WEAKLY SUPERVISED SEGMENTATION OF BRAIN TUMOURS USING K-MEANS GENERATED LABELS

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF BACHELOR OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2024

Ziggy Hughes

Supervised by Dr Fumie Costen

Department of Computer Science

Contents

List of Tables

List of Figures

Abstract

Brain tumours are a rare condition and can be difficult even for experienced medical Doctors and Radiologists to detect in medical images such as CT (Computed Tomography) and MRI (Magnetic Resonance Imaging) scans. This is because medical doctors and radiologists do not encounter these cases so often. This is where the AI (Artificial Intelligence) system comes in. AI systems are capable of performing the detection of tumours to discover their location, shape and size. Once medical doctors understand the level of the tumour, they can use their medical expertise to decide the next steps for treatment. Sometimes, medics conclude that the safest option is to leave tumours in the brain and to keep monitoring the size. The accurate segmentation of medical images with a brain tumour present is therefore important as the segmentation can reveal all the information the medical doctors need to make an informed decision. This project will develop a computationally-light, low-complexity and quick AI system for the segmentation of medical images of brains with tumours. Whilst pixel-level label production is extremely time consuming, the image-level labelling or bounding box-type labelling is not too difficult for radiologists and doctors. Based on the general need to reduce the workload whilst maintaining the accuracy of detection for doctors, this project focuses on the segmentation of tumours using bounding box-type labels rather than pixel-level labels. More specifically it focuses on the use of K-means clustering to enhance bounding boxes to more closely resemble the target ground truth. The project demonstrates that with the use of this improvement, we can outperform standard bounding boxes and rival the pixel-level labels without the requirement of increased workload to medics or costs attributed to the generation of pixel-level labels. This experimentation shows a rise in the intersection over union from 0.776 to 0.807 when switching from bounding boxes to this K-means method, despite requiring no extra training data or human input.

Acknowledgements

I would like to thank my supervisor Fumie Costen for her support and guidance during the development of this project. I would also like to show my gratitude towards my friends and family for their encouragement and motivation.

Chapter 1

Introduction

1.1 Motivation

Brain tumours occur when cells mutate and multiply at an uncontrollable rate [28]. This growth can be severely harmful due to healthy cells becoming overrun resulting in damage to key parts of the brain. Depending on the rate of replication tumours can be classified into four common groups, one through four. The higher the grade the more dangerous, types one and two being non-cancerous benign tumours and types three and four being cancerous malignant tumours. The survival rate of patients diagnosed with cancerous brain tumours is low, especially for those above the age of forty [39]. For this reason, it is imperative to detect tumours as early as possible so treatment can be received before the tumour gets too large and too much damage is done. Depending on the tumour, different approaches can be taken therefore knowing what areas of the brain the tumour has spread to as well as an approximate size can also be vital in deciding upon the most appropriate response. A common tool to visualise a patient's brain is an MRI scan giving doctors a 3D image to analyse. Unfortunately, a Doctor's time tends to be limited and costly therefore the development of a programme capable of analysing these 3D MRI scans on behalf of the doctor and reporting back a detailed annotation of the tumour's location and size can save time and money. The result could be a system able to scale to the demand of thousands of patients globally and support professionals in their crucial work. Image segmentation is the field of splitting an image into distinct regions, typically a background and an object of interest. It has been applied to many tasks including self-driving cars, satellite image analysis and facial recognition. Common image segmentation models require detailed training data annotated with the pixel locations of the object of interest, in our case a brain

tumour. To generate accurate segmentation is expensive as it requires the work of a skilled and experienced professional. One alternative is the use of weakly supervised models that choose to trade some performance for the benefit of simpler training data. Image-level labels and bounding boxes still require a skilled professional to create but are far less time-consuming thus lowering the cost and increasing the availability. Our main objective is to investigate improvements in weakly supervised methods to lower the performance gap between these and regular fully supervised methods.

1.2 Objectives

The aims of this project are as follows:

1. Research existing work in the field of weakly supervised image segmentation as well as understanding the foundations of brain tumours and medical imaging technology. This involves finding models and common techniques that have been tried and tested by peer-reviewed sources and assessing their application to the task of brain tumour segmentation. We would also like to record the effectiveness of these models to use as a benchmark when evaluating any potential improvements.

2. Collect and preprocess data that can be used to assess our success in improving upon existing methods. This involves the ethical sourcing of brain images of patients with tumours and the accompanying ground truths.

3. Implement a neural network that can effectively carry out image segmentation given valid training data, both strongly supervised and weakly supervised.

4. Use both our model and dataset to evaluate the effectiveness of our improved training labels in comparison to fully supervised alternatives and assess the real-life benefits our model could exhibit.

The final output of the project should be a machine learning model capable of accurately segmenting brain scans to detect tumours when trained with either strong or weak labels. The model should perform well given both types of labels without a significant drop in accuracy.

1.3 Structure

- In chapter 2 we discuss the background knowledge required to understand our following work. This includes detailed overviews of both brain tumours and machine learning techniques. First, we explore important characteristics of brain tumours including their types, symptoms and detection methods. Next, we give an introduction to common machine learning techniques and terminologies. We finally analyse existing work in the field of image segmentation and summarise our findings.
- Chapter 3 details the approach we have chosen to take. We explain our chosen dataset and the preprocessing techniques used to generate training labels as well as the implementation and training of our model.
- In chapter 4 we give precise specifications for our model and the environments we worked under.
- Chapter 5 comprehensively analyses the success of our chosen approaches with detailed metrics as well as side-by-side visualisations of the model's results.
- Chapter 6 suggests future work that can be undertaken to improve upon our results.
- We finally conclude with chapter 7.

Chapter 2

Background

The following chapter aims to give a comprehensive overview of both brain tumours and artificial intelligence methods to help with the understanding of our implementation. We also explore the work done by others in the field of image segmentation to highlight effective practices as well as to allow for later comparison against our model.

2.1 Brain tumours

2.1.1 What are brain tumours?

Tumours form when tissue cells grow more than they are meant to resulting in the infiltration of healthy cells [28]. The uncontrolled growth into other cells is what causes the tumour to become cancerous thus resulting in brain cancer. This is most commonly caused by genetic alteration leading to more cell duplication than decay [52]. Tumours can form all across the body but this research will only look at brain tumours. These can be tumours that start in the brain (primary) or tumours that start in other parts of the body and then spread to the brain (secondary) [19]. Symptoms of brain tumours include headaches, seizures, nausea, personality changes and eventually death [60]. Roughly one in ten people in England diagnosed with brain cancer survive more than 10 years [46] but early intervention can extend the life expectancy. Therefore detecting the tumour quickly can be vital for doctors to help the patient through treatment such as medicines, surgery or radiotherapy. Usually, medicines like steroids will be used to treat the symptoms and provide comfort whilst surgery can be used to remove all or most of the mutated tissue if it is safe to do so [40]. Either radiotherapy or chemotherapy can kill any unwanted tumour cells left behind. Radiotherapy is the process of

using radiation, usually in the form of X-rays, to kill cancer cells. For brain tumours, the most common type of radiotherapy is external radiotherapy in which a machine accurately aims beams from outside the patient. Whilst an effective method for killing unwanted cells, healthy tissue can also be damaged leading to side effects such as hair loss, sore skin and tiredness. Chemotherapy involves using powerful anti-cancer drugs such as temozolomide or procarbazine hydrochloride, lomustine and vincristine sulfate (PCV) to kill malignant cells [54]. Depending on the drug used, different side effects can be expected including vomiting, fatigue and physical pain. Sometimes doctors will decide the safest thing to do is to leave the tumour but continue monitoring it as any intervention could pose more danger than the tumour. Even if the removal of the tumour is successful it is still possible for it to return [33].

2.1.2 Types of brain tumours

Brain tumours can be described in a number of ways such as malignant or benign [34]. Benign tumours have little to no growth and therefore pose less of a risk to the surrounding cells. Malignant tumours on the other hand grow more rapidly and are more likely to spread to other parts of the brain. To identify the difference medics can monitor the tumour over time and observe any change in size. Alternatively, benign tumours can be identified by their smooth and clear edges and malignant tumours by their irregular edges caused by uncontrolled growth. These types can be further split into four grades [28]. Benign tumours can be grade one or two, the former being low-risk tumours with little growth and the latter still having slow growth but with a chance of spreading to other areas and developing into a higher grade. Grade three and four tumours are cancerous and grow more rapidly intruding on other parts of the brain. Grade four tumours are the most dangerous due to how quickly they grow and have a high likelihood of returning after treatment. More than 120 types of brain tumours have been discovered, grouped on their genetic makeup and location [8]. The most common type of malignant tumours are gliomas accounting for nearly 80 percent [50]. They start from glial cells and are notable due to their aggressive growth rate. Whilst they are most common in the cerebral lobes they can be found anywhere in the brain or spinal cord [31]. The glioma has a variety of subtypes depending on the type of glial cell it began in as well as certain genes in the cells. The three most common in adults are astrocytoma, oligodendroglioma and ependymomas [16]. Astrocytomas are the most common in adults [49], beginning in astrocyte cells whose purpose is to support neurones. These tumours can be grade two, three or four. Grade four astrocytomas are

also referred to as glioblastoma (GBM) and are very fast spreading, on average they lead to death within ten months of forming if untreated highlighting the importance of early discovery [14]. Oligodendrogliomas are less common [29], developing in oligodendrocyte cells. These tumours are less dangerous than astrocytomas as they can only appear as grade two or three. Ependymomas are also uncommon tumours, starting in ependymal cells lining the spinal cord. They can be grade one, two or three [15].

2.1.3 Detection methods

The purpose of this background section is to evaluate how tumours appear through different medical images along with common approaches for the detection of unhealthy cells. There are several ways doctors can image the brain to detect tumours. Before taking any images a patient will typically first speak to their General Practitioner (GP) about any symptoms they have developed. If the GP believes a brain tumour may be present then they will be referred to the neurology department which can do further checks as described below. Alternatively, they may present at an emergency department of a hospital and be referred from there.

2.1.3.1 Magnetic resonance imaging (MRI)

MRI scans are where magnetic fields and radio waves are used to create detailed images. A machine is used to create a strong magnetic field causing the atoms in the patient's body to become aligned. Radio waves are then sent causing the atoms to move out of position. The radio waves are then switched off causing the atoms to realign. This realignment results in energy being outputted which can be detected and translated by the MRI machine to form an image. Depending on the type of cells and tissue the atom belongs to, the speed of realignment and thus the speed at which energy is emitted will change, telling the MRI machine which type of cell is which. Multiple images can then be layered to form 3D models. To enhance the clarity of the image, a special contrast dye can be injected into the veins to highlight certain parts of the brain [22]. Varying radio frequencies can be used to create different images. MRIs have several sequences each with different radio frequencies capable of detecting different features. Common sequences include:

- T1: T1 weighted MRI (Figure 2.1)
- T1ce: T1 weighted MRI with contrast enhancement (Figure 2.2)

- T2: T2 weighted MRI (Figure 2.3)
- FLAIR: T2 weighted MRI with fluid-attenuated inversion recovery (Figure 2.4)

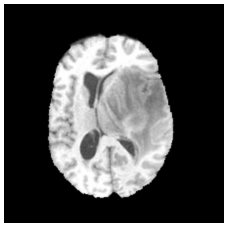

Figure 2.1: T1 MRI

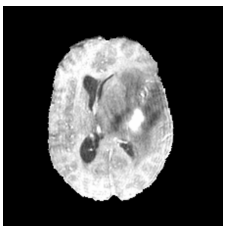

Figure 2.2: T1ce MRI

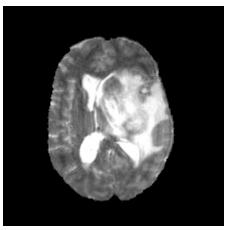

Figure 2.3: T2 MRI

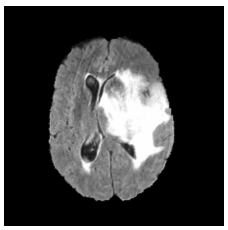

Figure 2.4: FLAIR MRI

2.1.3.2 Computed tomography (CT)

CT scans work by rapidly taking X-ray images of the brain at different angles and compiling them together to create a 3D model. Whilst not as detailed as MRI, CT scans still have their uses particularly if the patient is unable to get an MRI e.g. they have a pacemaker which can be damaged by an MRI. CT scans are especially good for looking at bone structure or bleeding. Similarly to MRI, the patient can be given a contrast dye to provide better detail in the image [17]. Figure 2.5 provides an example of a CT scan.

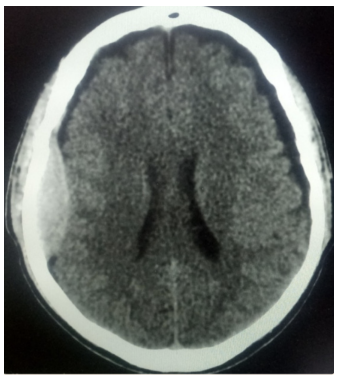

Figure 2.5: Axial non-contrast brain CT [57]

2.1.3.3 Positron emission tomography (PET)

PET scans work by injecting a small amount of radioactive substance into the body such as fluorodeoxyglucose. This is absorbed by cells but growing tumours will absorb

more than normal cells therefore when the radiation is detected it will be more dense in the tumours. This can be helpful in determining the rate of growth as faster-growing tumours will absorb more of the radioactive substance. Doctors will often combine PET and CT scans into PET-CT scans [3]. It is important to note some tumours do not take up much of the radioactive substance so a PET scan will not be appropriate to detect them. A visualisation can be seen in Figure 2.6

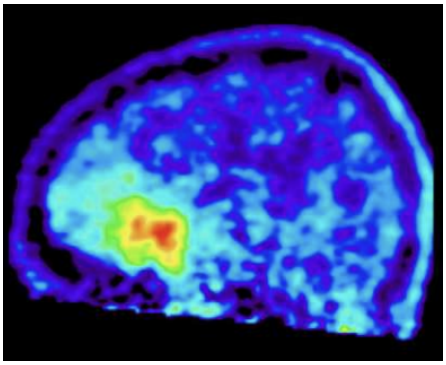

Figure 2.6: F-FET PET of brain with tumour [43]

2.2 Machine learning

Artificial intelligence has been an emerging field of computer science since the 1950s [11] and has gained wider attention in recent years due to a number of high-profile mainstream uses such as chatGPT and Amazon Alexa. With this rising popularity comes increased confusion and speculation largely as a result of the complexity of the ever-growing field. The following sections aim to provide clarity to the topic.
When deciding to implement a machine learning approach several key factors must be decided upon. These can be grouped into the following categories:

1. Data - What information will be used to train the model?
2. Task - What would you like the model to do with the data?
3. Model - What type of machine learning model best suits your task?
4. Evaluation - How will you assess if the model has successfully completed the desired task?

These factors are elaborated on below.

2.2.1 Tasks of AI

We can break down the general term AI into several unique models, all useful for different tasks. It helps to first understand the tasks AI is applied to. Four of the most common are classification, clustering, regression and segmentation.

- Classification is the process of assigning data points into a set of predefined classes for example filtering emails as either spam or not spam.
- Clustering works in a similar way but instead of giving predefined classes it simply groups based on similarity into a set number of groups. One real world use of clustering is by marketing agencies to group potential customers to better suit targeted advertisements [4].
- Regression models can predict continuous trends in data given suitable training samples such as predicting the value of a share on the stock market.
- Finally, we have the task this paper aims to address, segmentation. Segmentation is the process of splitting an image into multiple parts, this can be effective for detecting objects of interest, in our case unhealthy cells in a brain.

2.2.2 Training AI

A key principle of machine learning is the requirement of data to learn from. We call this the training data and it can sometimes be limited or costly to produce. Depending on the detail of the training data we can classify machine learning models into fully supervised, weakly supervised or unsupervised. Fully supervised models require training data as informative as the expected output. For image segmentation, this means pixel-level labels in which every pixel's true class is known. Weakly supervised only requires training data with some information for example image-level labels or bounding boxes. This would mean giving a list of classes present in an image or drawing approximate boxes around the objects of interest respectively. Some tasks can be undertaken with no training data at all, known as unsupervised learning. A common unsupervised task is clustering as the model only uses the similarity of data points to group objects. Once we have our data we can now train our model. The data collected along with their fully supervised or weakly supervised labels can be split into data for training and data for testing. It is important to keep these separate as we would like to evaluate our trained model using data points that have never been seen before to

ensure the generalisation of the model. This helps to avoid two common issues that may appear, underfitting and overfitting. Underfitting, as demonstrated in Figure 2.7, is where the model has not learned the relationship between the input data and the true outputs so can not properly make predictions. It can happen when not enough training has occurred. Alternatively overfitting, seen in Figure 2.8, is when the model is overly specialised to the data used for training and struggles to generalise to unseen samples.

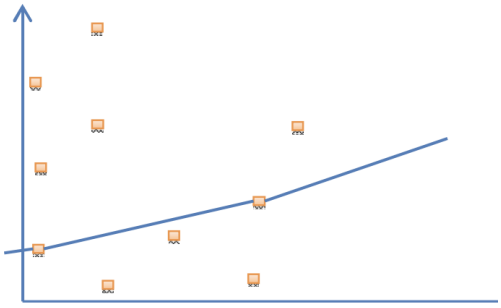

Figure 2.7: Underfitting example [25]

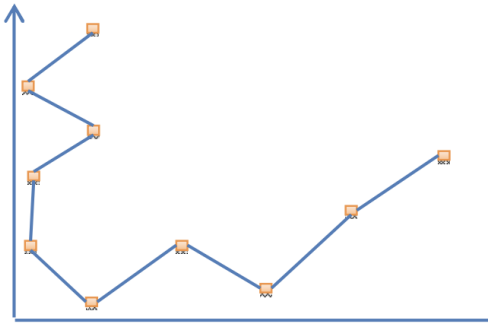

Figure 2.8: Overfitting example [25]

2.2.3 Gradient descent

To actively train the model, multiple methods are available but we will focus on gradient descent as it is computationally efficient and can scale to larger models. In the simplest abstraction, the purpose of a machine learning model is purely to make the value of a loss function as small as possible. This loss equation is chosen to accurately assess the success of the model for a given task. Gradient descent works by calculating the gradient of the loss equation at its current state and repeatedly updating the parameters of the model in the opposite direction of this gradient. Therefore given a large gradient, we will take large steps as it is assumed we are far away from the local minimum. On the other hand, given a small gradient we can predict we are close to a desired value so take smaller steps. We control the size of these steps using a learning rate or step size. The purpose is to approach a local minimum as quickly as possible but without overshooting and failing to reach convergence, a demonstration of which is given in Figure 2.9. We calculate the current gradient using the entire dataset but this poses a challenge both in terms of memory usage and computational power [36]. A common alternative is to use one sample at a time, that way computation at each iteration is drastically sped up but at the cost of slower convergence [5]. This is known as Stochastic Gradient Descent (SGD). SGD can be disadvantaged by its vulnerability to

noise, individual samples can have varying gradients and without averaging these out our model could be pushed in the wrong direction in some iterations. A third method known as mini-batch gradient descent can combine the positives of both [30]. We split the data into equal batches and processed one batch at a time. This allows us to gain from gradient descent's resistance to noise as well as SGD's ability to handle large datasets.

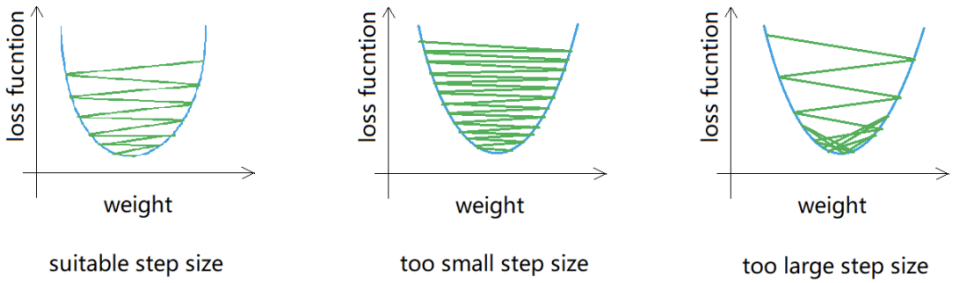

Figure 2.9: Learning rate example [9]

2.2.4 Common models

The following section describes some common AI models in the field of image segmentation, specifically clustering and neural network models.

2.2.4.1 K-means clustering

K-means clustering is a helpful algorithm for splitting a dataset into k distinct groups. It works by assigning cluster centres as starting points and through an iterative process assigning all points to their nearest cluster and recalculating the cluster centre. As the centre has changed the nearest points may also change so we repeat the process until convergence is reached or we hit a maximum number of iterations. The main benefit of this method is the lack of training data required, as it only bases its assignments on the given data values. This can make K-means a convenient tool to quickly partition a dataset without carrying out the costly process of collecting data, preprocessing and training. K-means is however sensitive to the initial cluster centres therefore careful consideration is required when deciding on a method to pick starting points.

2.2.4.2 Neural network

As the aim of AI is to simulate or replicate human intelligence in computers it is sensible to base machine learning models on the human brain. To do this we must first understand how our brains work. Information is processed through connected nodes known as neurons. The connections are known as synapses and their strength determines the flow of information. Over time their strength grows or deteriorates depending on the frequency of their use, hence why revising before an exam can strengthen your knowledge. To mimic this with computers we create layered nodes to simulate neurons and use weighted connections for synapses. The output y of a single node (or perception) with input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, bias b is given by:

$$y = \sum_{i=1}^{N} w_i x_i + b \tag{2.1}$$

Where:

- $\mathbf{x}$ is the input vector
- $\mathbf{y}$ is the output value
- $\mathbf{w}$ is the weight vector
- b is the bias term
- N is the number of input features

When training our model we repeatedly update these weights for example using a process called backpropagation to strengthen or weaken specific connections in order to recreate the process of a brain learning. Many different architectures can be used to create neural networks all with varying numbers of layers and nodes resulting in advantages and disadvantages. Selecting the best architecture for a specific task is a key step in creating a machine learning model.

2.2.4.3 Convolutional neural network

A type of neural network known as a convolutional neural network (CNN) is specialised for visual tasks as it takes inspiration from the human visual cortex. It is unique in its way of processing grid data in sections called kernels, this allows the model to learn from localised features instead of the whole dataset which could consist

of irrelevant information. There are multiple types of layers including convolution, pooling, upsampling and fully connected. Convolution kernels work by multiplying the values of the pixels they overlap with their weights and summing the values as seen in Figure 2.10. We often also add a bias value to the result.

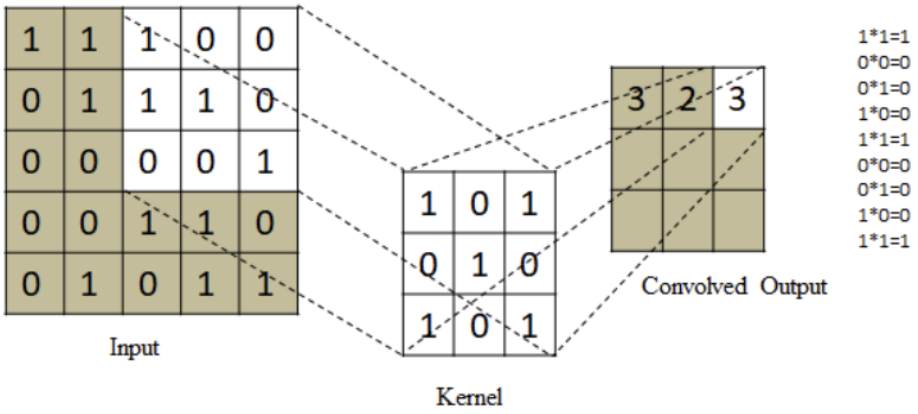

Figure 2.10: Convolution kernel example [12]

Pooling layers combine values to decrease the dimensions of the data and control the number of parameters. This can be done in various ways including max pooling (see Figure 2.11) where the highest value in the kernel is outputted or average pooling where the average of the values is used.

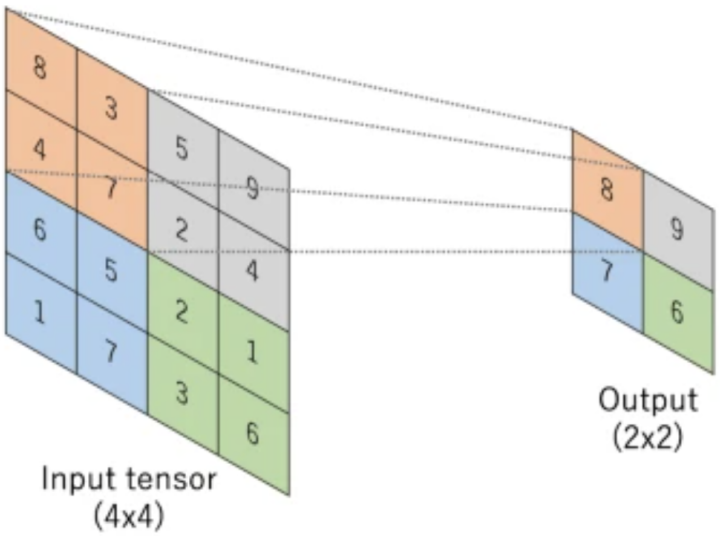

Figure 2.11: Max pooling kernel example [61]

Upsampling does the reverse of pooling to increase the resolution of data. A common

approach to this is using interpolation techniques such as nearest neighbour where the new datapoints gain the value of the nearest existing point.

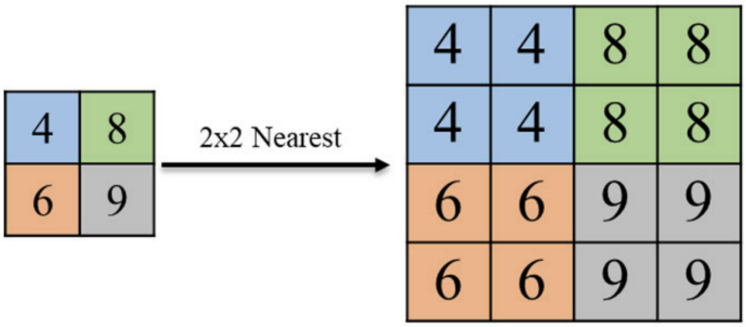

Figure 2.12: Nearest neighbour upsampling example [53]

Fully connected layers work similarly to regular neural networks as they operate on the whole data and do not use a kernel. This can allow the model to extract more general information and is often implemented as the last layer.

Implementing each kernel involves sliding it over the grid of data values and assigning the outputted value to the matching coordinates of the centre point in a new array. The amount we move the kernel each time is set by changing the stride, typically we default to one to ensure the output image is of equal dimensions. The kernel size decides how many pixels should be taken into account for example a 3x3 kernel will calculate a result depending on the pixel being analysed and its eight neighbours. A 1x1 kernel can be useful if we only wish to base our output on the individual data points. When operating near the edge of images we may exceed its dimensions resulting in an error. To address this concern we implement a method called padding in which we add the necessary rows and columns to the outside of the image. Deciding what values to fill these extra pixels with will depend on the task at hand and the kernel operation used. Some types include zero padding (see Figure 2.14) to set all numbers to zero, reflective padding (see Figure 2.15) in which the edges are treated mirrors and the values are equal to their corresponding pixels and circular padding where the kernel wraps around thus end values pad the start and starting values pad the beginning. Of these zero padding tends to be the most common due to its computational efficiency and lack of impact on the kernel's output [21]. For the purposes of this paper if padding is mentioned, assume zero padding is in effect unless otherwise stated.

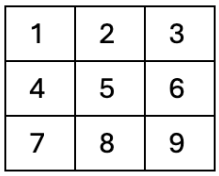

Figure 2.13: Default (no padding)

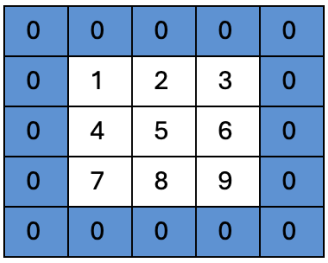

Figure 2.14: Zero padding applied

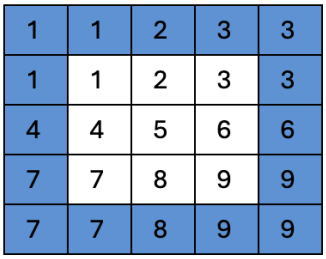

Figure 2.15: Reflective padding applied

2.2.5 Hyperparameters

Within a given machine learning model, many decisions must be made in regard to the values used for implementation. We refer to these as hyperparameters and their precise tuning is vital to create effective models. Hyperparameters are set by the programmer before running the training of the model and ideally should be thoroughly experimented with to ensure the best values are selected. Below we detail some common hyperparameters used in neural networks and their training.

2.2.5.1 Network Size and layers

The exact architecture used can be thought of as a hyperparameter. The developer must decide how many layers to include and what types of layers to use. Due to the vast number of possible combinations, it is not possible to test all architectures out and pick the best-performing one. Instead, developers must use their knowledge of machine learning to decide on an appropriate architecture. It is common practice to use an existing state-of-the-art model designed for a similar task and make modifications to tailor it to the desired application. Due to the increase in computational power, larger networks have become feasible resulting in more powerful models that can work on more complex tasks. The drawback of larger models is the increased number of weights needed to be trained thus increasing the length of training.

2.2.5.2 Layer values and activation functions

Different types of layers come with their own hyperparameters to decide upon. Convolutional layers have several parameters including kernel size, stride and padding. The choices made for each of these will likely depend on the overall architecture and the specific requirements of the task. Many layers will pass their output through an

activation function to give the model non-linearity, allowing it to learn more complex trends. The most common function is ReLU but others are available as detailed below in Figure 2.16.

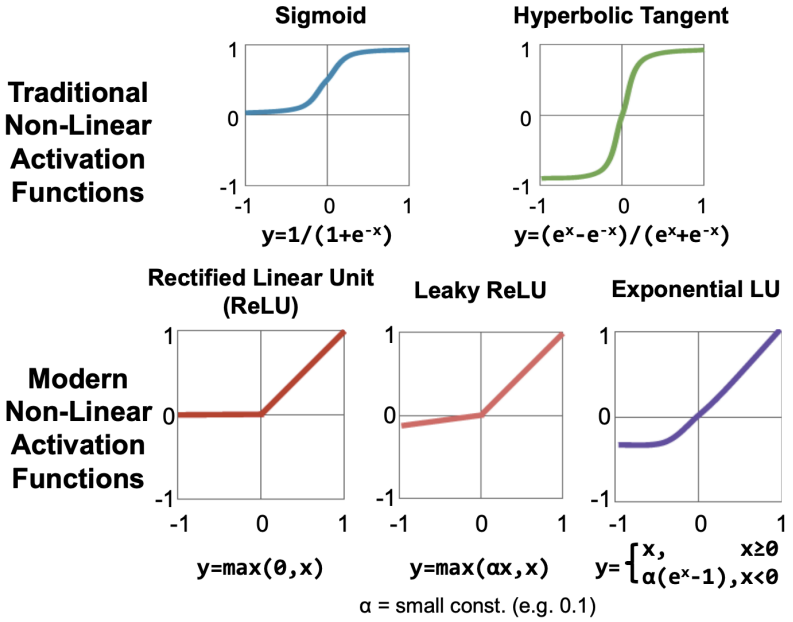

Figure 2.16: Five common activation functions [55]

2.2.5.3 learning rate and batch size

The learning rate determines how much each weight should be updated at each iteration. Larger learning rates will result in faster changes but too large and the weights could overshoot the ideal values. Optimisers can be used such as Adam (Adaptive moment estimation) [27] which calculates specific learning rates for individual weights. Batch size decides how many samples to use in each iteration when updating weights. A batch size of one is equivalent to stochastic gradient descent. When using a small batch size, a smaller learning rate should also be used.

2.2.5.4 Number of iterations

The number of iterations or epochs decides how many times the training process should loop. In the case of batch gradient descent, the entire training dataset will be used for each iteration. Larger numbers of iterations will result in higher training accuracies but too high and the model will overfit thus dropping the testing accuracy. The number of iterations is also limited by the computational power and time available.

2.3 Literature survey

In recent years much research has been done to find machine learning models that can accurately segment objects in images, particularly for brain tumours in medical scans [18]. Brain tumours are especially hard to segment as they become blurry at the edges due to their growth into normal cells as well as their ability to appear in different shapes, sizes and locations. Most research done has been into strongly supervised models such as Mohammad Havaei et al [23] and Pawel Mlynarski et al [37] which tend to have a higher classification accuracy but require training data with pixel-level labels as opposed to weakly supervised models that only require image-level labels or bounding boxes to train. As pixel-level labelled training data is limited and costly to produce we will instead look at research into weakly supervised models that have accuracies close to or equal to the strongly supervised models. Most image segmentation problems are addressed using a convolutional neural network (CNN), a deep learning technique that uses multiple layers such as convolutional, pooling and fully connected to recognise patterns in an input. The exact architecture of each neural network varies across models meaning two CNNs can have very different results for the same dataset therefore unique networks should be designed for each problem. Some architectures for weakly supervised image segmentation are detailed below.

2.3.1 Existing work

Yunhang Shen et al [51] proposes a parallel detection and segmentation learning (PDSL) model consisting of four parts: a pre-trained backbone network used to extract features from the image, a top-down object detection branch, a bottom-up image segmentation branch using bounding boxes and a correlation learning module to ensure synergy between the detection and segmentation branches. Object detection can often result in a smooth background at the cost of lower sensitivity whereas segmentation tends to have several false positives in the background but higher precision. These opposite strengths and weaknesses mean using the results of both can complement each other providing the best possible fine tuning. The model is tested on the PASCAL VOC 2012 dataset and the mean average precision (mAP) is compared with other models. The paper reports an mAP(0.50) of 49.7 which is greater than any other displayed top-down end-to-end WSIS model and some more strongly supervised models such as Khoreva et al [26], a leading segmentation model using bounding boxes.

Fatemeh Saleh et al [48] starts by using a CNN similar to the VGG-16-layer network with a few small modifications. The output gives a score to each pixel for each class. Where this model differs from others is the use of the scores to produce a foreground/background mask that can later be used for semantic segmentation. The mask is produced by taking the fourth and fifth layers and converting them from 512xWxH to WxH using average pooling then combining the two results using element-wise summing. Once scaling the results between zero and one we are left with a probability mask that each pixel is part of the foreground. The model is trained on the PASCAL VOC 2012 dataset with and without the mask and the mean intersection of union (mIoU) is displayed. By using the mask the mIoU rises from 31.0 to 44.8 and then to 46.6 when introducing CRF smoothing.

Pedro O. Pinheiro et al [45] suggests the use of an aggregation layer using a log-sum-exp (LSE) to convert the pixel-level labels computed by the network into image-level labels representing the probability of each class being present. These image-level labels are then used as an image level prior later on in the network by multiplying each pixel-level label by the image level prior for that class. The model also uses multiple smoothing layers such as superpixels to reduce noise in the final output. After training on images from ImageNet, PASCAL VOC is used for testing and the mAP is reported. When using the best smoothing prior a mAP of 40.6 is achieved. This is close to some of the best performers in the PASCAL VOC 2012 competition despite them using strong supervision, achieving scores of 47.8 to 51.6.

As strongly supervised models are more accurate at image segmentation in most cases, Jiwoon Ahn et al [1] use deep neural networks in training to convert the image-level labelled training images into pixel-level labelled images. These can then be used to train a strongly supervised neural network as if the original training data consisted of pixel-level labels. To generate pixel-level labels the model first creates class activation maps (CAMs) which show a rough prediction of the area in the image in which the object resides. Higher values relate to more certainty. These CAMs are then revised using AffinityNet to more accurately show the object's location then for each pixel in the image its class is predicted by taking the highest score from the CAM. After training and evaluating using the PASCAL VOC 2012 dataset, the mIoU is calculated giving a mean of 63.7 largely outperforming state-of-the-art weakly supervised models demonstrating the possibility to increase accuracies whilst not using more costly training data.

Deepak Pathak et al [42] propose the use of a Constrained Convolutional neural network (CCNN) that expands upon a regular CNN by adding linear constraints to the optimisation process. These linear constraints include a suppression constraint to ensure no pixels are labelled with a class not present in the image labels, a foreground constraint to ensure at least a certain percent of pixels are labelled with each class present, a background constraint to control the upper and lower bound of the pixels labelled as the background class and finally an optional size constraint for if the expected size of the object is known. These can all be controlled by hyperparameters to set the upper/lower bounds. After testing, a mIoU of 45.1 is achieved, coming close to some strongly supervised methods scoring between 51.6 and 66.4.

Jifeng Dai et al [10] opt to use bounding boxes as supervision as they provide more informative guidance to the segmentation model than pixel-level labels while maintaining the time and cost benefits of weakly supervised methods. The proposed method involves generating many candidate segmentation masks from the bounding box data and using these masks to train the convolutional neural network. The paper suggests several functions for the generation of these masks including selective search and GrabCut. The model then uses the inferred knowledge to improve upon the candidate masks and then iteratively uses the new masks for training. The model exhibits an effective performance when run on the PASCAL VOC 2012 dataset with a mean Intersection over Union of 62.0 but falls just short of strongly supervised methods with a result of 63.8 mIoU. This can however be overcome by the use of a mixed approach. By replacing ninety percent of the pixel-level labelled data in a fully supervised method with bounding boxes the paper reports almost equal results of 63.5 mIoU whilst drastically reducing the required amount of costly fully annotated data.

Hongkai Yu et al [62] implement a loose-cut method to improve upon the common GrabCut function used to separate the foreground from the background using bounding boxes. GrabCut is an effective tool but is limited in its reliance on tight boxes. This increases the quality of training data required and also rules out many computer-generated boxes. For this reason, the loose-cut method is proposed to carry out the same task but with the added benefit of allowing looser boxes. The algorithm follows a similar structure to GrabCut but introduces two features. First, it encourages differences in the foreground and the background. Second, it encourages consistent labeling

meaning two pixels of similar appearance are more likely to be labeled the same regardless of their locations. After comparing using three unique datasets, we see the loose-cut method matches GrabCut's error rate with results of 7.9 and 7.4 respectively when using tight bounding boxes. The strengths of the loose-cut method become far more significant when we increase the size of the box. With a looser box, GrabCut's error rate rises to 13.7 whilst the loose-cut method decreases to 6.8.

2.3.2 Summary of literature

1. Parallel detection and segmentation learning (PDSL) [51]
    - Data: PASCAL VOC 2012 and MS COCO
    - Preprocessing: None mentioned
    - Key features: Uses simultaneous detection and segmentation and computes a loss function based on both
2. CNN based on VGG-16-layer network [48]
    - Data: 10,582 training images, 1,449 validation images and 1,456 test images (PASCAL VOC 2012). Further training with 7238 images from a subset of the MIRFLICKR-1M dataset
    - Preprocessing: Image-level labels calculated from pixel-level labels
    - Key features: Produces a foreground/background mask to be used in the segmentation layer
3. CNN with an aggregation layer [45]
    - Data: Trained on 760,000 images from imageNet
    - Preprocessing: Random jitter added to training images (flipping/mirroring/contrast changes)
    - Key features: Calculates aggregated image-level labels and backpropagates the error
4. DNN trained with artificial labels [1]
    - Data: 10,582 images for training, 1,449 images for validation (PASCAL VOC 2012)

- Preprocessing: None mentioned
- Key features: Produces pixel-level training data from image-level labels to train a DNN

5. Constrained Convolutional neural network (CCNN) [42]
    - Data: 10,582 images for training, 1,449 images for validation (PASCAL VOC 2012). Pre-trained using the ILSVRC dataset
    - Preprocessing: None mentioned
    - Key features: CNN with added linear restrictions
6. Convolutional neural network trained with bounding boxes [10]
    - Data: 10,582 images for training, 1,449 images for validation (PASCAL VOC 2012). Pre-trained using ImageNet
    - Preprocessing: Generation of candidate segmentation masks
    - Key features: CNN iteratively trained with improving masks. Mixed supervision levels
7. Loose-cut improvement on GrabCut [62]
    - Data: 131 images from the GrabCut dataset, the Weizmann dataset and the iCoseg dataset
    - Preprocessing: Generation of bounding boxes with various sizes
    - Key features: Explicit foreground and background differences. Consistent labelling for similar pixels

Chapter 3

Methodology

Here we will explain the processes that have been undertaken from the procurement of data through to the training of the model. This includes preprocessing techniques such as K-means and thresholding as well as the structure of the UNet archetype.

3.1 Dataset

In order to train and test an image segmentation model we first need valid images along with their accompanying ground truths. We make use of the MICCAI BraTS 2020 dataset [35] consisting of 369 patients, each with four types of MRI scans (T1, T1ce, T2, FLAIR) along with the ground truth. Of these types we only use T2. A single slice of a T2 scan and the corresponding ground truth have been displayed in Figure 3.1 and Figure 3.2.

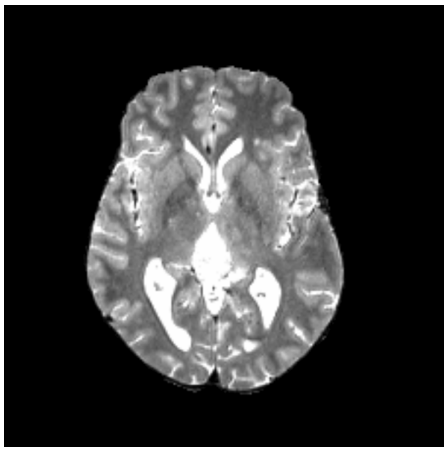

Figure 3.1: T2 scan

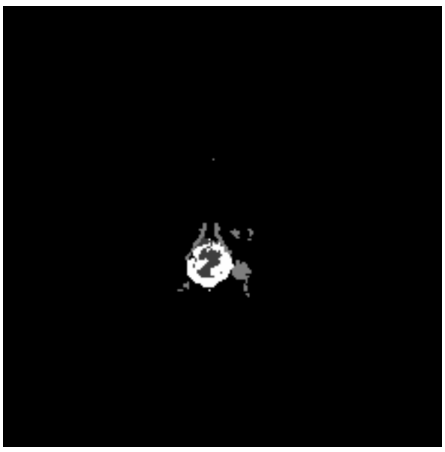

Figure 3.2: Ground truth

A scan is stored as a NiFTI (.nii.gz) file and has dimensions (240, 240, 155). The dataset has been collected and preprocessed by the Center for Biomedical Image Computing & Analytics (CBICA) ensuring all images are standardised and ready for training. Preprocessing includes skull-stripping, the process of removing non-brain tissue signals from MRI scans [24]. This allows analysis of the MRI data to focus entirely on insightful information without unnecessary skull signals. The ground truths have been labelled by hand by neuro-radiologists and contain the values zero, one, two and four depending on the type of tumour, zero meaning background/healthy cell and one, two and four being varying severities of tumour. For our experiment, we set all non-zero values equal to one as our focus is the detection of tumours, not the classification. We have selected the MICCAI BraTS dataset as it has been compiled specifically for the purpose of testing brain tumour segmentation models, resulting in a detailed set of MRI scans and ground truths that have been used across many segmentation projects. This helps provide an easy comparison to alternative weakly supervised and fully supervised models as many projects will have also trained and tested on the same dataset. We also gain assurance as to the professionalism and integrity of the data procurement process due to the reputation of CBICA and its long-standing competition along with its connection to the University of Pennsylvania. This guarantees all brain scans are collected ethically with the content of the patients and their anonymity insured.

After downloading the data we are left with three folders, each containing many sub-folders. These sub-folders represent a patient and each holds five NiFTI (.nii.gz) files, four for the MRI scans and one for the ground truth. For our model to process the data we need to rearrange the file structure, we do this using Python code. After restructuring we are left with one data folder holding two sub-folders, train and test. Train holds eighty percent of the patients selected at random whilst test holds the other twenty percent. This data split allows for 295 patients for training and 74 for testing. Within the two sub-folders are two more sub-folders named img and gt. The img folder contains the four MRI scans for each patient and the gt file contains the ground truth. Figure 3.3 and Figure 3.4 depict the structural changes we have made.

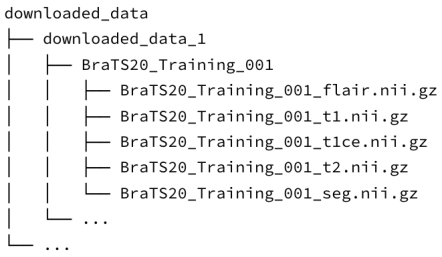

Figure 3.3: File structure before

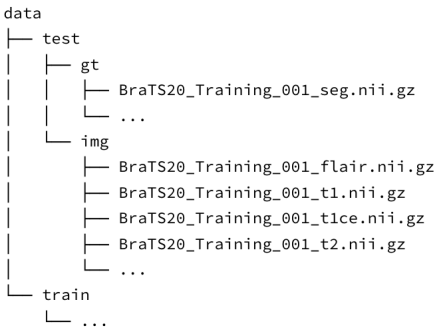

Figure 3.4: File structure after

3.2 Preprocessing

3.2.1 Formatting

Before the process of extracting weakly supervised labels can begin we must first ensure the scans will fit our model. To do this we must reshape the images by removing unwanted pixels from the edges to transform the dimensions from (240, 240, 155) to (216, 240, 155). We then modify the pixel values to go from integers to arrays of length three. This is due to our model expecting each pixel to store an RGB value so we must duplicate the greyscale value three times giving dimensions of (216, 240, 3, 155). We now remove forty slices from the top and forty slices from the bottom of the brain to reduce the quantity of data to deal with certain memory limitations of our environment. These limitations are expanded upon later in Chapter Four. Top and bottom slices were chosen due to a lower frequency of tumours as well as a smaller brain-to-background ratio giving the model less to learn from. We also must reduce the resolution of our images to fit within the memory requirements giving a final resolution of (72, 80, 3, 75). We are now ready to generate training labels for each slice in each patient's scan.

3.2.2 Generating bounding boxes

As previously explained, bounding boxes are a simple and easy to generate form of training labels and thus remain one of the most common weakly supervised learning techniques. To generate these labels for our dataset we analyse the ground truths and predict the bounding boxes that an expert would draw. We harness the Python library OpenCV along with its findContours function [38] to detect objects in our binary ground truth. The function works by detecting changes in image intensity and assuming this is the edge of an object. These contours are then returned and we create

multiple rectangles encapsulating them leaving us with several boxes surrounding the tumour's location. Any boxes smaller than a preset size are removed to stop individual pixels or small clusters of pixels from being recorded as it would be unreasonable to expect an expert to accurately label every pixel. The remaining boxes are further analysed to detect if some boxes are contained by other larger boxes. If so, the contained boxes are eliminated and all other boxes for that slice are recorded in a .csv file in the format: x coordinate, y coordinate, width, height. The resulting bounding boxes represent an educated prediction of human-drawn labels by containing the desired object without relying on many tiny boxes to contain individual pixels or clusters of pixels that are undetectable by the human eye.

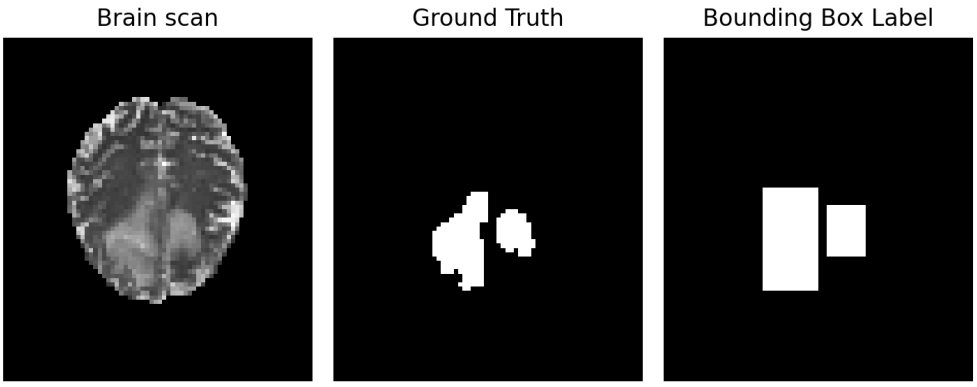

Figure 3.5: Example of bounding box generation

3.2.3 Improving bounding boxes

Our generated bounding boxes provide a good indication of the brain tumour's location but are only a rough approximation of the size and shape. Many background pixels get included in the rectangle borders which can confuse the model in training causing the misclassifications of pixels in our results. To improve upon this label we implement two methods proposed by [47] to generate better masks whilst still only requiring bounding box inputs to maintain a weakly supervised model: thresholding and k-means.

3.2.3.1 Thresholding

Thresholding is the process of assigning every pixel one of two values depending on whether its intensity is above or below a value. For our model, we loop through every pixel in each bounding box for a slice and assign values below a threshold value or above another threshold value as healthy and all pixels in between as tumours to produce a binary mask. Finally, we detect the contours in the image and draw a convex hull around them in order to fill any gaps left inside the generated shape. These bounding box masks are then stitched together in their original locations on the slice to produce one large mask for that image. An example of this can be seen in Figure 3.7.

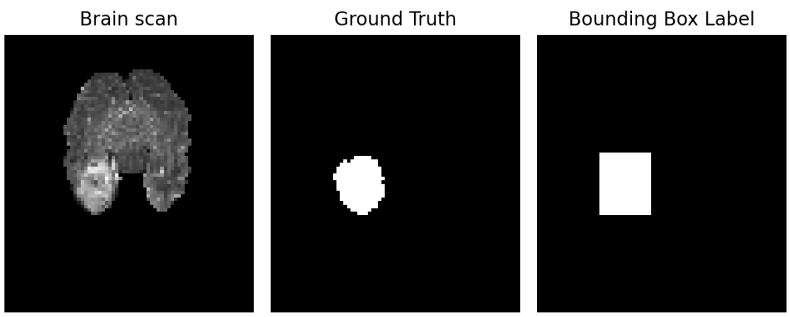

Figure 3.6: MRI brain scan, ground truth and generated bounding box

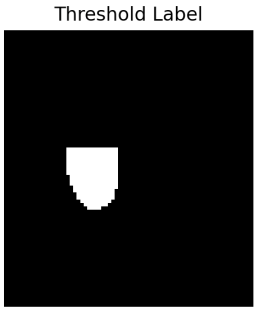

Figure 3.7: Thresholding result

3.2.3.2 K-means clustering

Our second proposed model uses k-means clustering to assign values to each pixel in the boxes. Similarly to thresholding, we analyse each bounding box independently and then stitch them together. OpenCV's kmeans function is implemented to carry out the task of classifying each pixel [38]. The function works by iteratively calculating cluster centres and assigning every pixel to its closest centre. In each loop the values in each cluster change so new centres can be calculated until we reach convergence or hit a limit on the number of iterations. In our implementation, we group all values by intensity into three clusters and assign the middle cluster as the tumour. In some cases, if the bounding box is below a certain size we will only use two clusters and assign the higher intensity cluster as the tumour. As with thresholding we now detect contours and draw a convex hull around them to fill any gaps. Once all the boxes are returned and combined we are left with a binary mask that more closely resembles the ground truth allowing for more informative training of the model. This is shown in Figure 3.9.

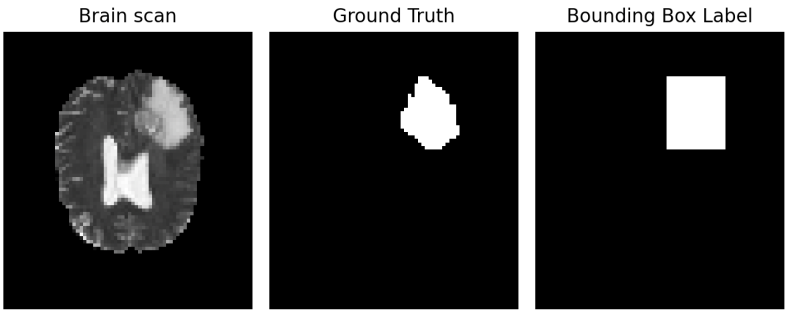

Figure 3.8: MRI brain scan, ground truth and generated bounding box

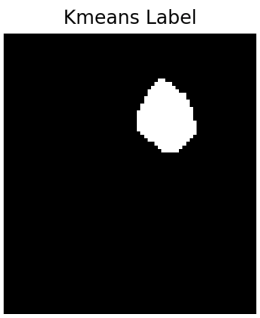

Figure 3.9: K-means result

3.3 Model

Our segmentation model follows the architecture of the UNet model, illustrated in Figure 3.10, a common neural network in the field of biomedical image segmentation due to its ability to learn from a small number of training samples. Our model is inspired by [47] and adapted for the task of brain tumour segmentation. The model contains two main segments, a contracting path and an expansive path. These work to extract feature data whilst lowering the spatial information and then expanding the data back to its original size. The result is an image with the same size as the input and a depth of sixteen. We finally implement a binary classification layer to convert this data to a binary mask.

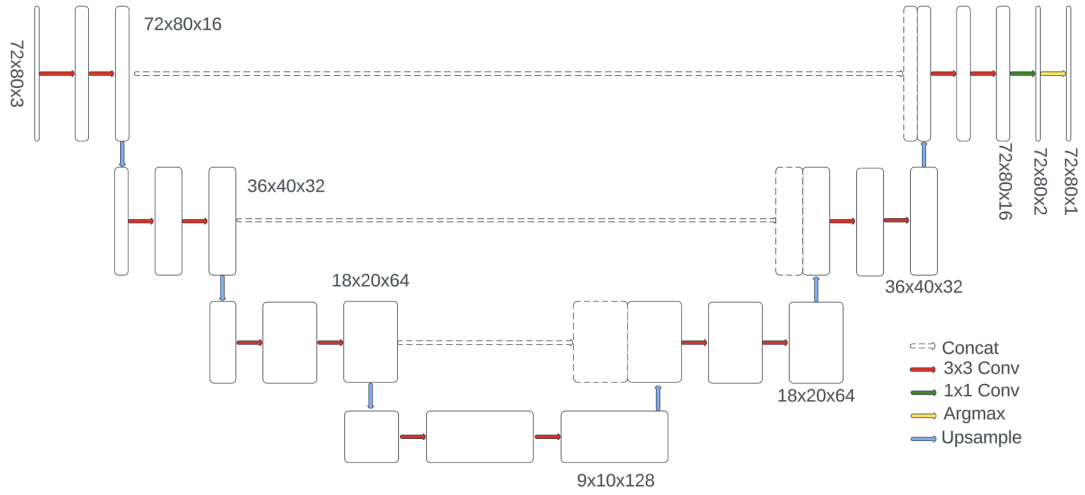

Figure 3.10: UNet model architecture

3.3.1 Contracting path

Following the preprocessing of the data, each slice is passed through the contracting path. This consists of a convolution layer into a max pooling layer repeated three times then a final convolution layer. Convolution layers are carried out by multiplying the input data by a 3x3 kernel containing nine weights and adding a bias value:

$$conv(x) = \sum_{i=1}^{9} w_i x_i + b \tag{3.1}$$

where w is the kernel weights and x is the pixel values.
These nine weights plus bias will all be modified during training to find the most optimal values. We then pass the result through a rectified linear unit (ReLU) that sets all negative values to zero.

$$ReLU(x) = \max(0, x) \tag{3.2}$$

Convolution with zero padding and a stride of one results in the image dimensions remaining unchanged but the depth of data in each pixel increasing. The benefit of using small 3x3 kernels is that it reduces the number of weights in our model. This works twofold to increase the rate of training and protect against overfitting. This becomes particularly important due to our limited supply of training data.

Max Pooling works by passing a 2x2 kernel over the image and outputting the maximum value thus reducing the dimensions by a factor of a half whilst retaining the most valuable data. At the end of the contracting path, we are left with data of size (9x10x128).

3.3.2 Expansive path

Each layer of the expansive path contains a deconvolution layer followed by a concatenation layer, this is done three times. The deconvolution layer consists of an upsample block to grow the dimensions by double. This is done using nearest-neighbour interpolation in which the value of the nearest pixel in the original image determines every target pixel's value. We then multiply by a 3x3 convolution kernel and use a ReLU activation function. Subsequently, we concatenate this result with the result of the matching convolution layer in the contracting path to preserve more general image data and increase the connection between both encoding and decoding sections. The significance of this is that the UNet reaches a bottleneck in between the contracting path and the expansive path when the data passed is of resolution (9x10x128). Whilst this data is of great depth, it does not cover a large area meaning many fine details are lost. The skip layers alleviate this by passing results directly from the previous section and combining them with the latest results. Concatenation is given by:

$$X^n_{concat} = [X^{N-n}_{skip}, X^n_{deconv}] \tag{3.3}$$

where N is the total number of layers and n is the current layer.
This concatenation then enters another convolution to reduce the depth further giving a final image of size (72x80x16), the same resolution as our input image.

3.3.3 Binary mask

To reduce our model's output to a binary mask for easy to view segmentation we harness a 1x1 convolution block to extract a final score for each class: background and tumour. We then use an argmax function to determine if the background score or the tumour score is higher, thus assigning a class to the respective pixel.

3.4 Training

3.4.1 Optimisation

After employing a four-to-one ratio of training to test data we are left with 295 patients to train each with seventy 2D slices leaving us with over 20,000 samples. To efficiently train our model with these samples we make use of a Stochastic Gradient Descent (SGD) approach with a mini-batch size of 200. This grants several benefits compared to batch gradient descent including a faster processing time and a lower memory usage whilst still providing a level of stability due to decreased noise in comparison to a batch size of one. More specifically we harness an extension of SGD known as Adam (Adaptive moment estimation) [27]. This algorithm is unique in its ability to assign adaptive learning rates to individual parameters based on their previous gradients allowing it to converge on an accurate set of weights in a shorter number of iterations. Each weight is given two values, m and v to represent the first and second moments respectively. The first moment (m) is computed by taking the average of past gradients and the second moment (v) is computed by taking the average of past gradients squared. These values are unique for each weight and bias and are used to modify the learning rate using the equation below:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \cdot \hat{m}_t \tag{3.4}$$

where:

- θ_{t+1} is the updated parameters at step $t+1$
- θ_t is the previous parameters at step t
- η is the learning rate
- $\hat{m}_t$ is the first moment
- $\hat{v}_t$ is the second moment
- ε is a very small value used to avoid division by zero

3.4.2 Loss equation

For SGD to evaluate the weights we require an effective loss function to assess the performance of the model at a given iteration. As we rely upon calculating the gradient

this must also be differentiable in a computationally efficient manner. A common loss function for binary classification tasks is cross-entropy loss due to its easy to compute gradient and resistance to class imbalances, in our case far more background pixels than tumour pixels. We equate this to the segmentation task by treating every pixel as a binary classification task and summing over all pixels.
The cross-entropy loss equation is given as:

$$\text{CrossEntropyLoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)] \tag{3.5}$$

where:

- N is the number of pixels
- y_i is the pixel's true value
- p_i is the pixel's predicted value

3.4.3 Training

With the loss function and optimiser in place, we can carry out training through an iterative process. We repeat twelve epochs in which the data is split into batches of 200 and each batch is passed through the current model and its results are compared to the true values using our loss function. We then use our optimiser to calculate the gradient and backpropagate to update the parameters accordingly. Once all batches of the dataset have been used we start the next epoch using the same dataset split into different batches. At set numbers of epochs, we perform validation to assess the performance of our model and ensure training is leading to the expected accuracy improvements.

3.5 Evaluation

We use our previously unused training data to compare our model's successes with existing work and past iterations. This data is passed into our trained model and a predicted mask is returned. By using Sklearn's confusion_matrix function [44] we can quickly generate the number of correctly and incorrectly classified pixels, this number can then be used to compute a number of evaluation metrics such as accuracy, precision and sensitivity. These are expanded upon in Chapter 5. We evaluate our model during

training after every three iterations to see if further training still results in performance enhancements. Finally, we evaluate the fully trained model to compare it with other state-of-the-art segmentation techniques.

3.6 Hyperparameters

After implementing the chosen UNet architecture we must now optimise the setup of the model to ensure the best possible segmentation results. Three main hyperparameters must be selected to ensure the training process converges at the ideal state. These are the number of epochs, learning rate and batch size. All these values must be carefully balanced to maximise the model's accuracy whilst still working within our environment's constraints. The number of epochs determines how many times the gradient decent loop will repeat. Each loop uses the entire dataset exactly once. By having a large number of epochs we can find better weights for the training data thus lowering the loss equation and increasing the model's accuracy. The two downsides of a large number of epochs are the risk of overfitting and increased training time. Better accuracy on the training set does not necessarily translate to better accuracy in general as the model may have learned specific features of the training samples and is unable to generalise to unseen images. Each epoch takes a set amount of time, therefore by increasing the number of epochs we also increase the training time linearly. We must carefully balance the number of epochs to improve the accuracy of our model whilst avoiding overfitting and allowing for a reasonable amount of time to complete training. Table 3.1 shows the most optimal value for the number of epochs is twelve. It is possible that by increasing the value we could reduce the loss further but this would increase the risk of overfitting and unfortunately we are limited by our environment. With more computational power we could run further experiments with more iterations.

Epochs	3	6	9	12
Loss	6.1456	2.6107	1.7916	1.4329

Table 3.1: change in loss when increasing number of epochs

The batch size determines how much the training set will be split up, for example with a training set of 100 samples and a batch size of 25 you will get four batches, each used to update the weights once in an epoch. Every epoch we split the data into new batches to add variance. Smaller batches introduce more noise into the weight updates,

this is key to reducing overfitting and increasing the model's generalisation capabilities. This extra noise can however reduce the stability of convergence, therefore it may take more iterations to reach an optimal point. Larger batch sizes allow the optimiser to make more accurate predictions as to the true gradient thus potentially speeding up training. As a larger batch size means fewer individual batches, it also means fewer iterations each epoch thus shortening the training time. Modern computers can also take advantage of parallelization to reduce the computation time with larger batches. Due to our environment and time constraints, we are limited to using the faster but sometimes less effective larger batches. Table 3.2 depicts the effects on the loss when changing the batch size. As we can see, our model favours a smaller batch size. As smaller batch sizes take longer to train we decided to use a batch size of 200. This means we can benefit from the superior loss whilst still ensuring a reasonable computation time.

Batch size	150	200	250	300
Loss	1.4253	1.4329	1.7916	1.9240

Table 3.2: change in loss when increasing batch size

The learning rate controls how much the weights should be updated at each step when computing backpropagation. A large learning rate causes the weights to update by more significant margins therefore we can reach a minimum in fewer steps. On the other hand, if the steps are too big you risk overshooting the minimum causing instability in the convergence process. You can increase stability and guarantee convergence without large swings by opting for a smaller learning rate. This will however require more iterations to reach an optimal point. Both batch size and learning rate go hand in hand. Larger batch sizes require smaller learning rates to ensure stability whilst smaller batch sizes require larger learning rates to ensure big enough steps are taken at every iteration. As we have chosen to take advantage of the ADAM optimiser, individual weights will have varying learning rates. This reduces the significance of our chosen learning rate although it is still important to pick a good starting value. Table 3.3 clearly shows our model favours a smaller learning rate. This is to be expected as we have already opted for a large batch size that complements the smaller learning rate.

learning rate	0.001	0.0015	0.002	0.0025
loss	1.4329	1.5274	1.5529	1.8024

Table 3.3: change in loss when increasing learning rate

Chapter 4

Implementation

The following chapter details the structure of the code used to train and evaluate the U-Net model using a UML diagram as well as features of the model's layers. We also state the environments used to execute preprocessing, training and evaluation.

4.1 UML design

To thoroughly analyse our proposed segmentation methods, we require several code files and classes, the most important of which have been detailed in a unified modelling language diagram (UML diagram). First, we see the preprocessing class in Figure 4.1. The functions held here generate labels for training including bounding boxes, k-means labels and threshold labels. Image processing such as splitting the 3D scan into slices and changing image dimensions also takes place here. Within Figure 4.1 we also see the visualisation functionality allowing us to display side-by-side images of the true labels and predicted results for each model. The UNet model is created and trained using the classes illustrated in Figure 4.2. Here we load in our preprocessed data and use Stochastic gradient descent to optimise the model whilst performing validation to find useful metrics such as accuracy and precision. As well as these five classes we also make use of a number of smaller Python scripts to execute simple functionality including calculating Hausdorff distance and rearranging data files.

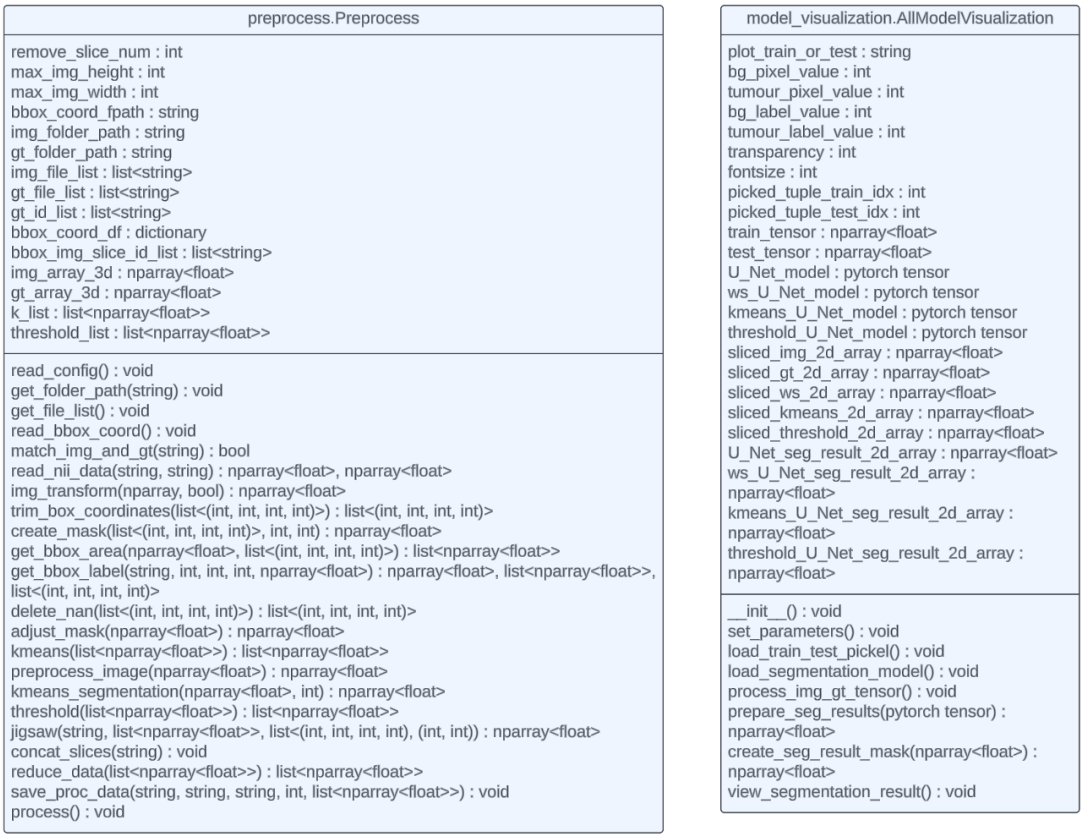

Figure 4.1: UML diagram for preprocessing and visualisation classes

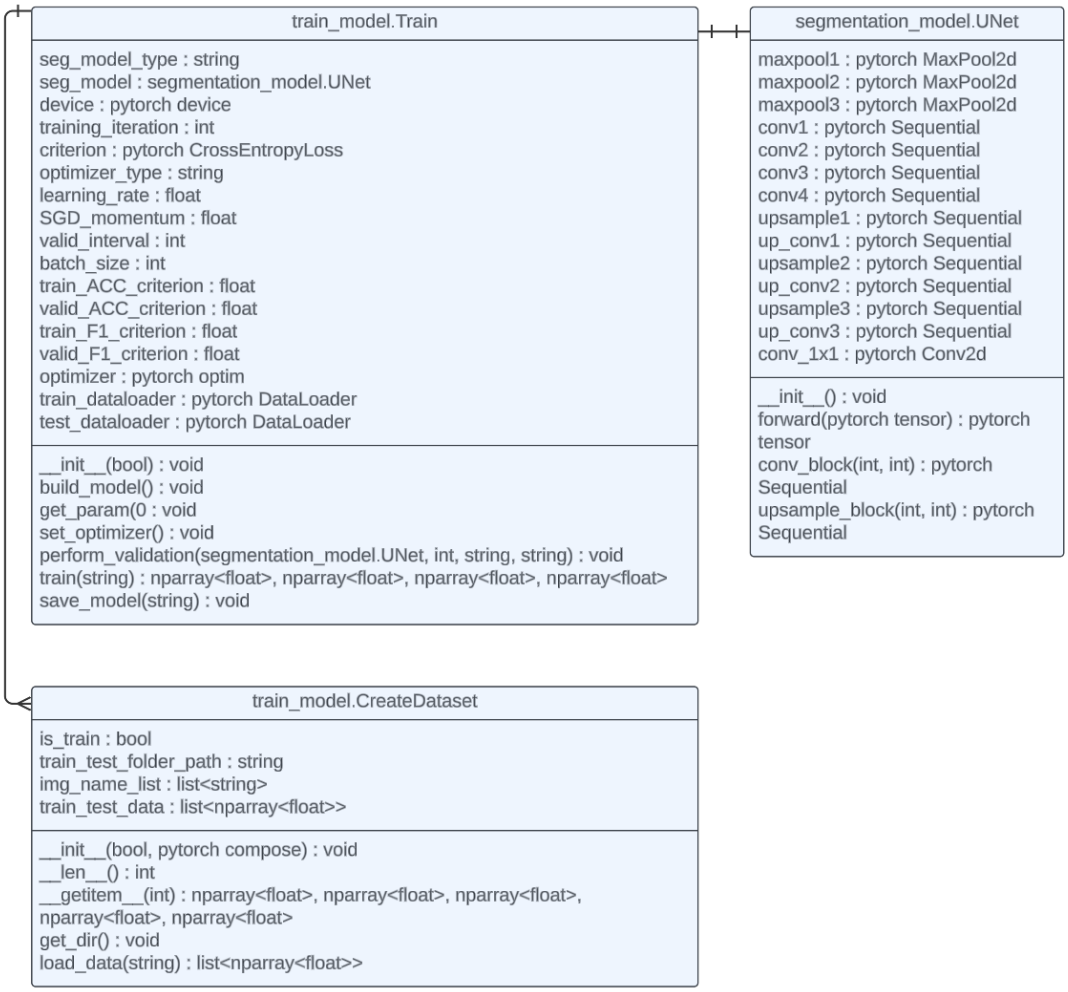

Figure 4.2: UML diagram for the UNet model and training classes

4.2 Model architecture

As previously discussed in Chapter 3 we implement a UNet model to carry out image segmentation. This comprises multiple layers as detailed in Table 4.1 and follows an encoding and decoding process. To do this convolution and max pooling layers are used to extract feature data whilst reducing the image size. We then use convolution and upsampling layers to restore the image to its original size whilst concatenating with outputs from the encoding steps to retain extra information. The output of our model is a binary mask indicating the predicted pixel locations of the brain tumours.

Layer	Type	Input	Output	Kernel	Stride	Pad
Preprocessing	preprocessing	*Height x width x 1*	72 x 80 x 3	-	-	-
Conv1	convolution	72 x 80 x 3	72 x 80 x 16	3 x 3	1	1
Maxpool1	max pooling	72 x 80 x 16	36 x 40 x 16	2 x 2	2	0
Conv2	convolution	36 x 40 x 16	36 x 40 x 32	3 x 3	1	1
Maxpool2	max pooling	36 x 40 x 32	18 x 20 x 32	2 x 2	2	0
Conv3	convolution	18 x 20 x 32	18 x 20 x 64	3 x 3	1	1
Maxpool3	max pooling	18 x 20 x 64	9 x 10 x 64	2 x 2	2	0
Conv4	convolution	9 x 10 x 64	9 x 10 x 128	3 x 3	1	1
Deconv1	upsampling + convolution	9 x 10 x 128	18 x 20 x 64	3 x 3	1	1
Concat1	concatenation + convolution	18 x 20 x 64 (Deconv1) + 18 x 20 x 64 (Conv3)	18 x 20 x 64	3 x 3	1	1
Deconv2	upsampling + convolution	18 x 20 x 64	36 x 40 x 32	3 x 3	1	1
Concat2	concatenation + convolution	36 x 40 x 32 (Deconv2) + 36 x 40 x 32 (Conv2)	36 x 40 x 32	3 x 3	1	1
Deconv3	upsampling + convolution	36 x 40 x 32	72 x 80 x 16	3 x 3	1	1
Concat3	concatenation + convolution	72 x 80 x 16 (Deconv3) + 72 x 80 x 16 (Conv1)	72 x 80 x 16	3 x 3	1	1
Conv1x1	convolution	72 x 80 x 16	72 x 80 x 2	1 x 1	1	0
Result	argmax	72 x 80 x 2	72 x 80 x 1	-	-	-

Table 4.1: Architecture of the UNet model

4.3 Environment

All preprocessing to transform 3D scans to Numpy files containing the slice image and training labels was carried out using Google Colab due to its impressive execution time [7]. The drawback of Colab is a 15GB memory limit when using the free version, as a result, the processed data had to be reduced in size including the removal of non-central slices and scaling by a third. When training the UNet models, an M1 Macbook was used to benefit from its 8-core CPU specifically optimised for training AI allowing for speeds of up to fifteen times that of previous intel chips when running machine learning tasks [32].

4.4 Language and Libraries

All code has been written using Python, a common language for machine learning due to its ease of use and vast number of libraries. The main libraries used were PyTorch [41], NumPy [20], and OpenCV [6]. PyTorch was used for implementing and training the convolutional neural network as it provides great flexibility for small-scale projects. NumPy provides many useful tools for handling multi-dimensional arrays such as our images and masks. OpenCV is a well-equipped library for image processing, allowing us to carry out important image processing. The main use is for generating weakly supervised training labels including bounding boxes and K-means clustering. Many other libraries were used for smaller tasks such as Matplotlib [56] for visualisations, SkLearn [44] for evaluation and SciPy [58] for calculating Hausdorff distance.

Chapter 5

Evaluation

In this chapter, we will discuss the performance of our model using each of the four training labels: fully supervised, bounding box, k-means and threshold. We will first describe the metrics used to evaluate the models and then give the results including side-by-side images of selected slices.

The four models we will compare are:

- FS: Fully supervised model trained with the ground truths.
- BB: Bounding box model trained with our computer generated bounding boxes.
- KM: K-means model trained with the masks generated after performing K-means clustering on our bounding boxes.
- THR: Thresholding model trained with the masks generated after performing thresholding on our bounding boxes.

5.1 Evaluation metrics

To help with comparing image segmentation models a set of evaluation metrics is needed, each to compute unique assessments of our model's success to determine the strengths and weaknesses. The most common are Accuracy (ACC), Error rate (ERR), Precision (PR), Sensitivity (SE), Specificity (SP), F1 score (F1), Intersection over Union (IoU) and Hausdorff distance (HD). To easily calculate these we must first generate a confusion matrix by finding the true positive (TP), true negative (TN), false

positive (FP) and false negative (FN). These are found by comparing every pixel in the segmented image to its corresponding pixel in the true mask.

- Pixels that are correctly labelled positive are recorded as true positive.
- Pixels that are incorrectly labelled positive are recorded as false positive.
- Pixels that are correctly labelled negative are recorded as true negatives.
- Pixels that are incorrectly labelled negative are recorded as false negative.

Performance metrics can be calculated using the confusion matrix as follows:

- Accuracy is the percentage of correctly assigned pixels.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

- Error rate is the percentage of incorrectly assigned pixels.

$$ERR = \frac{FP + FN}{TP + TN + FP + FN} = 1 - ACC \tag{5.2}$$

- Precision is the percentage of pixels labelled positive that are positive.

$$PR = \frac{TP}{TP + FP} \tag{5.3}$$

- Sensitivity is the percentage of actual positive pixels that were labelled correctly, it is sometimes referred to as recall.

$$SE = \frac{TP}{TP + FN} \tag{5.4}$$

- Specificity is the percentage of actual negative pixels labelled correctly.

$$SE = \frac{TN}{FP + TN} \tag{5.5}$$

- F1 score is the harmonic mean of sensitivity and specificity.

$$F1 = 2 \cdot \frac{PR \cdot SE}{PR + SE} \tag{5.6}$$

- Intersection over Union is the intersection of the predicted mask and the true mask divided by their union. It is the most common tool for comparing segmentation models.

$$IoU = \frac{\text{Intersection Area}}{\text{Union Area}} = \frac{TP}{TP+FP+FN} \tag{5.7}$$

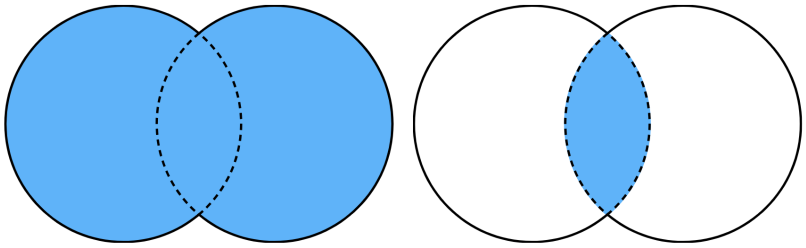

Figure 5.1: Left: Union area, Right: Intersection area

- Hausdorff distance is the largest of all the distances from a point on one set to the nearest point on another set. The smaller the value the more similar the two sets, in this case, the predicted mask and the true label. We use the equation:

$$H(A,B) = \max\left(\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right) \tag{5.8}$$

where:

  - A and B are our ground truth and prediction sets
  - $d()$ is the distance between two points
  - sup is the supremum
  - inf is the infimum

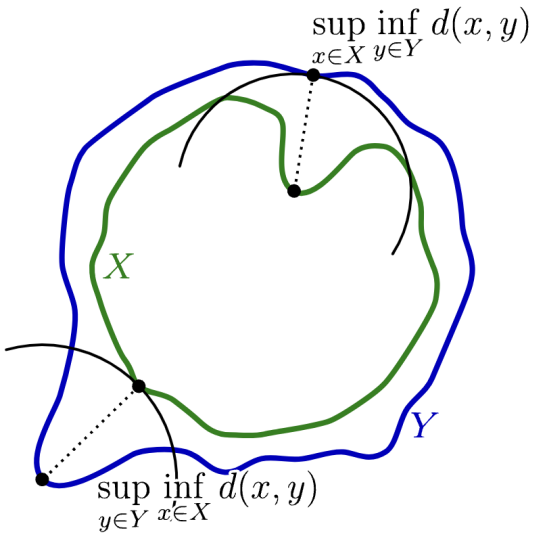

Figure 5.2: Hausdorff distance [59]

5.2 Comparison

5.2.1 Model results

After comparing the real ground truths to our predicted labels as seen in Table 5.1 we discover the benefits of the K-means approach when contrasted to other weakly supervised training methods, particularly the slight improvements over bounding boxes in almost all metrics, particularly precision. Fully supervised unsurprisingly remains the most effective method meanwhile thresholding can be seen to be far less functional especially when noting the intersection over union. Table 5.2 gives insight into the neural network's success by comparing the labels used for training with the predicted results. Overall the UNet model is shown to be an effective tool for medical image segmentation with all label types able to effectively train the model to a high standard.

	Model	ACC	ERR	PR	SE	F1	IoU
Train	FS	0.9954	0.0046	0.9575	0.9449	0.9510	0.9104
	BB	0.9820	0.0180	0.7879	0.9814	0.8583	0.7757
	KM	0.9870	0.0130	0.8358	0.9451	0.8822	0.8071
	THR	0.9763	0.0237	0.7670	0.5592	0.5924	0.5429
Test	FS	0.9954	0.0046	0.9568	0.9452	0.9508	0.9100
	BB	0.9820	0.0180	0.7878	0.9814	0.8583	0.7756
	KM	0.9870	0.0130	0.8358	0.9452	0.8823	0.8072
	THR	0.9763	0.0237	0.7664	0.5593	0.5924	0.5430

Table 5.1: Results of the UNet model with different training labels with respect to the ground truth

	Model	ACC	ERR	PR	SE	F1	IoU
Train	FS	0.9954	0.0046	0.9575	0.9449	0.9510	0.9104
	BB	0.9952	0.0048	0.9685	0.9706	0.9695	0.9422
	KM	0.9927	0.0073	0.9318	0.9487	0.9399	0.8919
	THR	0.9980	0.0020	0.9295	0.8801	0.9025	0.8372
Test	FS	0.9954	0.0046	0.9568	0.9452	0.9508	0.9100
	BB	0.9952	0.0048	0.9685	0.9706	0.9695	0.9422
	KM	0.9927	0.0073	0.9321	0.9487	0.9401	0.8923
	THR	0.9980	0.0020	0.9309	0.8808	0.9035	0.8384

Table 5.2: Results of UNet model with different training labels with respect to models training label

5.2.2 Errors

The data in Table 5.3 reports the number of times each model incorrectly classified the slice, either detecting a tumour that was not present or detecting none when at least one was present. It is important to note the training set is four times larger than the testing set resulting in a higher number of errors. When taking the data partitioning into account, all models had higher error rates for test data compared to training data which is to be expected of unfamiliar images. Both fully supervised and k-means methods exhibit similar results whilst the bounding box model has a similar test error rate but a much lower training error rate. This is perhaps due to the bounding box labels being the largest therefore more pixels are likely to be classified as unhealthy reducing the likelihood of predicting no tumours when one is present. Conversely, thresholding can be seen to have a far greater number of errors, with the majority of

samples being incorrectly classified. This happens because the thresholding algorithm requires constant values to compare each pixel value to. As different scans and sections of scans can have diverse values it is not always the case that the tumour's pixel values will fall between these two numbers resulting in the full mask being set to background. The model will then be taught to treat all cells as negative matches and often predict no presence of tumours despite their existence.

	FS	BB	KM	THR
Train errors	1782	1156	1641	13364
Test errors	822	781	772	3621

Table 5.3: Misclassifications of each model for each dataset

5.2.3 Hausdorff distance

Hausdorff distance can inform us of the similarity of two sets by looking at the most extreme differences. Table 5.4 compares the average Hausdorff distance of our four models in the events of a correct tumour classification. With respect to k-means we see slight improvements in both the labels and the prediction's similarity to the ground truth when compared to bounding boxes, even reaching approximately half a unit from the fully supervised models test results. Thresholding continues to perform sub-optimally with far higher distances than all other models due to static thresholding values setting entire masks to background.
This analysis of error rates and Hausdorff distance emphasises the benefit of k-mean's ability to improve upon the traditional bounding box method and closer align segmentation results with fully supervised methods.

		FS	BB	KM	THR
Train	Label	0.000	5.309	4.753	14.826
	Result	4.024	5.830	5.162	8.671
Test	Label	0.000	5.125	4.343	17.633
	Result	6.383	7.152	6.889	12.232

Table 5.4: Hausdorff distances of models results and labels

5.2.4 Training loss

Figure 5.3 illustrates the different training losses of the U-Net model when applying each type of training label over twelve iterations. We can first see the curves follow a rapid descent in the first four epochs followed by a more stable fall until terminating at twelve epochs. Thresholding can be seen to have the lowest loss values attributing to the high volume of training masks incorrectly set entirely to background making segmentation easier for the U-Net. Following this we see both fully supervised and bounding boxes showcase similar losses to each other, finally followed by K-means. This examination suggests running the model for more iterations using more powerful hardware will likely result in further reductions in loss potentially improving the segmentation model's effectiveness.

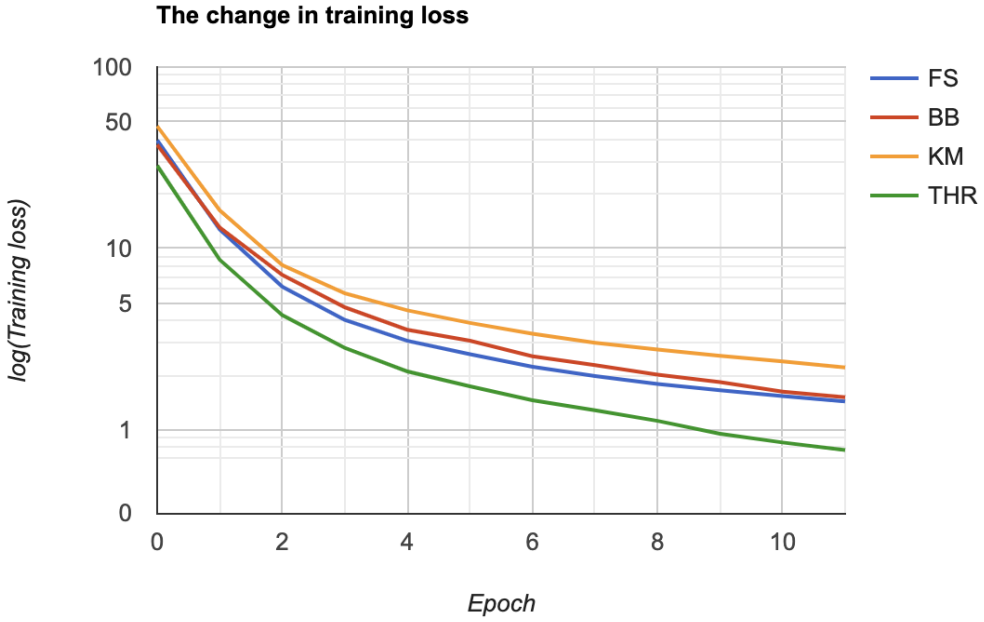

Figure 5.3: Graph of training loss at each epoch

5.2.5 Visualisation

The following images help to visualise the effectiveness of the weakly supervised methods in comparison to a fully supervised model and demonstrate what a user of the model will see. Figures 5.4, 5.5 and 5.6 depict three selected MRI slices, the four training labels used, and the respective segmentation outputs. Note the similarity between

K-means labels and results when compared to the fully supervised prediction, particularly how it improves upon the bounding box masks. Alternatively, thresholding can be seen to regularly create a full background mask leading to its subpar performance as discussed previously.

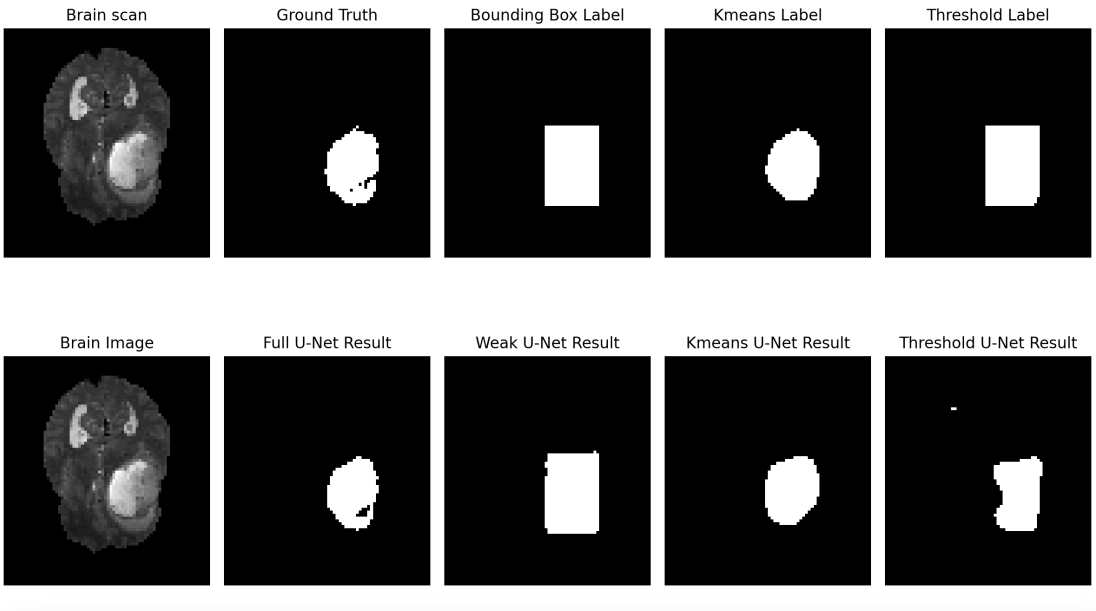

Figure 5.4: Visualisation of segmentation results for selected slice

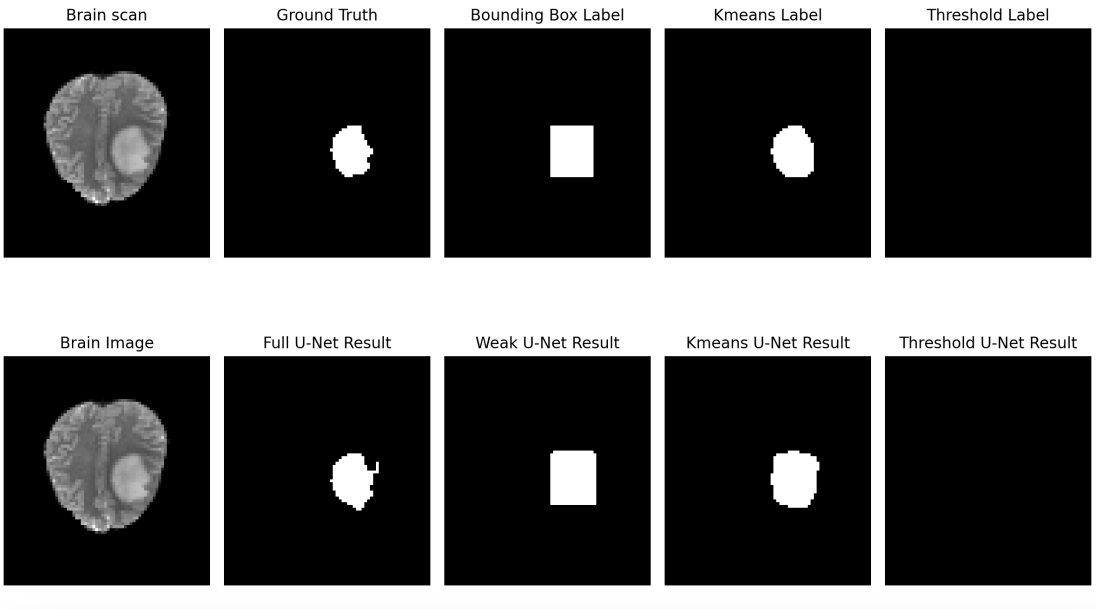

Figure 5.5: Visualisation of segmentation results for selected slice

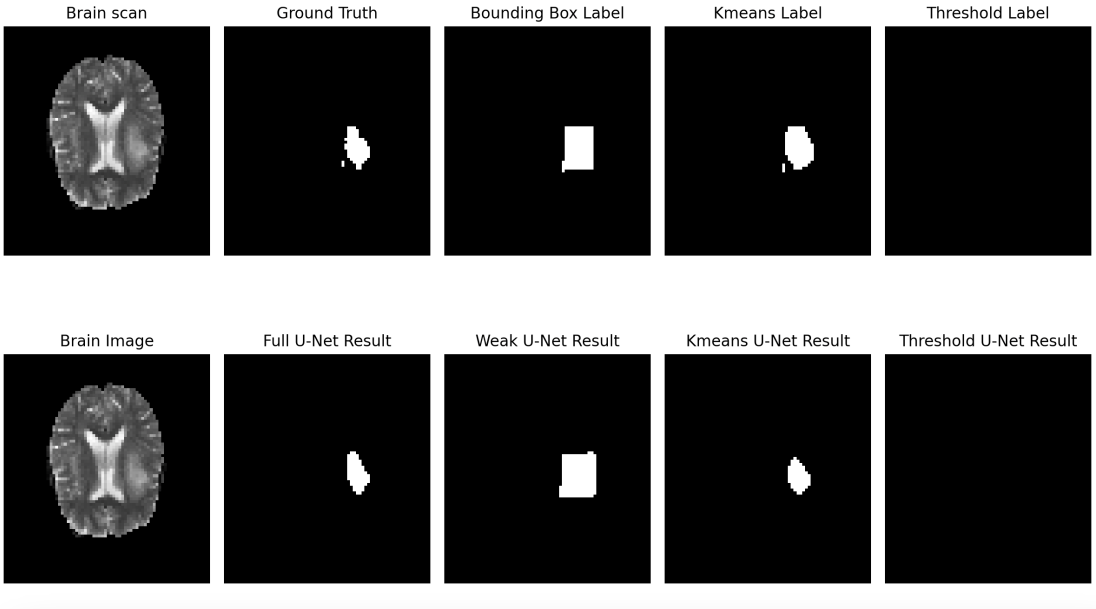

Figure 5.6: Visualisation of segmentation results for selected slice

Chapter 6

Discussion

As per our objectives set out in Chapter One, we have successfully researched existing work in medical image segmentation as well as laying out detailed descriptions of common machine learning methods. We then acquired the MICCAI BraTS 2020 dataset and preprocessed the images to generate multiple types of weakly supervised labels including bounding boxes, K-means generated labels and thresholded labels. Following this we implemented a convolutional neural network based on the UNet architecture capable of performing image segmentation when trained with a variety of data to test our new labels. With the UNet model, we then performed training and detailed analysis to asses our four training labels and evaluate the effectiveness of each approach.

Past work in the field of image segmentation has all pointed to fully supervised models being the front runners with weakly supervised methods trailing behind. By closing this gap in accuracy, we hope to provide a more cost and time effective alternative to fully supervised methods. Our proposed k-means label generation has demonstrated noticeable improvements over the typical weakly supervised method of bounding boxes without the requirement of further training. Whilst pixel-level labels still result in the best predictions, our K-means model can provide an alternative for tasks where detailed training data is not available. Our proposed thresholding method has exhibited sub-par results compared to all other labels. This is likely due to the inherent floor in the approach which is its non-dynamic nature. Whilst both our K-means and thresholding methods work by splitting all pixels inside of a bounding box into three categories by their intensity, K-means does this based on the given values allowing it to adapt to each segment of the brain. Thresholding on the other hand uses

pre-determined values that are unlikely to be relevant to every scenario. Despite the improvements we have demonstrated, there is still room to further push the success of the model. Below we suggest multiple areas that future work could build upon.

6.1 Future work

6.1.1 Memory limitations

Much of the training data collected was not used or reduced in resolution to account for the limitations of our environment. By implementing our model with more memory and advanced computational power, future work could benefit from four scans per patient, over double as many slices per scan and an increased resolution resulting in nine times the pixels per slice. All this excess data will allow the model to gain more insightful knowledge of brain images and tumours thus resulting in a closer accuracy to fully supervised models.

6.1.2 Increased training data

Initially, we explored growing the quantity of training data using image modification techniques such as reflecting slices. Ideally, this would produce extra images that are still accurate brain scans without the need to collect or preprocess more data and provide the added benefit of reducing overfitting by increasing the model's generalisation ability. Unfortunately, this did not have a positive effect on the model as when given the context of medical images, particularly of the brain, making modifications can cause drastic changes. Notably the left and right sides of the brain are not perfectly symmetrical. Further research can be done into image modification techniques that can increase the quantity of data and thus the model's generalisation capabilities.

6.1.3 Thresholding

As our results show, thresholding has shown lacklustre performance metrics. This is largely due to the naive approach of using fixed thresholding values for all slices. Experimentation with these values could lead to strong improvements in the training masks generated even rivalling our superior k-means training labels. One suggested approach would be to use an automated thresholding method to pick these values. Otsu's method [2] is an effective algorithm for thresholding when there are two distinct

classes, a foreground and a background. By removing static values and adapting the model to each slice we can hopefully raise the performance above standard bounding boxes.

6.1.4 K-means clustering

We now analyse the actual k-means algorithm to look for possible advancements. A large weakness in the algorithm is the sensitivity to cluster start points potentially leading to vastly different final clusters [13]. These clusters can result in varying results therefore picking the right starting centers requires thorough investigation and experimentation. The number of clusters we split into also has drastic effects on the output thus requiring further experimentation. We currently opt for three clusters and decrease this for images below a certain size, larger numbers of clusters could perhaps exhibit better masks.

6.1.5 Further application

Our model gains inspiration from a neural network for stroke detection used to find lesions in MRI scans [47]. As we have successfully applied the algorithms to brain tumours it follows that the model is suitable for multiple types of medical image segmentation opening the possibility of researching its application in other areas of interest. The detection of tumours in other parts of the body could provide life-saving information to doctors without the need for costly strongly supervised data.

Chapter 7

Conclusion

In summary, our research aims to find effective weakly supervised methods to improve upon the industry standard bounding boxes and even rival fully supervised models. Our approach involves modifying bounding box labels to more accurately encompass the true tumour location. We do this using two methods, k-means clustering and thresholding and have discovered that k-means poses a reliable alternative to common weakly supervised models. All labels have been tested through the training and evaluation of a UNet model specifically designed for medical image segmentation. Our proposed k-means approach is shown to noticeably improve on bounding box labels when using the same dataset and close the gap between the results of weakly supervised and fully supervised models.

Bibliography

[1] J. Ahn and S. Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4981–4990, 2018.

[2] S. L. Bangare, A. Dubal, P. S. Bangare, and S. Patil. Reviewing otsu's method for image thresholding. *International Journal of Applied Engineering Research*, 10(9):21777–21783, 2015.

[3] S. Basu, T. C. Kwee, S. Surti, E. A. Akin, D. Yoo, and A. Alavi. Fundamentals of pet and pet/ct imaging. *Annals of the New York Academy of Sciences*, 1228(1):1–18, 2011.

[4] T. Benslama and R. Jallouli. Clustering of social media data and marketing decisions. In *Digital Economy. Emerging Technologies and Business Innovation: 5th International Conference on Digital Economy, ICDEc 2020, Bucharest, Romania, June 11–13, 2020, Proceedings 5*, pages 53–65. Springer, 2020.

[5] L. Bottou. Stochastic gradient descent tricks. pages 421–436, 2012.

[6] S. Brahmbhatt. *Practical OpenCV*. Apress, 2013.

[7] T. Carneiro, R. V. M. Da Nóbrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. Reboucas Filho. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *Ieee Access*, 6:61677–61685, 2018.

[8] V. Collins. Brain tumours: classification and genes. *Journal of Neurology, Neurosurgery & Psychiatry*, 75(suppl 2):ii2–ii11, 2004.

[9] N. Cui. Applying gradient descent in convolutional neural networks. In *Journal of Physics: Conference Series*, volume 1004, page 012027. IOP Publishing, 2018.

[10] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

[11] R. Davis and J. J. King. The origin of rule-based systems in ai. *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, 1984.

[12] R. Dhiman, G. Joshi, and C. R. Krishna. A deep learning approach for indian sign language gestures classification with different backgrounds. In *Journal of Physics: Conference Series*, volume 1950, page 012020. IOP Publishing, 2021.

[13] S. Dolnicar. Using cluster analysis for market segmentation-typical misconceptions, established methodological weaknesses and some recommendations for improvement. 2003.

[14] R. Dubrow and A. S. Darefsky. Demographic variation in incidence of adult glioma by subtype, united states, 1992-2007. *BMC cancer*, 11(1):1–10, 2011.

[15] M. R. Gilbert, R. Ruda, and R. Soffietti. Ependymomas in adults. *Current neurology and neuroscience reports*, 10:240–247, 2010.

[16] C. Godfraind. Classification and controversies in pathology of ependymomas. *Child's nervous system*, 25:1185–1193, 2009.

[17] L. W. Goldman. Principles of ct and ct technology. *Journal of nuclear medicine technology*, 35(3):115–128, 2007.

[18] N. Gordillo, E. Montseny, and P. Sobrevilla. State of the art survey on mri brain tumor segmentation. *Magnetic resonance imaging*, 31 8:1426–38, 2013.

[19] C. Hardwidge and S. Hettige. Tumours of the central nervous system. *Surgery (oxford)*, 30:155–161, 2012.

[20] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[21] M. Hashemi. Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. *Journal of Big Data*, 6(1):1–13, 2019.

[22] R. H. Hashemi, W. G. Bradley, and C. J. Lisanti. *MRI: the basics: The Basics*. Lippincott Williams & Wilkins, 2012.

[23] M. Havaei, A. Davy, D. Warde-Farley, A. Biard, A. C. Courville, Y. Bengio, C. Pal, P.-M. Jodoin, and H. Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, 2015.

[24] A. Hoopes, J. S. Mora, A. V. Dalca, B. Fischl, and M. Hoffmann. Synthstrip: skull-stripping for any brain image. *NeuroImage*, 260:119474, 2022.

[25] H. Jabbar and R. Z. Khan. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). *Computer Science, Communication and Instrumentation Devices*, 70(10.3850):978–981, 2015.

[26] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 876–885, 2017.

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] P. Kleihues, P. C. Burger, and B. W. Scheithauer. The new who classification of brain tumours. *Brain pathology*, 3(3):255–268, 1993.

[29] K. K. Koeller and E. J. Rushing. Oligodendroglioma and its variants: Radiologic-pathologic correlation. *Radiographics*, 25(6):1669–1688, 2005.

[30] J. Konecný, J. Liu, P. Richtárik, and M. Takác. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10:242–255, 2015.

[31] S. Larjavaara, R. Mantyla, T. Salminen, H. Haapasalo, J. Raitanen, J. Jaaskelainen, and A. Auvinen. Incidence of gliomas by anatomic location. *Neuro-oncology*, 9(3):319–325, 2007.

[32] X. Liao, B. Li, and J. Li. Impacts of apple's m1 soc on the technology industry. In *2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*, pages 355–360. Atlantis Press, 2022.

[33] A. Mahmood, N. Qureshi, and G. Malik. Intracranial meningiomas: analysis of recurrence after surgical treatment. *Acta neurochirurgica*, 126:53–58, 1994.

[34] P. McKinney. Brain tumours: incidence, survival, and aetiology. *Journal of Neurology, Neurosurgery & Psychiatry*, 75(suppl 2):ii12–ii17, 2004.

[35] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014.

[36] F. Mignacco and P. Urbani. The effective noise of stochastic gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2022(8):083405, 2022.

[37] P. Mlynarski, H. Delingette, A. Criminisi, and N. Ayache. Deep learning with mixed supervision for brain tumor segmentation. *Journal of Medical Imaging*, 6:034002 – 034002, 2018.

[38] A. Mordvintsev and K. Abid. Opencv-python tutorials documentation. *Obtenido de https://media. readthedocs. org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals. pdf*, 2014.

[39] Q. Ostrom, H. Gittleman, P. Liao, C. D. Rouse, Y. Chen, J. Dowling, Y. Wolinsky, C. Kruchko, and J. Barnholtz-Sloan. Cbtrus statistical report: primary brain and central nervous system tumors diagnosed in the united states in 2007-2011. *Neuro-oncology*, 16 Suppl 4:iv1–63, 2012.

[40] T. K. Owonikoko, J. Arbiser, A. Zelnak, H.-K. G. Shu, H. Shim, A. M. Robin, S. N. Kalkanis, T. G. Whitsett, B. Salhia, N. L. Tran, et al. Current approaches to the treatment of metastatic brain tumours. *Nature reviews Clinical oncology*, 11(4):203–222, 2014.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[42] D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1796–1804, 2015.

[43] D. Pauleit, G. Stoffels, A. Bachofner, F. W. Floeth, M. Sabel, H. Herzog, L. Tellmann, P. Jansen, G. Reifenberger, K. Hamacher, et al. Comparison of 18f-fet and 18f-fdg pet in brain tumors. *Nuclear medicine and biology*, 36(7):779–787, 2009.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[45] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1713–1721, 2015.

[46] B. Rachet, E. Mitry, M. Quinn, N. Cooper, and M. Coleman. Survival from brain tumours in england and wales up to 2001. *British journal of cancer*, 99(1):S98–S101, 2008.

[47] ruiqing. K-means-based pixel-wise label generation from bounding box annotations for ischemic stroke image segmentation. 2023.

[48] F. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, S. Gould, and J. M. Alvarez. Built-in foreground/background prior for weakly-supervised semantic segmentation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 413–432. Springer, 2016.

[49] S. Sathornsumetee, J. N. Rich, and D. A. Reardon. Diagnosis and treatment of high-grade astrocytoma. *Neurologic clinics*, 25(4):1111–1139, 2007.

[50] J. A. Schwartzbaum, J. L. Fisher, K. D. Aldape, and M. Wrensch. Epidemiology and molecular pathology of glioma. *Nature clinical practice Neurology*, 2(9):494–503, 2006.

[51] Y. Shen, L. Cao, Z. Chen, B. Zhang, C. Su, Y. Wu, F. Huang, and R. Ji. Parallel detection-and-segmentation learning for weakly supervised instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8198–8208, 2021.

[52] C. Sherr. Cancer cell cycles. *Science*, 274:1672 – 1677, 1996.

[53] M.-H. Sheu, S. S. Morsalin, S.-H. Wang, L.-K. Wei, S.-C. Hsia, and C.-Y. Chang. Fhi-unet: faster heterogeneous images semantic segmentation design and edge ai implementation for visible and thermal images processing. *IEEE Access*, 10:18596–18607, 2022.

[54] R. Stupp, M. Gander, S. Leyvraz, and E. Newlands. Current and future developments in the use of temozolomide for the treatment of brain tumours. *The lancet oncology*, 2(9):552–560, 2001.

[55] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[56] S. Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[57] D. C. Ugwuanyi, T. F. Sibeudu, C. P. Irole, M. P. Ogolodom, C. T. Nwagbara, A. M. Ibekwe, and A. N. Mbaba. Evaluation of common findings in brain computerized tomography (ct) scan: A single center study. *AIMS neuroscience*, 7(3):311, 2020.

[58] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[59] Wikipedia. Hausdorff distance, 2024. Accessed on March 12, 2024.

[60] S. H. Wilne, R. C. Ferris, A. Nathwani, and C. R. Kennedy. The presenting features of brain tumours: a review of 200 cases. *Archives of disease in childhood*, 91(6):502–506, 2006.

[61] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[62] H. Yu, Y. Zhou, H. Qian, M. Xian, and S. Wang. Loosecut: Interactive image segmentation with loosely bounded boxes. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3335–3339. IEEE, 2017.