Ollscoil
Teicneolaíochta
an Atlantaigh

Atlantic
Technological
University

Medical Image Classifier (Med-AI)
Project Engineering
Year 4

by

Aleksander Kijewski
G00396018

Bachelor of Engineering (Honours) in Software and Electronic Engineering
Atlantic Technological University
2023/2024

Supervised by:
Ben Kinsella

School of Engineering
Atlantic Technological University

April 28, 2024

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Atlantic Technological University. This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Acknowledgements

I would like to express my sincere gratitude to Paul Lennon, Ben Kinsella, and Michelle Lynch for their invaluable guidance and support throughout this project. Their expertise and insights have been instrumental in shaping the direction and outcomes of this work.

I would also like to express my gratitude of all the tools and technologies that supported the completion of this project, this includes the use of Artificial Intelligence which provided assistance and insights that enhanced the quality of my work.

Contents

1 Definitions

1. **CNN:** A type of deep learning neural network designed to process data in a grid pattern such as videos or images. CNNs use convolution layers to learn patterns and features from input data[1]

2. **Neural Network:** A model which is inspired by the biological human brain's neural network. Neural networks are models which can be trained and organized in layers and are capable of learning complex patterns from data.[2]

3. **Convolution Filters:** These are inside convolution layers to extract features from the input data. Filters in simple terms "slide" across the input data and perform feature extraction to generate feature maps.[3]

4. **Filter Maps:** These represent the specific patterns and features extracted by filters that the neural network uses to then predict new data.[4]

5. **Depthwise Convolution:** Depthwise convolution is applied to carry spatial feature operations independently and separately within the channels, hence enabling computational efficiency by treating spatial information in each channel separately and appropriately for large or high-dimensional data.[5]

6. **Pointwise Convolution:** Pointwise convolution is generally applied after depthwise convolution to combine the channel-wise outputs of depthwise using 1x1 convolutional filters. Summing in this line is where the features in the channels are integrated, in a sense of condensing and enriching the information content without loss of professional rigor.[6]

7. **Sigmoid Activation:** This is a mathematical function that maps any input value to an output value between 0 and 1. It is commonly used in binary classification models because its output can be interpreted as a probability.[7]

   Sigmoid Formula

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

8. **Softmax Activation:** The softmax activation provides an extension to the sigmoid function for multi-class classification problems. It converts the raw output scores from a model into probabilities. This then distributes the probabilities between the classes.[8]

   Softmax Formula

   $$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2}$$

2 Summary

The project's primary objective is to elevate the level of diagnostic accuracy and efficiency in the field of medical science by applying artificial intelligence (AI), particularly in the analysis of medical images, for instance, X-rays, Magnetic Resonance Images (MRIs), and Computer Tomography (CT) scans. It gives automated and reliable assistance to professional and amateur medical practitioners in making timely medical diagnoses.

Project scope: Develop and train convolutional neural networks for the classification of different kinds of medical images into more than one category, such as normal findings or lesions, like damaged tissue, injury, or disease. Initially, the cats-vs-dogs binary classification model was trained using the techniques and datasets learned in the Xception network model from Keras. This has been laid to understand CNN and image classification further and to make this medical imaging case study possible with a more advanced but complex multi-class model that is built to achieve better accuracy.

The aspects of the project were model development in TensorFlow and Keras, multi-class classification, and implementation with techniques like early stopping to prevent over-fitting. This method combined theoretical studies with practical applications, including activating with layer tuning and softmax activation functions for multi-class classification.

The work has eventually condensed to the development of separate models for MRI, CT, and X-ray scans, each capable of singling out specific pathological conditions. This has been practically implemented on a convenient website where a user can upload any kind of medical image and be offered classification at a glance—which practically means that the AI model developed earlier does work effectively in practice.

The project utilizes Flask and Tensorflow to provide an advanced solution with CT scans, MRI images, and X-rays for medical image classification. The system at its base level uses intelligent discernment to make out the kind of medical image submitted and then calls on a sophisticated deep learning model to arrive to the correct diagnosis, differentiation from a host of conditions like COVID-19, pneumonia, and different types of tumors.

The main conclusion of this work is that AI and specifically deep learning models, such as CNN offer powerful augmentation to the analysis of medical imaging and hence present a valuable tool that offers aid in processing of medical diagnosis.

3 Poster

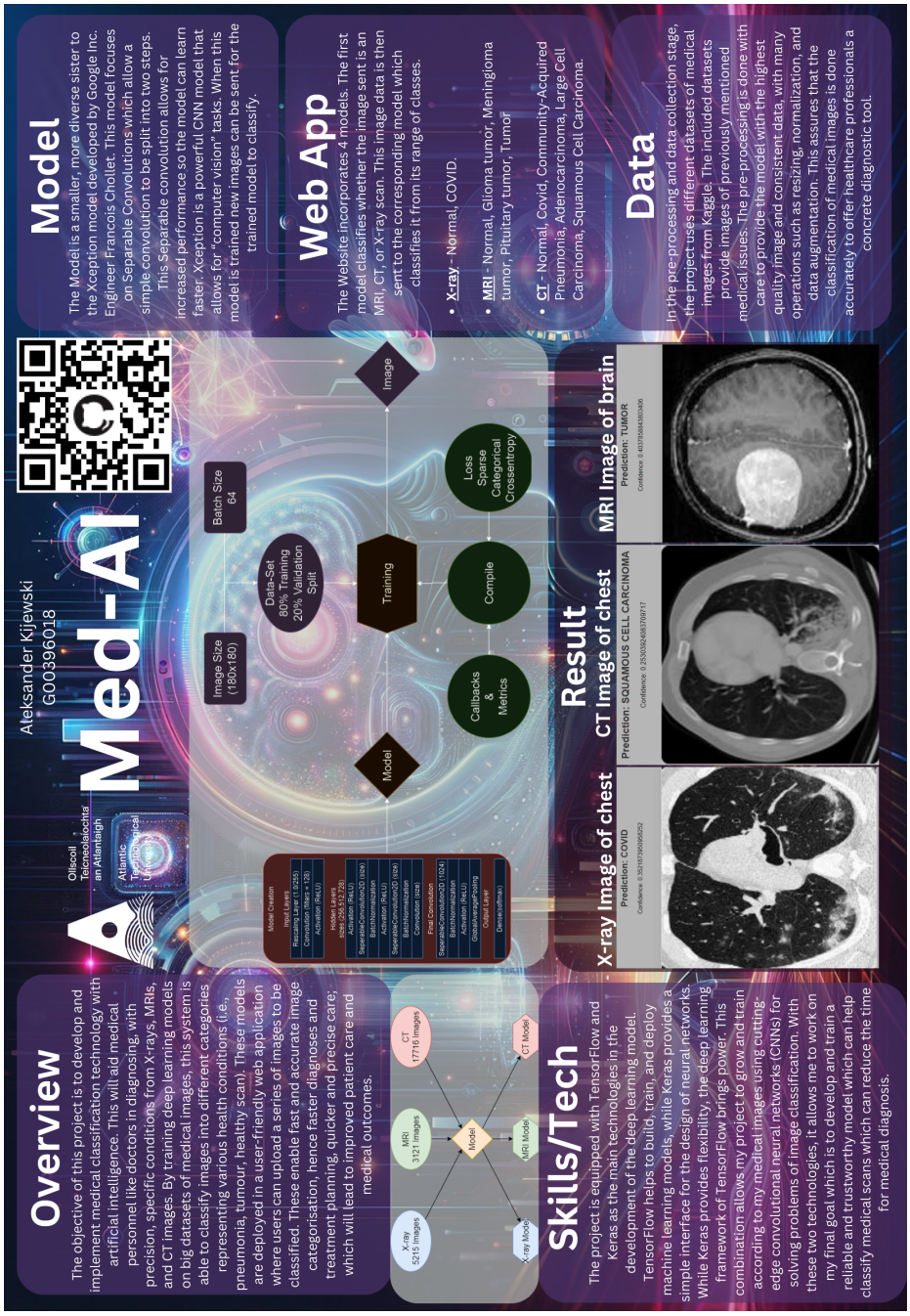

Figure 1: Medical Image Classifier Poster

4 Introduction

Artificial intelligence (AI) has emerged as a great game-changer that holds tremendous potential to enhance outcomes in healthcare, especially in the medical image analysis space. In this AI through deep learning and convolutional neural networks (CNNs) increases accuracy and reduced time during the diagnosis of medical conditions from images such as X-rays, MRIs, and CT scans. Reasons and motivation for this work are evidenced in the continued need for quicker and more reliable diagnostic procedures within the medical setting which can go a long way toward determining effective care and treatment protocols.

This project includes everything from developing AI models capable of classifying different categories indicative of several diseases in medical imagery, as well as the training of such models to further their deployment. This demands a complete approach, from the most elementary understanding of how image classification works using simpler models, to achieving applications with more complex multi-classification systems tuned for specific kinds of medical images. The project shows not only the possibility of AI use in medicine but also the development of a working instrument that can help medical professionals and researchers in their daily practice.

This report focuses on referencing solid AI methodologies with recognized technologies and frameworks like TensorFlow and Keras. The models would be trained using datasets sourced from publicly available standardized collections of images. These sets would then be utilized for the initial testing of the model.

5 Background

5.1 Motivation

The inspiration for this project began as a helping hand for medical professionals to increase accuracy in diagnostic measures. This became a challenge to understand the vast amount of data required to diagnose medical images. This was an opportunity to use technology to find deeper insights into patient health, leading to better outcomes for those in desperate need.

This then brought the attention of CNN which can make this task possible. CNN can provide useful feedback and results of what an image is showing which can allow medical professionals to continue doing what is more important like medical procedures. This technology would allow for second opinions and detection in the early stages of a disease. However, developing such an AI system that is capable of understanding complex medical images requires an understanding of the basic abilities and limitations of AI.

5.2 Introduction to AI in Medical Image Analysis

The application of AI in the medical field has cultivated some remarkable technological innovations, particularly in the analysis of medical images such as X-rays, MRI and CT scans. This represents a significant breakthrough. Utilizing AI models, notably Convolutional Neural Networks (CNNs) to extract useful information from complicated medical images has established a platform for accurate patient diagnosis across various health conditions. This offers invaluable assistance to doctors and even amateurs to quickly but efficiently and precisely diagnose patients.

5.3 Initial Exploration: Cats vs Dogs Classification

The adventure to AI-powered medical image recognition first started with something essential like creating an AI model, which is using the Cats Vs. Dogs dataset available on Keras' website [9]. At this initial stage, the aim was to develop an AI model which could classify Cats and Dogs to be able to determine if an image is showing either a cat or a dog.

5.4 Technologies

The main technologies that drove this project to success are TensorFlow [10], and Keras [11]. Which are well-known frameworks for developing a deep learning model in particular CNN's. Other technologies such as numpy [12], PIL [13], and flask [14] were also used within my project which will be explained further.

5.5 Dataset

The Datasets for this study were obtained from various locations, many of which are found on Kaggle:

- COVID-19 CT Slice Dataset [15]
- Chest CT Scan Images Dataset [16]
- Brain MRI images for Brain tumour detection Dataset [17]
- Brain Tumor classification MRI Dataset [18]
- Chest X-Ray Dataset [19]

5.6 Dataset Images

Below are the images and classes collected to utilise in model development.

Abbreviations for conditions shown below:

1. **CAP** Community-Acquired Pneumonia
2. **LCC** Large Cell Carcinoma
3. **SCC** Squamous Cell Carcinoma

Figure 2: X-Ray Images

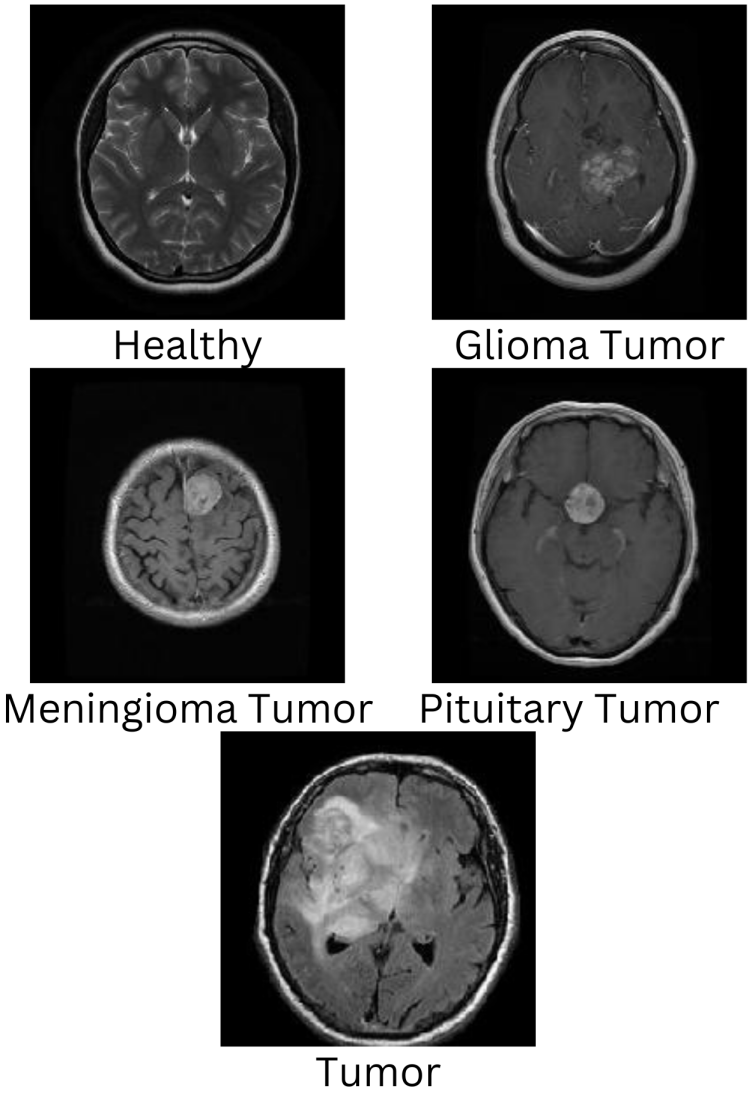

Figure 3: MRI Images

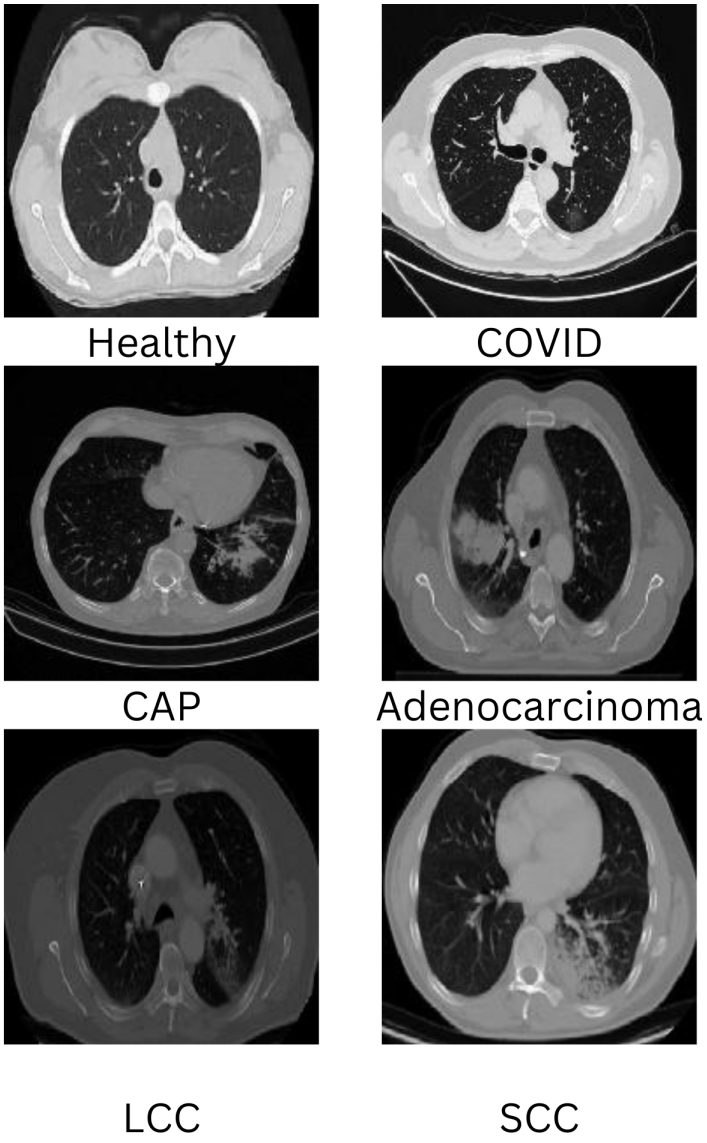

Figure 4: CT Images

6 Architecture

6.1 Model

Figure 5 depicts the model's architecture, which is an optimised version of the Xception model. There are three sections the input layers, hidden layers, and an output layer. See Section 9 for more details.

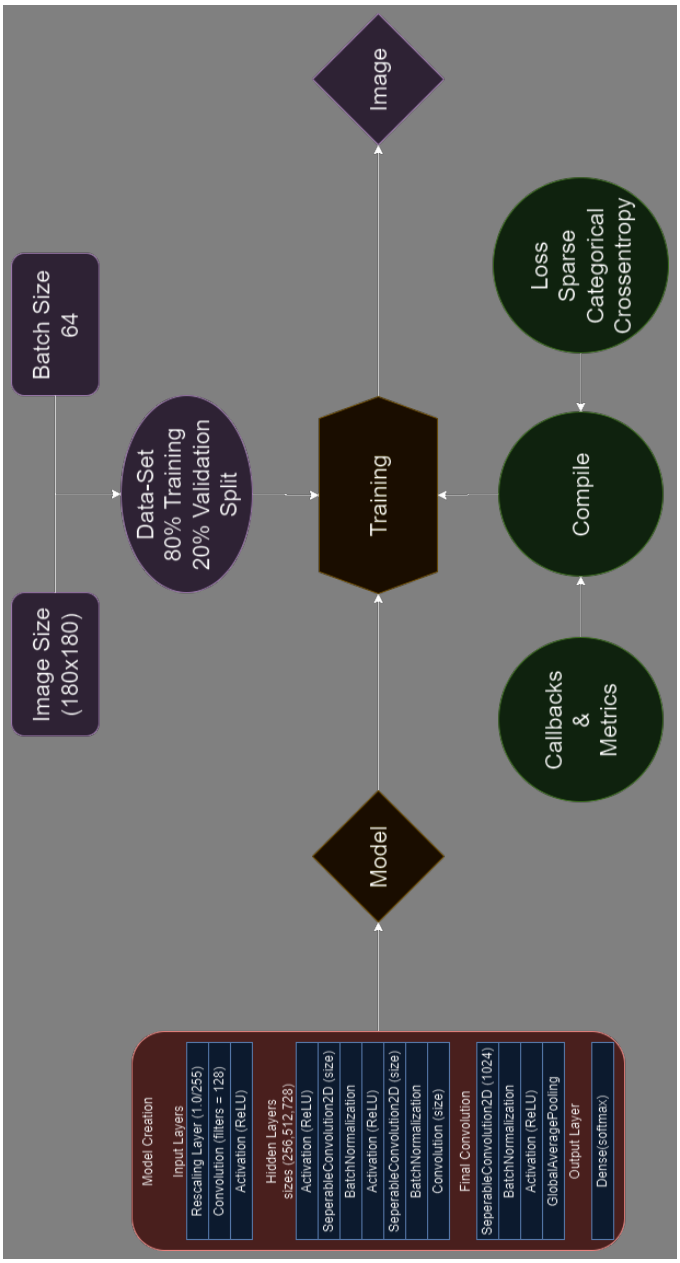

Figure 5: Architecture Diagram of downscaled Xception Model

6.2 Website

Figure 6 shows the workflow within the web application. This is the process from beginning where the user uploads an image, to where they get their response. See Section 13 for more details.

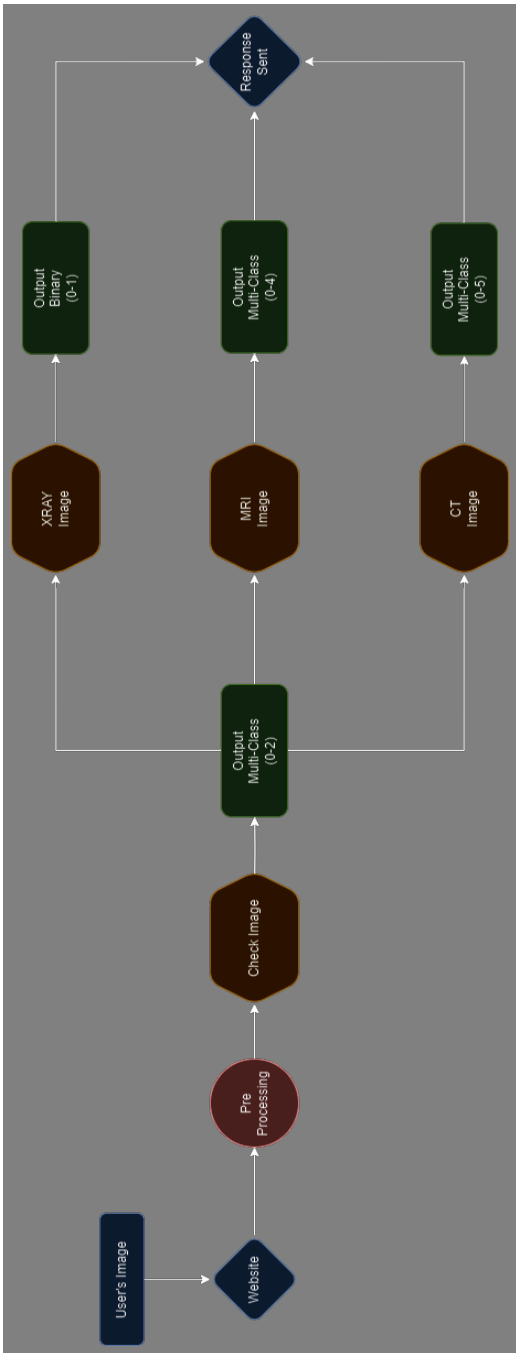

Figure 6: Architecture Diagram of the website

7 Project Plan

7.1 Initial Project Planning

Initial project plan was constructed as seen in Figure 7. While the timing aspect of this plan was largely accurate, several tasks enlisted did not unfold as anticipated.

Figure 7: Initial timeline

October – December: The main focus during Phases 1-3 was to set up a working proof of concept which as mentioned was classifying Cats vs Dogs. This proved to be a success with a working demo that was later uploaded to YouTube[20]. Engaging with this concept proved to be a great learning experience to get this project working with medical images effectively. These phases proved invaluable as they involved extensive research into CNN architectures, image pre-processing techniques, and AI frameworks (TensorFlow, Keras). This process facilitated gathering relevant material and resources related to medical image analysis and AI applications in healthcare.

January: In phase 4 of the project, data collection has commenced by scavenging Kaggle and eventually concluded with using a Google Scraper [21]. This resulted in acquisition of over 25000 images used for classification purposes.

February: In Phase 5 the focus has shifted to capturing a functioning model. This proved to be a challenging task in getting a satisfactory working model.

March: Phase 6 of the project was deemed successful with the implementation of working Keras models used for classifying the inputted images. Additionally, a website was developed to provide users with the capability to upload their images with the back-end built using Flask to classify and provide the user with a prediction.

April: During the final phase of the project all deliverables were generated. This included the report write-up creation of a poster, presentation and a video.

8 Convolutional Neural Network

CNNs are a class of deep neural networks which are commonly used in image recognition and classification. These networks are designed to automatically learn different features from raw data. This involves a substantial amount of adapting and supporting recognition of new patterns directly from pixel data, which makes it a well-suited candidate for image classification, object detection, and image segmentation.

Key Characteristics:

1. **Rescaling Layers:** In the rescaling layer the input image gets scaled to the size accepted by the model.

2. **Convolutional Layers:** These layers are operations that apply to the input data using a learnable filter or kernel. This operation then enables the Neural Network to extract and learn patterns and features.[3]

3. **Pooling Layers:** Pooling layers reduce the dimension and complexity of the input, this helps summarise features extracted by the convolutional layers. It distinguishes and keeps critical key features through operations like max pooling, which means taking the maximum values within a region, and average pooling, which computes the average value. This is a process that is targeted to make the network to focus on relevant patterns that would make it more efficient against changes in the input data.[3]

4. **Activation Functions:** An activation function such as ReLU(Rectified Linear Unit) is typically used for layers like Convolutional and Pooling. This activation function is commonly used after these layers to introduce non-linearity to the network. This allows it to learn complex relations to the data.[3]

5. **Batch Normalization:** This is a technique used to improve training stability and speed of the neural network. It normalizes the activation from previous layer by reducing it by the mean and dividing it by the standard deviation. This allows for higher learning rates.

CNNs have revolutionized the field of computer vision. This has achieved remarkable performance levels in tasks involving detection or analysis of different features such as tumours, cats and dogs, etc..

9 Model

The model used for this training is called Xception which was first introduced by François Chollet, a Google Inc. Engineer. It is based on the research paper "Xception: Deep Learning with Depthwise Separable Convolutions"[22]. The name "Xception" refers to "Extreme Inception" which is a type of build on Inception Architecture. This model offers a lot focusing on efficiency and performance with the following features:

1. **Depthwise Separable Convolutions:** Instead of a standard convolution, which applies filters across all input channels simultaneously, Xception offers separable convolutions which split the process into two steps: Depthwise convolution and Pointwise convolution. This reduces the amount of computation needed while maintaining accuracy or even improving it.[23]

2. **Architecture:** The original Xception model consists of 36 convolution layers whereas the scaled-down version used in this project consists of only 10 convolutional layers, 8 of which are Depthwise Separable Convolution. This decreases the accuracy factor of my model but it allows for quicker model learning.[22]

3. **Feature Learning:** Xception is created to allow the model to learn features from images, making it a good pick for my project as it offers image classification, object detection and classification of images especially in the detection of tumours and other features.[24]

4. **Performance:** Xception is one of the best models for image classification used to this day by Google with its benchmarks like the ImageNet dataset, surpassing other architectures in terms of accuracy and efficiency. Since the project is concerned with medical analysis, reliability is of the utmost importance, making Xception one of the best choices in this case.[22]

5. **Applications:** Xception has been used in a vast variety of real-world applications including image detection and shape detection. Many notable applications include Facial recognition, and object detection in autonomous vehicles.

Xception is a powerful CNN architecture that combines depthwise convolutions with a deep neural network. This makes this model efficient and effective for various "computer vision" tasks. This is why it is perfect to use this for this medical image classification project.

10 Proof of concept

10.1 Cats vs Dogs

Initially, the "Cats vs Dogs" dataset on Kaggle was explored as it is one of the most popular datasets to use in carrying out image classification tasks. Following the lead from Keras, building the model of a neural network has commenced to classify the images. This particular model determined whether an image contains a cat or a dog. This initial step provided insight into fundamental concepts such as convolutional layers for feature extraction, pooling layers for spatial down-sampling, and fully connected layers for classification.

Data pre-processing played a crucial role in preparing the dataset for training the neural network model. The processes here involved the normalization of pixel values from zero to a suitable range for training and resizing images to a standard size, then finally splitting the dataset into training and validation datasets.

These steps guaranteed that the model learned meaningful patterns from the images and generalized well to unfamiliar data. The pre-processed images were sequentially inputted into the neural network model and iteratively updated the model's weights at each step so that it could minimize the chosen loss function using an optimizer like Adam.

The hyper parameter tuning was further conducted on parameters such as: changes in the learning rate, batch size, and the number of training epochs to optimize model performance and merging during the training process.

The performance of the model was computed based on the accuracy, precision, and recall evaluation metrics from the training set and validation set. To gain insight into the learning process of the model and possible problems such as over-fitting[25] or under-fitting[26] concerning epochs, accuracy, and loss, the respective curves have been plotted in the training/validation set. From the result and analysis carried out in the proof of concept exercise, it brought a lot of insight on how the architecture of the neural network, the training process, and evaluation techniques for an image classification task works. This can be further extended in the future with techniques for advanced model architectures (e.g., transfer learning using pre-trained models) and data augmentation strategies for model generalization. For the main project in AI for Medical Science, transition into the bigger datasets that come up in the field of medical imaging.

10.2 Evolution of Cats vs Dogs

These are images of confusion matrices that provide important data on how the model is performing

Index:

Cat - Cat or Dog - Dog = Correct

Cat - Dog or Dog - Cat = Incorrect

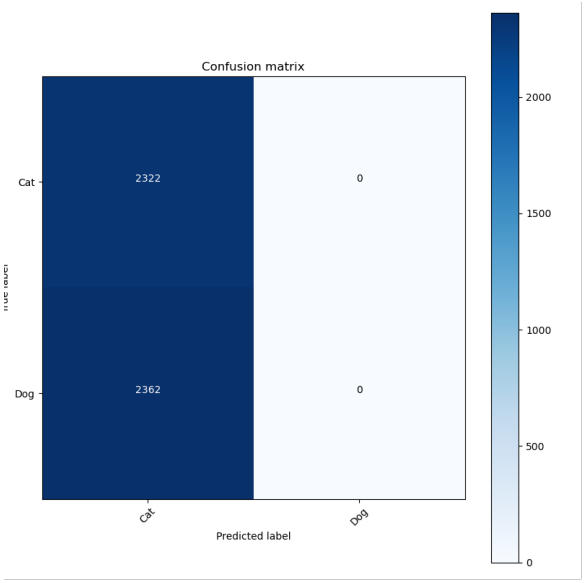

Figure 8: Initial Model of Cats vs Dogs

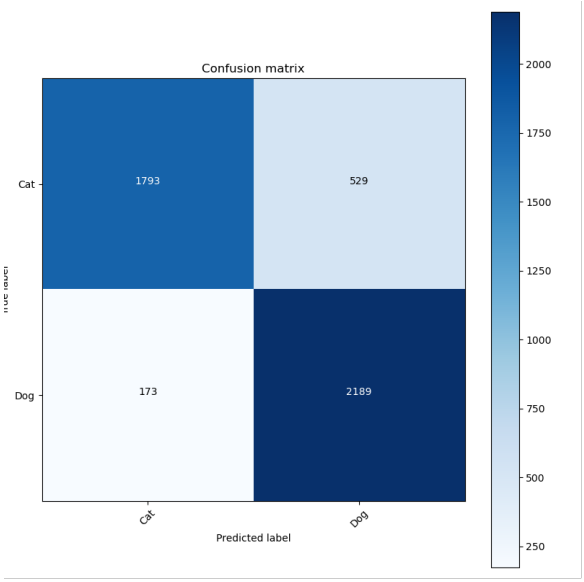

Figure 9: Final Model of Cats vs Dogs

11 Multi-Class Classification

The transition between binary and multi-class classification was one of the most troubling parts of this project as developing a model that can classify more than two classes became a tricky task very quickly. Accomplishing this did indeed allow to strengthen the model as there are many diseases and medical issues, and it is very difficult to classify them using binary classification. This task involved a lot of research into how to incorporate multiple classes and accurately predict them.

11.1 Techniques for Multi-Class classification

The initial problem encountered in this project was that the model wasn't using the correct activation as there is many that can be used like softmax or sigmoid. This involved refining the model to distinguish between initially thirteen different classes that represented various conditions detectable from X-ray, MRI, and CT scans.

The following changes needed to occur:

Output Layer: In the final layer of the CNN, the transition to using a softmax function was necessary, replacing the previous sigmoid function. This was done to ensure that the probability will be distributed between the number of classes and that the sum of all the probabilities is equal to one which is necessary due to probabilities only being between 0 and 100 percent this will allow it not to be over the limit.

Loss Function: The loss function for binary classification is binary cross-entropy[27]. This loss function was deemed useless as the function must be able to classify multiple classes. Therefore the loss function had to be changed to categorical cross-entropy[28] to be deemed satisfactory.

11.2 Challenges and Solutions

Several challenges came up in this transition to multi-class classification.

Evaluation Metrics: The first training involved the whole dataset being split into 13 classes, which due to the evaluation metrics present in the model seemed to be biased, particularly with the classes that had more images than others. This proved the model to be incorrect in many cases, therefore splitting the dataset into multiple models was implemented as a solution to this issue.

Imbalanced Dataset: Since the classes of the dataset were imbalanced, it was imperative to come up with a solution to prevent this bias in the predictions. Following are the steps taken to overcome this problem:

1. Split datasets into four. One big model is to be able to determine if the image is an MRI, CT, or X-Ray.
2. The initial big model will determine if the input image is MRI, CT or an X-ray
3. Models for each of the datasets MRI, CT, and X-Ray.
4. From that model's prediction it will go into the subset model and then predict what it thinks the image is showing.

Results:

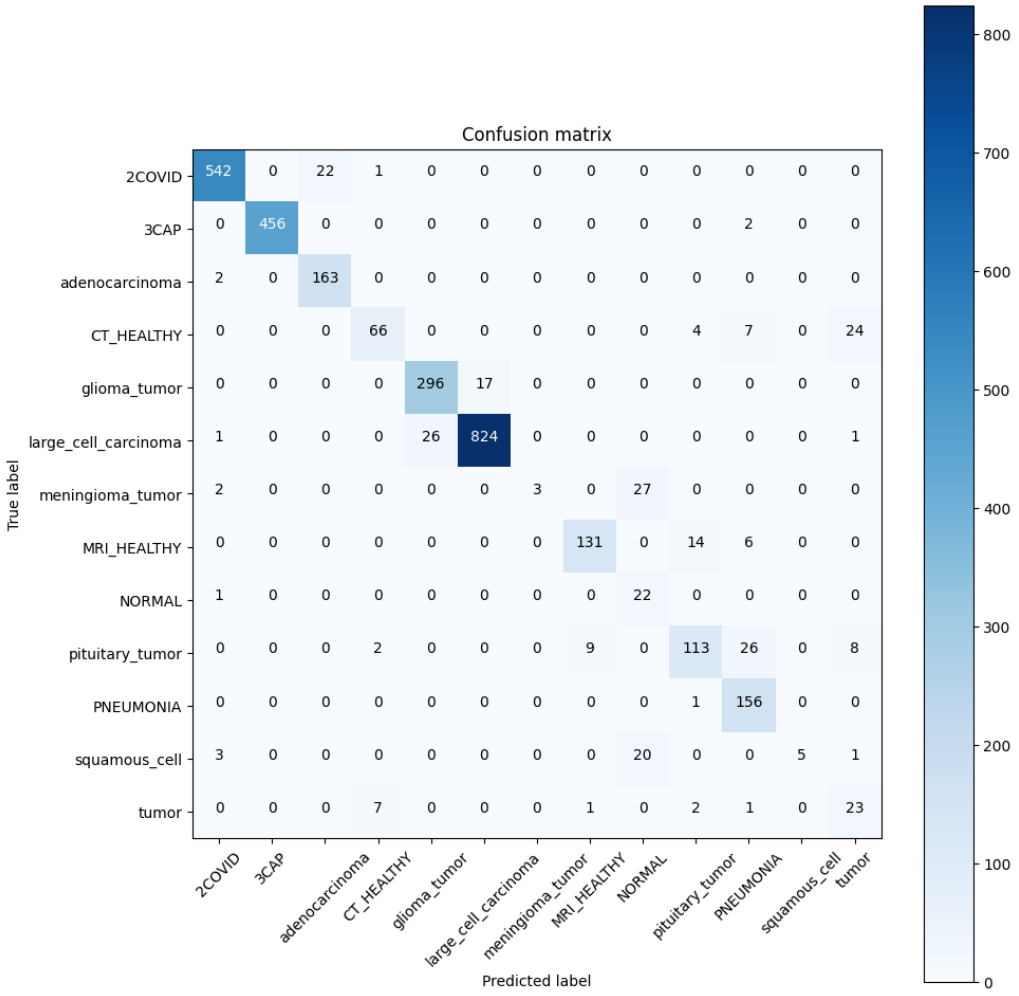

Figure 10: First Multiclass model (13 classes)

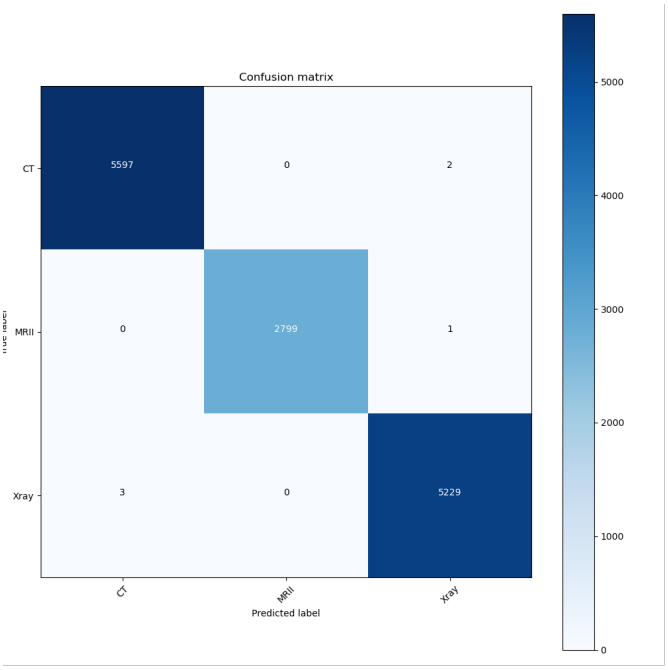

Figure 11: New Approach Classifying MRI, CT, X-Ray

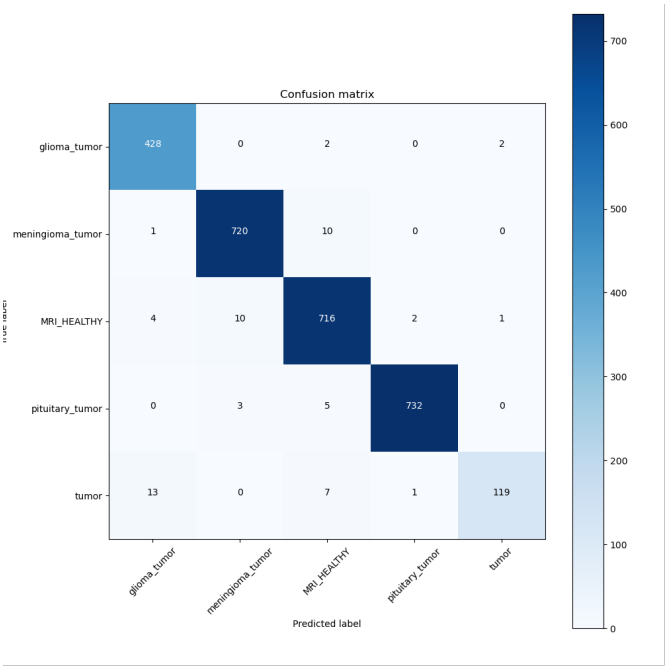

Figure 12: Classifying MRI

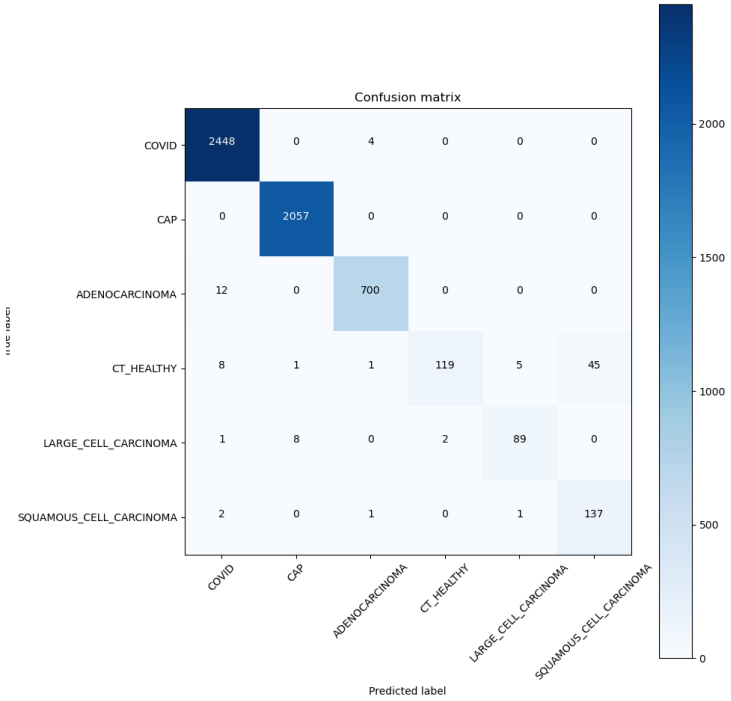

Figure 13: Classifying CT

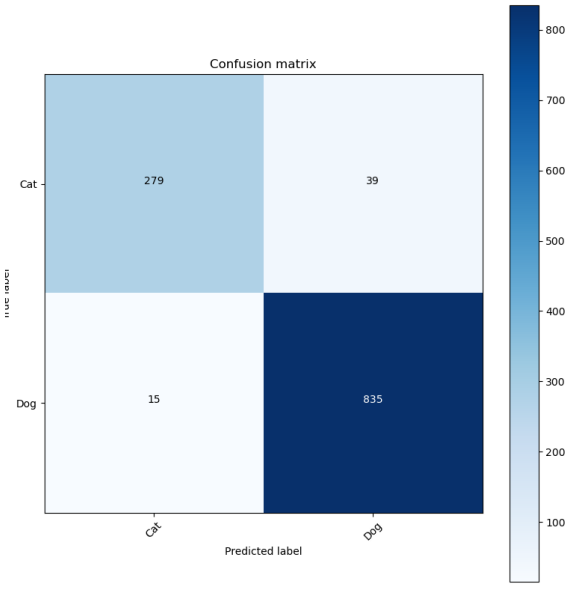

Figure 14: Classifying X-ray

12 Medical Image Classification

The project's purpose is the classification of medical images which was applied using a CNN. This is to find patterns and detect medical images which it can then classify. This section will outline what features the model is trying to detect during training and what is used to determine on how well it is detecting those features.

12.1 Challenges in Medical Imaging

Medical image classification presented many distinct challenges not commonly encountered in standard image processing tasks:

1. **Variable Image Quality:** The difference in the quality of the images was a major factor. This was due to the difference in all the equipment used and the conditions they were taken in. Therefore it was necessary to adapt the model to generalize through this.

2. **Subtle Features:** Most medical conditions have very subtle differences compared to healthy states, especially in images. One key focus of this project was to optimise workflow making sure there is a correlation between data and the model, and therefore improving the sensitivity of the model towards images with relatively weak cues or changes.

3. **Criticality of the Diagnostics:** The diagnostics hold a very high value in decision-making, meaning that the accuracy of the model needed to be very high, as any mistake could lead to a mis-classification of patients.

12.2 Adaptations

To address these challenges, several adaptations had to be made:

1. **Improved Pre-processing:** Image pre-processing of these medical images did not only include normalisation and resizing, but also, contrast adjusting and noise reduction to improve diagnosis of the features.

2. **Data Augmentation:** Since some of the diagnostic features were subtle, data augmentation came in useful to optimally and variably transform these images using specific techniques, such as, translating and rotating the image to provide more data and allow the model to detect different scenarios.

3. **Model Architecture Adjustments:** In the problems listed the model complexity and depth had to be fine-tuned to help bring out the fine patterns and characteristics of medical conditions, with particular focus at retaining high accuracy of the model.

12.3 Training and Validation

Due to the nature of this project and the fact that it focuses on medical diagnosis, there needed to be a more effective and deeper approach to training and validating the model:

1. **Diverse and annotated dataset:** The training heavily depended on accurate annotation to make sure the model learns from trustworthy data.
2. **Cross-validation:** In addition, cross-validation was performed to confirm model stability and the generalisation ability of the best parameters in the training process.
3. **Performance Metrics:** In addition to accuracy, another key measure of the model performance was recall which shows how well the model remembers or sensitivity which shows how the model is preforming in harder case scenarios.

13 Website

This project aimed to develop an AI image classifier, but in order to make it functional there was an addition to make it user-friendly. A website was developed to be incorporated into a Flask framework which allowed to perform the back-end and front-end in the same application.

13.1 Interface and User Interaction

The website was designed to offer users a very simple and efficient way of getting a clear indication of the classifier's interpretation of the image content. As such, the website layout avoids any kind of visual distractions from the core functionality of uploading images and displaying the result. This considered a design decision that had taken care of the fact that the users of this technology would come from various backgrounds and a range of medical workers with varying technology skills. This provides the easiest navigation possible.

13.2 Technical Workflow and Processing

When an image is uploaded and received, a multi-layer AI system is implemented in the web platform. Flask's back-end runs a sequence of pre-trained neural network models for each image uploaded.

It begins with first pre-processing the image into a form where the models can read the data. This then brings the image to the first detection in which it is determined if the image data is either CT, MRI, or X-ray. This data is then sent to the appropriate model which detects the image's appropriate classification.This process has undergone significant advancement, resulting in a substantial improvement in the classifier's detection rate compared to previous iterations. This is due to the initial approach having all of the trained data on one model.

13.3 Outcome and Data Handling

The resulting predictions, alongside the confidence scores, are sent back to the user clearly and concisely. Given the importance of trust and transparency in medical image diagnostics, it is imperative to offer quality guidance. Therefore, the platform applies the mechanism of viewing an uploaded image with the AI-predicted result, making it viewable for the user and providing a way of interpretation and validation of the AI decision-making process.

13.4 Website UI

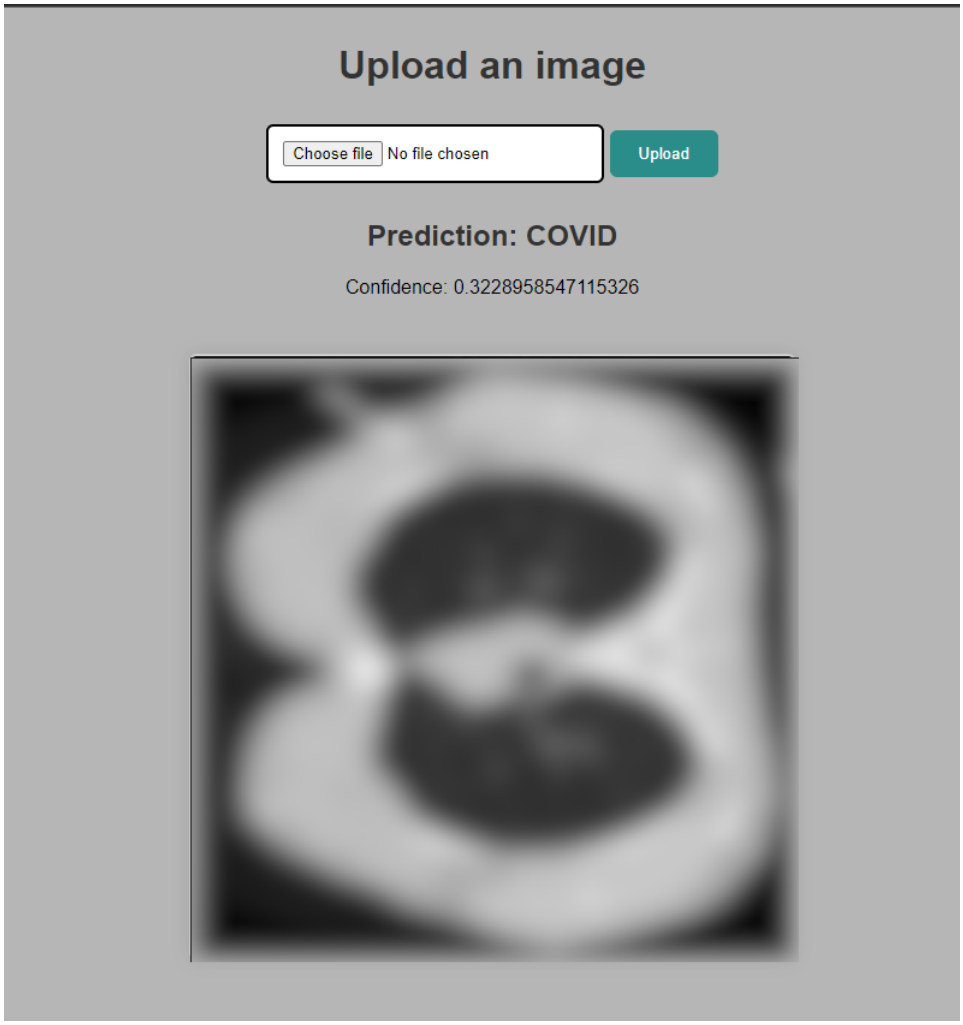

Figure 15: Website user interface

14 Code Review

This section is designed to show and explain pieces of code written throughout this project.

14.1 Data Pre-processing

This section of code deals with pre-processing the data and separating any bad data that could be in the dataset.

14.1.1 Loading images into an array

This function attempts to load an image from a specified file path, resize it to a target size (defaulting to 180x180 pixels), and normalize its pixel values to the range [0, 1]. If successful, it returns the processed image as an array. If an error occurs during loading or processing, it prints an error message and returns None.

```
def load_and_preprocess_image(file_path, target_size=(180, 180)):
    try:
        img = image.load_img(file_path, target_size=target_size)
        img_array = image.img_to_array(img)
        img_array /= 255.0
        return img_array
    except Exception as e:
        print(f"Error loading image: {e}")
        return None
```

Figure 16: Loading images to an array

14.1.2 Preprocessing

This code processes images across multiple categories by resizing them to 180x180 pixels and saves the pre-processed images into corresponding output directories. It iterates through specified folders, loads and pre-processes each image, and updates counters for processed and skipped images based on success. Finally, it prints out the total number of processed and skipped images.

```
num_processed = 0
num_skipped = 0

for folder_name in ("2COVID", "3CAP", "adenocarcinoma", "CT_HEALTHY", "glioma_tumor",
           "large_cell_carcinoma", "meningioma_tumor", "MRI_HEALTHY",
           "NORMAL", "pituitary_tumor", "PNEUMONIA", "squamous_cell_carcinoma", "tumor"):
    input_folder = os.path.join("pictures", folder_name)
    output_subfolder = os.path.join(output_folder, folder_name)

    if not os.path.exists(output_subfolder):
        os.makedirs(output_subfolder)

    for fname in os.listdir(input_folder):
        fpath = os.path.join(input_folder, fname)
        img = load_and_preprocess_image(fpath, target_size=(180, 180))

        if img is not None:
            output_path = os.path.join(output_subfolder, f"{fname[:-4]}_preprocessed.jpg")
            tf.keras.preprocessing.image.save_img(output_path, img)
            num_processed += 1
        else:
            num_skipped += 1

print(f"Processed {num_processed} images. Skipped {num_skipped} images.")
```

Figure 17: Preprocessing Images

14.2 Splitting Data

This snippet creates TensorFlow datasets for training and validation from images located in the directory. It uses a 20 per cent validation split to separate the data into training and validation datasets. The subset="both" makes sure it will prepare both training and validation datasets. The images are resized to 180x180 pixels, and data is batched with a size of 64. A seed is set to ensure reproducibility of the split.

```
image_size = (180, 180)
batch_size = 64

train_ds, val_ds = tf.keras.utils.image_dataset_from_directory(
    "Split/CT",
    validation_split=0.2,
    subset="both",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
```

Figure 18: Splitting data code

14.3 Model Code

This function constructs a convolutional neural network for image classification with Keras. It begins with a re-scaling layer, followed by a convolution layer for the first layer of feature extraction, and grows via stacks of separable convolutions, deepening in each stack and employing batch normalization and ReLU activations after each stack. It has residual connections to help the flow of gradient and closes with global average pooling and the dense output layer, adjusting between sigmoid or softmax activation based on the class number. A dropout layer was also included just before the output layer to enable regularization against over-fitting. The model structure is designed for efficiency and effectiveness in learning from image data.

```python
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)

    x = layers.Rescaling(1.0 / 255)(inputs)
    x = layers.Conv2D(128, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x

    for size in [256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
            previous_block_activation
        )
        x = layers.add([x, residual])
        previous_block_activation = x

    x = layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.GlobalAveragePooling2D()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes

    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)


model = make_model(input_shape=image_size + (3,), num_classes=6)
```

Figure 19: Model code snippet

14.4 Training Code

This code configures and trains a neural network model for image classification, specifying 100 training epochs. It uses a callback for early stopping; hence stopping training if the value of the validation loss does not increase by 8 epochs while restoring the weights from the best validation loss. The compiled model runs with the Adam optimizer having a learning rate of 0.001. It uses sparse categorical cross-entropy for the loss function since it would be preferable to the set metrics of accuracy that matter. The training then proceeds on the train_ds dataset, with the val_ds for validation. After training, the model is saved to 'model.keras', enabling later use or evaluation.

```
epochs = 100

callbacks = callbacks.EarlyStopping(
    monitor='val_loss',
    patience=8,
    restore_best_weights=True
)
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss= "sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
model.fit(
    train_ds,
    epochs=epochs,
    callbacks=callbacks,
    validation_data=val_ds,
)
model.save('image.keras')
```

Figure 20: Training code snippet

15 Organization

The organization tool I used for this project was Scrum, I adjusted it appropriately to be functional for a one-person team. This solo adaptation of scrum remained agile in an iterative sense while being adjusted to fit a one-person workflow.

15.1 Sprint Planning

Being a solo developer, I often had to have sprint planning sessions regarding the goals for every sprint. This required a lot of reflective work to set realistic yet challenging goals and keep up with the right balance of development, testing, and refinement with the constraints of sprint length, without having to consult anyone.

15.2 Stand-ups

In place of the classic team stand-up, I remained focused on my daily routine where I would look over the sprint keep it updated with the work I have done and identify any possible roadblocks that could affect my progress

15.3 Sprint Reviews

At the end of each sprint, I rated my work which I conducted according to the goals I set my main goal with an Epic, which was set for a period of 1-2 sprints max. This self-evaluation was useful to get a picture of how effective the workflows I made were, and to adjust them if needed.

15.4 Burn-down Reports

15.4.1 First Burn-down Report

This was my first sprint conducted in November where I started my project I was trying to familiarise myself with the software so this wasn't a very effective sprint in terms of visual represenation:

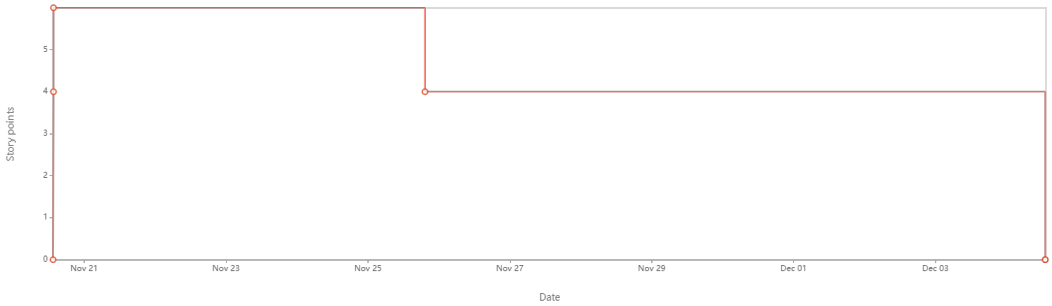

Figure 21: Burndown Report 1

15.4.2 Fourth Burndown Report

This burndown report was made in December when I started getting better at using the scrum software I had a certain plan and finally got into the routine of using scrum daily.

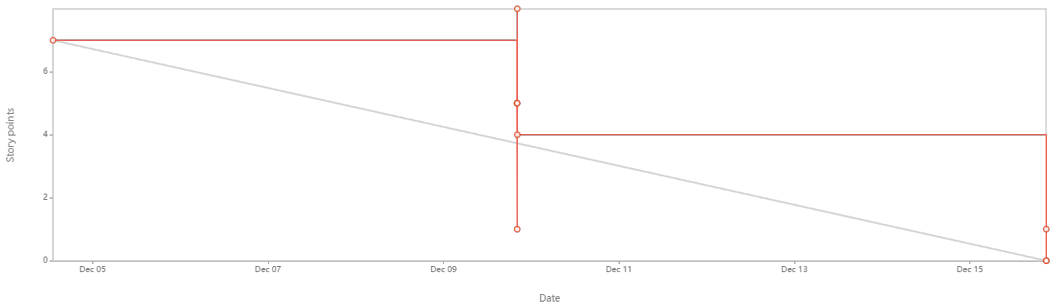

Figure 22: Burndown Report 2

15.4.3 Final Burndown Report

This is one of my final burndown reports where I had all my tasks set out and completed promptly keeping on top of my work is more visible in this report and it felt like I was more productive and there were fewer roadblocks since I was keeping on top of my tasks.

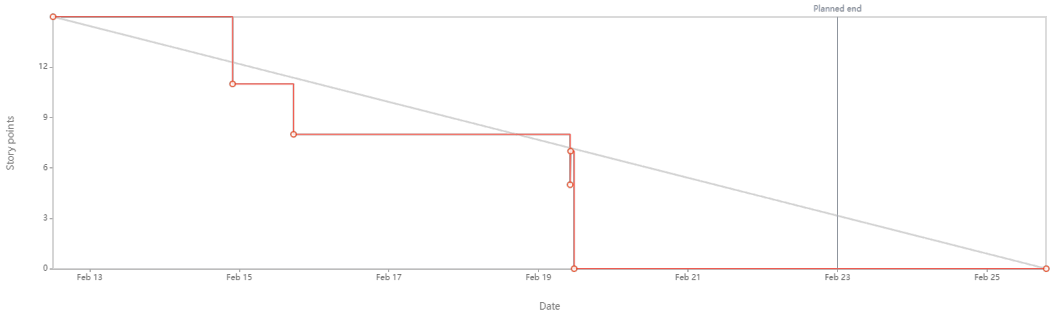

Figure 23: Burndown Report 3

16 Ethics

In a project like Medical image classification is extremely important and brings a lot of ethical considerations.

16.1 Confidentiality and Handling

1. **Patient Confidentiality:** The data collected that was previously mentioned on websites such as Kaggle have all been removed and extracted from any patient correlation under numerous acts such as the Health Insurance Portability and Accountability Act (HIPAA)[29] which is in the United States or General Data Protection Regulation (GDPR)[30]. This ensures the anonymity and security of patient data.

2. **Secure Data Handling:** The project is implemented with data handling protocols to protect any medical data that could be used with malicious temptations. Any images uploaded to the website are not stored on the server it uses base64 encoding and once the image is predicted the data gets removed.

16.2 Bias and Fairness

1. **Dataset Diversity:** In respect to that AI can produce biased results, special attention was taken to develop a fairly representative and diverse dataset by acquiring data from different sources. It was done to motivate inclusivity and reduce the bias to a minimum.

2. **Algorithmic Fairness:** Efforts in maintaining an appropriate level of transparency in the AI model, will allow for an understanding of how the decisions are made by the AI. This transparency of the model is important to let others feel confident when trying to interpret the model.

17 Conclusion

Following this project, the ambitious journey of harnessing artificial intelligence to assist in improved medical diagnostic processes, through the classification of medical images. The project has exposed the huge potential of AI in recognizing patterns within X-rays, MRI, and CT scans with the kind of accuracy and efficiency that could, in reality, assist medical professionals with their work.

Starting from baby steps - binary classification with a very elementary cats-vs-dogs model to building a sophisticated multi-class model, the project has touched a few complex parts of AI development: model architecture design, data pre-processing, and implementation of advanced techniques like early stopping to combat over-fitting. The resulting output is a user-friendly web platform with a compact and easy interface to allow for the upload of medical images for analysis with AI, which gives a result within seconds.

This shows that even a single person can develop and execute complex AI systems effectively with a structured and disciplined approach.

Issues of ethics in data privacy, security, and minimization of bias have been handled with a lot of strictness through development. The project sets the bar high for ethics in the development of AI in healthcare, which rests on secure data handling, representing datasets, and algorithmic transparency.

The results of this project show that integration of AI in a medical context would be successful, where it would only justify the technologies as helpful assistants for health professionals and one leap toward more trustworthy and faster diagnostic service.

As AI continues to evolve, so will this project as this will open a world of possibilities in the potential future by integrating larger datasets, exploring transfer learning[31], and collaborating with medical institutions on clinical validation. As such, this stands not simply as a testimony to what can be done but, in very real ways, a way towards future progress within the vast applications of AI in healthcare.

18 Appendices

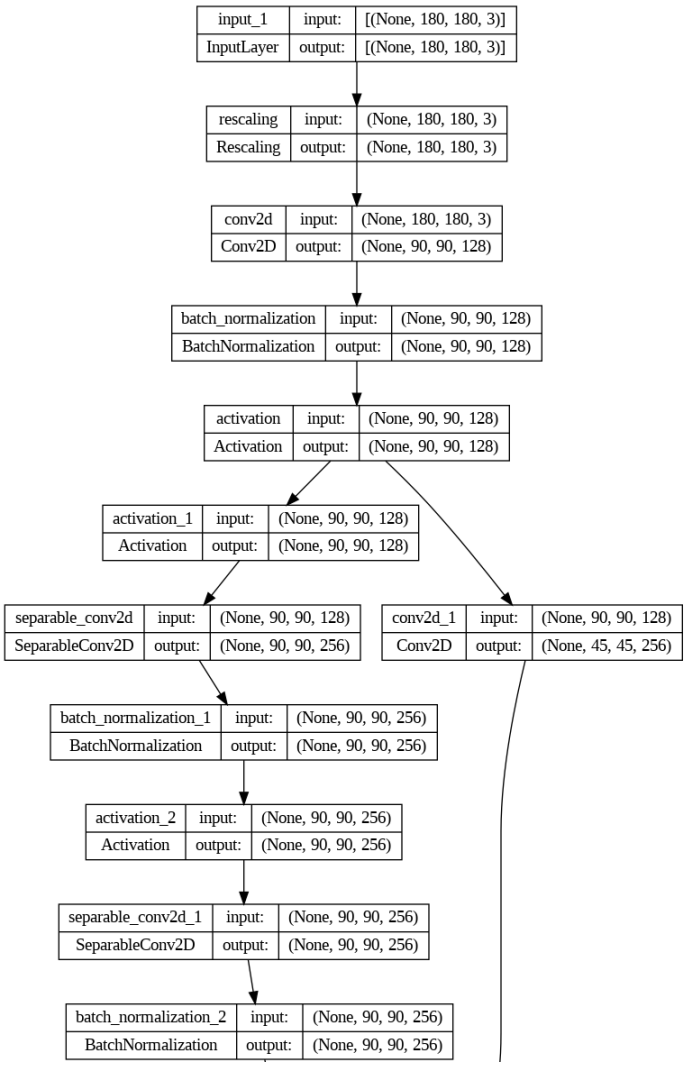

Figure 24: Model Plot pt.1

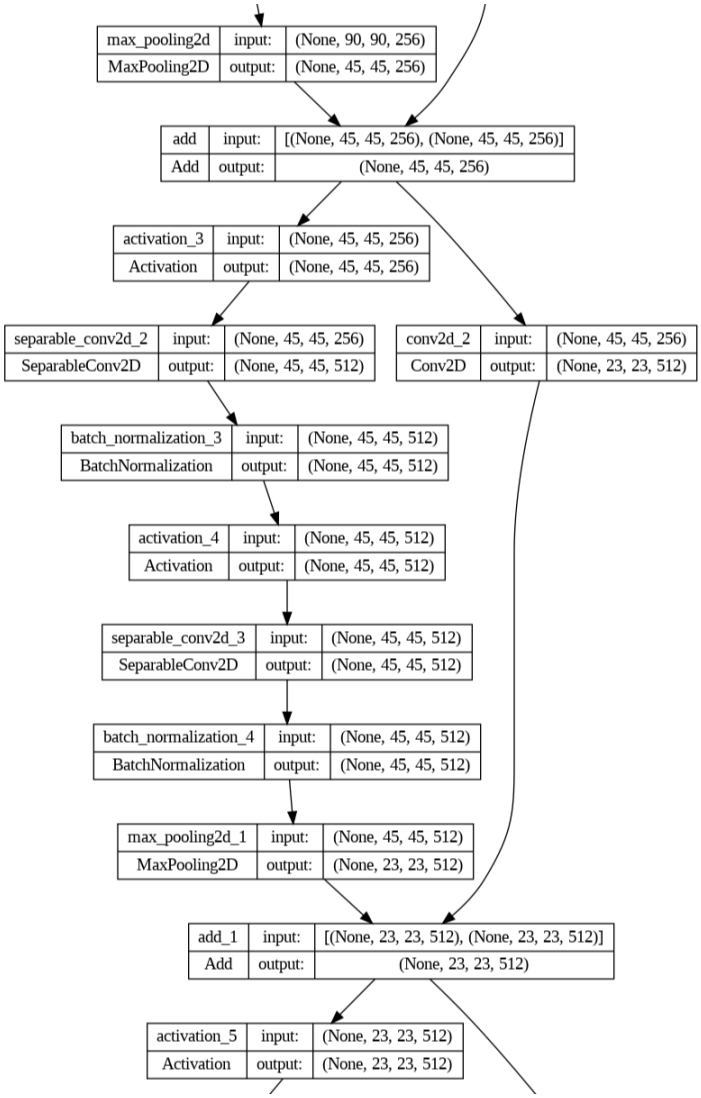

Figure 25: Model Plot pt.2

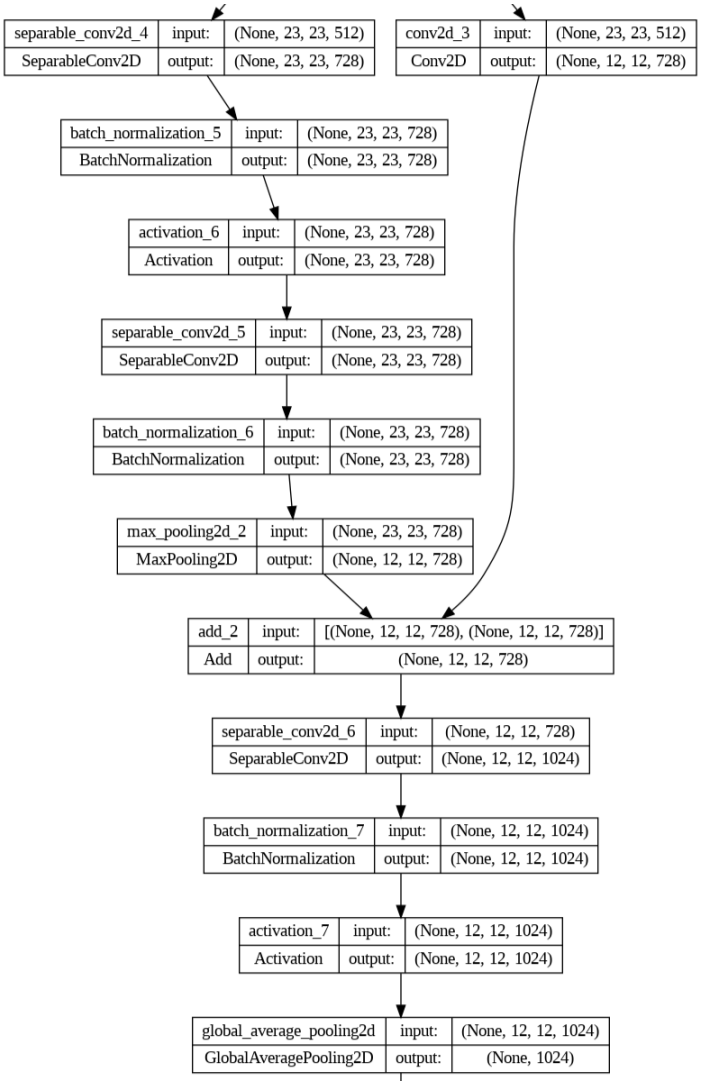

Figure 26: Model Plot pt.3

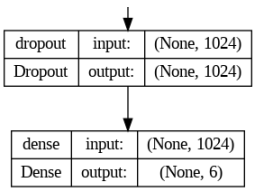

Figure 27: Model Plot pt.4

19 References

References

[1] Wikipedia Contributors (2019). Convolutional neural network.

[online] Wikipedia. Available at:

https://en.wikipedia.org/wiki/Convolutional_neural_network.

[2] www.linkedin.com. (n.d.). What is Neural Networks? — Neural Networks + AI - Brains Behind the Bots: Magic of Neural Networks in the World of AI. [online]

Available at:

https://www.linkedin.com/pulse/what-neural-networks-ai-brains-behind-bots-magic-world#:~:text=Neural%20networks%20are%20a%20class.

[3] Dertat, A. (2017). Applied Deep Learning - Part 4: Convolutional Neural Networks. [online] Towards Data Science.

Available at:

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584b

[4] Kelta, Z. (2023). An Introduction to Convolutional Neural Networks (CNNs). [online] www.datacamp.com.

Available at:

https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns.

[5] Sarkar, A. (2021). Understanding Depthwise Separable Convolutions and the efficiency of MobileNets. [online] Medium.

Available at:

https://towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficien

[6] Wang, C.-F. (2018). A Basic Introduction to Separable Convolutions. [online] Medium.

Available at:

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728.

[7] www.dremio.com. (n.d.). Sigmoid Function — Dremio. [online]

Available at:

https://www.dremio.com/wiki/sigmoid-function/#:~:text=The%20Sigmoid%20Function%20is%20commonly.

[8] Belagatti, P. (2024). Understanding the Softmax Activation Function: A Comprehensive Guide. [online] SingleStore.

Available at:

https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/#:~:text=The%20softmax%20function%2C%20often%20used.

[9] Team, K. (n.d.). Keras documentation: Image classification from scratch.

[online] keras.io. Available at

https://keras.io/examples/vision/image_classification_from_scratch/.

[10] TensorFlow (2019). TensorFlow.
[online] TensorFlow. Available at:
https://www.tensorflow.org/.

[11] Keras (2019). Home - Keras Documentation.
[online] Keras.io. Available at:
https://keras.io/.

[12] Numpy (2009). NumPy.
[online] Numpy.org. Available at:
https://numpy.org/.

[13] Author, Jeffrey A.C. (PIL. (n.d.). Pillow: Python Imaging Library (Fork).
[online] PyPI. Available at:
https://pypi.org/project/Pillow/.

[14] Flask (n.d.). Welcome to Flask — Flask Documentation (3.0.x).
[online] flask.palletsprojects.com. Available at:
https://flask.palletsprojects.com/en/3.0.x/.

[15] Author, Maede Maftouni. www.kaggle.com. (n.d.). Large COVID-19 CT scan slice dataset.
[online] Available at:
https://www.kaggle.com/datasets/maedemaftouni/large-covid19-ct-slice-dataset.

[16] Author, Mohamed Hany. www.kaggle.com. (n.d.). Chest CT-Scan images Dataset.
[online] Available at:
https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images.

[17] Author, Navoneel Chakrabarty. www.kaggle.com. (n.d.). Brain MRI Images for Brain Tumor Detection.
[online] Available at:
https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/.

[18] Author, Sartaj. www.kaggle.com. (n.d.). Brain Tumor Classification (MRI).
[online] Available at:
https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri.

[19] Author, Alif Rahman. www.kaggle.com. (n.d.). chest-xray-dataset.
[online] Available at:
https://www.kaggle.com/datasets/alifrahman/chestxraydataset.

[20] Author, Aleks Kijewski www.youtube.com. (n.d.). FYP v1.
[online] Available at:
https://www.youtube.com/watch?v=bz0M0rKOHxo.

[21] Author, Roy6801 (n.d.). gi-scraper: Google Image Scraper.
[online] PyPI. Available at:
https://pypi.org/project/gi-scraper/.

[22] Chollet, F. (n.d.). Xception: Deep Learning with Depthwise Separable Convolutions.
[online] Available at:
https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf.

[23] paperswithcode.com. (n.d.). Papers with Code - Depthwise Convolution Explained.
[online] Available at:
https://paperswithcode.com/method/depthwise-convolution#:~:text=Depthwise%20Convolution%20is%20a%20type.

[24] Wikipedia. (2021). Feature learning.
[online] Available at:
https://en.wikipedia.org/wiki/Feature_learning.

[25] Amazon Web Services, Inc. (n.d.). What is Overfitting? - Overfitting - AWS.
[online] Available at:
https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine.

[26] www.ibm.com. (n.d.). What Is Underfitting? IBM.
[online] Available at:
https://www.ibm.com/topics/underfitting#:~:text=Underfitting%20occurs%20when%20a%20model.

[27] Godoy, D. (2018). Understanding binary cross-entropy / log loss: a visual explanation.
[online] Towards Data Science. Available at:
https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanatio

[28] www.v7labs.com. (n.d.). Cross Entropy Loss: Intro, Applications, Code.
[online] Available at:
https://www.v7labs.com/blog/cross-entropy-loss-guide#:~:text=Categorical%20Cross%20Entropy%20is%20also.

[29] CDC (2022) Health insurance portability and Accountability Act of 1996 (HIPAA) — CDC, www.cdc.gov.
[online] Available at:
https://www.cdc.gov/phlp/publications/topic/hipaa.html#:~:text=The%20Health%20Insurance%20Portability%20and.

[30] GDPR (2018) General data protection regulation (GDPR), General Data Protection Regulation (GDPR). Intersoft Consulting.

[online] Available at:

https://gdpr-info.eu/.

[31] Wikipedia Contributors (2019) Transfer learning, Wikipedia. Wikimedia Foundation.

[online] Available at:

https://en.wikipedia.org/wiki/Transfer_learning.