Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Deep Tissue Characterisation for Diagnosis Support in Neurosurgery

Author:
Stamatios Kourkoutas

Supervisor:
Dr. Stamatia Giannarou

Submitted in partial fulfillment of the requirements for the MSc degree in Computing (Software Engineering) of Imperial College London

September 2020

Abstract

Accurate diagnosis of brain tumors during a resection operation has been proven to be a challenging task, even for trained surgeons. However, such diagnoses can improve the success rate of brain tumor resection operations. In this project, we propose a computer-aided diagnosis (CAD) system, which is capable of segmenting brain tumors in 2D ultrasound images. The architectural choice to make our framework work with 2D ultrasound images means that it can be used to obtain preoperative and intraoperative diagnoses of brain tumors in almost real-time. Our proposed CAD system consists of a deep Convolutional Neural Network (CNN), the Cascaded Partial Decoder with the ResNet backbone (CPD-R), and a post-processing Conditional Random Field (CRF) method. The whole system exhibited state-of-the-art performance in a custom dataset of ultrasound images of brain tumors. Finally, an effort is being made to interpret the learned features, regarding the tumor class, of the CPD-R architecture and to understand which features of an ultrasound input image the model considers the most important when producing its segmentation maps.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Stamatia Giannarou for her continuous support and supervision during the whole duration of this project. Dr. Giannarou's deep knowledge of the field and her keen insight when suggesting various research directions worth investigating is what made possible the completion of this project.

Furthermore, I would like to thank our clinical collaborators from the department of Neurosurgery in Charing Cross Hospital, Imperial College London, UK, for providing us with high quality US images of cases of brain tumors and their annotations. This whole project would not be possible otherwise since there aren't any publicly available datasets of US brain tumor images, such as this one, which we could use instead.

Finally, I would like to express my gratitude to my family for their emotional support throughout the whole MSc course at Imperial College London.

Contents

Chapter 1

Introduction

1.1 Clinical Motivation

The number of people that die due to brain tumors worldwide is quite large and so there have been many endeavors to try and tackle this disease. An in-time and accurate diagnosis of malignant brain tumors in the early stages can help in decreasing the mortality rate of the disease by performing a resection surgery. However, there are different kinds of tumors, with some being more difficult to detect and ultimately remove surgically than others due to their similarity with their surroundings. So there is need for trained specialists who are able to recognise and detect brain tumors accurately.

Moreover, even if the tumor region was detected accurately before a brain tumor surgery, the brain is substantially deformed according to (1) during a resection operation and so the surgeon requires to detect the tumor regions again during the surgery, which is quite hard, even for trained specialists. More accurate detection of tumorous regions means more accurate removal of the entire tumor and thus a bigger survival rate for the patient according to (2). Fast and accurate diagnosis of tumor regions before and during a resection surgery can thus improve the success rate of brain tumor operations.

To aid surgeons in their endeavors to accurately diagnose and remove brain tumors, significant research is being conducted in utilising computer vision and machine learning methods to build frameworks that can help with preoperative and intraoperative diagnosis. Such frameworks that facilitate computer aided diagnosis (CAD) are called CAD systems. The goal of this project is to propose a CAD system that can automatically identify tumor regions in an image modality.

1.2 Image Modalities

Some commonly used image modalities to visualise the brain are magnetic resonance imaging (MRI) and ultrasound (US) as seen in (3) and (4). Most of the time, MRI is preferred from other image modalities when studying brain structure, because MRI images are clear, with high resolution, and can be collected with no known medical harm to patients.

However, in our project, we chose to utilise US images for our CAD system for the following reason. US images can be collected from patients easily almost in real-time, which enables our CAD system to work almost in real-time and can thus be applied for intraoperative brain tissue characterisation. MRI images in contrast require more time to be collected and are also hard to acquire during brain tumor resection operations, because this would require to move the patient.

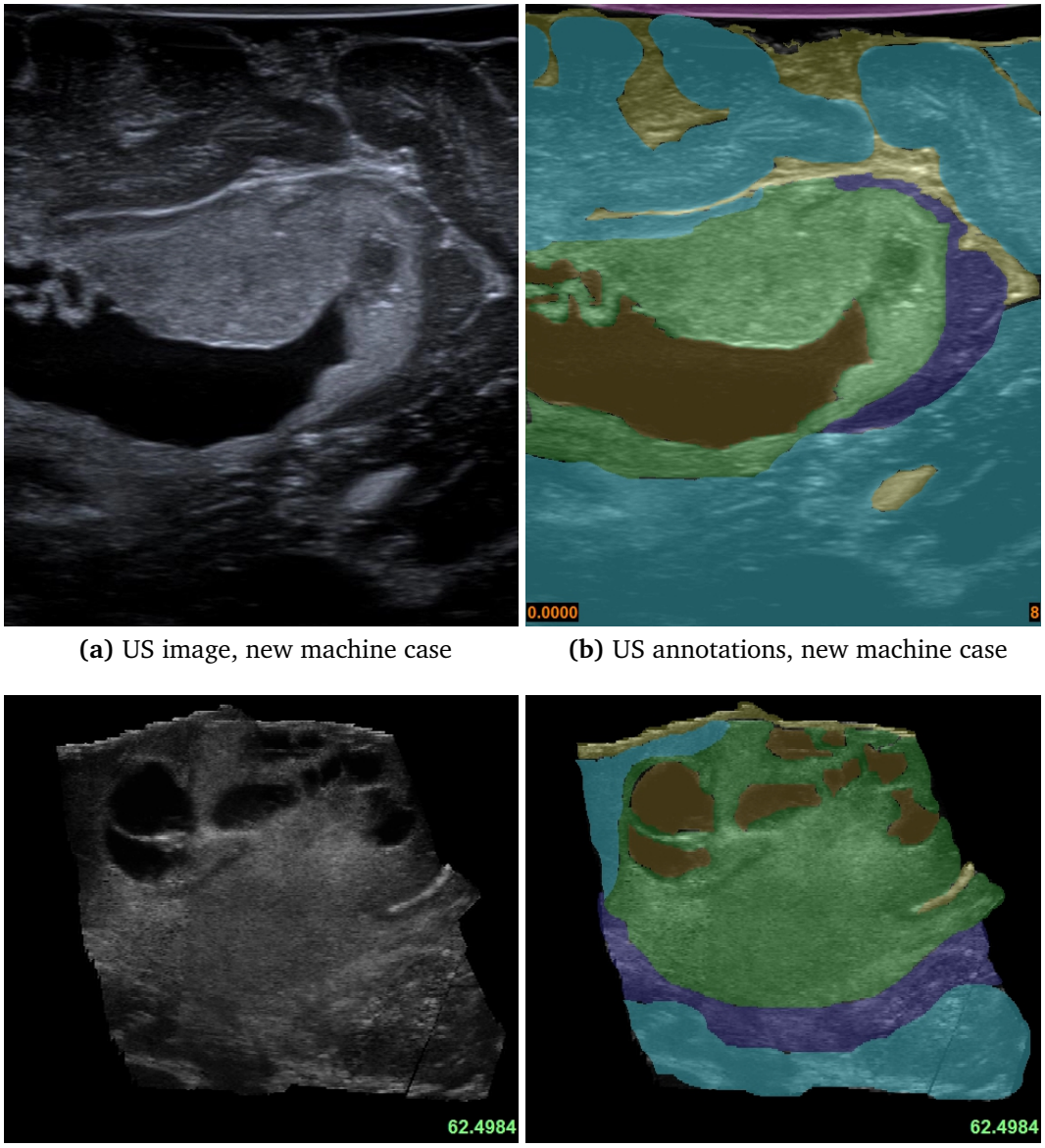
Our choice to work with US images instead of MRI introduces some challenges for the CAD system we propose. First of all, US images have lower resolution than MRI and thus constitute a harder to interpret signal. Moreover, there are no publicly available large datasets with annotated US images of brain tumors, as opposed to publicly available datasets with MRI brain tumor data, such as the Brain Tumor Segmentation (BraTS) dataset (5). This scarcity of annotated ultrasound data severely hinders our experiments.

1.3 Available Dataset

Because there are no publicly available large datasets with annotated 2D US images of brain tumors, as evidenced from the absence of such a dataset in the survey presented in (3), we used a custom dataset. The dataset is comprised of 2D US images of brain tumors and was collected and annotated by a clinical team from the department of Neurosurgery in Charing Cross Hospital, Imperial College London, UK.

The images of the dataset were extracted from US video clips from different patients, with each video lasting around 2 minutes. The videos were collected using two different machines, an older one, which was a Toshiba Aplio i800 (Canon Medical System Ltd, Japan), and a newer one, which was a Toshiba Aplio i900 (Canon Medical System Ltd, Japan). From each video, an image every 18 to 20 frames was extracted, so 8 to 12 images were collected in total for every video. This resulted in a dataset with 9 different cases, 6 cases from the new machine and 3 cases from the old machine with a total of 91 2D US images.

The annotations are of good quality containing much more information than the existence of a tumor or not. However, our main focus was on building a CAD system, which can recognise and segment tumorous from non-tumorous regions. In figure 1.1 we present two US images, representative of those in our dataset, alongside their annotated masks.



(a) US image, new machine case **(b)** US annotations, new machine case

(c) US image, old machine case **(d)** US annotations, old machine case

Figure 1.1: Two US images of two different patients, taken from our dataset alongside their annotated masks. Image 1.1a is from a case from the new machine while image 1.1c is from a case from the old machine.

The colours of the annotations in our dataset, which we can see in figure 1.1, are explained here.

- Green is used for the tumor region.
- Light blue is used for the normal brain tissue.
- Dark blue is used for oedema around the brain tumor.
- Light yellow is used for the arachnoid and dura matter of the brain.
- Dark yellow is used for necrosis.
- Pink is used for some artefacts in the ultrasound images.
- Finally, light pink is used in some cases to highlight more intense tumor areas inside the tumor.

As we mentioned before, US images are quite hard to interpret, as opposed to MRI images, and the differences between different machines used for the US image collection add an extra layer of difficulty to the problem. For example, as we can see from figure 1.1, images from the new machine differ slightly from those from the old machine in color intensity and contrast, which means that cases from different machines can't always be incorporated effectively in the same dataset. However, US data have other advantages over MRI data which we have previously described in section 1.2.

1.4 Legal and Ethical Considerations

In our project, we had to consider the ethical and legal aspects of working with medical images that contain medical personal data of individuals. As we mentioned in section 1.3, the US images we used in our project were collected from our clinical collaborators from the Department of Neurosurgery in Charing Cross Hospital, Imperial College London, UK, with the legal permission of individual patients. These data were annotated from our clinical collaborators and were given to us legally by them for use in our project and various other projects of our laboratory. As such, we are legally free to process and use these data in any way we deem fit for the completion of our project.

There are no further ethical and legal aspects of this project that we need to consider, according to our cross-check with the ethics checklist provided by the Department of Computing at Imperial College London.

1.5 Project Goal and Outcome

The aim of this project is to build a CAD system able to identify tumor regions in US brain images. In our approach to achieve this goal, we decided to employ deep

learning techniques and more specifically Convolutional Neural Networks (CNNs). The system we propose is comprised of a deep learning architecture originally used by its creators for salient object detection in natural, non-medical images. This architecture is trained with the custom dataset described in section 1.3 and expanded with a suitable post-processing method to produce high-quality results.

Due to our limited dataset, we focus mostly on a specific type of brain tumors called Glioblastomas (GB). Half of the data we used for training are US images of grade IV Glioblastomas, according to our clinical collaborators. Despite this, we are confident that the framework we propose can also be applied effectively to different kinds of tumors given sufficient medical datasets.

To determine the most appropriate machine learning architecture for our CAD system, we considered and experimented with several different proposed deep learning architectures. The results and the performance for each one are presented in detail throughout this report.

1.6 Project Delineation

At the start of this project, we tried to set some guidelines for its progress. Figure 1.2 is a Gantt chart containing all the main milestones of the project and establishing a rough schedule for all the tasks we wished to complete. This Gantt chart represented a rough initial schedule and was subject to alterations depending on the progress of individual milestones.

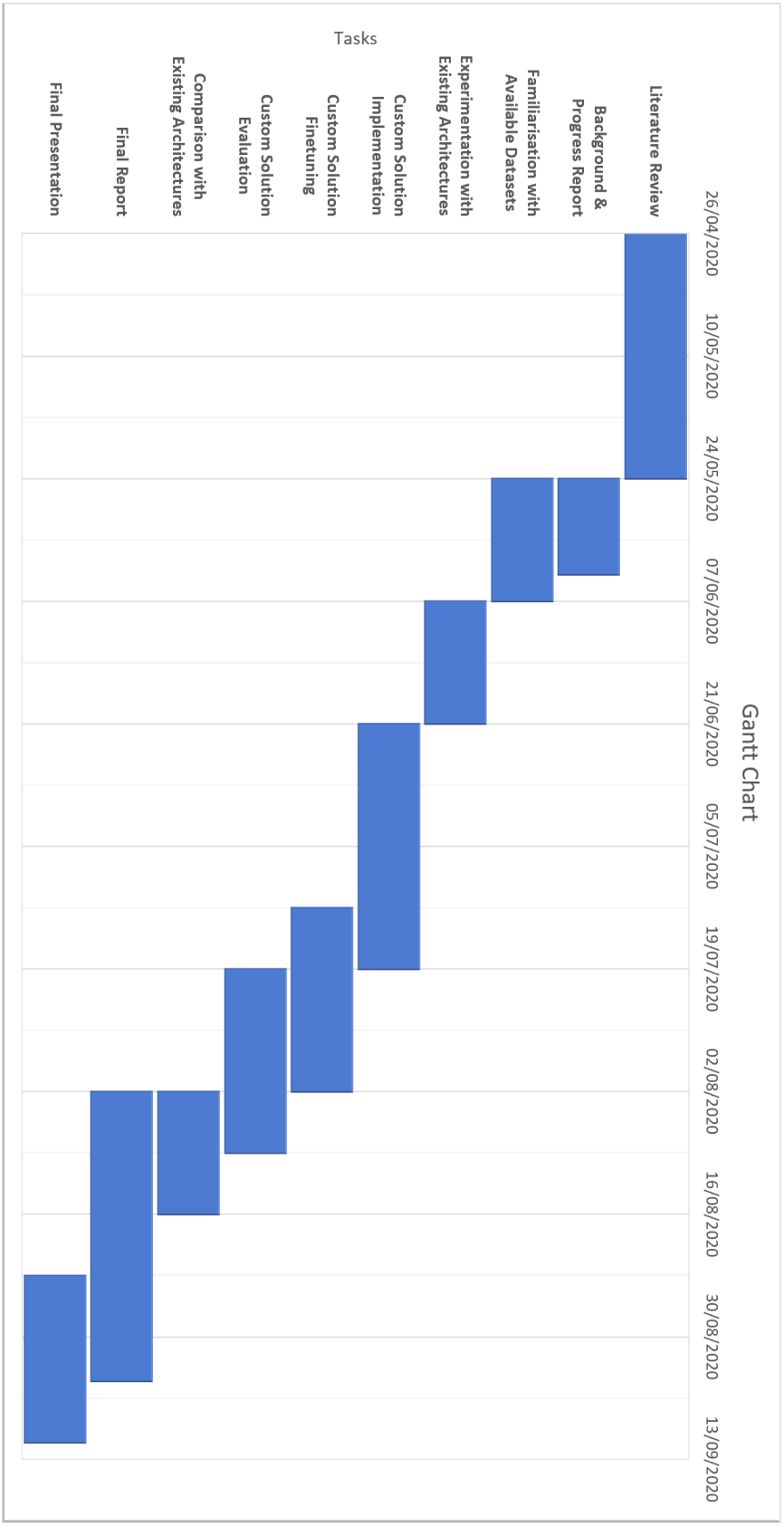


Figure 1.2: Gantt chart presenting the current progress and future milestones of the project.

Chapter 2

Background

Identifying brain tumors or brain lesions in a provided image modality is in actuality the problem of detecting and segmenting different regions of the brain. This problem of tissue segmentation or salient object detection has been studied extensively in the last couple of years. There have been many different methods proposed but the most promising ones leverage deep learning techniques. With the advent of deep learning and more specifically CNNs there have been many recent works trying to establish an optimal framework for brain tissue characterization. In this section, we try to present the state of the art techniques in brain tumor segmentation in different image modalities as it has been formed during the last years.

2.1 Salient Object Detection

As we previously mentioned, the problem of brain tumor segmentation is essentially a more specific case of salient object detection in medical images. Many of the most prominent salient object detection methods that have been proposed over the last couple of years utilise the idea of side outputs which were initially presented and incorporated in (6). In this work, the authors proposed the Holistically Nested Edge Detection (HED) architecture, a fully convolutional network, hence the characterization ”holistically”, for edge detection in 2D images. Edge detection, while an altogether different and somewhat easier problem than salient object detection, gave rise to ideas that enabled the creation of more accurate salient object detection frameworks. The architecture in (6) is utilising the popular VGG-16 architecture as a backbone, which is enriched with side outputs at different stages of the architecture. These side outputs are then fused together (weighted fusion) and the result is used in the loss function with the addition of a new term, thus providing deep supervision. This kind of deep supervision leads to feature maps of side outputs at different stages to be inherited and possess progressively refined edges, hence the name ”nested” for the proposed HED architecture. This in turn leads to the final output map to exhibit more accurate edge detection.

In (7) the authors, based upon the idea of the side outputs, expanded the HED architecture with the addition of short connections from deeper side outputs to all the shallower ones to produce a salient object detection architecture. The main inspi-

ration for this addition was that deeper side outputs are good at encoding semantic features like the location of salient objects in an image, but the produced shapes lack regularity, while shallower side outputs can more accurately detect the edges in an image. The proposed short connections architecture enabled the combination of features from both shallow and deep side outputs and demonstrated state of the art salient object detection capabilities. A post-processing conditional random field (CRF) method was also used to further refine the produced salient object maps.

Following up on the idea of combining the features of deeper and shallower side outputs in a more efficient manner than with short connections, the state of the art in salient object detection is further improved in (8). Again, with the HED architecture as a backbone, the authors proposed to apply residual learning in side outputs so that the architecture learns residual features during supervision and the resolution of saliency maps in side outputs is gradually enhanced. This approach achieves improved salient object detection by producing higher resolution saliency maps while utilising smaller kernels in convolutional layers of side outputs, thus achieving smaller model size and faster inference. What constitutes this approach possible is a proposed novel and light-weight reverse attention block R, which is included in each side output and used alongside a top-down reverse attention method, according to which, previously predicted regions are eliminated so that the blocks R are guided to learn remaining undetected regions of salient objects. The proposed architecture requires a saliency prior map which is gradually refined to produce the final results, although it is clear from (8) that practically most saliency priors, which can be derived using previously known methods, are sufficient for the proposed architecture to work.

Although the proposed architecture in (8) seems to currently be among the best-performing ones for state of the art salient object detection, the authors in (9) propose a different altogether architecture to perform salient object detection and produce only slightly worse, if not similar, state of the art results. In the proposed EGNet architecture, the main idea lies in utilising different side outputs of the same architecture to do different tasks. As we have previously explained when presenting the main idea behind (7), shallower side outputs are better at detecting edges. The EGNet architecture leverages this fact and is trained, with deep supervision, to do salient edge detection in the first side output, while the rest side outputs are trained, with deep supervision, to do salient object detection. The feature maps produced by deeper side outputs are combined in a novel one-to-one guidance module with the salient edge map of the first side output to produce sub-side features. These sub-side outputs are deeply supervised as well and are fused together to produce the final saliency map which is also supervised. In this way, the EGNet architecture can provide highly accurate salient object detection with clear edges in saliency maps.

There exist approaches for salient object segmentation that try to produce refined saliency maps with even more innovative ideas. A characteristic one for its distinct approach is proposed in (10). In this work, the authors use a recurrent fully convolutional network architecture to recursively refine saliency maps in each time-step. This approach requires a saliency prior map which is used for the initial refinement in the first time-step. While this architecture, as well as other ones, try

to leverage recurrent networks to do salient object detection, it seems that state of the art performance in salient object detection is currently achieved only by fully convolutional architectures such as the ones we presented previously.

As we can see, the architectures that have been proposed in the last years for salient object detection are getting more and more complex and the constant rise in computing power enables the creation of architectures with ever more intricate designs, which consist of several different modules, such as those in (8) and (9).

2.2 Brain Tumor Segmentation Using MRI Data

Brain tumor segmentation is essentially an extension of the classic problem of salient object detection. However, the nature of the data required, which are collected by different image modalities such as MRI or US, make it a distinct case worth studying and analysing. The fact that trained surgeons are needed to perform time-consuming manual labeling of medical images, as well as the data scarcity due to privacy reasons, are some of the idiosyncrasies of brain tumor segmentation which constitute this problem significantly harder than salient object detection. Moreover, all the methods that we presented until now were focused on 2D images, and while they can be applied to 2D slices of 3D MRI images or 3D US images, there is significant importance in analysing how 3D data can be used to take advantage of the spatial context of a 3D brain volume.

One groundbreaking entry utilising 3D MRI data is that of (11) which presents an architecture called Deepmedic. The authors propose an 11-layer deep CNN architecture for brain lesion segmentation which is trained directly with 3D patches of 3D MRI images. While it would be impossible, weight-wise, to train such a deep architecture with 3D images, a combination of the use of small 3x3x3 kernels in convolutional layers, as well as the use of two parallel convolutional pathways, enable the network to overcome GPU constraints during training and exhibit state of the art results. The two pathways are responsible for processing multi-scale 3D images. Each 3D image patch provided in the architecture is sampled in normal and lower resolution, which is obtained by taking a bigger patch of the 3D image with the same voxel as the center and downsampling it. The fist pathway processes patches of 3D images of the initial resolution and is responsible for identifying the finest details of the brain volume while the second pathway processes the reduced resolution patches and makes it possible to avoid loading large parts of the original 3D image into the memory but still preserve enough spatial information of the brain volume in each patch. Finally, the authors proposed and implemented a 3D post-processing conditional random field (3D CRF) method to further refine the produced segmentation maps.

In a later publication of the same authors (12), the Deepmedic architecture is further extended to include residual connections that connect the outputs between every two layers. The residual connections help in better preserving the signal in the network and provide increased sensitivity which leads to a slight enhancement in performance.

In (4) the authors offer a concise synopsis of the most promising approaches

for segmentation in brain MRIs and they classify almost all proposed architectures until then in three categories. These are ”patch-wise CNN architectures” with multiple pathways many times, like the Deepmedic one we discussed earlier, ”semantic wise CNN architectures”, in which training happens on the whole 3D image, usually by using encoder-decoder architectures, and ”cascaded CNN architectures”, in which multiple CNNs, with different structure each, are trained and used together, with the later ones refining the results of previous ones. Finally, the authors in (4) offer a detailed analysis of most of the publicly available datasets containing brain MRIs which are used in literature both for training and fair comparison of emerging architectures, performing brain lesion and brain tumor segmentation, with existing ones.

In a similar note, (3) branches out and summarises the state of the art deep learning techniques in medical image analysis in general, as it is formed by more than 300 different publications. Specifically for brain lesion segmentation, the authors point out one of the biggest challenges found and discussed in most works, the class imbalance problem. This problem is mainly caused by the fact that brain volumes do not contain that many lesion regions as healthy ones which causes an imbalance during the training of deep models. The Deepmedic architecture presented in (11) solved this problem by sampling 3D image patches with a fifty percent probability of being centered around a lesion voxel or a voxel of a healthy region. The class imbalance problem is a challenge that all works that push the state of the art in brain lesion segmentation and tumor segmentation have to address somehow.

In the last years, more complex architectures like those in (13) and (14) have emerged that outperform older approaches and push the state of the art in brain tumor segmentation further ahead. However, these architectures need big datasets, big in the context of available 3D MRI datasets, to be trained sufficiently.

In (13) the authors propose an encoder-decoder architecture, which has 2 decoder branches. The first branch is a regular decoder while the second branch is the decoder part of a variational autoencoder (VAE) architecture. The second branch is used only during training to regularize the encoder part and guide the architecture towards desirable segmentation and is not needed during inference. The results of this work are impressive, as it can segment not only tumorous regions but also multiple nested regions inside the tumor, such as the tumor core and a further nested class, the enhancing tumor core.

Another architecture producing state of the art segmentation in 3D MRI images is the OM-Net architecture proposed in (14). The main idea of this work is that the same network is trained simultaneously, with different data, to perform 3 different tasks, a ”coarse segmentation” of the whole tumor region, a ”refined segmentation” where the network can better identify the whole tumor as well as nested classes and a ”precise segmentation” where the enhancing tumor region is segmented. The tasks are introduced to the network one by one, from coarse to fine, after the network has exhibited partial convergence to the training data of the previous tasks. What enables the network to learn three different tasks simultaneously is the use of a shared backbone where the data of the three tasks are concatenated, provided to this backbone, and split again in the output. This sharing of the backbone, and thus

the weight parameters, is what causes the architecture to share knowledge between the three tasks. Produced features for all tasks are further shared from the easier to the harder task. The architecture involves even more intricate ways of leveraging learned features from the three tasks and combining them. What is important to note is that the authors consider their architecture as an improvement over the model cascade (MC) method which uses a cascade of CNNs to learn different tasks thus overcoming the class imbalance problem mentioned earlier.

After the authors of (13) and (14) present their architectures in full, they show that performance can be further improved by using an ensemble of 10 similar architectures like the proposed one. This is a trick proposed initially in (15) which explained that using, during inference, an ensemble of multiple, independently trained, CNNs with different parameters and even different architectures can result to even more accurate segmentation.

While the methods in (13) and (14) have been proven to produce state-of-the-art results, the scarcity of available 3D MRI data for brain lesion segmentation, other than a few public datasets which are described in full in (4), constitutes them harder to use. This is why training with only a few 3D samples is investigated in the last years.

In (16) a novel framework for ”few-shot segmentation” is proposed. The framework consists of two different arms that use almost identical encoder-decoder architectures. The two arms, ”conditioner arm” and ”segmenter arm”, are connected through ”squeeze and excite blocks” (SE) in all stages of the encoder-decoder architectures. The input to the conditioner arm, during inference, is a support set, which contains 2D slices of a 3D MRI volume with a new annotated class, previously unseen by the model during training. The segmenter arm is given as input during inference, a different query 3D volume, and the model can utilise the support set to perform segmentation of the previously unseen class on the query volume. While this method achieves much lower performance than the previously mentioned ones, it does not require a pre-trained model to perform 3D segmentation in medical images and can do so using only a few 2D slices of a 3D annotated volume as a support set.

2.3 Brain Tumor Segmentation Using US Data

In the introduction section, we described shortly the advantages of using US data for developing a CAD system able to perform automatic brain tumor segmentation. Although not much has been done towards this goal due to the aforementioned challenges of US data, there have been a few noteworthy approaches with encouraging results like the following one.

In (17), the authors propose an encoder-decoder architecture that modifies slightly the popular U-net architecture, initially proposed in (18), to handle consecutive 2D slices of 3D US data of brain volumes and perform intra-operative segmentation of brain volumes. The main advantage of using US data according to the authors of (17) is that they can be collected in real-time and the whole architecture can perform intra-operative brain segmentation thus taking into account the brain

deformation which happens during tissue resection. This would not be possible by using MRI data since they can not be collected in real-time, on the spot, without moving the patient from the operating table.

In figure 2.1 we try to summarise in a table all the architectures and methods we discussed so far and compare some of their traits regarding training data and processing times. In this way, we aim to make clearer the advantages and disadvantages of each one of these approaches.

Architecture	Training Data Type	Training Dataset(s) Used	Training Dataset Sise	Data Augmentation	Post-processing	Training Time	Inference Time	Saliency prior
HED	2D images	BSDS500, NYUDv2	200 - 381	✓	×	7 h on single NVIDIA K40 GPU	400 ms	×
Short Connections Architecture	2D images	MSRA-B	2,500	✓	CRF	< 8 h	< 500 ms	×
Reverse Attention-based Residual Network	2D images	MSRA-B, MSRA-10K, DUTS	2500 - 10,000	✓	×	< 2 h	40 fps = 25 ms	✓
Egnet	2D images	DUTS-TR	10,553	×	×	-	-	×
Recurrent Fully Convolutional Network	2D images	PASCAL VOC 2010, THUS10K	10,000 - 11,355	×	×	-	-	✓
Deepmedic	3D MRI volumes	TBI, BRATS 2015	46 - 220	✓	3D CRF	-	-	×
Deepmedic + Residual Connections	3D MRI volumes	BRATS 2015	220	-	Removal of components smaller than 250	-	35 s per multi-modal scan	×
Autoencoder Regularization Architecture	3D MRI volumes	BraTS 2018	285	✓	×	2 days on single NVIDIA Tesla V100 32 GB or 6 h on NVIDIA DGX-1 server	400 ms on single V100	×
OM-Net	3D MRI volumes	BraTS 2015, BraTS 2018	274 - 285	×	Removal of isolated small clusters + K-means based method	-	-	×
SE Guided Few-shot Architecture	3D MRI volumes	Visceral Dataset (silver corpus scans)	65 3D Volumes + few 2D slices of 1 3D Volume (support set)	✓	×	-	-	×
Modified U-net Architecture	Consecutive 2D slices of 3D US volumes	RESECT dataset	26 3D volumes	×	Thresholded output predictions	-	15 s	×

Figure 2.1: Comparison of discussed architectures. The architectures are presented in the table in the same order that they were introduced in the text.

Chapter 3

Methods

3.1 Pre-training

At its core, our proposed CAD system consists of a deep learning architecture capable of performing salient object detection. All the architectures for salient object detection that we and present in this chapter and experiment with, in chapter 4, employ the idea of transfer learning which is used in their main backbone.

Transfer learning is the practice of training a model in a large dataset and then using this model, or transferring the learned weights of the model in a new but similar model, and then training the resulting model on a different dataset. In this way, the weights from the first training session are used as an initialisation for the training in the new dataset. Training a neural network from scratch in a huge dataset can take a long time (weeks) but by using transfer learning from a pre-trained network, even when the pre-trained network was trained for a somewhat different job, the required training time can be reduced substantially. Moreover, transfer learning from huge datasets can help neural networks to converge to an acceptable solution when using smaller training datasets, which would not be possible otherwise due to the limited size of the smaller training dataset.

There are two different neural network architectures, commonly used as backbone modules in most salient object detection architectures, the VGGNet (19), and ResNet (20) architectures.

3.1.1 VGGNet

This neural network architecture was proposed in (19) and exhibited state-of-the-art performance in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). The input to the network is a 224x224x3 image which is classified into 1000 classes in the output. Its key characteristic is the use of small 3x3 convolution filters in the network's convolutional layers, which made it possible to build a deeper than usual architecture. Apart from the convolutional layers, the architecture includes five max-pooling layers, which are included between some of the convolutional layers. After each max-pooling layer, the convolutional layers used, have double the number of channels than those used before the max-pooling layer. Finally, VGGNet ends with

three fully connected layers and a softmax layer. VGGNet can be built with a varying number of weight layers (convolutional and fully connected layers) ranging from 11 to 19, all of which are followed by a rectification non-linearity (ReLU). In figure 3.1 we can see the popular VGG-16 architecture.

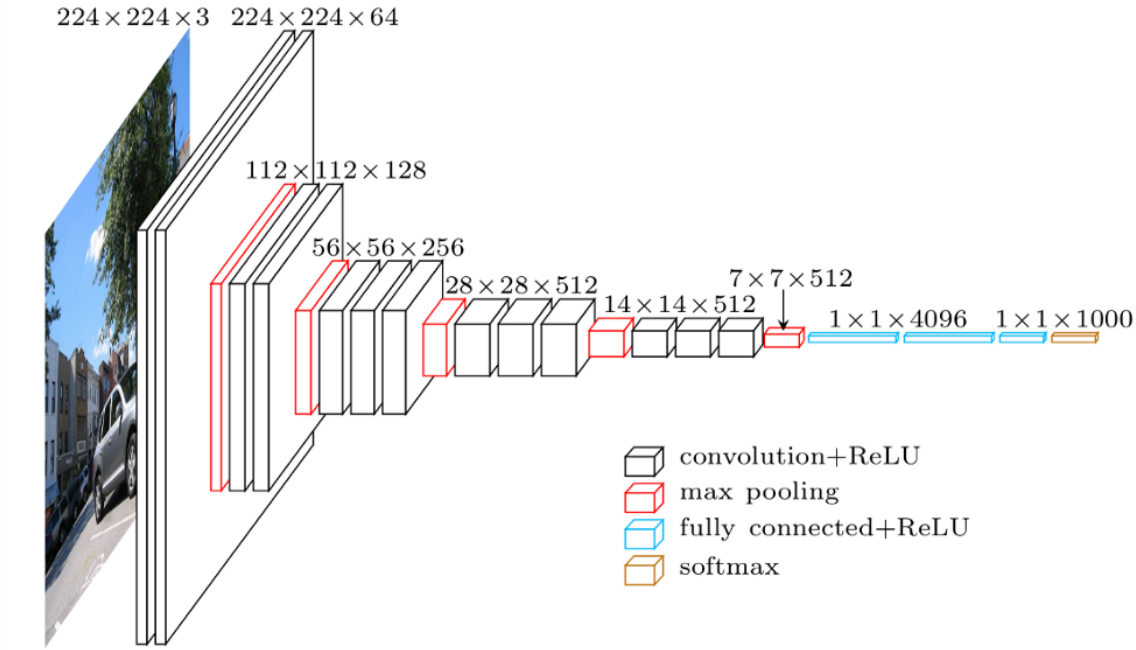


Figure 3.1: VGG-16 architecture. This architecture includes 13 convolutional layers and 3 fully connected layers. Figure is taken from: `https://towardsdatascience.com/extract-features-visualize-filters-and-feature-maps-in-vgg16-and-vgg19-cnn-models-d2da6333edd0`

3.1.2 ResNet

The ResNet model proposed in (20) has a deeper architecture than VGGNet models, achieving a depth of up to 152 layers, which is 8 times deeper than VGGNet, while having lower complexity. This architecture won the first place in the ILSVRC challenge of 2015 and has been used since then as a backbone in several deep learning frameworks.

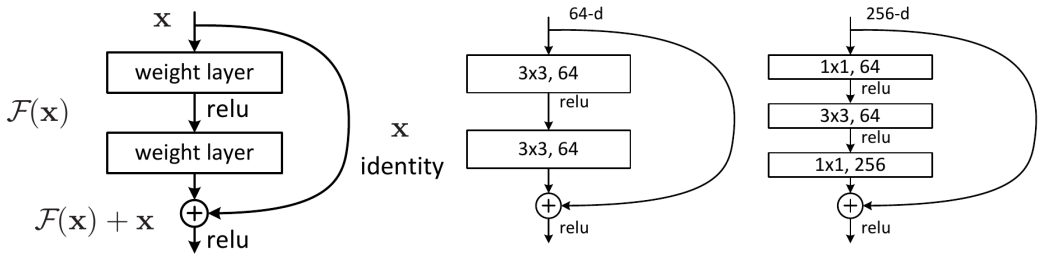
The main reason for the success of the ResNet architecture is that it managed to overcome the problem of vanishing and exploding gradients, as well as the degradation problem of the training accuracy. The authors in (20) use this term to explain that when adding deep layers to a model that is already deep enough for the problem it is trying to solve, the training accuracy deteriorates. Both these problems are tackled with the use of a residual learning building block.

The residual learning block, which can be seen in figure 3.2a, consists of some convolutional layers and shortcut connections that skip these layers. The shortcut connection outputs are added to the outputs of the skipped layers. The ResNet architecture consists of several such blocks used in a row.

The authors in (20) hypothesize that while the shortcut connections do not add any computational complexity to the network it is easier, when we want the skipped layers to fit an underlying function $H(x)$, to make the layers fit the underlying residual function $H(x) - x$. This happens because the underlying optimal function $H(x)$, that we want the layers to approximate, is closer to the identity mapping than the zero mapping and thus the network can better learn this underlying function $H(x)$ when it has a reference to the identity function x. Finally, the fact that the residual blocks converge easier to the optimal solution results in the deterioration of all the aforementioned problems encountered in deeper architectures.

The size of convolution filters used in ResNet is again 3×3 with stride 1 but we also have a stride of 2 when the shortcut connections are between feature maps of different sizes. This stride of 2 contributes to the down-sampling of the initial image dimensions by a factor of 2 and the increase in the number of channels by a factor of 2, as was the case in the VGGNet with the max-pooling down-sampling.

Finally, there are 2 different kinds of residual learning blocks, shown in figure 3.2b. The first kind has two 3×3 convolutional layers and is used in a ResNet-34 architecture. The second kind has a 3×3 convolutional layer in the middle and two 1×1 convolutional layers before and after for down-sampling and up-sampling the image to reduce the network complexity even further. This three-layer residual learning block is used in deeper ResNet architectures such as 50/101/152-layer ResNets.



(a) Shortcut connection. **(b)** Types of ResNet blocks.

Figure 3.2: Figure 3.2a shows the identity mapping performed by the short connection in a ResNet block. Figure 3.2b shows that a ResNet block can have either two or three convolutional layers. Figures taken from (20).

3.2 Deep Learning Architectures

In this section, we present all the different deep learning architectures we used in our experiments when trying to build the CAD system we described. These architectures were initially proposed for salient object segmentation in natural images, but as we explained in chapter 2, they can be applied in our problem of brain tumor segmentation as well.

3.2.1 Reverse Attention-Based Residual Network

We have already mentioned the Reverse Attention-Based Residual (RAS) Network architecture, proposed in (8), in section 2.1 and since it is one of the most recent works from our literature review as well as one of the best performing ones in most publicly available datasets for salient object detection in natural images, we decided to experiment with it first. According to (8), this network was based on the HED architecture and makes use of a VGG-16 architecture as the main backbone for HED. However, the authors have updated the code provided alongside their publication to a ResNet-50 backbone. The main ingredients of this architecture are as follows.

- A pre-trained ResNet-50 backbone.
- A "Multi-Scale Context Module" (MSCM).
- Four Reverse Attention (RA) Blocks R, used in side-outputs from the main backbone.

A simplified version of the whole architecture can be seen in figure 3.3.

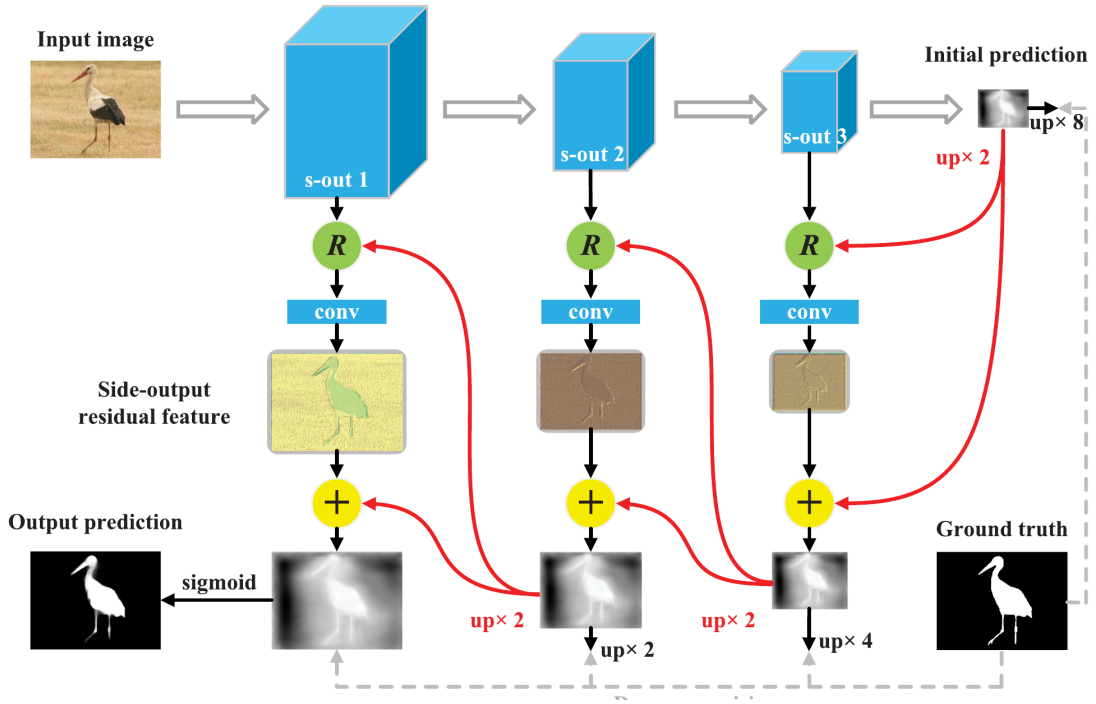


Figure 3.3: Simplified architecture of the Reverse Attention-based Residual Network with only 3 side outputs. Figure is taken from (8).

The input image is processed by the main ResNet-50 backbone and the result of the final layer is passed to an MSCM module which can make an initial rough saliency prediction. The MSCM module's architecture can be seen in figure 3.4. This module consists of 4 parallel branches $(b = 1, 2, 3, 4)$ with each branch having two convolutional layers, one layer with $(2b - 1) \times (2b - 1)$ kernel and one layer with

3×3 kernel. The four branches are concatenated and passed to a 3×3 convolutional layer to obtain an initial, one channel saliency map. The use of the MSCM module circumvents the need for an initial saliency prior to be provided to the network since the MSCM module's initial prediction is enough for the top-down reverse attention mechanism to work.

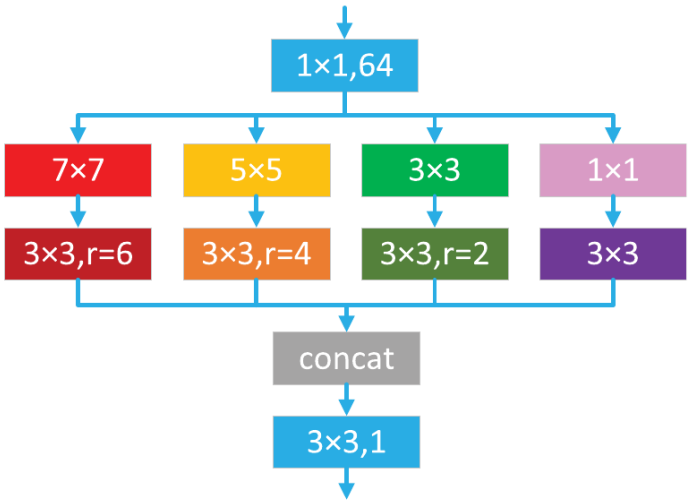


Figure 3.4: Architecture of the "Multi-Scale Context Module" (MSCM). Figure is taken from (8).

Finally, the RA blocks, utilise the up-sampled saliency prior of the MSCM module (in case of the last side-output) or the up-sampled prediction of the deeper side output, let us call this P, and perform an element-wise multiplication of the input of the RA block with the reverse attention weight, which is $1 - P$. We can see the RA block architecture clearly in figure 3.5. This is how the convolutional layers of each side-output are guided towards learning residual features until we get the final saliency map in the first side output.

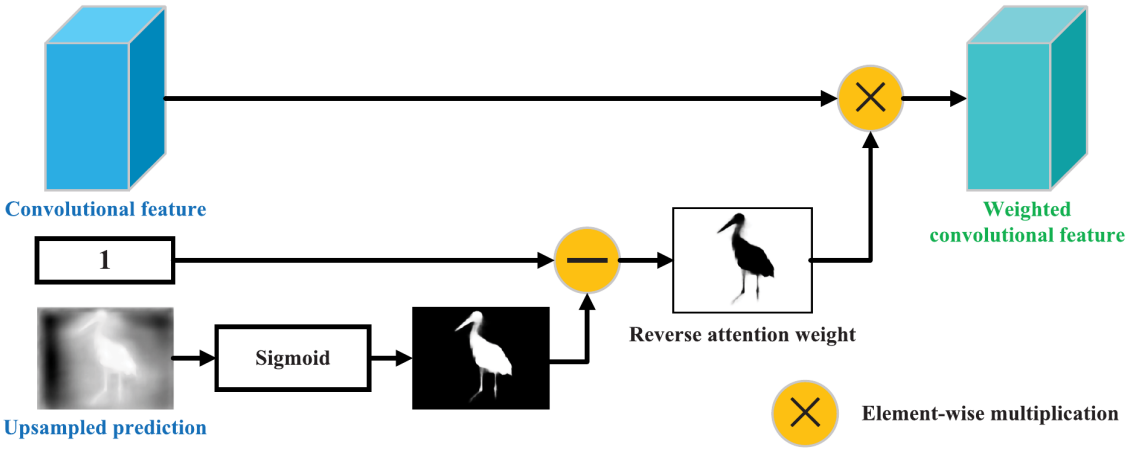


Figure 3.5: Architecture of the Reverse Attention (RA) block used in side-outputs. Figure taken from (8).

3.2.2 Cascaded Partial Decoder

The Cascaded Partial Decoder (CPD) architecture was proposed in (21) for performing salient object detection on 2D natural images. The main inspiration for this architecture is the fact that low-level feature maps, obtained from shallower layers of a deep network, have higher resolution, and thus their integration with high-level feature maps, obtained from deeper layers of a network, is time-consuming. Moreover, low-level features do not contribute as much to the end result as high-level features according to (21).

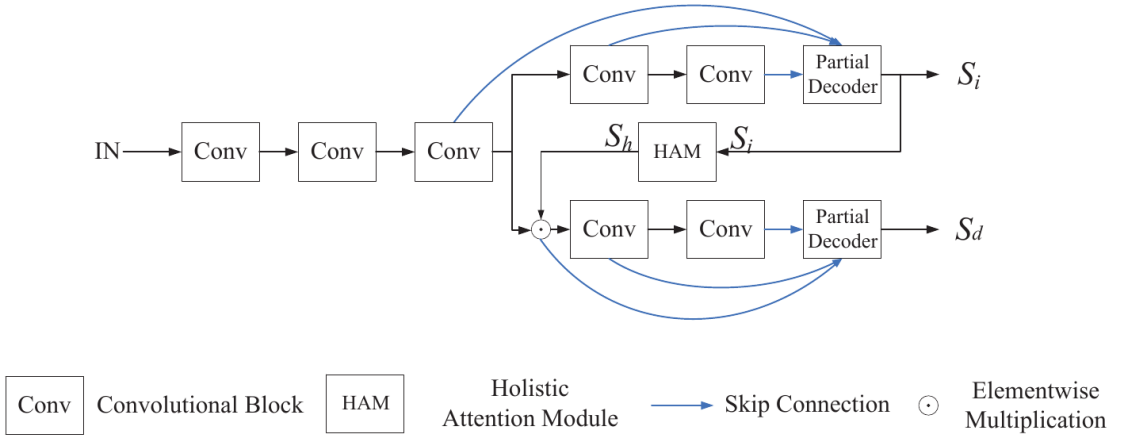


Figure 3.6: Cascaded Partial Decoder (CPD) architecture. Figure is taken from (21).

The CPD architecture, which can be seen in figure 3.6, ignores the lower level features by not integrating the results of the first two side-outputs of the main backbone with the results of deeper side-outputs. Only high-level features from deeper layers are passed in a decoder where they are aggregated and an initial rough

saliency map is produced. This saliency map is passed through a Holistic Attention Module (HAM) which enlarges the coverage area of this initial saliency map so that it can include some edge information, which may have been missing from the initial segmentation, or even areas that have been incorrectly segmented out. This happens with the following formula, which is presented in (21).

$$S_h = MAX\left(f_{min\_max}\left(Conv_g\left(S_i, k\right)\right), S_i\right) \tag{3.1}$$

S_i is the initial prediction of the decoder, $Conv_g()$ denotes a convolution operation with a Gaussian kernel k and $f_{min\_max}()$ is a function used to normalise the blurred areas in the resulting map in [0,1]. $MAX()$ is a maximum function responsible for increasing the weight coefficient of salient regions of S_i.

Furthermore, the CPD architecture has a second branch, in parallel and identical to the first one, which is comprised of layers identical to the deeper layers of the main backbone and a decoder as well, as can be seen in figure 3.6. The initial saliency map of the first decoder, after being processed by the HAM, is used in an element-wise multiplication to refine a side output, the third side output of the main backbone, which is passed to the second branch. The result of this operation is also aggregated with deeper side outputs of the second branch to produce the final saliency map. The second branch can refine the prediction of the first branch by "suppressing distractors" and focusing better on salient objects (21).

The two decoder modules used in the CPD branches are identical and each one consists of 3 basic blocks, which the authors call Receptive Field Blocks (RFB). The RFB block was initially presented in (22) and its creation was inspired by the visual system of humans and the structure of the receptive fields in it. In (21) the authors present a slightly altered RFB module which we can see in figure 3.7.

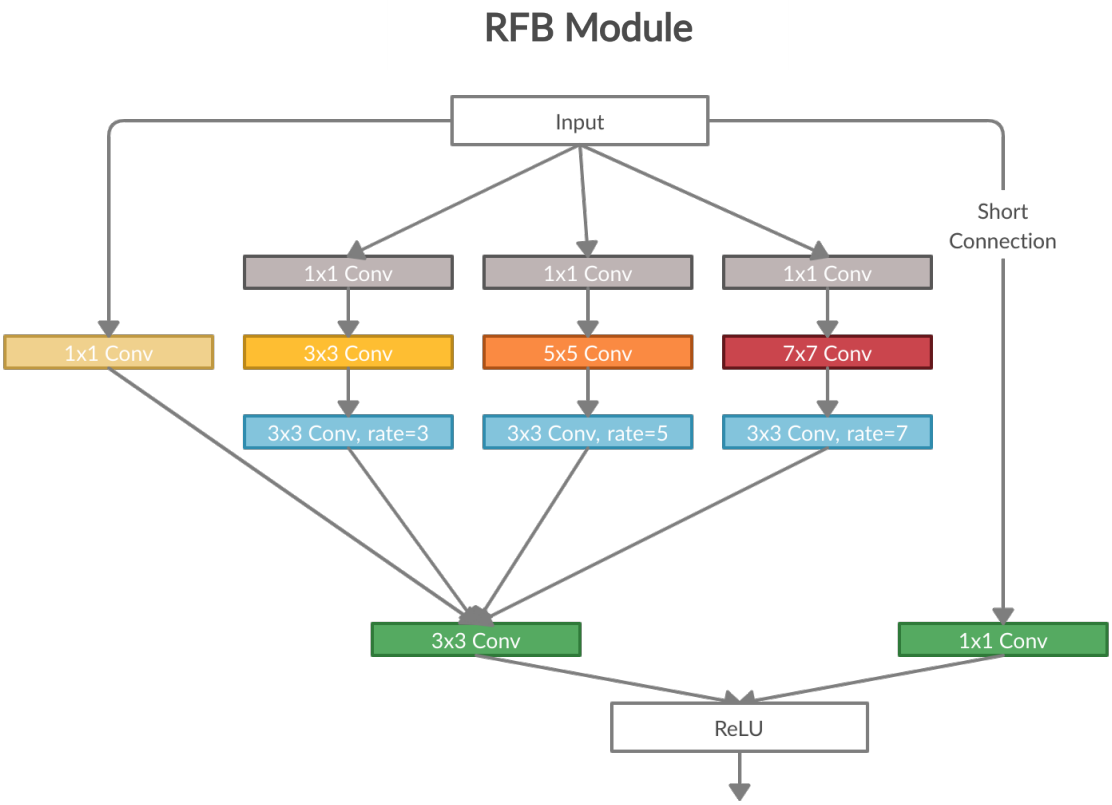


Figure 3.7: RFB architecture as it is described in (21). The architecture consists of 4 branches $(m = 1, 2, 3, 4)$ and a short connection. Each branch has a 32×32 convolutional layer responsible for reducing channel number to 32 for efficiency. Branches $2, 3, 4$ consist of varying size convolutional layers followed by a 3×3 convolutional layer with varying dilation rate.

Three RFB modules, such as the one presented in figure 3.7, are used in each decoder of the CPD architecture to enable each decoder to obtain discriminative features from the features of the main backbone of the CPD, which are fed into the decoders.

Finally, the main backbone of the architecture can be either a VGG-16 or a Resnet-50 network that is pre-trained on image classification. Both branches of the CPD are trained jointly with the total loss having the following formula.

$$L_{total} = L_{ce}\left(S_i, l \mid \Theta_i\right) + L_{ce}\left(S_d, l \mid \Theta_d\right) \tag{3.2}$$

$$L_{ce}\left(\Theta\right) = -\sum_{j=1}^{N} \sum_{c \in \{0,1\}} \delta\left(l^j = c\right) \log p\left(S^j = c \mid \Theta\right) \tag{3.3}$$

where $L_{ce}()$ is the sigmoid cross entropy loss, N is the number of pixels, $\delta()$ is the indicator function and Θ_i, Θ_d are parameter sets corresponding to the saliency maps S_i, S_d received by the two decoders.

3.2.3 F^{3}Net

The F^{3}Net architecture of (23), was proposed in an attempt to fuse multi-level features from different layers of a CNN in a more efficient way than addition or concatenation, thus improving salient object detection in natural images. The authors in (23) explain that features of different layers differ substantially from one another because of the varying receptive fields of different layers, which is a fact ignored by traditional feature fusion ways.

The F^{3}Net architecture consists of the following modules which tackle more efficiently the multi-level feature fusion problem.

- Encoder Module.
- Cross Feature Modules (CFM).
- Cascaded Feedback Decoder (CFD).

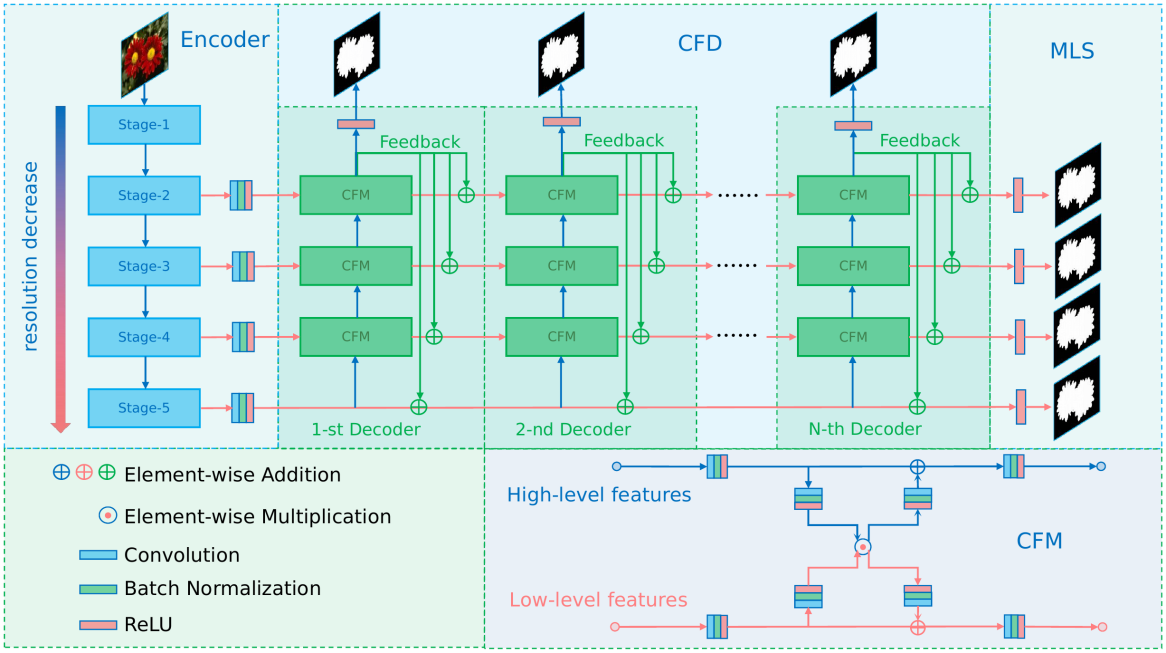


Figure 3.8: F^{3}Net architecture with the Encoder, CFMs, and CFD modules as well as the different points of supervision applied to the architecture. Figure is taken from (23).

The Encoder module is a simple ResNet-50 network, pre-trained to perform image classification, and can extract multi-level features with varying resolutions each.

Each CFM used in the F^{3}Net architecture takes as input lower level (f_l) and higher level (f_h) features from consecutive side-outputs, compares these together, and retrieves common parts by performing an element-wise multiplication of f_l, f_h and finally, adds the common parts back to f_l and f_h respectively by performing element-wise additions to f_l and f_h, as seen in figure 3.8. More formally,

$$f_l = f_l + M_l\left(G_l\left(f_l\right) * G_h\left(f_h\right)\right) \tag{3.4}$$
$$f_h = f_h + M_h\left(G_l\left(f_l\right) * G_h\left(f_h\right)\right) \tag{3.5}$$

where $M_l(), M_h(), G_l(), G_h()$ are each one the combination of convolution, BatchNorm and ReLU. At both inputs of the CFM 3×3 convolutions are used, both for f_l, f_h, to prepare them for the processing. At the outputs, 3×3 convolutions are used as well, to restore the original dimensions of f_l, f_h.

Thanks to CFM, the architecture gets rid of redundant information between the initial features, which can "corrupt" these features, and the framework avoids the background noise in f_l features while having sharper boundaries and more details in f_h features according to (23).

The CFD module consists of multiple identical decoders. Each decoder is comprised of 3 CFM modules, responsible for aggregating features of side-outputs 2 through 5 of the main Encoder in a bottom-up process. The produced features in the output of the final CFM module, let us call them f_{out}, which is supervised, constitute a coarse saliency map. However, this map may still suffer from inaccuracies because the features obtained from the encoder side-outputs may have not been stripped enough of noise or may have missed important salient object areas in the bottom-up process. For this reason, the authors in (23) propose to use f_{out} as feedback in a top-down process and refine the already refined by the CFMs side-output features even further. This top-down process is the down-sampling of f_{out} and its element-wise addition to the outputs of the CFM modules.

Having improved side-outputs with the top-down feedback process we described, we can use an identical second decoder with bottom-up and top-down processes as well, which can work with the improved side-outputs. In actuality, the F^3Net architecture uses N such decoders in parallel, as seen in figure 3.8, to perform iterative refinement of the side-outputs and get a refined final saliency map in the last decoder. This module, with the multiple identical decoders, is called Cascaded Feedback Decoder (CFD) in (23).

Finally, having obtained the final features for each level from the last decoder of the CFD module, we have Multi-Level Supervision (MLS), which can be seen in figure 3.8. The authors propose a "pixel position aware" loss (ppa) which combines a novel weighted binary cross-entropy loss (wBCE) and a novel weighted intersection over union loss (wIoU). They use this loss in the MLS as well as in the supervision of the final output of each decoder in the CFD module. So the final loss of the whole architecture is

$$L = \frac{1}{N}\sum_{i=1}^{N} L_{ppa}^{(i)} + \sum_{j=2}^{5} \frac{1}{2^{j-1}} L_{ppa}^{(j)} \tag{3.6}$$

The first term in equation 3.6 is an average of the ppa losses of the N decoders of the CFD module while the second term is a weighted average of the multi-level features obtained from the last decoder of the CFD module. Higher-level features have smaller weights in 3.6 because their ppa loss is bigger according to (23).

3.3 Post-Processing

3.3.1 Conditional Random Field

The Conditional Random Field (CRF) post-processing method was originally proposed in (24) but has been used in numerous works since then to improve image segmentation tasks. CRF is a class of discriminative probabilistic graph models that take into account the context information or the information of neighbouring pixels in an image to produce a prediction for each pixel. In this way, a segmentation map produced from a deep CNN can be further improved with the addition of a CRF as a post-processing step. However, a CRF model that is fully connected, meaning that every vertex in the graph is connected to all the others, has a high computational complexity during inference and thus introduces a significant run-time overhead when used as a post processing step. In (25) the authors propose an approximation algorithm that makes it possible to build efficient fully connected CRF models defined on all the pixels of an image.

The fully connected CRF model of (25) is formulated as follows. We define a conditional random field $(\boldsymbol{I}, \boldsymbol{X})$, where $\boldsymbol{I}$ is a random field defined over variables $I_1, I_2, ..., I_N$, representing possible input images of size N and $\boldsymbol{X}$ is a second random field, defined over the set of variables $(X_1, X_2, ..., X_N)$, with random variables $X_1, X_2, ..., X_N$ taking values of $\boldsymbol{L} = l_1, l_2, .., l_k$ labels. With this formulation of the conditional random field $(\boldsymbol{I}, \boldsymbol{X})$, an image of a segmentation problem can be treated as a fully connected graph $G = (V, E)$ with vertices $V = X_1, X_2, ..., X_N$ as the pixels of the image $\boldsymbol{I}$. The Gibbs distribution that characterises this Conditional Random field is

$$P(\boldsymbol{X} \mid \boldsymbol{I}) = \frac{1}{Z(\boldsymbol{I})} \exp\left(-E(\boldsymbol{X} \mid \boldsymbol{I})\right) \tag{3.7}$$

where $E(\boldsymbol{X} \mid \boldsymbol{I})$ is the energy function. In the proposed fully connected pairwise CRF model of (25) the energy becomes

$$E(\boldsymbol{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \tag{3.8}$$

where $\psi_u(x_i)$ is the unary potential. The unary potential is used to represent the relations between local image features of pixel i and its label. This can be obtained directly from a CNN model such as the ones we described in section 3.2. The term $\psi_p(x_i, x_j)$ is the pairwise potential. The pairwise potential represents the relationship between labels of neighboring pixels i, j and it is defined in (25) as follows

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}\left(\boldsymbol{f}_i, \boldsymbol{f}_j\right) \tag{3.9}$$

The $k^{(m)}(\boldsymbol{f}_i, \boldsymbol{f}_j)$ function symbolises m Gaussian kernels that are used to measure the spatial closeness and the color similarity of two pixels i, j of the image

with feature vectors $\boldsymbol{f}_i, \boldsymbol{f}_j$. Finally, the $\mu(x_i, x_j)$ function can model the compatibility between different labels and introduce a penalty for neighboring pixels with incompatible labels.

The key insight of (25) is the fast calculation of the distribution $P(\boldsymbol{X} \mid \boldsymbol{I})$ via a mean field approximation algorithm. The algorithm approximates $P(\boldsymbol{X})$ as a distribution $Q(\boldsymbol{X})$ which can be expressed as

$$Q(X) = \prod_i Q_i(X_i) \tag{3.10}$$

where $Q_i(X_i)$ are independent marginal distributions. The algorithm consists of multiple inference steps (epochs) and in each epoch, the main bottleneck of the algorithm lies in a message-passing step, which involves a convolution with a Gaussian kernel. This Gaussian filtering in the feature space can be computed in linear time with the use of down-sampling and up-sampling, thus reducing the overall computational complexity of the mean field algorithm, as seen in (25). This is how the CRF model we described is capable of enhancing image segmentation tasks of CNNs without introducing significant run-time overhead.

In a later work, the authors of (26), expressed the CRF model we described with the mean field approximation as a Recurrent Neural Network (RNN). In this way, a CNN can be bootstrapped with the proposed CRF as RNN model and the whole architecture can be trained together. This results in an architecture that has the advantages of CRF incorporated in it and doesn't require any post-processing steps.

3.4 Implementation Details

3.4.1 Implementation Environment

We used the same environment for all our experiments for fair and accurate comparison. We chose the pro version of Google Colab for our experiments since this cloud service provides cheap runtime environments with powerful GPU accelerators and enough processing power to conduct our experiments. The GPU accelerator in our runtime was an NVIDIA TESLA P100.

3.4.2 Implementation Code

The code for the RAS Network, the CPD, and the F^3Net architectures is taken from the public GitHub repositories accompanying each one of the respective papers that proposed the architecture. The code repository which contains the original code for each architecture can be seen in the following table.

Architecture	Code Repository
RAS Network	https://github.com/ShuhanChen/RAS-pytorch
CPD	https://github.com/wuzhe71/CPD
F^3Net	https://github.com/weijun88/F3Net

Table 3.1: Public GitHub code repositories with the code for each architecture we used.

We used the following hyperparameters in our experiments with each model. The RAS Network was trained using the Adam optimizer for 30 epochs with an epoch decay rate 0.1, which was applied after epoch 14. We also chose a learning rate $lr = 5 \times 10^{-5}$, a training batch size of 10, and a training image size of 448×448.

The CPD was again trained using the Adam optimizer for 39 epochs(we set epoch to 40 in our code) with an epoch decay rate 0.1, which was applied after epoch 24. We also chose a learning rate $lr = 1 \times 10^{-4}$, a training batch size of 10, and a training image size of 448×448.

The F^3Net was trained with an SGD optimizer for 32 epochs with momentum $m = 0.9$, a weight decay of 5×10^{-4}, and the learning rate was adjusted using warm-up and linear decay strategies like the authors propose in (23). We set the maximum learning rate as $lr = 0.005$ for the ResNet backbone and $lr = 0.05$ for the rest of the F3Net parts. Finally, we used a training batch size of 10 and a training image size of 448×448.

Finally, we used the code, with minor alterations, from the 4 public GitHub repositories that we present in table 3.2, to perform various post-processing and visualisation methods, which we describe in detail in the next chapter.

Type of Code	Code Repository
CRF (main library)	https://github.com/lucasb-eyer/pydensecrf
CRF (inference loop)	https://github.com/dhawan98/Post-Processing-of-Image-Segmentation-using-CRF
Visualisations	https://github.com/utkuozbulak/pytorch-cnn-visualizations

Table 3.2: Public GitHub code repositories containing the code for various post-processing or visualisation methods we used.

Chapter 4

Experiments and Results

4.1 Evaluation Metrics

We evaluated the produced maps of each deep learning model at the pixel level. For each pixel, we want to classify it as a pixel belonging in the tumor or background class. So we ended up using a fixed threshold to binarize the segmentation maps as mentioned in (27). The pixel values in the segmentation maps are normalized in the $[0, 1]$ area so for pixel values equal or above 0.5 (corresponds to pixel color values equal or above 128) we classify the respective pixel as one belonging to the tumor class, according to the model, and we denote this pixel as P_t whereas, for pixel values bellow 0.5 we classify the respective pixel as one belonging to the background class and we denote this pixel as P_b. If a pixel is correctly classified we denote this with the subscript t, from true, so a pixel symbolised as P_{tt} means it was classified correctly by the model as a tumor pixel (true positive), while a pixel symbolised as P_{bt} means it was classified correctly as a background pixel (true negative). If a pixel is incorrectly classified we denote this with the subscript f, from false, so a pixel symbolised as P_{tf} means it was classified incorrectly as a tumor pixel (false positive), while a pixel symbolised as P_{bf} means it was incorrectly classified as a background pixel (false negative).

With this formulation in mind, we introduce the evaluation metrics we used to assess the quality of the produced maps.

The first metric we used, $global\,accuracy$, is a simple ratio of the number of all the correctly classified pixels by the model, to the total number of pixels in the map. This is the simplest metric we can use to assess the accuracy of a model.

$$global\,accuracy = \frac{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{l} P_{bt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{l} P_{bt} + \sum_{i=1}^{m} P_{tf} + \sum_{i=1}^{n} P_{bf}} \tag{4.1}$$

The next metric, $mean\,accuracy$, is a mean of the achieved accuracy of the model for each one of the two classes, tumor and background.

$$mean\,accuracy = \frac{1}{2}\left(\frac{\sum_{i=1}^{k} P_{tt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{n} P_{bf}} + \frac{\sum_{i=1}^{l} P_{bt}}{\sum_{i=1}^{l} P_{bt} + \sum_{i=1}^{m} P_{tf}}\right) \tag{4.2}$$

We can evaluate the similarity between two sets by calculating the ratio of their intersection to their union. This is called Intersection over Union (IoU) or Jaccard Index and we can specify this metric for each one of the classes in our problem. For each class, the two sets are the pixels of this class in the ground truth map and the pixels of this class in the map produced by the model. Finally, by taking the mean of the Jaccard Indices for all classes, tumor and background, we have the $mean\,IoU$ metric (28).

$$mean\,IoU = \frac{1}{2}\frac{\sum_{i=1}^{k} P_{tt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{m} P_{tf} + \sum_{i=1}^{n} P_{bf}} + \\ + \frac{1}{2}\frac{\sum_{i=1}^{l} P_{bt}}{\sum_{i=1}^{l} P_{bt} + \sum_{i=1}^{m} P_{tf} + \sum_{i=1}^{n} P_{bf}} \tag{4.3}$$

A variant of the $mean\,IoU$ is the $WeightedIoU$, which takes into account the class imbalance problem, which can be extreme in medical brain tumor images, by weighting each one of the Jaccard Indices of a class with a weight according to the appearance rate of the class in the ground truth map. So for T number of tumor pixels and B number of background pixels we have

$$Weighted\,IoU = \frac{T}{T+B}\left(\frac{\sum_{i=1}^{k} P_{tt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{m} P_{tf} + \sum_{i=1}^{n} P_{bf}}\right) + \\ + \frac{B}{T+B}\left(\frac{\sum_{i=1}^{l} P_{bt}}{\sum_{i=1}^{l} P_{bt} + \sum_{i=1}^{m} P_{tf} + \sum_{i=1}^{n} P_{bf}}\right) \tag{4.4}$$

Finally, having defined $precision$ and $recall$ as

$$precision = \frac{\sum_{i=1}^{k} P_{tt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{m} P_{tf}} \tag{4.5}$$

$$recall = \frac{\sum_{i=1}^{k} P_{tt}}{\sum_{i=1}^{k} P_{tt} + \sum_{i=1}^{n} P_{bf}} \tag{4.6}$$

we define the last metric we used to evaluate the model produced maps, the $F1\text{-}score$ or $F\text{-}measure$ (29), which is the harmonic mean of $precision$ and $recall$

$$F1\text{-}score = 2\frac{precision \cdot recall}{precision + recall} \tag{4.7}$$

All these evaluation metrics were implemented in MATLAB with the help of the code of this public GitHub repository `https://github.com/jiwei0921/Saliency-Evaluation-Toolbox`. We mainly borrowed the structure of this code and some lines of code, but most of the metrics were implemented from scratch by us.

4.2 Dataset Use

4.2.1 Training and Evaluation Datasets

We presented in short in section 1.3 the data that we have in our availability to conduct our experiments and build our CAD system. However, contrast and intensity differences in the images produced by different US machines means that we can not use all our data, from the old and the new machine, in a single dataset. Moreover, the cases in our dataset were not of the same type of cancer, which made their combination even more challenging, given that we don't possess enough cases of each brain cancer type to make our CAD system completely independent from the type of cancer in a US image.

For these reasons, we decided to focus our experiments mainly on a specific type of tumor, Glioblastoma (GBM), which is the most aggressive type of cancer that begins within the brain, and use 3 out of the 6 cases, from the new machine, that we possess and are of such type. In our experiments we used 2 out of those 3 cases for training and the remaining case, case 7 (as it is named in our laboratory files), was used as a testing dataset for evaluation. We call this training dataset, which contains cases 2, 3 (as they are named in our laboratory files), "New", since it consists exclusively from cases from the new machine.

In a next step, we found out, through the trial of various case combinations, that the addition of 2 more training cases from the old machine further enhances our models' performance when tested on the same testing dataset(case 7). So we added these cases in the "New" dataset to create an improved dataset which we call "Final", which contains cases 2, 3, 9, 15 (as they are named in our laboratory files).

When extracting the binary labels out of the multi-label segmentation maps that we were provided with by our clinical collaborators, we consider only the green colour area as the tumor area which we want our CAD system to segment. Al the other areas such as the necrosis part are considered background pixels. Ideally, we would like our framework to be able to segment the necrosis part as well as the tumor area because, during a resection operation, the necrosis part must be removed by the doctor as well. However, in the experiments where we set as a tumor area to be segmented by the CAD system both the green and dark yellow colour (necrosis) areas we achieved much worse performance. Intuitively, trying to make our CAD system to segment both areas is like asking it to perform two separate jobs, which is why the network can't specialise in either one and we achieve worse segmentation results overall.

4.2.2 Data Augmentation

The two datasets we created, "New" and "Final", consist of extremely few images for a deep network to be trained sufficiently. Therefore, we need to perform data augmentation techniques to enlarge our training datasets. There are several data augmentation techniques commonly applied to image datasets, ranging from simpler to more intricate ones. For instance, we can employ simple image processing techniques like image cropping and flipping, or we can try more advanced methods

like the ones proposed in (30), in which the authors employ neural networks and Generative Adversarial Networks (GANs) to produce new images from the old ones and enlarge their dataset. In our project, we used the following image transformations to augment our data.

- Horizontal or vertical flipping of images.
- Left or right image rotation (rotation from -20 to $+20$ degrees).
- Random zooming in and out with a rate of 0.5 to 2.

There are two ways to apply the three techniques we mentioned and perform data augmentation, as a pre-processing step, in which we perform these transformations on the images of a dataset and we save the results to produce a much larger training dataset, or during training in the form of live data augmentation. In our case, our training datasets were extremely small for live data augmentation, a fact confirmed by our experiments, so we used data augmentation as a pre-processing step. In this way, our "New" training dataset grew to the size of 1560 US images and our "Final" training dataset grew to the size of 1600 US images.

We did not perform any data augmentation on the case of the new machine that we used for testing our models. So our testing dataset is comprised of 10 US images.

4.3 Experiments

4.3.1 Architecture Choice

We trained all 3 deep learning architectures that we mentioned with the "New" and "Final" datasets and evaluated them so as to choose the optimal architecture for our CAD system.

Model	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	F1-score
RAS	85.64	87.01	75.26	75.3	84.49
CPD	89.32	89.7	80.73	80.86	89.18
CPD-R	89.13	89.62	80.39	80.51	88.84
F3Net	87.13	87.84	77.27	77.41	86.61

Figure 4.1: A comparison of the performance of four different architectures, Reverse Attention Based Residual Network (RAS), Cascaded Partial Decoder with VGG-16 backbone (CPD), Cascaded Partial Decoder with ResNet-50 backbone (CPD-R), and Fusion, Feedback and Focus Network (F3Net). The New dataset(cases 2,3) was used for training and the evaluation was performed on the testing dataset(case 7) we described.

Figure 4.2: A graph format of the comparison presented in figure 4.1

Model	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	F1-score
RAS	88.43	88.73	79.28	79.45	88.69
CPD	88.66	88.67	79.58	79.74	88.94
CPD-R	91.02	91.26	83.51	83.63	90.91
F3Net	88.22	88.48	78.96	79.13	88.72

Figure 4.3: A Comparison of the performance of four different architectures, Reverse Attention Based Residual Network (RAS), Cascaded Partial Decoder with VGG-16 backbone (CPD), Cascaded Partial Decoder with ResNet-50 backbone (CPD-R), and Fusion, Feedback and Focus Network (F3Net). The Final dataset(cases 2,3,9,15) was used for training and the evaluation was performed on the testing dataset(case 7) we described.

Figure 4.4: A graph format of the comparison presented in figure4.3

It is clear from figures 4.3, 4.4, 4.3, and 4.4 that the CPD and CPD-R architectures outperform all the others both for the "New" and "Final" datasets.

As we explained in subsection 3.2.2, the CPD architecture does not fuse the lower level features to produce the final results because they do not contribute significantly to the accurate segmentation of images, since they contain mostly fine details such as edge information and not semantic information (21). The F^3Net architecture also does not fuse the features of the first side-output but still fuses the features from the second side-output, as opposed to the CPD architecture. This fact, coupled with the results from our experiments which show the CPD architectures outperforming the others consistently, allow us to make the educated guess that brain tumor segmentation in US images when using small training datasets like our own, does not benefit that much from low level features as it does from high level features that are produced by deeper layers.

Moreover, the CPD and CPD-R architectures achieve almost the same performance in the "New" dataset but the CPD-R architecture produces better results in the "Final" dataset. This is expected as the ResNet-50 architecture which is used as a backbone in CPD-R is deeper and more recent than the VGG-16 backbone used in CPD. For this reason, we chose the CPD-R architecture as the optimal architecture to use in our CAD System.

Finally, we compare the results from training each model with the "New"(cases 2,3) and "Final"(cases 2,3,9,15) datasets and we present this comparison in the graphs of figure 4.5.

Figure 4.5: A comparison between the performance achieved when training each architecture with the New(cases 2,3) and Final(cases 2,3,9,15) datasets.

Most of the models trained with the "Final"(cases 2,3,9,15) dataset outperform their counterparts trained with the "New"(cases 2,3) dataset. This makes sense because the images used to produce the "Final" dataset, are a super-set of the images used to produce the "New" dataset. This also proves that data collected by different US machines can be combined. However, this combination requires the combined cases to be carefully hand-selected, otherwise we may end up with worse performance, even though we used more cases, should the cases we combine differ substantially in contrast and intensity.

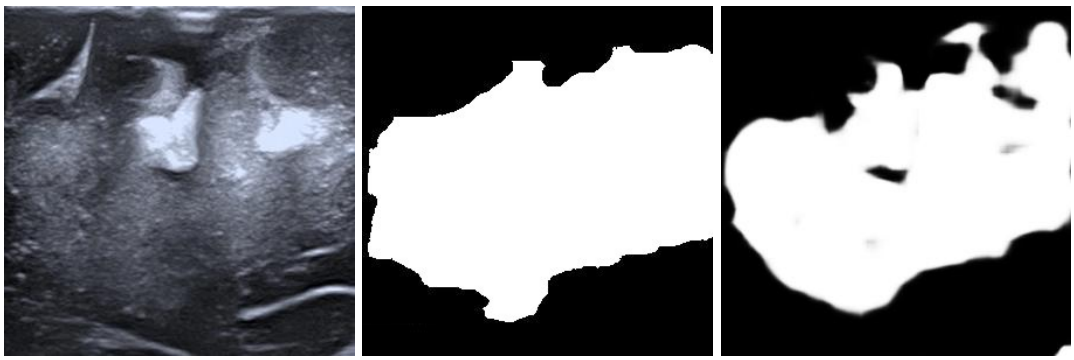
The challenge of combining cases from different machines in a single dataset is also reflected by the fact that one of our models, the CPD with the VGG-16 backbone,

performed marginally better on the "New" dataset, than the "Final" dataset as we can see from the respective graph in figure 4.5. One major difference of this model from the rest is that it is the only one with a VGG-16 backbone, while the other 3 models make use of a ResNet-50 backbone. While this could be an indication as to why this model was the only one that exhibited this behaviour, such a claim requires further experimentation.

From the comparison of figure 4.5 we can see that the CPD-R architecture trained with the "Final"(cases 2,3,9,15) dataset is the one achieving the best results overall, and so the one we propose for our CAD system.

4.3.2 Produced Segmentation Maps

In the following figures 4.6, 4.7, we present some segmentation maps produced by the CPD-R model, when trained with the "Final"(cases 2,3,9,15) dataset. We chose these two images because they were the best and the worst segmentation maps respectively from our testing dataset(case 7).



(a) Testing Image **(b)** Ground Truth Map **(c)** Segmentation Result

Figure 4.6: Positive segmentation example(case 7, image 4) of the CPD-R. (a) shows the testing image for which the CPD-R network achieved the best segmentation, (b) shows the ground truth map, and (c) shows the segmentation result of the CPD-R.

(a) Testing Image **(b)** Ground Truth Map **(c)** Segmentation Result

Figure 4.7: Negative segmentation example(case 7, image 9) of the CPD-R. (a) shows the testing image for which the CPD-R network achieved the worst segmentation, (b) shows the ground truth map, and (c) shows the segmentation result of the CPD-R.

In figures 4.8, 4.9 we present the evaluation metrics for the positive and negative segmentation examples of the CPD-R.

Image	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	F1-score
Positive example (case 7, image 4)	93.34	93.26	87.48	87.53	93.68
Negative example (case 7, image 9)	85.56	86.4	74.76	74.8	85.78

Figure 4.8: A comparison of the best and worst case segmentation results of the CPD-R architecture.

Figure 4.9: A graph format of the comparison presented in 4.8

As we can see our CAD system can detect accurately most of the tumor region. However, it is clear from the negative example that our proposed CAD system may sometimes falsely predict some harder to segment tumor regions near the boundaries of the tumor area as background pixels or some arachnoid as tumors. Moreover, even in the positive example, there are small regions inside the tumor area that are predicted as background pixels (small black holes). To try and mitigate this we decided to employ a post-processing CRF in our CAD system.

4.3.3 Post-Processing Results

Having trained the CPD-R architecture, we obtained the predicted segmentation map for each image in our testing(case 7) dataset, and we applied a post-processing CRF method on the resulting segmentation maps, as per (25). We used the algorithm described in (25) with two classes, one for the tumor and one for the background pixels, and we run the inference algorithm multiple times, for a different number of epochs each time.

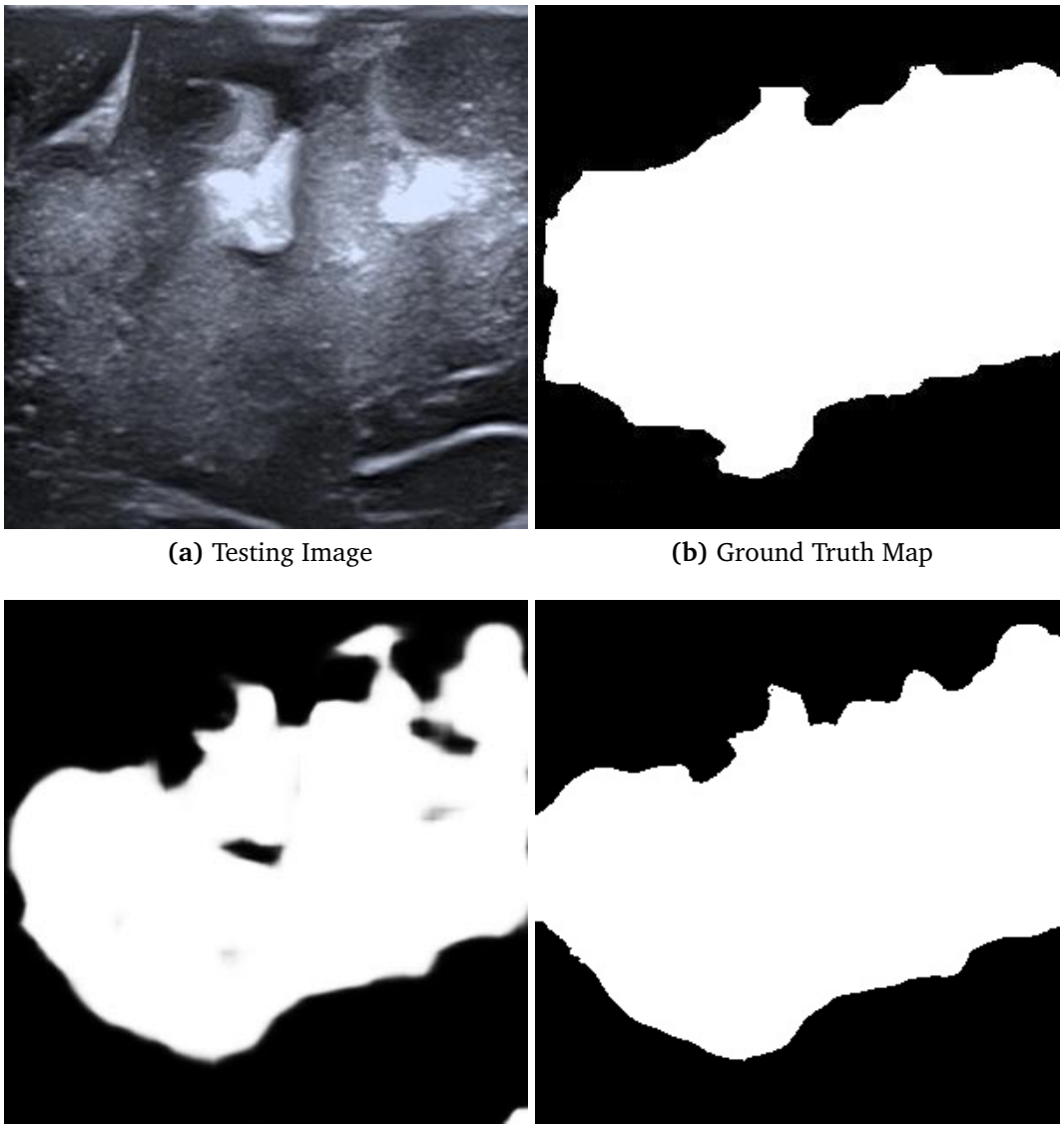
Method	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	F1-score
Basic	91.02	91.26	83.51	83.63	90.91
Basic + CRF (1 epoch)	91.47	91.69	84.27	84.39	91.38
Basic + CRF (5 epochs)	92.3	92.52	85.72	85.82	92.18
Basic + CRF (10 epochs)	92.52	92.72	86.11	86.21	92.4
Basic + CRF (50 epochs)	92.57	92.75	86.21	86.31	92.46
Basic + CRF (100 epochs)	92.57	92.75	86.21	86.31	92.46

Figure 4.10: A comparison of the performance of our CAD system with the addition of a CRF post-processing method. With Basic we refer to our CAD system without the addition of a CRF. Each line after the first one shows the performance of our CAD system with the addition of a CRF, for a different number of inference epochs in the CRF algorithm each time. The evaluation was performed in our testing(case 7) dataset.

As we can see from 4.10, with the addition of CRF, we get a performance improvement according to all our evaluation metrics. This improvement depends on the number of inference epochs of the CRF algorithm, with more improvement when we have more inference epochs. However, this improvement begins to saturate with higher numbers of inference epochs, while the inference time increases linearly with each inference epoch. Since we want our CAD system to be able to perform predictions in almost real-time, we propose to use the CRF algorithm with 10 inference epochs, since the slight performance improvement beyond this point does not justify the inference time overhead.

4.3.4 Post-Processing Segmentation Maps

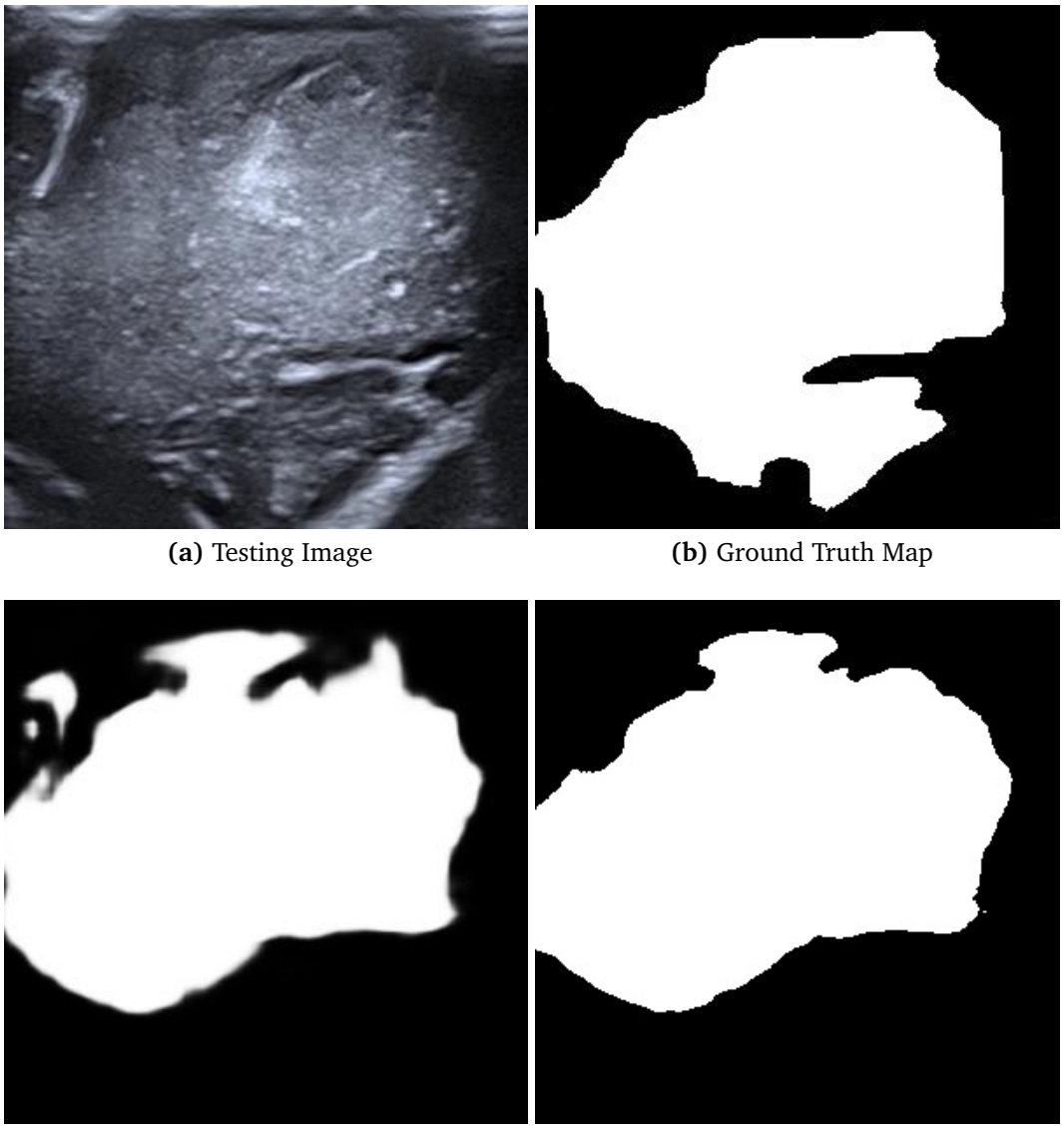
We present again the previous positive and negative segmentation examples of the CPD-R but we also add the resulting segmentation maps after we apply the CRF post-processing method we proposed in subsection 4.3.3.



(a) Testing Image **(b)** Ground Truth Map

(c) Original Segmentation **(d)** Segmentation after CRF

Figure 4.11: Positive segmentation example(case 7, image 4) of the CPD-R. (a) shows the testing image for which the CPD-R network achieved the best segmentation, (b) shows the ground truth map, (c) shows the segmentation result of the CPD-R, (d) shows the segmentation result of the CPD-R with the addition of the proposed CRF.



(a) Testing Image **(b)** Ground Truth Map

(c) Segmentation Result **(d)** Segmentation after CRF

Figure 4.12: Negative segmentation example(case 7, image 4) of the CPD-R. (a) shows the testing image for which the CPD-R network achieved the worst segmentation, (b) shows the ground truth map, (c) shows the segmentation result of the CPD-R, (d) shows the segmentation result of the CPD-R with the addition of the proposed CRF.

In figure 4.11d we can see a visual improvement of the segmentation map with the addition of CRF. The small black holes inside the tumor area that were not segmented from our basic CAD system, which were the main reason for the addition of a CRF post-processing method, have now been filled. We also have a visual improvement in the negative segmentation example, as seen in figure 4.12d, in which some falsely segmented arachnoid regions have been eliminated. However, the CRF method does not improve the performance drastically since most large missed tumor areas by our basic CAD system are still not segmented in the CRF improved segmentation maps.

Overall, we see that the proposed CAD system with the CPD-R architecture and the addition of the CRF achieves excellent performance.

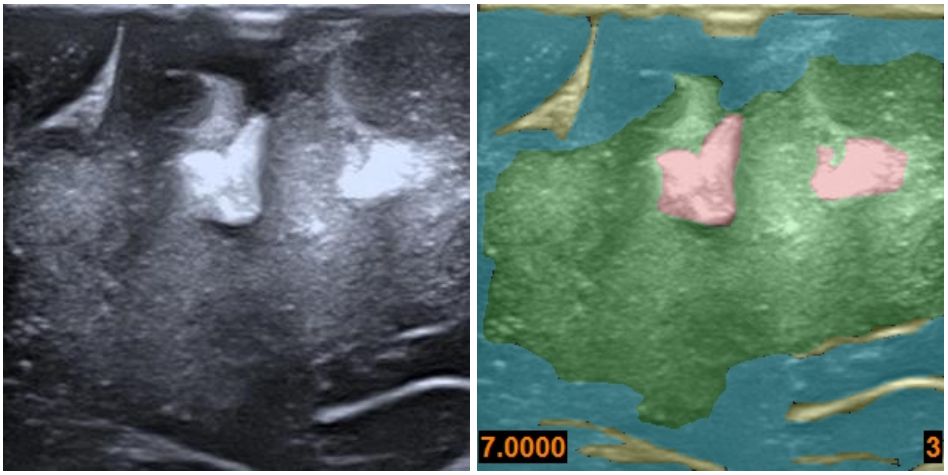
4.4 Interpretability

Despite the good performance, our CAD system lacks interpretability since it is based on a deep CNN for the original segmentation. By this, we mean that we aren't sure what features of an US image are mostly responsible for producing each prediction.

There are several different methods we can use to add interpretability in our system. In (31) the authors propose a method which, given a CNN classifier, a specific image, and a class c for the image, can produce the pixels in the image, which are mostly responsible for the classification of the image with the class c. To achieve this they feed the image in the CNN and then they use the backpropagation algorithm to obtain the gradients, from the first layer of the model, with respect to each pixel in the input image. The authors in (31) explain that the magnitude of the gradient for a pixel is a measure of the importance of this pixel in the classification of the input image as c, and so they use the magnitude of the gradient for every pixel to produce an image which can be seen as an image-specific visualisation of the importance of each pixel in the final prediction c.

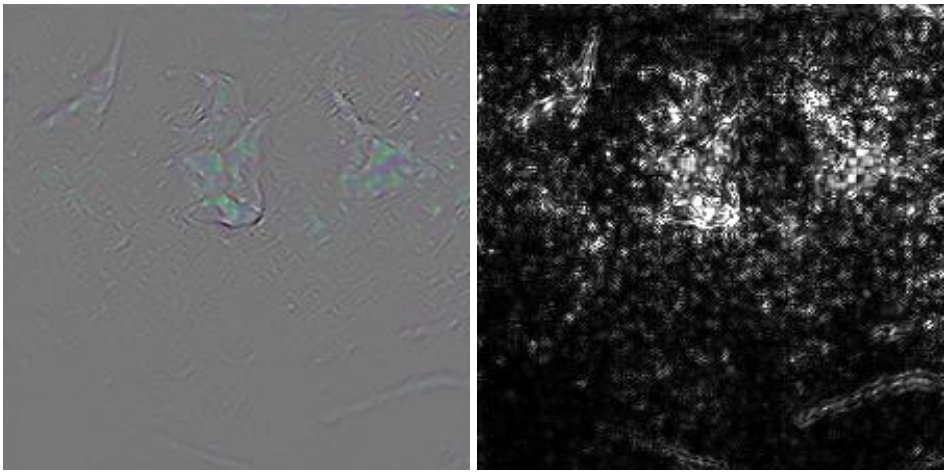
In (32), the authors use a similar technique called Guided Backpropagation (GB) to produce image-specific visualisations of the importance of each pixel in a model which performs polyp segmentation in images. The only difference of the GB algorithm from the vanila backpropagation algorithm is that in the backward pass through a CNN, the GB algorithm removes negative gradients that try to flow through each layer of the CNN. As explained in (32), it has been shown that negative gradients with high magnitude correspond to pixels which the network tries to subdue while positive gradients with high magnitude correspond to pixels which the network considers important. So by using the GB technique we can obtain more clear image-specific visualisations of the importance of each pixel.

We employed the GB method in our model to produce image-specific visualisations of the importance of each pixel in our segmentation maps. The results of the GB visualisation method for the image which was our positive segmentation example (case 7, image 4) in subsections 4.3.2 and 4.3.4 can be seen in 4.13.



(a) Testing Image **(b)** Color mask

(c) Ground Truth Map **(d)** Segmentation Result



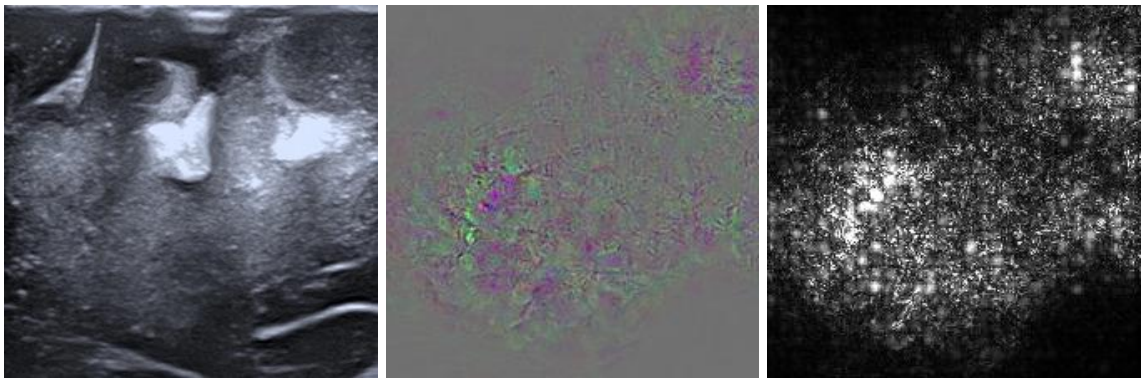
(e) GB Result **(f)** GB Result grayscale

Figure 4.13: Positive segmentation example(case 7, image 4) of the CPD-R. (a) The testing image for which the CPD-R network achieved the best segmentation, (b) the color mask we were provided by our clinical collaborators with different classes segmented, (c) the ground truth map, (d) the segmentation result of the CPD-R, (e) the guided backpropagation result, (f) the guided backpropagation result in grayscale.

As we can see in figures 4.13e, 4.13f the GB method shows that the model considers the more white regions of the input image to be of higher importance. This shows that our model values highly the color intensity information in the input image. As it happens, the really white regions of the input image are more intense areas of the tumor. In the segmentation mask we were provided from our clinical collaborators 4.13b, these areas were highlighted with a light pink color to denote their importance. So our CAD system, focuses more on the prevalent tumor areas of an image, which in turn denotes that the features that the model learns to recognise a tumor are somewhat correct.

Of course, we can see that some arachnoid regions in the input image are also considered somewhat important from our CAD system, despite them not being parts of the tumor. While these regions are not part of the segmentation result of our model, this means that the features learned by our model to recognise tumor regions, while highly correlated to tumors, are not 100% correlated to them. This may be one of the reasons as to why we have some miss-classified areas by our CAD system, as seen in subsections 4.3.2 and 4.3.4, a problem which can be mostly attributed to the really small size of our training dataset.

The image-specific visualisation we got by employing the GB algorithm and visualising the gradients is called a sensitivity map. An alternative way to obtain a more sharp sensitivity map is described in (33). The authors propose a visualisation method called SmoothGrad, which again visualises the gradients of the backpropagation algorithm with respect to an input image, but it does this multiple times (50 times). The input image is changed slightly each time by adding some Gaussian noise so we end up with 50 different sensitivity maps. Finally, SmoothGrad averages the 50 sensitivity maps to produce a clearer sensitivity map. We employed this method as well for the same input image (case 7, image 4) as before and the results can be seen in 4.14.



(a) US Image **(b)** SmoothGrad Result **(c)** Grayscale Result

Figure 4.14: Sensitivity maps produced by SmoothGrad, with respect to the input image (case 7, image 4) which was our positive segmentation example. (a) US image, (b) the SmoothGrad sensitivity map in color, (c) the SmoothGrad sensitivity map in grayscale.

The sensitivity map produced by the SmoothGrad method highlights the whole tumor region and doesn't focus on arachnoid regions or the white intensity of the

tumor, like the previous sensitivity map produced by the Guided Back-propagation method. Instead, the method helps us better interpret our model's learned features about the tumor class. From the sensitivity map of figure 4.14b, we can see that our model has learned to recognize the texture of the whole tumor area and leverages this information to produce segmentation maps.

Chapter 5

Conclusion

5.1 Conclusion

In this thesis, we propose a CAD system that can be used to detect and segment tumor regions in US 2D images in real-time. We experimented with various deep CNN architectures to choose the most appropriate one for our CAD system. The most appropriate one according to our experiments is the CPD-R architecture. We trained this architecture using our dataset of 2D US images of brain tumors, which were provided to us by our clinical collaborators. Given the extremely limited size of our dataset, we had to carefully choose the cases we used in our training dataset, as well as employ various data augmentation techniques to augment our data and construct a large enough training dataset. A post-processing CRF method was included in our CAD system to further enhance the segmentation maps produced by the CPD-R architecture. Our proposed CAD system exhibited high performance achieving an F1-score of 92.40% in our testing dataset.

Finally, we made an effort to understand and interpret the learned features of our CAD system by visualising the most important parts of a US image with tumors, according to our CPD-R architecture. We found out that the color intensity in the US images is highly correlated with the learned features of our CAD system about tumor regions. Moreover, we found out that our model learns to recognise tumor areas by the texture features of the whole tumor area. This is a desirable trait for our CAD system, because it means that the criteria used by the CPD-R architecture to produce its predictions, are correct and meaningful. The success of our CAD system in encoding meaningful features of tumor regions is the main reason why the whole CAD system is producing high quality segmentation maps.

5.2 Future Work

Clearly, the most challenging aspect of creating our CAD system and training it effectively is the limited data in our possession. However, with the emergence of new datasets, we may be able in the future to try our CAD system's performance when it is trained with a larger dataset. Furthermore, our clinical collaborators continue

to provide us with more cases of US brain tumors, so our dataset can be expanded even further.

Another aspect we wish to further investigate is the adaptation of the advantages of CRF in our main backbone and the removal of the CRF post-processing step. This endeavour can be approached in various ways. One way is by expressing the CRF as an RNN and appending this RNN to our main backbone. Then the improved backbone can be trained from scratch in our dataset and hopefully, the new architecture will be capable of producing segmentation maps during inference, on par with those currently produced by our post-processing CRF method. Another way that may work towards eliminating the post-processing CRF is the following. We may use the obtained segmentation maps (for a given testing dataset) of our CAD system with the CRF to augment our training dataset and train the CPD-R architecture further with the new augmented dataset. Hopefully, the resulting architecture will be able to provide improved segmentation maps in a different second dataset without the need for the addition of a post-processing CRF. Both these methods are worth investigating as next steps of this work.

Finally, an interesting next step is to investigate the alteration of the proposed CAD system to provide segmentation maps of other classes, besides tumorous and background pixels, such as the enhancing tumor core and the necrosis parts. Of course, such an endeavour will require a larger training dataset as well.

Bibliography

[1] Unsgaard G, Rygh O, Selbekk T, Müller T, Kolstad F, Lindseth F, et al. Intra-operative 3D ultrasound in neurosurgery. *Acta neurochirurgica*. 2006;148(3):235–253. pages 1

[2] Brown TJ, Brennan MC, Li M, Church EW, Brandmeir NJ, Rakszawski KL, et al. Association of the extent of resection with survival in glioblastoma: a systematic review and meta-analysis. *JAMA oncology*. 2016;2(11):1460–1469. pages 1

[3] Litjens G, Kooi T, Bejnordi BE, Setio AAA, Ciompi F, Ghafoorian M, et al. A survey on deep learning in medical image analysis. *Medical image analysis*. 2017;42:60–88. pages 1, 2, 10

[4] Akkus Z, Galimzianova A, Hoogi A, Rubin DL, Erickson BJ. Deep learning for brain MRI segmentation: state of the art and future directions. *Journal of digital imaging*. 2017;30(4):449–459. pages 1, 9, 10, 11

[5] Bakas S, Reyes M, Jakab A, Bauer S, Rempfler M, Crimi A, et al.. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge; *Arxiv*. [Preprint] 2018. Available from: `https://arxiv.org/abs/1811.02629`. pages 2

[6] Xie S, Tu Z. Holistically-nested edge detection. In: *Proceedings of the IEEE international conference on computer vision*; 2015. p. 1395–1403. pages 7

[7] Hou Q, Cheng MM, Hu X, Borji A, Tu Z, Torr PH. Deeply supervised salient object detection with short connections. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2017. p. 3203–3212. pages 7, 8

[8] Chen S, Tan X, Wang B, Lu H, Hu X, Fu Y. Reverse Attention-Based Residual Network for Salient Object Detection. *IEEE Transactions on Image Processing*. 2020;29:3763–3776. pages 8, 9, 17, 18, 19

[9] Zhao JX, Liu JJ, Fan DP, Cao Y, Yang J, Cheng MM. EGNet: Edge guidance network for salient object detection. In: *Proceedings of the IEEE International Conference on Computer Vision*; 2019. p. 8779–8788. pages 8, 9

[10] Wang L, Wang L, Lu H, Zhang P, Ruan X. Salient object detection with recurrent fully convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*. 2018;41(7):1734–1746. pages 8

[11] Kamnitsas K, Ledig C, Newcombe VF, Simpson JP, Kane AD, Menon DK, et al. Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. *Medical image analysis*. 2017;36:61–78. pages 9, 10

[12] Kamnitsas K, Ferrante E, Parisot S, Ledig C, Nori AV, Criminisi A, et al. DeepMedic for brain tumor segmentation. In: *International workshop on Brainlesion: Glioma, multiple sclerosis, stroke and traumatic brain injuries*. Springer; 2016. p. 138–149. pages 9

[13] Myronenko A. 3D MRI brain tumor segmentation using autoencoder regularization. In: *International MICCAI Brainlesion Workshop*. Springer; 2018. p. 311–320. pages 10, 11

[14] Zhou C, Ding C, Wang X, Lu Z, Tao D. One-pass multi-task networks with cross-task guided attention for brain tumor segmentation. *IEEE Transactions on Image Processing*. 2020;29:4516–4529. pages 10, 11

[15] Kamnitsas K, Bai W, Ferrante E, McDonagh S, Sinclair M, Pawlowski N, et al. Ensembles of multiple models and architectures for robust brain tumour segmentation. In: *International MICCAI Brainlesion Workshop*. Springer; 2017. p. 450–462. pages 11

[16] Roy AG, Siddiqui S, Pölsterl S, Navab N, Wachinger C. 'Squeeze & excite'guided few-shot segmentation of volumetric images. *Medical image analysis*. 2020;59:101587. pages 11

[17] Carton FX, Noble JH, Chabanas M. Automatic segmentation of brain tumor resections in intraoperative ultrasound images. In: *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*. vol. 10951. International Society for Optics and Photonics; 2019. p. 109510U. pages 11

[18] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer; 2015. p. 234–241. pages 11

[19] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition; *Arxiv*. [Preprint] 2014. Available from: `https://arxiv.org/abs/1409.1556`. pages 14

[20] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*; 2016. p. 770–778. pages 14, 15, 16

[21] Wu Z, Su L, Huang Q. Cascaded partial decoder for fast and accurate salient object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2019. p. 3907–3916. pages 19, 20, 21, 33

[22] Liu S, Huang D, et al. Receptive field block net for accurate and fast object detection. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 385–400. pages 20

[23] Wei J, Wang S, Huang Q. F3Net: Fusion, feedback and focus for salient object detection; *Arxiv*. [Preprint] 2019. Available from: `https://arxiv.org/abs/1911.11445`. pages 22, 23, 26

[24] Lafferty JD, McCallum A, Pereira FCN. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the eighteenth international conference on machine learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2001. p. 282–289. pages 24

[25] Krähenbühl P, Koltun V. Efficient inference in fully connected crfs with gaussian edge potentials. In: *Advances in neural information processing systems*; 2011. p. 109–117. pages 24, 25, 37

[26] Zheng S, Jayasumana S, Romera-Paredes B, Vineet V, Su Z, Du D, et al. Conditional random fields as recurrent neural networks. In: *Proceedings of the IEEE international conference on computer vision*; 2015. p. 1529–1537. pages 25

[27] Borji A, Cheng MM, Jiang H, Li J. Salient object detection: A benchmark. *IEEE transactions on image processing*. 2015;24(12):5706–5722. pages 27

[28] Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Garcia-Rodriguez J. A review on deep learning techniques applied to semantic segmentation; *Arxiv*. [Preprint] 2017. Available from: `https://arxiv.org/abs/1704.06857`. pages 28

[29] Csurka G, Larlus D, Perronnin F, Meylan F. What is a good evaluation measure for semantic segmentation?. In: *BMVC*. vol. 27; 2013. p. 2013. pages 28

[30] Perez L, Wang J. The effectiveness of data augmentation in image classification using deep learning; *Arxiv*. [Preprint] 2017. Available from: `https://arxiv.org/abs/1712.04621`. pages 30

[31] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps; *Arxiv*. [Preprint] 2013. Available from: `https://arxiv.org/abs/1312.6034`. pages 41

[32] Wickstrøm K, Kampffmeyer M, Jenssen R. Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation. In: *2018 IEEE 28th international workshop on machine learning for signal processing (MLSP)*. IEEE; 2018. p. 1–6. pages 41

[33] Smilkov D, Thorat N, Kim B, Viégas F, Wattenberg M. Smoothgrad: removing noise by adding noise; *Arxiv*. [Preprint] 2017. Available from: `https://arxiv.org/abs/1706.03825`. pages 43