5.15: Chest X-ray of a patient with COVID-19. The first image is extracted randomly from the dataset used in this work. The X-ray image was predicted positive for COVID-19 using the trained model. Next, the Heat Map was obtained using the gradients calculated from the predictions of the model. Then, the Grad-Cam was obtained and overlayed.

In Figure 5.16, we observe peripheral bilateral opacities highlighted in red which are very common in COVID-19 patients. So, our trained model is performing excellent in recognizing the patterns of this disease.

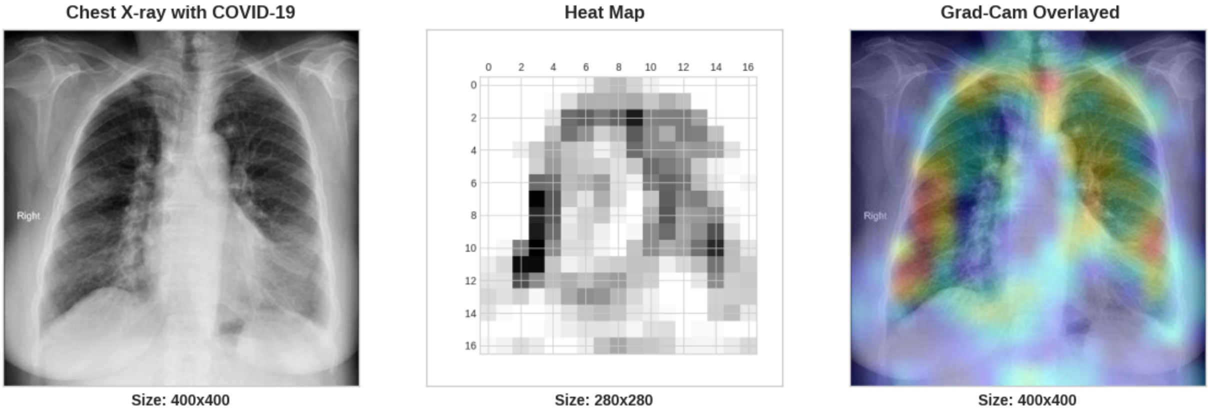


Figure 5.16: This image is another example of the Grad-Cam on a patient with COVID-19.

From Figure 5.17, we can say that the red zones indicate the zones that present opacities. In this case, both lungs present a general opacity. We also appreciate the red zones at the bottom of the image. This tells us that the model might be learning patterns that are not necessary to predict a label. This could be solved by segmenting the images and creating a new image that puts in black all the zones out of the lungs. So, the model will not learn unnecessary structures.

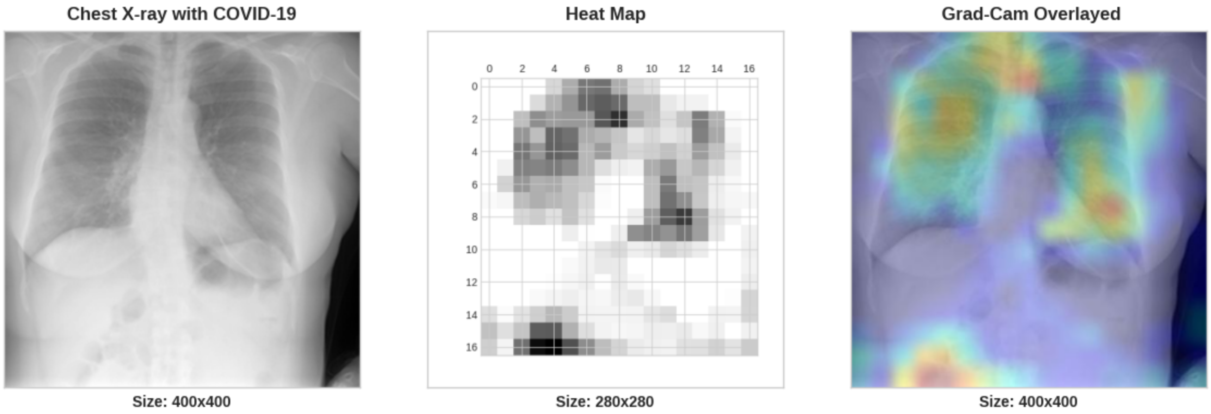


Figure 5.17: Female Chest X-ray positive for COVID-19 extracted from the dataset.

The Grad-Cam applied in healthy patients shows less pattern recognition compared to COVID-19 Figure 5.18. The red areas are also recognizing unnecessary structure out of the lungs.

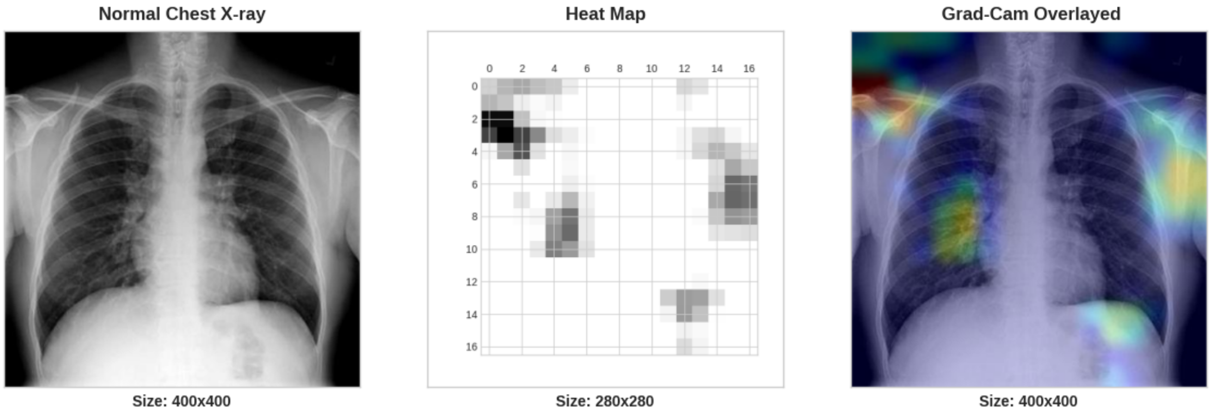


Figure 5.18: Chest X-ray of a Healthy Patient. Image extracted from our processed dataset.

We suggest using segmentation and retraining the neural network with the segmented images. Also, the classes Lung Opacity and Viral Pneumonia could be added to COVID-19 to improve even more the prediction of the model. We showed that the VGG19 model is capable of inferring with high accuracy whether a patient has or not COVID-19. The zones in red also give us insights into what patterns are capturing our model during training allowing us to improve the performance and saving computational resources.

## 5.7 VGG19 Heat Maps

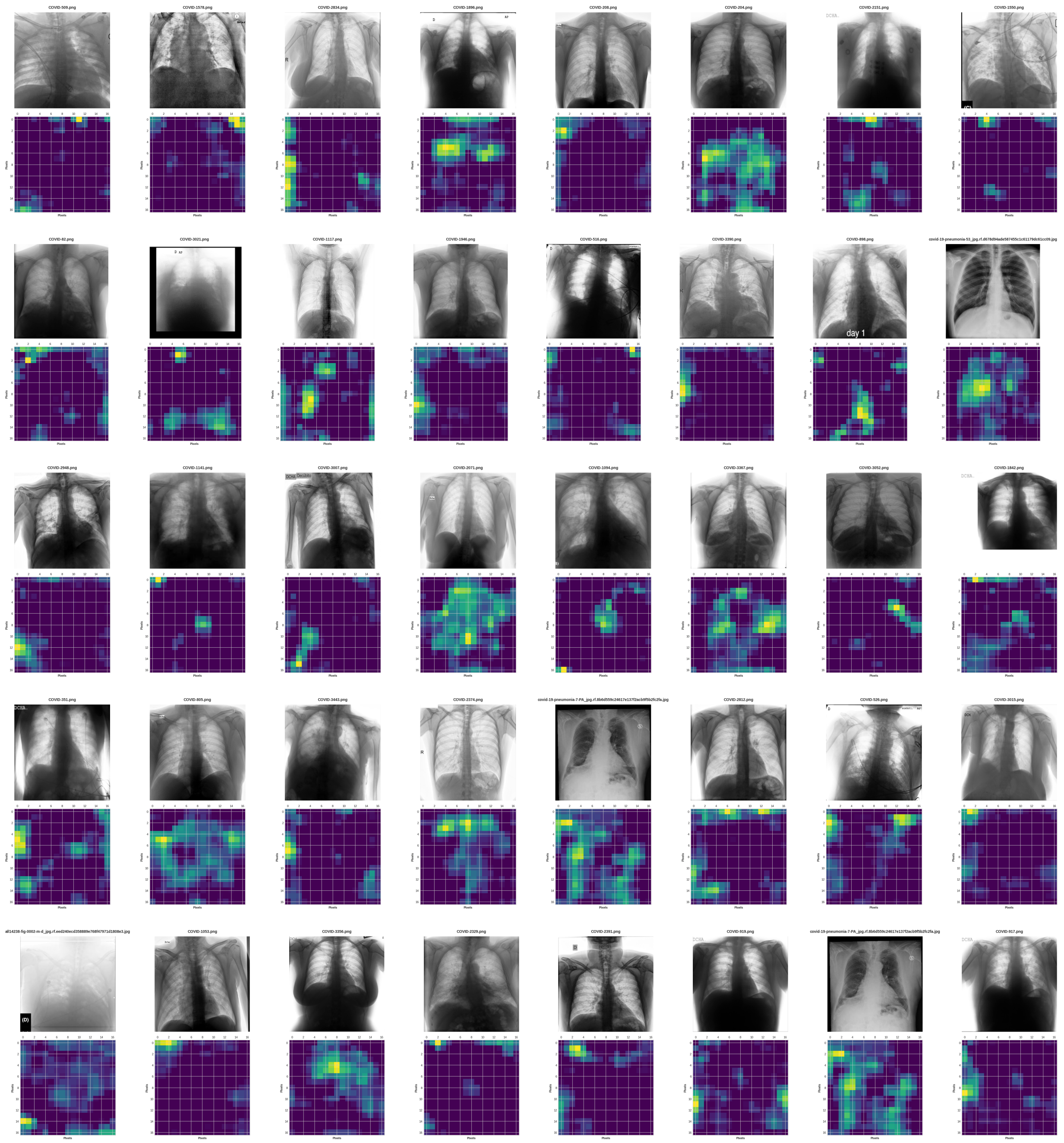


Figure 5.19: Covid-19 heat maps obtained from the VGG19 Model. The heat maps were obtained using the trained model on our processed dataset. The last convolutional layer was used to create the heat maps. The areas in yellow are patterns captured by the neural network.

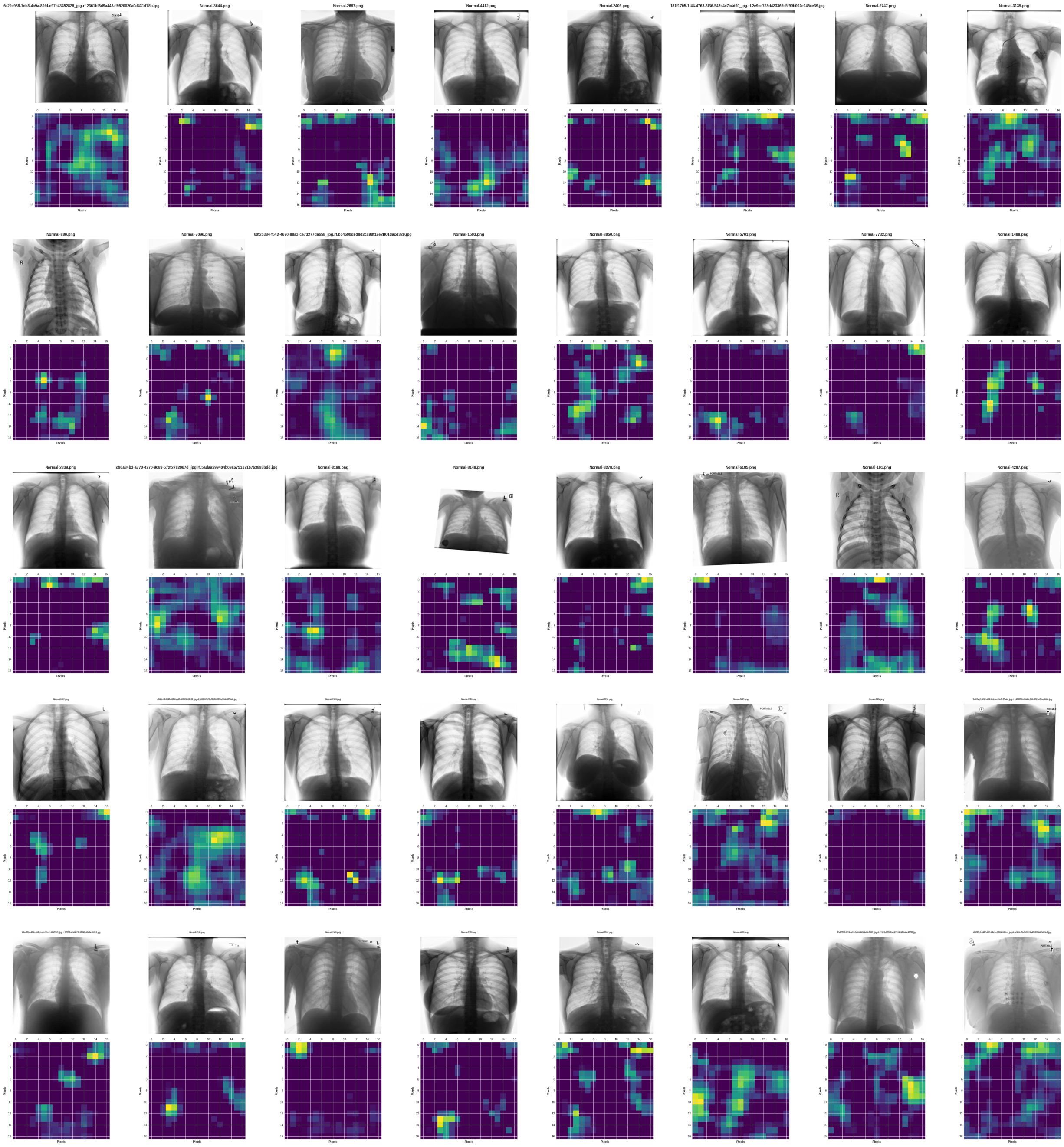


Figure 5.20: Healthy patient's heat maps obtained from the VGG19 model.

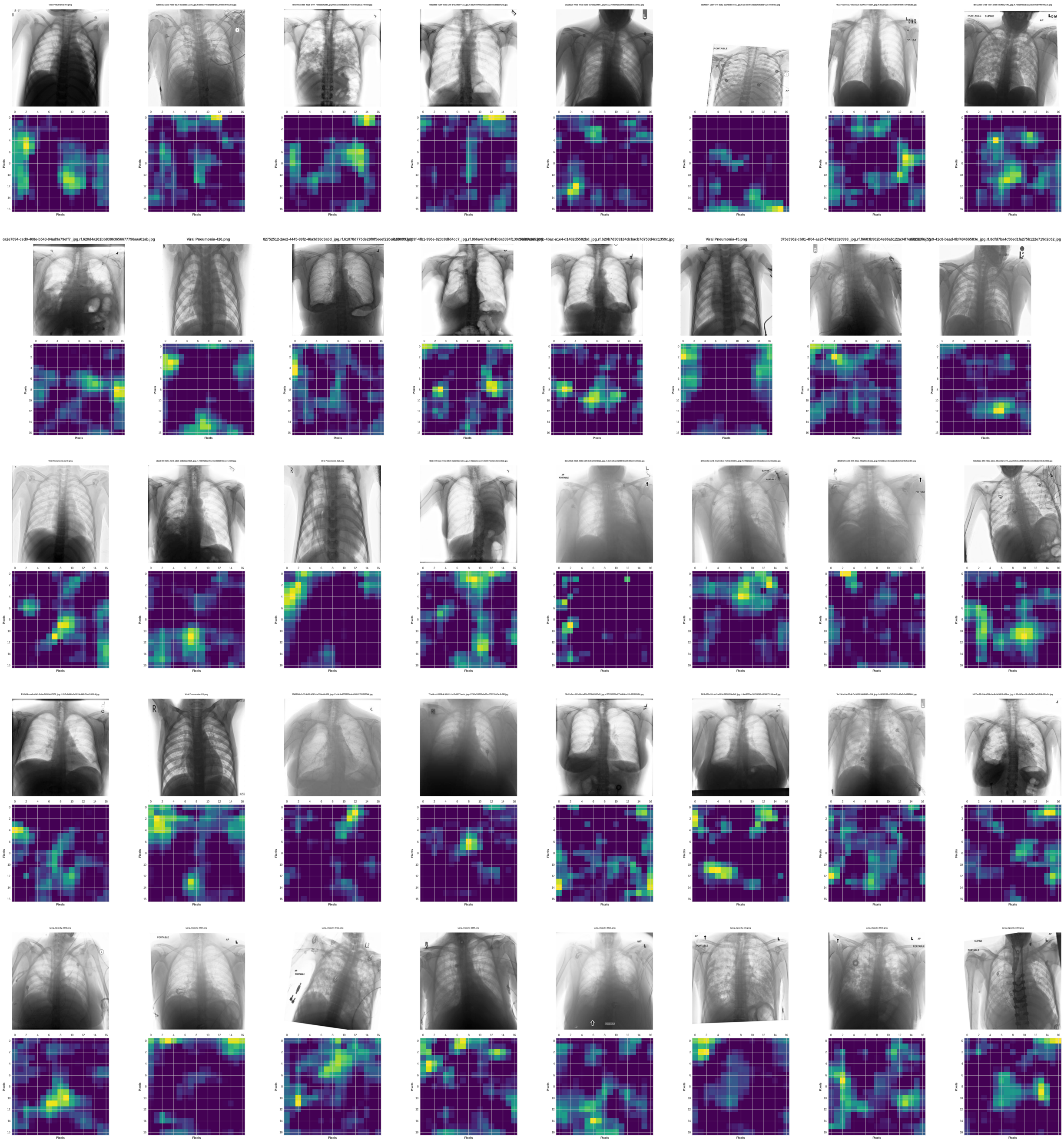


Figure 5.21: Viral Pneumonia class heat maps VGG19 model.

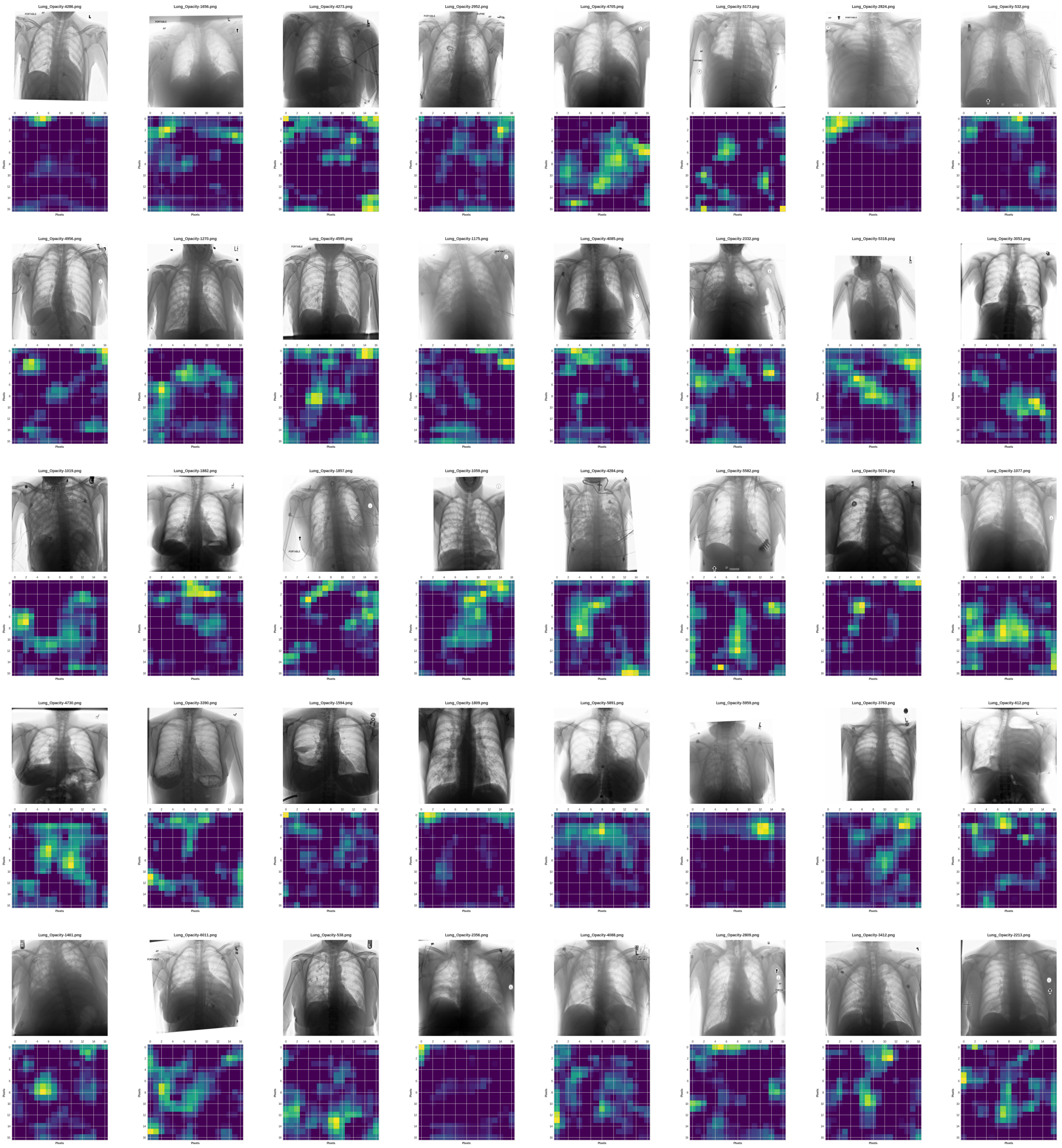


Figure 5.22: Lung Opacity heat maps VGG19 model.

From Figures 5.19, 5.20, 5.21, 5.22, we observe heat maps that were extracted using VGG19 model used in this work. The code we used to process the images and extract the last convolutional layer of the model was compiled in Google Colab. All four classes were tested in the model producing the heat maps that give us an idea of the patterns the model considers important. We observe that some of the captured patterns are around the lungs. Also, the shape of the lung is considered in COVID-19 class. The viral Pneumonia patterns are very similar to the Covid-19. So, we recommend considering merging these classes. Finally, healthy chest X-rays do not present a high amount of yellow areas meaning that the model is not capturing structures that are unique in sick patients.

# Chapter 6

# Discussion

## 6.1 VI. Discussion of Results

In this section, a comparative analysis is made with other authors that have used similar parameters and transfer learning with the models we have used.

### 6.1.1 Results of the Xception Neural Network

Shazia and its team used transfer learning and obtained 98.34% accuracy 98.49% F1-Score for binary classification [137]. They used 1536 COVID-19 and 5629 pneumonia images with a resize of 224×224. They report to have balanced the dataset but do not report the images used after the processing. Also, they claimed to have randomly selected 10% of the dataset images as testing. Also, they said to have used 20 epochs when they noticed no improvement. In comparison, we obtained a binary accuracy of 95.46% and F1-score of 95.57% with 20 epochs in freezing and 10 epochs for fine tuning with a total of 7630 images divided equally into two classes. The small improvement in the work of Shazia might be due to the quality of the images.

### 6.1.2 Results of the VGG16

Kong et al. [138] report an average accuracy for binary classification of 98.0%. They used a model based on VGG16 and DenseNet. Also, they report an average accuracy of 97.3% for three classes. They used a total of 5230 for training and 1288 for validation. In this paper [139], Chen et al obtained a testing accuracy of 98% on 10 epochs using transfer learning for binary classification. Kong used segmentation and more epochs in comparison with Chen but they obtained the same results for binary classification. We obtained a 95.72% for accuracy in binary classification using the VGG16 model, which is very close to the mentioned works. They used more image size and segmentation so their results are slightly better than ours. Also, in this work [140], they obtained an accuracy of 99.5% using 2000 images divided into COVID and Normal classes. They divided the dataset into training, validation, and testing. They used also transfer learning and resized the images to 64X64 to make the transfer learning model even faster. The high accuracy in this last work might be attributed to the perfect balance of the dataset and the advanced state of COVID-19, so the images might show better patterns and be better captured by the model. In general, the VGG19 using transfer learning proves to be time and resource-efficient compiling faster and with a low amount of epochs.

### 6.1.3 Results of the VGG19

In Table 6.1, we have summarized the values for accuracy and weighted average F1 score of the VGG19 on the 3 and 4 classes. We observe that adding more classes reduces the model F1 score. The reason might be the similarity in patterns between classes. Overall, the model performed very well on the four-class dataset. The weighted average F1 score of 89.04% indicates that this model might be used in real applications with very low errors in predicting the image class. The model might be trained on the biggest Kaggle dataset to see if the F1 score improves. Finally, the VGG19 model on the 3 and 4 classes is excellent at predicting the COVID class because the false positive numbers are very low compared to the other classes. We have obtained better results than the work of Sitaula et al. [128] on their VGG19 testing. The reasons might be the use of more images in our work and more pre-processing techniques on the datasets.

| VGG19 | Accuracy [%] | Weighted F1 Score[%] |
|---|---|---|
| 3-Classes | 92.44 | 92.32 |
| 4-Classes | 92.63 | 89.04 |

Table 6.1: Summary of the metrics that were calculated by using the VGG19 model on three and four-class datasets.

In this article [141] they tested modified transfer learning models. They used a data set consisting of 125 COVID-19, 500 atypical pneumonia, and 500 healthy X-ray images. Their data set is very small compared with ours. Let us review the results they obtained in Table 6.2.

| Model name | F1 Score[%] | Accuracy [%] |
|---|---|---|
| VGGCOV19-NET | 98.00 | 97.60 |
| Cascade VGGCOV19-NET | **99.75** | 99.84 |
| VGG19 | 97.68 | 98.56 |
| Cascade VGG19 | 98.49 | 99.04 |

Table 6.2: Summary of the results on binary classification of the modified VGG19 CNN.

If we compare the VGG19 results of both Table 5.5 and Table 6.2, our VGG19 transfer learning model performs slightly better with a bigger dataset and 30 epochs compared to the 100 epochs they used. The other modified models they used had very similar results, with Cascade VGGCOV19-NET obtaining the best result.

In another study [142], they used 3616 COVID-19 chest X-ray images and 10,192 healthy chest X-rays with VGG19 with 95% accuracy, which is a similar result to our binary accuracy. Now let us review VGG19 in three-class classification.

| Source | Dataset structure | Method | Accuracy [%] |
|---|---|---|---|
| [127] | 224 COVID-19, 700 Pneumonia, 504 Normal | TL-VGG-19 | **93.48** |
| [143] | 1300 COVID-19, 1300 Pneumonia, 1300 Normal | TL-VGG-19 | 92.92 |
| [144] | 130 COVID-19, 140 Pneumonia, 400 Normal | TL-VGG-19 | 87.00 |
| [145] | 260 COVID-19, 300 Pneumonia, 300 Normal | TL-VGG-19 | 89.30 |
| [146] | 1493 COVID-19, 2780 Pneumonia, 1538 Normal | VGG-19 | 72.52 |

Table 6.3: Review of VGG19 transfer learning (TL) models used three class datasets with different sizes.

The worst accuracy in Table 6.3 is seen in the VGG-19 simple model; the reason might be the lack of use of transfer learning compared to the other studies. If we compare these results with ours, we observe similar results around 92% for the three-class classification.

Also, comparing our work results with Table 3.10 mentioned in the state of the art, we need to make slight improvements in our experiment. They are using 33,920 images in total in comparison with our 27,052 images. Also, they use segmentation with U-net. Future work might include implementing their large dataset with segmentation and including more neural networks.

### 6.1.4 Work Limitations in Our Experiment

The principal limitation we encountered was the lack of computational resources. Initially, the project was tested on a local personal computer with 8GB of RAM and no GPU. The need to use a platform like Google Colab appeared. The compilation of the codes was running at the limit of the crash of the 1 GPU and 16 GB of RAM. When running the multiclass codes, Google Colab crashed due to a lack of RAM space. Then we have to run the codes in Kaggle, which offers a free 30 GB of RAM and two Tesla T4 GPUs. In the future, problems might appear with the compilation of the 33,920 image dataset. So, a paid version might be needed for this future work.

# Chapter 7

# Conclusions

## 7.1 Thesis General Conclusions

In this work, we present Xception, VGG16, and VGG19 convolutional neural networks using the transfer learning technique. The CNNs were trained using the public-access Kaggle dataset that contained chest X-ray images collected before and during the pandemic. The dataset was prepared, separating no relevant archives and not desired images; augmentation was used to increase the number of images and increase the accuracy both for training and prediction; and shuffling allowed for randomness in the training, ensuring better training and variability. We designed the CNNs using the pre-trained models provided by Keras following the documentation for the implementation using transfer learning. We trained validated and tested the models using a 70-15-15 percentage ratio. The CNNs successfully classified the images using two, three, and four classes. We used COVID-19, Normal, Lung-Opacity, and Pneumonia classes. The results were reported and compared to other investigations. The best model in our work was the VGG19. It performed better than the VGG16 and Xception in all the datasets. Our VGG19 neural network obtained a weighted F1 score of 98.53%, 92.32%, and 89.04% for the binary, three, and four class classifications, respectively. We were able to learn what zones of the chest X-ray images is capturing our model by using Grad-Cam. In the discussion of the results we mentioned the work of Tahir et al. that obtained accuracies above 99% using more images with U-net and neural networks. Finally, we stored in Google Drive the trained models which can be deployed to create software tools for public and private hospitals to help diagnose COVID-19.

## 7.2 Future Work Insights

The VGG19 neural network can be trained with the biggest Kaggle chest X-ray dataset to get even better results. This new model can be deployed in mobile apps and on other platforms. A new real-time detection system can be implemented using Nvidia hardware and similar portable GPUs. The neural network deployment can be implemented in several public institutions around Ecuador. The model is not limited to being trained on COVID. It can be trained with medical images like cancer histology datasets, magnetic resonance images, ocular diseases, etc. So, the model can be used to give access to new diagnostic tools with the help of artificial intelligence. Training the model with a larger dataset would require more than 30 GB of RAM and a robust GPU. The time to compile the model should not be more than 2 hours using transfer learning and the parameters presented in this work.

# Bibliography

[1] M. Ciotti, M. Ciccozzi, A. Terrinoni, W.-C. Jiang, C.-B. Wang, and S. Bernardini, "The covid-19 pandemic," *Critical reviews in clinical laboratory sciences*, vol. 57, no. 6, pp. 365–388, 2020.

[2] K. Yuki, M. Fujiogi, and S. Koutsogiannaki, "Covid-19 pathophysiology: A review," *Clinical immunology*, p. 108427, 2020.

[3] F. Almazán, I. Sola, S. Zuñiga, S. Marquez-Jurado, L. Morales, M. Becares, and L. Enjuanes, "Coronavirus reverse genetic systems: infectious clones and replicons," *Virus research*, vol. 189, pp. 262–270, 2014.

[4] N. H. Leung, D. K. Chu, E. Y. Shiu, K.-H. Chan, J. J. McDevitt, B. J. Hau, H.-L. Yen, Y. Li, D. K. Ip, J. Peiris *et al.*, "Respiratory virus shedding in exhaled breath and efficacy of face masks," *Nature medicine*, vol. 26, no. 5, pp. 676–680, 2020.

[5] J. Zhao, Q. Yuan, H. Wang, W. Liu, X. Liao, Y. Su, X. Wang, J. Yuan, T. Li, J. Li *et al.*, "Antibody responses to sars-cov-2 in patients with novel coronavirus disease 2019," *Clinical infectious diseases*, vol. 71, no. 16, pp. 2027–2034, 2020.

[6] A. L. Wyllie, J. Fournier, A. Casanovas-Massana, M. Campbell, M. Tokuyama, P. Vijayakumar, B. Geng, M. C. Muenker, A. J. Moore, C. B. Vogels *et al.*, "Saliva is more sensitive for sars-cov-2 detection in covid-19 patients than nasopharyngeal swabs," *MedRxiv*, pp. 2020–04, 2020.

[7] W. H. Organization. (2023) WHO COVID-19. Accessed: December 6, 2024. [Online]. Available: https://covid19.who.int/region/amro/country/ec

[8] I. N. de Estadística y Censos (INEC). (2019) Estadísticas defuncionales. Accessed: December 6, 2024. [Online]. Available: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Sitios/Defunciones/#ancla-1

[9] S. Stephanie, T. Shum, H. Cleveland, S. R. Challa, A. Herring, F. L. Jacobson, H. Hatabu, S. C. Byrne, K. Shashi, T. Araki *et al.*, "Determinants of chest radiography sensitivity for covid-19: a multi-institutional study in the united states," *Radiology: Cardiothoracic Imaging*, vol. 2, no. 5, p. e200337, 2020.

[10] S. Loza Travez, "Covid-19: Asi es la lucha diaria del hospital mas grande de quito," *Criterios Digital*, 02 2021, accessed on 2024-01-25. [Online]. Available: https://criteriosdigital.com/covid-19-asi-es-la-lucha-diaria-del-hospital-mas-grande-de-quito/

[11] Kaggle, "Covid-qu-ex dataset," https://www.kaggle.com/datasets/anasmohammedtahir/covidqu/?select=COVID-QU-Ex+dataset.txt, 2021.

[12] B. Mondal, "Artificial intelligence: state of the art," *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, pp. 389–425, 2020.

[13] W. Wang and K. Siau, "Artificial intelligence, machine learning, automation, robotics, future of work and future of humanity: A review and research agenda," *Journal of Database Management (JDM)*, vol. 30, no. 1, pp. 61–79, 2019.

[14] B.-h. Li, B.-c. Hou, W.-t. Yu, X.-b. Lu, and C.-w. Yang, "Applications of artificial intelligence in intelligent manufacturing: a review," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, pp. 86–96, 2017.

[15] C. Bravo, L. Saputelli, F. Rivas, A. G. Pérez, M. Nikolaou, G. Zangl, N. De Guzmán, S. Mohaghegh, and G. Nunez, "State of the art of artificial intelligence and predictive analytics in the e&p industry: a technology survey," *Spe Journal*, vol. 19, no. 04, pp. 547–563, 2014.

[16] M. Vanjani, M. Aiken, and M. Park, "Chatbots for multilingual conversations," *Journal of Management Science and Business Intelligence*, vol. 4, no. 1, pp. 19–24, 2019.

[17] L. Stark, "Facial recognition is the plutonium of ai," *XRDS: Crossroads, The ACM Magazine for Students*, vol. 25, no. 3, pp. 50–55, 2019.

[18] H. Dehghani and P. C. Kim, "Robotic automation for surgery," *Digital Surgery*, pp. 203–213, 2021.

[19] M. S. Rafieee, S. Jafari, H. S. Ahmadi, and M. Jafari, "Considerations to spoken language recognition for text-to-speech applications," in *2011 UkSim 13th International Conference on Computer Modelling and Simulation.* IEEE, 2011, pp. 304–309.

[20] A. Sohail, "Genetic algorithms in the fields of artificial intelligence and data sciences," *Annals of Data Science*, vol. 10, no. 4, pp. 1007–1018, 2023.

[21] M. Thielscher, "General game playing in ai research and education," in *KI 2011: Advances in Artificial Intelligence: 34th Annual German Conference on AI, Berlin, Germany, October 4-7, 2011. Proceedings 34.* Springer, 2011, pp. 26–37.

[22] M. Tian, Z. Shen, X. Wu, K. Wei, and Y. Liu, "The application of artificial intelligence in medical diagnostics: A new frontier," *Academic Journal of Science and Technology*, vol. 8, no. 2, pp. 57–61, 2023.

[23] N. Fatima, L. Liu, S. Hong, and H. Ahmed, "Prediction of breast cancer, comparative review of machine learning techniques, and their analysis," *IEEE Access*, vol. 8, pp. 150 360–150 376, 2020.

[24] H.-Y. Chiu, H.-S. Chao, and Y.-M. Chen, "Application of artificial intelligence in lung cancer," *Cancers*, vol. 14, no. 6, p. 1370, 2022.

[25] Y. Widaatalla, T. Wolswijk, F. Adan, L. Hillen, H. Woodruff, I. Halilaj, A. Ibrahim, P. Lambin, and K. Mosterd, "The application of artificial intelligence in the detection of basal cell carcinoma: A systematic review," *Journal of the European Academy of Dermatology and Venereology*, vol. 37, no. 6, pp. 1160–1167, 2023.

[26] S. Hansun, A. Argha, S.-T. Liaw, B. G. Celler, and G. B. Marks, "Machine and deep learning for tuberculosis detection on chest x-rays: Systematic literature review," *Journal of Medical Internet Research*, vol. 25, p. e43154, 2023.

[27] G. M. M. Alshmrani, Q. Ni, R. Jiang, H. Pervaiz, and N. M. Elshennawy, "A deep learning architecture for multi-class lung diseases classification using chest x-ray (cxr) images," *Alexandria Engineering Journal*, vol. 64, pp. 923–935, 2023.

[28] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: an overview," *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, pp. 599–608, 2021.

[29] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[30] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, "A systematic review on supervised and unsupervised machine learning algorithms for data science," *Supervised and unsupervised learning for data science*, pp. 3–21, 2020.

[31] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[32] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," *The elements of statistical learning: Data mining, inference, and prediction*, pp. 485–585, 2009.

[33] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval.* Springer, 2008, pp. 21–49.

[34] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[35] C. Spampinato, S. Palazzo, I. Kavasidis, D. Giordano, N. Souly, and M. Shah, "Deep learning human mind for automated visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6809–6817.

[36] J. Nayak, K. Vakula, P. Dinesh, B. Naik, and D. Pelusi, "Intelligent food processing: Journey from artificial neural network to deep learning," *Computer Science Review*, vol. 38, p. 100297, 2020.

[37] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017.

[38] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, “Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions,” *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[39] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[40] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep learning for computer vision: A brief review,” *Computational intelligence and neuroscience*, vol. 2018, 2018.

[41] H.-J. Yoo, “Deep convolution neural networks in computer vision: a review,” *IEIE Transactions on Smart Processing & Computing*, vol. 4, no. 1, pp. 35–43, 2015.

[42] J. Gao, Y. Yang, P. Lin, D. S. Park *et al.*, “Computer vision in healthcare applications,” 2018.

[43] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, “Computer vision for autonomous vehicles: Problems, datasets and state of the art,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[44] W. A. Hoff, K. Nguyen, and T. Lyon, “Computer-vision-based registration techniques for augmented reality,” in *Intelligent robots and computer vision XV: algorithms, techniques, active vision, and materials handling*, vol. 2904. SPIE, 1996, pp. 538–548.

[45] T.-H. The, Q.-V. Pham, X.-Q. Pham, T. Do-Duy, and T. Reddy Gadekallu, “Ai and computer vision technologies for metaverse,” *Metaverse Communication and Computing Networks: Applications, Technologies, and Approaches*, pp. 85–124, 2023.

[46] DEEPNETS, “Breast cancer image segmentation — attention unet,” https://www.kaggle.com/code/utkarshsaxenadn/breast-cancer-image-segmentation-attention-unet/notebook, 2023, accessed on 2024-02-11.

[47] C. Gershenson, “Artificial neural networks for beginners,” *arXiv preprint cs/0308031*, 2003.

[48] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, “What’s hidden in a randomly weighted neural network?” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 893–11 902.

[49] M. Islam, G. Chen, and S. Jin, “An overview of neural network,” *American Journal of Neural Networks and Applications*, vol. 5, no. 1, pp. 7–11, 2019.

[50] K. Fukushima, “Neocognitron,” *Scholarpedia*, vol. 2, no. 1, p. 1717, 2007.

[51] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[52] G. Boesch, "Vgg very deep convolutional networks (vggnet)-what you need to know," *Read more at: https://viso. ai/deep-learning/vgg-very-deep-convolutional-networks*, 2022.

[53] H. Law, "Large scale visual recognition challenge 2014."

[54] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics.* Elsevier, 2013, vol. 31, pp. 35–59.

[55] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[56] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[57] J. He, L. Li, J. Xu, and C. Zheng, "Relu deep neural networks and linear finite elements," *arXiv preprint arXiv:1807.03973*, 2018.

[58] J. Oh, S. Kim, C. Lee, J.-H. Cha, S. Y. Yang, S. G. Im, C. Park, B. C. Jang, and S.-Y. Choi, "Preventing vanishing gradient problem of hardware neuromorphic system by implementing imidazole-based memristive relu activation neuron," *Advanced Materials*, p. 2300023, 2023.

[59] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

[60] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *arXiv preprint arXiv:1903.06733*, 2019.

[61] J. Turian, J. Bergstra, and Y. Bengio, "Quadratic features and deep architectures for chunking," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 2009, pp. 245–248.

[62] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International workshop on artificial neural networks.* Springer, 1995, pp. 195–201.

[63] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[64] R. M. Neal, "Connectionist learning of belief networks," *Artificial intelligence*, vol. 56, no. 1, pp. 71–113, 1992.

[65] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning.* PMLR, 2017, pp. 933–941.

[66] V. Podlozhnyuk, “Image convolution with cuda,” *NVIDIA Corporation white paper, June*, vol. 2097, no. 3, 2007.

[67] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, “Dive into deep learning,” *D2L.ai*, 2022, convolutional Neural Networks chapter, Figure **??**. [Online]. Available: https://d2l.ai/

[68] M. Sewak, M. Karim, and P. Pujari, *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python.* Packt Publishing, 2018. [Online]. Available: https://books.google.com.ec/books?id=bOlODwAAQBAJ

[69] S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, “Impact of fully connected layers on performance of convolutional neural networks for image classification,” *Neurocomputing*, vol. 378, pp. 112–119, 2020.

[70] B. Machine, “Vggnet-16 architecture: A complete guide,” https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide, accessed: ¡access date¿.

[71] Y. Mangalmurti and N. Wattanapongsakorn, “Practical machine learning techniques for covid-19 detection using chest x-ray images.” *Intelligent Automation & Soft Computing*, vol. 34, no. 2, 2022.

[72] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[73] J. Kim, J. Moon, E. Hwang, and P. Kang, “Recurrent inception convolution neural network for multi short-term load forecasting,” *Energy and buildings*, vol. 194, pp. 328–341, 2019.

[74] M. Z. Alom, M. Hasan, C. Yakopcic, T. M. Taha, and V. K. Asari, “Improved inception-residual convolutional neural network for object recognition,” *Neural Computing and Applications*, vol. 32, pp. 279–293, 2020.

[75] M. Fabien, “Xception model and depthwise separable convolutions,” *URL: https://maelfabien. github. io/deeplearning/xception/#(Last accessed: 20.03. 2019)*, 2019.

[76] F. Chollet, “” xception: Deep learning with depthwise separable convolutions”, arxiv preprint,” *arXiv preprint arXiv:1610.02357*, 2016.

[77] J. Xu, “An intuitive guide to deep network architectures,” *An Intuitive Guide to Deep Network Architectures*, 2017.

[78] A. Mehmood, Y. Gulzar, Q. M. Ilyas, A. Jabbari, M. Ahmad, and S. Iqbal, “Sbxception: A shallower and broader xception architecture for efficient classification of skin lesions,” *Cancers*, vol. 15, no. 14, p. 3604, 2023.

[79] P. Baheti and P. Baheti, “A newbie-friendly guide to transfer learning,” *Last accessed*, vol. 1, 2022.

[80] D. Mwiti, “Transfer learning guide: A practical tutorial with examples for images and text in keras,” 2022.

[81] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, “Spottune: transfer learning through adaptive fine-tuning,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4805–4814.

[82] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, “Transfer learning in deep reinforcement learning: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[83] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[84] M. L. Mastery, “Difference between a batch and an epoch in a neural network,” 2020.

[85] M. Fiszman, W. W. Chapman, D. Aronsky, R. S. Evans, and P. J. Haug, “Automatic detection of acute bacterial pneumonia from chest x-ray reports,” *Journal of the American Medical Informatics Association*, vol. 7, no. 6, pp. 593–604, 2000.

[86] A. Taghizadieh, A. Ala, F. Rahmani, and A. Nadi, “Diagnostic accuracy of chest x-ray and ultrasonography in detection of community acquired pneumonia; a brief report,” *Emergency*, vol. 3, no. 3, p. 114, 2015.

[87] M. Blaivas, “Lung ultrasound in evaluation of pneumonia,” *Journal of Ultrasound in Medicine*, vol. 31, no. 6, pp. 823–826, 2012.

[88] D. S. Balk, C. Lee, J. Schafer, J. Welwarth, J. Hardin, V. Novack, S. Yarza, and B. Hoffmann, “Lung ultrasound compared to chest x-ray for diagnosis of pediatric pneumonia: a meta-analysis,” *Pediatric pulmonology*, vol. 53, no. 8, pp. 1130–1139, 2018.

[89] J. G. Bartlett, “Diagnostic test for etiologic agents of community-acquired pneumonia,” *Infectious Disease Clinics*, vol. 18, no. 4, pp. 809–827, 2004.

[90] D. Yaradou, “Legionella pneumophila : De la dÉtection dans les rÉseaux d'eau a l'Étude de l'invasion des cellules ÉpithÉliales pulmonaires humaines,” Ph.D. dissertation, 10 2007.

[91] A. Russo, “Spotlight on new antibiotics for the treatment of pneumonia,” *Clinical Medicine Insights: Circulatory, Respiratory and Pulmonary Medicine*, vol. 14, p. 1179548420982786, 2020.

[92] T. Jilani, A. Avula, A. Zafar Gondal *et al.*, “Active tuberculosis,” *StatPearls [Internet]*, Jan 26 2023, [Figure, Scanning electron micrograph, SEM, rod-shaped...].

[93] O. Filchakova, D. Dossym, A. Ilyas, T. Kuanysheva, A. Abdizhamil, and R. Bukasov, “Review of covid-19 testing and diagnostic methods,” *Talanta*, vol. 244, p. 123409, 2022.

[94] D. Caruso, M. Zerunian, M. Polici, F. Pucciarelli, T. Polidori, C. Rucci, G. Guido, B. Bracci, C. De Dominicis, and A. Laghi, “Chest ct features of covid-19 in rome, italy,” *Radiology*, vol. 296, no. 2, pp. E79–E85, 2020.

[95] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. J. Soufi, “Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning,” *Medical image analysis*, vol. 65, p. 101794, 2020.

[96] S. S. Khandker, N. H. H. Nik Hashim, Z. Z. Deris, R. H. Shueb, and M. A. Islam, “Diagnostic accuracy of rapid antigen test kits for detecting sars-cov-2: a systematic review and meta-analysis of 17,171 suspected covid-19 patients,” *Journal of Clinical Medicine*, vol. 10, no. 16, p. 3493, 2021.

[97] N. M. Okba, M. A. Müller, W. Li, C. Wang, C. H. GeurtsvanKessel, V. M. Corman, M. M. Lamers, R. S. Sikkema, E. de Bruin, F. D. Chandler *et al.*, “Severe acute respiratory syndrome coronavirus 2- specific antibody responses in coronavirus disease patients,” *Emerging infectious diseases*, vol. 26, no. 7, p. 1478, 2020.

[98] M. Cascella, M. Rajnik, A. Aleem, S. C. Dulebohn, and R. Di Napoli, “Features, evaluation, and treatment of coronavirus (covid-19),” *Statpearls [internet]*, 2022.

[99] E. C. Lloyd, T. N. Gandhi, and L. A. Petty, “Monoclonal antibodies for covid-19,” *Jama*, vol. 325, no. 10, pp. 1015–1015, 2021.

[100] D. Ndwandwe and C. S. Wiysonge, “Covid-19 vaccines,” *Current opinion in immunology*, vol. 71, pp. 111–116, 2021.

[101] G. Sternbach and J. Varon, “Wilhelm konrad roentgen: A new kind of rays,” *The Journal of emergency medicine*, vol. 11, no. 6, pp. 743–745, 1993.

[102] M. Manges, “The roentgen ray in the diagnosis of pneumonia, pleural diseases and pulmonary tumors,” *Transactions of the American Climatological and Clinical Association*, vol. 33, p. 162, 1917.

[103] E. K. Lewis and F. B. Lusk, “Roentgen diagnosis of primary atypical pneumonia,” *Radiology*, vol. 42, no. 5, pp. 425–434, 1944.

[104] M. Lederman, “The early history of radiotherapy: 1895–1939,” *International Journal of Radiation Oncology* Biology* Physics*, vol. 7, no. 5, pp. 639–648, 1981.

[105] A. U. Desjardins, “Radiotherapy for inflammatory conditions,” *Journal of the American Medical Association*, vol. 96, no. 6, pp. 401–408, 1931.

[106] E. J. Calabrese and G. Dhawan, "How radiotherapy was historically used to treat pneumonia: could it be useful today?" *The Yale journal of biology and medicine*, vol. 86, no. 4, p. 555, 2013.

[107] J. H. Musser and D. L. Edsall, "A study of metabolism in leukemia, under the influence of the x-ray. with a consideration of the manner of action of the x-ray and of some precautions desirable in its therapeutic use," *Transactions of the Association of American Physicians*, vol. 20, p. 294, 1905.

[108] J. Rousseau, W. Johnson, and G. T. Harrell, "The value of roentgen therapy in pneumonia which fails to respond to the sulfonamides," *Radiology*, vol. 38, no. 3, pp. 281–289, 1942.

[109] E. M. Chamorro, A. D. Tascón, L. I. Sanz, S. O. Vélez, and S. B. Nacenta, "Radiologic diagnosis of patients with covid-19," *Radiología (English Edition)*, vol. 63, no. 1, pp. 56–73, 2021.

[110] H. Y. F. Wong, H. Y. S. Lam, A. H.-T. Fong, S. T. Leung, T. W.-Y. Chin, C. S. Y. Lo, M. M.-S. Lui, J. C. Y. Lee, K. W.-H. Chiu, T. W.-H. Chung *et al.*, "Frequency and distribution of chest radiographic findings in patients positive for covid-19," *Radiology*, vol. 296, no. 2, pp. E72–E78, 2020.

[111] Saludesa, "Salud familiar," *Saludesa*, 2023. [Online]. Available: https://saludesa.org.ec/wp/salud-familiar/

[112] S. Manna, J. Wruble, S. Z. Maron, D. Toussie, N. Voutsinas, M. Finkelstein, M. A. Cedillo, J. Diamond, C. Eber, A. Jacobi *et al.*, "Covid-19: a multimodality review of radiologic techniques, clinical utility, and imaging features," *Radiology: Cardiothoracic Imaging*, vol. 2, no. 3, p. e200210, 2020.

[113] S. Schiaffino, S. Tritella, A. Cozzi, S. Carriero, L. Blandi, L. Ferraris, and F. Sardanelli, "Diagnostic performance of chest x-ray for covid-19 pneumonia during the sars-cov-2 pandemic in lombardy, italy," *Journal of thoracic imaging*, vol. 35, no. 4, pp. W105–W106, 2020.

[114] H. Dawit, J.-P. Salameh, S. Kazi, N. Fabiano, L. Treanor, M. Absi, F. Ahmad, P. Rooprai, A. Al Khalil, K. Harper *et al.*, "Thoracic imaging tests for the diagnosis of covid-19," *Cochrane Database of Systematic Reviews*, no. 5, 2022.

[115] Ecuador, "Govierno del ecuador," 2023. [Online]. Available: https://www.salud.gob.ec/hospital-de-duran-cuenta-con-nuevo-tomografo-de-alta-tecnologia/#:~:text=En%20el%20sector%20privado%20el,de%20Salud%20a%20nivel%20nacional.

[116] G. D. Rubin, C. J. Ryerson, L. B. Haramati, N. Sverzellati, J. P. Kanne, S. Raoof, N. W. Schluger, A. Volpi, J.-J. Yim, I. B. K. Martin, D. J. Anderson, C. Kong, T. Altes, A. Bush, S. R. Desai, o. Goldin, J. M. Goo, M. Humbert, Y. Inoue, H.-U. Kauczor, F. Luo, P. J. Mazzone, M. Prokop, M. Remy-Jardin, L. Richeldi, C. M. Schaefer-Prokop, N. Tomiyama, A. U. Wells, and A. N. Leung, "The role of chest imaging in patient management during the covid-19

pandemic: A multinational consensus statement from the fleischner society," *Radiology*, vol. 296, no. 1, pp. 172–180, 2020, pMID: 32255413. [Online]. Available: https://doi.org/10.1148/radiol.2020201365

[117] N. Tsiknakis, E. Trivizakis, E. E. Vassalou, G. Z. Papadakis, D. A. Spandidos, A. Tsatsakis, J. Sánchez-García, R. López-González, N. Papanikolaou, A. H. Karantanas *et al.*, "Interpretable artificial intelligence framework for covid-19 screening on chest x-rays," *Experimental and Therapeutic Medicine*, vol. 20, no. 2, pp. 727–735, 2020.

[118] F. Zhang, "Application of machine learning in ct images and x-rays of covid-19 pneumonia," *Medicine*, vol. 100, no. 36, 2021.

[119] Z. Jiao, J. W. Choi, K. Halsey, T. M. L. Tran, B. Hsieh, D. Wang, F. Eweje, R. Wang, K. Chang, J. Wu *et al.*, "Prognostication of patients with covid-19 using artificial intelligence based on chest x-rays and clinical data: a retrospective study," *The Lancet Digital Health*, vol. 3, no. 5, pp. e286–e294, 2021.

[120] B. Ghoshal and A. Tucker, "Estimating uncertainty and interpretability in deep learning for coronavirus (covid-19) detection," *arXiv preprint arXiv:2003.10769*, 2020.

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

[122] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya, "Automated detection of covid-19 cases using deep neural networks with x-ray images," *Computers in biology and medicine*, vol. 121, p. 103792, 2020.

[123] W. Liu, J. Luo, Y. Yang, W. Wang, J. Deng, and L. Yu, "Automatic lung segmentation in chest x-ray images using improved u-net," *Scientific Reports*, vol. 12, no. 1, p. 8649, 2022.

[124] G. Gaál, B. Maga, and A. Lukács, "Attention u-net based adversarial architectures for chest x-ray lung segmentation," *arXiv preprint arXiv:2003.10304*, 2020.

[125] Y. Cao, Z. Xu, J. Feng, C. Jin, X. Han, H. Wu, and H. Shi, "Longitudinal assessment of covid-19 using a deep learning–based quantitative ct pipeline: illustration of two cases," *Radiology: Cardiothoracic Imaging*, vol. 2, no. 2, p. e200082, 2020.

[126] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[127] I. D. Apostolopoulos and T. A. Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and engineering sciences in medicine*, vol. 43, pp. 635–640, 2020.

[128] C. Sitaula and M. B. Hossain, "Attention-based vgg-16 model for covid-19 chest x-ray image classification," *Applied Intelligence*, vol. 51, pp. 2850–2863, 2021.

[129] M. Jang, M. Kim, S. J. Bae, S. H. Lee, J.-M. Koh, and N. Kim, "Opportunistic osteoporosis screening using chest radiographs with deep learning: development and external validation with a cohort dataset," *Journal of Bone and Mineral Research*, vol. 37, no. 2, pp. 369–377, 2022.

[130] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi *et al.*, "Can ai help in screening viral and covid-19 pneumonia?" *Ieee Access*, vol. 8, pp. 132 665–132 676, 2020.

[131] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. A. Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan *et al.*, "Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images," *Computers in biology and medicine*, vol. 132, p. 104319, 2021.

[132] A. M. Tahir, M. E. Chowdhury, A. Khandakar, T. Rahman, Y. Qiblawey, U. Khurshid, S. Kiranyaz, N. Ibtehaz, M. S. Rahman, S. Al-Maadeed *et al.*, "Covid-19 infection localization and severity grading from chest x-ray images," *Computers in biology and medicine*, vol. 139, p. 105002, 2021.

[133] K. COVID, "Radiography database," *Radiological Society of North America (2019). Available online at: https://www. kaggle. com/tawsifurrahman/covid19-radiography-database (accessed October 1, 2021)*, 19.

[134] F. Chollet. (2020) Complete guide to transfer learning & fine-tuning in keras. Date created: April 15, 2020. Last modified: June 25, 2023. Accessed: December 6, 2024. [Online]. Available: https://keras.io/guides/transfer_learning/

[135] B. Juarez, "Vgg19 multiclass transfer learning," https://www.kaggle.com/code/bryanjuarez/vgg19-multiclass-transfer-learning, Year Uploaded.

[136] Nvidia. T4. https://www.nvidia.com/en-us/data-center/tesla-t4/. Date accessed: December 6, 2024.

[137] A. Shazia, T. Z. Xuan, J. H. Chuah, J. Usman, P. Qian, and K. W. Lai, "A comparative study of multiple neural network for detection of covid-19 on chest x-ray," *EURASIP journal on advances in signal processing*, vol. 2021, pp. 1–16, 2021.

[138] L. Kong and J. Cheng, "Classification and detection of covid-19 x-ray images based on densenet and vgg16 feature fusion," *Biomedical Signal Processing and Control*, vol. 77, p. 103772, 2022.

[139] A. Chen, J. Jaegerman, D. Matic, H. Inayatali, N. Charoenkitkarn, and J. Chan, "Detecting covid-19 in chest x-rays using transfer learning with vgg16," in *CSBio'20: proceedings of the eleventh international conference on computational systems-biology and bioinformatics*, 2020, pp. 93–96.

[140] A. Panthakkan, S. Anzar, S. Al Mansoori, and H. Al Ahmad, "Accurate prediction of covid-19 (+) using ai deep vgg16 model," in *2020 3rd International Conference on Signal Processing and Information Security (ICSPIS)*. IEEE, 2020, pp. 1–4.

[141] A. Karacı, “Vggcov19-net: automatic detection of covid-19 cases from x-ray images using modified vgg19 cnn architecture and yolo algorithm,” *Neural Computing and Applications*, vol. 34, no. 10, pp. 8253–8274, 2022.

[142] S. Akter, F. J. M. Shamrat, S. Chakraborty, A. Karim, and S. Azam, “Covid-19 detection using deep learning algorithm on chest x-ray images,” *Biology*, vol. 10, no. 11, p. 1174, 2021.

[143] M. A. Fayemiwo, T. A. Olowookere, S. A. Arekete, A. O. Ogunde, M. O. Odim, B. O. Oguntunde, O. O. Olaniyan, T. O. Ojewumi, I. S. Oyetade, A. A. Aremu *et al.*, “Modeling a deep transfer learning framework for the classification of covid-19 radiology dataset,” *PeerJ Computer Science*, vol. 7, p. e614, 2021.

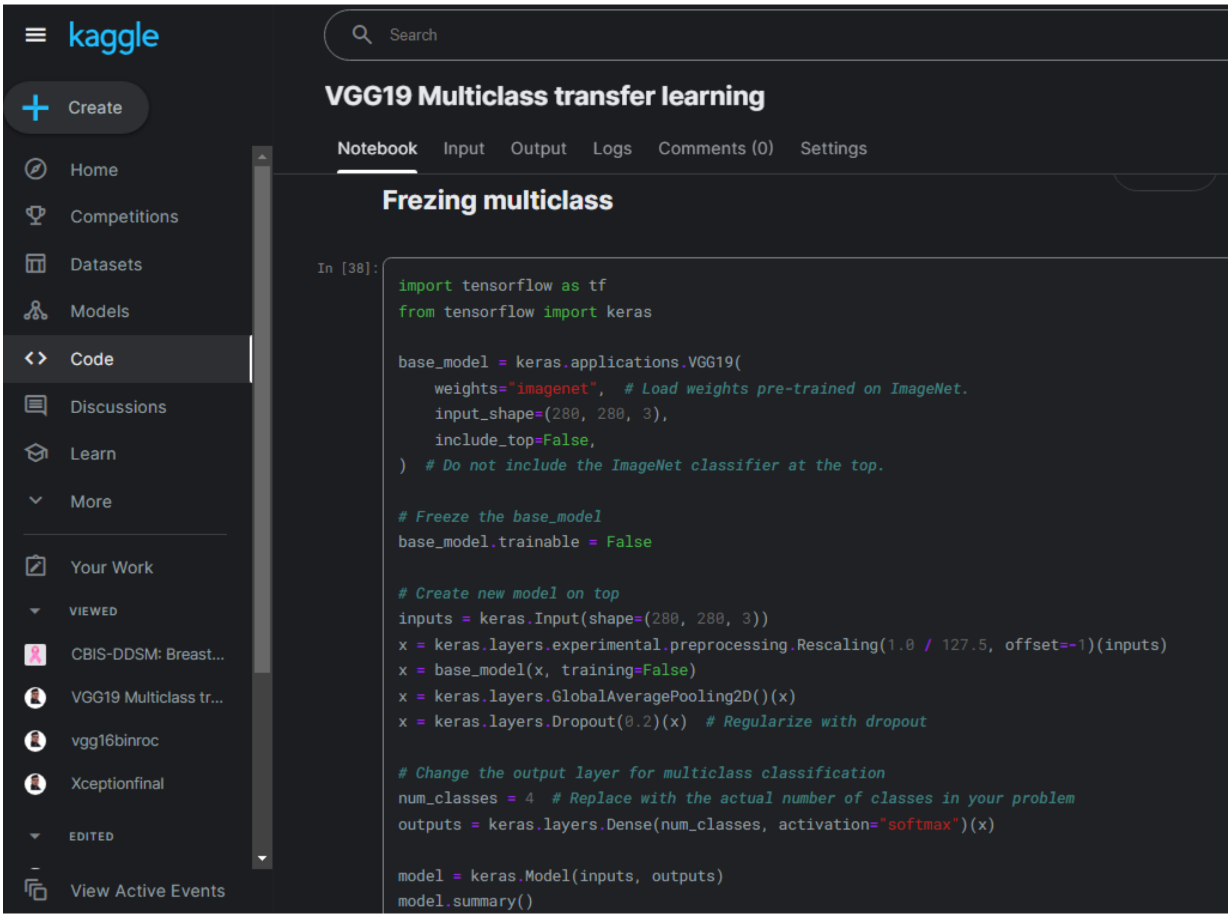
[144] M. J. Horry, S. Chakraborty, M. Paul, A. Ulhaq, B. Pradhan, M. Saha, and N. Shukla, “Covid-19 detection through transfer learning using multimodal imaging data,” *Ieee Access*, vol. 8, pp. 149 808–149 824, 2020.

[145] M. M. Rahaman, C. Li, Y. Yao, F. Kulwa, M. A. Rahman, Q. Wang, S. Qi, F. Kong, X. Zhu, and X. Zhao, “Identification of covid-19 samples from chest x-ray images using deep learning: A comparison of transfer learning approaches,” *Journal of X-ray Science and Technology*, vol. 28, no. 5, pp. 821–839, 2020.

[146] K. El Asnaoui and Y. Chawki, “Using x-ray images and deep learning for automated detection of coronavirus disease,” *Journal of Biomolecular Structure and Dynamics*, vol. 39, no. 10, pp. 3615–3626, 2021.

# Appendices

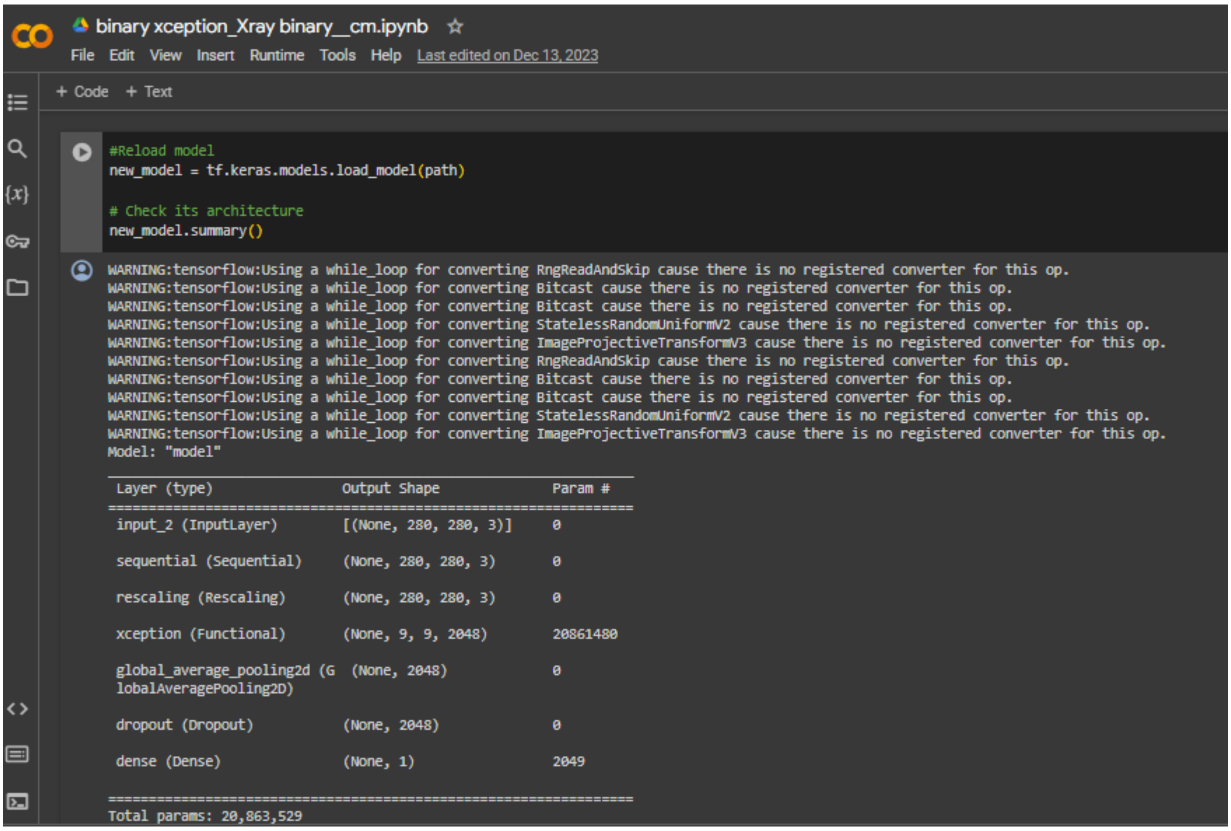
## 7.3 Appendix 1



```
import tensorflow as tf
from tensorflow import keras

base_model = keras.applications.VGG19(
    weights="imagenet",  # Load weights pre-trained on ImageNet.
    input_shape=(280, 280, 3),
    include_top=False,
)  # Do not include the ImageNet classifier at the top.

# Freeze the base_model
base_model.trainable = False

# Create new model on top
inputs = keras.Input(shape=(280, 280, 3))
x = keras.layers.experimental.preprocessing.Rescaling(1.0 / 127.5, offset=-1)(inputs)
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)  # Regularize with dropout

# Change the output layer for multiclass classification
num_classes = 4  # Replace with the actual number of classes in your problem
outputs = keras.layers.Dense(num_classes, activation="softmax")(x)

model = keras.Model(inputs, outputs)
model.summary()
```

Figure 7.1: Screen capture of the model we built in the Kaggle platform.

## 7.4 Appendix 2


binary xception_Xray binary__cm.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Last edited on Dec 13, 2023
+ Code
+ Text

```
#Reload model
new_model = tf.keras.models.load_model(path)

# Check its architecture
new_model.summary()
```

```
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting RngReadAndSkip cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting Bitcast cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting StatelessRandomUniformV2 cause there is no registered converter for this op.
WARNING:tensorflow:Using a while_loop for converting ImageProjectiveTransformV3 cause there is no registered converter for this op.
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 280, 280, 3)]     0

 sequential (Sequential)     (None, 280, 280, 3)       0

 rescaling (Rescaling)       (None, 280, 280, 3)       0

 xception (Functional)       (None, 9, 9, 2048)        20861480

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 20,863,529
```

Figure 7.2: Google Colab code used to train and test the Xception pre-trained model.