Universitat Oberta de Catalunya

uoc.edu

Screening Chest X-rays for Covid-19 with Deep Learning

Eric Robert Gill

Computer Engineering

Artificial Intelligence

Professor responsable de l'assignatura: Joan Arnedo Moreno

Consultor: Joan M. Nuñez Do Rio

13/06/21

FITXA DEL TREBALL FINAL

Títol del treball:	*Screening Chest X-rays for Covid-19 with Deep Learning*
Nom de l'autor:	*Eric Robert Gill*
Nom del consultor/a:	*Joan M. Nuñez Do Rio*
Nom del PRA:	*Joan Arnedo Moreno*
Data de lliurament (mm/aaaa):	*06/2021*
Titulació o programa:	*Enginyera Informàtica*
Àrea del Treball Final:	*Intel·ligència Artificial*
Idioma del treball:	*Anglès*
Paraules clau	*COVID-19, X-rays, Convolutional Neural Network*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

El novel·lo Coronavirus ha provocat una pandèmia global amb alts costos econòmics i socials. La naturalesa emergent de la malaltia associada del Coronavirus, Covid-19, ha trobat moltes nacions mal preparades per controlar la seva propagació, resultant en altes taxes d'infecció i pressió en els sistemes de salut.

La detecció ràpida i la subsegüent quarantena de persones infectades és la mesura més eficaç contra la propagació del virus amb l'excepció de la vacunació. Generalment, el diagnòstic es porta a terme amb tests de Polímer invertit o antígens que poden ser cars i no sempre fàcilment disponibles. Aquests mètodes requereixen personal especialitzat i contacte físic amb el pacient, així com temps per processar els resultats.

Les màquines de raigs X estan disponibles en hospitals arreu del món en països de gairebé totes les situacions econòmiques. La radiografia s'ha utilitzat en molts casos d'ús de garbellat i diagnòstic i hi ha una investigació àmplia sobre la seva aplicabilitat a la pandèmia de Coronavirus.

Aquest projecte investiga la viabilitat de l'ús de Xarxes Neurals Convolucionals per a vetar imatges radiogràfiques de pulmons per als símptomes de Covid-19. Els rajos X són massivament disponibles, econòmics i no invasius. Els resultats mostren que les Xarxes Neurals Convolucionals poden classificar les imatges de raigs X en classes de Covid, Normal i Pneumonia Viral amb alts nivells de Precisió i Sensitivitat.

Abstract (in English, 250 words or less):

The novel Coronavirus has caused a global pandemic with high economic and social costs. The emergent nature of the Coronavirus' associated disease, Covid-19, found many nations ill-prepared to control its spread, leading to high rates of infection and strain on health systems.

Rapid detection and the subsequent quarantine of infected people is the most effective measure against the spread of the virus outside of vaccination. Diagnosis is generally carried out by Reverse Polymer or antigens tests which can be expensive and not always readily available. These methods require specialised staff and physical contact with the patient, as well as time to process results.

X-ray machines are available in hospitals across the world in countries of nearly all economic situations. Radiography has been used in many screening and diagnostic use cases and there is wide-spread investigation into its applicability in the Coronavirus pandemic.

This project investigates the viability of employing Convolutional Neural Networks to screen radiographic images of lungs for symptoms of Covid-19. X-rays are massively available, economical and non-invasive. The results show that Convolutional Neural Networks can classify x-ray images into Covid, Normal and Viral Pneumonia classes with high levels of Precision and Recall.

Index

Table of Figures

Table of tables

1. – Introduction

1.1 - Context and Justification of the Project

Since its identification on the 7th of January 2020, the novel Coronavirus and its associated disease, Covid-19, have gone on to disrupt economic, logistical, and medical systems around the globe. Although the modern world has experienced outbreaks of viruses that have threatened the security and health of many people (SARS, MERS, Ebola), a virus has not made such an impact since the emergence of HIV/AIDS at the beginning of the 1980s [1] .

According to the European Centre for Disease Prevention and Control, since 31st December 2019 and as of 2nd June 2021, a total of 167,547,945 cases of Covid-19 (in accordance with the applied case definitions and testing strategies in the affected countries) have been reported, including 3,467,722 deaths [2].

As a result of this rapid and extensive spread, measures of varying severity were taken throughout the world's nations. Many economies were negatively affected by this, with market performance in the US severely affected [3]. In the United Kingdom, the FTSE dropped by 14.3 percent, its worst performance in a calendar year since the economic crisis of 2008 (31.3%) [4] as illustrated in figure 1.

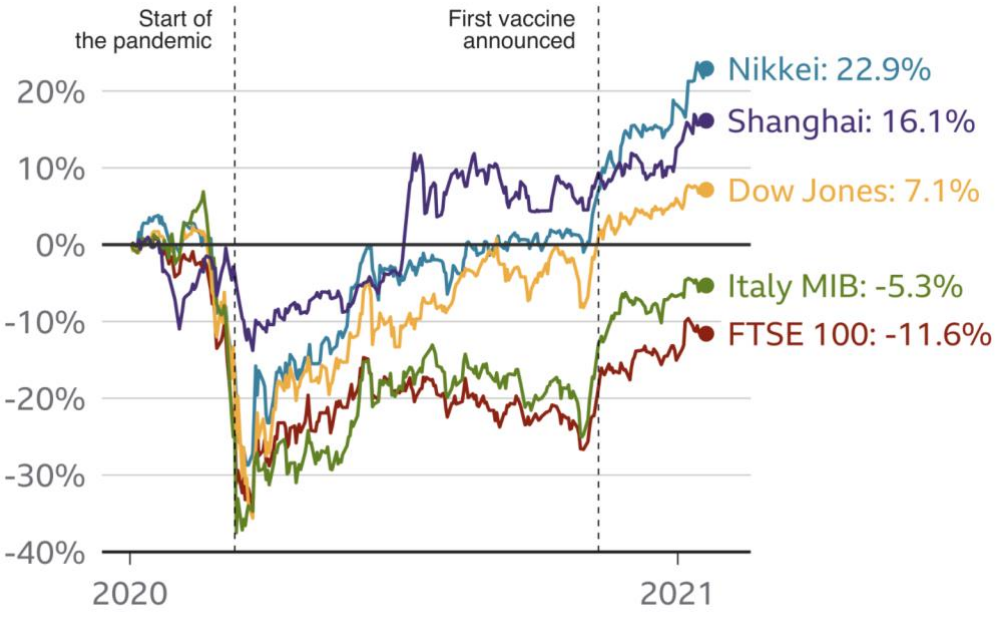

Figure 1: COVID-19 Impact on Markets (bbc.com/news)

Away from Britain, although an upturn in markets has been seen since the announcement of vaccines, not all economies have recaptured pre-pandemic levels.

Health systems have also been massively affected. Despite the best efforts of medical professionals, the death toll is still rising. As well as the loss of human life, Covid-19 has forcibly isolated vulnerable people who are at greater risk [5] due to underlying conditions, propensity to lung and heart problems, and advanced age.

Many strategies used to prevent the spread of Covid-19 employed quick detectiot, contact tracing and the subsequent isolation of those exposed or infected. This project intends to lend itself to these strategies.

Diagnoses are generally carried out via PCR (Polymerase chain reaction) or antigen rapid tests. Both have been shown to be very effective [6, 7] however, they can be considered expensive [8].

To reduce costs and diversify methods, researchers have incorporated the use of both radiological (x-ray) and computed tomography (CT) scans in the identification of symptoms consistent with COVID-19 infection [9]. The available literature indicates that applying deep learning to this question is of great interest clinically. X-ray images have been documented as showing visual indexes correlated with Covid-19 [10]. X-rays are appropriate for this exercise due to their massive availability, low-cost and the speed in which they can be produced. There has been work on using deep learning models to identify COVID-19 symptoms in radiographic images with high levels of accuracy [9], [11, 12].

1.2 - Project Objectives

1.2.1 - General Objectives

- Use deep learning to classify images of lungs into one of three classes: Covid-19, Normal or Viral Pneumonia.
- Investigate Convolutional Neural Networks and techniques to increase performance and compare results with baseline results found in literature.
- Investigate feasibility of application of deep learning to this kind of medical screening.
- Compare results obtained with results from a pretrained model using transfer-learning to fine-tune its parameters.

1.2.2 - Specific Objectives

- Research supervised deep learning, convolutional neural networks (CNN) and their application to medical image classification.
- Investigate state of the art deep learning frameworks and libraries such as Keras, TensorFlow and PyTorch and decide which is the best candidate to carry out this project.
- Use the official documentation to familiarize myself with the chosen library.
- Design and implement a basic CNN and use it as a starting point for investigation of different techniques to improve performance such as transfer learning and data augmentation.
- Employ the performance-improving techniques to extend the simple pipeline and isolate the influence of each on the results.
- Compare results obtained using the various techniques with baseline clinical levels available in literature.

- Use these results and comparisons as basis to discuss whether radiography could be applicable in screening patients for COVID-19.

1.3 - Focus and Method Followed

The environment used to develop and test the various convolutional neural networks was Google Colab [13], an online platform which provides access to GPU and CPU powered machine learning in the Cloud. This service allows projects to be developed in Python using virtually all the major libraries for constructing deep learning projects.

Machine learning models are run in the popular Jupyter Notebooks program available in Python. This format allows for programs to be developed in steps and is particularly useful for projects of this type given that data and metrics can be studied and displayed during development.

The library chosen to develop this project is Pytorch [14], an open-source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Facebook's AI Research lab. Pytorch supports GPU usage for the training of deep learning models. This technique is common in modern machine learning as it allows for the execution of calculations in parallel.

The dataset employed (see section 4.1) consists of three classes: Normal, Covid-19 and Viral Pneumonia. Convolutional Neural Networks will be used to classify these images. The evaluation of the models will consider metrics such as Accuracy, Precision, Recall and F1 score. A full explanation of these metrics is found in chapter 3.

1.4 - Planning the Project

This subject is made up of 4 continuous evaluation tests, a presentation, and a public defence.

Title	Start date	End date	Duration (days)
PAC0	17/02/2021	01/03/2021	9
PAC1 - Work Plan	02/03/2021	16/03/2021	11
PAC2 - Work development Phase 1	17/03/2021	19/04/2021	24
Configure environment in Google Colab	17/03/2021	18/03/2021	2
Analysis of Data	19/03/2021	22/03/2021	2
Research X-Ray processing with CNN	23/03/2021	26/03/2021	4
Revision of Pytorch documentation	27/03/2021	09/04/2021	10
Construction of first simple CNN model	10/04/2021	15/04/2021	4
Report progress in PAC2 deliverable	16/04/2021	18/04/2021	1
PAC3 - Work development Phase 2	20/04/2021	17/05/2021	20
Hyperparameter investigation and tuning	20/04/2021	26/04/2021	5
Tests with Data Augmentation	27/04/2021	29/04/2021	3
Tests with Batch Normalization	30/04/2021	01/05/2021	1
Tests with Dropout	02/05/2021	03/05/2021	1
Use of pre-trained model to assess Transfer-learning	04/05/2021	12/05/2021	7
Report progress in PAC3 Deliverable	13/05/2021	17/05/2021	3
PAC 4 - Redaction of Bachelor's Thesis	18/05/2021	08/06/2021	16
Complete First Draft	18/05/2021	26/05/2021	7
Revise and Second Draft	27/05/2021	02/06/2021	5
Final Version	03/06/2021	08/06/2021	4
Record Presentation	09/06/2021	13/06/2021	3
Public Defense	16/06/2021	23/06/2021	6

Table 1: Project Timeline Table

The distribution of tasks can be seen in figure 2.

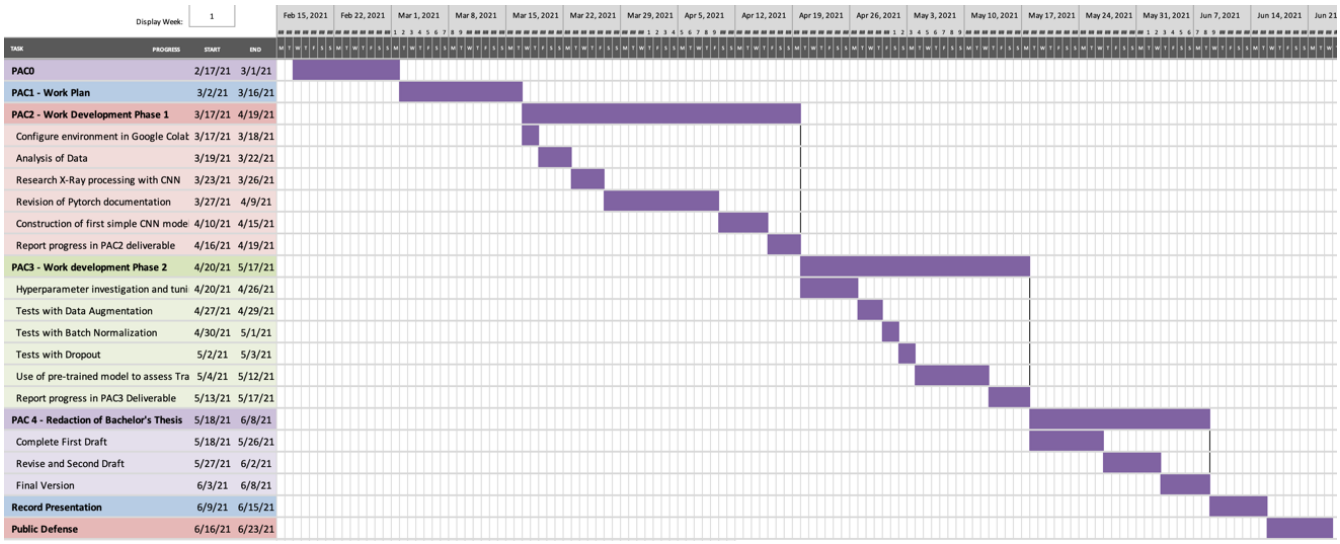

Figure 2: Gannt Diagram

1.5 - Summary of the products obtained

The product obtained by this project will be an investigation and exploration of solutions to the three class (Covid-19, Normal and Pneumonia) classification problem.

A full explanation of the methodology and techniques applied, along with a comparative study of their results. A pre-trained neural network will be fine-tuned on the x-ray images to compare this project's results with the state of the art.

1.6 - Brief descriptions of the other chapters in the bachelor's thesis

The second chapter of this bachelor's thesis will give the medical context of the project. It will cover the use of medical imaging in diagnosis, the use of deep-learning in concert with medical imagery and a brief examination of the state of the art using deep-learning and X-rays to diagnose and screen for covid-19.

The third chapter is a general explanation of machine learning with a focus on CNNs. This chapter explains CNNs in terms of the techniques employed in this project and serves to contextualise why this specific type of machine learning techniques were chosen.

The fourth chapter outlines the methodology employed. It features an explanation and exploration of the dataset, the different architectures and configurations employed, the techniques used to improve performance and the metrics used to evaluate this same performance.

The fifth chapter presents the results obtained by the various models constructed and the pre-trained model used. This chapter compares the models in terms of the metrics explained in the previous chapter exposing the strengths and weaknesses of the individual models.

Chapter six is a discussion of the results obtained. It will compare the results in the context of the techniques employed in each experiment, examining the influence of each, and referencing similar results in available literature where applicable.

The final chapter contains the conclusions obtained. It considers the results obtained in this study along with the baseline clinical results available in literature to determine if radiography could be a useful tool in the medical screening Covid-19 patients.

2. – Medical Context

2.1 – Covid-19

The effects of Covid-19 were first detected as pneumonia of an unknown source in Wuhan City, Hubei Province of China in December 2019 [15]. It was rapidly shown to be caused by a novel coronavirus that is structurally related to the virus that causes severe acute respiratory syndrome (SARS) [16].

As early into the crisis as May 2020, Covid-19 was shown to be highly transmissible. It was shown that, in cities with similar transmission conditions to Wuhan, the Coronavirus had a 50% chance of establishing itself within the population once four independent cases had been introduced [17].

Detection of Covid-19 is usually carried via reverse PCR and antigens tests that, while effective [6, 7], can be expensive [8]. These characteristics have led to investigation into other forms of diagnosis of Covid-19 [18].

2.2 – Imaging in Diagnosis of Lung Problems

Imaging techniques such chest radiographies (CXR), computed tomography (CT) scans and magnetic resonance imaging (MRI) can all be used in the diagnosis of issues such as pneumonia, bronchial carcinoma, pulmonary hypertension, cystic fibrosis, and pulmonary fibrosis [19].

Each technique has advantages and disadvantages in terms of precision, availability and exposure to radiation as can be seen in table 2.

	CXR	CT	MRI
Advantages	Widely available Exploratory first study	High spatial resolution High sensitivity High speed Workflow	Intermediate spatial resolution High contrast resolution High temporal resolution Function No radiation exposure
Disadvantages	Low sensitivity Low specificity	Allergy to contrast agent Contraindications: impaired renal function, thyroid function	Availability Study acquisition time Allergy to contrast agent Contraindications: implants
Indications	Pneumonia, Bronchial carcinoma (detection) Pulmonary hypertension COPD Cystic fibrosis Fibrosis	Complicated pneumonia, pneumonia in at-risk patients Bronchial carcinoma (staging) Acute pulmonary embolism Pulmonary hypertension COPD Fibrosis	Bronchial carcinoma Pulmonary hypertension Cystic fibrosis
Remuneration EBM points GöA rate (basic)	430 €26.23	1865; + 645 with contrast agent €134.06; with contrast agent €151.55 + additional consumables	3430; + 1260 with contrast agent €250.64; with contrast agent €326.42 + additional consumables
Dose [20]	0.1 mSv	Low-dose-CT 0.2–1 mSv Routine now 1–5 mSv Routine 10 years ago 10 mSv	None
Number Performed (Germany) [21, 22]	15 million/year	Total estimated 2 million/year Inpatient 830000/year	Inpatient 12000/year

Table 2: Comparison of imaging techniques in diagnosis [19]

CXR has a wide range of applications, is cheaper than the other options, delivers a lower dose of radiation and is carried out far more frequently. CXR's low cost has led to it being studied as a detection tool for acute respiratory infection in children in the developing world [23].

2.3 – Machine Learning in Diagnosis

The utilization of machine learning (ML) in medical diagnosis is common. It has been applied in the diagnosis and treatment of neurodegenerative diseases, the prediction and prognosis of cancer, research into diabetes and a host of other medical uses [24]–[27].

The application of ML, specifically deep learning (DL), to classifying medical images is also well researched [28]. Studies have made use of DL techniques in areas as diverse as neuro, retinal, pulmonary, digital pathology, breast, cardiac, abdominal, and musculoskeletal investigation/diagnosis [29].

Among these imaging techniques, deep learning in the form of CNN have been shown to be effective in the detection of pneumonia in CXR images [30, 31]. This success in pneumonia detection indicates that CNN could be used to detect the effects of the coronavirus in radiographies.[9, 10], [32]

2.4 – Covid-19 detection with Deep Learning in Chest X-rays

The type of pneumonia caused by Covid-19 has been shown to manifest similar features to other types of viral pneumonia in radiographic images [9, 10], [32], which has led researchers to apply DL to CXR to detect and screen Covid-19.

Some of the leading studies have achieved admirable results, Kedia et al [11] achieved an overall accuracy of 98.28% on classification of Covid, Pneumonia and Normal images with F1 scores of 0.99, 0.98 and 0.98 respectively.

Due to the Coronavirus novel status, dataset size has been limited. Many papers focus on mitigating the issues that small datasets imply. Transfer learning and data augmentation are used in papers by Chowdhury et al [9] and Minaee et al [12] to achieve sensitivities of up to 97.94 and 98% respectively.

This project will investigate the same three-class classification problem as Kedia et al. Here we start from a base model and progressively apply techniques to improve performance to contribute to this body of work. In parallel with work by Chowdhury et al and Minaee et al, this project evaluates the effectiveness of data augmentation and transfer learning to mitigate the technical limitations of small datasets.

3. – Machine Learning

3.1 – Introduction

The first formal study of Artificial Intelligence (AI) was carried out by ten scientists in Dartmouth College in Hanover, New Hampshire, in 1956 [33]. In the proposal for this summer project, themes as wide-ranging as natural language processing, neural networks, the theory of computation, abstraction, and creativity were mentioned.

Machine Learning (ML) is a subset of AI, it is concerned with a computer system's ability to acquire domain knowledge. More concretely, ML is the study of algorithms' ability to improve automatically through experience and the use of data. The aim of machine learning is that a computer system can learn from data, identify patterns, and make decisions with minimal human intervention.

The way computers can be made to act without receiving explicit instructions depends on what kind of problem is to be solved. The four main machine learning techniques are Supervised Learning, Unsupervised Learning, Semi-Supervised Learning and Reinforcement Learning. This project employs supervised learning and thus will be the focus of our explanation, but a brief overview of the other techniques is included for context.

Unsupervised Learning

Unsupervised learning is a type of machine learning in which the system is expected to discover patterns and information that were previously unknown. Clustering is one such technique, it is focused on finding a structure or pattern in uncategorized data.

Data used in unsupervised learning is known as unlabelled data as we do not provide information a priori to the computer system about the type or category of the data. It is the responsibility of the system to categorize that data and group samples together based on correlations it detects. Unlabelled data is inexpensive to assemble, and unsupervised learning algorithms can be useful in identifying correlations in high-dimensional data.

Semi-supervised Learning

Semi-supervised learning is a technique that combines concepts from unsupervised and supervised learning. A small amount of labelled data (data whose category of type is pre-defined, usually manually by humans) is combined with a large amount of unlabelled data.

The labelled data serves as a reference point in the grouping of the unlabelled data. The cost of manually categorizing large amounts of data may render constructing large, labelled datasets unfeasible. Semi-supervised learning intends to mitigate this problem by leveraging small amounts of labelled data to categorize large amounts of unlabelled data.

Semi-supervised learning may be transductive learning or inductive learning. The goal of transductive learning is to infer the labels for the provided unlabelled data using the labelled data as reference. The goal of inductive learning is to infer the correct mapping of unlabelled data to the labels provided.

Reinforcement Learning

Reinforcement learning is the area of machine learning which studies how intelligent agents should take actions in an environment to maximise reward. The purpose of reinforcement learning is for the agent to learn an optimal, or nearly optimal, policy regarding decision making that maximizes a reward function or other type of human-designed reinforcement signal that accumulates from the immediate rewards.

There are many approaches which define the reward function, but it can generally be understood as a function which will return higher values when an action that is more favourable to the policy of the system is taken. As the system explores and compares possible actions, it should identify those which will provide the highest reward and learn to follow these actions.

3.2 – Supervised Learning

Supervised learning works on the basis that an algorithm can be trained to perform more accurately by being fed labelled data and continuously adjusting its parameters according to its performance on each sample. This technique is used for classification and regression tasks.

3.2.1 – Loss function

In supervised learning for classification, an algorithm is trained using labelled data. Given a sample of data, the algorithm must predict its class. A loss function that quantifies the difference between the prediction and the actual label is used to express the quality of the prediction. The parameters of the algorithm are then adjusted to minimise this loss function. As this process is repeated, the idea is that as the loss function is reduced, the quality of the predictions improves.

3.2.2 - Overfitting

One of the great challenges of supervised learning is to train algorithms in such a way that they generalise well to new data. An algorithm that performs well on the training data but poorly on the new, unseen data is said to have overfit: it has learned the characteristics of the training data and not the shared characteristics of further domain data. We will study methods to identify and avoid overfitting in the following sections.

To properly assess classification algorithms, it is not acceptable to simply drive down the loss function on training data and assume that the model generalises well to data that it has never seen. A subset of data must be withheld during training so that the model can be evaluated rigorously. Techniques defining how data is withheld and a model evaluated are known as validation techniques.

3.2.3 – Validation techniques

There are several techniques to calculate the loss function of a supervised learning algorithm so its parameters can be adjusted. Each technique attempts to evaluate the quality of the model on data that has not been used to train the model to avoid overfitting. Some of the main methods are explained below:

The hold-out method

Data is split into three subsets: training, validation, and testing. The training set is the largest and it is used, as the name suggests, to train the algorithm. The format of the training data is an input-output pair. A loss function is calculated for samples in this set by feeding the input sample into the algorithm and comparing the algorithm's output with the output sample. The parameters of the model are subsequently adjusted to minimize the loss function.

The validation data has the same input-output format and is data that has not been included in the training set and is, therefore, unseen by the algorithm. A loss function is also calculated on the validation set, but it is not used to adjust the parameters of the model, it is strictly for evaluation purposes during training.

The test set is unseen labelled data that is used to evaluate the adjustment of the hyperparameters carried out during the training phase. Once training is completely finished, the model is evaluated using the test set.

K-Fold Cross Validation

In this technique, k-1 subsections (folds) are used for training, and one for testing as seen in the figure 3.

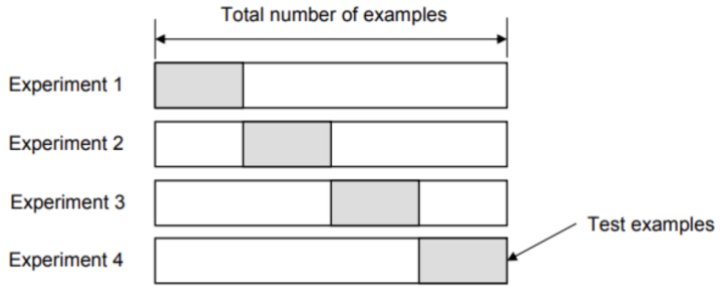

Figure 3: K-folds [34]

The average of the error rate of each iteration is taken to calculate the overall average.

Leave-One-Out Cross-Validation (LOOCV)

In this method, every data-item except one sample is used for training, the left-out record is then used for testing. This technique uses every sample for training and evaluation and takes the average error of each iteration as the final overall error. A depiction of this technique can be seen in figure 4.

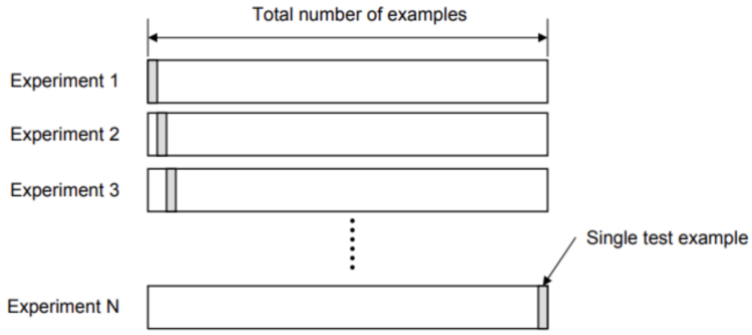

Figure 4: LOOCV [34]

Random Subsampling

In this technique, a randomly selected subset of data is selected to make up the test set in each iteration. An average error is taken to calculate an overall score at the end of training. This technique is depicted in figure 5.

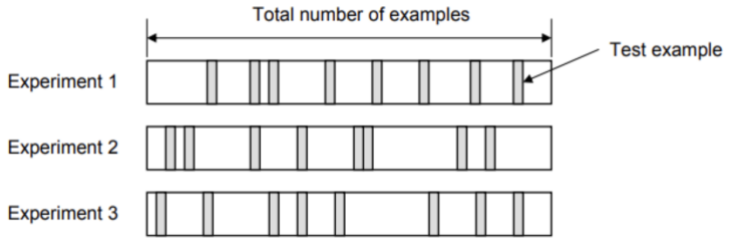

Figure 5**: Random Subsampling [34]**

3.3 – Artificial Neural Networks

3.3.1 – The Multilayer Perceptron

To understand the concept of the multi-layer perceptron (MLP), we must first establish what a perceptron is and what limitations it has. A perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a learning algorithm called a threshold function; a function that maps X (a matrix) to an output value f(x) that is a binary value.

$$f(x) = \begin{cases} 1 \; if \; w \cdot x + b > 0 \\ 0 \; otherwise \end{cases}$$

Where *w* is a weight, *x* is an element of the vector and *b* is the bias. The bias defines the distance between the function the origin. As *w·x* is the dot product, if b is negative, the sum with weight of all inputs must result in a positive value greater than |*b*| to exceed threshold 0. In terms of dimensional space, the bias alters the position of the decision boundary.

The limitation of perceptrons is that they can solve only linearly separable problems. A classic example of this limitation is the impossibility of modelling XOR with a single perceptron. Both the AND and OR functions are basic Boolean operations, and linearly separable. XOR on the other hand is formed of three basic Boolean operations:

$$(p \vee q) \wedge \neg(p \wedge q)$$

This behaviour cannot not be modelled by a single perceptron and must be handled by a multi-layer perceptron. Multi-layer perceptrons have their roots in the need to add perceptual layers to solve more complex problems.

A multi-layer perceptron layer consists of at least three layers; the input layer, the hidden layer, the output layer, and the connections between perceptrons (also called units or neurons).

An MLP consisting of at least these three layers is an Artificial Neural Network (ANN). The term deep learning (DL) comes from using many layers to extract features from data. A standard ANN is made of fully connected layers. That is, every node in a layer is connected to each in the next and previous layers.

The behaviour of an ANN can be generally said to be the following:

- Input units receive values from the outside, and will be activated (i.e., produce a certain value in the output) or not based on the input received.
- The outputs of the input layer are, in turn, the inputs of the hidden layer, so these units receive a set of inputs against which they will react. To do this, each unit of the hidden layer has a vector of weights, a value for each incoming connection, which combines with the corresponding signals and, as a result, causes the output of each hidden unit to be activated or not.
- Finally, the output layer units also receive the signals from the hidden layer, perform an operation with these signals and their own weight vectors and, as a result, calculate their own output, which will be the result of the network.

As we have seen in the simple perceptron, the output z is calculated from the inputs X with the weights W (in the input and hidden inputs) as:

$$z = W^T X + b$$

As we have mentioned, if *b* is negative, the output *z* will be positive only if the result of the dot product between $W^T X$ is greater than the absolute value of *b*. This value is fed forward into the next layer through an activation function.

3.3.2 – Activation Functions

The activation function serves to determine the output of a model as well as adding non-linearity into the ANN. There are many types of functions, choosing which to use depends on the specific use case. Two common activations are ReLU and Sigmoid, as seen in figure 6.

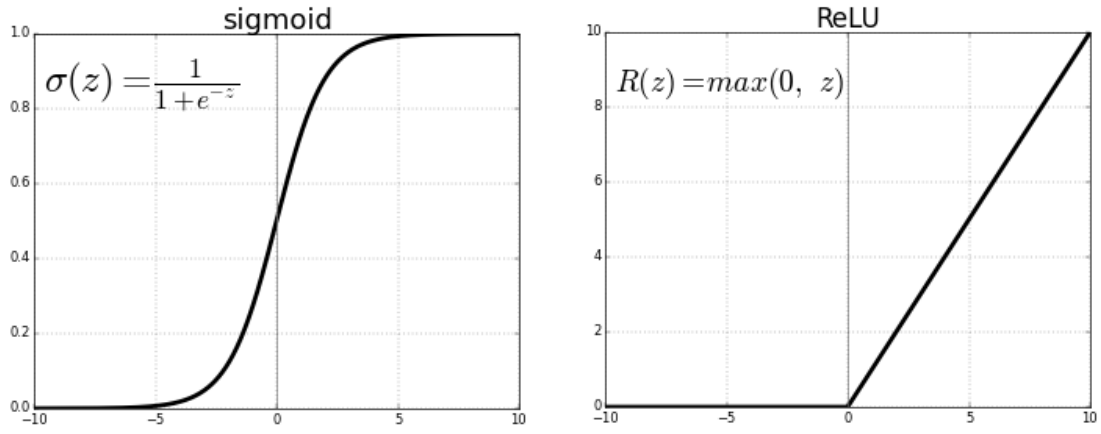

Figure 6: Sigmoid and ReLU

In terms of the output layer, the activation function depends on the desired output type. For example, if this is a real value, for example in a regression, the output is linear. In a case of binary classification, the Sigmoid function that we have seen before is usually used. For multiclass values, a function known as the Softmax function is used. Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.

3.3.3 – Backpropagation

ANN parameter adjustment in supervised learning situations is done via backpropagation. For every node, W and b are initialised with random values between 0 and 1. Training examples are then fed into the ANN and the result generated compared with the desired result. We express the difference between the result obtained and the expected result through a loss function C. The goal of training is to reduce the values of this function.

This reduction is obtained by using the method of gradient descent, calculating the loss gradient with respect to the weights and output biases:

$$\frac{\partial C}{\partial \hat{W}}$$

$$where\ \widehat{W} = \{W, b\}$$

Calculating this derivative, we find the optimal $\widehat{W}$ to reduce C. However, if we apply this only to the output layer, this only affects the output unit(s) and we have not modified the weight or bias of any hidden layer. This is where the process of backpropagation is applied: once the output layer is adjusted, the hidden layer is set by the same procedure, and so the layers are adjusted from the output to the input.

There are three variants of gradient descent as shown in table 3.

	Batch Gradient Descent	**Stochastic gradient descent**	**Mini-batch gradient descent**
Technique	Use all training samples at the same time	Use samples one by one	Executes samples in blocks
Advantages	Good results	Fast	Reduces variance
Disadvantages	Computationally expensive	High variance	High computational cost than Stochastic gradient descent

Table 3: Gradient Descent Variants

3.3.4 – Hyperparameters

The parameters of an ANN are the weights and the biases. Apart from these parameters, however, there are further configurations we can make to the models that will affect its behaviour.

Some examples of algorithm hyperparameters are:

- **Learning Rate**: When applying gradient descent, the parameters (weights and biases) are adjusted according to the cost function. The learning rate

defines how much these settings are adjusted. If it's too large we can overshoot the local minimum and if it's too small, it can take a long time to reach.

- **Momentum**: Momentum serves to indicate the direction of the next step (of the descent) to avoid oscillations.
- **Number of epochs**: The number of times the training data is inserted into the network.
- **Batch size**: The number of examples the algorithm should study before setting the parameters.

3.4 – Overview of Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a type of Artificial Neural Network (ANN) designed for the classification of images. The first proposal for a neural network of this type was made by Dr. Kunihiko Fukushima in 1980. In the same paper, Dr Fukushima claimed that, once trained, the network has a structure like the hierarchy model of the visual nervous system proposed by Hubel and Wiesel [35]. According to Hubel and Wiesel's model, individual neurons respond to stimuli only in a restricted region of the visual field known as the *Receptive Field*. A collection of such fields overlap to cover the entire visual area. As we will see, this structure is emulated by CNNs.

Like the ANN seen earlier, a CNN is made up of three fundamental parts: the input layer, the hidden layers, and the output layer. The input serves as an entrance for visual information. The hidden layers generally consist of one of three main types, convolutional, pooling and fully connected, and is the information processing engine of the CNN. The output layer produces a signal that expresses the CNN's classification of the image. A basic structure of a CNN is shown in figure 7.

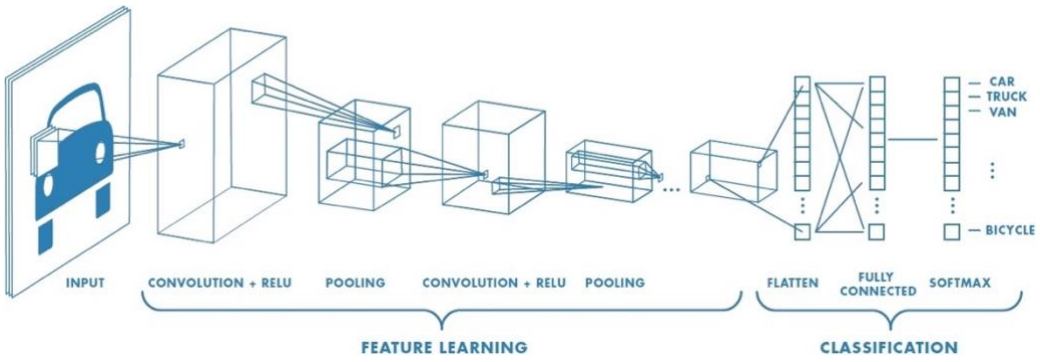

Figure 7: The layers of CNN and their functions [36]

The input layer is a single vector with one position for each input value. In the case of colour images, there would be a total of $h * w * 3$, where h and w are the height and width in pixels respectively, multiplied by 3 to consider the 3 layers of an RGB image.

Each hidden layer is made up of neurons or nodes. Each of these neurons can be semi or fully connected to all neurons in the previous layer and function completely independently of other neurons in the same layer. The last fully connected layer is called the "output layer" and in classification settings it represents the class scores. These class scores are compared with the labelled

data used in training to calculate a loss function and train parameters using backpropagation. Each neuron has learnable weights and biases, which are adjusted during backpropagation. Each neuron receives some inputs, performs a dot product with the weights, sums the bias, and optionally follows it with a non-linearity.

The advantage that CNNs have over standard ANNs in image processing is that the layers are made up of 3D volumes; neurons arranged in width, height and depth. As we saw in section 3.3.1, standard ANNs have fully connected layers, meaning each neuron is connected to each neuron of the next layer and has a weight and bias for each connection. An ANN, therefore, given a colour image of 200 pixels in height and width would have to manage $200 * 200 * 3 = 120{,}000$ weights in its first fully connected input layer. As we will see, CNNs leverage the 3D volumes of its layers to allow neurons to be connected to only a region of neurons in the next layers as opposed to every single one [37]. This reduces this type of dimensionality in weights and allows for more efficient computation. A visual representation of this can be seen in the figure 8.

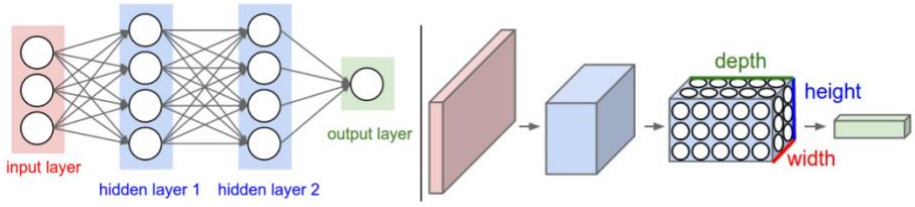

Figure 8: ANN vs CNN [37]

3.4 – The Layers of a CNN

3.4.1 - The Input Layer

A CNN makes use of an combination of different types of layers to extract features, reduce dimensionality, avoid overfitting, and activate signals in such a way that classification can be expressed. The first layer is obviously the input layer. This layer receives one input for every pixel value in every layer of the image and passes it to the first convolutional layer.

3.4.2 - The Convolutional Layer

The convolutional layer applies filters to the input to extract features. Every filter is small spatially (along width and height) but extends through the full depth of the input volume. This format allows the convolutional layer to avoid the massive amounts of connections that traditional ANNs would have.

An image of size $NX\ x\ NY$ pixels and NC color channels is usually represented by a data array of dimension $NX\ x\ NY\ x\ NC$. A convolution layer applies a set of NF filters to the volume of the input image. Each filter consists of an array of weights of a certain size $M\ x\ M\ x\ NC$. The spatial extent of the $M\ x\ M$ filter is known as the receptive field of a given neuron. Applying the operation for each pixel of the input image and each filter of the convolutional layer, we end up having a result in the form of volume of dimensions $(NX - M + 1)x\,(NY - M + 1)x\,NF$ [37]. This operation is represented visually in figure 9.

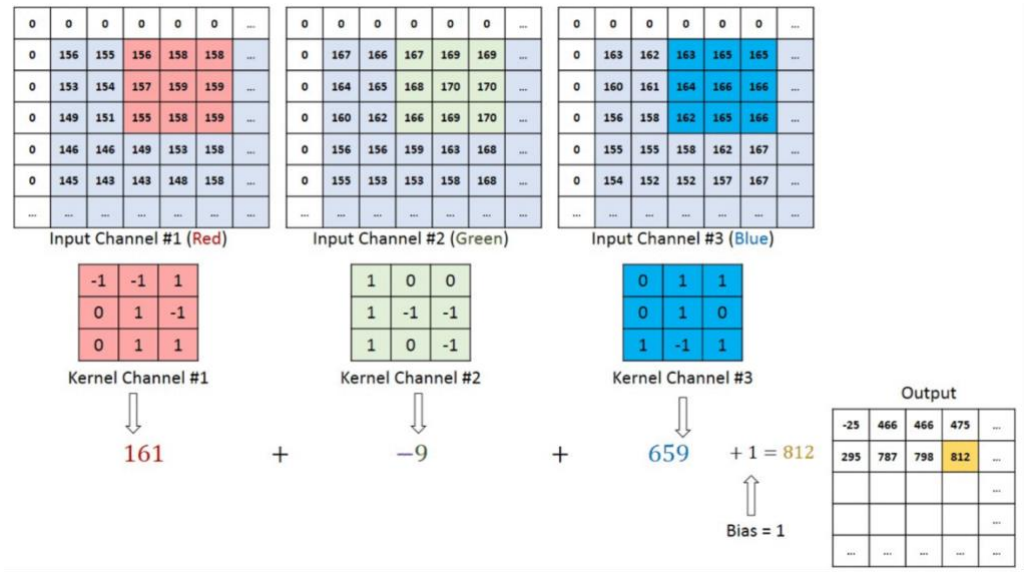

Figure 9: Filtering in Convolutional Layer [38]

The dimensions of the output volume are defined by the three hyperparameters of the convolutional layer: the **depth**, **stride,** and **padding**.

- The depth corresponds to the number of filters we would like to use, each learning to look for something different in the input.
- The stride is the number of pixels at a time we slide the filter across the image. A stride of one means moving the entire filter one pixel.
- Padding (also known as zero padding) is the technique of lining the image with values of zeros. This padding alters the dimensioning effect of the filter. Zero padding is used to control the spatial sizes of the output volumes.

The spatial size of the output volume as a function of the input volume size (W) can be calculated as follows. Considering the receptive field size of the convolutional layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border, the spatial size of the output is given by $(W - F + 2P)/S + 1$. We can see this effect in figure 10.

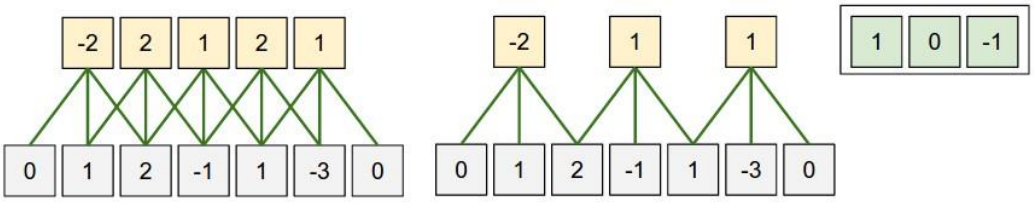

Figure 10: Output volume [37]

On the left side, $W = 5, P = 1, F = 3$ and $S = 1$ giving $(5 - 3 + 2 * 1)/1 + 1 = 5$. Similarly, the right has values such that $(5 - 3 + 2 * 1)/2 + 1 = 3$.

3.4.3 - The Pooling Layer

Pooling layers are typically inserted between successive convolutional layers in a CNN. Their function is to progressively reduce the spatial size of the representation of the image to reduce the number of parameters and computation in the network. This spatial reduction is also useful in controlling overfitting. The

pooling layer does not change the depth of the input, opeationg independently on each slice and resizing it spatially, typically using either the MAX or AVG operation. In figure 11 we can see a $2x2$ MAX pooling filter applied with a stride of 2.

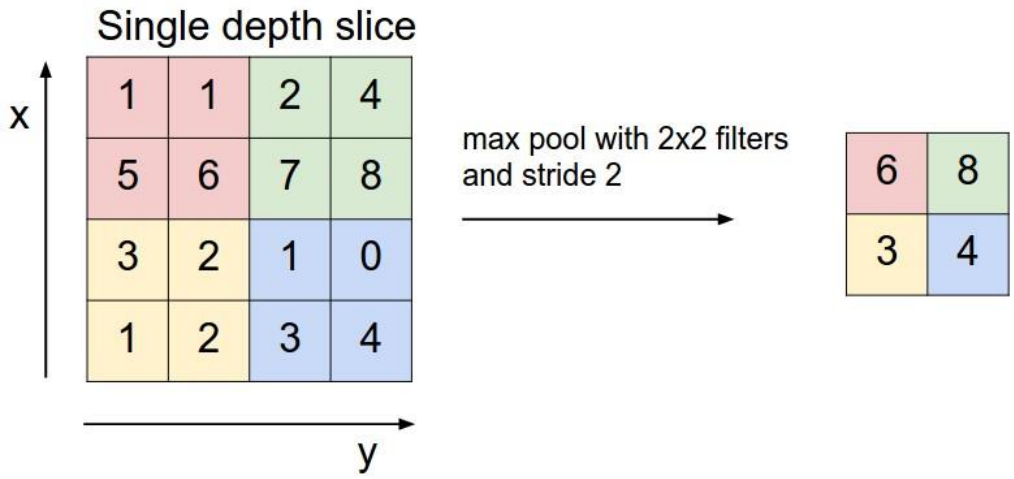

Figure 11: MAX Pooling [37]

3.4.4 - The Fully Connected Layer

The fully connected layer is used to learn non-linear combinations of the high-level features that are represented by the output of a previous convolutional layer. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in fully connected ANNs. Their activations can therefore be computed with a matrix multiplication followed by a bias offset.

The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer. The output of this layer is fed into a layer which uses an activation function such as Soft-max to generate a class prediction as seen in figure 12.

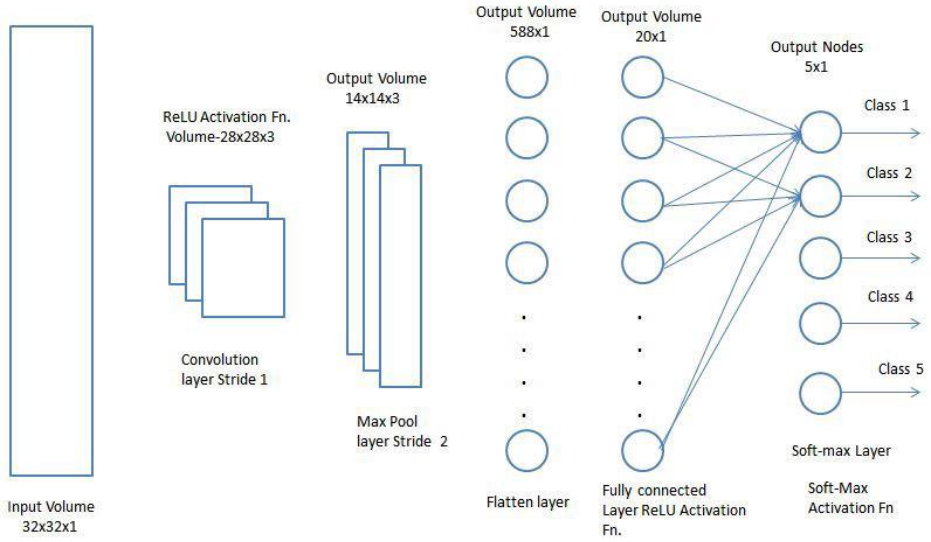

Figure 12: Fully Connected Layer with Soft-max [39]

3.5 – Techniques to Improve CNN performance

3.5.1 - Introduction

This section serves to introduce the techniques and methods employed in the attempts to improve performance.

3.5.2 – Data Augmentation

As we have seen, deep learning algorithms would ideally dispose of large amounts of data to train on. In practice, large amounts of data are not always readily available. To mitigate this issue, data augmentation can be applied to data sets to re-use data while minimising the possibility of over-fitting.

In the context of image data, data augmentation consists of altering the images in such a way that they can be re-used in training without simply repeating the exact same image. Pytorch's TorchVision library offers a set of functions, Transforms, which can be applied to images on the fly. Generally, they are probabilistic and are applied to a sub-set of images in each batch. Some common transformations and those used in this project are described below.

RandomRotation

Given degrees as a parameter, this function rotates the image by a random value with the range of [-degrees, +degrees].

RandomVerticalFlip

Vertically flip the image randomly with the given probability.

RandomHorizontalFlip

Horizontally flip the image randomly with the given probability.

RandomGreyscale

Randomly convert image to grayscale with a probability passed as a parameter.

3.5.3 – Batch Normalization

Deep learning models are generally formed by various layers with each one receiving an input, applying some computations, and passing the result on to the next layer as output. As the model is trained, the intention is that each layer becomes progressively better at fitting the input data distribution.

In practice, each batch of data will contain variations in its input distribution, which will not only be more challenging to fit, but also lead the model to adjust to that specific distribution. As well as the input distributions changing, as the model is changed and parameters adjusted, the distribution of a layer's input will be altered and modified by adjustments made to the parameters in the prior layer. As a result, the model is faced with the challenge of extrapolating the underlying distributions of the data from batch distributions which are in constant flux, a phenomenon known as internal covariate shift.

Batch Normalization seeks to alleviate these issues by scaling the output of each layer [40]. It does so by standardizing the activations of each input variable per batch, such as the activations of nodes from the previous layer. Standardizing the

activations of the prior layer means that assumptions the subsequent layer makes about the spread and distribution of inputs during the weight update will not change dramatically. The desired effect is the stabilizing and speeding-up of the training process of deep neural networks.

3.5.4 – L2 Regularization

L2 regularization is a method for preventing overfitting. It works by penalising complex models by regularizing the weights of the features. If we consider model complexity as a function of weights, a feature weight with a high absolute value is more complex than a feature weight with a low absolute value.

We can quantify complexity using the L2 regularization formula, which defines the regularization term as the sum of the squares of all the feature weights:

$$L2\ Regularization\ term = \|w\|_2^2 = w_1^2 + w_2^2 + w_3^2 + \cdots + w_n^2.$$

Given this formula, low weights will have little effect on model complexity while larger outlier weights will have a large affect. L2 regularization helps drive outlier weights (those with high positive or low negative values) closer to 0.

The loss function of a model including L2 regularization is:

$$Loss = error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

Where lambda is the regularization rate. L2 regularization has the following effects on a model:

- Encourages weight values toward 0 (but not exactly 0)
- Encourages the mean of the weights toward 0, with a normal (bell-shaped or Gaussian) distribution.

3.5.5 – Dropout

Dropout is a computationally inexpensive technique for avoiding overfitting. The key idea is to randomly drop nodes (along with their connections) from the neural network during training [41].

In Pytorch, a layer is added with a probability of nodes being zeroed. In a layer with 100 nodes, a probability of 0,5 would result in roughly 50 nodes being ignored in the forward and backwards passes of training. A visual representation of dropout is seen in figure 13.

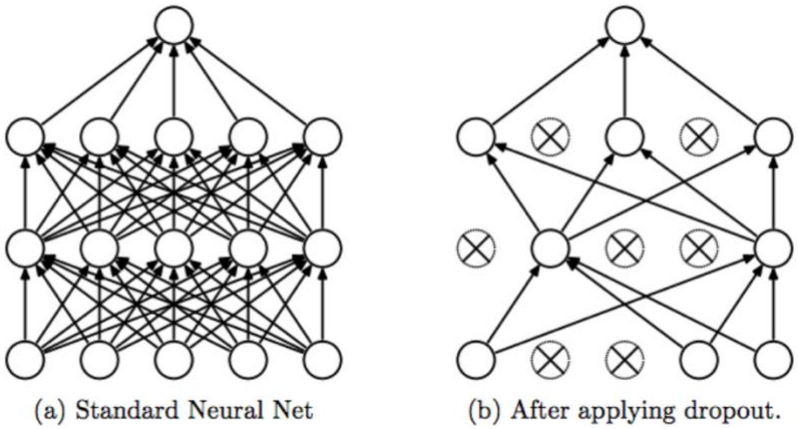

Figure 13: Dropout [41]

The process in the training phase is the following: for each hidden layer, for each training sample, for each iteration, zero out a random fraction, p, of nodes (and corresponding activations) and calculate the loss without the influence of those zeroed nodes.

3.5.6 – Transfer Learning and ResNet18

Transfer Learning is the technique of taking advantage of already trained models to solve new problems. It consists of adapting the models already created so that they work with the new scenario. Therefore, the problem for which the original model was created and the new problem to be solved must be related. We call the model from what we want to learn 'source model' and the model we want to teach from the source model 'target model'.

In this project, a pretrained ResNet18 model from Pytorch will be used. In section 3.3.3 we commented on backpropagation. During the backpropagation stage, the error is calculated, and gradient values are determined. The gradients are sent back to hidden layers and the weights are updated accordingly. This process of gradient calculation and weight adjustment is calculated until the input layer is reached.

As more layers are added to a neural network, backpropagation through many layers of very deep models can lead to gradient vanishing or explosion [42, 43]. Gradient vanishing is the process in which the gradients calculated get progressively smaller as they are passed through the layers until the adjustments made to weights become negligible. This leads to weights not being adjusted sufficiently in shallow layers of the model, and convergence either taking a very long time, or simply becoming impossible. Gradient explosion is essentially the opposite problem. Here, gradients accumulate, and weights are over-adjusted. In either case, very deep models can be hard to train.

To combat this effect, He et al proposed a residual learning framework to ease the training of networks that are substantially deeper than those used previously [44]. The idea is that instead of letting layers learn the underlying mapping, let the network fit the residual mapping. Instead of learning the initial mapping $H(x)$, let the network fit $F(x) = H(x) - x$ which gives $H(x) = F(x) + x$. The approach involves adding an identity connection allowing data to pass uninterrupted, skipping weighted layers, as well as through the same weighted layers [45] as seen in figure 14.

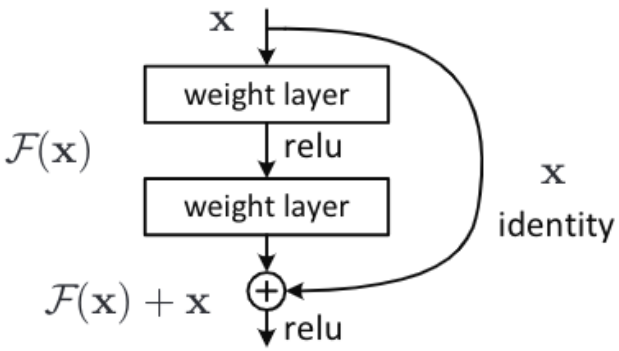

Figure 14: A Residual Block [44]

In the paper by He et al, the built a 34-layer ResNet. In this project, an 18-layer version as seen in figure 15 is used.

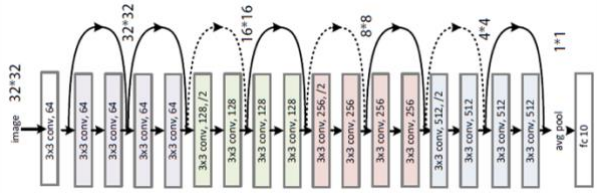

Figure 15: ResNet18 [46]

The construction of this model used two blocks. The first block is the identity block as seen in figure 16. The identity block is the standard block used in ResNet and corresponds to the case where the input activation has the same dimension as the output activation.

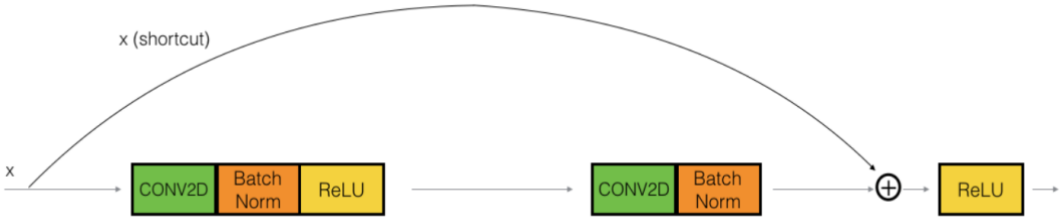

Figure 16: Identity Block [45]

The second block is the Conv Block as seen in figure 17. The conv block serves to modify and restructure the incoming data so that the output of the first layer matches dimensions of the third layer.

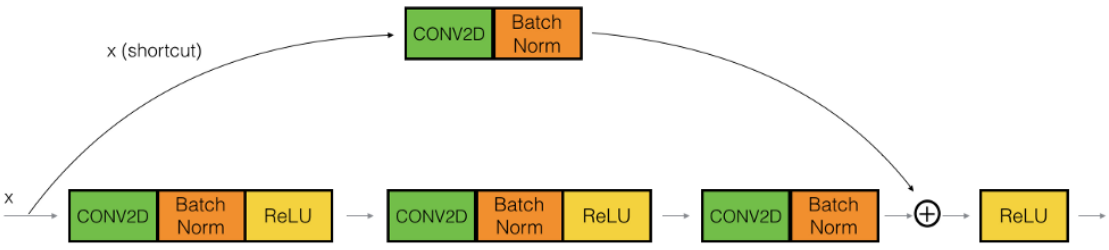

Figure 17: Conv Block [45]

Using this structure, the authors won 1st place in ILSVRC and COCO 2015 competition in ImageNet Detection, ImageNet localization, Coco detection and Coco segmentation.

This project uses Pytorch's ResNet 18 model: the same structure but made up of only 18 layers. The model employed is pre-trained on the ImageNet data set and will be fine-tuned on the radiological images from this study's dataset.

3.5.8 – Other Transfer Learning Options

The options considered were chosen to allow reasonable comparison between the results obtained here and results obtained in the paper by Minaee et al [12]. In this paper, ResNet18, ResNet54 and SqueezeNet and DenseNet. In their paper, ResNet18 achieved scores that are very comparable with the other models employed and was also shown to be effective and have a short training time in classification tasks concerning radiographical images [47].

3.6 – Metrics to Evaluate Performance

To measure the ability of a model's ability to classify samples, many metrics can be used. In this section, the metrics used in this project are outlined.

Accuracy

Accuracy is the expresses the number of correct classifications. It is the number of correct classifications divided by the total number of samples to classify, generally expressed as a percentage.

This metric, while useful, does not give a nuanced idea of the model's performance. If a dataset is imbalanced, for example, high accuracies can be achieved despite misclassifying many samples from the small class. To fully evaluate models, other metrics must also be considered.

Precision, Recall and F1

In a classification problem, we can define predictions as positives and negatives. For example, in this multi-class classification problem with classes Covid, Normal and Viral Pneumonia, a sample classified as Covid would be considered a positive prediction for Covid, and negative for Normal and Viral Pneumonia.

Given this definition for positive and negative predictions, we can define a true positive (TP) as a positive prediction that was in fact positive in the predicted class. For example, a correctly identified Viral Pneumonia sample is a true positive in that class and a true negative (TN) in the other classes. On the other hand, an image from the Covid class that is incorrectly classified as Normal is a false positive (FP) of the Normal Class, and a false negative (FN) of the Covid class.

Precision, also known as positive predictive value, is the proportion of correct positive predictions. It can be expressed as a decimal number less than one, or indeed as a percentage. The formula is the following:

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

Recall, also known as sensitivity, is the proportion of actual positives that were correctly identified. It can be expressed in the same format as precision and is defined in the following way:

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

Both metrics are useful in evaluating a model. For example, in the case of an infectious disease like Covid-19, recall is useful in measuring how many positive samples are undetected.

The F1 score is the harmonic mean of the precision and recall, it allows us to get a balanced understand of a model's performance on a class. The F1 score informs us of the balance between the precision and recall and allows us to evaluate by class. The formula is the following:

$$F1 = 2 \cdot \frac{Precsion \cdot Recall}{Precision + Recall}$$

Weighted and Macro Averages

For each of the three metrics in the last section (Precision, Recall and F1), we will calculate a weighted and macro average for each one. As the scores are calculated by class, these averages express the score in general terms.

The weighted average considers the dimension of each class and weights its contribution to the average accordingly. The formula is the following:

$$weightedAverage_{score} = \frac{\%weight(class_0)}{100} \cdot score(class_0) +$$

$$\frac{\%weight(class_1)}{100} \cdot score(class_1) + .. + \frac{\%weight(class_n)}{100} \cdot score(class_n)$$

Where the %weight proportion is the proportion that class represents.

The macro average does not consider the weights of the classes, instead each class score contributes equally to the average:

$$macroAverage_{score} = \frac{numberClasses}{100} \cdot score(class_0) +$$

$$\frac{numberClasses}{100} \cdot score(class_1) + .. + \frac{numberClasses}{100} \cdot score(class_n)$$

4. – Methodology

4.1 – The data

This project employs a dataset of 15,153 radiographic images containing three classes: COVID-19, Normal and Viral Pneumonia. The classes are of the dimensions shown in table 4.

Class	Number of images	Percentage of Total
COVID-19	3,616	23.86%
Normal	10,192	67.26%
Viral Pneumonia	1,345	8.88%

Table 4: Dataset dimensions

This dataset is a subset of the publicly available *COVID-19 Radiography database* [48]. Each image is either AP (anterior to posterior) or PA (posterior to anterior) picturing the thorax including the lungs. Every image employed in this project was collected by researchers in universities in Pakistan, Bangladesh, Malaysia, and Qatar. The dataset was assembled as part of a study by Chowdhury et al [9] and is available publicly on Kaggle [49].

In the following sections, the sources used by the researchers to compile the data are listed along with the number of images taken from each.

4.1.1 - COVID-19 Class

The images in this class were taken from six different sources. In each, patients were confirmed as positive for the coronavirus by either PCR or antigens test. The sources of this class can be seen in table 5.

Source	Number of images taken
SIRM, The Italian Society of Medical and Interventional Radiology **[50]**	199
COVID-19 Image Repository (Github) – Hannover Medical School **[51]**	183
EuroRad - European Society of Radiology **[52]**	258
COVID-CXNet - Github **[53]**	400
Covid ChestXray Dataset - Github **[54]**	182
BIMCV - Medical Imaging Databank of the Valencia Region **[55]**	2474

Table 5: COVID-19 Class Images Sources

4.1.2 – Normal Class

The normal lung image set is made up of a total of 10,192 X-rays taken from two sources as seen in table 6.

Source	Number of images taken
RSNA Pneumonia Detection Challenge - Kaggle **[56]**	8,851
Chest X-Ray Images (Pneumonia) **[57]**	1,341

Table 6: Normal Class Image Sources

4.1.3 – Viral Pneumonia Class

The Viral Pneumonia folder contains a total of 1345 images all of which come from the same source as noted in table 7.

Source	Number of images taken
Chest X-Ray Images (Pneumonia) **[57]**	1,345

Table 7: Viral Pneumonia Class

4.1.4 – Further Details

According to the researchers, the *COVID-19 Radiography database* [48] has been published on Kaggle as part of an effort to make clinically useful COVID-19 imagery widely available. The authors request the citation of their two papers that have used this set [9], [59]. The cited studies have used the previous iterations of this dataset, which is currently on version 4.

It is important to note that, although the data was published online by the same researchers responsible for compiling the data used in the papers [9], [59], the images used in our study are distinct from those currently available in the Kaggle dataset. At the time of publication of these papers, there were less images available. The dataset compiled by the authors, therefore, consisted of less images despite being made up of data from the same sources listed above. This project uses the data which is currently available to capitalize on the methodological benefits that this increased sample size affords. The author considers that, while the larger quantity of images may make comparisons between results obtained in this project and those of published papers less direct, the flux in dimensions reflects the nature of a pandemic where both the situation and the available data are in constant evolution.

4.1.5 – Data exploration

All the images are in Portable Network Graphics (PNG) file format and have been resized to a resolution of 299·299 pixels. Each image is either PA or AP, no lateral images have been included.

Three images of each class are displayed in each row in figure 13:

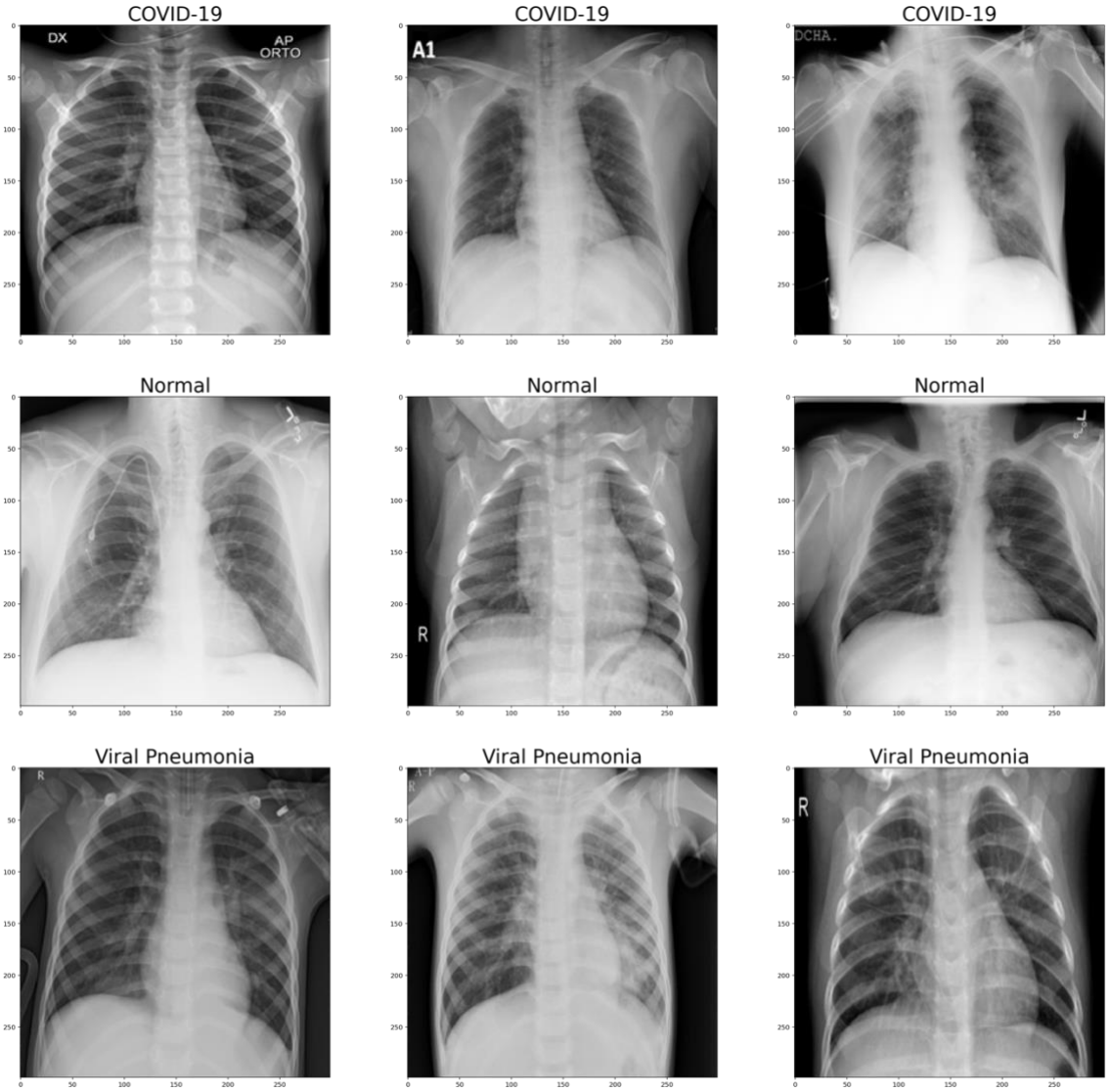

Figure 13: Classes of images

4.2 – Training, Validation and Testing Set Preparation

The dataset was split into training, validation and testing sets with a proportion of 79%, 15% and 6% respectively. This resulted in the dimensions in table 8.

Training set	**11,964 images**	**79%**
Validation set	2,279 images	15%
Training set	910 images	6%

Table 8: Set dimensions

Splitting the dataset into the three subsets indicated in table 7 was not a as straightforward as simply dividing the images in the proportions indicated. The dataset is considerably imbalanced as seen in figure 13. Images of the Normal class make up 67.26% of the total, while the Covid and Viral Pneumonia classes account for 23.86% and 8.88% respectively.

Apart from this imbalance, the images of the Normal and Covid classes were collected from various repositories. If images from a single repository were to

share a characteristic owing to the repository (average age of patient of the hospital where the images were taken, calibration of radiographic machine used to capture images, etc), inadvertently grouping images from the same repository in the same subset may accidently introduce similarities into the set. To avoid this, we must ensure that the subsets created are not only proportional to the original class imbalance, but also shuffled randomly to minimise the inadvertent characteristic effect described above.

To solve these problems, the Scikit-learn [60] library was used in conjunction with Pytorch's TorchVision ImageFolder to create three directories of images, each containing a subset of images the classes. Images were added to these folders in a shuffled manner to avoid images of the same source being grouped together as much as possible. The distribution of these subsets can be seen in figure 15.

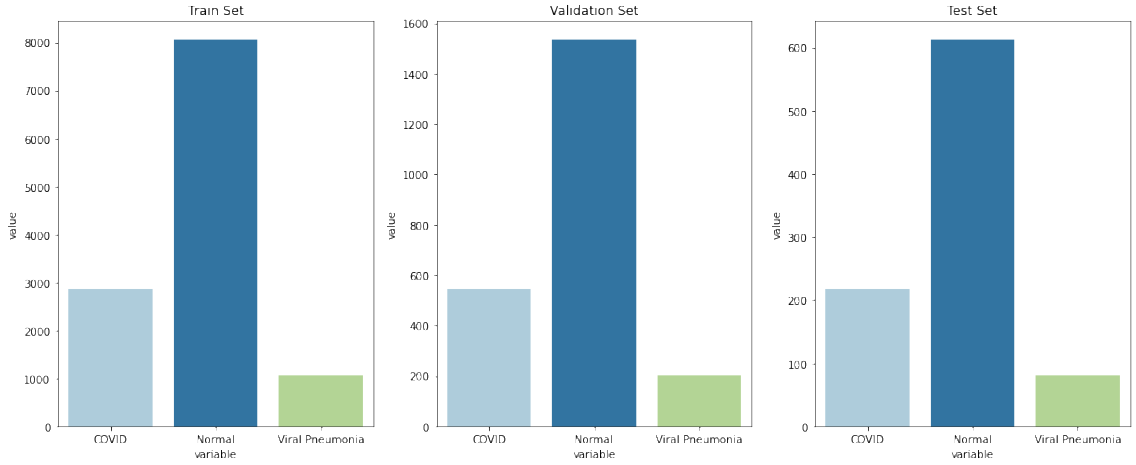

Figure 15: Training, Validation and Testing Set Distribution

The exact numbers of each subset are shown in table 9.

Set	Covid	Normal	Viral Pneumonia
Training	2,856	8,048	1,063
Validation	545	1,534	203
Testing	217	612	81

Table 9: Dimensions of Subsets

4.3. – Experiments

For the original experiments at the beginning of the project, several simple architectures were tested before a sufficiently performant model was chosen as a starting point. As the project progressed, both the model and the data were modified in the attempt to improve results. The following sections outline these modifications and the theory underlying them.

4.3.1 – Experiment 1 Architecture and Configuration

The base model used in this project is a simple architecture consisting of three convolutional layers, each followed by a ReLU activation layer and a pooling layer. The output of the third pooling layer is flattened, the output of which is the input to a fully connected layer. The output of this first fully connected layer is the input to a second fully connected layer, the output layer. The input to this architecture is 299x299x3.

The size of the kernel in each of the convolutional layers is 5, with a stride of 1 and no padding. The pooling layers use a filter of size 2x2 and a stride of 1. We can visualize this architecture more clearly in the form of a table:

Layer type	**Output**	**Kernel/Filter**	**Stride**	**Padding**
Convolutional	(295,295,32)	5	1	0
ReLU	(295,295,32)	N/A	N/A	N/A
Max Pooling	(147,147,32)	2x2	1	0
Convolutional	(138,138,64)	5	1	0
ReLU	(138,138,64)	N/A	N/A	N/A
Max Pooling	(69,69,64)	2x2	1	0
Convolutional	(50,50,128)	5	1	0
ReLU	(50,50,128)	N/A	N/A	N/A
Max Pooling	(25,25,128)	2x2	1	0
Flatten	(80,000, 1)	N/A	N/A	N/A
Fully Connected	(128, 1)	N/A	N/A	N/A
Output	(3, 1)	N/A	N/A	N/A

Table 10: Experiment 1 Architecture

This base architecture is used to obtain the results of the first experiment and, excluding experiment 6 which uses a ResNet architecture, is the foundation upon which all posterior experiments are built. This experiment is simple and will serve the purpose of being a baseline for the rest of the experiments.

The configuration of the model can be seen in table 11.

Image shape	**299·299 pixels**
Transformed to tensor	yes
Normalized	yes
Total Epochs trained:	20
Lowest Validation Loss Epoch:	20
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	32
Data Augmentation	No
Transfer learning	No
Batch Normalization	No
L2 Regularization	No
Drop Out	No

Table 11: Experiment 1 Configuration

4.3.2 – Experiment 2 Architecture and Configuration

This experiment uses the same architecture as experiment 1, the only change is in the preparation of the data. Here we apply data augmentation to simulate a larger data set. The configuration of this experiment is the following:

Image shape	299·299 pixels
Transformed to tensor	yes
Normalized	yes
Total Epochs Trained	40
Lowest Validation Loss Epoch	32
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	32
Data Augmentation	Yes
RandomGrayscale	0.05 probability
RandomVerticalFlip	0.08 probability
RandomRotation	±10 degrees
Transfer learning	No
Batch Normalization	No
L2 Regularization	No
Drop Out	No

Table 12: Experiment 2 Configuration

These transformations, although not extreme, serve to allow the model to be trained for more epochs without overfitting. This allows us to emulate experiments seen in the literature in which data augmentation is used to improve results despite the limited number of clinical Covid-19 X-ray images available.

4.3.3 – Experiment 3 Architecture and Configuration

This experiment investigates the effect of batch normalization (section 3.5.3) used in conjunction with data augmentation.

Three batch normalization layers are added to the architecture employed in experiment 1 as well as maintaining the use of data augmentation from experiment 2. A batch norm layer is added after each convolutional layer. The first batch normalization layer has a momentum of 0.04, while the second and third have a momentum of 0.06 each. This architecture can be visualized in the form of a table:

Layer type	**Output**	**Kernel/Filter**	**Stride**	**Padding**
Convolutional	(295,295,32)	5	1	0
BatchNorm (0.04)	(295,295,32)	N/A	N/A	N/A
ReLU	(295,295,32)	N/A	N/A	N/A
Max Pooling	(147,147,32)	2x2	1	0
Convolutional	(138,138,64)	5	1	0
BatchNorm (0.06)	(138,138,64)	N/A	N/A	N/A
ReLU	(138,138,64)	N/A	N/A	N/A
Max Pooling	(69,69,64)	2x2	1	0
Convolutional	(50,50,128)	5	1	0
BatchNorm (0.06)	(50,50,128)	N/A	N/A	N/A
ReLU	(50,50,128)	N/A	N/A	N/A
Max Pooling	(25,25,128)	2x2	1	0
Flatten	(80,000, 1)	N/A	N/A	N/A
Fully Connected	(128, 1)	N/A	N/A	N/A
Output	(3, 1)	N/A	N/A	N/A

Table 13: Experiment 3 Architecture

The configuration of this experiment can be seen in table 14.

Image shape	299·299 pixels
Transformed to tensor	yes
Normalized	yes
Total Epochs Trained	25
Lowest Validation Loss Epoch	25
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	124
Data Augmentation	Yes
RandomGrayscale	0.05 probability
RandomVerticalFlip	0.08 probability
RandomRotation	±10 degrees
Transfer learning	No
Batch Normalization	Yes
L2 Regularization	No
Drop Out	No

Table 14: Experiment 3 Configuration

4.3.4 – Experiment 4 Architecture and Configuration

Experiment 4 investigates the effect that L2 regularization (section 3.5.4) can have when used in conjunction with the techniques employed in experiment 3.

In this experiment, the architecture from experiment 3 was recycled and L2 regularization was also employed. To achieve L2 regularization in Pytorch, weight decay must be added to the optimizer used [61]. The weight decay was set to a value of 1^{-6}. The configuration for this experiment is shown in table 15.

Image shape	299·299 pixels
Transformed to tensor	yes
Normalized	yes
Total Epochs Trained	30
Lowest Validation Loss Epoch	24
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	124
Data Augmentation	Yes
RandomGrayscale	0.05 probability
RandomVerticalFlip	0.07 probability
RandomRotaton	±8 degrees
Transfer learning	No
Batch Normalization	Yes
L2 Regularization	Yes
Drop Out	No

Table 15: Experiment 4 Configuration

4.3.5 – Experiment 5 Architecture and Configuration

Experiment 5 investigates another type of regularization effect, in this case using dropout (section 3.5.5). The architecture employed in this experiment includes batch normalization and data augmentation is also used. Two dropout layers are added to the architecture, after the second and third batch normalization layers.

As explained in section 3.5.5, these layers are given a probabilistic value to define how many nodes are zeroed out, 0.08 in the first dropout layer, 0.10 in the second. We can appreciate this structure visually in table 16.

Layer type	**Output**	**Kernel/Filter**	**Stride**	**Padding**
Convolutional	(295,295,32)	5	1	0
BatchNorm (0.04)	(295,295,32)	N/A	N/A	N/A
ReLU	(295,295,32)	N/A	N/A	N/A
Max Pooling	(147,147,32)	2x2	1	0
Convolutional	(138,138,64)	5	1	0
BatchNorm (0.06)	(138,138,64)	N/A	N/A	N/A
Dropout (0.08)	(138,138,64)	N/A	N/A	N/A
ReLU	(138,138,64)	N/A	N/A	N/A
Max Pooling	(69,69,64)	2x2	1	0
Convolutional	(50,50,128)	5	1	0
BatchNorm (0.06)	(50,50,128)	N/A	N/A	N/A
Dropout (0.10)	(50,50,128)	N/A	N/A	N/A
ReLU	(50,50,128)	N/A	N/A	N/A
Max Pooling	(25,25,128)	2x2	1	0
Flatten	(80,000, 1)	N/A	N/A	N/A
Fully Connected	(128, 1)	N/A	N/A	N/A
Output	(3, 1)	N/A	N/A	N/A

Table 16: Experiment 5 Architecture

The configuration of this experiment can be seen in the following table:

Image shape	299·299 pixels
Transformed to tensor	yes
Normalized	yes
Total Epochs Trained	30
Lowest Validation Loss Epoch	26
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	124
Data Augmentation	Yes
RandomGrayscale	0.05 probability
RandomVerticalFlip	0.07 probability
RandomRotaton	±8 degrees
Transfer learning	No
Batch Normalization	Yes
L2 Regularization	No
Drop Out	Yes

Table 17: Experiment 5 Configuration

4.3.6 – Experiment 6 Architecture and Configuration

In this experiment the ResNet18 architecture described in section 3.6 is used. The pretraining of this model was carried out on ImageNet [62], a dataset of more than 14 million images.

Here, we change the last layer so that it has three outputs and fine tune its parameters over our images for 20 epochs. It is also necessary to normalise the inputs with the following means and standard deviations:

- Means: 0.485, 0.456, 0.406
- Standard Deviations: 0.229, 0.224, 0.225

The configuration for this experiment is shown in table 18.

Image shape	299·299 pixels
Transformed to tensor	Yes
Normalized	Yes
Epochs	20
Loss Function	Cross Entropy Loss
Optimizer	Stoic Gradient Decent
Learning rate	0.0001
Momentum	0.9
Batch Size	124
Data Augmentation	No
Transfer learning	Yes
Batch Normalization	No
L2 Regularization	No
Drop Out	No

Table 18: Experiment 6 Configuration

5. – Results

5.1 – Introduction

This section is a summary of the results obtained in each iteration of the project. Graphics show the evolution of loss and accuracy in the training and validation sets during the training of the model. The pipeline is such that the state of the model at the point of the lowest validation loss is saved, and the test results are obtained from that model.

There will also be a summary of the performance of the model on the testing set using the metrics mentioned in section 3.7.

5.2 – Experiment 1

5.2.1 – Training

In figure 18, we can see the evolution of the two loss functions over the epochs of training. The training loss is used to modify the parameters of the CNN through back propagation while the validation loss is simply an evaluation of the network's performance over time. By saving the model's state at the lowest validation loss, we can ensure that we have the most performant state at the end of the experiment.

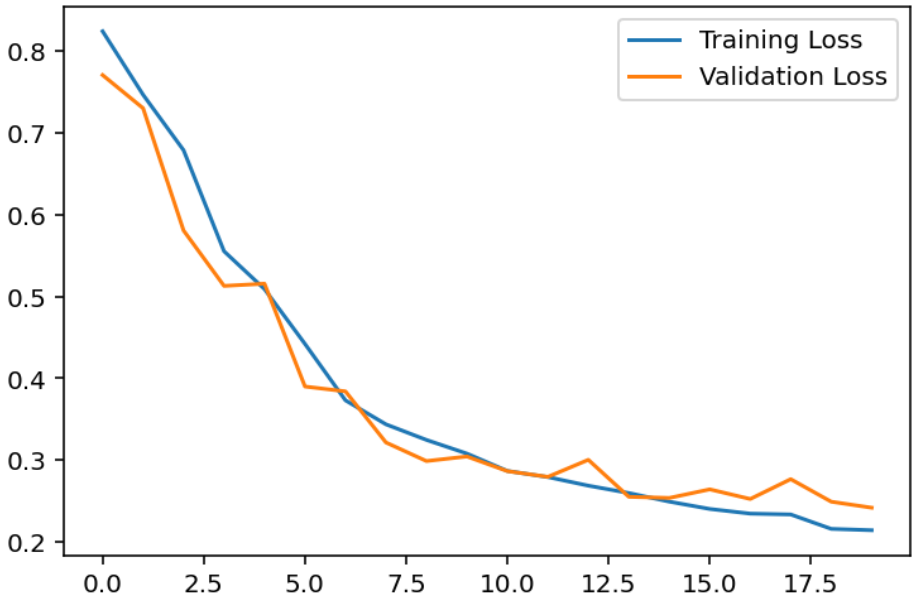

Figure 18: Experiment 1 Losses

In figure 19 we can see the accuracy with which the training and validation images have been classified over the 20 epochs.

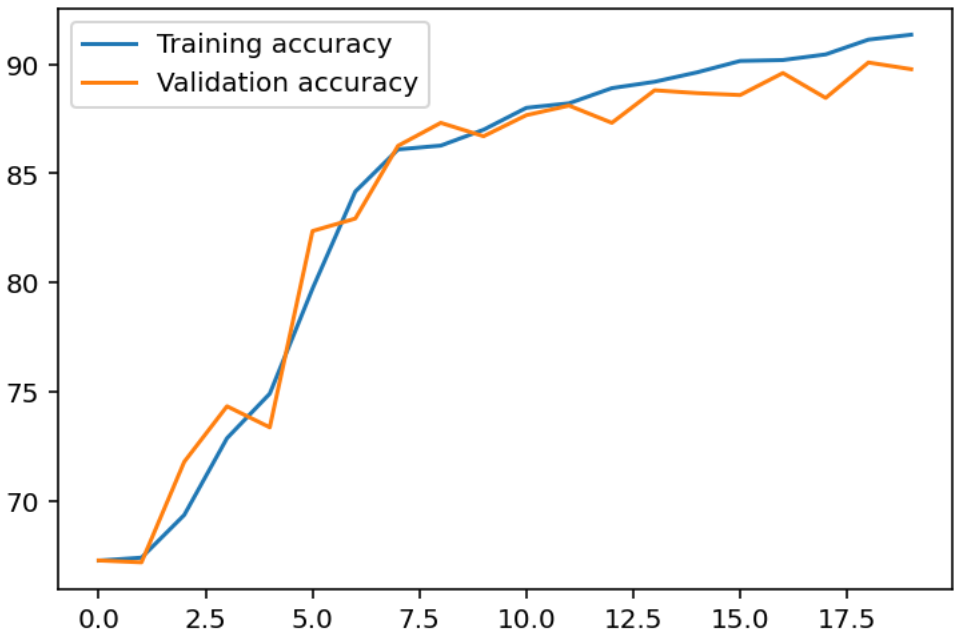

Figure 19: Experiment 1 Accuracies

5.2.2 – Evaluation

In figure 20 we can see the confusion matrix of the predictions made by the trained model.

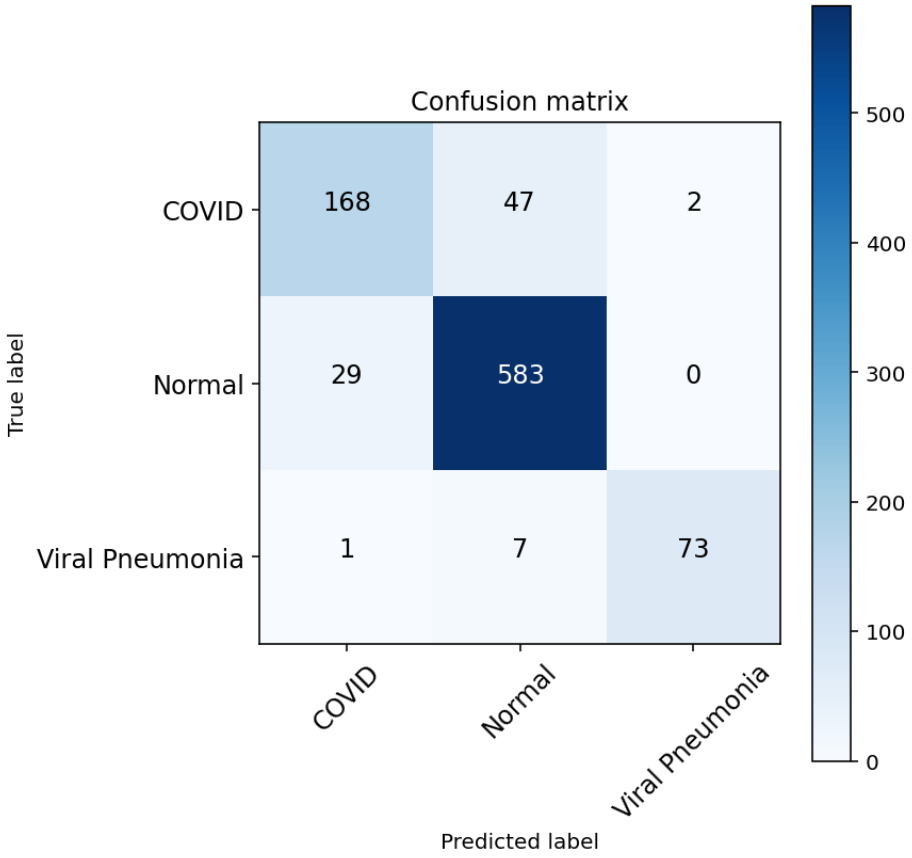

Figure 20: Experiment 1 Confusion Matrix

The classification report can be seen in table 19.

	Precision	Recall	F1-score	Support
COVID	84.85%	77.42%	80.96%	217
Normal	91.52%	95.26%	93.35%	612
Pneumonia	97.33%	90.12%	93.59%	81
Accuracy			90.55%	910
Macro avg	91.23%	87.60%	89.30%	910
Weighted avg	90.45%	90.55%	90.42%	910

Table 19: Experiment 1 Classification Report

824 of 910 images have been classified correctly resulting in an accuracy of 90.55%. Table 19 shows metrics by class and the macro and weighted averages mentioned in the methodology section.

5.3 – Experiment 2

5.3.1 – Training

This experiment is the base model with augmented data. The expectation is that the variance introduced in each epoch slows down overfitting. In figure 21, we see that the validation loss falls until almost the last epoch.

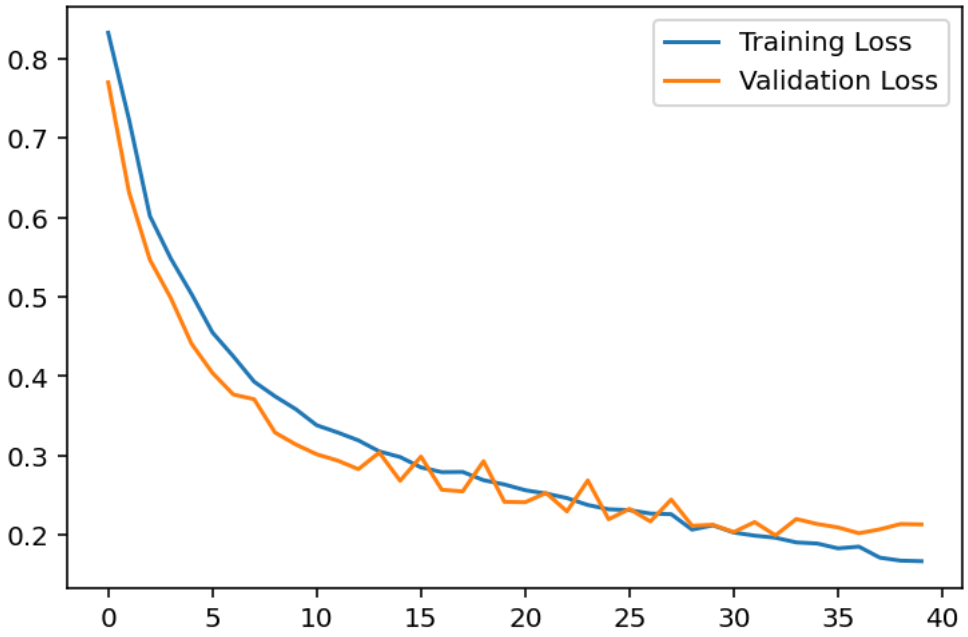

Figure 21: Experiment 2 Losses

Consistent improvements in both the training and validation accuracies can also be observed in figure 22.

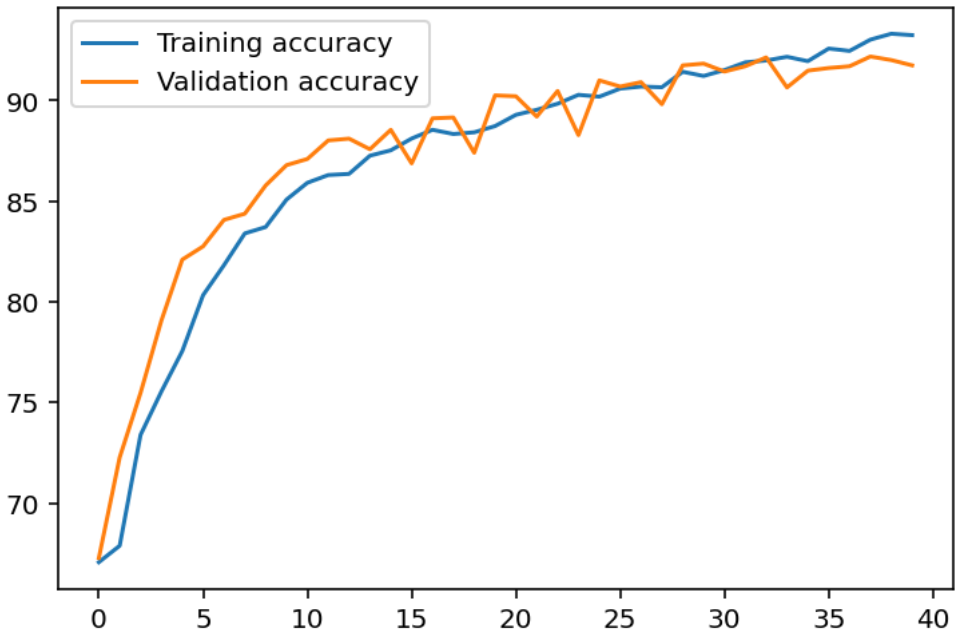

Figure 22: Experiment Accuracies

5.3.2 – Evaluation

In figure 23 we can see the confusion matrix for this model:

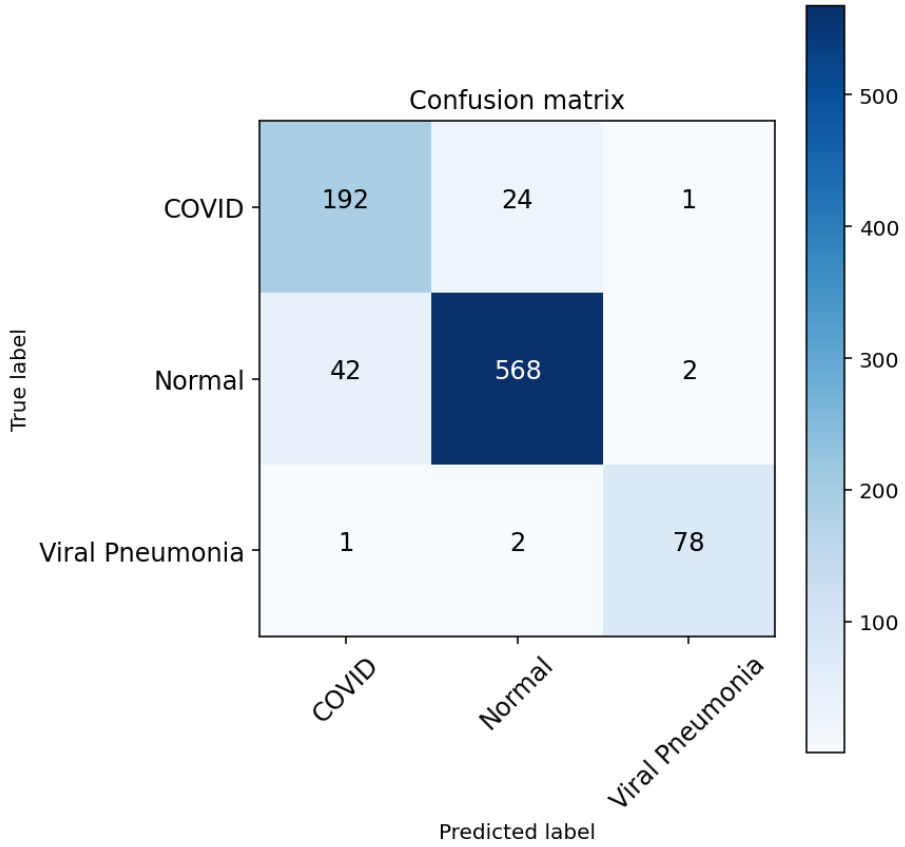

Figure 23: Experiment 2 Confusion Matrix

Looking along the diagonal, we can see that the model has correctly classified 838 of 910 images, giving an accuracy of 92.08%. Further metrics can be seen in table 20.

	Precision	**Recall**	**F1-score**	**Support**
COVID	81.70%	88.48%	84.96%	217
Normal	95.62%	92.81%	94.20%	612
Pneumonia	96.30%	96.30%	96.30%	81
Accuracy			92.09%	910
Macro avg	91.21%	92.53%	91.82%	910
Weighted avg	92.36%	92.09%	92.18%	910

Table 20: Experiment 2 Classification Report

This experiment has yielded high percentages in the pneumonia class. Percentages above 90% are also achieved on the Normal class, while the COVID class achieves, 81, 88 and 95 percent in precision, recall and F1 respectively.

5.4 – Experiment 3

5.4.1 – Training

In figure 24 we can see the loss of this model during training:

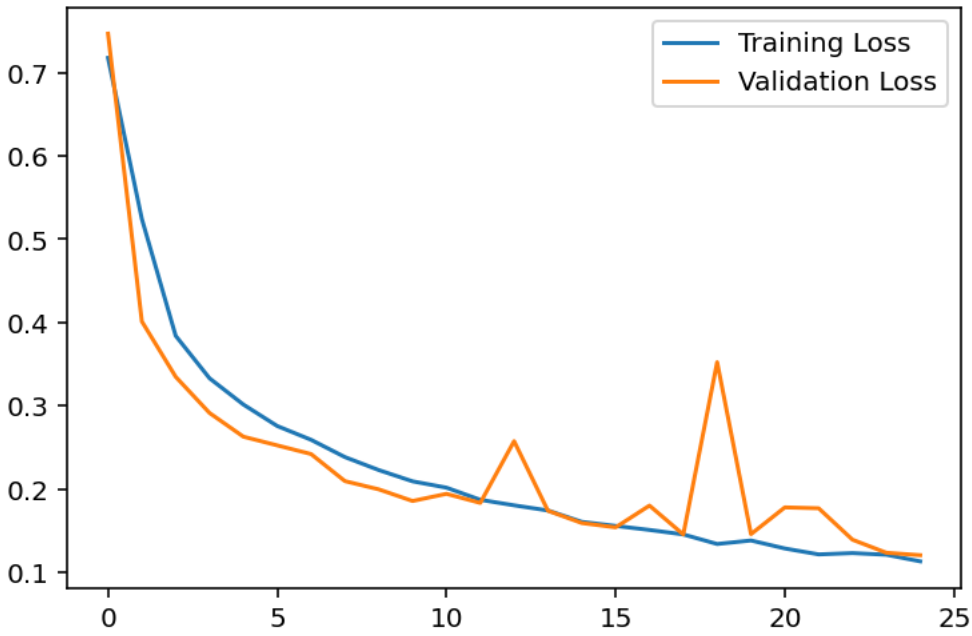

Figure 24: Experiment 3 Losses

Two erratic peaks can be seen roughly around the 12th and 17th epochs, but the two functions re-converge for the last several. The accuracy functions present similar tendencies as seen in figure 25:

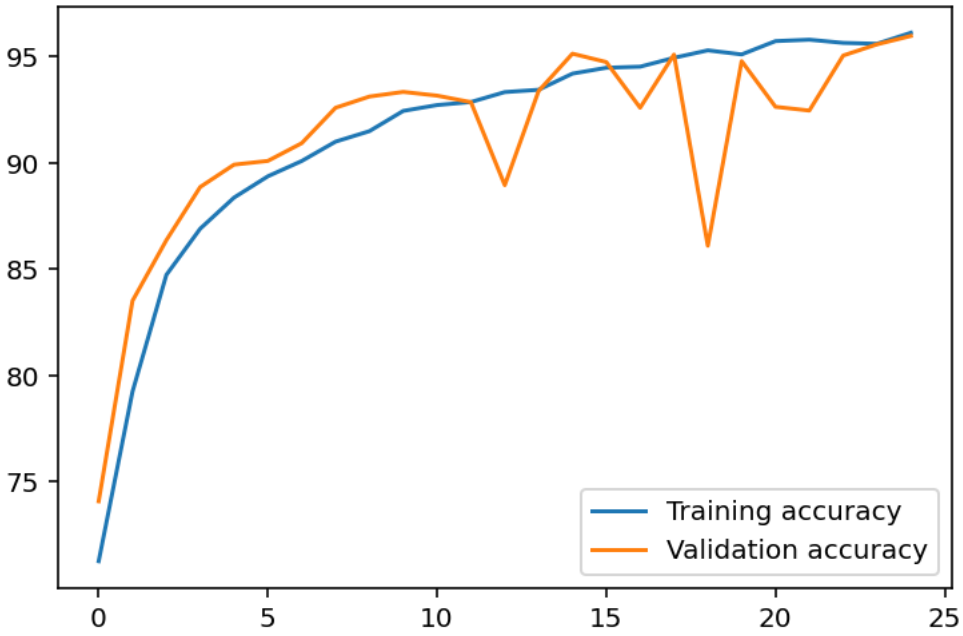

Figure 25: Experiment 3 Accuracies

Here the validation accuracy presents some erratic behaviour in the same epochs but reconverges with the training accuracy in the last number of epochs.

5.4.2 – Evaluation

An accuracy of 95% was achieved on the testing set, classifications can be seen in the confusion matrix in figure 26:

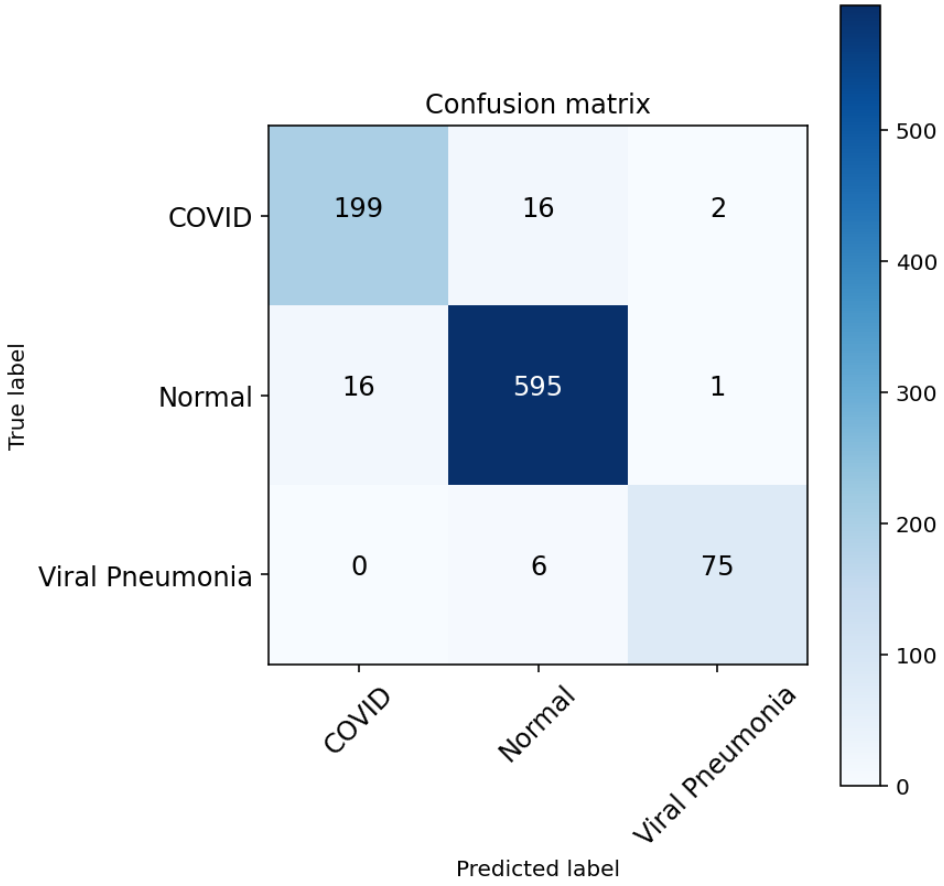

Figure 26: Experiment 3 Confusion Matrix

In this case, 869 of 910 images have been classified correctly, 95.49%. In table 21 further performance metrics can be seen:

	Precision	Recall	F1-score	Support
COVID	92.56%	91.71%	92.13%	217
Normal	96.43%	97.22%	96.83%	612
Pneumonia	96.15%	92.59%	94.34%	81
Accuracy			95.49%	910
Macro avg	95.05%	93.84%	94.43%	910
Weighted avg	95.49%	95.49%	95.49%	910

Table 21: Experiment 3 Classification Report

This model has performed well on each class. Both the precision and recall on the covid class are above 90%, meaning very few false negatives and positives.

5.5 – Experiment 4

5.5.1 – Training

Figure 27 shows the training and validation losses during training.

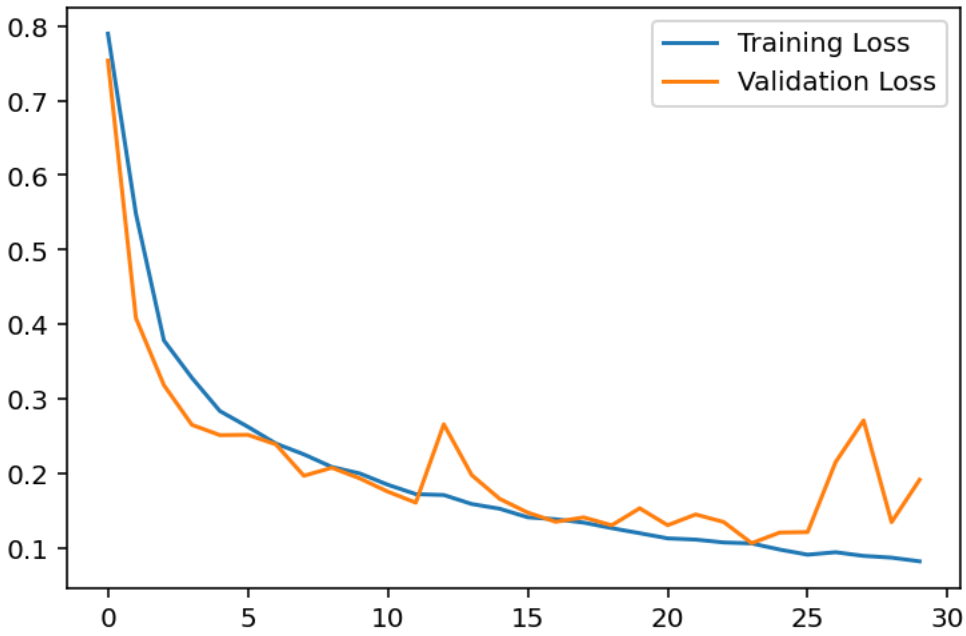

Figure 27: Experiment 4 Losses

The validation loss seems to destabilize in the last 5 epochs. A similar trend is seen in the accuracies in figure 28:

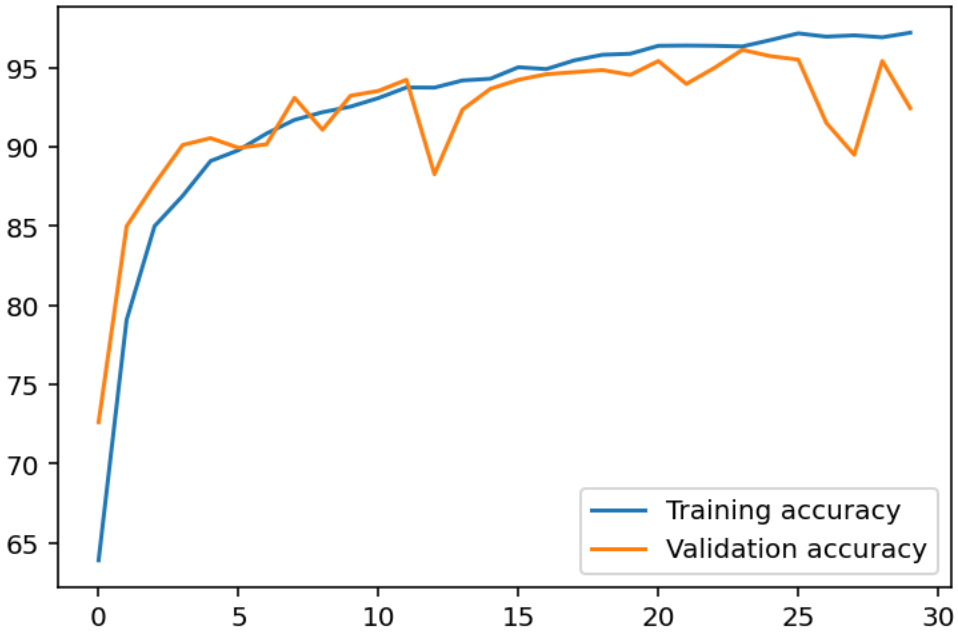

Figure 28: Experiment 4 Accuracies

5.5.2 – Evaluation

The confusion matrix of this model can be seen in figure 29:

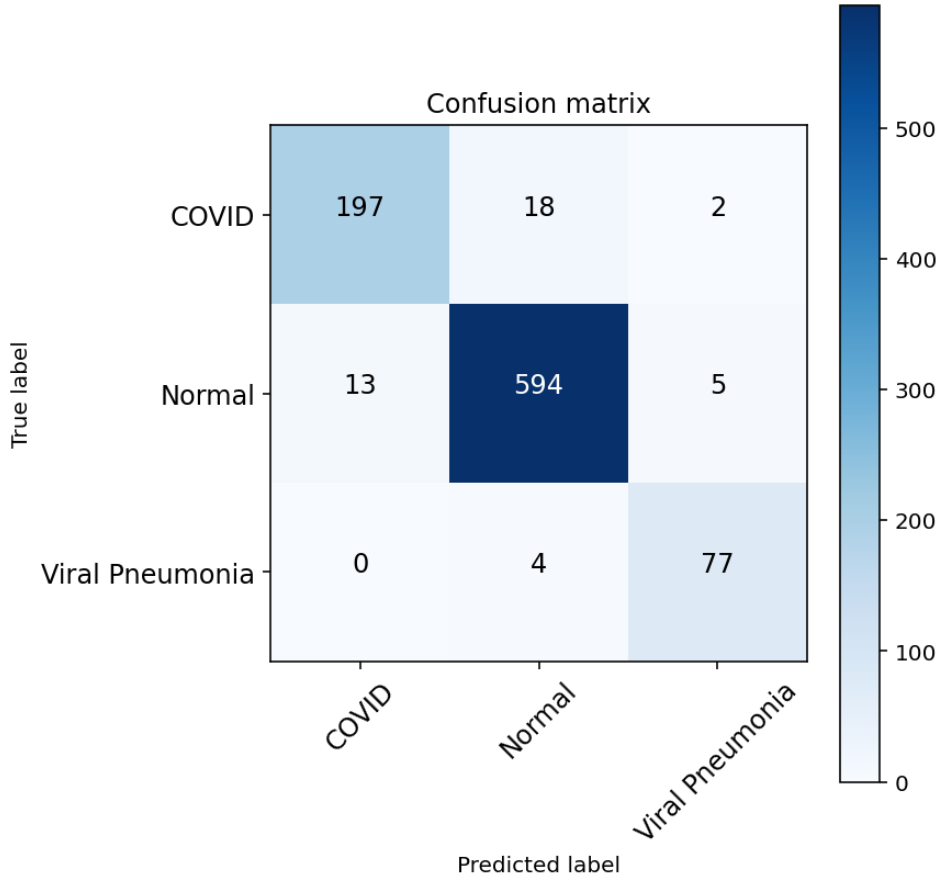

Figure 29: Experiment 4 Confusion Matrix

In figure 29 we see that 868 of 910 images have been correctly classified, giving an accuracy of 95.38%. We can have a closer look at the rest of the metrics in table 22.

	Precision	Recall	F1-score	Support
COVID	93.81%	90.78%	92.27%	217
Normal	96.43%	97.06%	96.74%	612
Pneumonia	91.67%	95.06%	93.33%	81
Accuracy			95.38%	910
Macro avg	93.97%	94.30%	94.12%	910
Weighted avg	95.38%	95.38%	95.37%	910

Table 22: Experiment 4 Classification Report

We can see that this model has also performed well across the three classes. Both precision and recall are high across classes, although the COVID class is just above 90%.

5.6 – Experiment 5

5.6.1 – Training

This experiment employs data augmentation, batch normalization and drop-out, drop-out layers were used, as detailed in section 4.3.5.

The losses during training can be seen in figure 30.

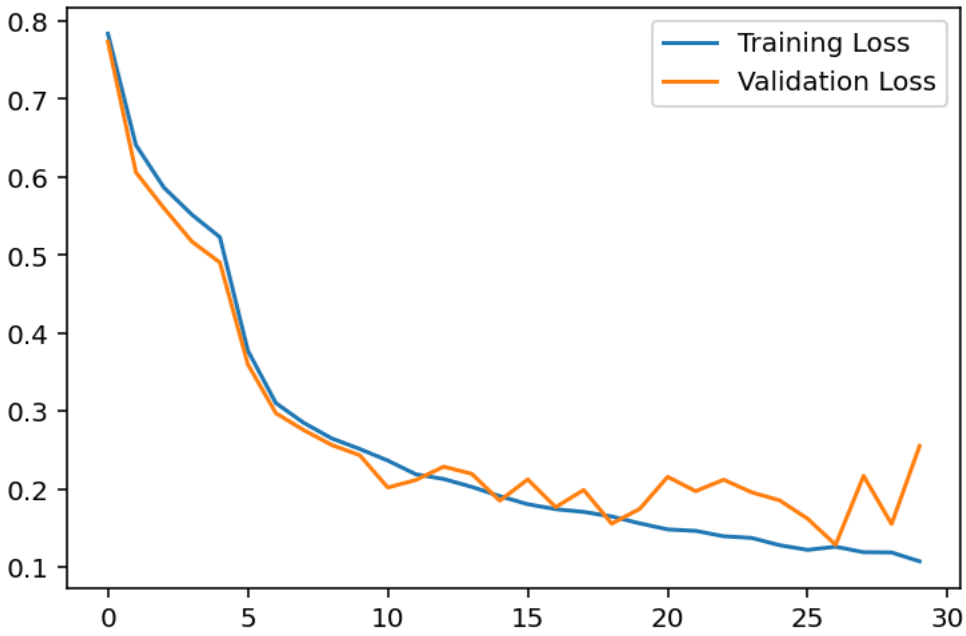

Figure 30: Experiment 5 Losses

Notably, from epoch 26 onwards, there is a large growth in validation loss. The accuracy trends show similar patterns as seen in figure 31.

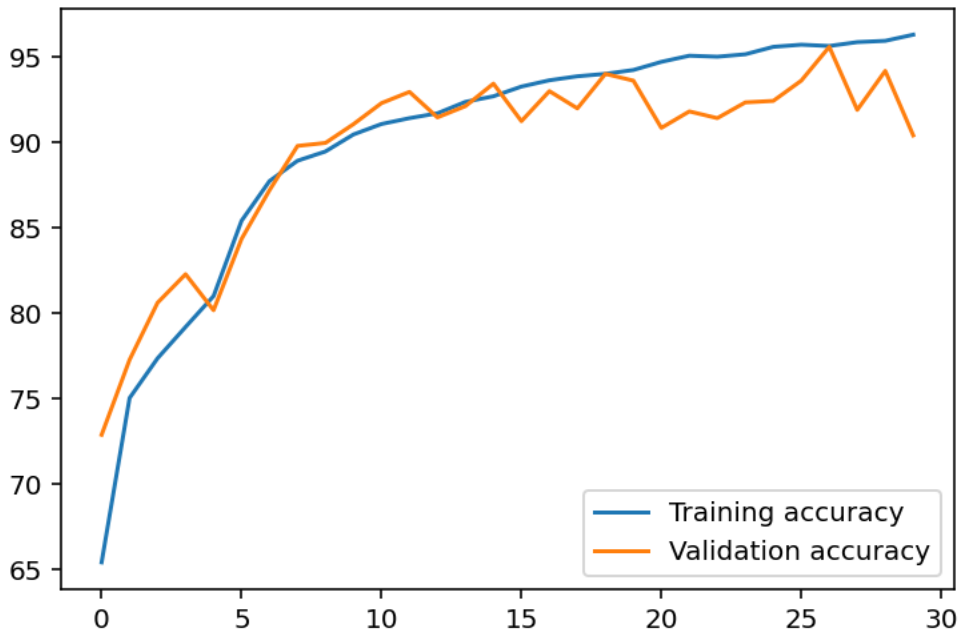

Figure 31: Experiment 5 Accuracies

5.6.2 – Evaluation

The confusion matrix of this model on the testing data is in figure 32.

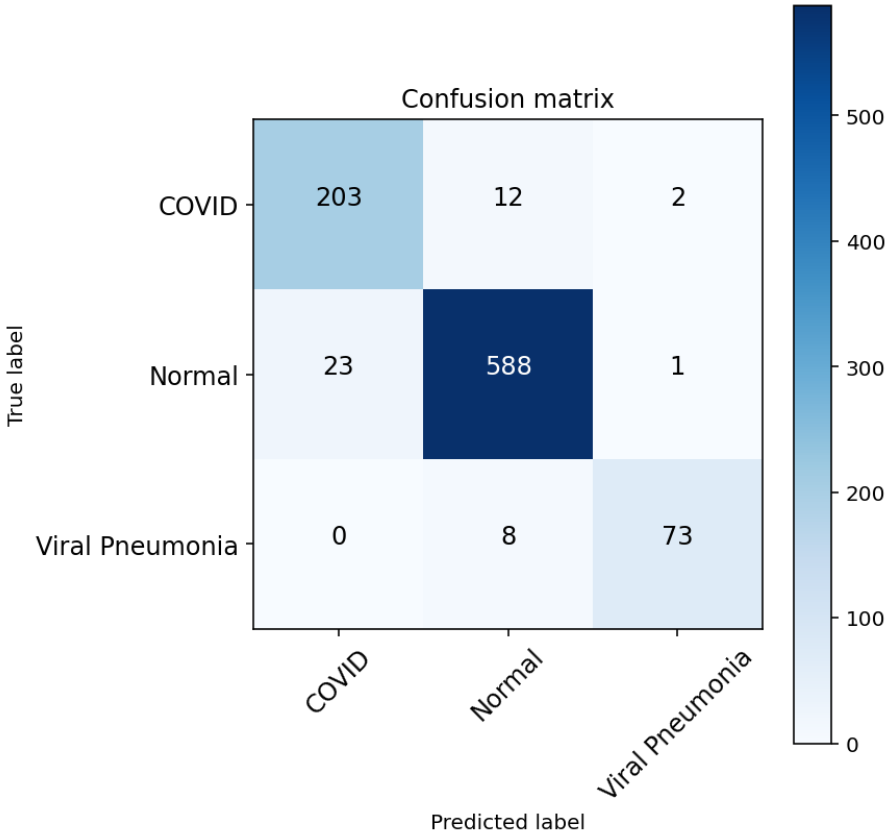

Figure 32: Experiment 5 Confusion Matrix

Here 94.95% of images have been classified correctly. Briefly, we can see that while the recall of the Covid class will be quite high, the precision has suffered

somewhat in comparison with the previous model. Further metrics can be found in table 23.

	Precision	Recall	F1-score	Support
COVID	89.82%	93.55%	91.65%	217
Normal	96.71%	96.08%	96.39%	612
Pneumonia	96.05%	90.12%	92.99%	81
Accuracy			94.95%	910
Macro avg	94.20%	93.25%	93.68%	910
Weighted avg	95.01%	94.95%	94.96%	910

Table 23: Experiment 5 Classification Report

This model presents recall of Covid-19 with 94%, only 14 COVID images misclassified. It also remains strong in other categories, maintaining precision of 0.90 in Covid and performing well on the other classes, although it does produce 8 false negatives on the Pneumonia class.

5.7 – Experiment 6

5.7.1 – Training

This experiment involves the fine tuning of the a pretrained ResNet18 [63] model as detailed in 4.3.6.

The losses of this training process can be seen in figure 33:

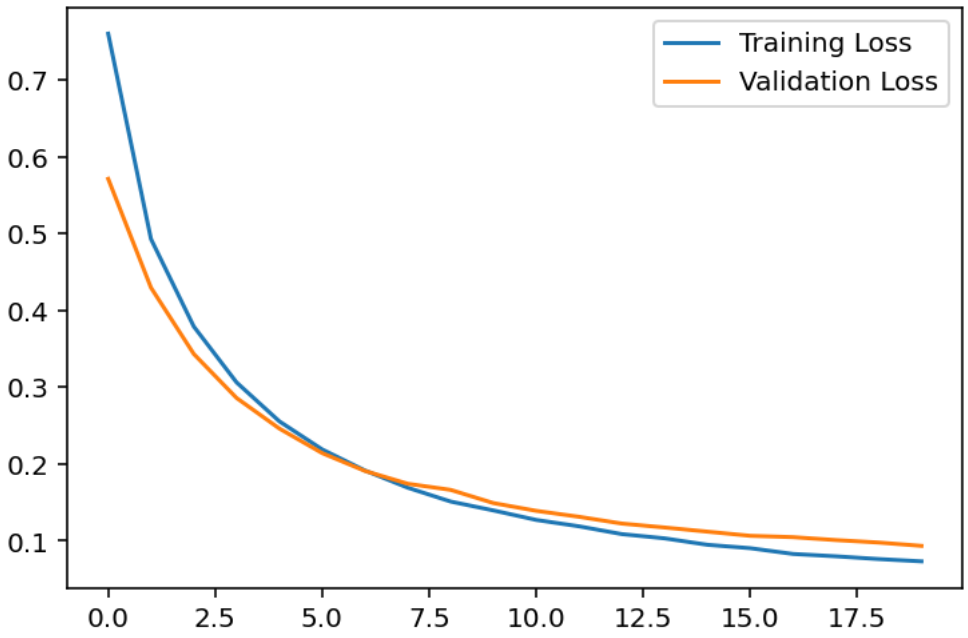

Figure 33: Experiment 6 Losses

A steady reduction in loss can be observed in both the both the training and validation calculations. The inverse trends can be seen in figure 34:

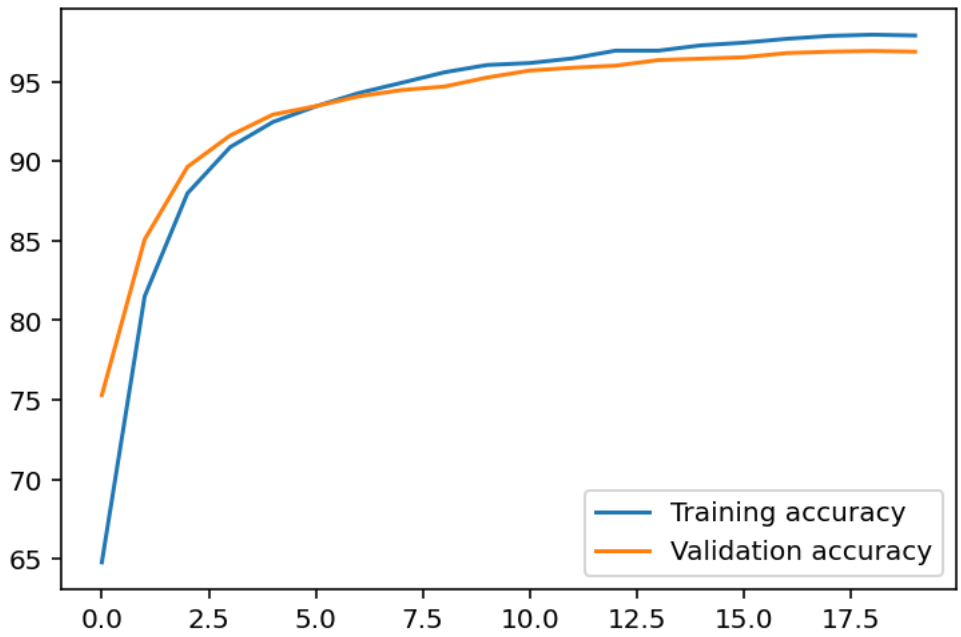

Figure 34: Experiment 6 Accuracies

Here we can see a steady increase in accuracy across the epochs. It can also be observed that the validation accuracy, although increasing, had flattened somewhat. Further epochs would surely lead to overfitting.

5.7.2 – Evaluation

Figure 35 presents the results of the evaluation on the training set in a confusion matrix:

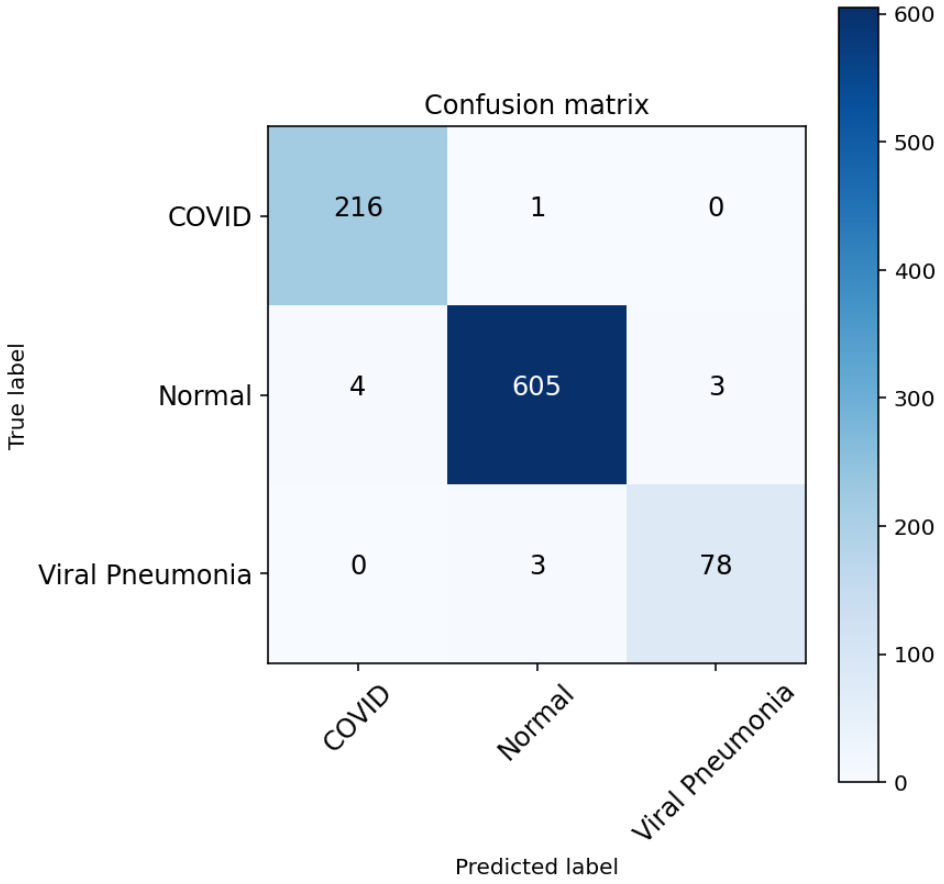

Figure 35: Experiment 6 Confusion Matrix

In is not difficult to see that this model has performed extremely well; only 11 images have been misclassified giving an overall accuracy of 98.79%. Further metrics can be observed in the table 24.

	Precision	**Recall**	**F1-score**	**Support**
COVID	98.18%	99.54%	98.86%	217
Normal	99.34%	98.86%	99.10%	612
Pneumonia	96.30%	96.30%	96.30%	81
Accuracy			98.79%	910
Macro avg	97.94%	98.23%	98.08%	910
Weighted avg	98.80%	98.79%	98.79%	910

Table 24: Experiment 6 Classification Report

5.8 – Metrics by Experiment

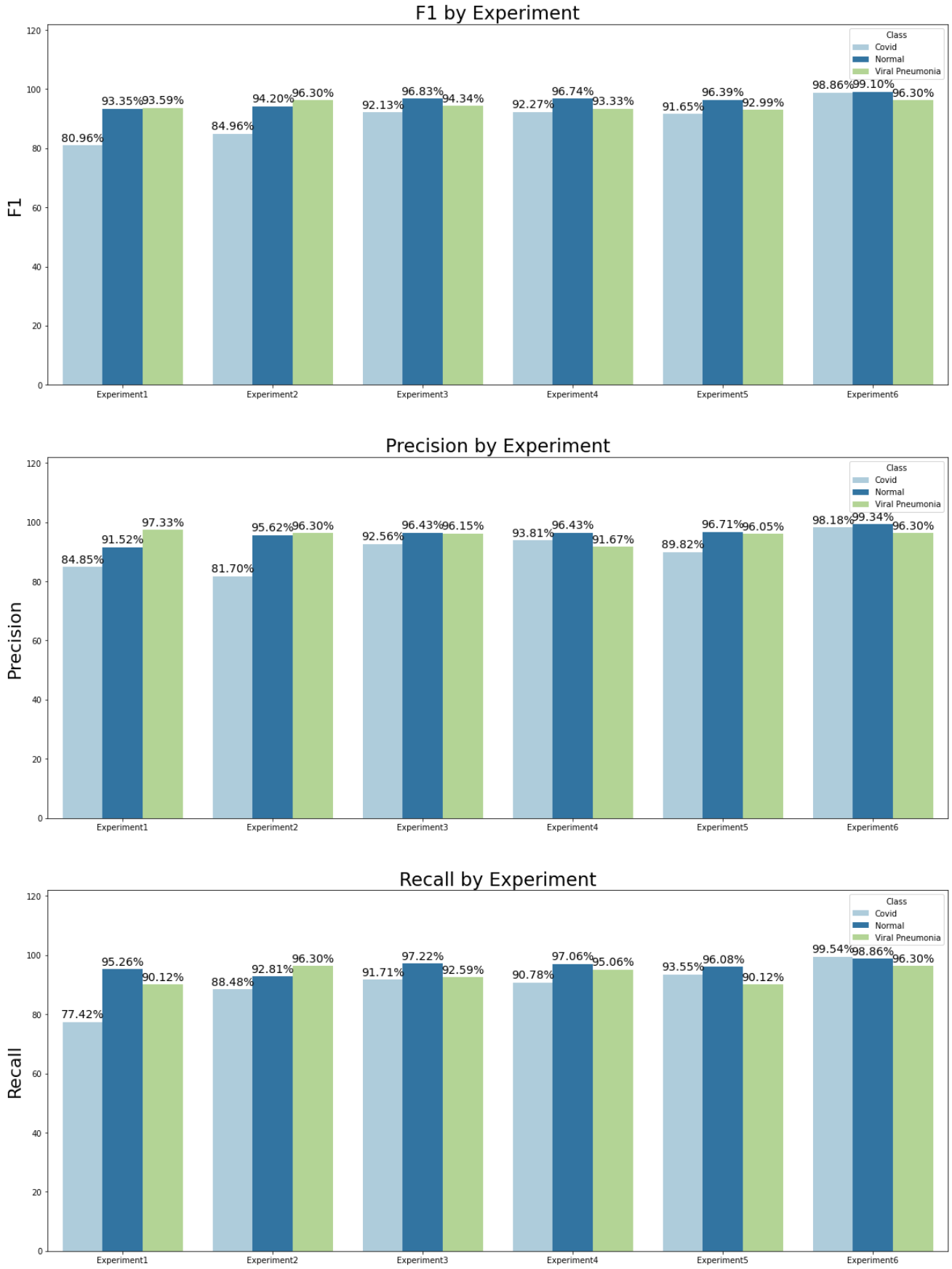

Figure 36: Metrics by Experiment

Figure 36 shows the metrics from the experiments side by side. It is easy to visualize the effects that the techniques employed in each experiment have had on the results. There is a score for each class for each experiment: light represents the Covid class, the darker shade of blue representing Normal, and green representing Viral Pneumonia.

The first diagram contains the F1 scores. The F1 scores are the harmonic means of the Precision and Recall. The F1 scores of the Normal Class are above 96% in 4 out of 6 experiments. The Viral Pneumonia class is also consistently above 92%. The Covid class' F1 scores improve across the experiments. A difference of almost 18 percentage points can be observed between experiment 1 and experiment 6.

The second diagram depicts the precision. As can be seen, precision on the Viral Pneumonia class is quite uniform across the models except for Experiment 4, where it falls below 96% for the first time. The precision on the Normal class for experiment 1 is the lowest, but improves in the other experiments, the maximum score being 99.34% in experiment 6.

The precision in the Covid class varies more. Experiment 1 and 2 are 84.85% and 81.70% each. The remaining experiments, however, do achieve scores in the 90th percentile, with experiments 3, 4 and 6 achieving 92.56%, 93.81% and 98.18% respectively.

Recall measures a model's ability to avoid false negatives. The Covid class in experiment 1 achieves the lowest score, with 77.42% of positive cases correctly identified. The other models perform well on Covid, with Experiment 6 achieving a 99.54%, a considerable improvement.

Each experiment performs well on the Normal class. The set being imbalanced in the Normal Class' favour probably helps this high performance. Each model avoids false negatives in the Viral Pneumonia class quite well with experiments 2 and 6 achieving scores of 96.30%.

6. – Discussion

Experiment 1 is considered the baseline model of this project. It involves no techniques other than the standard layers of a CNN. Experiment 1 achieves scores of 93% in both the Normal and Pneumonia classes, but only 80.69% in the Covid class. The precision scores for each class are 84, 91 and 97 percent for Covid, Normal and Viral Pneumonia respectively. In total, only 8 Viral Pneumonia images were misclassified.

The recall in the Covid-19 class is 77.42%. This means that only 77.42% of actual Covid-19 positives we identified correctly. As such, while the precision and recall of the Normal and Pneumonia classes are quite high, this model fails to identify a high number of the Covid-19 cases. These results can be taken as a benchmark to study the effects of the techniques used in the other experiments.

Experiment 2 uses the same architecture, augmenting the data. The expectation is that this technique will allow us to emulate a larger data set. This means that we should be able to train the model for more epochs, extracting more features that are useful while avoiding overfitting.

We can see that the F1 score has improved for each class in this experiment with respect to experiment 1. The precision on the Normal and Viral Pneumonia classes are 95.62% and 96.30% respectively. The precision on the Covid class drops slightly with respect to experiment 1 to 81.70%. The recall on the Covid class has increased by more than 10%, producing 10% less false negatives.

Kedia et al [11] and Minaee et al [12] also applied data augmentation to the data in their paper. Here is appears that data augmentation has allowed our model to better identify the features of each class, coinciding with a considerable improvement in the Covid class' F1 score.

Experiment 3 adds batch normalization layers which appear to have improved the performance of the model. In theory, batch normalization helps models converge more quickly and stabilises the results. We see here an improvement with respect to experiment 2.

In experiment 3, the F1 scores for Covid, Normal and Viral Pneumonia are 92.13, 96.83 and 94.34 percent respectively. This represents an improvement on the Covid and Normal classes with respect to experiment 2 and is the first experiment that achieves F1 above 90 for the Covid class. The Precision in the three classes have improved with respect to experiment 2, whereas the recall improved for the Covid and Normal classes. Batch normalization appears to have helped reduce the disparity between performance on Covid with respect to the other two classes.

Experiment 4 makes use of L2 regularization as described in section 4.3.4. The F1 scores of this experiment are similar those of experiment 3, with a slight increase in the Covid class, and small drops in the Normal and Viral Pneumonia classes. This experiment achieves the highest precision on the Covid class that we have seen so far with 93.81%. The Normal class stays the same as experiment 3 while the viral pneumonia precision falls to 91.67%. The viral pneumonia recall increases to 95.06% while the same metric on Covid is 90.78%.

The L2 regularization appears to not have had a very large effect on the results with numbers that a very similar to the previous experiment. It did achieve the best precision results on Covid and has not obviously adversely affected the results.

Experiment 5 makes use of drop out layers to prevent overfitting as detailed in section 4.3.5. Drop out is a simple technique that can have powerful results. Here we see that the F1 scores for Covid, Normal and Viral Pneumonia are 91.65, 96.39 and 92.99 percent respectively. These scores are comparable to those of experiment 4 and are achieved simply randomly zeroing out nodes during training. The precision in the Normal and Viral Pneumonia classes are 96.71 and 96.05 percent respectively.

Precision in Covid has fallen with respect to experiment 4 to 89.82%. However, this experiment achieves the highest recall on the Covid class so far at 93.55%. This means very few false negatives and, in the context of screening for covid, is a positive.

Experiment 6 makes use of transfer learning using a pretrained ResNet18 model. The expectation is that this pretrained model can be fine-tuned using our dataset and leverage the image-classifying capacity achieved through pretraining.

The F1 scores are 98.86%, 99.10% and 96.30% for Covid, Normal and Viral Pneumonia. These are the highest scores of any experiment. It is obvious that the pretrained model has been successfully fine-tuned. The recall on the Covid class is 99.54% meaning almost no false negatives were produced. The precision scores on the Normal and Covid classes are above 98%.

Of the techniques employed, transfer learning on a pretrained ResNet18 model achieves the best results. This result reflects findings in Minaee et al [12] where pretrained models were compared in effectiveness of identifying Covid-19 in x-rays. Given that Covid-19 x-ray image data is not massively available, transfer learning appears to present an interesting opportunity.

Screening patients is not equivalent to diagnosing patients with illnesses. The intention of this system is not to say, without doubt, that a patient has or does not have Covid-19. The intention is to classify, within reasonable certainty, of the people who come to a health centre, who should be tested and/or quarantined.

This use case will lay specific demands on the model chosen, it will not be enough, for example, to correctly identify many true positives and achieve high precision score while performing poorly on recall and false negatives. In the context of a pandemic, it is much more damaging to tell someone who is positive for Covid-19 that they are healthy and can go home to their family than it is to spend a PCR test on a healthy patient.

With this mind, a model which successfully meets the criteria for this use case will have a high recall score on Covid-19. This means minimising false negatives and avoiding inadvertent spread for the virus. We will also value high precision in the Normal and Viral Pneumonia classes. Correctly identifying healthy people is obviously favourable and will allow medical professionals to use PCR and antigens tests on people who need them. Viral Pneumonia, while not the cause of the current pandemic, is a serious disease and must be treated.

7. – Conclusions

The results have shown that there is a compelling argument to be made in favour of using radiographic images in the screen of patients for Covid-19. We have seen that both pretrained models and more basic models following good practices can achieve impressive results.

The high scores in Recall in the Covid and Viral Pneumonia classes to prevent inadvertent spread of the Coronavirus and failure to treat pneumonia along with the high precision scores in the Normal class suggest viability in practice.

As discussed, X-rays are cheap and massively available. They can be produced quickly and do not necessarily put machine operators at risk. These reasons suggest that radiographical imagery could play an important part in the management of the remainder of the pandemic. The high F1 scores of experiments 3 to 6 indicate that the models could indeed play a part in the control and prevention of Covid-19 in practice.

It is also necessary to recognise the limitations of this project. The dataset is 15,153 images in total 67% of which were images of healthy lungs. Due to the nature of the pandemic, medical institutions simply have not had time to produce and anonymise sufficient images. The approval to use these techniques in clinical settings should follow further investigation with bigger datasets to avoid problems like over-fitting. However, the results on this dataset are encouraging.

A continuation of this project would be to explore the features of the images which gave us our results. This project has been limited exclusively to quantitative results, meaning we have explored viability and results exclusively numerically. It would be of much interest to explore which specific features allowed the models to classify the images. These features could be validated by medical professionals, eliminating the possibility that the results have been influenced by characteristics of the machines taking the images or other noise.

Overall, the results of the investigation are compelling. Transfer learning and data augmentation both proven useful in the given use case and helped mitigate the issues that small datasets pose. Across the experiments, the metrics have shown that CNNs can differentiate healthy lungs from those affected by Covid-19 and Viral Pneumonia. As more images become available, more robust models can be developed and hopefully put to work for the medical professionals working to protect people during this pandemic.

8. – Glossary

Accuracy: The number of correct classifications divided by total number of classifications, usually expressed as percentage. 24

Activation Function: Defines the output of that node given an input or set of inputs. 14

Artificial Intelligence: Field of study concerning computer systems developing intelligence. 10

Artificial Neural Network: A multi-layer perceptron consisting of at least input, hidden and output layers. 13

Backpropagation: Process by which layers of ANN are adjusted using the loss function. 14

Batch Normalization: Scaling the output of each convolutional layer to avoid internal covariate shift. 20

Batch size: Number of samples shown to algorithm before adjustments to parameters. 16

Bias: Defines distance between function and origin. 13

Computed tomography: A computerized x-ray imaging procedure 2

Convolutional Neural Networks: Type of ANN specialised in images. 16

Coronavirus: Novel virus identified 7th January 2020 1

Covid-19: Associated disease of Coronavirus 1

Data Augmentation: Process of applying transformations to artificially increase size of dataset. 20

Deep Learning: An artificial neural network that uses multiple layers to extract high level features from data.

Dropout: Method to prevent overfitting by randomly dropping nodes during training 21

Epoch: Full set of training data shown to algorithm. 15

F1 score: The harmonic mean of the precision and recall. 25

False negative (FN): Sample incorrectly classified as not being of a class. 24

False positive (FP): Sample incorrectly classified as being of a class. 24

Flter/Kernel: In Convolutional layer, applies spatial operations to extract features from data. 17

Fully connected layer: Layer used to learn non-linear combinations of high-level features. 19

9. – Bibliography

[1] J. Feehan and V. Apostolopoulos, “Is COVID-19 the worst pandemic?,” *Maturitas*, pp. S0378-5122(21)00018–9, Feb. 2021, doi: 10.1016/j.maturitas.2021.02.001.

[2] ECDC, “COVID-19,” *https://www.ecdc.europa.eu/*, 2021. https://www.ecdc.europa.eu/ (accessed Mar. 04, 2021).

[3] A. S. Baig, H. A. Butt, O. Haroon, and S. A. R. Rizvi, “Deaths, panic, lockdowns and US equity markets: The case of COVID-19 pandemic,” *Finance Research Letters*, vol. 38, p. 101701, Jan. 2021, doi: 10.1016/j.frl.2020.101701.

[4] D. Jones, Lora; Palumbo, Daniele; Brown, “Coronavirus: How the pandemic has changed the world economy,” *BBC News*, 2021. .

[5] B. Gallo Marin *et al.*, “Predictors of COVID-19 severity: A literature review,” 2020, doi: 10.1002/rmv.2146.

[6] S. Wacharapluesadee *et al.*, “Evaluating the efficiency of specimen pooling for PCR-based detection of COVID-19 This paves the way for large-scale population screening, allowing for assured policy,” 2020, doi: 10.1002/jmv.26005.

[7] M. Pavelka *et al.*, “The impact of population-wide rapid antigen testing on SARS-CoV-2 prevalence in Slovakia,” *Science*, p. eabf9648, 2021, doi: 10.1126/science.abf9648.

[8] A. M. Neilan *et al.*, “Clinical Impact, Costs, and Cost-effectiveness of Expanded Severe Acute Respiratory Syndrome Coronavirus 2 Testing in Massachusetts,” *Clinical Infectious Diseases*, 2020, doi: 10.1093/cid/ciaa1418.

[9] M. E. H. Chowdhury *et al.*, “Can AI Help in Screening Viral and COVID-19 Pneumonia?,” *IEEE Access*, vol. 8, pp. 132665–132676, 2020, doi: 10.1109/ACCESS.2020.3010287.

[10] J. P. Kanne, B. P. Little, J. H. Chung, B. M. Elicker, and L. H. Ketai, “Essentials for Radiologists on COVID-19: An Update—Radiology Scientific Expert Panel,” *Radiology*, vol. 296, no. 2, pp. E113–E114, 2020, doi: 10.1148/radiol.2020200527.

[11] P. Kedia, Anjum, and R. Katarya, “CoVNet-19: A Deep Learning model for the detection and analysis of COVID-19 patients,” *Applied Soft Computing*, vol. 104, p. 107184, Jun. 2021, doi: 10.1016/j.asoc.2021.107184.

[12] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. Jamalipour Soufi, “Deep-COVID: Predicting COVID-19 from chest X-ray images using deep transfer learning,” *Medical Image Analysis*, vol. 65, p. 101794, Oct. 2020, doi: 10.1016/j.media.2020.101794.

[13] Google, “Google Colab Pro,” 2021. https://colab.research.google.com/signup#.

[14] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[15] “WHO | Pneumonia of unknown cause.” 2020, Accessed: Apr. 28, 2021. [Online]. Available: https://www.who.int/csr/don/05-january-2020-pneumonia-of-unkown-cause-china/en/.

[16] A. S. Fauci, H. Clifford Lane, and R. R. Redfield, “Covid-19-Navigating the Uncharted,” 2020, doi: 10.1056/NEJMoa2002032.

[17] Q. Li *et al.*, “Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus–Infected Pneumonia,” *New England Journal of Medicine*, vol. 382, no. 13, pp. 1199–1207, 2020, doi: 10.1056/nejmoa2001316.

[18] T. A. Harahwa, T. H. Lai Yau, M.-S. Lim-Cooke, S. Al-Haddi, M. Zeinah, and A. Harky, “The optimal diagnostic methods for COVID-19,” *Diagnosis*, vol. 7, no. 4, pp. 349–356, 2020, doi: doi:10.1515/dx-2020-0058.

[19] M. O. Wielpütz, C. P. Heußel, F. J. F. Herth, and H. U. Kauczor, “Radiologische diagnostik von lungenerkrankungen: Beachtung der therapieoptionen bei wahl des verfahrens,” *Deutsches Arzteblatt International*, vol. 111, no. 11. pp. 181–187, 2014, doi: 10.3238/arztebl.2014.0181.

[20] F. A. J. Mettler, W. Huda, T. T. Yoshizumi, and M. Mahesh, “Effective doses in radiology and diagnostic nuclear medicine: a catalog.,” *Radiology*, vol. 248, no. 1, pp. 254–263, Jul. 2008, doi: 10.1148/radiol.2481071451.

[21] Statistisches Bundesamt, “Fallpauschalenbezogene Krankenhausstatistik (DRG-Statistik) Operationen und Prozeduren der vollstationären Patientinnen und Patienten in Krankenhäusern - Ausführliche Darstellung -,” 2013. [Online]. Available: https://www.statistischebibliothek.de/mir/servlets/MCRFileNodeServlet/DEHeft_derivate_00012373/5231401127014.pdf.

[22] Bundesamt für Strahlenschutz, “Strahlenexposition durch Medizinische Maßnahmen,” 2013. [Online]. Available: https://www.bfs.de/SharedDocs/Downloads/BfS/DE/fachinfo/ion/medizin.pdf?__blob=publicationFile&v=1.

[23] K.-C. Chen *et al.*, “Diagnosis of common pulmonary diseases in children by X-ray images and deep learning,” 2020, doi: 10.1038/s41598-020-73831-5.

[24] N. Caballé, J. Castillo-Sequera, J. A. Gomez-Pulido, J. Gómez, and M. Polo-Luque, “Machine Learning Applied to Diagnosis of Human Diseases: A Systematic Review,” *Applied Sciences*, vol. 10, p. 5135, Jul. 2020, doi: 10.3390/app10155135.

[25] I. Kavakiotis, O. Tsave, A. Salifoglou, N. Maglaveras, I. Vlahavas, and I. Chouvarda, “Machine Learning and Data Mining Methods in Diabetes

Research," *Computational and Structural Biotechnology Journal*, vol. 15. Elsevier B.V., pp. 104–116, Jan. 01, 2017, doi: 10.1016/j.csbj.2016.12.005.

[26] J. A. Cruz and D. S. Wishart, "Applications of machine learning in cancer prediction and prognosis.," *Cancer informatics*, vol. 2, pp. 59–77, Feb. 2007.

[27] M. A. Myszczynska *et al.*, "Applications of machine learning to diagnosis and treatment of neurodegenerative diseases," *Nature Reviews Neurology*, vol. 16, no. 8, pp. 440–456, 2020, doi: 10.1038/s41582-020-0377-8.

[28] J. Ker, L. Wang, J. Rao, and T. Lim, "Deep Learning Applications in Medical Image Analysis," *IEEE Access*, vol. 6, pp. 9375–9389, 2018, doi: 10.1109/ACCESS.2017.2788044.

[29] G. Litjens *et al.*, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42. Elsevier B.V., pp. 60–88, Dec. 01, 2017, doi: 10.1016/j.media.2017.07.005.

[30] N. M. Elshennawy and D. M. Ibrahim, "Deep-Pneumonia Framework Using Deep Learning Models Based on Chest X-Ray Images," *Diagnostics*, vol. 10, no. 9, 2020, doi: 10.3390/diagnostics10090649.

[31] T. Rahman *et al.*, "Transfer learning with deep Convolutional Neural Network (CNN) for pneumonia detection using chest X-ray," *Applied Sciences (Switzerland)*, vol. 10, no. 9, 2020, doi: 10.3390/app10093233.

[32] S. Salehi, A. Abedi, S. Balakrishnan, and A. Gholamrezanezhad, "Coronavirus Disease 2019 (COVID-19): A Systematic Review of Imaging Findings in 919 Patients," *American Journal of Roentgenology*, vol. 215, pp. 1–7, Mar. 2020, doi: 10.2214/AJR.20.23034.

[33] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955," *AI Magazine*, vol. 27, no. 4 SE-Articles, p. 12, Dec. 2006, doi: 10.1609/aimag.v27i4.1904.

[34] A. Kumar, "Machine Learning: Validation Techniques," 2018. .

[35] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980, doi: 10.1007/BF00344251.

[36] A. Vo, "Deep Learning – Computer Vision and Convolutional Neural Networks," 2018. .

[37] Stanford, "Convolutional Neural Networks (CNNs / ConvNets)," 2021. https://cs231n.github.io/convolutional-networks/.

[38] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *ArXiv e-prints*, Mar. 2016.

[39] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[40] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *CoRR*, vol. abs/1502.0, 2015, [Online]. Available: http://arxiv.org/abs/1502.03167.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014, [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[42] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, 2010.

[43] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994, doi: 10.1109/72.279181.

[44] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[45] R. Nandepu, “Understanding and implementation of Residual Networks(ResNets),” *Medium.com*. .

[46] programmersought.com, “ResNet-18 implements Cifar-10 image classification Pytorch.” https://www.programmersought.com/article/68543552068/.

[47] K. K. Bressem, L. C. Adams, C. Erxleben, B. Hamm, S. M. Niehues, and J. L. Vahldiek, “Comparing different deep learning architectures for classification of chest radiographs,” vol. 10, p. 13590, 2020, doi: 10.1038/s41598-020-70479-z.

[48] T. Rahman *et al.*, “COVID-19 Chest Radiography Database,” 2020. https://www.kaggle.com/tawsifurrahman/covid19-radiography-database.

[49] Kaggle, “Kaggle.” .

[50] Società Italiana di Radiologia Medica e Interventistica, “COVID-19 DATABASE,” 2021. https://sirm.org/.

[51] B. C. Winther, Hinrich B. and Laser, Hans and Gerbel, Svetlana and Maschke, Sabine K. and B. Hinrichs, Jan and Vogel-Claussen, Jens and Wacker, Frank K. and Höper, Marius M. and Meyer, “COVID-19 Image Repository,” *figshare*, 2020. https://figshare.com/articles/dataset/COVID-19_Image_Repository/12275009/1.

[52] Eurorad, “Eurorad.” https://www.eurorad.org/.

[53] A. Haghanifar, M. M. Majdabadi, and S. Ko, “COVID-CXNet: Detecting covid-19 in frontal chest x-ray images using deep learning,” *arXiv*. 2020, [Online]. Available: https://sirm.org/category/senza-categoria/covid-19/.

[54] J. P. Cohen, P. Morrison, and L. Dao, “COVID-19 image data collection,” *arXiv 2003.11597*, 2020, [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset.

[55] BIMCV, “BIMCV-COVID-19,” 2020. https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/.

[56] RSNA, “RSNA Pneumonia Detection Challenge,” *Kaggle*, 2018. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data.

[57] P. Mooney, “Chest X-Ray Images (Pneumonia),” *Kaggle*, 2018. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[58] M. Kermany, Daniel; Zhang, Kang; Goldbaum, “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification,” *Mendely Data*, 2018.

[59] T. Rahman *et al.*, “Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images,” *Computers in Biology and Medicine*, vol. 132, p. 104319, 2021, doi: https://doi.org/10.1016/j.compbiomed.2021.104319.

[60] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011, [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html.

[61] Pytorch, “TORCH.OPTIM.” https://pytorch.org/docs/stable/optim.html (accessed May 01, 2021).

[62] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[63] PyTorch, “ResNet PyTorch Vision.” .