University of BRISTOL

DEPARTMENT OF ENGINEERING MATHEMATICS

# GI-Tract Image Segmentation using Deep Learning U-Net Architecture

## Multi-class Semantic Segmentation

Yashwanth Jayaramaiah

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Wednesday 21st September, 2022

Supervisor: Prof. Peter Flach

# Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Yashwanth Jayaramaiah, Wednesday 21st September, 2022

# Contents

# List of Figures

# List of Tables

# Abstract

Over the past twenty years, artificial intelligence has advanced dramatically along with Deep learning developed by using algorithms to create an artificial neural network. The task of classification of an image involves classifying it into an array of categories. The image segmentation model is built under deep learning using U-net Architecture in this thesis research.
Medical images are segmented at the pixel level using multi-class semantic segmentation. The problem statement chosen for this research is from a Kaggle competition; anonymous MRI scans are used to track healthy organs to improve cancer treatment. As a pioneer in MR-Linac-based radiotherapy at UW-Madison Carbone Cancer Center, the hospital has treated patients using MRI guidance since 2015. Using MRI scans, cancerous tumours are treated with radiotherapy. To avoid the stomach and intestines, the X-ray beam must be angled to deliver the dose to the tumour while avoiding the organs. Oncologists must manually frame the tumour's position and treat it after manually segmenting it through MRI scans.

This therapy lasts 10 - 15 minutes since the tumour is between the stomach and intestines. Therefore, patients have to endure hours of pain at a time to accomplish this task. In addition to accelerating treatment, the segmented organs make it easier for doctors to find tumours and treat a significantly higher number of patients. As a result, patients can receive more effective treatment.
A U-net model architecture is used to solve this problem and is built based on a few metrics (Dice coefficient, Harsdorf distance (3D model), and Jaccard Index). The dice coefficient and Jaccard index were used to build an efficient 2D model. A Convoluted Neural Network (CNN) under U-net architecture is being tested in this thesis research to determine which model performs best. Extra functionalities were added, such as simple data augmentation on the dataset and training the model using Keras (TensorFlow). The data is modelled using pre-trained models to compare the performance of image segmentation models VGG16, Resnet34, and ResNet50 architectures using cross-entropy loss, dice coefficient, and Jaccard index (IOU). However, all three of these models solve the same image classification problem. We have concluded that ResNet50 is the best architecture based on the comparison. For VGG16, Resnet34, and ResNet50 at epoch 50, these models produced training mean IOU scores of 0.791, 0.790, and 0.80 with training dice scores of 0.860,0.867, and 0.871.

- I spent 100 hours in studying the required deep learning concepts regarding the project.
- I spent 100 hours on literature review and analyzing few machine learning blogs.
- I spend 100 hours on understanding the dataset by performing exploratory data analysis.
- I spent 200 hours on building a pre-trained model and performing experiments using hyperparameters.
- I spent 100 hours on writing the thesis report.

# Supporting Technologies

The various technologies used in my dissertation project is listed in this section.

- I have used latest version python to build its functionalities along with some python libraries such as *Pandas*,*NumPy*, *Seaborn* and *TensorFlow* including many more public-domian Python Libraries. All the libraries used for the project is present in the code through the GitHub link.
- I have used pre-trained models from the GitHub.
- I have used Google Colab for data analysis and kaggle website notebook for building deep learning models.
- I used LaTeX to format my thesis, via the online service *Overleaf*.

# Notation and Acronyms

**The notations and acronyms used in the dissertation document is shown in this section.**

| | | |
|---|---|---|
| GI Tract | : | Gastrointestinal Tract |
| MRI-LINAC | : | Magnetic Resonance Imaging Guided Linear Accelerator |
| CT | : | Computed Tomography |
| MRI | : | Magnetic Resonance Imaging |
| RLE | : | Run-Length Encoding |
| EDA | : | Exploratory Data Analysis |
| FCN | : | Fully Convolutional Network |
| GPU | : | Graphics Processing Unit |
| CPU | : | Central Processing Unit |
| TPU | : | Tensor Processing Unit |
| TP | : | True Positive |
| FP | : | False Positive |
| FN | : | False Negative |
| PNG | : | portable network graphics |
| VGG | : | Visual Geometry Group |
| ResNet | : | Residual Network |
| CNN | : | Convolutional neural network |
| IoU | : | Intersection over union |

# Acknowledgements

I would like to thank my friends and family who have extended their support, and my supervisor for his consistent guidance during this dissertation project.

# Ethics

”This project did not require ethical review, as determined by my supervisor, Prof. Peter Flach”

# Chapter 1

# Introduction

**This chapter provides a rationale for research in this field, followed by an explanation of the motivation for the thesis. The purpose of this section is to introduce the project and its application. An overview of the project's aims, key objectives, and achievements follows.**

The number of people diagnosed with GI tract cancer in 2019 was approximately five million. This type of cancer occurs between the stomach and intestines. Several tests are used to diagnose GI tract cancer using medical imaging. The medical imaging techniques are colonoscopies, endoscopies, and biopsies. The most common type of scan used is an MRI. Using radiotherapy, oncologists remove these cancerous cells. The DNA of cancer cells gets damaged in radiation therapy, which kills or slows them down. It is well known that radiation therapy kills cancer cells and shrinks tumours when a high dose of radiation is used. Radiation is directed at a tumour during X-ray imaging to prevent it from spreading to other body parts. Furthermore, radiotherapy can damage some healthy cells in the treated area and kill cancer cells. By manually framing the tumour on the basis of the MRI scans, which is labour-intensive and takes much time, the doctor can avoid this problem. Nonetheless, healthy tissues are at risk during this process. The solution to this problem is Image segmentation, which is a process of segmenting the image into multiple segments and identifying each pixel that belongs to a specific class. As a result of the need for medical advancements in segmenting organs, semantic segmentation was developed; for instance, brain segmentation while performing surgery is challenging and time-consuming for doctors [32]. But using semantic segmentation on scanned images, the segmentation of delicate areas will be straightforward and give effective treatment to patients, thus saving a lot of time [32]. There are several enhancements and improvements in the medical treatment of cancer patients from image segmentation [44]. From 2000 to 2016, there was a rapid surge in the usage of medical imaging in the United States [44][35].

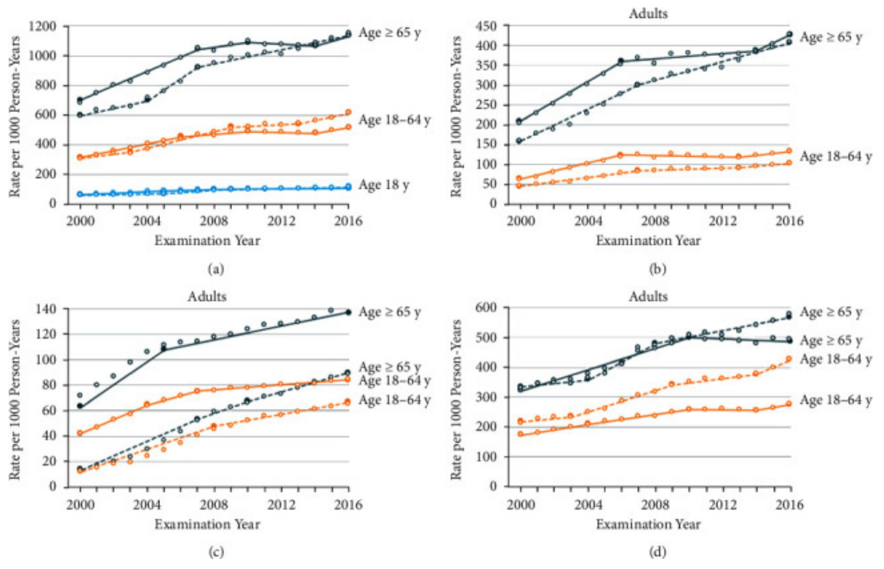


Figure 1.1: Representation of different medical imaging of people of different ages from 2000 to 2016 [44].

The above figure represents a retrospective cohort study of patterns of medical imaging (a) represents CT, MRI and Ultrasound examinations of 16 to 21 million patients undergoing medical imaging with respective ages, (b) Represents CT imaging analysis, (c) represents MRI analysis and (d) represents Ultrasound imaging analysis between the year 2000 to 2016 [35]. In recent years, CT, MRI, and ultrasound use have increased at a slower rate among adults [44].

## 1.1 Source of the problem

The research is a Kaggle competition supported by the UW-Madison Carbone Cancer Center. Since 2015 this centre has been treating patients with cancer through MRI-guided therapy [1]. An anonymous MRI scan of GI-tract cancer patients is made available in PNG format, a 16-bit grayscale image for further research.

### 1.1.1 The Actual Problem to be solved

MRI-LINACs (Magnetic Resonance Imaging Guided Linear Accelerators) are radiotherapy devices that target x-ray beams at tumours to eradicate them [16]. MRI scans are used to treat cancerous tumours with radiotherapy. The Gastro-Intestinal tract tumour is located between the stomach and the intestines (Large bowel and Small bowel). It regularly moves within the GI-Tract. A magnetic resonance imaging scan is the most effective method for determining the location of a tumour. Oncologists treat this type of cancer via radiotherapy, figuring out the tumour's location with an MRI scan and administering the necessary doses of radiation according to its location. In order to avoid damaging healthy organs such as the stomach and intestines, the X-ray beam must be angled more precisely to the tumour's location. As a result of the location of the tumour, the patient's MRI scans have been recorded over a period of six weeks. The tumour's position changes daily; therefore, it is not recommended that only one MRI scan be used for segmentation. In light of this behaviour, there will be multiple scans over six weeks.

In most cases, the treatment lasts between ten and fifteen minutes following the oncologist's manual segmentation of parts in the MRI scan. The oncologist must manually frame and treat the tumour because it is located between the stomach and intestine. Manual segmentation by doctors is a labour-intensive process that takes hours to complete. As a result, patients experience pain during the radiotherapy treatment process and must endure this pain until the treatment is complete. Radiotherapy can last for hours, which is not easy on the patient [1].

As this is a Kaggle competition, the goal of this competition is to create a deep learning medical image segmentation model that can automatically segment the stomach and intestines(large bowel and small bowel) using medical MRI scans. The evaluation of this competition is mainly based on two metrics which are the "Dice coefficient" and "Hausdorff distance". Kaggle's leadership table classifies winners primarily based on their models' dice coefficient scores. Based on a dataset of these scans, the dataset was preprocessed using python libraries and efficiently segmenting models were built with 15 and 50 epochs and compared between them. In this experiment, pre-trained models were used. The models VGG-16, ResNet-34 and ResNet-50 were found to be the better performing models and also the best to compare between them [20]. The model was built using the metrics Dice Coefficient, Jaccard index (IOU) and cross-entropy (loss function).

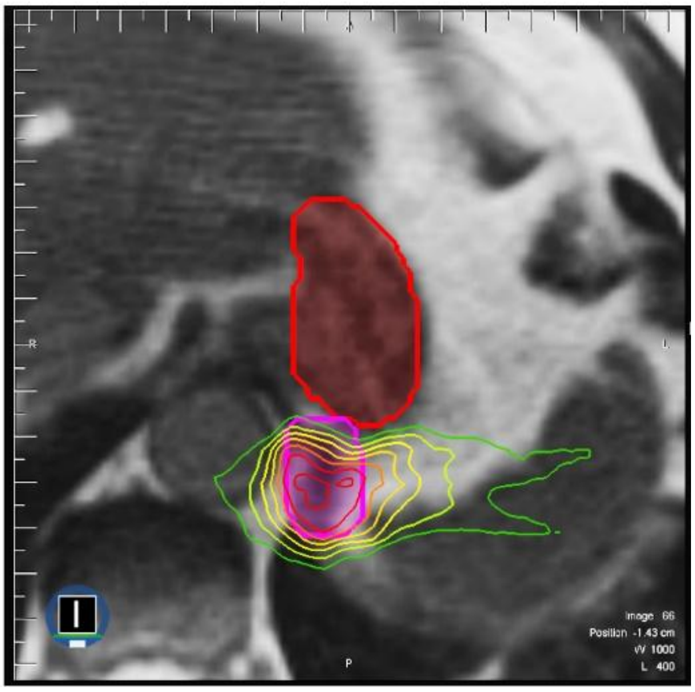


Figure 1.2: In this figure, A rainbow of outlines is shown, which represents the radiation power more on the red and less radiation given by green. Tumour is present exactly next to the stomach [1].

### 1.1.2 Radiation delivery and segmented boundaries

The data of MRI scans are anonymous patient data with GI tract cancer. These Scans are in the PNG format of a 16-bit grayscale image. These are the training dataset containing annotations of RLE-encoded masks along with different classes (large bowel – lb, small bowel – sb and stomach – st) used for modelling to perform the semantic segmentation later. Each PNG image is categorized into distinct categories based on the slice height and width, which define the resolution and pixel spacing of height and width, representing each pixel's physical size. From a broader scale, the data is classified under different cases, with Scans taken on other days with annotations from a single patient referring each case to an individual patient.

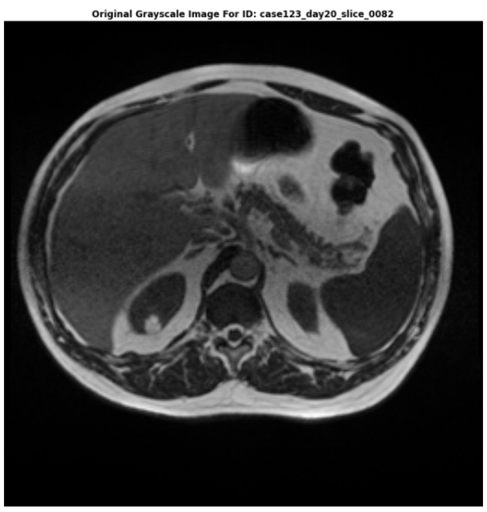


Figure 1.3: A sample MRI scan of a patient with GI tract cancer from the Training dataset [1].

## 1.2 Applications of Sematic segmentation

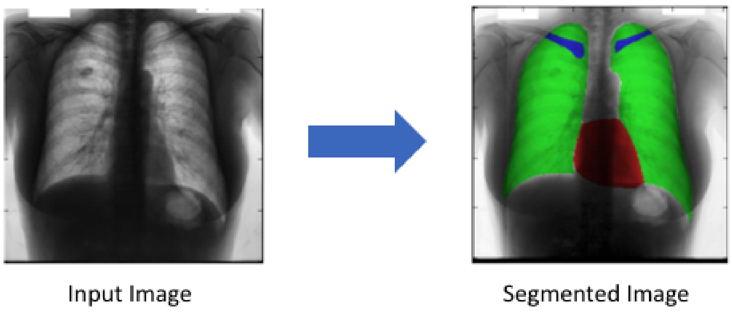


Figure 1.4: Semantic segmentation of lung, referenced from [25]

### 1.2.1 Medical image segmentation

Machine learning image segmentation can detect medical abnormalities in CT or MRI scans. Doctors can manage their time more effectively by using computer vision algorithms that are fast and accurate; thus, each scan need not be analysed as it gets done manually and saves time.

### 1.2.2 Facial Recognition

As part of semantic segmentation of faces, skin, hair, and background categories may be considered, as well as categories such as eyes, noses, and mouths. To differentiate ethnicity, age, and expression, computer vision applications can be trained by segmenting facial features in a detailed manner.

### 1.2.3 Self-Driving Cars

Computer vision has been applied to autonomous vehicles for a long time, but they are among the most complex applications to date. In addition to lowering performance, poorly-trained algorithms can endanger pedestrians and other vehicles on the road. Artificial intelligence companies rely on semantic segmentation to maximise accuracy when detecting lanes and traffic signs.

## 1.3 Challenges

A review of the current challenges of medical image segmentation is presented in this section, making it unavoidable that U-Net-based deep learning approaches must be improved and innovated to meet these challenges.

### 1.3.1 Dataset pre-processing

Generally, multiple cells and organs cannot be distinguished in an image [43]. For disease diagnosis, medical image processing requires exceptionally high accuracy [45][42]. It is necessary to pre-process the datasets in a data frame format for the model to be easily trained, even though the datasets are open-source data available on the public platform. Furthermore, the image data is pre-processed, the relevant network is built, and it continues to run by adjusting the parameters even when an appropriate deep learning model reaches a certain level of accuracy [31].

### 1.3.2 Dataset Format

CT/MRI machines have a varying range of standards for annotations. Therefore, the training models associated with deep learning are only suitable for a few scenarios like U-net. While with the weak generalization of neural networks in analyzing medical images is possible for it to capture the wrong features. The format in which the scans are available with proper RLE encoded masks annotated is essential. The dataset used is in the form of 16-bit grayscale PNG format. In addition, negative samples are always smaller than positive samples, which may affect segmentation more. Nevertheless, U-Net can achieve better performance in reducing overfitting [28].

### 1.3.3 Model Learning requirements

There are several domain-specific unexpected challenges based on the different libraries we use. While performing semantic segmentation demands greater CPU power for better prediction and faster learning. Using libraries like TensorFlow, Keras and PyTorch demand GPU for parallel processing, which gives better results. CPUs perform much worse than GPUs when optimizing image processing solutions. In GPU architecture, image pixels can be processed in parallel, which reduces the processing time (latency) for each image. Hyperparameter selection complements this process and enhances generalization [22].

### 1.3.4 Need for Pre-trained models

It is much easier to consider when evaluating transfer learning since the models are pre-trained to extract features more accurately and perform very well on prediction tests. Developing the model from scratch is unnecessary since the pre-trained models have already been trained with a large amount of data to perform segmentation in a particular field. We can also expect higher generalization when there is a lack of labelled data for supervised model learning [33][46].

### 1.3.5 Need for Interpretable Models

Medical images require interpretable deep learning models but lack confidence in their predicted results [31, 32]. The U-Net is a CNN with poor interpretability. It is possible to diagnose a disease accurately by segmenting images in medical imaging that reflect the patient's physiological condition. Professional doctors are unlikely to trust and recognize segmentation that lacks interpretability and confidence for clinical application. The diagnosis of diseases relies primarily on images, although other supplements have also been added, increasing the complexity of the process. As a result of perceiving and adjusting these trade-offs, achieving interpretability and confidence in medical image segmentation can be challenging. It is possible to improve the flow of treatment, segment tumours more quickly, accelerate treatment, and treat a more significant number of patients with deep learning. Recent years have seen an increase in the use of deep learning technologies for medical image analysis, and such applications as segmentation and registration have shown excellent results. Several mathematical algorithms and edge detection filters are used in the classical image segmentation method. The U-Net comprises both unsampled features from the encoder network and unsampled features to help improve the learning of representations through convolutions.

Several differences exist between the U-Net architecture and the CNN (Convolutional Neural Networks) architecture. Furthermore, this technique can identify the location of the infection in addition to identifying biomedical images and distinguishing them from infectious tissues. After convolution blocks encode the input image into multiple feature representations at various levels, followed by a max pool downsampling, the U-Net takes over in the second half as the encoder.

Using this segmentation method, you can identify where different classes of objects exist in the image and generate masks or matrices containing elements that describe the class or instance of objects. Several relevant heuristics or high-level image features can assist in segmenting images. In this way, patients will be able to receive more effective treatment. Using a deep learning model, we can automatically identify tumours on MRI images from stomachs and intestines[34]. As a result of the labour-intensive and time-consuming nature of this method, the treatment time can be easily extended from 15 minutes to an hour without deep learning methods. Through deep learning, this process can be accelerated and will provide patients with more effective treatment through the segmentation of images.

In this study, the data used are the patients' X-ray scans, each saved as 16-bit grayscale PNG files. Models are constructed using three layers: a convolutional layer, a pooling layer, and a fully connected layer. To create segmentation masks, U-Net uses deep learning convolutional networks. Using this method, radiation oncologists can deliver higher doses of radiation to tumours without causing harm to the stomach and intestines. As a result, cancer patients will be able to receive more effective treatments with fewer side effects and a better chance of surviving the disease in the long run. Kaggle provides free access to the data used in this project [1].

## 1.4 Focused Aim

This project aims to perform automatic semantic segmentation of scanned medical images of patients suffering from GI tract cancer. The multi-class image segmentation is more appropriate since the model should predict three different classes (stomach, large bowel, and small bowel) it should mask based on the RLE annotations. To achieve this, we must create a deep learning model which utilizes FCN to perform the task. Using a pre-trained U-Net model rather than a deep learning model built from scratch as the pre-trained model performs well. It has been found that U-net architecture models perform well in segmenting medical images among many architectures [44]. Building U-net models for semantic segmentation allows oncologists to provide effective treatment to their patients and to treat a large number of them in a short period.

## 1.5 Objectives

Here are some proposed systematic objectives required to perform to achieve the final aim.

- Extracting the required GI-tract segmentation dataset from the Kaggle website.
- Selection and training of various segmentation models using different hyperparameters.
- Selection and analysis of the best model post-training.
- Results compared to the competition's winning model.

## 1.6 Achievements

The goal was achieved by following the objectives. Since GPUs execute 800 times more instructions per clock than CPUs, the model was trained using GPUs. A total of three U-Net pre-trained models were trained, VGG16, Resnet34, and Resnet50. Comparison of results with the winning model of the competition. The model was built using several hyperparameters, using an Adam optimizer, sigmoid activation function at the output layer. Medical images, which are MRI scans available in PNG format, were used to train the three models over 15 and 50 epochs with a batch size of 32. In comparison to other models, the Resnet50 model performed well. In the Kaggle competition, first place was determined using the dice score, which was 0.892 but achieved a dice score of 0.871 using the model ResNet-50.

## 1.7 Important links

As part of this dissertation document, I would like to provide four links.

- The following is a one-drive link that contains the presentation of my project from start to finish, which lasts less than 10 minutes.Video Link
- This is a GitHub repository link that contains the code for this project. Link to GitHub
- The following link provides access to the dataset used from the Kaggle website. Link to dataset
- A GitHub link to the repository where the pre-trained models were installed and modelled can be found here. Link to GitHub

# Chapter 2

# Background

**This section contains continued literature from the previous section with more detailed and technically required background explanations.**

## 2.1 CNN (Convolutional neural network)

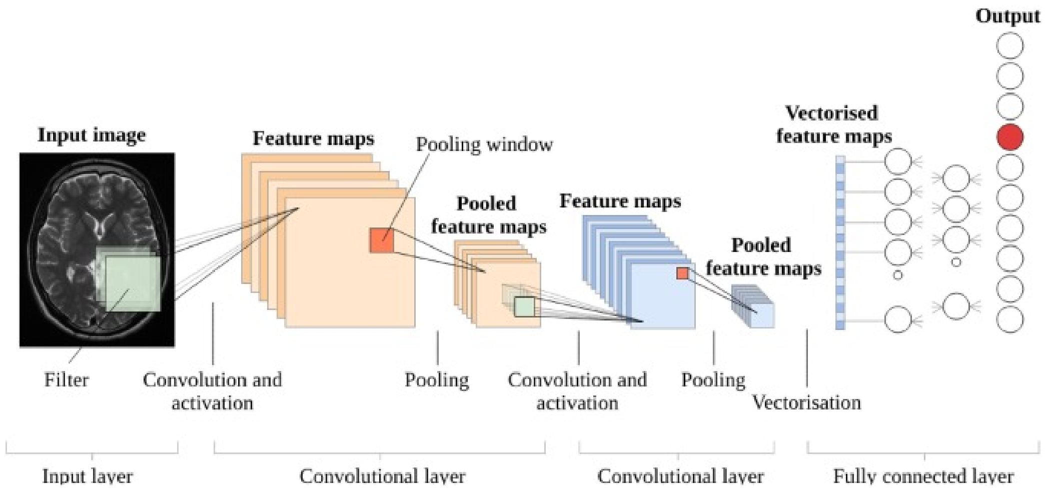


Figure 2.1: A simple representation of the building blocks of the CNN model performing image processing [18].

The figure 2.1 represents the process of feature extraction from the medical image and later max polling for giving the inputs to the input layers as vectorized feature maps. Thereon the hidden layer in the network learns from the data by forming a convolutional layer and creating a fully connected neural network. In the end, using the activation function, the images are predicted to classify organs [18].
The subsequent research focused on implementing CNN (Convolutional Neural Network) models and the type of model to be used once the method was understood. The paper [14][30][6] illustrates how U-Net architectures are used by many medical imaging researchers to obtain excellent results through the instance and segmentation process. Convolutional, pooling, and fully connected layers are used to construct models.

## 2.2 Image Segmentation

Analysing and processing images requires image segmentation. Analysis of segmented images has implications for subsequent image analysis tasks, such as describing and representing objects, measuring features, and determining their classification. Image segmentation is the most essential and crucial step in delineating, characterising, and visualising regions of interest in any medical image [4]. The collection of measurements in two-dimensional (2-D) or three-dimensional (3-D) space is an image. It is possible to measure radiation absorption using X-rays, acoustic pressure using ultrasounds, or radio frequency signals using MRIs. Images that have only one measurement at each location are called scalar images. The term vector or multi-channel image refers to images created with more than one measurement called dual-echo MRI. For example, an X-ray film or an MRI may acquire images in a discrete space or in a continuous domain. Images with discrete measurements in 2-D are called pixels, and those with 3-D measurements are called voxels[27]. A mask gets generated using background image data when segmenting CT, MRI, and other scans. It is possible to perform segmentation on scans in two or three dimensions, depending on the user's preference. As a result of segmenting medical images, it is possible to analyze anatomical data more accurately by excluding unnecessary areas. Segmentation also allows for separating soft tissues from bone tissues and removing any unwanted details from scans, including noise and background .

## 2.3 Why U-Net?

In U-Net, input images are segmented using deep learning convolutional networks. It has been found that the ResNet-34 architecture was found to have superior performance in terms of producing semantically accurate predictions in GI tract tumour detection when compared to ResUNet++, Basic U-Net and ResNet(U-Net) architectures [34][14][30]. A preprocessing 16-bit grayscale scan involves filtering and normalizing the images so that the model can learn precisely what to segment according to the predicted pattern [11]. Usually, models are constructed using convolution, pooling, and fully connected layers. U-Net creates segmentation masks from input images using deep learning convolutional networks [19][5]. Encoders and decoders play an important role in the U-Net architecture. Several encoders exist depending on the application's encoding needs: EfficientNet, MobileNet, VGG16, and ResNet. Based on U-Net architecture, these networks have 1:3 input-output channel ratio [9][36][29][10].

By converting X-ray images with specific dimensions into densely encoded formats, the decoder will later convert the dense encoded images back into the segmented output images [30]. Three different output channels determine the availability of different segmented masks. Each output is a mask representing the respective classes depending on the channel. The stomach, large bowel, and small bowel are the three classes in our case. The U-Net architecture encodes different pixels for each of these classes. To make the output more efficient, downsampling and upsampling are used to reconstruct the segmented mask [30][9][29]. The intersection over union(IoU) and Dice coefficient metrics can be used to optimize the training of the deep learning model after 50 epochs, leading to better results [48].

Using the UNET architecture, four encoders and four decoders are connected through a bridge connected by four encoders. A double number of filters (feature channels) at each encoder block reduces the spatial dimension and multiplies the number of encoder blocks (contracting paths). U-Net facilitates semantic segmentation. There are two paths, one of which contracts and one of which expands, called the encoder and the decoder. As a convolutional network contract, its architecture follows a typical pattern [30]. Thousands of annotated training samples are required to train deep networks successfully. Annotated samples augment this paper's network and training strategy to improve training efficiency. An expanding symmetric path enables precise localization while a contracting path captures context. A sliding window convolutional network outperforms the best method for training end-to-end from very few images.

## 2.4 UNET — Network Architecture

A U-NET architecture consists of four encoders and four decoders connected via a bridge in a U-shape. A network of encoders (contracting paths) will have the capacity of half the number of channels at each encoder block and double the number of filters which is the feature channels. The spatial dimensions of decoding networks are doubled, and the number of feature channels is halved. Blue boxes represent multi-channel feature maps. On the top of the box is a count of all channels. In the lower-left corner of the box, the x-y-size is displayed. Copies of feature maps are represented by white boxes [30]. From the figure 2.2 in order of importance, the architecture can be divided into four components:

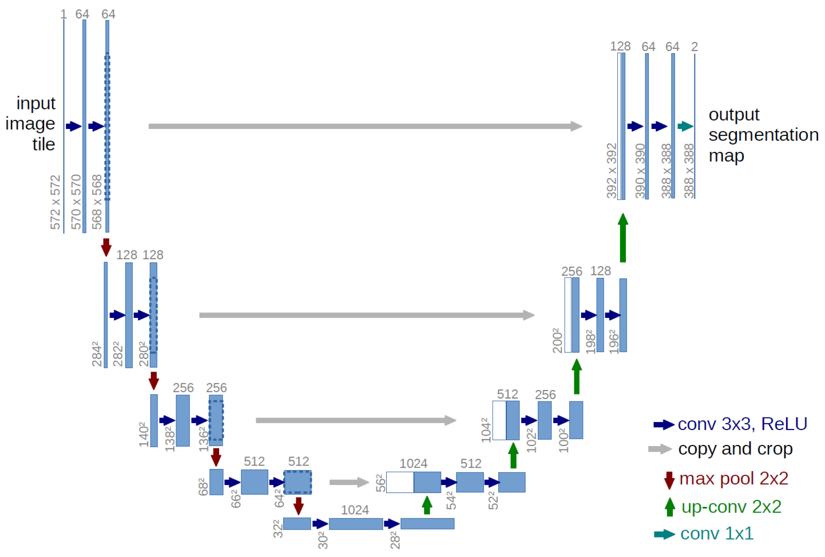


Figure 2.2: This figure represents a basic U-net architecture. The directions indicate the different unique operations performed in the architecture. It is referenced in the paper [30].

### 2.4.1 Encoder Network

In encoder networks, features are extracted from an input image and transformed into abstract representations using encoder blocks. Activation functions for each ReLU (Rectified Linear Unit) are followed by convolutions of 3x3. Through the ReLU activation function, we introduce nonlinearity to the network, allowing the training data to be generalized better. A skip connection is made between the output of the ReLU and the corresponding decoder block [30]. Following that comes a 2x2 max pooling of feature maps, which reduces their spatial dimensions by half[47]. The number of trainable parameters is reduced, reducing the computational cost.

### 2.4.2 Skip connections block

As a result of these skip connections, a decoder can generate more accurate semantic features. Also, they act as shortcuts connecting earlier layers without degrading gradients[39]. Backpropagation is improved with skip connections, allowing the network to learn better representation through gradient flows.

### 2.4.3 Connector bridge

In each convolution, a ReLU activation function follows a 3x3 convolution. It fills in the gaps between the encoder and decoder blocker, functioning as a connector bridge between the left and right blocks [41].

### 2.4.4 Decoder Network

The decoder network converts an abstract representation into a semantic segmentation mask. Decoder blocks use 2x2 transpose convolutions as an initial step. IA feature map for the skip connection is concatenated with this feature map within the encoder block. The network depth enables these skip connections to provide features from earlier layers. A ReLU function is then used to activate each 3x3 convolution. Finally, the output of the last decoder is convolutional 1x1 with sigmoid activation. Based on sigmoid activation functions, segmentation masks represent pixel-wise classification [30].

## 2.5 Training the U-net model with medical images

Unpadded convolutions result in a smaller output image than the input. As a result, we reduce the batch size to a single image, which minimizes overhead and GPU memory utilization . Multiple layers of CNN require proper weight initialization. Alternatively, some parts of a network might produce excessive activations while others do not. In a network, each feature map should have approximately a unit variance based on the initial weights [30][8]. The computation of the weight map is given by :

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \tag{2.1}$$

The above mathematical equation 2.1 represents the calculations for the computing weight map based on the classes [30]
where $w_c$: $\Omega \to$ R is the weight map, d1: $\Omega \to$ R denotes the distance of nearest classes and d2: $\Omega \to$ R is for as many classes present for another segment.

## 2.6 Data Augmentation

Data augmentation is a popular technique used to enhance deep neural networks' generalization capabilities, which can be viewed as indirect regularization. It is essential to have ground-truth data when there is a shortage of high-quality actual data, and obtaining new examples is time-consuming. The problem of tumour delineation is prevalent in medical image analysis [23]. In the absence of annotated masks for the particular case, the training data can be increased more appropriately[40].

There is often a lack of labelled training data for medical image segmentation. With 'data augmentation', the network is more likely to perform well on data outside the training set, as it prevents memorizing training data. Training sets can be significantly increased by using data augmentation. The MRI data of patients with GI-tract cancer were augmented using Python library preprocessing via Keras. Besides the horizontal and vertical flips, rotation range 50, zoom range 0.5, height shift range and wide shift range 0.2 were also used for the augmentation.

## 2.7 Need for Transfer learning using pre-trained models

In transfer learning, a model built using a large amount of training data to perform specific can later be reused based on the input. In the Pre-trained Model Approach, a source model is selected that has been previously trained. In addition to being able to perform a prediction task, this model can also be applied to other tasks. Depending on the modelling technique, the model must be tuned for the task of interest based on the input-output pair data [24]. The benefit of transfer learning is more significant optimization because it saves time and improves model performance at the same time. From figure 2.3, In general, transfer learning is not evident until the model has been developed and evaluated that it will be beneficial. A high start, a higher slope, and a higher asymptote of the converged skill of the trained model are three possible benefits that Lisa Torrey and Jude Shavlik describe in their paper [26].

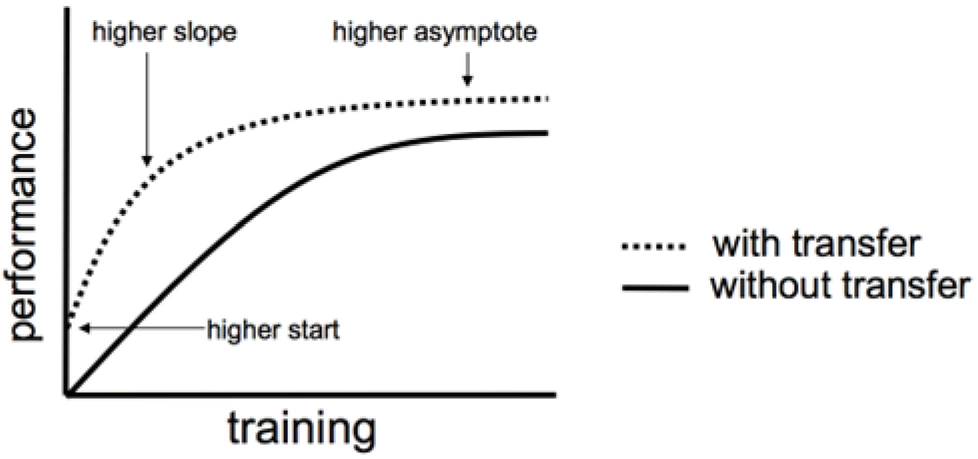


Figure 2.3: Figure 6: Three ways in which transfer might improve learning [26].

When there is a lack of data available, transfer learning supports building skilled models that would not have been possible without transfer learning [3].

Chapter 3

# Execution

**This Section explains about the detailed execution of the project.**

A few functionalities of the code are referenced from the online sites, and the entire project code is written in Python code. At the beginning of the project, executing the code on a Jupyter notebook took a long time because of the CPU power limitations. Later, the code was built and executed using the Kaggle notebook environment. Kaggle notebooks offer a variety of computational resources, such as CPU(None), GPU, and TPU. TPUs are mainly used for training large neural networks, which perform 15-30 times faster than GPUs. In order to run the code, GPUs were sufficient.

## 3.1 Overview of Dataset

This project aims to construct a model that can segment different organs in MRI images based on different classes. The datasets are MRI images in a grayscale format in PNG format that belongs to patients with GI tract cancer. The oncologist can more effectively treat cancer patients by segmenting the image into different classes. RLE-encoded masks are provided in the training data. These medical images are divided into cases which are around 50 to 60 cases. These scans are represented by multiple scanned annotated slices containing RLE encoded masks. In each case, the dataset is collected over a period of one to six weeks. Slices are made with a minimum of 80 to 144 slices for each scan. A part of the training set is also available for testing and validation.

## 3.2 RLE-Encoded Pixel masks

During segmentation, run-length encoding is used to encode foreground objects' locations. The mask is produced by a list of pixels. A U-net model is then built, which tries to learn which pixels are classified in each image. A pixel in the image is classified into a class based on ground truth from training data, supervised by masks in the training dataset. The RLE-Encoded segmentation values are present for each class under the segmentation column in the training set. The RLE is later encoded, and RGB mapping is applied to the respective organ defined with pixel values [21]. Initially, the training data which is in the CSV format is stored in the data frame. Later it is divided into training and testing sets. Only a few samples (480) were taken as the testing set. To train the model, we need to extract additional information from the image filenames. The model needs more features to segment the classes correctly. In figure 3.1, According to the observations, image filenames are defined using four different values. Slice height and slice width are the first values, followed by height pixel spacing and width pixel spacing, respectively. A slice's height and width define the slide's resolution, represented in integer pixels, and the slice's height and width define its pixel spacing, which is expressed in millimeters (mm).

Data preprocessing is used to extract the information from the image file names into the training data frame. Python's "glob" library returns the file path from a specified directory. String slicing was applied after extracting each image file's name path sequentially to extract slice width height and pixel spacing width height.

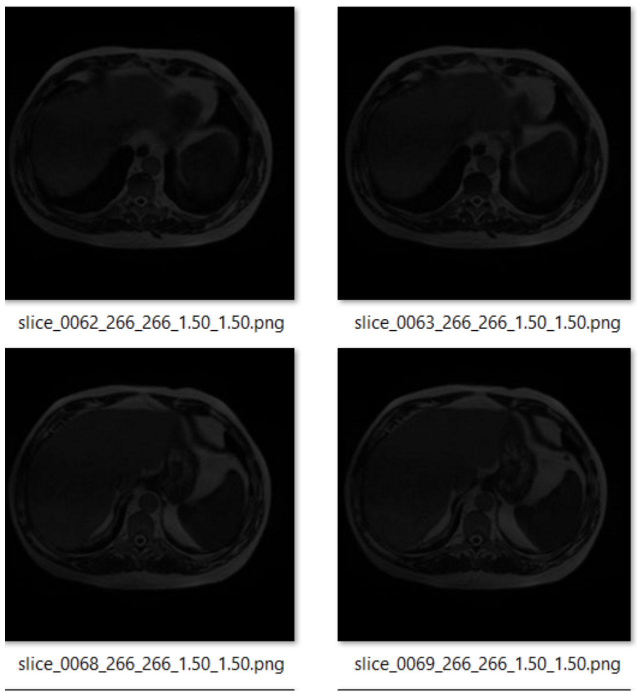

slice_0062_266_266_1.50_1.50.png
slice_0063_266_266_1.50_1.50.png
slice_0068_266_266_1.50_1.50.png
slice_0069_266_266_1.50_1.50.png

Figure 3.1: represents the random samples of sliced MRI image data for file names format.

## 3.3 Exploratory Data Analysis on Restructured data for segmentation masks

In order to determine the segmentation masks available for each class, all instances of the restructured data frame were taken into account in the Exploratory Data Analysis. Almost half of the samples, or more than half, did not have annotations of segmentation masks. From figure 3.2, Out of 38,500 samples available, 22,000 samples had no masks, marking 57 percent of the samples without masks. There are therefore 16.5 thousand samples with one or more annotated masks available with a percentage of 43 percent. For the wider view on one, two and three annotated masks on the same image, 16,500 samples with annotations were narrowed down. Almost 2,500 samples had a single annotation, 11,000 samples had two annotations, and only 3,000 samples had three annotations.

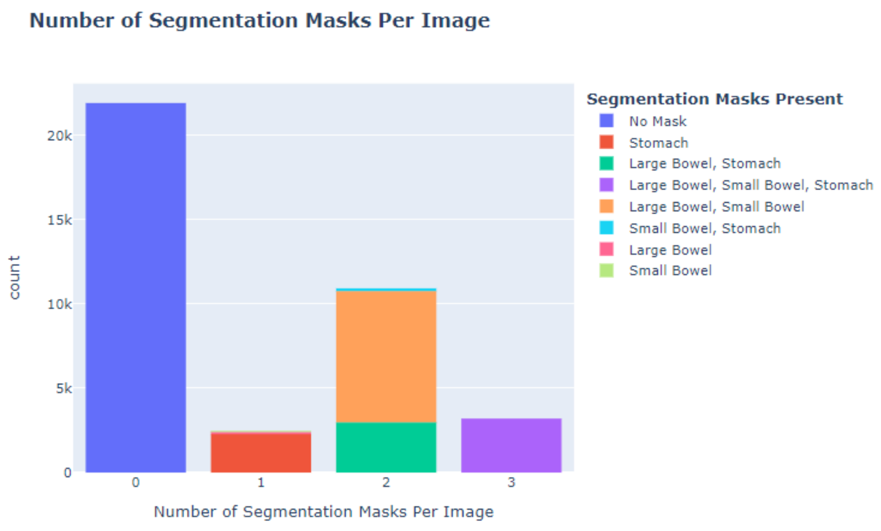


Figure 3.2: This figure represents the annotated samples available with and without mask based the different classes present

### 3.3.1 Data Pre-processing

In this study, the dataset used for training is freely available on Kaggle. "UW-Madison GI Tract Image Segmentation" allows researchers to build a deep learning model to segment GI tract scans semantically. Inconsistent widths and heights are the problems with these scans. Most images are rectangular, but a few are square. Similarly, masks were preprocessed, and 320x320 images were the model input. A zero padding was used where the length was less than 288, while pixel trimming was used where the size was more significant than 288 to match the input shape of the model (320x320).

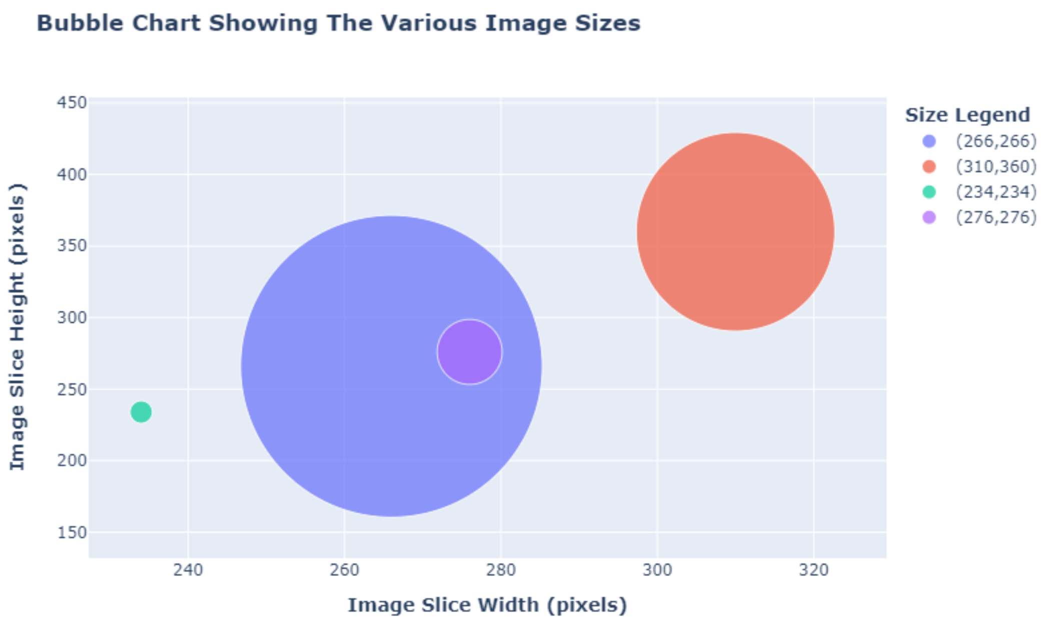


Figure 3.3: Represents various image sizes of the MRI scans of GI-tract tumour patients

As per the previous analysis of the data set, there are 38,500 samples there are 4 different sizes of images.
• (266,266) is the image size in large numbers with almost 26,000 samples out of 38,500.
• (310,360) is the next large number of available samples with around 11,300 samples out of 38,500
• (276,276) is the third largest sample size of images available with a number of 11,300 out of 38,500 samples.
• (234,234) is the image size with the least number of available samples around 150 out 38,500.

## 3.4 Model Architectures

Model architectures of models used in the training and evaluation are discussed in this section. To determine whether these models could adequately learn from the datasets, later testing and comparison are made with the research [34].

### 3.4.1 U-Net

Biomedical image segmentation was the focus of the U-Net architecture introduced in 2015. Image segmentation serves two purposes: localization and classification, unlike Image Classification, which predicts a single class for the whole input image. "localization" refers to finding an object's location (pixels) within a much larger image. Following localization comes classification, which is self-explanatory; it refers to classifying the object within the image [30].

In 2015, Ronneberger introduced U-Net a "Convolutional Networks for Biomedical Image Segmentation". A sliding window convolutional network was used in the U-Net to segment images. It used very few images in the training dataset and performed image augmentation to make it more capable of learning. At the time, U-Net outperformed other architectures [30]. It's clear from the name alone what the network is all about as the U-Net is a U-shaped network. Contracting and expansive paths are a part of the U-Net architecture. A network's basic intuition is that it will learn to classify an object on the downslope (contracting path), while on the upslope (Expansive path), it will learn to localize the object.

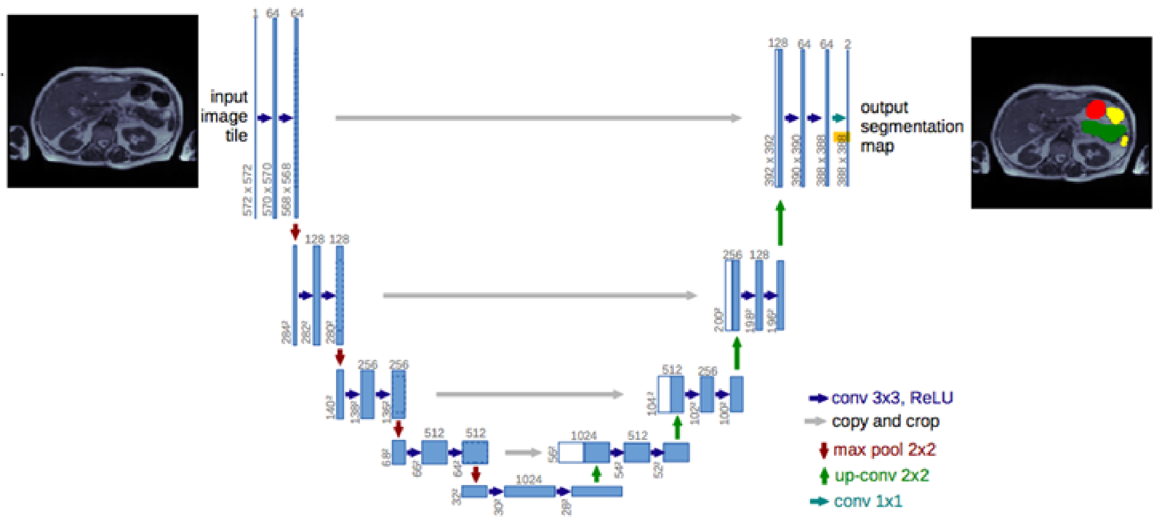


Figure 3.4: Represents the input and output expected after segmentation using U-net architecture [30]. The network was modified by adding hyperparameter and suitable metrics.

By observing the network from figure 3.4, it is evident that the contracting layer is passing information to the expansive layer. Doing so transfers context from the classification module to the localization module, enhancing the overall network's quality. Figure 3.4 represents the segmentation process from the input layer to the output layer, where the predictions are the segmentations of the labels stomach, large bowel and small bowel. Later, patches are created using the predicted segmentations on the MRI image classifying organs using respective colours for the labels. The results generated by pre-trained models are superior to those produced by models built from scratch. This paper [38] detects seven primary states of the face using convolutional neural networks and transfer learning. A comparison of three pre-trained networks is presented in the paper. The model ResNet50 is compared with VGG16 and SE-ResNet50. The models after training achieved validation accuracies of 96.8%, 99.47%, and 97.34% for VGG16, ResNet50 and SE-ResNet50, respectively. Based on the final evaluation results, Resnet50 was the most accurate network among all three [38]. Based on this reference, VGG16, Resnet34, and Resnet50 were selected to build models with GI-Tract cancer patients' data. VGG-16 is considered the base model to compare with ResNet-34 and ResNet-50. Based on intensive experimentation and analysis, the paper [20] concluded that VGG16 and Resnet50 are the most appropriate models that can be used for comparison.

### 3.4.2 VGG-16 Model architecture

ImageNet Large Scale Visual Recognition Challenge (ImageNet) was won in 2014 by a convolution neural net (CNN) architecture called VGG16. Figure 3.5 represents the structure of simple VGG-16 architecture, it is considered one of the best vision model architectures. VGG16 is unique in having only three-layer convolutions, stride 1, with the same padding and max pool layer as a 2x2 filter, stride 2. Instead of having many hyperparameters, they use 3x3 convolutions. Throughout the entire architecture, convolution and max pool layers are arranged consistently [37]. Finally, it has two FC layers followed by a Sigmoid. There are 16 layers in VGG16, and each layer has a weight. This network has approximately 138 million parameters, which is quite large. The model used in the project is the pre-trained model based on ImageNet weights.

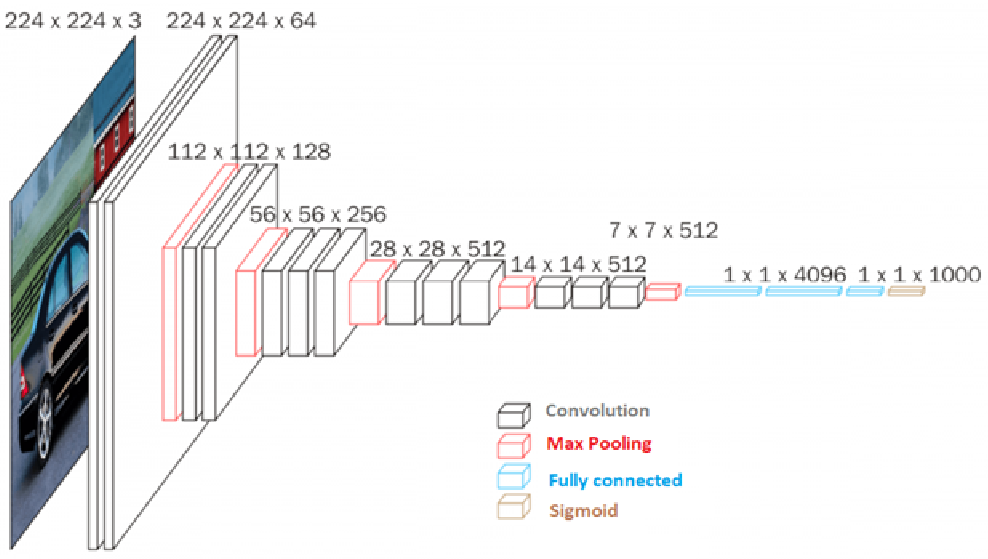


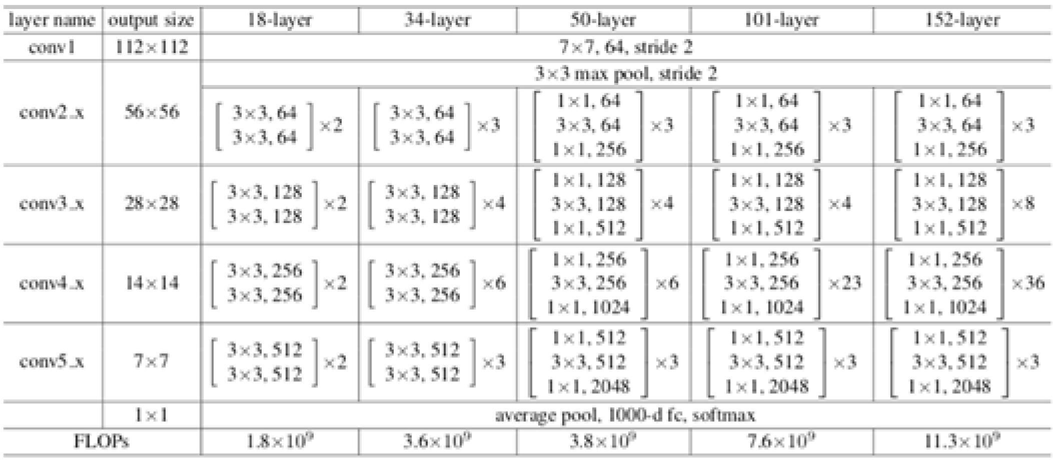
Figure 3.5: Implementation of the VGG-16 architecture [37]

All the layers in the pre-trained model can be used to extract all the required features from the data. It is possible to load the VGG16 models by specifying encoder weights as "imagenet", activation function as sigmoid, and input shape as (image height,image width,3) followed by the number of classes. As part of our project, we have three classes - stomach, large intestine, and small intestine.

### 3.4.3 Residual neural networks Model architecture (ResNet)

The Resnet network is a deep forward neural network with hundreds of layers. The problem of Models' accuracy saturating with more layers, then rapidly decays, and training errors increase can be solved using the ResNet model architecture. The models ResNet-34 and ResNet-50 are used for comparison with the VGG-16 considering it as the base model.

### 3.4.4 ResNet-34 and ResNet-50



| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | $112\times112$ | $7\times7$, 64, stride 2 | | | | |
| conv2_x | $56\times56$ | $3\times3$ max pool, stride 2 | | | | |
| | | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ |
| conv3_x | $28\times28$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$ |
| conv4_x | $14\times14$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$ |
| conv5_x | $7\times7$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ |
| | $1\times1$ | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

Figure 3.6: This figure shows the output size at every layer, along with the dimensions of the convolutional kernels [7]

There are four stages in the architecture of ResNet-34 and ResNet-50, with a residual network layer of 34 and 50, respectively. The convolutions for different residual models can be seen clearly in figure 3.6.

Taking into account the input image's height and width and the batch size of 32 and 3 as channel width. For example, if the input size is 224 x 224 x 3, every ResNet architecture conducts its initial convolution and maximum pooling with 7 x 7 and 3 x 3 kernels, with strides two, respectively. There are 3 Residual blocks in Stage 1 of the network, each containing three layers. For ResNet-34 and ResNet-50, the kernel size for the convolution operation is 64, 64, and 64,64,256, respectively, for all three block layers. Half will reduce input size in height and width, but channel width will be doubled. The channel width doubles from one stage to another and the input size is halved. Bottleneck designs are used on deeper networks, such as ResNet-50, ResNet-101, and ResNet-152. A residual function consists of three layers stacked on top of each other. Each layer consists of 1×1, 3×3, and 1×1 convolutions. A single layer of convolution reduces and then restores the dimensions using 1x1. With smaller input/output dimensions, the 3x3 layer remains a bottleneck. In our project, in the end, instead of the SoftMax function in the output layer, we use the hyperparameter sigmoid activation function.

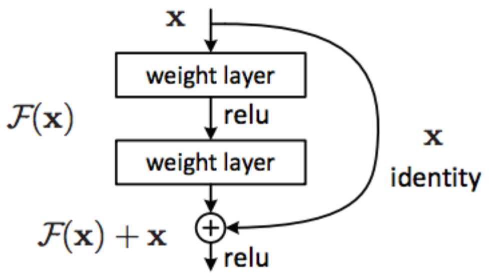

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.7: A building block of residual learning network [7]

The learning function by the model layer is given by $F(x)$. $H(x)$ is the function for data mapping and the inputs are given by $X$. The learning function of the Residual layer is given by :
$F(x) = H(x) — X$.

Figure 3.8 represents the basic building residual block for ResNet-34 and ResNet-50/101/152 respectively

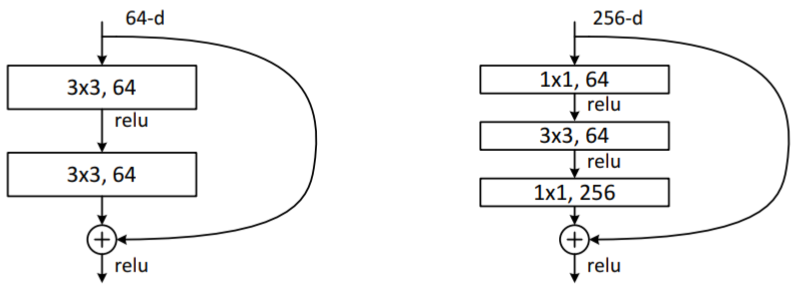


Figure 3.8: Left block is a building block for ResNet34. Right block is a "bottleneck" building block for ResNet-50/101/152 [7]

## 3.5 Training

Training data is divided into training, validation, and testing. To train the datasets, selecting the hyper-parameters for the pre-trained model is essential. The key parameters are a batch size, an optimizer, an activation function, and an epoch.

**Hyperparameters**

### 3.5.1 Batch size

Batch size describes how many training examples are used in one iteration while training the model. Two sets of batch sizes are used to determine the best model with the best performance. These are 16 and

32. The batch size is one of the essential hyperparameters to tune. When training models, practitioners often use a larger batch size to take advantage of the parallelism of GPUs. Nevertheless, too large a batch size will lead to poor generalization. When using a batch that consists of the entire dataset, the objective function is guaranteed to reach its global optima. However, those optimal solutions are less likely to be called empirically due to a slower convergence rate. In contrast, empirical studies show that smaller batch sizes rapidly lead to more "good" solutions [2]. The model can "start learning before seeing all the data with a smaller batch size." However, a smaller batch size can't guarantee convergence to global optima. Consequently, one should start with a small batch size to benefit from faster training dynamics, then slowly increase the batch size throughout training to ensure convergence is also achieved. It was found in this paper [2] that the model architecture trained with batch size 16 performed well at binary classification and that the model architecture trained with batch size 32 performed well at multilabel classification. Researchers trained fourteen different network architectures ten times, each with a multilabel-classification head. Five times each were used with a batch size of 16 and the remaining with a batch size of 32, respectively.

### 3.5.2 Optimizer

As part of deep learning training, the weights of each epoch must be adjusted, and the loss function must be minimized. Optimizers modify attributes of neural networks, such as weights and learning rates, by modifying their attributes. Later, the overall loss can be reduced, and the accuracy can be improved. The model will be trained using Adam optimizer to reach global minima. Global minima can be achieved using the Adam Optimizer when we are stuck in local minima during training. When we bounce a lot between epochs during training, the learning rate needs to be decreased.

### 3.5.3 Activation function

Sigmoid functions are used as activation functions. Based on the model's output, it is expected to give either a 0 or a 1, which is the probability of the prediction being correct. For values less than 0.5, the predicted value is 0, and for values greater than 0.5, it is 1.
Sigmoid functions are nonlinear, bounded functions that map real-valued inputs to outputs between 0 and 1. An artificial neural network uses this mathematical function to calculate probability. SoftMax is another activation function that can be used when we want to sum the probability distribution. For this project, we need sigmoid, which is used when you want the output to range from 0 to 1 but not sum to 1.

### 3.5.4 Epochs

The training was performed with 15 epochs, 50 epochs, and 16 and 32 batches. An epoch represents one complete cycle of training. Each epoch uses all data exactly once. Each epoch is divided into one or more batches, and we use part of the dataset for training: a forward pass and a backward pass are combined into one pass.

**Metrics**

### 3.5.5 F1 Score

"F1-score is the harmonic mean of precision and recall". F1 score is similar to the dice coefficient. As a way of measuring classification models, F1 scores, also known as F-Scores and F-Measures, are commonly used. Each precision and recall value in F1 scores is skewed toward the lowest value. Therefore, a higher F1 score will result in higher precision and recall.

$$F_1 = 2 * \frac{\text{precision} \ * \ \text{recall}}{\text{precision} \ + \ \text{recall}} \tag{3.1}$$

The equation should be further simplified:

$$F_1 = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})} \tag{3.2}$$

$$F_1 = 2 * \frac{\mathrm{TP}}{2 * \mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{3.3}$$

The dice score in terms of True positive, False negative and False positive is given by:

$$Dice = 2 * \frac{\text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{3.4}$$

Hence the dice coefficient and F1 score are similar.

### 3.5.6 Jaccard index(IoU)

The intersection-over-union (IoU) metric is one of the most common metrics used in semantic segmentation. IoU is a very effective metric for measuring the intersection of segmentation masks. IoU can be defined as the area of overlap between a predicted segmentation and the ground truth divided by its area of union.

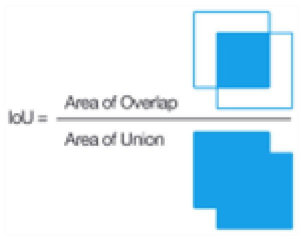


Figure 3.9: The IoU metric calculation

In binary (two classes) or multi-class segmentation, the mean IoU of an image is calculated by averaging the IoUs of each class. The IoU metric evaluates more accurately than pixel accuracy.

### 3.5.7 Dice coefficient

As a statistical tool, the Dice coefficient enables comparisons between two sets of data based on their degree of similarity in pixels. There is a great deal of similarity between IoU coefficients and Dice coefficients. There is a positive correlation between the two models, which means that they agree when one says that model A segmented an image better than model B. The two measures range from 0 to 1, with 1 indicating the greatest similarity between the predicted and the truth. The formula for dice coefficient is given below.

### 3.5.8 Hausdorff distance

In mathematics, the Hausdorff distance is measured by the Pompeiu-Hausdorff distance, which measures the distance between two subsets of a metric space. It converts the set of non-empty compact subsets of a metric space into a new metric space. When all the points of one set are close to some point of another set, two sets are consolidated in the Hausdorff distance. Using the Hausdorff distance, the longest distance can be estimated between two sets by choosing a point in one set to travel to the other. [12] It is the most significant distance between two points at two different locations.In 3D models, this metric determines the distance between two classes after the segmentation.

### 3.5.9 The loss function

In supervised training, a loss function tells the model how well it converges on the optimal parameters during the training process. When comparing two probability distributions, cross-entropy is used. The similarity metric is used to compare distributions of random events and for classification (in the broader sense) and segmentation. By measuring the difference in information content between the actual and predicted image masks, the binary cross-entropy loss (BCE) is calculated. The Bernoulli distribution is more generally used and works better when data distribution among classes is equal. The Binary Cross Entropy may not be able to adequately evaluate image masks with very heavy class imbalances, such as those found in X-ray images of very small, rare tumours. In an image mask, the Binay Cross Entropy treats both positive (1) and negative (0) samples equally. BCE loss may not accurately reflect the performance of the deep-learning model due to an unequal distribution of pixels for each object [13].

# Chapter 4

# Critical Evaluation

**The critical evaluation of the experimental results is primarily based on the decisions and options made while training the model compared to other models with different hyper-parameters to determine the best predicting model. In this section, we explain all the findings in a proper experimental finding order followed from beginning to end to achieve the optimally performing model.**

## 4.1 Data Evaluation

The training data is initially divided into two sets, one for the training set and one for the testing set. In a later step, the training set is again divided into training and validation sets. StratifiedGroupKFold was used to perform cross-validation using 5-splits (2-fold). The cross-validation process ensures that the same features appear in both the training and testing sets, which helps better use the data and improve the model's performance. It now applies to both training and validation.

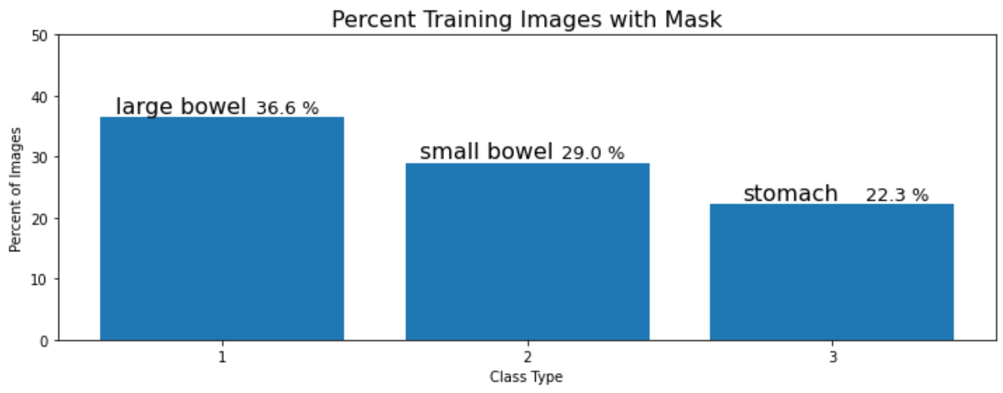


Figure 4.1: Representation of the masks available for the respective classes.

MRI image data samples are available in PNG format, excluding training data in CSV format. Not all samples have annotated masks in the different classes (large bowel, stomach, small bowel). Figure 4.1 shows that among the 38,500 training images, only 36.6% have masks on the larger bowel, followed by 29% for the small bowel and 22.3% for the stomach with the least amount of training images with masks. As the scans are divided into different case numbers, there are data anomalies due to improper masking data in the training set. Case 7 on day 0 and case 81 on day 30 have improper segmentation masks. In Figure 4.2,4.3, shows the improver segmentation masks in the training set. Before training the model, the data anomalies were removed from the training set. While performing exploratory analysis, it is necessary to pay special attention to data anomalies. Though these anomalies are few in number, they may influence model performance. Including reducing loss and improving model efficiency along with evaluating the data anamolies is an excellent work and good practice for every data scientist, even a 0.1%

improvement in model performance is significant in deep learning. It is possible to create patches on an image using the mpatches package from the matplotlib library along with the class of the respective colour using encoded mask annotations.

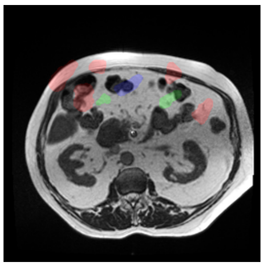


Figure 4.2: Case 7 improper segmentation mask.

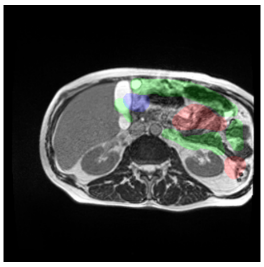


Figure 4.3: Case 81 improper segmentation mask.

Figure 4.4 shows a sample display of training images showing how masks are represented on an MRI scan.

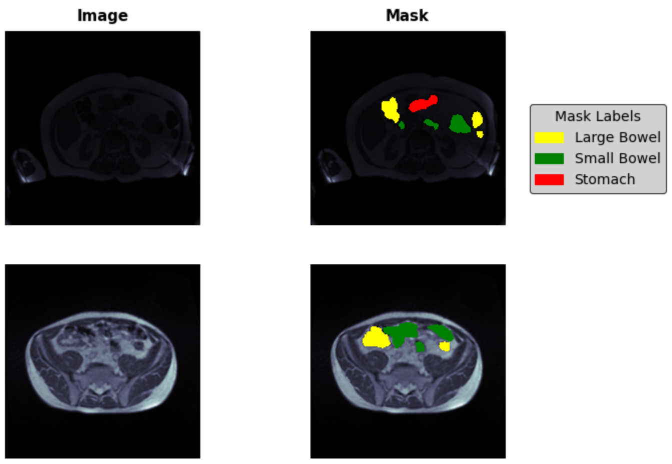


Figure 4.4: Represents the sample masks of training data

## 4.2 Training

Two important metrics are used to determine the model's predicted mask: the Dice coefficient and the Jaccard index. Where A is the set of pixels the model predicted to be class "1", B is the ground truth mask's set of pixels, and the absolute value indicates the number of pixels.
There is a lot of similarity between the Dice coefficient and the IoU. A positive correlation exists between them. The most common and preferred loss function is the pixel-wise cross-entropy loss used while training the model. The training was performed using the Adam optimizer [15] implementation of Keras using TensorFlow, which begins with the input images and their segmentation masks with a learning rate

of 2e 3 and a weight decay of epochs +2. Using Cosine Annealing [17], the learning rate of 2e-3 was used for decay. The model loss was not decreasing and fluctuated within the e range. Later, with factor =1, minimum delta =0.0001, and patience =5, the LR_PLEATUE was used to minimize the overhead and maximize GPU memory usage. The model was trained for 50 and 15 epochs for similar reasons, with batch sizes of 16 and 32. Sigmoid units were used in the final output layer to pass the model output through and update it to 0 if its less than 0.5 and update it to 1 if it was greater than 0.5.

## 4.3 Modelling

Based on transfer learning, a pre-trained model was used instead of one built from scratch for the training. The pre-trained model was modelled using ImageNet encoder weights. Three different models were selected for training: VGG-16, ResNet-34, and ResNet-50 considering the VGG16 model as a base. Based on this base model, the other models were compared [20].
ModelCheckpoint and EarlyStopping methods are imported after initializing the required model using the Keras libraries. By monitoring a specific model parameter, ModelCheckpoint allows us to save the model. Monitoring validation loss was observed by passing val_loss to ModelCheckpoint. Only if the current epoch's validation loss is less than the last one is saved. When there is no further decrease in the parameter set in EarlyStopping, EarlyStopping can stop training the model early. As a result, val_loss is passed to EarlyStopping to monitor validation loss. When the model doesn't see a decrease in validation loss within five epochs, patience is set to 5, which means it won't continue training.
In data augmentation, when the image was flipped horizontally, vertically, the zoom was set to 0.5, the wide shift range was 0.2, and the height shift range was 0.2; the data augmentation was not significantly different from simple data augmentation, which was done by adding new dimensions to the array of the data. To train the model, a batch size of 16 and a batch size of 32 were selected. This is because batch size is an essential hyperparameter for tuning the model because a larger batch size will result in poor generalization [2]. Using smaller batch sizes makes it easier to reach the global optima. According to the study, batch 16 performs well in binary classification, while batch 32 performs well in multilabel classification [2]. Based on the few proofs from the recent research, the model was trained with the smallest batch size possible. As 16 is the smallest possible batch size, the experimental evaluation started with 16 and then batch size 32 was selected. The models (Vgg-16, ResNet-34, ResNet-50) were trained in cycles of 15, and 50 epochs for batch sizes 16 and 32. Increasing the number of epochs beyond 11 decreases training set loss, which is nearly zero. Overfitting the model to training data increases validation loss. It depends on the dataset and how many epochs should be used. For data scientists, it is a good practice to start choosing the epoch value three times the number of columns in the data. A higher value should be tried if the model is still improving after all epochs have been completed. If it is not improving, a lower value should be considered. When training the model, GPUs were used. The total training time for 15 epochs was approximately 3 hours, and for 50 epochs was around 9 hours. In total, 80 hours were spent training all models with different variations of hyperparameters.

### 4.3.1 Models trained with batch size 16 and epochs (15, 50)

From table 4.1, the training loss was found to be the least in model ResNet-50 with a value of 0.3 when compared to the other two models, VGG-16 and ResNet-34. When the loss is less, the model is more accurate, so if the loss is 0.0, then the model is 100% accurate. Training loss represents the model's capacity to fit the training data, while validation loss indicates how well it fits the new data. Based on ResNet-50's metric values, dice score and IoU were 0.71 and 0.58, respectively. With batch size 16 and training epoch 15, the ResNet-50 performed better than the other two models.

According to table 4.2, all models have experienced a decrease in training losses. There is a decrease from 0.34 to 0.30 in training loss for the model Vgg-16. similarly, ResNet-34 has a decrease from 0.33 to 0.28 and ResNet-50 has a decrease of 0.1 loss. Increased training cycles have decreased training losses and increased IoU scores. According to the dice scores of all the models again, the ResNet-50 has the best dice score of 0.70, however, the same model score in 15 epoch training was 0.71, so rather than an increase, the dice score decreased by 0.1. As a result, the batch size should be increased to increase the probability of the dice score and with the reduced results of training loss reaching optimal levels.

| Models | Training loss | Dice coefficient | IoU |
|---|---|---|---|
| VGG-16 | 0.34 | 0.65 | 0.54 |
| ResNet-34 | 0.33 | 0.68 | 0.55 |
| ResNet-50 | 0.30 | 0.71 | 0.58 |

Table 4.1: Results of training models with batch size 16 and epochs 15

| Models | Training loss | Dice coefficient | IoU |
|---|---|---|---|
| VGG-16 | 0.30 | 0.64 | 0.61 |
| ResNet-34 | 0.28 | 0.69 | 0.64 |
| ResNet-50 | 0.29 | 0.70 | 0.66 |

Table 4.2: Results of training models with batch size 16 and epochs 50

### 4.3.2 Models trained with batch size 32 and epochs (15, 50)

Models were trained with batch size 32 and epoch 15. Comparing table 4.3 to table 4.1, the model trained with batch sizes 16 and 15 training cycles shows a reasonable increase in dice scores and IoU values, decreasing the training loss significantly. As a result, the choice of the perfect batch size assists in training the model efficiently that can predict accurately the unseen data. Comparing the performance of the models from table 3, the ResNet-50 model has retained its performance by performing better than others. ResNet-34 and ResNet-50 achieved the same training loss of 0.23. However, the dice score, which is also related to the F1 score, is 77.6% in ResNet-50 and 77.1% in ResNet-34.
To determine whether the model performance improved with epoch 50, the model was further trained. According to table 4.4, there is a significant decrease in training loss and an optimal increase in all metrics. After training from epoch 15 to 50, ResNet-50's dice score increased from 0.77 to 0.87. ResNet-50 performed optimally compared to VGG16 and ResNet-34 with batch size 32 and epochs 50. As for the ResNet-50 model metrics score, it has a dice score of 0.87, an IoU of 0.80, and a training loss of 0.13. Comparing ResNet-50 to VGG16 gives a better understanding of the performance of ResNet-50. As a result of its greater sensitivity to edges, Resnet-50 performs better than VGG16. It is possible to evaluate models more precisely using the dice metric since the dice score can be considered the F1 score.

| Models | Training loss | Dice coefficient | IoU |
|---|---|---|---|
| VGG-16 | 0.25 | 0.760 | 0.59 |
| ResNet-34 | 0.23 | 0.771 | 0.58 |
| ResNet-50 | 0.23 | 0.776 | 0.61 |

Table 4.3: Results of training models with batch size 32 and epochs 15

## 4.4 Graphical evaluation on trained model ResNet-50

We can conclude from figure 4.5 that the training data for model ResNet-50 is mostly unrepresentative based on the loss learning curves. When the number of samples in a dataset is too small, it can result in the validation dataset failing to capture the statistical characteristics of the training dataset. A StratifiedGroupKFold of 2 folds was performed before training the model to ensure data features are equally distributed between the training and validation sets. An exploratory analysis of the dataset revealed that out of 38,500 samples, 22,000 had no annotated masks, resulting in fewer data available for training. A dataset like this is called an unrepresentative dataset. In unrepresentative training datasets,

| Models | Training loss | Dice coefficient | IoU |
|---|---|---|---|
| VGG-16 | 0.14 | 0.860 | 0.791 |
| ResNet-34 | 0.14 | 0.867 | 0.790 |
| ResNet-50 | 0.13 | 0.871 | 0.80 |

Table 4.4: Results of training models with batch size 32 and epochs 50

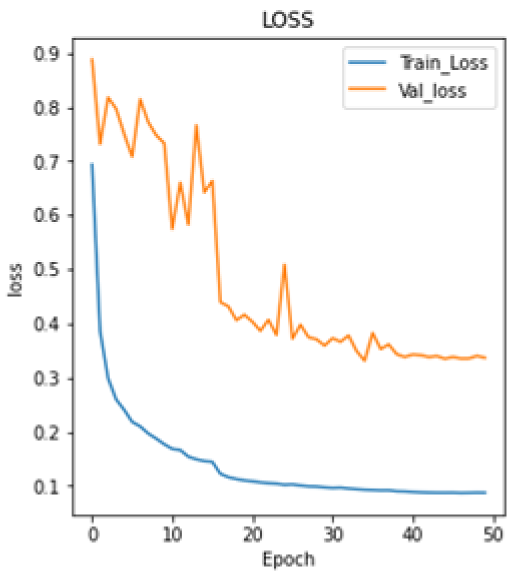


Figure 4.5: The training and validation loss of the model ResNet-50

the training dataset does not provide sufficient information to learn the problem, relative to the validation dataset. Figure 4.5 shows that there is a large gap between training and validation losses. As a result of learning noise from the training data, the model cannot completely generalize to the new data, which is the validation set, resulting in a gap between the training and validation losses. Based on the gap, we can determine the total noise learned from the model, which is Validation loss - training loss, which is 0.22. ResNet-50 underfits the data. Based on the training and validation sets of the model ResNet-50, figure 4.6 provides the learning cure values of the IoU coefficients for epochs ranging from 0 to 50. IoU values can be used to determine the overlap between predicted segmentation and ground truth in the probability values between 0 and 1, where 1 indicates 100% overlap. Tables 4.4 and 4.6 represent the mean IoU coefficient score of training and validation IoU of ResNet-50 is 0.80 and 0.70, respectively, as shown in figure 4.6.

According to figures 4.6 and 4.7, dice and IoU scores are positively correlated in both training and validation curves. The dice score of the validation set differs by 0.2 from the training curve score because of model learning noise from the training set. As shown in Figure 4.7, training and validation dice scores increased rapidly until epoch 30. Later, it maintained almost saturation scores until epoch 50, with a slight increase in values at every epoch after 30.

## 4.5 Predictions on the validation set by the model ResNet-50

A model based on Resnet-50 is used to predict segmentation on the validation set. Figure 4.8 illustrates three columns, one with a normal MRI scan, one with a masked image using training data, and one with a masked image using predicted data for visually comparing. The mask labels represent large bowel in the yellow, small bowel in green, and stomach in red. Using "mpatches" and "Matplotlib" python library, segmentation masks are created on the MRI scans using the predicted segmentation masks and compared with the sample mask of training data.

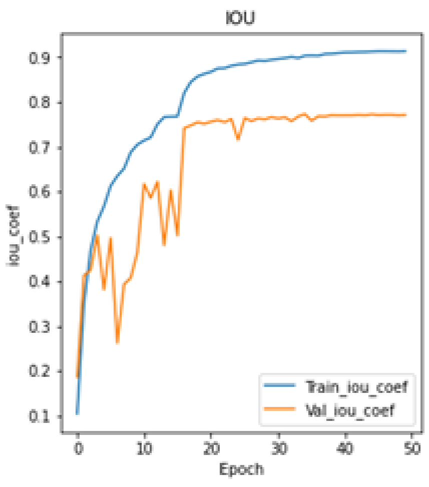


Figure 4.6: Training and validation IoU coefficient values of ResNet-50

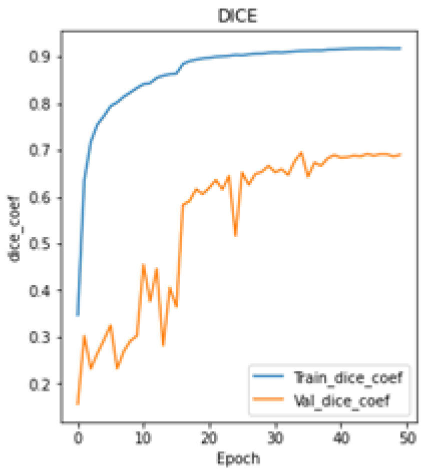


Figure 4.7: Training and validation Dice coefficient values of ResNet-50

## 4.6 Further investigation

Further research was conducted on the ResNet-18 and ResNet-101 models to determine whether they perform better than ResNet-50. The model with a lesser layer learned the data more accurately and modeled more quickly. ResNet-18 performed well in predicting binary classifications [2]. Therefore, to improve the research findings on ResNet-50, the models ResNet-18 and ResNet-101 were trained on the same dataset used for training ResNet-50 with the same hyperparameters (Batch size = 32 and Epoch = 50). This study was conducted to determine if ResNet-101, the next version of ResNet-50, can perform better than ResNet-50 on a broader scale. In addition, on a larger scale, the models ResNet-18, ResNet-50, and ResNet-101 were chosen for better understanding in sequential order for easier evaluation.

| Models | Training loss | Dice coefficient | IoU |
|---|---|---|---|
| ResNet-18 | 0.128 | 0.88 | 0.76 |
| ResNet-101 | 0.23 | 0.77 | 0.59 |

Table 4.5: Results of training models with batch size 32 and epochs 50

Initially, the results of training curves with loss and metrics values were compared with those of the models ResNet-18 and ResNet-101. Table 4.5 shows that model ResNet-18 performed much better

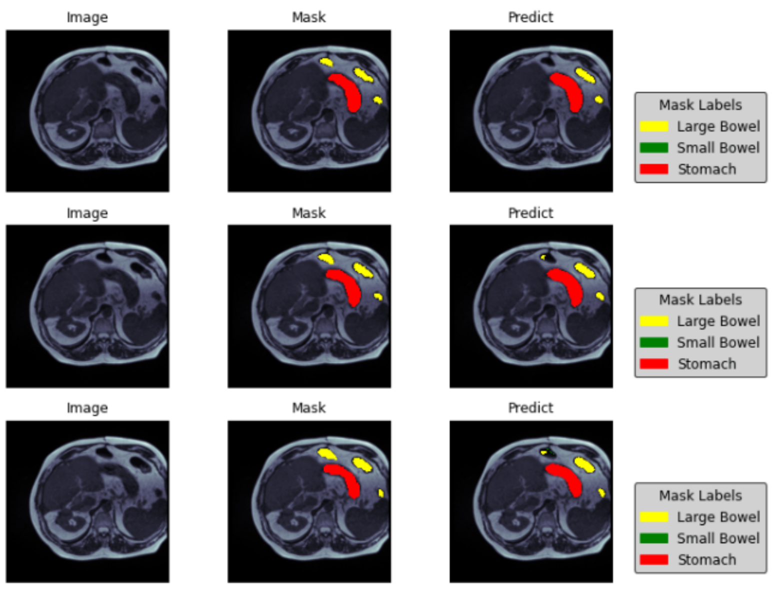


Figure 4.8: This represents the prediction on the validation set by ResNet-50

than ResNet-101 as the ResNet-18 had a training loss of 0.128 with a Dice score of 0.88 and IoU pixel comparison with predicted and ground truth of 0.76.

| Models | Validation loss | Validation IoU |
|---|---|---|
| ResNet-18 | 0.49 | 0.71 |
| ResNet-50 | 0.35 | 0.70 |
| ResNet-101 | 0.67 | 0.49 |

Table 4.6: Results of validation loss and IoU of models with batch size 32 and epochs 50

ResNet-101 was excluded from the comparison since ResNet-50 has proven to perform the best so far. As part of the final selection evaluation, ResNet-18 and ResNet-50 were compared. Based on table 4.4 and table 4.5, ResNet-18 had a training loss of 0.13, which is negligible compared to ResNet-50, with a training loss of 0.128. ResNet-50's mean IoU coefficient of 0.80 was higher than ResNet-18's mean IoU coefficient of 0.76. From the table 4.6 ResNet-18's validation loss was 0.49 despite having a better dice score; ResNet-50's validation loss is 0.35, with less noise learnt from training. Models that learn less noise during training perform better in predicting unseen data. The model was made to predict values on the testing set, which predicted the RLE encoded segmentations of the multi-class. In conclusion, ResNet-50 performed well compared to other models when predicting multiclass semantic segmentation of medical images. Due to the limited number of datasets with masked images, the model ResNet-50 is underfitting, whereas the model would have fit perfectly if the training data had more RLE encoded segmentations for their respective classes. As a result, the model achieved a dice score of 0.871, close to the dice score of the first-place winning model, which achieved a dice score of 0.892.

# Chapter 5

# Conclusion

**The purpose of this chapter is to discuss my final conclusions and thoughts about future research.**

## 5.1 Overall Dissertation summary

Gastrointestinal tract image segmentation was selected as the topic of interest from a Kaggle competition. The task was to mainly build the best model using deep learning to perform segmentation on MRI scans of patients with gastrointestinal tract cancer. As a result of a thorough literature review of many articles and research papers, the residual neural network architecture was found to be one of the best architectures in deep learning to segment images. The models ResNet-50, VGG-16, and ResNet-34 were initially evaluated by considering VGG-16 as the base model and comparing the results. Initially, the primary task was to preprocess and restructure data into a format suitable for training the model. According to the results of exploratory analysis of the dataset, almost 57% of the training data lacked RLE segmentation masks. Further research was conducted on possible approaches for augmenting data to produce meaningful results from deep learning. Compared to the model trained with simple data augmentations, the results didn't significantly change after performing the different data augmentation techniques. Model training required more computing power, so GPUs were used. The pre-trained models under ImageNet weights were selected by understanding the importance of transfer learning. Two important hyperparameters were varied to tune the model for better performance: batch size and epochs. Training with small batches and increasing them gradually enabled to optimize the model and select the best parameter for tuning more quickly. A total of 15 to 50 epoch cycles were used to train the model with the best parameter to train on the dataset. It was determined to be the batch size 32 with epochs 50 after 80 hours of comprehensive training on all models with varied hyperparameters with a parameter combination of batch size 16,32 and epoch 15,50. It was found that the model ResNet-50 performed significantly well compared to other models. The model ResNet-50 was underfitting due to the availability of less training data, as half of the sliced images out of 144 slices were null with no mask. Despite less availability of annotated data, the model achieved a validation loss of 0.35. Considering the hyperparameter batch size and epochs as the main parameters to tune the model, the model ResNet-50 showed good performance achieving the training dice coefficient score of 0.871. The model can produce optimal results in real-time with the more proper annotated RLE encoded data on MRI scans.

## 5.2 Project status and achievements

The objectives set at the project's beginning were achieved as they were reasonably achievable but needed more understanding of the concepts in deep learning. Understanding the need to use specific metrics and hyperparameters while training the model was necessary, and these basic tasks were first achieved. The research performed on this project was quite engaging as it needed more training hours, and also the importance of pre-trained models was understood. Among the model selected, first, the model was trained with a pattern of varying batch sizes and epochs. Finally, with batch size 32 and epochs 50, the model ResNet-50 achieved optimal results. The model was then compared with its next version of residual neural network ResNet-101 with more layers than ResNet-50. After a critical evaluation of these models, ResNet-50 performed better. Since this is a Kaggle competition, the model's performance is mainly evaluated on the dice score, and the top winning model dice score is 0.892. Finally, I achieved a dice score of 0.871 with the model ResNet-50.

## 5.3 Concluding thoughts

As the training data were not sufficient to train the model this training data can be called unrepresentative data as half of the data had no segmentation masks. Transfer learning plays a vital role in data modeling as the pre-trained models used were trained under ImageNet weights. Even though the possible required annotations were fewer the pre-trained model learned more features very easily and performed well. If the model was built from scratch using the available dataset, then the model would have not performed to the level of the pre-trained model. I personally believe that there are many other unexplored probabilistic possible ways to augment the data that need to be explored, which can train the model from scratch very easily and allow them to exhibit performance similar to the pre-trained model. Any model with a lesser layer if trained with proper parameters and tuned accordingly then they can perform exceptionally well. The ResNet-50 could have been the best fit with a little more RLE encoded segmentation masks of MRI scans in the training data. From the medical industry perspective, Advanced deep learning with improved quality of medical imaging can tremendously improve the health sector with faster disease diagnosis and proper treatment.

## 5.4 Further work

Even though the data was modeled using the data from the Kaggle competition this is a real-time problem that needs solutions in real-time. If the solution is achieved with the best model the implementation needs far more advancements starting from the quality of the medical image. This research can be further taken forward for further research, considering few thoughts and problems faced during the dissertation; if these could be resolved then further research will be achievable.

• The data provided for modeling had fewer features, whereas out of 38,500 samples only around 16,000 samples had segmentation data in varied categories. If there are advancements in data augmentations other than the techniques which are present, this increases the ground truth data and better model learning.

• The need for computational power, as even the use of GPUs took hours together to train the pre-trained model. The use of TPUs (Tensor processing units) should be initiated.

• In real life the computer self-built model performing image segmentation on the actual MRI scan is not trusted even though the model works perfectly in segmentation. The reason behind this is a lack of belief and awareness of the possibilities of Deep learning techniques.

• The medical imaging techniques capturing images in the current generation do not capture the position of the organs accurately. Where in which after a couple of trials partial medical images can be achieved, even though the use age of medical imaging is improving every day it strictly needs advancements to capture high-quality medical images.

• Use of pre-trained models provides great results with fewer data, More availability of pre-trained models can provide large-scale improvements in deep learning.

• Good quality medical images help build perfect 3D medical images. The Harsdorf distance calculated between the organs in the 3D models can be more accurate and helpful in medical sectors.

# Bibliography

[1] Uw-madison gi tract image segmentation.

[2] Keno K. Bressem, Lisa Adams, Christoph Erxleben, Bernd Hamm, Stefan Markus Niehues, and Janis Lucas Vahldiek. Comparing different deep learning architectures for classification of chest radiographs. *CoRR*, abs/2002.08991, 2020.

[3] Binge Cui, Xin Chen, and Yan Lu. Semantic segmentation of remote sensing images using transfer learning and deep convolutional neural network with dense connection. *Ieee Access*, 8:116744–116755, 2020.

[4] Aarish Shafi Dar and Devanand Padha. Medical image segmentation: A review of recent techniques, advancements and a comprehensive comparison. *Int. J. Comput. Sci. Eng*, pages 114–124, 2019.

[5] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, 2020.

[6] Sourodip Ghosh, Aunkit Chaki, and KC Santosh. Improved u-net architecture with vgg-16 for brain tumor segmentation. *Physical and Engineering Sciences in Medicine*, 44(3):703–712, 2021.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] Lei Huang, Jie Qin, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Normalization techniques in training dnns: Methodology, analysis and application. *arXiv preprint arXiv:2009.12836*, 2020.

[12] Daniel P Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[13] Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–7. IEEE, 2020.

[14] Debesh Jha, Pia H Smedsrud, Michael A Riegler, Dag Johansen, Thomas De Lange, Pål Halvorsen, and Håvard D Johansen. Resunet++: An advanced architecture for medical image segmentation. In *2019 IEEE International Symposium on Multimedia (ISM)*, pages 225–2255. IEEE, 2019.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Jan JW Lagendijk, Bas W Raaymakers, and Marco Van Vulpen. The magnetic resonance imaging–linac system. In *Seminars in radiation oncology*, volume 24, pages 207–209. Elsevier, 2014.

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[18] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.

[19] Priyanka Malhotra, Sheifali Gupta, and Deepika Koundal. Computer aided diagnosis of pneumonia from chest radiographs. *Journal of Computational and Theoretical Nanoscience*, 16(10):4202–4213, 2019.

[20] Sheldon Mascarenhas and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99. IEEE, 2021.

[21] Sumanth Meenan. Generating masks from encoded pixels-semantic segmentation, Oct 2019.

[22] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of big data*, 2(1):1–21, 2015.

[23] Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. Data augmentation for brain-tumor segmentation: a review. *Frontiers in computational neuroscience*, 13:83, 2019.

[24] Shuteng Niu, Meryl Liu, Yongxin Liu, Jian Wang, and Houbing Song. Distant domain transfer learning for medical imaging. *IEEE Journal of Biomedical and Health Informatics*, 25(10):3784–3793, 2021.

[25] Alexey A Novikov, Dimitrios Lenis, David Major, Jiří Hladvka, Maria Wimmer, and Katja Bühler. Fully convolutional architectures for multiclass segmentation in chest radiographs. *IEEE transactions on medical imaging*, 37(8):1865–1876, 2018.

[26] Emilio Soria Olivas, Jos David Mart Guerrero, Marcelino Martinez-Sober, Jose Rafael Magdalena-Benedito, L Serrano, et al. *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques: Algorithms, methods, and techniques.* IGI global, 2009.

[27] Dzung L Pham, Chenyang Xu, and Jerry L Prince. A survey of current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(3):315–337, 2000.

[28] Pravda Jith Ray Prasad, Ole Jakob Elle, Frank Lindseth, Fritz Albregtsen, and Rahul Prasanna Kumar. Modifying u-net for small dataset: a simplified u-net version for liver parenchyma segmentation. In *Medical Imaging 2021: Computer-Aided Diagnosis*, volume 11597, pages 396–405. SPIE, 2021.

[29] Anindya Apriliyanti Pravitasari, Nur Iriawan, Mawanda Almuhayar, Taufik Azmi, Irhamah Irhamah, Kartika Fithriasari, Santi Wulan Purnami, and Widiana Ferriastuti. Unet-vgg16 with transfer learning for mri-based brain tumor segmentation. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 18(3):1310–1318, 2020.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Terrence J Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, 2020.

[32] P Shafi and D Padha. Medical image segmentation a review of recent techniques. *Adv. Compr. Comp. Int. J. Comput. Sci. Eng.*, 7(7):114–124, 2019.

[33] Ling Shao, Fan Zhu, and Xuelong Li. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034, 2014.

[34] Manhar Sharma. Automated gi tract segmentation using deep learning. *arXiv preprint arXiv:2206.11048*, 2022.

[35] Rebecca Smith-Bindman, Marilyn L Kwan, Emily C Marlow, Mary Kay Theis, Wesley Bolch, Stephanie Y Cheng, Erin JA Bowles, James R Duncan, Robert T Greenlee, Lawrence H Kushi, et al. Trends in use of medical imaging in us health care systems and in ontario, canada, 2000-2016. *Jama*, 322(9):843–856, 2019.

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[37] Rohit Thakur. Step by step vgg16 implementation in keras for beginners, Nov 2020.

[38] Dhananjay Theckedath and R. R. Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1(2), 2020.

[39] Song-Toan Tran, Ching-Hwa Cheng, Thanh-Tuan Nguyen, Minh-Hai Le, and Don-Gey Liu. Tmd-unet: Triple-unet with multi-scale input features and dense skip connection for medical image segmentation. In *Healthcare*, volume 9, page 54. MDPI, 2021.

[40] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[41] Haonan Wang, Peng Cao, Jiaqi Wang, and Osmar R Zaiane. Uctransnet: rethinking the skip connections in u-net from a channel-wise perspective with transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2441–2449, 2022.

[42] Xiao-Xia Yin, Sillas Hadjiloucas, Le Sun, John W Bowen, and Yanchun Zhang. A review on the rule-based filtering structure with applications on computational biomedical images. *Journal of Healthcare Engineering*, 2022, 2022.

[43] Xiao-Xia Yin, BW-H Ng, Qing Yang, Alex Pitman, Kotagiri Ramamohanarao, and Derek Abbott. Anatomical landmark localization in breast dynamic contrast-enhanced mr imaging. *Medical & biological engineering & computing*, 50(1):91–101, 2012.

[44] Xiao-Xia Yin, Le Sun, Yuhan Fu, Ruiliang Lu, and Yanchun Zhang. U-net-based medical image segmentation. *Journal of Healthcare Engineering*, 2022, 2022.

[45] XX Yin, S Hadjiloucas, and Y Zhang. Pattern identification of biomedical images with time series: Contrasting thz pulse imaging. 2016.

[46] Yuan Yuan and Lei Lin. Self-supervised pretraining of transformers for satellite image time series classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:474–487, 2020.

[47] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Learning pyramid-context encoder network for high-quality image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1486–1494, 2019.

[48] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019.