TÉCNICO LISBOA

Classification of Chest X-Ray Images with Deep Neural Networks for Detecting COVID-19

Inês Vanessa Carneiro Filipe

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Arlindo Manuel Limede de Oliveira

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. Jacinto Carlos Marques Peixoto do Nascimento

June 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to offer my most sincere thank you to my family, my parents, for providing me with the opportunity and tools to have an education and the guidance to become a better person, and my sister for her friendship, encouragement and care. I would also like to thank my grandfather, that would have wished to be here to see this Thesis finished.

I would also like to acknowledge my dissertation supervisors Prof. Bruno Martins and Prof. Arlindo Oliveira for their endless patience, insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that supported me through my Bachelor's and Master's with their cooperation in both group projects and in understanding the most various concepts, thank you for being willing to offer a helping hand.

Abstract

This work addresses the task of accurately classifying chest X-ray images, differentiating images from healthy patients, patients with pneumonia, and patients with COVID-19. Experiments have compared convolutional and Vision Transformer architectures. The main research objectives focused on assessing the use of self-attention in neural networks for classifying medical images, testing the Swin Transformer architecture, and modifying well-known convolutional neural networks based on recent state-of-the-art models, e.g. as in the ConvNeXT architecture. The modifications included the use of depthwise separable convolutions, new activation functions, or inverted bottleneck blocks, among others. While Vision Transformers are considered the state-of-the-art architecture for image classification, in comparison convolutional neural networks possess a natural advantage in computer vision tasks involving smaller datasets, and were thus considered a viable path for this specific task. The experimental results show that the task of COVID-19 detection in chest X-ray images is complex. While the obtained results were relatively satisfactory, with an accuracy of over 80%, there were no significant differences in the results for baseline models, namely conventional convolutional neural networks such as ResNet50, VGG16, or DenseNet121, and the more advanced models that were studied in this work, namely a variant of the Swin Transformer and a variant of the ConvNeXT architecture. The ConvNeXT model did provide a slight performance improvement in the accuracy, macro precision, macro recall, weighted precision, and weighted recall metrics, but the difference was not significant enough to claim that this model corresponds to an improvement over more conventional architectures for the task.

Keywords

Deep Learning, Computer Vision, COVID-19, Image Classification.

Resumo

Este trabalho aborda a tarefa de classificar de forma exata imagens de raio-X para deteção de COVID-19, diferenciando imagens de pacientes saudáveis, pacientes com pneumonia, e pacientes com COVID-19. Experiências compararam redes neuronais convolucionais e *Vision Transformers*. Os objetivos principais foram avaliar o uso de *self-attention* em redes neuronais para classificação de imagens de contexto médico, testar a arquitetura *Swin Transformer*, e modificar redes neuronais convolucionais conhecidas com base em modelos estado-da-arte recentes, e.g. como na arquitetura ConvNeXT. As modificações incluíram o uso de *depthwise separable convolutions*, novas funções de ativação, *bottleneck blocks* invertidos, entre outras. Embora *Vision Transformers* sejam consideradas as arquiteturas de estado-da-arte para classificação de imagens, em comparação as redes neuronais convolucionais possuem uma vantagem natural em tarefas de visão computacional em situações nas quais os *datasets* têm uma dimensão reduzida, e foram deste modo consideradas um caminho viável a seguir nas nossas experiências. Os resultados experimentais mostram que a tarefa de classificar raios-X do tórax é complexa. Embora os resultados obtidos sejam relativamente satisfatórios, com exatidão superior a 80%, não houve diferenças significativas nos resultados dos nossos modelos de *baseline*, nomeadamente redes neuronais convolucionais tais como a *ResNet50*, *VGG16*, *DenseNet121*, e os modelos estudados nesta dissertação, uma variante da *Swin Transformer* e uma variante da *ConvNeXT*. O modelo ConvNeXT forneceu uma ligeira melhoria de desempenho em exatidão, macro e *weighted precision* e *recall*, mas a diferença não é suficientemente significativa para afirmar que este modelo se apresenta como uma melhoria sobre arquiteturas mais convencionais nesta tarefa.

Palavras Chave

Aprendizagem Profunda, Visão Computacional, COVID-19, Classficação de Imagens.

Contents

1 Introduction **1**

1.1 Objectives . . . 4

1.2 Methodology . . . 4

1.3 Results and Contributions . . . 5

1.4 Thesis Outline . . . 5

2 Fundamental Concepts **7**

2.1 Introduction to Neural Networks . . . 9

2.2 Convolutional Neural Networks for Image Classification . . . 9

2.2.1 Essential Aspects of Convolutional Neural Networks . . . 9

2.2.2 Residual Connections . . . 13

2.2.3 Dense Connections . . . 15

2.3 Transformers for Image Classification . . . 17

2.3.1 Transformer Architecture . . . 17

2.3.2 Vision Transformer . . . 20

2.4 Summary . . . 22

3 Related Work **23**

3.1 Conventional Convolutional Neural Network Models for COVID-19 Detection . . . 25

3.2 Recent Models Created or Adapted for COVID-19 Detection . . . 31

3.3 Summary . . . 42

4 Model Architectures and Baselines **43**

4.1 Models and Training Methods . . . 45

4.1.1 Baselines . . . 45

4.1.2 Swin Transformer . . . 46

4.1.3 ConvNeXT . . . 46

4.1.4 Training Methods . . . 47

4.2 Dataset and Metrics Used . . . 47

4.2.1 COVIDx CXR-3 Dataset . . . 47

4.2.2 Metrics . . . 49
4.3 Summary . . . 51
5 Experimental Evaluation **53**
5.1 Results . . . 55
5.1.1 Training and Testing Accuracy . . . 55
5.1.2 Confusion Matrices, Precision, Recall, and F1 Score per Class . . . 55
5.1.3 Macro and Weighted Metrics . . . 56
5.2 Discussion . . . 57
5.2.1 Dataset and Associated Challenges . . . 57
5.2.2 Baselines Versus Experimental Models . . . 58
5.2.3 Epochs and Overfitting . . . 58
5.3 Summary . . . 58
6 Conclusions and Future Work **61**
6.1 Contributions . . . 63
6.2 Future Work . . . 63
Bibliography **65**

List of Figures

2.1 Graphical representation of the convolution operation. The input is a 5×5 tensor with zero padding added, with $p_w = p_h = 1$. The kernel is a 3×3 tensor, and the input feature map is a 5×5 tensor. Given these parameters, if $s_h = s_w = 1$, then we can conclude that the output feature map's size is 5×5. 10

2.2 On the left of the figure, a convolution with an input involving multiple channels is represented. The convolution is performed between the corresponding input and kernel values of each channel, and then summed. On the right, we see a representation of a convolution layer with more than one kernel. The convolution is performed on the same input with different kernels, generating multiple feature maps. 11

2.3 On the left of the figure, we see the pooling operation represented. The operation is applied on a 3×3 window of the input. If $s_h = s_w = 1$ then we can conclude that the feature map's new size is 3×3. On the right we see an example of pooling on a 4×4 input, with 2×2 window, $s_h = s_w = 2$, and with the max function. 12

2.4 Architecture of the LeNet5 network. Annotations provide the number of channels and size of the feature maps, as well as the number of layers for fully-connected layers. The operations performed at each stage are also specified (convolution, pooling). 12

2.5 Original residual block of the ResNet (left) and residual block with bottleneck (right). . . . 14

2.6 A 5-layer dense block with a growth rate of $k = 3$. Each layer takes all preceding feature maps as input. The set of operations represented with H_l corresponds to batch normalization (BN), the ReLU activation and a 3×3 convolution (only explicitly represented in H_1). 16

2.7 DenseNet with 3 dense blocks. Convolution and pooling layers between dense blocks are transition layers that alter the feature map sizes. 16

2.8 Representation of Transformer structure. 19

2.9 Preparation of the input image for the Vision Transformer. $*$ is the $[class]$ embedding. . . . 21

2.10 On the left, the structure of the ViT. On the right, the internal structure of the ViT encoder. 21

3.1 Structure of the VGG19 network. 25
3.2 On the left, we can see the naive Inception module. On the right, we see the Inception module with dimensionality reductions (bottleneck), used in Inceptionv1. 26
3.3 On the left, we see one of the building blocks of the Inception-ResNet architecture. Note the residual connection from the input (after ReLU activation) and the summation of the output of the convolutions and the input. On the right, we can see a representation of the convolution operation performed by the Xception and MobileNet architectures. 26
3.4 SqueezeNet's fire module. 30
3.5 COVID-Net's PEPx module from Wang and Wong (2020). 32
3.6 The COVID-Net architecture from Wang and Wong (2020). 32
3.7 The DenResCov-19 architecture from Mamalakis et al. (2021). 36
3.8 Swin Transformer block pair. 38
3.9 Ilustration of Swin Transformer's shifted window. 39
3.10 Swin Transformer Architecture (Swin-T variant). 40

4.1 On the left, the ResNet bottleneck block. On the right, the ConvNeXT inverted bottleneck block. 47

List of Tables

3.1 Table of parameters used in the networks evaluated by Apostolopoulos and Mpesiana (2020). . . . 27
3.2 Table of results for the networks evaluated in Apostolopoulos and Mpesiana (2020). . . . 28
3.3 Confusion matrix of the best-performing networks evaluated by Apostolopoulos and Mpesiana (2020). . . . 28
3.4 Results obtained by Narin et al. (2021) with the 3 binary datasets: dataset 1 (COVID-19 and normal), dataset 2 (COVID-19 and viral pneumonia) and dataset 3 (COVID-19 and bacterial pneumonia). . . . 29
3.5 Results obtained in Minaee et al. (2020). . . . 31
3.6 Number of parameters of each architecture and sensitivity per class in Wang and Wong (2020). . . . 33
3.7 Positive prediction value (PPV) per class for each architecture tested by Wang and Wong (2020). . . . 33
3.8 Results for each class for the 4-class CoroNet model from Khan et al. (2020). . . . 35
3.9 Results for each CoroNet model from Khan et al. (2020). . . . 35
3.10 Data in each dataset used by Mamalakis et al. (2021). . . . 37
3.11 Results for each dataset used by Mamalakis et al. (2021). . . . 37
3.12 Results obtained in Le Dinh et al. (2022) 42

4.2 Dataset sample count per class and data source, rearranged for our experiments. . . . 48
4.1 Dataset sample count per class and data source, with Wang and Wong (2020)'s original train/test split. . . . 48
4.3 Example of a confusion matrix. . . . 50

5.1 Accuracy results of baselines and studied models. . . . 55
5.2 Confusion matrices of baselines and studied models. . . . 56
5.3 Precision, Recall, and F1 per class for each model. . . . 56

5.4 Macro and Weighted Precision, Recall, and F1 per for each model. 57

1

Introduction

Contents

1.1 Objectives 4

1.2 Methodology 4

1.3 Results and Contributions 5

1.4 Thesis Outline 5

The coronavirus disease (COVID-19) is a contagious infection caused by the Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2). This disease, firstly identified in December 2019, in Wuhan, China, has since spread worldwide, originating the ongoing pandemic.

SARS-CoV-2 spreads mainly through small droplets or aerosols from infected people (for example, from breathing, coughing, or sneezing) entering in contact with other people's eyes, nose or mouth. At least one third of infected people remain asymptomatic, while in the remaining cases symptoms begin one to 14 days after exposure, commonly including fever, cough, fatigue, breathing difficulties, loss of smell and taste. Around 14% of the symptomatic individuals develop more severe symptoms, such as dyspnea or hypoxia, and 5% develop critical symptoms, such as respiratory failure or organ dysfunction.

Currently, there are a few tests available to diagnose COVID-19, the most notable one being the Reverse Transcription-Polymerase Chain Reaction (RT-PCR) test. However, for diagnosing the infection, the RT-PCR test is complex and expensive, and may be of limited availability. It is then necessary to find alternative diagnostic techniques to identify infected patients, especially in a disease with high infection rates.

X-ray imaging is one of the most common techniques used in the medical field for diagnosis. The ionizing radiation emitted from the machines towards the patients' bodies is absorbed unevenly by bones and soft tissues, allowing the creation of two-dimensional images of white (i.e., bones) and black (i.e., soft tissues) contrast.

Chest X-rays (CXR) are taken to ascertain the condition of the lungs, heart, and chest wall, and they are common diagnostic tools in respiratory diseases like influenza and pneumonia, at the same time also being relatively cheap. As such, with COVID-19 being a disease that affects the respiratory system and causes pneumonia, it is logical that chest X-rays can be used as part of clinical protocols for diagnosis: chest X-ray images are collected from patients suspected to have the coronavirus disease, analyzed by radiologists and, if evidence is found, other tests can be performed.

However, given that CXR of infected people have diverse characteristics, the diagnosis of COVID-19 requires expert radiologists to analyze the images. In this context, automatic image classification methods can be valuable for obtaining faster diagnosis, especially with the surge of infected patients. Modern methods based on deep neural networks are already successful at identifying pneumonia from X-ray images, rivalling expert radiologists.

Recent studies (Ilyas et al., 2020; Shi et al., 2021; Ulhaq et al., 2020) have reported the use of deep neural networks for diagnosing COVID-19, and although discriminating between COVID-19 and other types of pneumonia remains a challenging problem, automatic classification methods can be used to allocate resources more effectively (e.g. to prioritize the selection of images to be analyzed by expert radiologists, or to prioritize patients for RT-PCR testing), especially for understaffed and/or underfunded hospitals.

These automatic classification methods often employ convolutional neural networks, but recent developments have also made vision transformers a viable alternative, achieving state-of-the-art results in several computer vision tasks.

1.1 Objectives

This Master of Science thesis project aims to explore the use of transformer and deep convolutional neural network architectures to analyze chest X-ray images and discriminate between healthy patients, patients with COVID-19, and patients diagnosed with other types of pneumonia. This would be a valuable tool in situations where other tests are not readily available, and where radiologists would otherwise be overworked. To accomplish this, we will use deep learning techniques integrating state-of-the-art developments in transformer architectures, and convolutional models based on transformer architectures for computer vision.

Training and testing will be performed with a publicly available dataset, the COVIDx CXR-3 (Wang and Wong, 2020) dataset, a database of thousands of CXR images of COVID-19 cases, along with healthy and viral pneumonia images.

1.2 Methodology

The first chapters of this thesis dissertation seek to provide a good theoretical foundation to justify the reasoning behind the proposed model. For this purpose, we conducted a survey on the current state-of-the-art methods in the field of computer vision and image classification tasks, with special focus on studies that applied these methods on X-ray image classification for COVID-19 detection.

After understanding the challenges that we could encounter while performing this task, we sought to find a model with potential to rival or surpass state-of-the-art results, and a suitable dataset. After downloading and adapting the dataset for the task ahead, three convolutional neural network baselines were established (i.e. ResNet (He et al., 2015), VGG (Simonyan and Zisserman, 2015), and DenseNet (Huang et al., 2018)), and a convolutional neural network inspired by developments in vision transformers was fine-tuned and tested on the adapted dataset.

The models were implemented in Python, making use of its machine learning libraries, namely Pytorch[1], numpy[2], scikit-learn[3], transformers[4], among others. The source code is available on GitHub[5].

[1] https://pytorch.org
[2] https://numpy.org
[3] https://scikit-learn.org/
[4] https://huggingface.co/transformers
[5] https://github.com/inesfilipe/COVIDSwinConvNeXT

1.3 Results and Contributions

The main idea and contribution of this work is the study of the performance of traditional convolutional neural network models and state-of-the-art models for COVID-19 chest X-ray image classification. We evaluate the impact that advancements in architectures for computer vision have on performance for tasks in a medical context. The models were pre-trained on ImageNet dataset (Deng et al., 2009), and fine-tuned on the COVIDx CXR3 (Wang and Wong, 2020) dataset.

We find that the task of classifying chest X-rays is complex. While the results obtained were relatively satisfactory, with an accuracy over 80%, there were no significant improvements in results between our baseline models, conventional convolutional neural networks such as ResNet50 (He et al., 2015), VGG16 (Simonyan and Zisserman, 2015), DenseNet121 (Huang et al., 2018), and the studied models in this thesis, a variant of the Swin Transformer (Liu et al., 2021) and a variant of the ConvNeXT (Liu et al., 2022). The ConvNeXT model did provide a slight performance improvement in accuracy, macro precision, macro recall, weighted precision and weighted recall metrics, however we considered the difference to not be statistically significant to claim that the studied model presents itself as an improvement on conventional architectures for the task.

1.4 Thesis Outline

This dissertation is organized as follows: Chapter 2 introduces fundamental concepts regarding convolutional networks, and popular convolutional architectures, along with concepts regarding transformers, specifically transformers used for computer vision and image classification. Chapter 3 reviews related work pertaining to image classification in the context of COVID-19 diagnosis. Chapter 4 details the models, introduces the dataset and the modifications performed, training methods and metrics employed. Chapter 5 presents and discusses the results obtained. Chapter 6 presents the conclusions of this project, and future work that can be performed.

2

Fundamental Concepts

Contents

2.1 Introduction to Neural Networks 9
2.2 Convolutional Neural Networks for Image Classification 9
2.3 Transformers for Image Classification 17
2.4 Summary 22

This chapter presents fundamental convolutional neutral network (CNN) and transformer concepts, explaining the theory behind these models and introducing relevant architectures used for the task of image classification.

2.1 Introduction to Neural Networks

In neural networks research, several studies have been inspired by what is seen in the real world. For example, the perceptron, i.e. the basic unit of neural networks, was a model inspired by biological neurons, like the ones in our brains, according to Rosenblatt (1957). Another example is the case of convolutional neural networks (CNNs). Studies focusing on cat and monkey visual cortexes (Hubel and Wiesel, 1959, 1968) noted that they possess neurons that respond individually to small regions of the visual field, called the receptive field of the neurons. The neurons can have simple or complex receptive fields, with neurons having complex fields being of higher order, receiving input from neurons with simple fields.

Hubel and Wiesel's work inspired the Neocognitron (Fukushima and Miyake, 1982), i.e. a neural network for visual pattern recognition. This model is composed of layers of cells, where each cell receives input from cells from a limited region of the preceding layer. The cells are able to extract features, going from local features (such as lines or edges), generally in earlier layers, to more global features in higher layers.

The LeNet5 (LeCun et al., 1998) model was created from improvements over the Neocognitron model, and it is considered to be the first CNN. It allowed for character recognition to be performed without designing features by hand, being more robust towards differences in the input such as size, slant or positions, compared to fully-connected neural networks, due to the use of local receptive fields, shared weights, and spatial sub-sampling.

2.2 Convolutional Neural Networks for Image Classification

This section introduces fundamental concepts of convolutional neural networks. We first address essential aspects of the architecture, followed by residual connection and dense connections.

2.2.1 Essential Aspects of Convolutional Neural Networks

CNNs receive as input n-dimensional tensors, which undergo several operations while traversing the layers of the network. The typical layers found in all CNNs include convolutional layers, non-linear projection layers, and pooling layers. The results of the operations of these layers, corresponding to

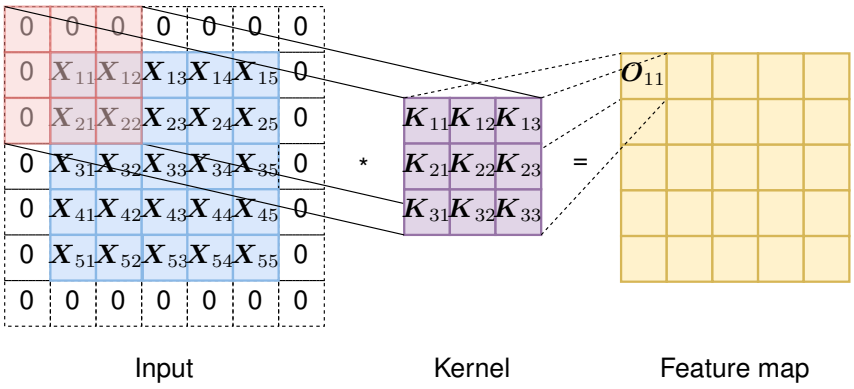

Figure 2.1: Graphical representation of the convolution operation. The input is a 5×5 tensor with zero padding added, with $p_w = p_h = 1$. The kernel is a 3×3 tensor, and the input feature map is a 5×5 tensor. Given these parameters, if $s_h = s_w = 1$, then we can conclude that the output feature map's size is 5×5.

m-dimensional tensors, are the input for a final fully-connected layer, which generates a probability distribution for a set of classes, in the case of an image classification task.

The primary operation performed by these types of networks is a convolution (Figure 2.1), represented by the symbol $*$. Given a $k_h \times k_w$ tensor $\boldsymbol{K}_{ij}$, for a given input tensor $\boldsymbol{X} \in \mathbb{R}^{H \times W}$, the convolution of $\boldsymbol{X}$ by $\boldsymbol{K}$ is the tensor $\boldsymbol{O} = \boldsymbol{X} * \boldsymbol{K}$ where the coordinates are defined as:

$$\boldsymbol{O}_{ij} = (\boldsymbol{X} * \boldsymbol{K})_{ij} = \sum_{h=i}^{k_h} \sum_{w=1}^{k_w} \boldsymbol{X}_{(i+h-1)(j+w-1)} \boldsymbol{K}_{hw}, \tag{2.1}$$

More intuitively, a kernel $\boldsymbol{K}$ corresponding to a tensor of weights slides along the input, horizontally, starting at the top left corner, until the top right corner, where it goes back to the left but shifted down, and continues until it reaches the bottom right of the input. At each step, the dot product between the kernel and the receptive field (the window of the input covered by the kernel in each step) is computed, and the result is stored in the output tensor, commonly referred to as the feature map.

The kernel has a height H and weight W, which determine the size of the area of the input over which the dot product will be performed on, at each step. The kernel also has parameters to define the amount it shifts in each step: the stride across the width s_w, and the stride across the height s_h. Furthermore, the kernel also has a parameter called padding, which corresponds to adding padding on the width (p_w) and on the height (p_h) of the input during the convolution, i.e. adding pixels (typically with value 0) around the borders of the input. This technique allows, for example, for the kernel to not skip areas at the boundary of the original input due to its size/stride, or to prevent the dimensionality of the output from being reduced.

Usually, convolution is not only applied to one input matrix, but to a collection of tensors where each is called a channel. For example, as we can see in Figure 2.2, an input image is represented using RGB colors, it has 3 channels (red, green and blue), each with its set of pixel values. In this case, the

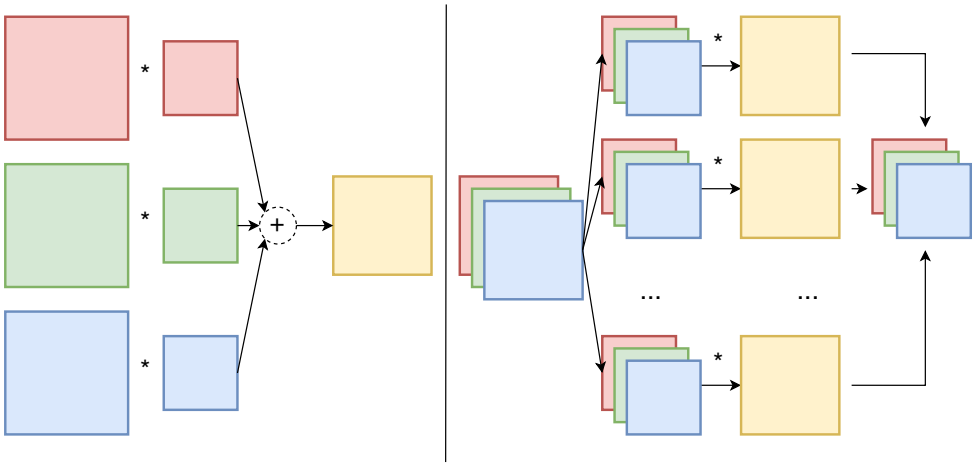

Figure 2.2: On the left of the figure, a convolution with an input involving multiple channels is represented. The convolution is performed between the corresponding input and kernel values of each channel, and then summed. On the right, we see a representation of a convolution layer with more than one kernel. The convolution is performed on the same input with different kernels, generating multiple feature maps.

kernel will also have 3 channels, one for each color channel, and the result of the convolution between the kernel and the pixel tensor in each channel is summed element-wise. Multiple kernels can be used to create output feature maps with multiple channels, by concatenating the outputs of each kernel.

Generally, a convolution layer computes the operation between the input and a set of different kernels. We can say then that the output of a convolution layer with k kernels is a tensor with k channels.

After the convolutions, CNNs also typically involve a non-linear projection, i.e. applying a non-linear activation function to the output of the convolutional layer (also referred to as the feature map). A few examples of activation functions are the hyperbolic tangent and the ReLU (Nair and Hinton, 2010).

Another operation that is typically employed in CNNs is the pooling operation, shown in Figure 2.3. This is similar to the convolution operation, in the sense that it is also applied in steps, with a window sliding along the layer's input. However, unlike in convolutions, which have a kernel with weights that can be changed as the network learns, the pooling operation has no kernel, and performs a function, which is generally either taking the maximum, the minimum, or the average value within the receptive field, yielding the corresponding value to the output. This operation reduces the resolution of the feature map, leading to a reduction of dimensionality (improving training time) and decreasing the sensitivity to displacements in the input image (shift invariance). Similarly to the convolution layer's kernels, the pooling layer has a window size, depth (number of channels) and stride. Additionally, pooling layers also include the function which they apply.

CNNs, in general, will have several convolutional and pooling layers, feeding the final feature maps into a fully-connected network. The CNN's layers process the input by extracting the relevant features, and the fully-connected network classifies the image by generating a probability distribution for all classes with the softmax function. Both the CNN and the fully-connected network are trained simul-

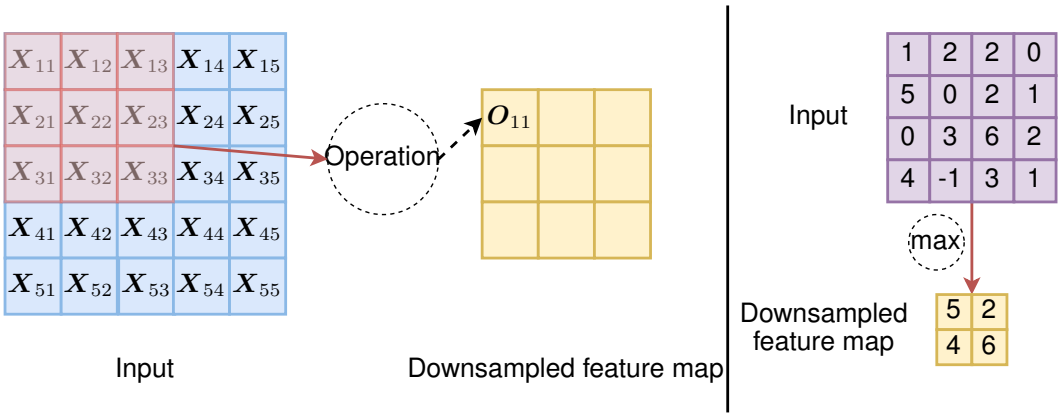

Figure 2.3: On the left of the figure, we see the pooling operation represented. The operation is applied on a 3×3 window of the input. If $s_h = s_w = 1$ then we can conclude that the feature map's new size is 3×3. On the right we see an example of pooling on a 4×4 input, with 2×2 window, $s_h = s_w = 2$, and with the max function.

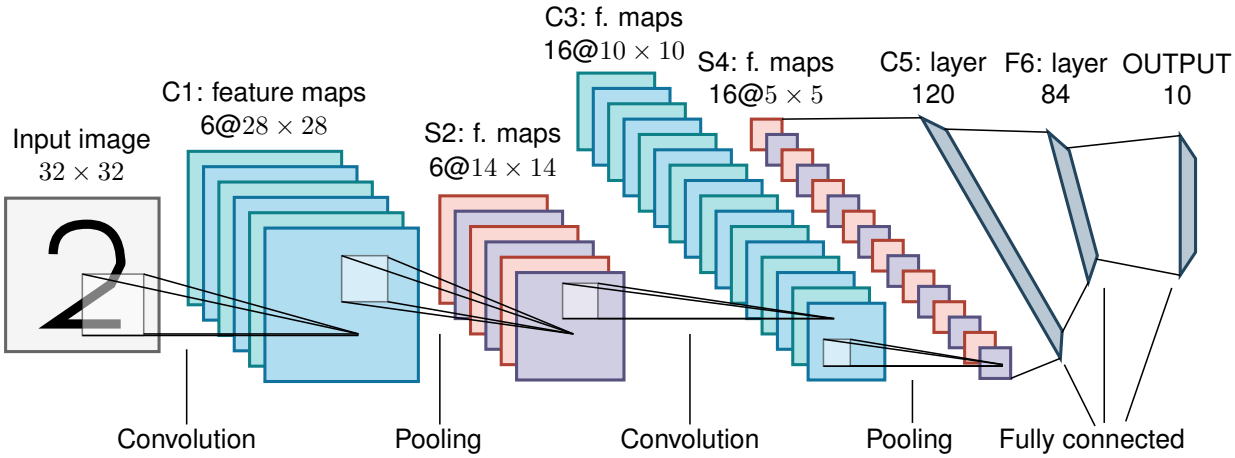

Figure 2.4: Architecture of the LeNet5 network. Annotations provide the number of channels and size of the feature maps, as well as the number of layers for fully-connected layers. The operations performed at each stage are also specified (convolution, pooling).

taneously using backpropagation.

As mentioned previously, the LeNet5 (LeCun et al., 1998) architecture is a representative of early CNN architectures, applying the fundamental concepts explained above to classify images of handwritten digits from the MNIST database. The model, represented in Figure 2.4, is composed of 7 layers: 2 sets of convolutional layers, 2 sets of pooling layers, 2 fully-connected layers, and a softmax classifier. For the non-linear projections, the sigmoid activation function is applied, and the average function is performed in the pooling layers.

Following the appearance of LeNet5 (LeCun et al., 1998), further research was made on image classification using CNNs, leading to results which made the usage of these networks commonplace for the task.

One of the architectures which contributed to this change in paradigm was AlexNet (Krizhevsky et al., 2012), i.e. the CNN that won the ILSVRC edition of 2012. This was an image classification competition using a subset of the dataset named ImageNet (Deng et al., 2009), and AlexNet was the first CNN to win the competition.

Key aspects associated to this model included the use of GPUs for training, making use of their parallel processing capabilities, the application of ReLU as the activation function, allowing for a reduction in training time, and the employment of data augmentation and dropout regularization, reducing overfitting.

Networks with ReLU consistently converged faster than equivalents with other activation functions (like the hyperbolic tangent) to low error rates. Data augmentation artificially increases the size of the dataset, transforming the images in the dataset without changing their labels by generating image translations (cutting random patches of a specific size from the original images) and creating horizontal reflections, training the network with both the original and reflected versions, as well as changing the intensities of the RGB channels. Dropout (Hinton et al., 2012) is a technique that consists of setting the output of hidden neurons to 0 randomly, with a defined probability (for example, 0.5 probability). This contributes to reducing the dependency between neurons, forcing them to learn more robust features and creating a better generalization of the features.

2.2.2 Residual Connections

AlexNet opened the path for further research on image classification with CNNs, with its record-breaking results. Further hardware advancements allowed for larger and deeper networks to be created, although certain problems were observed when training very large networks (e.g., in networks with larger depth, accuracy saturated and decreased rapidly). ResNet (He et al., 2015) attempted to address this problem.

The authors of the ResNet model claimed that, in principle, a deeper model should not perform worse than a shallow one if the added layers are the identity mapping, with the others being copies of the shallow model's layers. However, what was observed was the opposite, signifying that the addition of more layers did not necessarily lead to better results, and will lead to worse results at a certain threshold.

The authors explored if the observed results were due to the vanishing gradients problem, i.e. the gradients converged to zero, but they verified that the problem remained even with the use of BN, i.e. batch normalization (Ioffe and Szegedy, 2015), a method that alters a layer's inputs to achieve zero mean and unit variance within each input batch of the training process. This method guarantees that forward signals do not have zero variance, and ensures that backward gradients do not vanish.

They suggested the introduction of deep residual learning. Considering $\mathrm{H}(\boldsymbol{X})$ to be the underlying mapping fit by a few of the layers, with $\boldsymbol{X}$ being the input to those layers, and hypothesizing that non linear layers can approximate any function, then non linear layers can also approximate the function $\mathrm{F}(\boldsymbol{X}) := \mathrm{H}(\boldsymbol{X}) - \boldsymbol{X}$, if $\mathrm{H}(\boldsymbol{X})$ and $\boldsymbol{X}$ have the same dimensions. This means that, instead of attempting

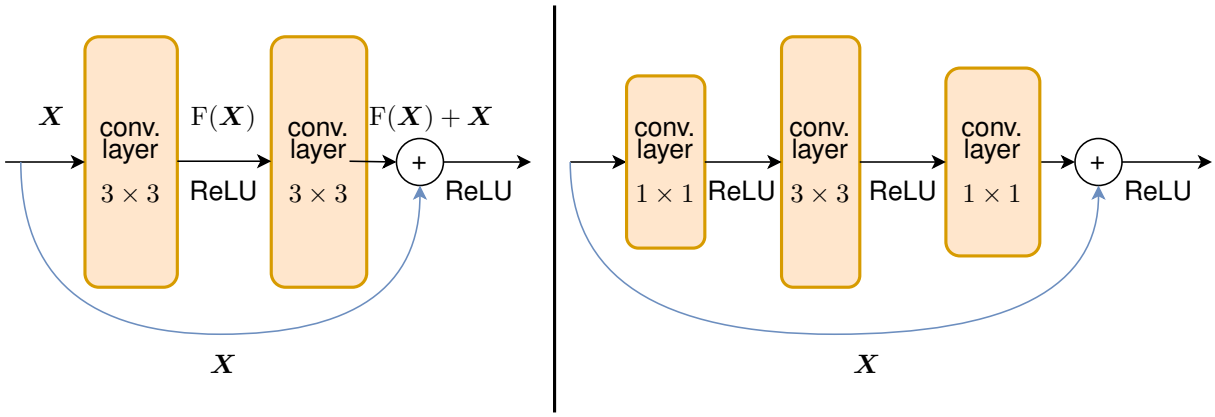

Figure 2.5: Original residual block of the ResNet (left) and residual block with bottleneck (right).

to approximate the function $\mathrm{H}(\boldsymbol{X})$, the layers can attempt to approximate the function $\mathrm{F}(\boldsymbol{X})$, making the original function $\mathrm{H}(\boldsymbol{X}) = \mathrm{F}(\boldsymbol{X}) + \boldsymbol{X}$. The intuition is that, in the extreme case where the identity mapping is optimal, then the residual $\mathrm{F}(\boldsymbol{X})$ only needs to be set to zero, instead of approximating the identity mapping using non linear layers.

In practice, He et al. (2015) suggested the implementation of residual learning through residual blocks, as seen in Figure 2.5: in every few stacked layers, a shortcut connection is added, i.e. a connection that skips layers. The input of the first convolutional layer of the stack skips to the output of the last convolutional layer, and element-wise addition is performed with the output of the last layer, with the activation function of the last layer being applied to the result of the element-wise addition, or $\mathrm{F}(\boldsymbol{X}) + \boldsymbol{X}$. Batch normalization is performed after every convolution layer and before the application of the activation function.

These residual blocks can not only address the accuracy degradation, but the vanishing gradient problem as well, since the error is directly backpropagated through the shortcut connections. This contributes to solving two of the issues that arise from the increase in network depth.

Additionally, the authors also proposed a building block with modifications in the convolution layers. Instead of having 2 3×3 convolution layers in a block, the modified block (Figure 2.5) has 3 layers of 1×1, 3×3 and 1×1 convolutions. The first 1×1 layer reduces the dimensionality, while the last 1×1 layer restores the dimensionality for the element-wise addition with the input, creating a bottleneck on the 3×3 layer. The introduction of the 1×1 convolution layers to reduce dimensions (largely used in the Inception architecture by Szegedy et al. (2015), i.e. the winner of ILSVRC 2014) leads to a smaller increase in the number of parameters of the network with the increase in depth, and consequently to a smaller increase in the training time with network depth increase. This bottleneck block design keeps training times from increasing considerably with the increase in the number of layers in the network.

The previously described building blocks were used to create a set of networks called ResNets, which were deeper than the state-of-the-art networks at the time. These networks achieved lower training and

validation errors than their shallower counterparts. Like the previously mentioned AlexNet, the ResNet entered and won the ILSVRC competition (2015 edition). The bottleneck block design allowed for the creation of ResNets with over 100 layers, achieving state-of-the-art results at the time.

2.2.3 Dense Connections

The concept of skip connections from the ResNet architecture was further extended, notably by Huang et al. (2018) with the proposal of the DenseNet architecture, partly inspired also by Huang et al. (2016), which explored stochastic depth to successfully train networks with over 1000 layers. Huang et al. (2016) showed that dropping layers randomly during training produced better performance, indicating that certain layers in the network can be redundant.

Huang et al. (2018) then introduced dense connectivity, by not only adding direct connections from a layer l to the layer $l+1$, but introducing direct connections from layer l to all layers after. In consequence, where in ResNets, the input of layer l would be:

$$\boldsymbol{X}_l = \mathrm{H}_l(\boldsymbol{X}_{l-1}) + \boldsymbol{X}_{l-1}, \tag{2.2}$$

in the DenseNet proposal the input of layer l is:

$$\boldsymbol{X}_l = \mathrm{H}_l([\boldsymbol{X}_0, \boldsymbol{X}_1, ..., \boldsymbol{X}_{l-1}]), \tag{2.3}$$

where H_l, in the case of DenseNet, is a composite function comprised of batch normalization (BN), ReLU activation, and a 3×3 convolution, where $[\boldsymbol{X}_0, \boldsymbol{X}_1, ..., \boldsymbol{X}_{l-1}]$ is the concatenation of all of the feature maps of the previous layers. This design ensures that information is fully preserved in subsequent layers, unlike in ResNet blocks where features are summed element-wise.

The dense connections maximize information and gradient flows between layers, making them easier to train: each layer receives direct supervision from the loss function, improving accuracy.

It is important to notice that the concatenation operation cannot be applied on feature maps with differing sizes, since it interferes with the use of downsampling layers, essential in convolutional networks. To overcome this problem, Huang et al. (2018) created dense blocks, represented in Figure 2.6, in which the layers within are densely connected, much like the residual blocks from ResNet (He et al., 2015) have residual connections. DenseNets (Figure 2.7) are built with dense blocks interspersed with transition layers, making the use of downsampling operations viable. Specifically, DenseNet performs batch normalization (BN) after dense blocks, and has 1×1 convolution and 2×2 average pooling as transition layers.

Additionally, the authors introduced a hyperparameter k for the growth rate of the network, which sets the number of feature maps produced in each layer. If H_l produces k feature maps, then the

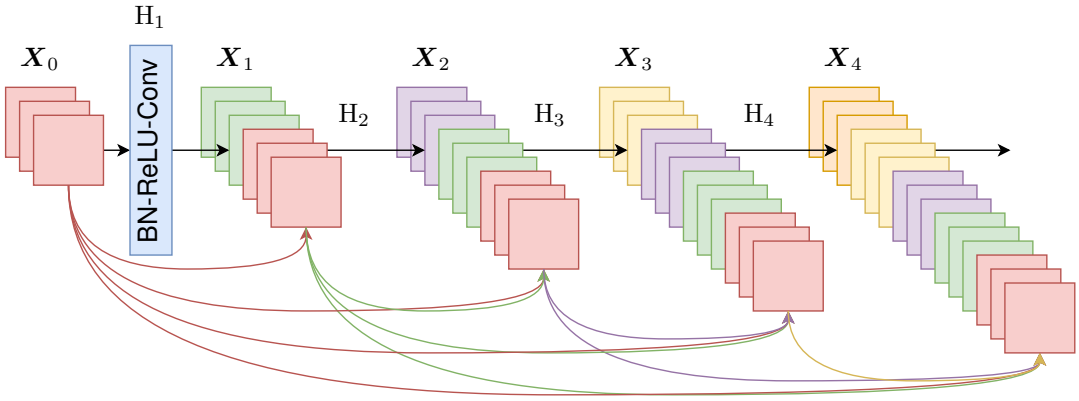

Figure 2.6: A 5-layer dense block with a growth rate of $k = 3$. Each layer takes all preceding feature maps as input. The set of operations represented with H_l corresponds to batch normalization (BN), the ReLU activation and a 3×3 convolution (only explicitly represented in H_1).

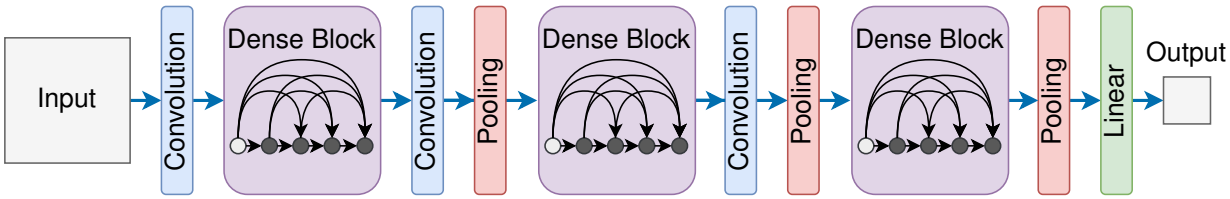

Figure 2.7: DenseNet with 3 dense blocks. Convolution and pooling layers between dense blocks are transition layers that alter the feature map sizes.

l^{th} layer has $k_0 + k \times (l - 1)$ input feature maps, with k_0 being the number of channels of the input layer. Notably, Huang et al.'s experiments showed that DenseNets could achieve state-of-the-art results with "narrow" layers, or with a relatively low growth rate, proving that this model is parameter efficient: the combination of dense connections with input concatenation allows for the use of narrower layers. Each layer has access to unmodified knowledge of previous layers and adds only a small amount of information, encouraging feature reuse and reducing relearning of redundant feature maps.

Dense blocks can also employ bottleneck layers to increase computational efficiency, like ResNets, namely by adding a 1×1 convolution before each 3×3 convolution layer, reducing the number of input feature maps. The resulting operations performed are BN-ReLU-Conv(1×1)-BN-ReLU-Conv(3×3), instead of BN-ReLU-Conv(3×3) in the original block. Additionally, the number of feature maps can also decrease at the transition layers: given the compression factor $\theta \in [0, 1]$ and a dense block containing m feature maps, the following transition layer generates $\theta \cdot m$ feature maps. When $\theta = 1$, the number of feature maps stays unchanged.

State-of-the-art results were obtained in tests for DenseNets with original dense blocks, with growth rate $k = 12$ and $k = 24$. The best results overall, in 3 out of the 4 datasets that were used for tests, were obtained for DenseNets with bottleneck layers, compression with $\theta = 0.5$ and with growth rate $k = 12$, $k = 24$, and $k = 40$.

2.3 Transformers for Image Classification

In this section we explore an alternative architecture for computer vision tasks, the transformer. We start by explaining the original concepts behind the architecture, followed by presenting the model applied to image classification.

2.3.1 Transformer Architecture

The transformer architecture Vaswani et al. (2017) was initially created to tackle machine translation tasks. It is organized in two different components, an encoder and a decoder. The goal is to map an input sequence into an output sequence which can be of different lengths. The encoder receives an input sequence $\boldsymbol{x} = \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots \boldsymbol{x}_n$, which in the task of machine translation is a sequence of tokens where each token is a representation of a word, and maps each item into a hidden vector $\boldsymbol{h} = \boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \ldots \boldsymbol{h}_n$ known as context vector, that represents the item and its context. These context vectors are provided to the decoder, which uses the new representation to generate an output sequence $\boldsymbol{y} = \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \ldots \boldsymbol{y}_n$, which in machine translation is a sequence of tokens where each represents a word in a different language, using the previous output as additional input when generating the current output.

Unlike previous architectures in the field of natural language processing, transformers employ attention mechanisms. Attention provides not only improvements in machine translation tasks but also in computer vision, with the concept of soft and hard attention. In soft attention, the whole image is being attended to, and in hard attention, patches of the image are attended to. Additionally, it is also being used in the form of self-attention, where the model attends to the input sequence itself, in an attempt that, by focusing on different words, the model is able to understand the internal structure and connections between the elements of the input.

The self-attention mechanism in the transformer works like the following: the input sequence vectors are stacked (Equation 2.4) and multiplied by an initial set of weights. The result of these multiplications is a set of 3 matrices - the query $\boldsymbol{Q}$ (Equation 2.5), the keys $\boldsymbol{K}$ (Equation 2.6), and the $\boldsymbol{V}$ values (Equation 2.7) matrices.

$$\boldsymbol{I} = \text{Concat}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \tag{2.4}$$

$$\boldsymbol{Q} = \boldsymbol{I} \cdot \boldsymbol{W}_Q \tag{2.5}$$

$$\boldsymbol{K} = \boldsymbol{I} \cdot \boldsymbol{W}_K \tag{2.6}$$

$$\boldsymbol{V} = \boldsymbol{I} \cdot \boldsymbol{W}_V \tag{2.7}$$

$$\boldsymbol{y} = Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right) \cdot \boldsymbol{V} \tag{2.8}$$

The attention is calculated as in Equation 2.8 by first multiplying the query matrix by the transpose of the keys matrix, and dividing the result by the square root of d, the dimension of the key vectors. This formula, called the scaled dot product attention, prevents exploding or vanishing gradients. Afterwards, the result is fed to a softmax function and multiplied by the values matrix, providing the model with the attention weights that indicate how each token relates to one another in the sequence.

In order to better capture the different relations between the elements of the input sequence, the authors have also included multi-head attention mechanisms (Equations 2.9, 2.10), i.e. the attention mechanisms have multiple units calculating the self-attention using different sets of weights for the query, key, and value matrices, $\boldsymbol{W}_q^s$, $\boldsymbol{W}_k^s$, $\boldsymbol{W}_v^s$, for each head.

$$\text{head}_s = \text{Attention}(\boldsymbol{I}\boldsymbol{W}_q^s, \boldsymbol{I}\boldsymbol{W}_k^s, \boldsymbol{I}\boldsymbol{W}_v^s) \tag{2.9}$$

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_s) \cdot \boldsymbol{W_O} \tag{2.10}$$

The result of the attention heads is then concatenated and multiplied by the matrix $\boldsymbol{W}_O$.

In order to maintain a notion of the different positioning of the elements in the input even when the model receives input sequence in a concatenated form and processes it simultaneously, each of the vectors that represent the items in the input sequence have information about their position added, called positional encoding, as per Equations 2.11 and 2.12.

$$\text{PE}_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}) \tag{2.11}$$

$$\text{PE}_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}) \tag{2.12}$$

The approach of Vaswani et al. (2017) uses sinusoidal functions, where pos is the position of the element in the input sequence, i is the dimension and d_{model} the size of the model encodings. For example, in a model where $d_{model} = 4$, the resulting positional encoding $\boldsymbol{p}_e$ would be of the form

$$\begin{aligned} \boldsymbol{p}_e &= \left[\sin\left(\frac{pos}{10000^0}\right), \cos\left(\frac{pos}{10000^0}\right), \sin\left(\frac{pos}{10000^{2/4}}\right), \cos\left(\frac{pos}{10000^{2/4}}\right)\right] \\ &= \left[\sin\left(pos\right), \cos\left(pos\right), \sin\left(\frac{pos}{100}\right), \cos\left(\frac{pos}{100}\right)\right] \end{aligned} \tag{2.13}$$

Equation 2.13 satisfies important criteria: the encoding for each element in the input sequence is

unique; distance between any two positions is consistent even when input sequences have different lengths, the values are bounded (namely in the interval $[-1, 1]$), and it is deterministic. Additionally, the authors argue this function makes it possible for the model to attend to relative positions, since any position can be represented as a linear function of any other position.

After calculating the positional encoding of each element in the input sequence, the positional encoding is summed to the vector of the respective element in the input sequence.

$$\boldsymbol{I} = \text{Concat}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \text{Concat}(\boldsymbol{i}_1 + \boldsymbol{p}_e^1, \boldsymbol{i}_2 + \boldsymbol{p}_e^2, \ldots, \boldsymbol{i}_n + \boldsymbol{p}_e^n) \tag{2.14}$$

We can now understand the overall structure of the transformer architecture.

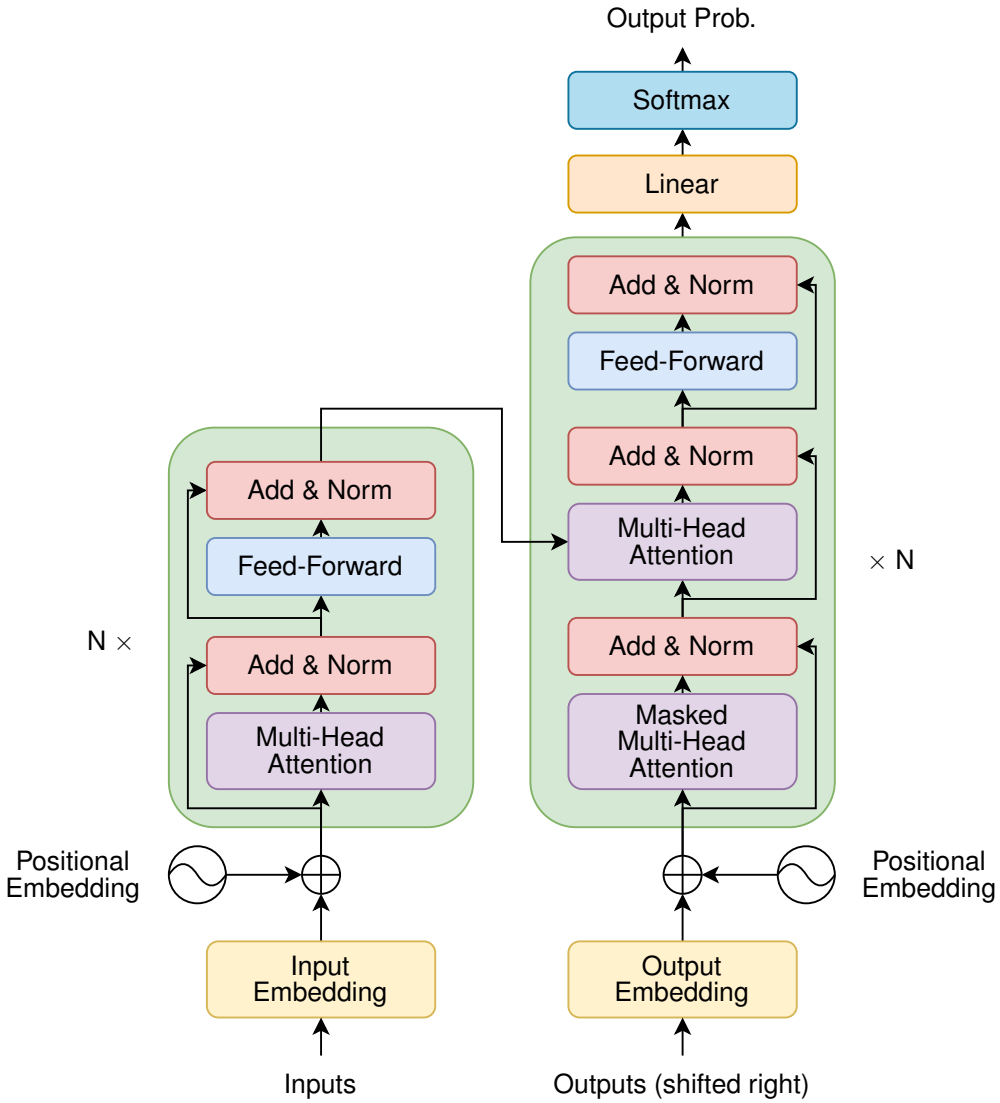

Figure 2.8: Representation of Transformer structure.

It consists of an encoder and a decoder of N layers each. The encoder block receives the input

sequence (in the first encoder, that input sequence has the positional encoding embedded), and has two sub-layers: the multi-head self-attention, that outputs the self-attention for the input sequence, and a fully-connected feed-forward network.

Both of these sub-layers' outputs are first added to a residual connection and normalized, before being fed to the next layer/block in the model. The decoder blocks have a similar structure, with the addition of a multi-head encoder-decoder attention layer between the multi-head self-attention and fully-connected feed-forward networks, so that the decoder can attend to the input sequence, and the modification of the first multi-head self-attention layer, by adding a mask, since the decoder is given the entire output sequence as input, and without masking, it would have access to the elements of the sequence it hadn't yet predicted. The output of the last decoder block is then provided to a linear layer, which redimensions it to a vector with the dimension of the output space (i.e. the number of words in the vocabulary for the task of machine translation), and then through softmax, to normalize the scores for each element in the output space, and the one with the highest score is chosen.

2.3.2 Vision Transformer

The transformer architecture was initially created with the task of machine translation in mind, without concerns for whether it could be used for tasks such as image classification, where convolutional neural networks were the gold standard.

To experiment on whether transformers would also be an adequate option for computer vision tasks, a Vision Transformer (Dosovitskiy et al., 2020) was designed as close as possible to the original Transformer design in Vaswani et al. (2017).

The first modification is the input of the transformer. In natural language processing tasks, the input is a word sequence with positional encodings. In the case of images, the logical input would be the pixels of an image. However, computing the attention matrix has quadratic complexity, and having every pixel attend to every other pixel in the input would be too heavy for most hardware.

As such, the authors suggest dividing the input image into fixed size patches, and flattening them. The flattened patches are sent through a feed-forward layer to obtain a linear patch projection. Then, a learnable $[class]$ embedding is concatenated to the other patch projections. Like in the original transformer model, positional encoding is added to each patch according to its position in the full image (including the class embedding).

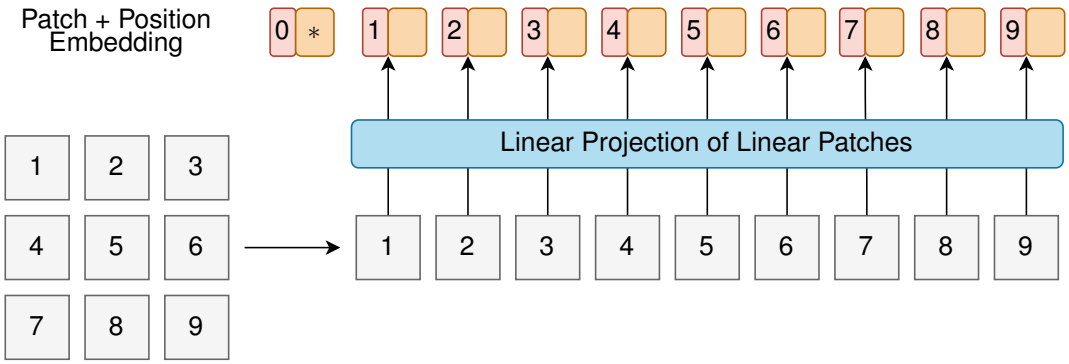

Figure 2.9: Preparation of the input image for the Vision Transformer. $*$ is the $[class]$ embedding.

Another modification is the removal of the transformer's decoder. The encoder provides multiple outputs, but only the learnable $[class]$ output is sent to a multiple layer perceptron head - the other outputs are ignored. The MLP head then provides the class it deems the image most likely corresponds to. Additionally, the encoder block layout was rearranged from the original transformer, as can be seen in Figure 2.10.

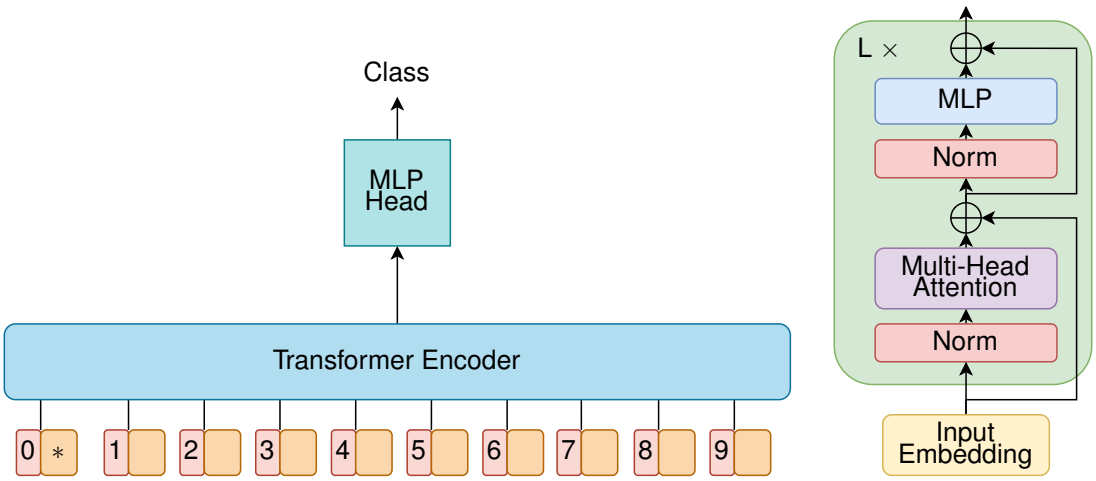

Figure 2.10: On the left, the structure of the ViT. On the right, the internal structure of the ViT encoder.

The ViT achieved or beat state-of-the-art results on various image classification datasets, but only when trained on very large datasets. Otherwise, the ViT underperformed even when compared to ResNets. The authors claim it's expected, since transformers lack the inductive biases inherent to convolutional neural networks, and therefore need more data to explicitly learn them.

2.4 Summary

This chapter reviewed the fundamental concepts of deep learning by introducing neural networks in Section 2.1. Fundamentals on convolutional neural networks and transformer architectures for computer vision were introduced in Section 2.2 and Section 2.3.

3

Related Work

Contents

3.1 Conventional Convolutional Neural Network Models for COVID-19 Detection 25

3.2 Recent Models Created or Adapted for COVID-19 Detection 31

3.3 Summary 42

This chapter explores related work in the area of image classification for diagnosing COVID-19 from chest X-ray images. The first section addresses studies using standard CNN architectures, and the second section covers studies with architectures that were created or that were adapted for this task.

3.1 Conventional Convolutional Neural Network Models for COVID-19 Detection

Since the beginning of the pandemic, there has been extensive work on techniques to automatically diagnose COVID-19 using CT scans and chest X-ray images. One of the approaches was to directly use known architectures for image classification, taking advantage of transfer learning (i.e., making use of networks trained on a task for which there is much data as the starting point of a model for a second, different task). We provide an overview of the studies by Apostolopoulos and Mpesiana (2020), Narin et al. (2021), and Minaee et al. (2020).

Apostolopoulos and Mpesiana (2020) used networks (Table 3.1) that were trained with the ImageNet dataset: VGGNet (Simonyan and Zisserman, 2015), Inception (Szegedy et al., 2016), Inception-ResNet (Szegedy et al., 2016), Xception (Chollet, 2017), and MobileNet (Howard et al., 2017).

The VGGNet architecture is a model with interspersed convolutional and max pooling layers, similar to the AlexNet from the previous section, but deeper and with small receptive fields (i.e. convolutional layers have kernels with small size, 3×3 or 1×1, enabling it to learn more complex features with a lower amount of parameters). In the case of VGG19, i.e. the variant chosen by the authors and represented in Figure 3.1, there are 19 weight layers (16 convolutional layers, and 3 fully-connected layers) and only 3×3 kernels are used.

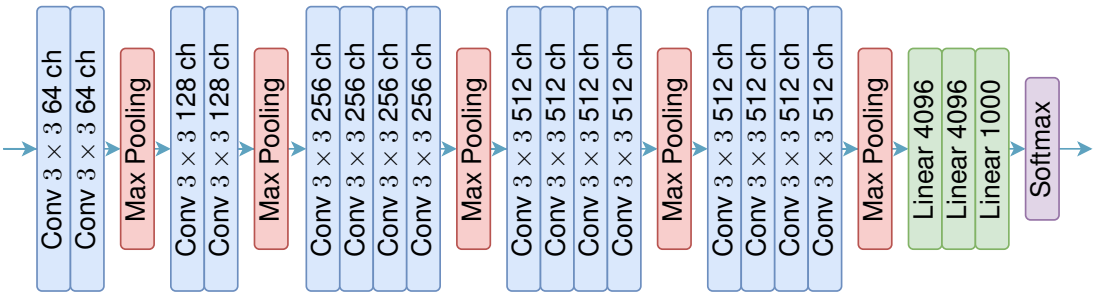

Figure 3.1: Structure of the VGG19 network.

The Inception model is a network built on the premise that the ideal kernel size depends on the distribution of information in an image, i.e. a larger kernel is preferred for information distributed more globally, while a smaller kernel is preferred for information distributed locally. The solution suggested by Szegedy et al. (2016) is to create the Inception module (Figure 3.2), where convolutions with multiple

filters of different sizes are performed on the same input, along with max pooling operations, and the output is concatenated and sent to the next module in the network. Additionally, 1×1 convolutions are added to perform dimensionality reduction. Further versions of the model exist, i.e. Inceptionv2, Inceptionv3, or Inceptionv4, with improvements to the model, such as refactorization of convolutions with larger kernels into multiple convolutions with smaller kernels.

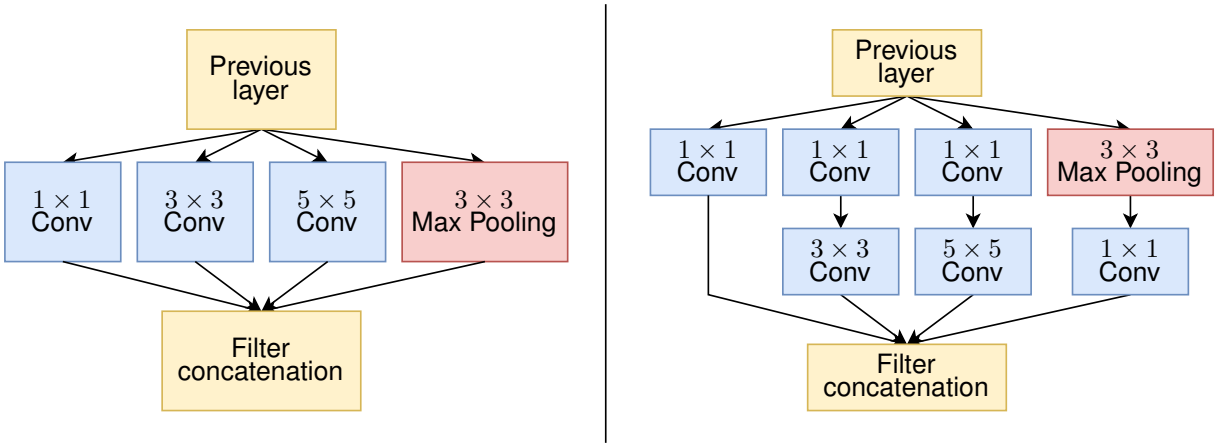

Figure 3.2: On the left, we can see the naive Inception module. On the right, we see the Inception module with dimensionality reductions (bottleneck), used in Inceptionv1.

The Inception-ResNet architecture is a model combining the structure of the Inception module with the residual connections of the ResNet, adding the input of the convolution operations to the output, as we can see in Figure 3.3. The variant used by the authors is the Inception-ResNetv2.

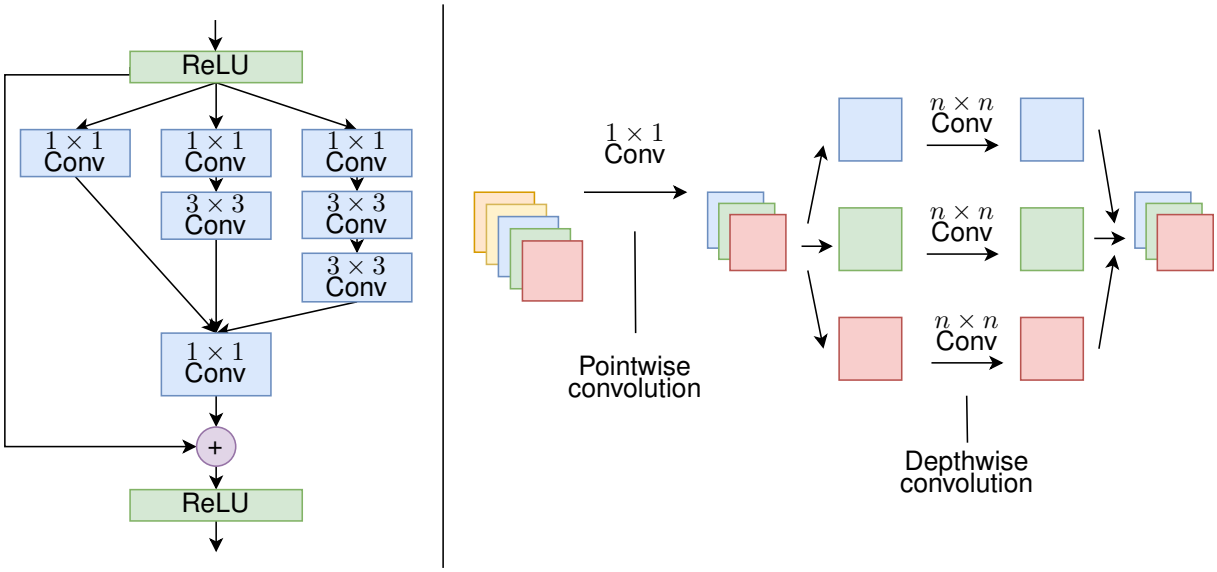

Figure 3.3: On the left, we see one of the building blocks of the Inception-ResNet architecture. Note the residual connection from the input (after ReLU activation) and the summation of the output of the convolutions and the input. On the right, we can see a representation of the convolution operation performed by the Xception and MobileNet architectures.

The Xception model applies convolutions differently from the previous models. Instead of performing convolutions with a filter along all channels and regions simultaneously, the Xception's convolution (represented in Figure 3.3) is done separately: first a pointwise convolution, using 1×1 filters and changing the number of feature maps, then a depthwise convolution, where the operation is performed between each channel in the kernel and the input, and the resulting feature maps are concatenated. This means that the number of connections is smaller, since it is not necessary to perform convolutions over all channels. Additionally, a few residual connections with 1×1 convolutions are added.

Finally, the MobileNet architecture is similar to the Xception architecture, utilizing separate depthwise convolutions as well, yet focused on ensuring accuracy while decreasing the number of parameters when compared to state-of-the-art architectures.

All of the previous CNNs use the ReLU as activation function. Moreover, all of the networks with two hidden layers had a dropout layer added to prevent overfitting, and training was conducted for 10 epochs, using 10-fold-cross-validation. Additionally, all networks have a set amount of untrainable layers (layer cutoff), maintaining the weights from the original model trained with the ImageNet dataset.

Network	Parameter	Description
VGG19	Layer Cutoff	18
	Neural Network	1024 nodes
Inception	Layer Cutoff	249
	Neural Network	1000
Inception-ResNetv2	Layer Cutoff	730
	Neural Network	None
Xception	Layer Cutoff	120
	Neural Network	1000 nodes, 750 nodes
MobileNet	Layer Cutoff	10
	Neural Network	1000 nodes, 750 nodes

Table 3.1: Table of parameters used in the networks evaluated by Apostolopoulos and Mpesiana (2020).

The dataset used by Apostolopoulos and Mpesiana is a collection of 1427 chest X-ray images from publicly available repositories (Cohen et al., 2020; Kermany et al., 2018), of which 224 were of COVID-19 confirmed cases, 700 images corresponded to confirmed pneumonia (non-COVID-19), and 504 images corresponded to normal conditions. The results were obtained for 3-class (COVID-19, pneumonia, normal) and 2-class (COVID-19, and non-COVID-19) predictions.

Network	Acc 2-class (%)	Acc 3-class (%)	Sensitivity (%)	Specificity (%)
VGG19	98.75	93.48	92.85	98.75
Inception	86.13	92.85	12.94	99.70
Inception-ResNetv2	84.38	92.85	0.01	99.83
Xception	85.57	92.85	0.08	99.99
MobileNet	97.40	92.85	99.10	97.09

Table 3.2: Table of results for the networks evaluated in Apostolopoulos and Mpesiana (2020).

CNN	TP	FP	TN	FN
VGG19	**208**	**15**	**1189**	16
MobileNet	222	35	1169	**2**

Table 3.3: Confusion matrix of the best-performing networks evaluated by Apostolopoulos and Mpesiana (2020).

The results obtained, shown in Table 3.2 and Table 3.3, in these conditions suggest that VGG19 and MobileNet achieve the best accuracy and specificity. However, the number of samples per class is uneven, with a smaller number of COVID-19 confirmed images, which skews these metrics, evident by the sensitivity presented by Inception, Xception and Inception-Resnetv2 networks, with inadequate sensitivity (i.e. ratio of true positives among all positive cases) for the task. The authors conclude that MobileNet outperforms VGG19, since it has similar accuracy and a lower number of false negatives, preventing a situation where a patient with COVID-19 would spread the virus further due to wrong diagnosis by the network. However, the dataset used in this study is small, compared to datasets such as the ImageNet dataset, and it ideally should include a larger proportion of COVID-19 confirmed images.

Ahmed et al. (2021) analyzed a few studies on COVID-19 chest X-ray classification, and concluded that the use of subsets of train/validation/test images with mixed sources, instead of performing training and testing using data from different sources, may lead to the high accuracy achieved in some studies, which indicates that the network may be learning based on confounders instead of actual COVID-19/pneumonia features.

Narin et al.'s work also evaluated CNNs with transfer learning for COVID-19 prediction. The CNNs used were ResNet50, ResNet101, ResNet152 (i.e. variants of ResNet with 50, 101, and 152 layers respectively), Inceptionv3 and Inception-ResNetv2, pre-trained with the ImageNet dataset, providing the input to a global average pooling layer, a fully-connected layer with ReLU as activation function, and a fully-connected layer classifying the input between 2 classes with the softmax function.

The dataset used by Narin et al. combined chest X-ray images from publicly available repositories (Cohen et al., 2020; Wang et al., 2017), with 341 images from COVID-19 patients, 2800 normal images, and 2772 bacterial and 1493 viral pneumonia images. The dataset was divided in 3 binary datasets: dataset 1 contained the COVID-19 and normal images, dataset 2 contained the COVID-19 and viral pneumonia images, and dataset 3 was composed of the COVID-19 and bacterial pneumonia images.

Model pre-training was performed with random initialization weights and optimizing the cross-entropy loss function with the ADAM optimizer (Kingma and Ba, 2017), i.e. an adaptive learning rate optimization algorithm that adapts the learning rate based on the second moment of the gradients (variance). The dataset was randomly split into two independent datasets, with 80% of the samples for traning and 20% of the samples for testing. The training dataset was used for k-fold cross-validation, with $1 \leq k \leq 5$.

Models	Dataset 1			
	Acc (%)	Rec (%)	Spe (%)	Pre (%)
Inceptionv3	95.4	90.6	96.0	73.4
ResNet50	96.1	91.8	96.6	76.5
ResNet101	96.1	78.3	98.2	84.2
ResNet152	93.9	65.4	97.3	74.8
Inception-ResNetv2	94.2	83.5	95.4	67.7

Models	Dataset 2			
	Acc (%)	Rec (%)	Spe (%)	Pre (%)
Inceptionv3	98.6	99.7	98.3	93.2
ResNet50	99.5	99.4	99.5	98.0
ResNet101	97.1	88.3	99.1	95.6
ResNet152	97.5	90.9	99.1	95.7
Inception-ResNetv2	94.4	92.1	94.9	80.5

Models	Dataset 3			
	Acc (%)	Rec(%)	Spe (%)	Pre (%)
Inceptionv3	97.7	100	97.4	82.4
ResNet50	99.7	98.8	99.8	98.3
ResNet101	94.7	52.5	99.9	98.9
ResNet152	92.8	51.0	98.0	75.5
Inception-ResNet v2	95.3	70.7	98.3	84.0

Table 3.4: Results obtained by Narin et al. (2021) with the 3 binary datasets: dataset 1 (COVID-19 and normal), dataset 2 (COVID-19 and viral pneumonia) and dataset 3 (COVID-19 and bacterial pneumonia).

The results obtained (Table 3.4) by Narin et al. show that ResNet50 has overall the best performance across all datasets, achieving the highest or one of the highest accuracies, while also achieving recall, specificity and precision on par with the other models. However, the datasets used in this study were again relatively small and unbalanced, compared to datasets like the ImageNet dataset, which is to be expected due to the nature of the problem: it is not easy to obtain anonymized labeled chest X-ray images, especially a relatively recent problem such as COVID-19.

The work from Narin et al. also performs training and testing on datasets from the same sources, which according to Ahmed et al. (2021) may mean that the network is possibly learning based on confounders instead of actual COVID-19/pneumonia features. Testing on a dataset that does not include

images from the dataset used for training would provide more insight on this matter.

The work by Minaee et al. analyzes the performance of ResNet18 (i.e., the 18 layer variant of ResNet), ResNet50 (as in Narin et al. (2021)), SqueezeNet, and DenseNet161 (i.e., the 161 layer variant of DenseNet) for COVID-19 detection on the COVID-Xray-5k dataset.

The SqueezeNet (Iandola et al., 2016) is an architecture designed to have fewer parameters, more easily fitting GPU memory, while maintaining good performance. It equals AlexNet performance while being much smaller in size, achieving 0.5MB in size using model compression techniques. The main building block of the SqueezeNet, represented in Figure 3.4, is a "fire" module, with a 1×1 convolution for dimensionality reduction, followed by 1×1 and 3×3 convolutions.

The COVID-Xray-5k dataset was created by the authors, with chest X-ray images (of both COVID-19 and non-COVID-19 patients) from a publicly available repository (Cohen et al., 2020) and re-labeled by an expert radiologist, and non-COVID-19 images from the ChexPert (Irvin et al., 2019) dataset. Due to the scarcity of COVID-19 image samples, data augmentation was performed, flipping, rotating, and distorting the images slightly. The images were split in 3100 for the test set (3000 non-COVID-19, 100 COVID-19) and 2084 for the training set (2000 non-COVID-19, and 84 COVID-19 samples, increased to 420 after augmentation).

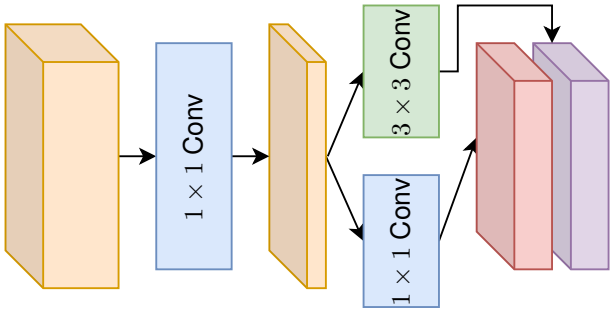

Figure 3.4: SqueezeNet's fire module.

All of the models were trained using the cross-entropy loss function and using the ADAM optimizer. Only the last layer of the networks was fine-tuned for the task, with the previous layers maintaining the weights from training on the ImageNet dataset, since Minaee et al. considered that the number of COVID-19 samples was too limited to fine-tune all of the layers.

Due to the imbalance in COVID-19 and non-COVID-19 samples, the authors decided to compare the probability score predicted by the network with a variable threshold to estimate the specificity and sensitivity of the networks. Other evaluation metrics used were derived from the Precision-Recall curve (average precision) or from the Receiver Operating Characteristic curve (AUC, i.e. area under the ROC curve).

Model	Sensitivity	Specificity	Average Precision	AUC
ResNet18	$98\% \pm 2.7\%$	$90.7\% \pm 1.1\%$	0.869	0.989
ResNet50	$98\% \pm 2.7\%$	$89.6\% \pm 1.1\%$	0.899	0.990
SqueezeNet	$98\% \pm 2.7\%$	$92.9\% \pm 0.9\%$	0.897	0.992
DenseNet161	$98\% \pm 2.7\%$	$75.1\% \pm 1.5\%$	0.863	0.976

Table 3.5: Results obtained in Minaee et al. (2020).

As seen on Table 3.5, at certain thresholds, all of the networks have similar sensitivity, being apparently able to correctly identify patients with COVID-19. A larger variation can be seen in specificity, with SqueezeNet achieving the best specificity, and with ResNet18 as the runner-up. However, since the sensitivity and specificity metrics depend on specific thresholds, the average precision and AUC provide a broader perspective of general performance. Both ResNet18 and SqueezeNet have high average precision and AUC as well, and were considered by the authors as the two top-performing networks of the study.

Additionally, the authors employed a technique from Zeiler and Fergus (2013) to visualize the results of the prediction. More specifically, this method is based on classifying COVID-19 samples with an $N \times N$ square of pixels being occluded, starting from the top-left corner of the image, and comparing the result obtained with the actual label. If an image is no longer identified as COVID-19 positive, it is considered that the occluded pixels belong to a region with important characteristics for the network to detect COVID-19. Otherwise, the region is considered irrelevant for COVID-19 detection. Minaee et al. compared the results from the visualization with the regions marked by the expert radiologist, and claimed that the heatmaps generated by this technique agreed with the marked regions. However, no metric was provided to evaluate how accurate the heatmaps were when compared to the marked regions.

Minaee et al.'s study also suffers from the same problem as the two previous studies, as explained in Ahmed et al. (2021). Further testing with a dataset with images from different sources would allow to check the generalization ability of these models when faced with unseen data.

3.2 Recent Models Created or Adapted for COVID-19 Detection

This section provides an overview and analysis on some studies that have created or adapted existing models specifically for the task of diagnosing COVID-19.

One the earliest models tailored for this task was the COVID-Net from Wang and Wong (2020), i.e. a deep CNN that classifies X-rays into 3 classes: no infection, non-COVID-19 infection, and COVID-19 infection.

The COVID-Net architecture makes use of the PEPx module, as seen in Figure 3.5, composed

of 2 layers of 1×1 convolutions, a layer with a depthwise 3×3 convolution, and two other layers of 1×1 convolutions. The 1×1 convolutions are used to reduce and expand the input and output feature maps. The authors argue that this module allows for great representational capacity while being computationally efficient.

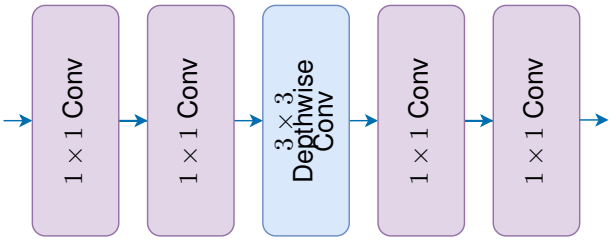

Figure 3.5: COVID-Net's PEPx module from Wang and Wong (2020).

This module is incorporated in a deep CNN with several connections between the PEPx modules, similar to the residual/dense connections in Section 2.2. The module and the architecture (Figure 3.6) were obtained by machine-driven exploration, starting with an initial prototype and using generative synthesis (Wong et al., 2018) to identify the optimal architecture. The requirements for the network related to achieving a sensitivity of at least 80% and a positive predictive value of at least 80% for COVID-19 positive cases. The rationale used is that these requirements allow for the correct prediction of COVID-19 cases while reducing the probability of false positives.

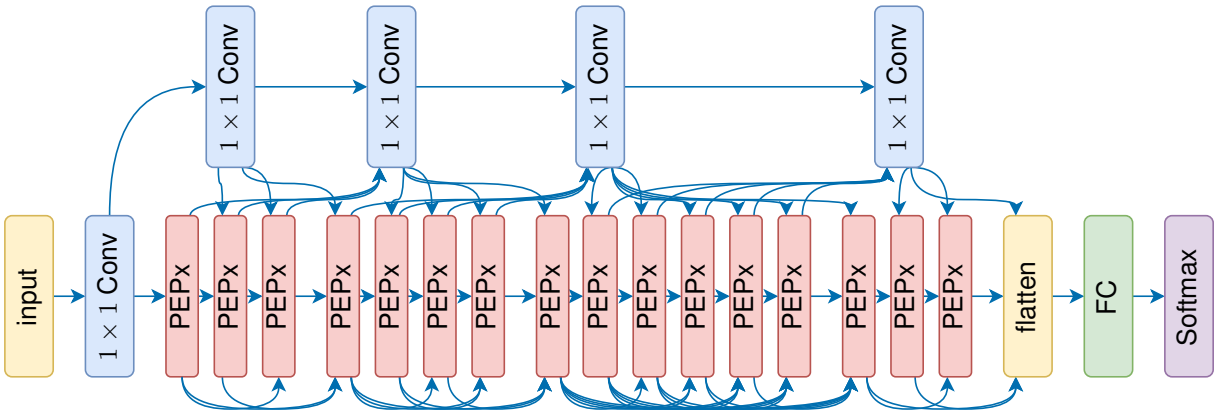

Figure 3.6: The COVID-Net architecture from Wang and Wong (2020).

The authors observed that the architecture has long-range connectivity and is heterogeneous, with diverse kernel parameters. Long-range connections improve the network's representational capacity and make it easier to train. However, they note that extensive DenseNet-like connectivity increases computational complexity and memory overhead. This architecture solves this problem by using the layer's results selectively, leveraging the pros and cons of long-range connections. Wong et al. (2018) note that these characteristics are the result of the machine-driven exploration optimizing the design of

the network for the task at hand.

For training, the COVID-Net was first pre-trained on the ImageNet dataset, followed by training on the COVIDx dataset using the ADAM optimizer. The COVIDx dataset was created by the authors, by compiling 5 publicly available COVID-19 datasets. It is comprised of 13,975 CXR images (at that time), with 358 CXR images from COVID-19 patients. Furthermore, data augmentation was performed using translation, rotation, horizontal flip, zoom, and intensity shift (i.e. shifting the color intensity) operations, and the training batches are balanced to encourage the presence of images of different infections in each batch.

Furthermore, the authors tried to address the problem of transparency and explainability of the results. In critical tasks such as medical applications, it is important to understand how neural networks have obtained the results, and guarantee that the network is classifying the images using relevant features. For that purpose, the authors used GSInquire (Lin et al., 2019), i.e. an explainability method that is essential for the machine-driven exploration: an inquisitor ($\mathscr{I}$) and a generator ($\mathscr{G}$) pair work together, with $\mathscr{G}$ generating new networks, that $\mathscr{I}$ analyzes. The information obtained by $\mathscr{I}$ is used to improve $\mathscr{G}$ to generate better networks, as well as to create an interpretation, that can in this case be visualized relative to the X-ray image.

To analyze the performance of the COVID-Net architecture, the authors compared its performance with VGG19 and ResNet-50 networks on the COVIDx test dataset, computing accuracy, sensitivity, and PPV. The ResNet-50 network is a variant of the ResNet, that is 50 layers deep and that utilizes the bottleneck residual block.

Architecture	**Parameters (M)**	**Acc. (%)**	**Sensitivity (%)**		
			Normal	**Non-COVID-19**	**COVID-19**
VGG19	20.37	83.0	98.0	90.0	58.7
ResNet-50	24.97	90.6	97.0	92.0	83.0
COVID-Net	11.75	93.3	95.0	94.0	91.0

Table 3.6: Number of parameters of each architecture and sensitivity per class in Wang and Wong (2020).

Architecture	**PPV (%)**		
	Normal	**Non-COVID-19**	**COVID-19**
VGG19	83.1	75.0	98.4
ResNet-50	88.2	86.8	98.8
COVID-Net	90.5	91.3	98.9

Table 3.7: Positive prediction value (PPV) per class for each architecture tested by Wang and Wong (2020).

The results obtained (Table 3.6 and Table 3.7) show that COVID-Net has less parameters and better accuracy than the other two networks. The author's proposal also has better sensitivity for non-COVID-

19 and COVID-19 cases, and better PPV in all cases, while VGG19 wins in sensitivity in normal cases. Sensitivity for COVID-Net is superior to 90% in all cases, showing that it can detect many true positives, and PPV for COVID-19 cases is 98.9%, indicating that very few COVID-19 false positives should be detected.

It should nonetheless be noted that Ahmed et al. analyzed the datasets used on a variety of networks, and observed that Wang and Wong (2020) used a pediatric dataset for the normal/pneumonia class samples, while the average age of COVID-19 class patients were over 40 years old. This could mean that the network could be using age specific features to differentiate COVID-19 samples from other samples. Removing the pediatric dataset from tests could show more truthful results regarding the network's ability to classify COVID-19 samples, along with using a separate dataset to test the network's ability to generalize. Notably, the dataset used is heavily unbalanced, but since the network was generated by requiring a sensitivity value of at least 80%, the effect in this metric is not noticeable like in Apostolopoulos and Mpesiana (2020).

Another architecture created for COVID-19 chest X-ray classification is CoroNet (Khan et al., 2020), i.e. a CNN based on Xception (Chollet, 2017) with modifications. The first part of the model is Xception itself, 71 layers deep, with an output shape of $5 \times 5 \times 2048$. This output is provided to a flatten layer that will output a single vector to a dropout layer, followed by two fully-connected layers, the last of which providing the prediction of the image's class using the softmax function.

The authors implemented a main multi-class model to classify images into COVID-19, normal, bacterial pneumonia, and viral pneumonia, as well as 2 modified models: one to classify images into COVID-19, normal, and pneumonia, and the other to classify into COVID-19 and normal.

The CoroNet was first pre-trained on the ImageNet dataset, to avoid overfitting due to the small size of the training dataset. The dataset used for fine-tuning the network is a combination of two publicly available datasets, one of them being (Cohen et al., 2020). At the time, this dataset contained 290 COVID-19 chest X-ray images, 1203 normal chest X-ray images, 660 bacterial pneumonia samples, and 931 viral pneumonia images. Due to the unbalanced number of samples from each class (especially the COVID-19 samples), random under-sampling was performed, i.e. examples of the majority class were deleted until the dataset is balanced. After under-sampling, the dataset contained 310 normal, 330 bacterial pneumonia, and 327 viral pneumonia images, along with the COVID-19 samples. The retraining of the network on the new dataset was performed using the ADAM optimizer with data shuffling enabled, and 4-fold cross-validation was performed.

As we can see in Table 3.8 and Table 3.9, the performance for non-COVID-19 pneumonia classes is lower than COVID-19 and normal classes, leading to a lower accuracy overall. However, when combining the two pneumonia classes into one for the 3-class CoroNet, the accuracy, recall, and F-measure increase significantly. There's a further smaller improvement for binary CoroNet.

Class	Precision (%)	Recall (%)	Specificity (%)	F-measure (%)
COVID-19	93.17	98.25	97.9	95.61
Normal	95.25	93.5	98.1	94.3
Bacterial Pneumonia	86.85	85.9	95	86.3
Viral Pneumonia	84.1	82.1	94.8	83.1

Table 3.8: Results for each class for the 4-class CoroNet model from Khan et al. (2020).

Model	Precision (%)	Recall (%)	Specificity (%)	F-measure (%)	Accuracy (%)
4-class CoroNet	90.0	89.92	96.4	89.8	89.6
3-class CoroNet	95.0	96.9	97.5	95.6	95.0
Binary CoroNet	98.3	99.3	98.6	98.5	99.0

Table 3.9: Results for each CoroNet model from Khan et al. (2020).

The high recall obtained in this study indicates that a low number of false negatives are produced, which means that a low number of COVID-19 cases are missed by the model. The authors also tested the 3-class model on a different dataset from Ozturk et al. (2020), with the same COVID-19 samples but different normal and pneumonia X-ray images, achieving an overall accuracy of 90% after fine-tuning.

Khan et al. claim that CoroNet's results are superior compared to results obtained in other studies, by comparing accuracy to other models. However, since the studies mentioned are not tested in the same dataset as CoroNet, further studies comparing the networks with the same dataset/similar conditions would be necessary to confirm this statement. Moreover, comparisons using not only accuracy but other metrics as well (such as sensitivity and sensibility) would provide further insight. A comparison with a baseline (such as ResNet or DenseNet) would also have been a good addition to the study.

More recently, Mamalakis et al. (2021) combined the ResNet and DenseNet architectures into a pipeline to diagnose whether a patient has COVID-19, tuberculosis, pneumonia, or is healthy. The authors analyzed the performance of DenseNet-121, ResNet-50 and VGG16 on a CT scan dataset, and verified that although ResNet-50 had better recall, precision, and F1, DenseNet-121 has a better AUC-ROC. They than hypothesized that the combination of these two networks could result into an accurate, balanced network.

The resulting architecture (Figure 3.7) contains 4 layers with equal sized kernels from each network, which have their output concatenated and fed to blocks of convolution and average pooling layers, until their outputs all have equal dimensions. The results of the redimensioning are then concatenated, and provided as input to fully-connected layers with dropout in between. The final layer uses the softmax function for classification.

The authors used three publicly available open-source collections of chest X-ray images (Kermany et al., 2018; Cohen et al., 2020; Jaeger et al., 2014) to create 4 different datasets, DXR1, DXR2, DXR3, and DXR4, as shown in Table 3.10. DXR1 is a dataset with pneumonia and healthy cases in pediatric

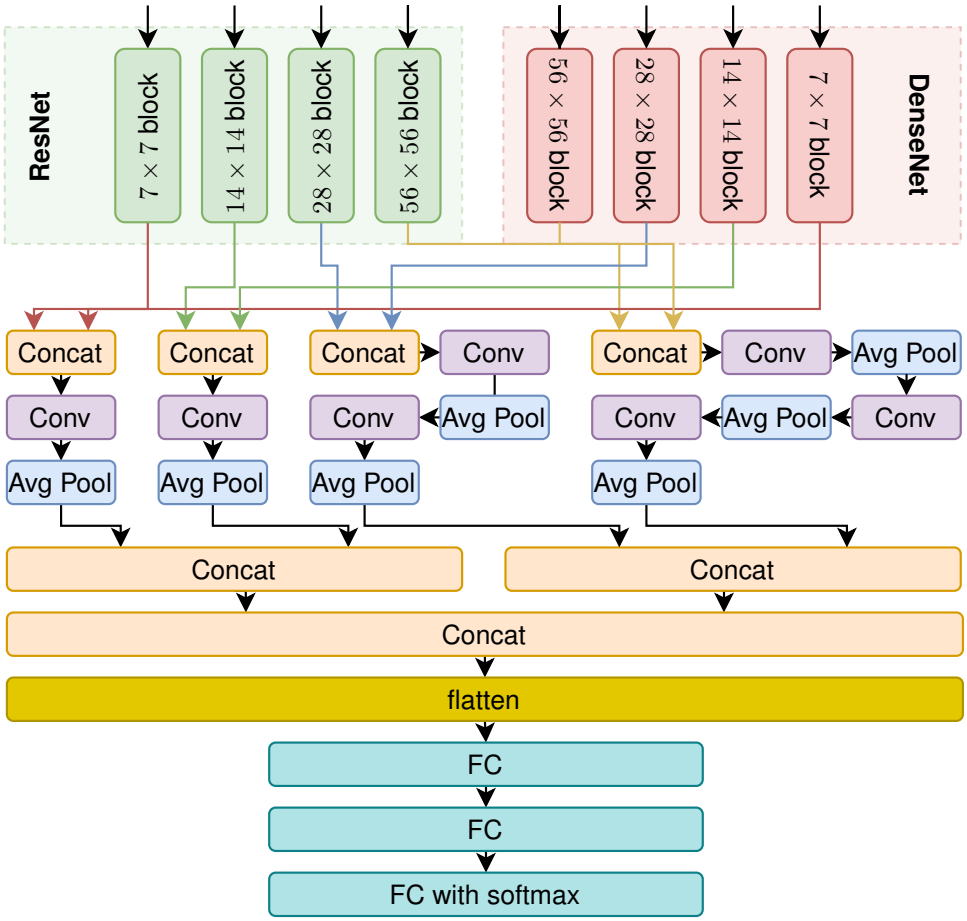

Figure 3.7: The DenResCov-19 architecture from Mamalakis et al. (2021).

population from Kermany et al. (2018), while DXR2 is the dataset with cases of COVID-19, pneumonia, and healthy cases from Cohen et al. (2020). DXR3 is a dataset with images from Cohen et al. (2020) and Jaeger et al. (2014), showing tuberculosis, COVID-19, pneumonia and healthy cases. Since Cohen et al. (2020) used a small dataset compared with Jaeger et al. (2014), a few images were randomly selected from the latter, to make sure DXR3 was balanced. Finally, DXR4 is an extension of the DXR3 dataset, with tuberculosis, COVID-19, pneumonia, and healthy cases from all collections. In this dataset, there's some unbalance in the frequency distribution, due to the COVID-19 dataset only having 69 CXR samples.

Mamalakis et al. performed pre-processing on the images, removing noise and normalizing them by subtracting the mean value from each pixel in the image and dividing the pixel values by the standard deviation. Additionally, the images underwent data augmentation techniques such as rotation, width shift, height shift, and ZCA whitening (Koivunen and Kostinski, 1999), i.e. transforming the pixel data so that the covariance matrix becomes the identity matrix. The datasets were split into the training and test set with Monte Carlo cross-validation split, i.e. the images were split randomly between the test and

Dataset	Healthy	Pneumonia	Tuberculosis	COVID-19
DXR1	1350	3883	-	-
DXR2	79	79	-	69
DXR3	79	79	79	69
DXR4	330	300	310	69

Table 3.10: Data in each dataset used by Mamalakis et al. (2021).

training sets multiple times, with 70% of the samples in the training set, and 30% of the images in test set. The loss function applied was cross-entropy, and the DenseNet-121 and ResNet-50 networks were pre-trained on the ImageNet dataset.

Metric	DXR1 dataset			
	DenResCov-19	**DenseNet-121**	**ResNet-50**	**Inceptionv3**
Recall (%)	98.12	97.80	97.71	93.32
Precision (%)	98.31	94.62	95.01	90.10
AUC-ROC (%)	99.60	99.10	98.95	92.80
F1 (%)	98.21	96.27	96.34	91.68

Metric	DXR2 dataset			
	DenResCov-19	**DenseNet-121**	**ResNet-50**	**VGG16**
Recall (%)	89.38	83.54	83.53	99.83
Precision (%)	85.28	77.45	73.35	33.38
AUC-ROC (%)	96.51	93.2	92.39	50.07
F1 (%)	87.29	80.37	78.11	49.51

Metric	DXR3 dataset			
	DenResCov-19	**DenseNet-121**	**ResNet-50**	**VGG16**
Recall (%)	59.28	57.71	56.66	66.53
Precision (%)	79.56	74.87	74.00	26.53
AUC-ROC (%)	91.77	89.49	92.12	53.11
F1 (%)	68.09	65.17	64.17	38.00

Metric	DXR4 dataset			
	DenResCov-19	**DenseNet-121**	**ResNet-50**	**VGG16**
Recall (%)	69.70	62.70	62.00	93.69
Precision (%)	82.90	79.35	78.60	27.17
AUC-ROC (%)	95.00	91.00	93.21	54.99
F1 (%)	75.75	70.07	69.51	42.13

Table 3.11: Results for each dataset used by Mamalakis et al. (2021).

The obtained results (Table 3.11) show that across all datasets, DenResCov-19 achieves better precision, AUC-ROC and F1 than DenseNet-121, ResNet-50 and VGG16/Inceptionv3. In DXR1, DenResCov-

19 achieves better recall, although for the other datasets VGG16 obtains a higher recall. Note that performance decreases as the number of classes increase, and that it improves from DXR3 to DXR4, likely due to the larger dataset.

A more recent work in COVID-19 chest X-ray classification is Le Dinh et al. (2022) which leverages 5 neural networks for differentiating samples of COVID-19, pneumonia, and healthy patients, i.e. ResNet50 (He et al., 2015), DenseNet121 (Huang et al., 2018), Inception (Szegedy et al., 2015), Swin Transformer (Liu et al., 2021), and Hybrid EfficientNet-DOLG (Henkel, 2021).

The Swin Transformer (Liu et al., 2021) is a transformer architecture based on the original ViT (Vaswani et al., 2017). The authors sought to solve some issues of the original transformer architecture. One of those issues relates to the fact that objects in images can vary greatly in size. In ViT, the patches are all of the same size, which means that objects of differing sizes may not be captured properly - a large object can be split in multiple patches, or an object of a small number of pixels may be in a very large patch. Another issue that arises is that the ViT's self-attention calculation mechanisms are of quadratic complexity when compared to the size of the image, which is especially problematic when the resolution of the different images increases.

The Swin transformer (Liu et al., 2021) attempts to solve these issues by dividing the image in small patches, and gradually merging neighbouring patches along the transformer layers. It also changes the blocks in the encoder block to calculate the self-attention in a less computationally intensive way, reducing the complexity from quadratic to linear on the image size.

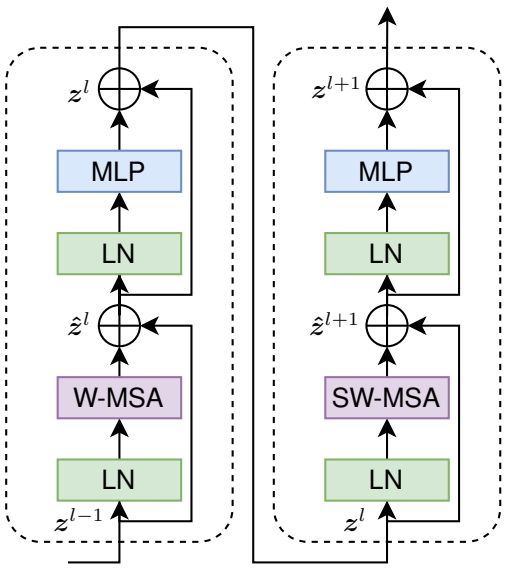

Figure 3.8: Swin Transformer block pair.

The transformer block of the original ViT is modified by changing the Multi-Head Self-Attention module to a Shifted-Window based Self-Attention. In practice, the self-attention of a patch is not calculated

over all patches of the image, but only the neighbouring patches within a window. These windows are arranged in a non-overlapping manner over all the patches. Additionally, in order to establish connections across windows while still maintaining the complexity improvements, at each stage there's pairs of Swin Transformer blocks, where the second block in a pair calculates the self-attention in a shifted window from the first block in the pair.

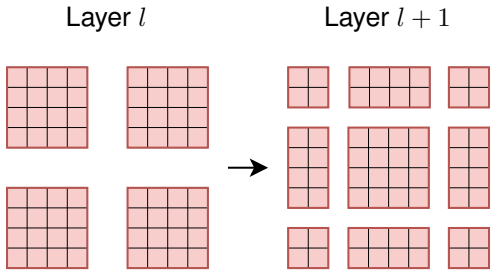

Figure 3.9: Ilustration of Swin Transformer's shifted window.

Assuming the window size is set to $M = 4$, and an image has been divided in 8×8 patches, then in the first block the image will have 4 windows, starting from the top left of the image. When the output of the first block is fed to second block, the windows are displaced by $\left(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor\right)$. With this approach, the computations of consecutive Swin Transformer blocks are

$$\hat{\boldsymbol{z}}^l = \text{W-MSA}(\text{LN}(\boldsymbol{z}^{l-1})) + \boldsymbol{z}^{l-1} \tag{3.1}$$

$$\boldsymbol{z}^l = \text{MLP}(\text{LN}(\hat{\boldsymbol{z}}^l)) + \hat{\boldsymbol{z}}^l \tag{3.2}$$

$$\hat{\boldsymbol{z}}^{l+1} = \text{SW-MSA}(\text{LN}(\boldsymbol{z}^l)) + \boldsymbol{z}^l \tag{3.3}$$

$$\boldsymbol{z}^{l+1} = \text{MLP}(\text{LN}(\hat{\boldsymbol{z}}^{l+1})) + \hat{\boldsymbol{z}}^{l+1} \tag{3.4}$$

where LN is the linear normalization and MLP is the multilayer perceptron - also present in the ViT - W-MSA and SW-MSA are the modified multi-head attention units, and $\hat{\boldsymbol{z}}^l$ $\boldsymbol{z}^l$ are the outputs of the MSA unit and MLP of block l, respectively.

Additionally, the model performs patch merging after each stage. Before the first stage, the size of input is $\frac{H}{4} \times \frac{W}{4} \times 48$, where H is the height of the original input image, W is the width, and 48 is the number of pixel values of each 4×4 patch, assuming an RGB image, after which each of the patches is converted into a 1-dimensional vector with linear embedding. After the input goes through the Swin Transformer blocks, it is in the format $\frac{H}{4} \times \frac{W}{4} \times C$, where C is the size of the linear embedding. Then

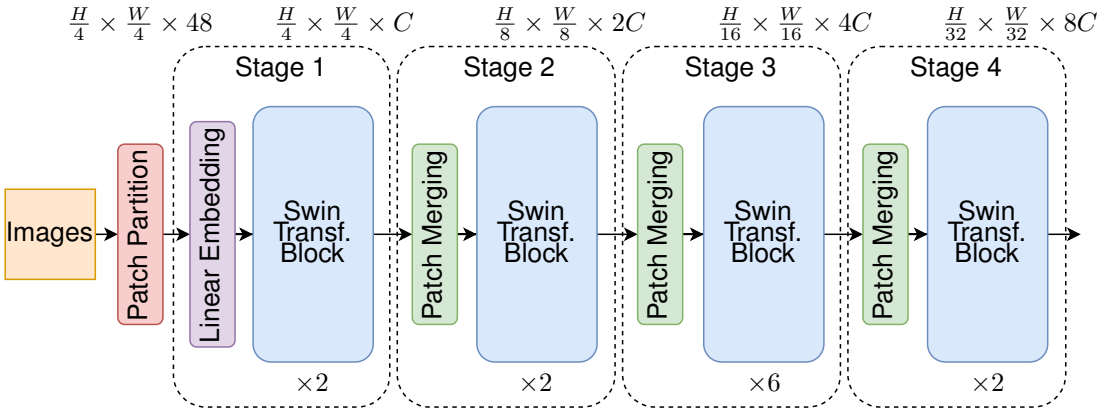

Figure 3.10: Swin Transformer Architecture (Swin-T variant).

the first patch merging layer concatenates the features of each group of 2×2 neighbouring patches, increasing the dimensions to $\frac{H}{8} \times \frac{W}{8} \times 4C$, and a linear layer scales the dimension down to $\frac{H}{8} \times \frac{W}{8} \times 2C$. As the input goes through the model, the patch merging layers scale down the first two dimensions by a factor or 2 and increase the third one also by a factor of 2.

After all of the stages, average pooling followed by normalization convert the output in a representation with $C \times 2 \times 2 \times 2$ embeddings, which are the input of a classifier that will output the correct class of the input image.

Some other modifications made on the Swin Transformer are the addition of a relative position bias to each self-attention head, and an optimization when calculating the attention in a shifted window in situations where there are windows that are smaller than the expected size - instead of adding a padding to the windows, the patches in the windows are re-arranged, so the total number of windows remains the same.

As for the Hybrid EfficientNet-DOLG (Henkel, 2021), it is an architecture that combines the EfficientNet and DOLG architectures.

The EfficientNet (Tan and Le, 2019) is an improvement on CNNs based on the concept that increasing the accuracy of a CNN involves arbitrarily scaling the network in a specific dimension, i.e. in depth, width, resolution. Tan and Le introduced a compound scaling method to uniformly scale all dimensions of a model

$$\text{depth: } d = \alpha^{\phi} \tag{3.5}$$

$$\text{width: } w = \beta^{\phi} \tag{3.6}$$

$$\text{resolution: } d = \gamma^{\phi} \tag{3.7}$$

$$\text{so that } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \tag{3.8}$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \tag{3.9}$$

where ϕ is the scaling coefficient that represents the amount of resources available for scaling, and α, β and γ are hyper-parameters for each scaling dimension, which can be found by performing a grid search. The resource considered for scaling was the number of FLOPS, i.e. Floating Point Operations Per Second.

The authors started with a baseline model called EfficientNet, generated using the AutoML MNAS framework. Once the hyper-parameters are found for $\phi = 1$, the hyper-parameters can be fixed, and the scaling coefficient is increased, creating different scales of EfficientNet, from the B0 to the B7 model. The largest model, EfficientNet-B7, achieved state-of-the-art results on the ImageNet dataset.

The DOLG architecture (Yang et al., 2021), i.e. Deep Orthogonal Local and Global framework, was originally created for the task of image retrieval. It is built upon the ResNet architecture, and feeds the output of the first 3 stages of the ResNet to a global and local branch. The local branch is in charge of capturing local features, with multi-atrous convolutions and self-attention. The global branch leverages the fourth stage of the ResNet. The outputs of the branches are fed to an orthogonal fusion module, which calculates the projection of local features onto global features.

Henkel (2021) created a hybrid of these two models, the Hybrid EfficientNet-DOLG, for landmark recognition and retrieval, which leverages the structure of the DOLG, and replaces the ResNet model with an EfficientNet pre-trained on the ImageNet dataset.

The datasets used by Le Dinh et al. combined chest X-ray images from two publicly available repositories (Wang and Wong, 2020; Kermany et al., 2018). The first dataset is COVIDx CXR-3, an updated version of COVIDx dataset, which contains over 30,000 images. The authors combined the two datasets, and filtered out some COVID-19 samples from the COVIDx CXR-3 to balance the dataset, followed by removal of low-quality images and application of data augmentation techniques such as rotation, horizontal flip, brightness change, re-scale, height and width shift.

As for training, the authors employed early stopping with 120 as the maximum number of epochs, the ADAM optimizer, and Sparse Categorical Cross Entropy, a variant of the cross entropy loss function.

As can be seen in Table 3.12, Swin Transformer has the best precision for the COVID-19 class, Hybrid EfficientNet-DOLG has the best precision for Pneumonia, while for the Normal class DenseNet121 has

Models	Precision				
	COVID-19	Normal	Pneumonia	Macro-Avg	Micro-Avg
DenseNet121	0.98	0.91	0.94	0.94	0.95
ResNet50	0.98	0.83	0.93	0.92	0.94
Inception	0.98	0.83	0.90	0.90	0.92
Swin Transformer	0.99	0.62	0.89	0.83	0.87
Hybrid EN-DOLG	0.98	0.93	0.93	0.95	0.96

Models	Recall				
	COVID-19	Normal	Pneumonia	Macro-Avg	Micro-Avg
DenseNet121	0.98	0.94	0.92	0.95	0.95
ResNet50	0.94	0.96	0.89	0.93	0.93
Inception	0.93	0.88	0.94	0.92	0.92
Swin Transformer	0.70	0.97	0.90	0.86	0.82
Hybrid EN-DOLG	0.97	0.94	0.95	0.96	0.96

Models	F1-score				
	COVID-19	Normal	Pneumonia	Macro-Avg	Micro-Avg
DenseNet121	0.98	0.95	0.93	0.94	0.95
ResNet50	0.96	0.89	0.91	0.92	0.93
Inception	0.96	0.85	0.92	0.91	0.92
Swin Transformer	0.82	0.75	0.90	0.82	0.82
Hybrid EN-DOLG	0.99	0.94	0.94	0.95	0.96

Table 3.12: Results obtained in Le Dinh et al. (2022)

.

the best precision. The Hybrid EfficientNet-DOLG leads in both macro and micro-averaged precision.

When it comes to recall, the Hybrid EfficientNet-DOLG has the best scores for three categories: Pneumonia, macro and micro-average. For the COVID-19 class, the best performing model was the DenseNet-121, while Swin Transformer obtained the best recall for the Normal class.

Finally, for the third metric, F1-score, the hybrid EfficientNet-DOLG achieved the best results in the Pneumonia, macro and micro-average categories once more, as well as the best score for the COVID-19 class, while DenseNet121 achieved the best F1-score for the Normal class.

3.3 Summary

This chapter reviewed the related work on COVID-19 image classification, namely the usage of conventional convolutional neural networks in Section 3.1 and the creation or adaptation of new models for the task in Section 3.2.

4

Model Architectures and Baselines

Contents

4.1 Models and Training Methods 45

4.2 Dataset and Metrics Used 47

4.3 Summary 51

The goal of this thesis is to analyze chest X-ray images, and discern if they correspond to a healthy patient, a positive COVID-19 diagnosis, or a positive pneumonia diagnosis. For this purpose we evaluate the performance of some neural network architectures, and compare them to baselines to better assess if they present any improvements.

The neural networks we evaluate are the Swin Transformer (Liu et al., 2021), and a convolutional neural network with improvements based on characteristics presented by vision transformers. We expect that using more recent architectures and developments in computer vision, the results obtained will also be an improvement when compared to the baselines.

4.1 Models and Training Methods

This section provides an overview on the baselines and the models explored, as well as the training methods employed.

4.1.1 Baselines

When creating baselines, we chose conventional neural network models used in other studies for COVID-19 detection, namely ResNet, VGGNet, and DenseNet.

The ResNet variant we use is ResNet50, which consists of 4 stages, each with 3, 4, 6, 3 residual bottleneck blocks respectively. Each residual bottleneck block has a residual connection from before to after the block, and the last residual block of each stage performs a convolutional operation with stride 2, reducing the size of the input in half. The stem cell, i.e. the first operations in the model that define how the input will be processed from then on, reduces the input size and has a 7×7 convolution layer with stride 2, followed by a pooling operation.

The VGGNet variant we use is VGG16, which has 3 convolutional layers instead of 4 convolutional layers between pooling operations, thus having 3 less convolutional layers in total than the VGG19 suggested by Simonyan and Zisserman (2015).

The DenseNet variant we use is DenseNet121, which is composed of 4 stages, each with a defined number of dense blocks (6, 12, 24, and 16, respectively), and followed by a transition layer with a 1×1 convolution and 2×2 average pool with stride 2 for downsampling, except for the last sequence of dense blocks, which is followed by the classification layer.

The baseline models used ReLU as the activation function, softmax for classification and cross-entropy as loss function, and ADAM as optimizer.

4.1.2 Swin Transformer

The first model we experimented on is the Swin Transformer by Liu et al., introduced in Chapter 3.

The authors present multiple variations of the architecture, and the one we will be using is Swin-T, with $C = 96$, and where each of the 4 stages has $[2, 2, 6, 2]$ Swin transformer blocks, the patches are 4×4 pixels and the windows span 7×7 patches.

Similarly to the baseline models, the transformer used softmax for classification and cross-entropy as loss function. However, instead of ReLU, the activation function used is GELU (Hendrycks and Gimpel, 2016), an activation function based on the cumulative distribution function of normal distribution, and the optimizer is ADAMW (Loshchilov and Hutter, 2017), a modified version of the ADAM optimizer that decouples the regularization in the ADAM optimizer from the step calculations.

4.1.3 ConvNeXT

The ConvNeXT (Liu et al., 2022) is a model created with a ResNet (i. e. ResNet50) as starting point. Then the authors attempt to make modifications to the original ResNet to obtain better results than both previous convolutional networks and vision transformers like Swin, coupling the modern approaches of the vision transformers with the natural advantage of convolutional neural networks' inductive biases.

The first modification the authors made to the model itself was alter the block ratio in each stage from (3, 4, 6, 3) to (3, 3, 9, 3). Then the ResNet stem cell was replaced by a "patchify" layer with a size 4×4 kernel with non-overlapping convolution (stride 4), simulating the windows in Swin. Additionally, the convolution operations in the residual blocks were altered to depthwise convolutions, similar to the self-attention operations which are also calculated per channel. Along with that, to further simulate the windows in Swin, which are of size 7×7, the kernel size of the convolutions in the residual blocks is changed from 3×3 to 7×7

Another important change to the residual blocks are inverted bottlenecks. In the transformer block, the hidden dimension of the MLP is 4 times wider than the input dimension.

As such the new residual block is composed of 3 layers: a 7×7 depthwise convolutional layer, and two 1×1 convolutional layers.

The remaining changes in the residual blocks are the usage of less normalization and activation functions, leaving only the normalization between the depthwise and 1×1 layer of a block and the activation function between the two 1×1 layers, as well as the usage of different normalization/activation functions altogether: while the original ResNet architecture uses Batch Normalization and ReLU, in the ConvNeXT these are replaced by Layer Normalization and the smoother GELU activation function.

The last changes performed by the authors are the addition of a 2×2 convolution downsampling layer with stride 2 between stages, simulating the dimension halving after each Swin transformer stage.

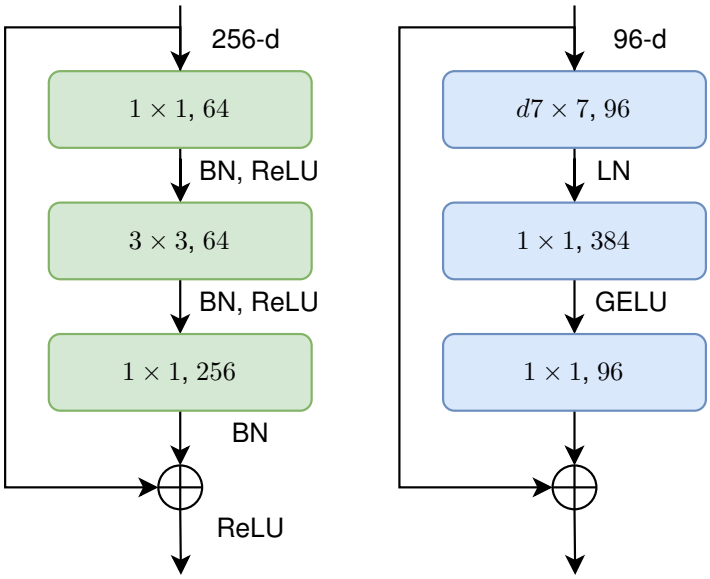

Figure 4.1: On the left, the ResNet bottleneck block. On the right, the ConvNeXT inverted bottleneck block.

Similarly to the Swin Transformer, the authors of ConvNeXT also present multiple versions of the model, and the one we will evaluate is ConvNeXT-T, with $C = (96, 192, 384, 768)$, and where each of the 4 stages has $3, 3, 9, 3$ residual blocks.

As in the Swin Transformer model, we used softmax for classification and cross-entropy as loss function. Additionally, we also used GELU as the activation function and ADAMW optimizer.

4.1.4 Training Methods

All of the models were pretrained on the ImageNet dataset, and fine-tuned with the COVIDx CXR-3 dataset (Wang and Wong, 2020). Fine-tuning was performed for all models for 20 epochs.

Additionally, we used data augmentation and pre-processing methods, namely rotation, horizontal flip, random crop, and normalization.

4.2 Dataset and Metrics Used

This section presents the dataset and modifications leveraged for training and testing the baselines and models, along with the metrics used for evaluating them.

4.2.1 COVIDx CXR-3 Dataset

The main dataset that will be used for both training and testing the model is the COVIDx CXR-3 dataset Wang and Wong (2020), available on Kaggle. The dataset is comprised of 30,530 chest X-

	Train			Test		
Class	COVID-19	Pneumonia	Normal	COVID-19	Pneumonia	Normal
ActualMed	25	0	0	0	0	0
BIMCV-COVID19+	200	0	0	0	0	0
Cohen	0	0	0	270	52	0
Figure1	24	0	0	0	0	0
RICORD	0	0	0	1096	0	0
RSNA	0	4723	7166	0	880	1019
SIRM	943	0	0	0	0	0
Stony Brook	14132	0	0	0	0	0
Total/Class	15324	4723	7166	1366	932	1019
Total	27213			3317		

Table 4.2: Dataset sample count per class and data source, rearranged for our experiments.

rays, combined from various sources (Cohen et al., 2020; Chowdhury et al., 2020; Rahman et al., 2020; of North America, 2019; de la Iglesia Vayá et al., 2020), of COVID-19, pneumonia, and healthy patients.

	Train			Test		
Class	COVID-19	Pneumonia	Normal	COVID-19	Pneumonia	Normal
ActualMed	25	0	0	0	0	0
BIMCV-COVID19+	200	0	0	0	0	0
Cohen	270	52	0	0	0	0
Figure1	24	0	0	0	0	0
RICORD	896	0	0	200	0	0
RSNA	0	5503	8085	0	100	100
SIRM	943	0	0	0	0	0
Stony Brook	14132	0	0	0	0	0
Total/Class	16490	5555	8085	200	100	100
Total	30130			400		

Table 4.1: Dataset sample count per class and data source, with Wang and Wong (2020)'s original train/test split.

The authors provide metadata containing patient ID, filename, class, and data source, in a pair of suggested train/test split files. However, after proper analysis, some images were incorrectly labelled, i.e. in COVID-19 datasets, the images were labelled as "positive" instead of "COVID-19". After re-labelling, we also remarked that the suggested train/test split was 98.69%/1.31%. We deemed that the test split was too small for our purposes, and reorganized the split to 89.14%/10.86%.

To balance the size of each split, the images from the Cohen and RICORD datasets were transferred in its entirety to the test split. Our thinking was that, if our models are able to correctly learn what differentiates images from different images, without relying on confounders, then it should be able to generalize to data from sources it hasn't seen while training. Additionally, we further transferred images from the RSNA dataset, until we achieved a train/test split close to 90/10.

4.2.2 Metrics

Evaluation methods are essential in any machine learning project. They not only allow us to assess a neural network's performance, but also allow us to compare different models, and decide which ones are more appropriate for certain tasks. Using different evaluation metrics is important as well, since different models will have different results for different metrics, and it is the combination of all metrics, or the combination of specific metrics that are more important for a task, that indicate which are the better models for the task.

One of the metrics to be considered is the accuracy of the model. Accuracy is the fraction of correct predictions made by a model. Formally, its definition for a multiclass model is:

$$\text{accuracy} = \frac{\sum_{i=1}^{k} \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i}}{k}, \tag{4.1}$$

where k is the number of classes, tp_i is the number of true positive cases of class i detected by the model, tn_i is the number of true negative cases of class i detected, fp_i is the number of false positive cases of class i detected by the model, and fn_i is the number of false negative cases of class i detected.

Other metrics often used are precision (or positive predictive value), recall (or sensitivity), and F-score. Informally, for a given class, the precision corresponds to the amount of positive predictions that were correct, the recall corresponds to the number of actual positive cases that were detected, and thw F-score is a combination of the precision and recall metrics. The F-score generally used is F_1, which is a harmonic mean of the precision and recall. The formulas for these metrics are

$$\text{precision}_i = \frac{tp_i}{tp_i + fp_i}, \tag{4.2}$$

$$\text{recall}_i = \frac{tp_i}{tp_i + fn_i}, \tag{4.3}$$

$$\text{F}_1^i = \frac{2 \times \text{precision}_i \cdot \text{recall}_i}{\text{precision}_i \cdot \text{recall}_i} \tag{4.4}$$

where i is the class for which the metrics are being calculated, tp_i is the number of true positive cases of class i detected by the model, tn_i is the number of true negative cases of class i detected, fp_i is the number of false positive cases of class i detected by the model, and fn_i is the number of false negative cases of class i detected. It is also possible to combine these class metrics, as follows

$$\text{precision}_M = \frac{\sum_{i=1}^{k} \text{precision}_i}{k}, \tag{4.5}$$

$$\text{recall}_M = \frac{\sum_{i=1}^{k} \text{recall}_i}{k}, \tag{4.6}$$

$$\text{F}_1^M = \frac{\sum_{i=1}^{k} \text{F}_1^i}{k}, \tag{4.7}$$

where k is the number of classes.

Note that, according to the formulas provided, all of items have equal weight, which means classes with more elements have more influence in the metric. It is possible to calculate these metrics by giving the same importance to each class, independently of the number of elements each contains:

$$\text{precision}_{weighted} = \frac{\sum_{i=1}^{k} \#i \cdot \text{precision}_i}{\sum_{i=1}^{k} \#i}, \tag{4.8}$$

$$\text{recall}_{weighted} = \frac{\sum_{i=1}^{k} \#i \cdot \text{recall}_i}{\sum_{i=1}^{k} \#i}, \tag{4.9}$$

$$\text{F}_1^{weighted} = \frac{\sum_{i=1}^{k} \#i \cdot \text{F}_1^i}{\sum_{i=1}^{k} \#i}, \tag{4.10}$$

In the previous expressions, $weighted$ means that this metric gives equal weight to each class, while a macro-level metric, i.e. a metric that gives equal weight to all items, is represented with an M, that we can see in the expressions 4.5, 4.6, and 4.7.

The aforementioned metrics are often calculated based on a confusion matrix (Figure 4.3), i.e. a table where the columns and rows correspond to the classes, and each row contains the total number of the actual samples belonging to a class, while each column contains the total number of samples predicted to belong to a class.

		Predicted Class		
		A	**B**	**C**
True Class	**A**	7	1	3
	B	1	8	2
	C	3	1	9

Table 4.3: Example of a confusion matrix.

4.3 Summary

This chapter detailed the baselines and models implemented for the image classification task, along with associated training method details, the dataset used for experiments, and the evaluation metrics. Section 4.1 presented the baseline models, the Swin Transformer model, and the ConvNeXT architecture, and Section 4.2 presents the dataset and metrics used in this thesis.

5

Experimental Evaluation

Contents

5.1 Results 55

5.2 Discussion 57

5.3 Summary 58

This chapter introduces and analyzes the results obtained in our experiments. First, we present the the training and test accuracy, followed by the confusion matrices and associated precision, recall and F1 scores for the baseline and studied models in Section 5.1. Finally, Section 5.2 presents possible explanations for the results we obtained.

5.1 Results

In this section we present the results we obtained on the chosen baselines and proposed models. We start by presenting the accuracy, followed by precision, recall and F1 score per class, and finally the macro and weighted precision, recall and F1 score.

5.1.1 Training and Testing Accuracy

The first models we will analyze are the baselines: ResNet50, VGG16, and DenseNet121.

Model	Train Accuracy (%)	Test Accuracy (%)
ResNet50	95.5690	88.2122
VGG16	95.5625	88.0916
DenseNet121	95.6508	86.3732
Swin-T	95.0022	85.7401
ConvNeXT-T	96.8073	88.3328

Table 5.1: Accuracy results of baselines and studied models.

From Table 5.1, we can see that the results are not significantly different between each model, with the DenseNet having the best accuracy in the training set, and the ResNet having the best accuracy in the test set.

As for the studied models, the ConvNeXT has both the best training and testing accuracy overall, while the Swin Transformer has the lowest testing accuracy overall. On the test dataset, the DenseNet121 and Swin-T slightly underperform in accuracy, with a difference of 2-3% to the other models.

We also remark that the accuracy on the test set decreases by 7%-11% when compared to the training accuracy for all models.

5.1.2 Confusion Matrices, Precision, Recall, and F1 Score per Class

The confusion matrices (Table 5.2) provide insight on each baseline model's performance on the test dataset, especially when the values are converted into precision, recall, and F1 score for each class.

ResNet50	COVID-19	Normal	Pneumonia
COVID-19	1140	45	181
Normal	6	989	24
Pneumonia	58	77	797

VGG16	COVID-19	Normal	Pneumonia
COVID-19	1126	43	197
Normal	6	965	48
Pneumonia	47	54	831

DenseNet121	COVID-19	Normal	Pneumonia
COVID-19	1069	74	223
Normal	11	996	12
Pneumonia	39	93	800

Swin-T	COVID-19	Normal	Pneumonia
COVID-19	1015	96	255
Normal	0	1006	13
Pneumonia	26	83	823

ConvNeXT-T	COVID-19	Normal	Pneumonia
COVID-19	1105	54	207
Normal	1	991	27
Pneumonia	42	56	834

Table 5.2: Confusion matrices of baselines and studied models.

		COVID-19	**Normal**	**Pneumonia**
ResNet50	**Precision**	0.94684	**0.83455**	**0.88716**
	Recall	0.89019	0.97056	0.92864
	F1	**0.79541**	0.85515	0.82420
VGG16	**Precision**	0.95505	0.82430	0.88487
	Recall	**0.90866**	0.94701	0.92744
	F1	0.77230	0.89163	0.82769
DenseNet121	**Precision**	0.95532	0.78258	0.86036
	Recall	0.85641	0.97743	0.91292
	F1	0.77295	0.85837	0.81342
Swin-T	**Precision**	**0.97502**	0.74305	0.84337
	Recall	0.84895	**0.98724**	0.91289
	F1	0.75435	0.88305	0.81364
ConvNeXT-T	**Precision**	0.96254	0.80893	0.87908
	Recall	0.90009	0.97252	**0.93491**
	F1	0.78090	**0.89485**	**0.83400**

Table 5.3: Precision, Recall, and F1 per class for each model.

According to these results (Table 5.3), among the baselines we can say that, for the COVID-19 class, the DenseNet121 has greater precision, and the VGG16 has greater recall; for the Normal class, the ResNet50 has greater precision, while the DenseNet121 has greater recall; and for the Pneumonia class, the ResNet50 has greater precision and recall.

However, when compared to the studied models, the Swin Transformer has the greatest precision for the COVID-19 class, and the greatest recall for the Normal class. As for the ConvNeXT, it has larger recall for the pneumonia class than the ResNet50 model.

5.1.3 Macro and Weighted Metrics

The metrics per class, after averaged per class (macro) and per sample (weighted), can be used to evaluate the overall performance of the models.

The macro and weighted metrics (Table 5.4) for each baseline model show that VGG16 has larger macro and weighted precision, and macro recall, while the ResNet50 has larger weighted recall. When also analyzing the studied models, we can see that ConvNeXT has larger precision and recall than all of the other models in both macro and weighted averages, and consequently also a better F1 score.

For the purpose of classification in a medical context, we want to balance precision and recall, to

	Ma. Precision	W. Precision	Ma. Recall	W. Recall	Ma. F1	W. F1
ResNet50	0.87748	0.88689	0.88675	**0.88212**	0.88000	**0.88221**
VGG16	**0.87867**	**0.88945**	**0.88765**	0.88092	**0.88000**	0.88188
DenseNet121	0.86156	0.87369	0.87279	0.86373	0.86224	0.86332
Swin-T	0.85944	0.87429	0.87111	0.85740	0.85663	0.85637
ConvNeXT-T	**0.88118**	**0.89232**	**0.89210**	**0.88333**	**0.88266**	**0.88356**

Table 5.4: Macro and Weighted Precision, Recall, and F1 per for each model.

avoid resources from being wasted and also assuring that the correct treatment is given to patients. In this case, despite the model with the best precision/recall varying per class, when averaging for all classes/samples, the model with the best metrics - accuracy, macro/weighted precision, macro/weighted recall - is the ConvNeXT model. However, the performance difference is not sufficient to declare that ConvNeXT is the better alternative among all without a doubt.

5.2 Discussion

In this section we analyze the results we obtained in our experiments, and explore issues that may have affected the performance of our models. First we analyze the dataset used, followed by a comparison of performance between our baselines and studied models. Finally we address the training methods employed, and their impact on the performance of our models.

5.2.1 Dataset and Associated Challenges

As mentioned previously, the COVIDx CXR-3 dataset is comprised of over 30,000 chest X-rays. Considering datasets used in other image classification challenges, such as ImageNet, the size of this dataset is rather small, even if it is not the smallest dataset when compared to datasets used in other tasks with potential medical applications, which are particularly difficult to obtain in the first place, due to patient privacy concerns.

A consequence of this is that there may not be enough data to for the models to learn properly what characteristics make it possible for radiologists to differentiate between an X-ray of a COVID-19 patient, an X-ray of a pneumonia patient, and an X-ray of a healthy patient. Additionally, there may be situations where a radiologist would have difficulty diagnosing an X-ray, and the only method to do so would be through an RT-PCR test. In situations such as these, while it would be significant if a model could outperform a certified radiologist, it is not expected for a network to do so.

The dataset does not provide information on the origin of the classification label - whether it is from diagnosis by certified radiologists, or the result of other means of diagnosis. It is also possible that

the labelling method is inconsistent and was obtained with different methods, depending on the origin dataset of the chest X-ray.

It also does not provide information on ethnicity, age, or other characteristics of the patients that would make models trained on the dataset prone to bias (Cruz et al., 2021). As a consequence, it's difficult to consider those characteristics in training to mitigate any potential biases.

Following on the fact that the datasets come from different sources - different hospitals, different original resolutions, different X-ray machines - there's a probability that the results obtained are classifying the images based not on the diseases' characteristics, but confounders, such as hospital-specific information, or annotations, that are not removed by the transform methods while training. This is particularly significant when the model needs to differentiate between COVID-19 and Pneumonia/Normal X-rays, since the data for Pneumonia/Normal is only from one source that is different from all COVID-19 X-ray sources.

5.2.2 Baselines Versus Experimental Models

The baselines presented performed relatively well for the task. While the idea of adding attention mechanisms to models seems like a potential path for improvements in image classification, in practice, the Swin Transformer did not obtain significant improvements when compared to the baselines. Neither did the ConvNeXT, a convolutional network that attempts to combine features from both regular convolutional networks and transformer models. Each model has obtained better results than others in the different metrics analyzed in this thesis, but ultimately no significant differences in performance were found in either the baseline models, nor the models studied, that could signify an improvement in the classification task.

5.2.3 Epochs and Overfitting

The approach we used for training limited the number of epochs for all models to 20, to avoid overfitting on the training set. However, there are no guarantees that overfitting hasn't occurred before 20 epochs, or that the models would have suffered overfitting past 20 epochs. It is possible that the models would have performed better on the test set if training had stopped at an earlier epoch, or if the models had been allowed a larger number of training epochs.

5.3 Summary

This chapter described the experiments' results, along with a discussion on the latter. Section 5.1 presents the results obtained with the baselines and the proposed models for the image classification

task. Finally, Section 5.2 provides a reflection on the challenges posed by the dataset, a comparison between the baselines and models, and on the proposed training methods.

6

Conclusions and Future Work

Contents

6.1 Contributions 63

6.2 Future Work 63

This thesis presented a comparison on various architectures for image classification for COVID-19 detection with chest X-rays. This chapter offers a final review of the main contributions provided by this work and suggests possible future avenues of exploration to improve the models' performance.

6.1 Contributions

This thesis provides a study on automatic image classification of chest X-rays with deep neural networks, determining whether a patient has COVID-19, pneumonia, or is healthy. Our experiments on the COVIDx CXR-3 dataset show that the task of classifying chest X-rays is complex. We observed that both baselines and the studied models performed relatively well, with an accuracy over 80%, and similar results in the other evaluation metrics. However, there were no significant improvements in our studied models, Swin-T and ConvNeXT, when compared to our baseline models, ResNet50, VGG16, and DenseNet121. The ConvNeXT model did provide a slight performance improvement in accuracy and macro/weighted metrics, however we considered the difference to not be statistically significant to claim that the studied model presents itself as an improvement on conventional architectures for the task.

6.2 Future Work

The obtained results shown in Chapter 5 highlight the difficulty of neural models in significantly improving their classification ability. There was no significant increase in the performance of the studied models, when compared to the baselines. As highlighted in Section 5.2 in chapter 5, this may suggest that:

- The training methods used in this study were not adequate to ascertain whether the performance obtained is the best performance possible for the models on the used dataset;
- The dataset used, along with its modifications, is not sufficiently representative of the data distribution, which does not allow the models to learn the characteristics present in the images that correspond to the purported conditions, and generalize them to unseen data;
- The models studied and associated parameters are a limitation that do not show the full potential that the architectures possess in this task.

To address the first issue, one possible approach would be to split the training set further into a training and validation set, and evaluate when the models would be at risk of overfitting based on their performance on the validation set (Morgan and Bourlard, 1990; Reed, 1993; Prechelt, 2012), with the consequence that there will be less data available for training the models. Additionally, an early stopping criteria could be introduced where training would be stopped once performance on the validation set had decreased.

To address the second issue, possible solutions would be to either change the dataset used to one with further information on possible confounders or information on limitations that could affect the model's training (Cruz et al., 2021; Ahmed et al., 2021), or further analyze the current dataset, and adjust the dataset or training methods according to our findings.

The first option would be ideal, but does not seem feasible with the current publicly available datasets, which are, for the most part, either small-sized datasets i.e. the Cohen dataset (Cohen et al., 2020), or collections of various COVID-19/pneumonia datasets, such as the one we used in this thesis (Wang and Wong, 2020).

The second option would require considerable work, since we would have to analyze the sources of the collection for characteristics that could prove to be possible confounders. Additionally, we could employ the method in Minaee et al. (2020) of occluding parts of the images until the models no longer classified the image as a COVID-19 chest X-ray, or Grad-CAM (Selvaraju et al., 2019), to visualize if the models were classifying the samples based on information present in the relevant portion of the X-rays. However, there would be no guarantees that the areas relevant for the model are the areas relevant for diagnosis without the assistance of expert radiologists or a dataset with the relevant areas labeled.

As for the third issue, multiple approaches could be taken: one such approach would be to use larger variants of the architectures. We used the "T" variants of the Swin and ConvNeXT architectures, which are the smallest available. However, "B" and larger variants have obtained improved results on the ImageNet and other datasets, which could indicate they have a better classification ability compared to smaller versions. However, fine-tuning on larger variants will also require more resources.

Another approach would be to further modernize parameters associated with training, such as the optimizer. The optimizer used in this thesis for the proposed models was the ADAMW optimizer (Loshchilov and Hutter, 2017). More recent alternatives for optimizers have appeared, such as Sharpness-Aware Minimization (Foret et al., 2020) or SAM, which attempts to find the minimum loss in neighbourhoods with both low loss and low curvature (hence why its sharpness-aware), or the Surrogate Gap Guided Sharpness-Aware Minimization (Zhuang et al., 2022) or GSAM, an improvement on the SAM optimizer. Both optimizers have empirically shown improved model generalization on various datasets.

Bibliography

Ahmed, K. B., Hall, L. O., Goldgof, D. B., Goldgof, G. M., and Paul, R. (2021). Deep Learning Models May Spuriously Classify COVID-19 from X-ray Images Based on Confounders. *arXiv 2102.04300*.

Apostolopoulos, I. D. and Mpesiana, T. A. (2020). COVID-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. *arXiv 2003.11617*.

Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv 1610.02357*.

Chowdhury, M. E. H., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M. A., Mahbub, Z. B., Islam, K. R., Khan, M. S., Iqbal, A., Emadi, N. A., Reaz, M. B. I., and Islam, M. T. (2020). Can AI Help in Screening Viral and COVID-19 Pneumonia? *arXiv 2003.13145*.

Cohen, J. P., Morrison, P., and Dao, L. (2020). COVID-19 image data collection. *arXiv 2003.11597*.

Cruz, B. G. S., Bossa, M. N., Sölter, J., and Husch, A. D. (2021). Public Covid-19 X-ray datasets and their impact on model bias – A systematic review of a significant problem. *medRxiv 2021.02.15.21251775*.

de la Iglesia Vayá, M., Saborit, J. M., Montell, J. A., Pertusa, A., Bustos, A., Cazorla, M., Galant, J., Barber, X., Orozco-Beltrán, D., García-García, F., Caparrós, M., González, G., and Salinas, J. M. (2020). BIMCV COVID-19+: a large annotated dataset of RX and CT images from COVID-19 patients. *arXiv 2006.01174*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv 2010.11929*.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv 2010.01412*.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv 1512.03385*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). *arXiv 1606.08415*.

Henkel, C. (2021). Efficient large-scale image retrieval with deep feature orthogonality and Hybrid-Swin-Transformers. *arXiv 2110.03786*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv 1207.0580*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv 1704.04861*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2018). Densely Connected Convolutional Networks. *arXiv 1608.06993*.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep Networks with Stochastic Depth. *arXiv 1603.09382*.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive Fields of Single Neurons in the Cat's Striate Cortex. *Journal of Physiology*, 148:574–591.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive Fields and Functional Architecture of Monkey Striate Cortex. *Journal of Physiology*, 195:215–243.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$0.5MB model size. *arXiv 1602.07360*.

Ilyas, M., Rehman, H., and Nait-ali, A. (2020). Detection of COVID-19 From Chest X-ray Images Using Artificial Intelligence: An Early Review. *arXiv 2004.05436*.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv 1502.03167*.

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., Seekins, J., Mong, D. A., Halabi, S. S., Sandberg, J. K., Jones, R., Larson, D. B., Langlotz, C. P., Patel, B. N., Lungren, M. P., and Ng, A. Y. (2019). CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *arXiv 1901.07031*.

Jaeger, S., Candemir, S., Antani, S., Wáng, Y. X., Lu, P. X., and Thoma, G. (2014). Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. *Quantitative Imaging in Medicine and Surgery*, 4(6):475–477.

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M. K., Pei, J., Ting, M. Y., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., Shi, A., Zhang, R., Zheng, L., Hou, R., Shi, W., Fu, X., Duan, Y., Huu, V. A., Wen, C., Zhang, E. D., Zhang, C. L., Li, O., Wang, X., Singer, M. A., Sun, X., Xu, J., Tafreshi, A., Lewis, M. A., Xia, H., and Zhang, K. (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 172(5):1122–1131.

Khan, A. I., Shah, J. L., and Bhat, M. M. (2020). CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images. *arXiv 2004.04931*.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv 1412.6980*.

Koivunen, A. C. and Kostinski, A. B. (1999). The Feasibility of Data Whitening to Improve Performance of Weather Radar. *Journal of Applied Meteorology*, 38(6):741–749.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*.

Le Dinh, T., Lee, S.-H., Kwon, S.-G., and Kwon, K.-R. (2022). COVID-19 Chest X-ray Classification and Severity Assessment Using Convolutional and Transformer Neural Networks. *Applied Sciences*, 12(10).

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, Z. Q., Shafiee, M. J., Bochkarev, S., Jules, M. S., Wang, X. Y., and Wong, A. (2019). Do Explanations Reflect Decisions? A Machine-centric Strategy to Quantify the Performance of Explainability Algorithms. *arXiv 1910.07387*.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv 2103.14030*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A ConvNet for the 2020s. *arXiv 2201.03545*.

Loshchilov, I. and Hutter, F. (2017). Decoupled Weight Decay Regularization. *arXiv 1711.05101*.

Mamalakis, M., Swift, A. J., Vorselaars, B., Ray, S., Weeks, S., Ding, W., Clayton, R. H., Mackenzie, L. S., and Banerjee, A. (2021). DenResCov-19: A deep transfer learning network for robust automatic classification of COVID-19, pneumonia, and tuberculosis from X-rays. *arXiv 2104.04006*.

Minaee, S., Kafieh, R., Sonka, M., Yazdani, S., and Soufi, G. J. (2020). Deep-COVID: Predicting COVID-19 From Chest X-Ray Images Using Deep Transfer Learning. *arXiv 2004.09363*.

Morgan, N. and Bourlard, H. (1990). *Generalization and Parameter Estimation in Feedforward Nets: Some Experiments*, page 630–637. Morgan Kaufmann Publishers Inc.

Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the International Conference on Machine Learning*.

Narin, A., Kaya, C., and Pamuk, Z. (2021). Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks. *arXiv 2003.10849*.

of North America, R. S. (2019). RSNA Pneumonia Detection Challenge. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/.

Ozturk, T., Talo, M., Yildirim, E. A., Baloglu, U. B., Yildirim, O., and Rajendra Acharya, U. (2020). Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine*, 121:103792.

Prechelt, L. (2012). *Early Stopping — But When?*, pages 53–67. Springer Berlin Heidelberg.

Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Abul Kashem, S. B., Islam, M. T., Al Maadeed, S., Zughaier, S. M., Khan, M. S., and Chowdhury, M. E. (2020). Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *arXiv 2012.02238*.

Reed, R. (1993). Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747.

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2019). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *arXiv 1610.02391*.

Shi, F., Wang, J., Shi, J., Wu, Z., Wang, Q., Tang, Z., He, K., Shi, Y., and Shen, D. (2021). Review of Artificial Intelligence Techniques in Imaging Data Acquisition, Segmentation, and Diagnosis for COVID-19. *IEEE Reviews in Biomedical Engineering*, 14:4–15.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv 1602.07261*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going Deeper with Convolutions. *arXiv 1409.4842*.

Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv 1905.11946*.

Ulhaq, A., Khan, A., Gomes, D., and Paul, M. (2020). Computer Vision For COVID-19 Control: A Survey. *arXiv 2004.09420*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention Is All You Need. *arXiv 1706.03762*.

Wang, L. and Wong, A. (2020). COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images. *arXiv 2003.09871*.

Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *arXiv 1705.02315*.

Wong, A., Shafiee, M. J., Chwyl, B., and Li, F. (2018). FermiNets: Learning generative machines to generate efficient neural networks via generative synthesis. *arXiv 1809.05989*.

Yang, M., He, D., Fan, M., Shi, B., Xue, X., Li, F., Ding, E., and Huang, J. (2021). DOLG: Single-Stage Image Retrieval with Deep Orthogonal Fusion of Local and Global Features. *arXiv 2108.02927*.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and Understanding Convolutional Networks. *arXiv 1311.2901*.

Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N., Tatikonda, S., Duncan, J., and Liu, T. (2022). Surrogate Gap Minimization Improves Sharpness-Aware Training. *arXiv 2203.08065*.