T.R.

EGE UNIVERSITY

Graduate School of Applied and Natural Science

A NEW LOSS FUNCTION TO BE USED IN DEEP NETWORKS FOR IMAGE SEGMENTATION OF COLORECTAL POLYPS

Mahmut Ozan GÖKKAN

Supervisor : Prof. Dr. Mehmet KUNTALP

Department of Biomedical Technologies

Industrial Ph.D. Programme of Advanced Biomedical Technologies

İzmir

2023

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum "A New Loss Function to be used in Deep Networks for Image Segmentation of Colorectal Polyps" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

27/01/2023

Mahmut Ozan Gökkan

ÖZET

KOLOREKTAL POLİPLERİN GÖRÜNTÜ BÖLÜTLEMESİ İÇİN DERİN AĞLARDA KULLANILACAK YENİ BİR KAYIP FONKSİYONU

GÖKKAN, Mahmut Ozan

Doktora Tezi, Biyomedikal Teknolojiler Anabilim Dalı

Tez Danışmanı: Prof. Dr. Mehmet KUNTALP

Ocak 2023, 65 sayfa

Poliplerin geç saptanması sonucunda kolorektal kanserler ortaya çıkabilmektedir. Kolonoskopistler, eksizyonel biyopsi ile polipleri çıkarmak için bir kolonoskopi cihazı kullanır. Bu çalışmanın amacı, klinik uygulamaya katkı niteliğinde bir polip bölütleme modeli sağlayan bir web uygulaması geliştirmektir. Polip segmentasyonu için, derin sinir ağlarında hem dengesiz veri setinin hem de kaybolan gradyan probleminin üstesinden gelmek için kullanılmak üzere yeni bir dengesizlik farkındalıklı kayıp fonksiyonu, yani her şeyi kapsayan kayıp geliştirilmiştir. Yeni bir fonksiyon geliştirmenin ikinci önemli nedeni, bölge bazlı, şekle duyarlı ve piksel bazında dağıtım kaybı yaklaşımlarının değerlendirme yeteneklerine sahip daha kapsamlı bir fonksiyonunu bir kerede üretebilmektir. Bunu yapmak için, bir algoritma karmaşıklığını tanımlamanın temsili bir parametresi olarak her biri saniyede farklı kayan nokta işlemine (FLOPS) sahip iki farklı evrişimsel sinir ağı (CNN) ele alınmıştır. İlk olarak, kodlayıcı olarak ResNet18 ve kod çözücü olarak bir UNet gerçekleştirilmiştir. İkinci olarak, kodlayıcı olarak 34 katmanlı bir artık ağ ve kod çözücü olarak bir UNet tasarlanmıştır. Her iki CNN mimarisi için, popüler dengesizlik farkında kayıpları kullanmanın sonuçları, önerilen yeni kayıp fonksiyonumuzu kullanmanın sonuçlarıyla karşılaştırılmıştır. Eğitim, 5 katlamalı çapraz doğrulama ve test adımları için, erişime açık birden çok veri kümesi kullanılmıştır. Bu veri kümelerindeki orijinal verilere ek olarak, çevirme, ölçekleme, döndürme ve kontrast sınırlı uyarlanabilir histogram eşitleme işlemleriyle bunların artırılmış

örnekleri de oluşturulmuştur. Arttırılmış örnek veriler hem eğitim hem de doğrulama aşamasında kullanılırken, orijinal görülen veri kümeleri test aşamasında işlenmiştir. Sonuç olarak, önerilen yeni özel kayıp fonksiyonumuz, popüler kayıp işlevleriyle karşılaştırıldığında en iyi performans ölçümlerini üretmiştir.

Anahtar Kelimeler: Polip bölütleme, kayıp fonksiyonu, derin sinir ağı, sınıf dengesizliği, kaybolan gradyan problemi.

ABSTRACT

A NEW LOSS FUNCTION TO BE USED IN DEEP NETWORKS FOR IMAGE SEGMENTATION OF COLORECTAL POLYPS

GÖKKAN, Mahmut Ozan

PhD thesis in Biomedical Technologies Department
Supervisor: Prof. Dr. Mehmet KUNTALP
January 2023, 65 pages

Colorectal cancers may occur as a result of late detection of polyps. Colonoscopists use a colonoscopy device to remove polyps by excisional biopsy. The aim of this study is to develop a web application that provides a polyp segmentation model that contributes to the clinical application. For polyp segmentation, a new imbalance-aware loss function, i.e. omni-comprehensive loss, has been developed to be used in deep neural networks to overcome both imbalanced datasets and the vanishing gradient problem. The second crucial reason for developing a new loss function is to be able to produce a more comprehensive one with the evaluation capabilities of region-based, shape-sensitive and pixel-wise distribution loss approaches at once. To do this, two different convolutional neural networks (CNNs) have been implemented, each with different floating point operations per second (FLOPS) that is a representative parameter to identify an algorithm complexity. First, ResNet18 as the encoder and a UNet as the decoder is implemented. Second, a 34-layer residual network is designed as the encoder and a UNet as the decoder is designed. For both CNN architectures, the results of using the popular imbalance-aware losses are compared with the results of using our new proposed loss function. Multiple publicly available datasets are used for training, 5-fold cross validation, and testing steps. In addition to the original data in these datasets, augmented versions of these datasets have also been generated by flipping,rotating and contrast-limited adaptive histogram equalization operations. While the augmented samples are used both in the training and validation phases, the original datasets are tackled in the testing phase.While the augmented samples

are used both in the training and validation phases, the original datasets are tackled in the testing phase. As a result, our proposed new custom loss function produced the best performance metrics compared to the popular loss functions.

Keywords: Polyp segmentation, loss function, deep neural network, class imbalance, vanishing gradient problem.

PREFACE

Colorectal polyps may cause cancer and may carry a mortal risk. Due to the epidemiology of colorectal polyps, such an abnormal tissue structure with a high prevalence may differ according to age, gender and demographic structure. Therefore, early diagnosis and polyp resection by excisional biopsy play important roles. At this point, it has become possible with this thesis study to provide an application model integrated into the clinical application by using advanced computer vision technologies. Thanks to the developed web application module, segmentation of polyps and the number of segmented polyps are monitored. It can be said that this is a very important technological development in providing a fast and accurate diagnosis for colonoscopists.

I sincerely would like to thank my supervisor Prof. Dr. Mehmet Kuntalp for his valuable guidance, suggestions, criticism, and support during this study.

I would also like to thank dear thesis committee members to their time, generous share of knowledge, and constructive comments.

Finally, I want to thank my family for supporting me.

İZMİR

27/01/2023 Mahmut Ozan GÖKKAN

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

Figure Page

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Definition
μ_c	Mean value of RGB channels
σ_c	Standard deviation RGB channels
α_t	Coefficient used in focal loss and focal Tversky loss
γ	Controlling factor used in focal loss
α	Adjusting parameters used in Tversky loss and Omni-Comprehensive loss
θ	Penalizing factor for false positive and false negative used in Omni-comprehensive loss
β	Penalizing factor for false positive and false negative used in Tversky loss
$\partial\int$	Derivative of Omni-comprehensive loss function
$\partial\theta(\mathbf{w})$	Derivative of normalized cross correlation

Abbreviation	Definition
CNN	Convolutional Neurl Network
TI	Tversky Index
TL	Tversky loss
FTL	Focal Tversky loss
BCE	Binary Cross Entropy
NCC	Normalized Cross Correlation
FPNet	Feature Pyramidal Network

LIST OF SYMBOLS AND ABBREVIATIONS (Continued)

Abbreviation	Definition
RPNet	Region Proposal Network
FN	False Negative
CLAHE	Contrast-limited adaptive histogram equalization
CVC-Clinic DB	Colonoscopic polyp database extracted from colonoscopy videos
CVC-Colon DB	Colonoscopic polyp database extracted from colonoscopy videos
DC	Dice Coefficient
DL	Dice loss
ETIS-Larib DB	Colonoscopic polyp image database
FL	Focal Loss
FLOPS	Floating Point Operations per Second
FP	False Positive
GFLOPS	Giga Floating Point Operations per Second
Kvasir-SEG DB	Gastrointestinal and colonoscopic polyp image database
mAcc	Mean of accuracy values using test set
MCC	Matthews correlation coefficient
mDice	Mean of dice values using test set
mF2	Mean of F2 values using test set
mPrec	Mean of precision values using test set

LIST OF SYMBOLS AND ABBREVIATIONS (Continued)

Abbreviation	Definition
mSens	Mean of sensitivity values using test set
mSpec	Mean of specificity values using test set
Nvidia GTX	Nvidia Graphics Card Model
Params (M)	Parameters in million
PUE	Power Usage Effectiveness
ReLu	Rectified linear unit
ResNet	Residual Neural Network
ResNet18	18-layered residual neural network
ResNet34	34-layered residual neural network
TN	True Negative
TP	True Positive
UNet	U-shaped Neural Network
DL	Dice loss
Params (M)	Parameters in million
mDice	Mean of dice values using test set
mAcc	Mean of accuracy values using test set
mSens	Mean of sensitivity values using test set
mSpec	Mean of specificity values using test set

LIST OF SYMBOLS AND ABBREVIATIONS (Continued)

Abbreviation	Definition
mPrec	Mean of precision values using test set
mF2	Mean of F2 values using test set
mMCC	Mean of matthews correlation coefficient values using test set

1 INTRODUCTION

Colorectal polyps may develop into cancerous tissue types. Operator-dependent colonoscopy is the gold standard and colonoscopists often use this procedure for viewing the entire colon to remove polyps by excisional biopsy. Colonoscopy devices need special training and experience to prevent misdiagnosis of suspected lesions. For this reason, computer vision applications are developed using colonoscopic different type of images, including white light and filtered images for the purpose of classification and segmentation. In addition, the experience of a colonoscopist and the characteristics of a physician also play an important role in the correct determination of adenoma (Mehrotra et al., 2018).
Deep neural networks can be a good decision-making system for polyp detection. Hence, such an application model could help clinicians make excisions without damaging healthy mucosal tissue. At this point, the encoder-decoder neural networks can be used to segment a polyp on an image to contribute to the clinical application in gastroenterology units.

For pattern recognition and feature extraction step, one of the encoders may be used as a backbone such as residual neural networks (ResNets). Channel concatenation, known as skip connection, can be used for combining a decoder with one of the encoders. Datasets used for obtaining reliable and high scores play an important role for acquaring high performance. When analyzing CVC-Clinic DB, CVC-Colon DB, ETIS-Larib polyp DB and Kvasir-Seg polyp datasets and their spatial information as shown in Table 3.1, it is possible to say that these datasets have relatively much more background area than the foreground, identified as non-polyp region and polyp region, respectively. Due to the imbalanced distribution of the datasets, the number of the 0-labeled class is much higher than the respective number of the 1-labeled class in all datasets. That’s why building a democratized loss layer has a critical importance when developing a loss function and focusing more on polyp regions than non-polyp regions. In this thesis, the ratio of 0-labeled pixel class is 89.6% while the ratio of 1-labeled pixel class is 10.4% in all datasets due to changeable polyp size. 0-labeled class has no polyp region and 1-labeled class has polyp region. This kind of imbalanced distribution of classes may cause unfair and degenerated learning models. A solution is to use a suitable and generalizable loss function. In addition, such a loss function should have two important properties: First, a loss function should be aware of the unbalanced distribution between both intra-class distribution of images and inter-class distribution of pixel-wise levels in an image. Second, a loss function should overcome the vanishing gradient problem to prevent overfitting.

For this situation, a novel loss function is developed for making training model democratize to segment the precise region of a polyp by using 18-layer and 34-layer ResNets as encoders and a UNet as the decoder network. For model training, the encoders are initialized using pre-trained imagenet weights in a way of transfer learning. The challenge of this study is to create a new paradigm, developing an artificial intelligence enabling technology with the loss function called omni-comprehensive loss, to be used in deep neural networks to overcome both imbalanced dataset and the vanishing gradient problem. The second reason and important advantages of developing this function is to be able to produce a more comprehensive one that has evaluation capabilities of region-based, shape-aware, and pixel-wise distribution loss approaches at once. Also, the proposed loss function is compared with the popular loss functions such as binary cross entropy loss and focal loss for distribution-based approach, whereas dice loss, Tversky loss (Salehi et al., 2017) and focal Tversky loss (Abraham et al., 2019) for region-based approach. A boundary-aware loss function can also be handled to tackle the segmentation of polyps such as hausdorff distance loss (Ribera et al., 2019). This loss is a great way if an object has a contour with bounded or closed border (Ribera et al., 2019; Attouch et al., 1991). However, a neural network may not have predicted some pixels and that predicted binary object may not have a closed contour. In addition, another drawback of hausdorff is that it doesn't have adjustable parameters to penalize the false positives and false negatives. That's why, hausdorff loss function is not take into account for this thesis. In our proposed loss function, there are two adjustable hyperparameters, such as α and θ. The purpose of α is to make the function more comprehensive and adjustable due to handle shape-aware condition and semantically asymmetric similarity case, while using θ, it is possible to emphasis and weigh more on false negatives to increase recall.

The critical choice of α and θ plays an crucial role on the morphological template of polyps and on balancing the class imbalances, respectively. Another important problem is the vanishing gradient that may occur in the loss layer when using a loss function. Sigmoid function is often preferred to use for binary classification. However, the gradient of sigmoid goes to zero at some points and this situation can lead to infinitesimally small weights during backpropagation. At this point, our loss function is constructed and is outperformed popular losses in overcoming these problems. From research to production, a mobile-friendly web application is developed to make polyp segmentation and to integrate colonoscopy devices in clinical application during intervention. To do this, a frame grabber device (i.e. blackmagic mini recorder) can be used to transmit frames from the

device to an external web server based PC for providing communication in real-time to make polyp segmentation. A colonoscopy device has an endoscopy processor and for example, Fujinon EPX-4450D is one of the them to acquire and process colonoscopic white light images. Images can be captured using a frame grabber device compatible with this processor that has a HD-SDI type digital output. Then, web-server can acquire those images by a frame grabber to segment polyps.

2 LITERATURE SURVEY

Deep learning has an efficient methodology for polyp segmentation. For instance, ResUNet++ has been designed for polyp segmentation. With the use of Kvasir-Seg dataset, 81.33% dice score was acquired, while using the cvc-clinic db dataset, 79.55% dice score was achieved (Jha et al., 2019). A miss rate of 27% was obtained for serrated polyps reached (Zhao et al., 2019) in another study. Moreover, the rate of missed colorectal cancers is between 2% and 6%, shown by another study (Bressler et al., 2007).

PLPNet (Jha et al., 2020) is a CNN architecture that perform for detection task. It consists of ResNet50 as encoder and feature pyramid network (FPNet) (Lin T-Y. Et al., 2017) as the decoder part. Depending on this, PLPNet has two steps; i) polyp proposal using region proposal network (RPN) for both classification and bounding box regression, and ii) polyp segmentation using ground truth masks. For the segmentation task, fully convolutional networks (Long et al., 2015) are applied using feature pyramids and initialized the network by feature sharing from polyp proposals. Thereby, it is claimed that richer spatial contents can be acquired with a fully convolutional architecture to achieve better accuracy.PLPNet is trained with CVC-Colon DB using binary cross entropy loss and is tested with CVC-Clinic DB. Hence, 74.7% IoU and 83.9% dice scores are obtained. While using only two databases for training and testing may cause poorly generalized model, it can be deduced that there is a need to develop a model that performs better due to the low success rate when compared with our study.

ColonSegNet is another architecture for detection, localization, and segmentation at once (Jha et al., 2021). It consists of an encoding - decoding neural network that include residual building blocks with squeeze and excitation network. ColonSegNet has both less trainable parameters and multifunctional specialty, and it achieved a dice score of 82.06%, 84.35% precision, 84.96% recall and 82.06% F2 score using Kvasir-SEG dataset for the case of cross entropy and dice loss combination.

There is another study to segment polyps using CVC-Clinic DB dataset, U-Net architecture, and focal loss (Yeung et al., 2021). With the hyperparameters used in (Lin et al., 2017), a dice score of 86.8%, intersection over union score of 79%, precision of 84.4% and recall of 93.3% are acquired. With the same model concept, 87.4% dice score, 79.6% intersection over union, 86.4% precision and 90.9% recall are achieved with using Tversky loss while using focal Tversky loss, 89.4% dice, 83.1% intersection over union, 89.6% precision and 91.9% recall are obtained (Yeung et al., 2021). In another study, PolypSegNet architecture is developed and a dice score of 84.04%, intersection over union score of 77.83%, precision of 95.06% and recall of 85.11% by using Tversky loss are achieved. Likewise, in the same study, a dice score of 84.79%, intersection over union score of 78.32%, precision of 95.71% and recall of 84.34% are achieved by focal Tversky loss (Mahmud et al., 2020).

3 PUBLICLY AVAILABLE DATASETS

3.1 Content structure of the datasets

KUDO (pit pattern) classification (Cassinotti et al., 2020) and Paris classification (Doorn et al., 2015) are the standards for the categorization of polyps. KUDO classification is based on the surface textural pattern of polyps such as tubulo-villous adenoma, tubular adenoma and hyperplastic polyp. For Paris classification, polyps can be subclassified into morphological structures such as pedunculated, flat, or sessile type.

Convolutional neural networks can be a way to recognize a pattern and can provide a well-generalized model using different type of datasets. Especially, having different polyp types would provide us with creating a better generalizable learning model. According to this paradigm, four publicly available datasets are used in this thesis: CVC Clinic DB, CVC Colon DB, Etis-Larib and Kvasir-Seg datasets. These datasets include white light images with variations in size, color, shape and pattern. CVC clinic DB consists of 612 images (Bernal et al., 2015). CVC Colon DB has 380 sequential images extracted from 15 videos (Rodriquez et al., 2021; Bernal et al., 2012). Kvasir-SEG database has 1000 polyp images (Jha et al., 2020). ETIS-Larib DB contains 196 polyp images (Jha, Smedsrud and Riegler et al., 2021). The augmented versions of images in all datasets are combined and used in deep neural networks for train and validation steps. For the testing step, each original dataset we used has been evaluated as unseen or an

external dataset and the results have been obtained as shown in Table 6.5 and Table 6.6.

Table 3.1 Properties of the datasets used in our experimental study.

Database	# of polyp images	Pixel resolution	Website
CVC-Clinic DB	612	384x288	Link
CVC-Colon DB	300	500x574	Link
Kvasir-SEG DB	1000	1225x966	Link
ETIS-Larib DB	196	712x480 or 1920x1080	Link

3.2 Augmentation tecniques & "Ready-to-Feed" data preparation

In this thesis, data augmentation using "albumentations" package in PyTorch is used to only training and validation phases. The trained model is then evaluated on all original images with no augmentation for testing phase. The trend analysis of validation error should continue to decrease in parallel with the training error with tackling data augmentation approach.

For data preparation, data augmentation and data normalization steps are conducted, respectively. First, the augmentation techniques are used as follows; i) random horizontal-vertical flipping or both, ii) scaling (scaling limit as 0.1), iii) determining rotation angle among 0, 10, 45, 90, 180, 270, iv) Contrast-limited adaptive histogram equalization (CLAHE) by adjusting clip limit as 1 and tile grid size as 8 by 8. For scaling, image sizes are rescaled with a multiplier factor of 0.1. For flipping case, the rows and columns of an image are symmetrically flipped depending on the horizontal and vertical cases. For CLAHE, each image is divided by 8x8 grids and then histogram equalization is applied to each local subimage or grid. Contrast limiting or thresholding value as clip limit is applied to each grid's histogram to enhance the contrast. Then bilinear interpolation is applied to whole-slide image to make good combination of grids. The second crucial step is to prepare data to input to the neural network by normalization process to provide computational ease and to avoid overfitting issue. To do this, each input image at the beginning of pretrained ResNet18 is normalized using the mean values of each RGB channels as 0.485, 0.456, 0.406 and standard deviations as 0.229, 0.224, 0.225, respectively. Hence, each image is normalized by the equation (1) (Li et al., 2022).

$$x_{i,j} = \frac{(x_{i,j} - \mu_c)}{\sigma_c} \quad (1)$$

where (i, j) represents the pixel coordinate, μ_c refers to the mean of corresponding channel (c) value as red, green, and blue, and σ_c is the standard deviation with related to each channel.

4 PROPOSED DEEP LEARNING MODEL

4.1. Training CNN models

For training of a network, CNN have many expedients when a problem gets more complex. Deeper layers in neural networks can be optimal to overcome obstacles for making an image segmentation.

Even if there are difficulties in the use of the colonoscopy device during imaging depending on a colonoscopist's experience, a polyp should be able to be detected as the target object by a decision-making system. At this point, two different scenarios have been conducted for polyp segmentation in this thesis.

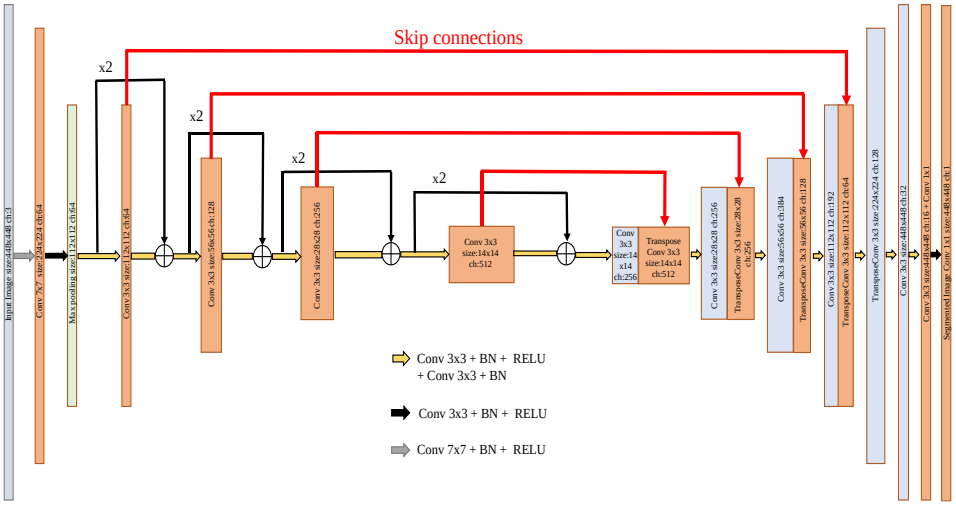


Figure 4.1 The training pipeline of the combination of ResNet18 and UNet.

First, an 18 layered residual network as decoder and UNet as decoder is implemented. Then, as a second stage, a 34 layered residual network as an encoder and a UNet as decoder are connected to perform a polyp segmentation model. Residual neural networks are deep neural networks and one of their abilities is to use residual blocks in terms of getting an identity map. It has more comprehensive and rich spatial image content due to using its input reference. The other important benefits of ResNets provide us with the following strengths: 1) To accelerate the speed of training of the networks, 2) To increase depth of the network that results in less extra parameters when comparing with widen networks, 3) To reduce the effect of vanishing gradient problem and, hence

achieve a good generalization performance (He et al., 2016). That's why, residual neural networks are preferred to use to extract meaningful feature maps in this study. A residual neural network is a neural network of building on constructs known from pyramidal cells in the cerebral cortex (Wen et al., 2018, Meijer et al., 2019). For modelling approach, ResNets are based on a residual learning that constructs a building block to add the output from the previous layer to the layer ahead.

The objective of this thesis is to build a novel loss function beyond making comparisons with state-of-the-art loss functions. Our loss function, called omni-comprehensive loss, is proposed and integrated to CNN models for overcoming both class imbalance and the gradient vanishing issue. Another reason of constructing a new paradigm on loss function is to be able to produce multiple evaluations of region-based, shape-aware, and pixel-wise distribution loss approaches at once. Hence, aside from the popular loss functions used by many studies in the literature, a more comprehensive one has been built in this thesis. Also, the choice of optimum hyperparameters used in a loss function has a critical importance (Eelbode et al., 2020).

The performance of deep neural networks can be affected by these criteria; data preparation, using a suitable architecture, learning "from scratch" or using "transfer learning", using an optimal loss function, and tunable hyper-parameters. In the training phase, the combination of an 18 layered ResNet and a UNet (Ronneberger et al., 2015) are used and the model is fed with pre-trained imagenet weights based on transfer learning.

ResNet18+UNet architecture consists of an encoder network and decoder network that are connected by skip connections (see APPENDIX A for source code). In this model architecture, while ResNet is used as a backbone, the decoder part of the UNet architecture is combined to this architecture. The reason for using ResNet18 as the backbone is to make the model faster and less complex than those, which contain more complicated encoder parts like ResNet50, ResNet101 and ResNet152. ResNet34 model, on the other hand, is used as the backbone in the second scenario to see whether the use of omni-comprehensive loss increases the performance of a simpler model, i.e., ResNet18, to a level which is close to that obtained with a more complex model, i.e., ResNet34.

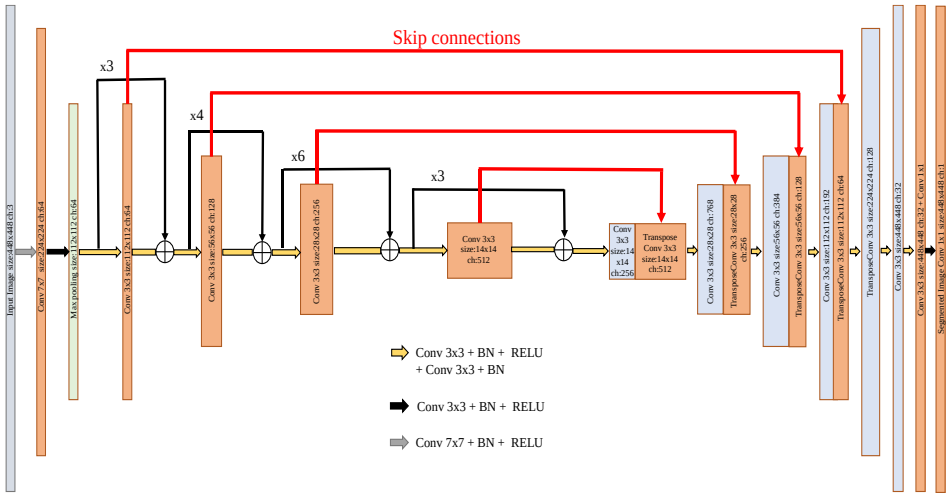


Figure 4.2 The training pipeline of the combination of ResNet34 and UNet.

The use of decoder part of UNet is to obtain a good segmentation result with the use of transpose convolutions and skip connections. Transpose convolutions have learnable parameters that positively effect the model performance. In residual blocks, the specified numbers of repeated operation to reach better spatial resolution are applied as shown in Figure 4.1 and Figure 4.2. Also, Rectified linear unit (ReLu) activation function has been used to overcome the overfitting and the gradient vanishing issue. As shown in Figure 1 and Figure 2, the meaning of operations x2, x3, x4 and x6 are the number of times to apply the respective techniques (He et al., 2016). Hence, ResNet18 includes 18 convolutional layers, whereas ResNet34 includes 34 convolutional layers. As shown in Figure 4.2, four different identity shortcuts in the encoder part are used as building blocks. Just as in ResNet18, pre-trained imagenet weights are used to initialize the network for producing feature maps and then multiple feature maps are concatenated by skip connections. The combinations of 3x3 convolutional filters, batch normalization and ReLu activation functions are used during the training of the pipeline. This way, the vanishing gradient problem can be prevented because the derivative of ReLu doesn't go to zero. At the end of the network, sigmoid function is used to calculate the probabilistic value of each pixel. Depending on this, the backpropagation algorithm is performed using the loss function. In addition, each probabilistic map is thresholded by 0.5 and binarized to calculate dice score and intersection over union using target mask as shown in Figure 4.3.

In this thesis, all images are resized as $448x448$, convolved with pretrained imagenet weights using 64 different 7x7 convolutional filters at the beginning of ResNet18. In the decoder part, UNet with the contributions such as a kernel size of 3x3 and the specified channel numbers are implemented as shown in Figure 4.1

and Figure 4.2. Bridge connections function as a network binding to provide both concatenation of image channels and translation invariance property. For each passing step in the decoder part, feature maps are upsampled by transpose convolutions. At the output of the network, a sigmoid function is used to predict the latest feature or probabilistic map. Then the loss function is calculated between probabilistic maps and target masks for weight updation. Another deep neural network architecture is also used with ResNet34 and Unet combination (see APPENDIX B for source code) for the comparison of performance metrics. In this thesis, six different losses are functioned in the loss layer, including omni-comprehensive loss and the performances are compared as shown in Table 6.1 and Table 6.2.

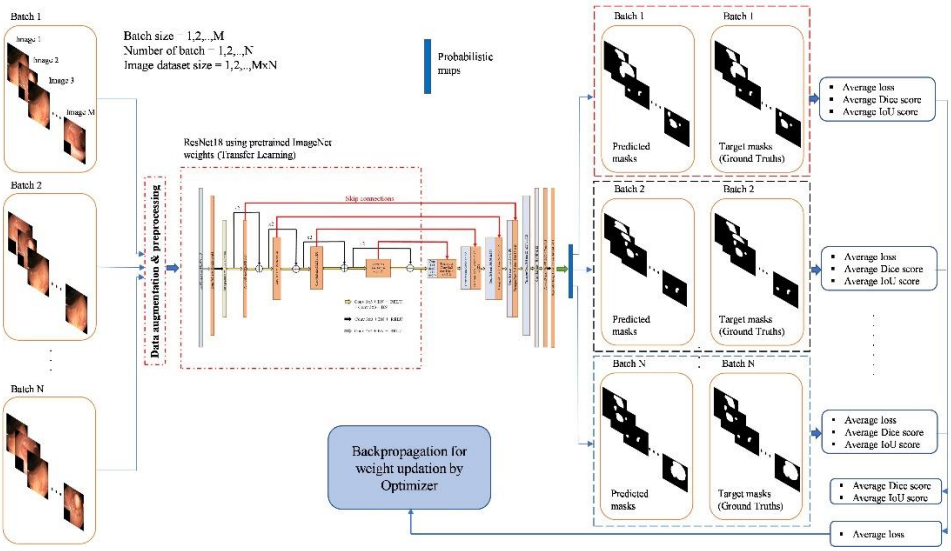


Figure 4.3 A flowchart for data augmentation and training-validation process.

For validating the trained model at each iteration, different image sets are used to calculate dice and intersection over union scores between the predicted map and the target one. To do this, first step is to calculate true positives (TP), false positives (FP), and false negatives (FN) that represent how many of pixel values matched between predicted mask and target mask as illustrated in Figure 4. According to the validation scores, early stopping criteria, weight decay regularization, and hyperparameter optimization are also taken into consideration. For example, learning rate is updated with a multiplication factor of 0.1 as the learning rate decay if there is no improvement at the end of each three epochs. Likewise, the training process is terminated if there is no improvement or decreasing situation of loss scores at the end of each ten epochs. Also, decoupled weight decay regularization is used for adaptive gradient descent optimization. At the end, the performances of loss functions are compared to decide which one is the best in this thesis.

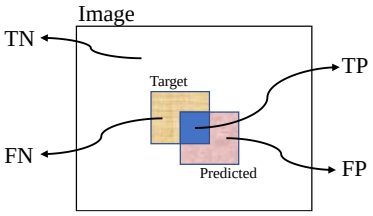
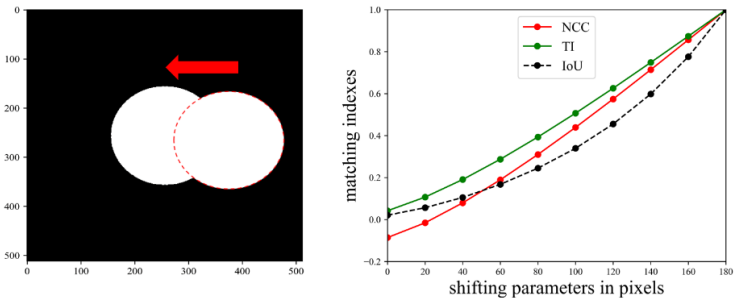


Figure 4.4 An illustration of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN).

4.2 *Omni-Comprehensive* Loss Function

The important novelty and the challenge of this thesis is to produce *Omni-comprehensive* loss function (see APPENDIX C for source code) that is developed for creating a new paradigm to overcome both class imabalanced datasets and the gradient vanishing issue. Another reason of developing this novel loss function is to be able to produce a more comprehensive one that has calculation capabilities of region-based, shape-aware, and pixel-wise distribution loss approaches at once. As known, there are critical priority steps to better train deep neural networks. Loss layer has a critical importance in achieving this as the heart of learning stage. In the loss layer, a loss functional model is used to be able to produce a generalizable deep learning model. The main objective should be comprehensively serving as a pattern recognition of morphological structures, pixel-manner distributions, and shape-sensitive condition in an image at once in terms of providing a fair and more generalizable model.



(a) (b)

Figure 4.5 (a) Illustration of matching between circle and shifted circle masks. (b) Matching index values corresponding to shifting parameters in pixels.

As mentioned in Section (1), class imbalance and the vanishing gradient problem should be considered when building a loss function to achieve better accuracy and generalization results. In this thesis, a loss is modeled considering

three components, which are i) normalized cross correlation coefficient (NCC), (ii) Tversky index (TI) and (iii) binary cross entropy (BCE). The aim of all matching indices used in this thesis would be to keep improving their predictions with each passing epoch, until the predicted and target images perfectly overlap. BCE handles in a pixel wise distribution-based manner. Similarity indices like NCC, TI and IoU related to two same-sized circle images are calculated as illustrated in Figure 4.5(b). To do this, each circle is shifted 20 pixels to the left at each step as shown in Figure 4.5(a), then, the related indices are plotted on the graph. Finally, the relationships among the indices are evaluated. Whole-slide shape-sensitive based normalized cross correlation (NCC) can be used to determine how much matching there is between two images, and it is possible to create an index between them. As it is known, NCC matching index is in the range of -1 and 1. As the similarity between two images increases, the index approaches 1 which means strongly positive correlation. When it approaches to -1, it means strongly negative correlation, and 0 means no correlation exists. TI denotes a region-based similarity and is a generalizable form of dice coefficient. Also, it has a tunable parameter that weights false positives and false negatives due to the case of both input-class imbalance and inter-input-class imbalance as shown in equation (8).

Moreover, this novel method is proposed based on the use of both NCC and Tversky indices. Intersection over union (IoU) is used for describing an overlap ratio of two images between predicted image and target one. As shown in Figure 4.5 (b), the matching indices of NCC, TI and IoU are plotted and it is observed that there is a positive relationship between them and by summing NCC and TI terms as indices and then subtracting from 1 due to calculating a loss, the functionality of both shape-sensitive and region-based loss is provided together. NCC can be used for measuring the similarity between an image and a template in a way of template matching (Huang et al., 2022). A template can be a part of an image that contains a target object or can be a whole slide image. Also, it can be used as a loss function. It has been observed that the NCC index can be considered in a shape-sensitive manner by performing trend analysis as shown in Figure 4.5 (b). In equation (2), BCE plays important roles as a stabilizer and a multiplier factor. The aim of using BCE as a feature of pixel-level distribution-based is to control the omni-comprehensive loss to provide the minimization of that, overcoming the vanishing gradient problem. Equation (3) represents the three components of omni-comprehensive loss.

$$\iint(\alpha, \theta, p_i, t_i) = [1 - (\alpha * \mathrm{NCC} + (1 - \alpha) * \mathrm{TI})] * \mathrm{BCE} \quad (2)$$

$$\mathrm{NCC} = \frac{\sum_i^N (p_i - \mu_p)(t_i - \mu_t)}{N\sigma_p\sigma_t},\ \mathrm{TI} = \frac{1 + \sum_i^N p_i t_i}{1 + \sum_i^N p_i t_i + \theta \sum_i^N p_i(1 - t_i) + (1-\theta)\sum_i^N t_i(1 - p_i)} \quad (3a)$$

$$\mathrm{BCE} = -\frac{1}{N}\sum_i^N t_i \log(p_i) + (1 - t_i)\log(1 - p_i) \quad (3b)$$

where N is the flattened image size of 448x448, p_i is the one-dimensional flattened array of predicted image, t_i is the one-dimensional flattened array of the target image, μ_p is the average value of the predicted image, μ_t is the average value of the target image, σ_p and σ_t are the standard deviations of p and t images, $\mathrm{TP} = \sum_i^N p_i t_i$, $\mathrm{FP} = \sum_i^N p_i(1 - t_i)$, $\mathrm{TN} = \sum_i^N (p_i - 1)(t_i - 1)$, and $\mathrm{FN} = \sum_i^N (1 - p_i)t_i$.

α parameter weights NCC and TI, and θ parameter weights FPs and FNs in Tversky index due to the class imbalance issue, so β provides to optimize the trade-off between precision and recall. The parameters of α and θ for omni-comprehensive loss function take values between 0 and 1. For example, in this study, the optimal values of α and θ are set to 0.5 and 0.1, respectively. In our case, θ is focused more on false negatives. The values of hyperparameters in our loss function, α and θ, are selected by considering tuning the shape-sensitive case and class imbalance ratio (89.6% for 0-labeled class and 10.4% for 1-labeled class) by Tversky index, respectively. It is observed that using lower θ in our democratized loss function in training led us to decrease FNs and to boost recall. For BCE, the first part, $t_i \log(p_i)$, is penalized on false negatives whereas the second part, $(1 - t_i)\log(1 - p_i)$, is punished part of the false positives during training. Through all the tunable hyperparameters used in our loss function, It is handled to provide the necessary and the sufficient conditions such as, 1) penalizing on FNs and FPs, 2) Balancing the class imbalance case, 3) Addressing the shape-aware approach, 4) Providing pixel-manner distribution loss function, and 5) Overcoming the gradient vanishing issue. Thus, the loss function is modeled, considering these properties without using any other additional hyperparameters that may cause a risk of slow execution of model training.

$$\frac{\partial \int}{\partial \mathbf{w_i}} = \int' = X'Y + Y'X, \text{ where } X = \left[1 - \frac{\mathrm{NCC} + \mathrm{TI}}{2}\right], Y = \mathrm{BCE} \quad (4)$$

The vanishing gradient problem occurs when the derivative of a function goes to zero at some values. This situation may cause reusing of feature maps due to unchangeable or very smooth variation of weights during backpropagation. For both architectures used in this study, the pipeline of the neural networks till the loss layer uses ReLu activation function. Derivative of ReLu activation function never goes to zero, so the problem is solved by our loss function during backpropagation. As seen in equation (4), the outcome is the derivative of our loss function that is needed to update weights during backpropagation. The derivative

of the sum of NCC and TI is taken into account with the contribution of BCE to overcome the vanishing gradient problem caused by the sigmoid function as its derivative goes to zero at some points. BCE functions as a stabilizer factor for considering pixel-level distribution in our loss function. It is possible to fix the effects of class imbalance problem with the contribution of NCC and TI, and to also fix the vanishing gradient problem with the contribution of BCE, in the output of the deep neural network.

As seen in equation (5), the derivative of BCE, Y', never goes to zero. For proof of this, let's assume that $h = \mathbf{w}X + b$ and $p = \sigma(h) = \frac{1}{1+e^{-h}}$, where $\mathbf{w}$ is the filter weight, X is the input image, b is the bias factor, $\hat{p}$ is the predicted output of sigmoid function and t is the target value. If we say that BCE is a function of w and denoted by $J(\mathbf{w})$, then according to chain rule;

$$Y' = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial J(\mathbf{w})}{\partial p}\frac{\partial p}{\partial h}\frac{\partial h}{\partial \mathbf{w}} = \frac{p-t}{p(1-p)}\, p(1-p)\, X \tag{5a}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = (p-t)X \tag{5b}$$

Likewise, the derivative of NCC, NCC', as a function of $\mathbf{w}$ and let's assume that NCC denoted by $\theta(\mathbf{w})$;

$$NCC = \frac{\sum_i^N (p_i-\mu_p)(t_i-\mu_t)}{N\sqrt{\frac{\sum_i^N (p_i-\mu_p)^2}{N}}\sigma_t} \tag{6a}$$

$$NCC' = \frac{\partial \theta(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial \theta(\mathbf{w})}{\partial p_i}\frac{\partial p_i}{\partial h}\frac{\partial h}{\partial \mathbf{w}} = \frac{\partial \theta(\mathbf{w})}{\partial p_i}\, p_i(1-p_i)\, X; \tag{6b}$$

$$\frac{\partial \theta(\mathbf{w})}{\partial p_i} = \frac{1}{N\sigma_t}\frac{\partial}{\partial p_i}\left[\frac{\sum_i^N (p_i-\mu_p)(t_i-\mu_t)}{\sqrt{\frac{\sum_i^N (p_i-\mu_p)^2}{N}}}\right] \tag{6c}$$

$$\frac{\partial}{\partial p_i}\left[\frac{\sum_i^N (p_i-\mu_p)}{\sqrt{\frac{\sum_i^N (p_i-\mu_p)^2}{N}}}\right] = \frac{\sum_i^N (t_i-\mu_t)\sqrt{\frac{\sum_i^N (p_i-\mu_p)^2}{N}} - \sum_i^N (p_i-\mu_p)(t_i-\mu_t)\frac{1}{2\sqrt{N}\sqrt{\sum_i^N (p_i-\mu_p)}}}{\frac{\sum_i^N (p_i-\mu_p)^2}{N}} \tag{6d}$$

$$NCC' = \frac{\sum_i^N (t_i-\mu_t)\sqrt{\frac{\sum_i^N (p_i-\mu_p)^2}{N}} - \sum_i^N (p_i-\mu_p)(t_i-\mu_t)\frac{1}{2\sqrt{N}\sqrt{\sum_i^N (p_i-\mu_p)}}}{\frac{\sum_i^N (p_i-\mu_p)^2}{N}} . p_i(1-p_i).X \tag{6e}$$

In equation (5b), the derivative of BCE doesn't contain the derivative of sigmoid function. In addition, the derivative of NCC is also calculated as seen in equation (6a-6e). As seen in equation (8), constant 1 has been added to the numerator and denominator in the Tversky index formulation. The reason for this is to get rid of the division by zero problem and, hence to overcome the vanishing

gradient problem. The NCC formula alone does not overcome the vanishing gradient problem as seen in equation (6e) because of the derivative of sigmoid function still remaining. However, this formula is added to the Tversky index to solve the problem. Moreover, the contribution of BCE with a multiplicative manner has been provided to guarantee completely eliminate the vanishing gradient problem. Hence, it can be said that the derivative of the *omni-comprehensive* loss has no way to be zero when combining all components of the loss. At this point, AdamW optimization algorithm is applied to update filter weights during backpropagation (Loshchilov et al., 2019).

4.3 Popular Loss Functions

A loss function should sense of imbalance distribution among different classes. In this thesis, imbalanced datasets are used to balance by the popular loss functions. Almost each polyp image has a large area of background and has a small part of foreground which is polyp region. This situation may lead to two negative outcomes: 1) high bias factor in training model and 2) a poorly generalized model.

Focal loss (FL) functions as a popular loss function for dealing with class-imbalance problem. It emphasizes more on background samples to down-weight the contribution of true negatives during training. As can be seen in equation (7), there are two hyperparameters that can be tackled to fine-tune the function. These are α as the balanced factor and γ as the controlling parameter between foreground and background. Also, it can be said that focal loss is an extended version of cross entropy. After trying three different cases of the parameter pairs for our experimental procedure, the optimum values of α are set to 0.25 and of γ to 2.0 based on the best produced performance of FL using equation (7).

$$\mathrm{FL}(\mathrm{p_t}) = -\alpha_t(1-\mathrm{p_t})^{\gamma}\, log(\mathrm{p_t}) \tag{7}$$

where $\mathrm{p_t}$ refers to the predicted output.

Tversky loss (TL) is our second choice. It uses Tversky index that is a special case of dice similarity coefficient. Tversky's ratio model is an appropriate basis for computational approaches to semantic and asymmetric similarity, which is the comparison of objects such as images in a semantically meaningful way. The aim of using TL is to control the magnitude of penalties for FPs and FNs.

$$\text{Tversky index} = \frac{\mathrm{TP}}{\mathrm{TP}+\boldsymbol{\alpha}\mathrm{FP}+\boldsymbol{\beta}\mathrm{FN}};\ \text{Tversky loss} = 1-\left[\frac{1+\mathrm{TP}}{1+\mathrm{TP}+\boldsymbol{\alpha}\mathrm{FP}+\boldsymbol{\beta}\mathrm{FN}}\right)] \tag{8}$$

where α and β are the adjustable parameters. α is set to 0.1 in the range of [0,1] and β is set to 0.9 in the range of [0,1] in equation (8) in our study. Also, constant 1 is added to both numerator and denominator to avoid *zero divided by zero* and *division by zero* errors (Jadon et al., 2020).

In the case of using Focal Tversky loss (FTL), it has an extra controllable parameter,γ, when comparing with Tversky loss. This parameter controls the balance between background and foreground regions. It is possible to say that FTL also balances the trade-off between precision and recall, so it is possible to weight FNs more than FPs to prevent high precision and low recall values as seen in equation (9).

$$\text{Focal Tversky loss} = [(1 - \text{Tversky index})]^{\left(\frac{1}{\gamma}\right)} \quad (9)$$

where, α=0.1 in the range of [0,1], β=0.9 in the range of [0,1] and γ=1.33 in the range of [1,3] are selected using equation (9). For dice loss, it is produced from dice coefficient (DC), also known as the Sørensen dice index, that it measures the similarity between two images. Tversky index represents a dice coefficient when α and β are set to equal number as 0.5 as seen in equation (8). DC is simply formulated as $2TP/(2TP + FP + FN)$ and hence dice loss becomes 1-DC (Jadon et al., 2020).

5 EXPERIMENTAL PROCEDURE

5.1. Carbon Footprint of Training the CNN models

The development of deep learning (DL) algorithms and hardware developments are progressing rapidly day by day. Such rapid developments have led to the need for powerful hardware. This situation affects energy consumption. Ensuring energy efficiency with the use of DL algorithms will make a significant positive contribution to climate change. In terms of carbon emission, it is possible to mitigate the energy consumption with low algorithm complexity and low-capacity hardware processing units. With this awareness, keeping the algorithm complexity low can be seen as an important improvement. Moreover, there are other factors that affect the carbon footprint such as how many hours the hardware works, region and cloud provider. In this study, the open source "carbontracker" tool is used. Thus, energy consumption estimation and tracking were made during the training of DL models. The carbon footprint is predicted using forecasted carbon intensity with an API used in the tool. The carbon intensity varies according to the region (country) lived in, and this also affects the carbon footprint during training of a DL model. The aforementioned API has an ability to

fetch IP address of a user who make training a DL model, and thus the region can be found. Combining FLOPS (floating point operations per second), which is one of the parameters that can represent algorithm complexity, and some GPUs can provide higher efficiency, as well as have the effect of reducing carbon emissions. Converting energy consumption to carbon emission can be calculated using the following formulations respectively (Anthony et al., 2020).

$$PUE\ (Power\ Usage\ Effectiveness) = \frac{Total\ Facility\ Energy}{IT\ Equipment\ Energy} \quad (10a)$$

$$Enegy\ Consumption = PUE \sum_{e\epsilon\xi} \sum_{d\epsilon D} P_{avg,de} T_e \quad (10b)$$

$$Carbon\ Footprint = Energy\ Consumption\ x\ Carbon\ intensity \quad (10c)$$

where $P_{avg,de}$ is the average power consumed by used device, $d\epsilon D$ in epoch $e\epsilon\xi$. T_e is the training time for epoch e. PUE is the ratio of the total energy used in a server or a data center to the energy used by IT equipment that are hardware and network equipment. Carbontracker uses 1.58 as constant value for PUE. The predicted carbon emission results are obtained developing a code function as seen in APPENDIX D section.

Carbon emission result for ResNet18+UNet CNN model

CarbonTracker: The following components were found: GPU with device(s) NVIDIA GeForce GTX 1070 with Max-Q Design.
epoch: 1 --- step: train --- time: 15:07:29
100%|| 238/238 [01:32<00:00, 2.58it/s]
Loss: 0.2749 --- dice: 0.4308 --- IoU: 0.3203
epoch: 1 --- step: val --- time: 15:09:01
100%|| 60/60 [00:12<00:00, 4.94it/s]
Loss: 0.1579 --- dice: 0.4974 --- IoU: 0.3911
******** Saved optimum case ********

CarbonTracker: Average carbon intensity during training was 294.21 gCO2/kWh at detected location: İzmir, İzmir, TR.
CarbonTracker:
Actual consumption for 1 epoch(s):
Time: 0:01:45
Energy: 0.002224 kWh
CO2eq: 0.654208 g
This is equivalent to:
0.005434 km travelled by car
CarbonTracker:
Predicted consumption for 100 epoch(s):
Time: 2:54:37
Energy: 0.222364 kWh
CO2eq: 65.420772 g
This is equivalent to:
0.543362 km travelled by car
CarbonTracker: Finished monitoring.

Carbon emission result for ResNet34+UNet CNN model

CarbonTracker: The following components were found: GPU with device(s) NVIDIA GeForce GTX 1070 with Max-Q Design.
epoch: 1 --- step: train --- time: 15:11:19
100%|| 238/238 [01:41<00:00, 2.33it/s]
Loss: 0.2930 --- dice: 0.3321 --- IoU: 0.2421
epoch: 1 --- step: val --- time: 15:13:01

```
100%|██████████| 60/60 [00:11<00:00,  5.30it/s]
Loss: 0.1416 --- dice: 0.4657 --- IoU: 0.3677
******** Saved optimum case ********

CarbonTracker: Average carbon intensity during training was 294.21 gCO2/kWh at
detected location: İzmir, İzmir, TR.
CarbonTracker:
Actual consumption for 1 epoch(s):
    Time:      0:01:54
    Energy:   0.002695 kWh
    CO2eq:   0.793022 g
    This is equivalent to:
    0.006587 km travelled by car
CarbonTracker:
Predicted consumption for 100 epoch(s):
    Time:      3:09:50
    Energy:   0.269546 kWh
    CO2eq:   79.302204 g
    This is equivalent to:
    0.658656 km travelled by car
CarbonTracker: Finished monitoring.
```

6 RESULTS

In this thesis, it is aimed to provide a performance comparison by evaluating popular loss functions, instead of comparing different state-of-the-art CNN models. As it is known, dice loss, Tversky loss and focal Tversky loss studies are produced respectively by using dice coefficient, Tversky index and focal loss, all of which already exist in the literature. The most important emphasis of our study is to provide the performance evaluation of our proposed loss function. To do this, two approaches have been conducted. 1) To compare different loss functions using the same CNN model, 2) To compare the same loss functions using a second CNN model. Thus, a general performance evaluation is provided for CNN models with two different algorithm complexities.

Because in this study, it is not aimed to achieve high success rates by choosing CNN models with very high complexity. Instead, the models that have low complexities are preferred among the residual neural network architectures as backbone. Thereby, it is provided to achieve higher success rates by using our loss function compared to existing popular loss functions as seen in Table 6.1 and Table 6.2. At this point, while taking into account the trade-off between accuracy and algorithm complexity, indeed, the performance of the loss function and its results it produces are focused in this study. Even though deeper residual neural networks achieve higher performance metrics, it has been preferred to use ResNets, having lower complexity for making training faster.

As seen in Table 6.5 and Table 6.6, the performance results of the state-of-the-art models are available. Multiple performance metrics are handled in this study. The reason for this is to provide reliable and comprehensive analysis when evaluating the performance of a CNN model (Hicks et al., 2022). However, recent studies including the state-of-the-art models do not have such multiple performance metrics or success criteria as shown in Table 6.5 and Table 6.6.

Table 6.1 Performance comparison of losses using ResNet34 + UNet architecture.

Validation scores	Dice	IoU
Omni-comprehensive loss (α=0.5, θ=0.1)	**0.8767**	**0.8163**
Omni-comprehensive loss (α=0.3, θ=0.3)	0.8537	0.7840
Omni-comprehensive loss (α=0.7, θ=0.3)	0.8459	0.7757
BCE loss	0.8721	0.7984
Dice loss	0.8303	0.7597
Focal loss (α = 0.25, γ=2.0)	0.8433	0.7711
Tversky loss (α = 0.1, β=0.9)	0.7747	0.6747
Focal Tversky loss (α = 0.1, β=0.9, γ=1.33)	0.7919	0.6952

Table 6.2 Comparison of loss functions using ResNet18 + UNet architecture.

Validation scores	Dice	IoU
Omni-comprehensive loss (α=0.5, θ=0.1)	**0.8653**	**0.8046**
Omni-comprehensive loss (α=0.3, θ=0.3)	0.8588	0.79
Omni-comprehensive loss (α=0.7, θ=0.3)	0.8642	0.8043
BCE loss	0.8605	0.7931
Dice loss	0.8443	0.7760
Focal loss (α = 0.25, γ=2.0)	0.8135	0.7396
Tversky loss (α = 0.1, β=0.9)	0.7689	0.6662
Focal Tversky loss (α = 0.1, β=0.9, γ=1.33)	0.7669	0.6653

Dice scores and other CNN model properties are given in Table 6.3.

Table 6.3 Summarized parameters of the training models.

Architecture	# of epochs	#Params (M)	Validation Dice score (%)	Loss function	GFLOPS
ResNet18+UNet	39	14.3	81.35	Focal loss	16.6
ResNet34+UNet	40	24.44	84.33	Focal loss	24.02
ResNet18+UNet	41	14.3	**86.53**	**Omni-comprehensive loss**	16.6
ResNet34+UNet		24.44	**87.67**	**Omni-comprehensive loss**	24.02

As can be seen in Figure 6.1, training and validation sets are separately handled for each epoch and loss values are accordingly calculated. When doing this, the loss function is calculated by the use of the training set and the backpropagation is conducted. Next, the predicted mask is produced based on the validation set. Finally, the validation loss is calculated between the predicted and target mask by using the same loss function.

In Figure 6.2, the starting score of dice is around ~51%. In addition to this, the starting score of intersection over union is around ~40% according to our experiments. Therefore, it is observed that the NCC coefficient, thus the template matching index, never falls below zero in this experimental process. Trend analysis of losses and dices in Figure 6.2 and Figure 6.4 show that the model using omni-comprehensive loss prevents overfitting due to the tuning of hyperparameters such as batch size, learning rate, weight decay, learning decay rate and early stopping criteria.

As illustrated in Figure 6.1, Figure 6.2, Figure 6.3 and Figure 6.4, train and validation sets are separately tackled to produce loss and dice scores. The ResNet18+UNet model is evaluated and achieved as re-validation scores of 92.61% dice, 87.28% IoU, 93.81% sensitivity, 92.60% precision and 93.21% F2 score. By using ResNet34+UNet model, it is achieved as re-validation scores of 93.09% dice, 88.09% IoU, 93.27% sensitivity, 93.78% precision and 93.11% F2 score. One of our goals in this study is to keep the encoder part as low in complexity as possible compared to 34, 50, 101 and 152 layered residual neural networks. The results showed that floating point operations per second (FLOPS) of the 18-layered ResNet model is $1.8x10^9$, whereas FLOPS of 152-layered ResNet model is $11.3x10^9$.

Table 6.4 Evaluation metrics.

Dice coefficient (F1)	$\frac{2xTP}{2xTP + FP + FN}$
Intersection Over Union (Jaccard index)	$\frac{TP}{TP + FP + FN}$
Sensitivity (Recall)	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
Precision	$\frac{TP}{TP + FP}$
F2	$\frac{5xTP}{4x(TP + FN) + TP + FP}$
Matthews correlation coefficient	$\frac{TPxTN - FPxFN}{\sqrt{(TP + FP)x(TP + FN)x(TN + FP)x(TN + FN)}}$
Accuracy	$\frac{TP + TN}{TP + FP + FN + TN}$

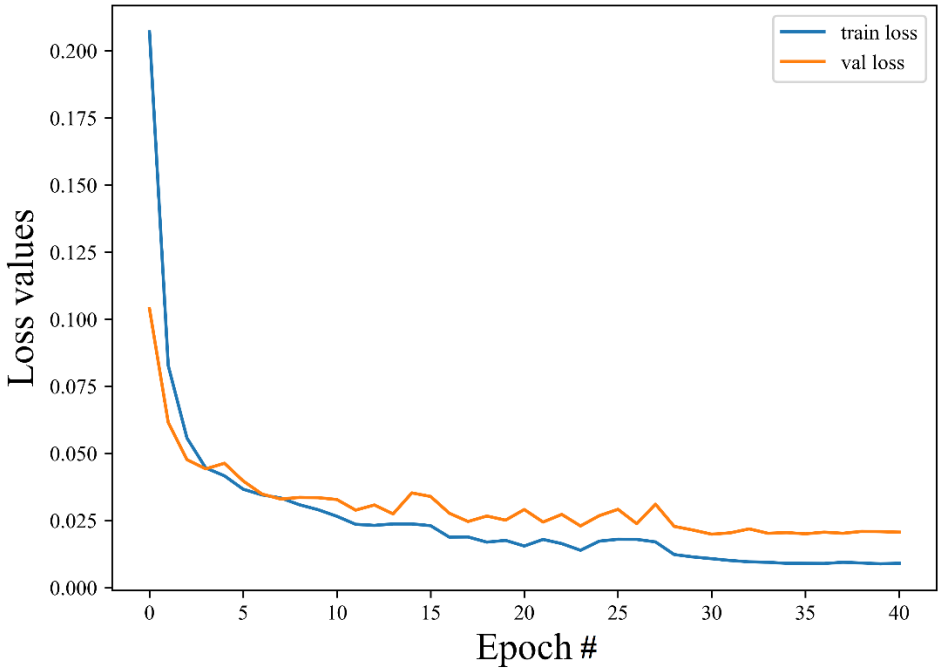


Figure 6.1 Loss scores for the case of *omni-comprehensive* loss and ResNet18-UNet architecture.

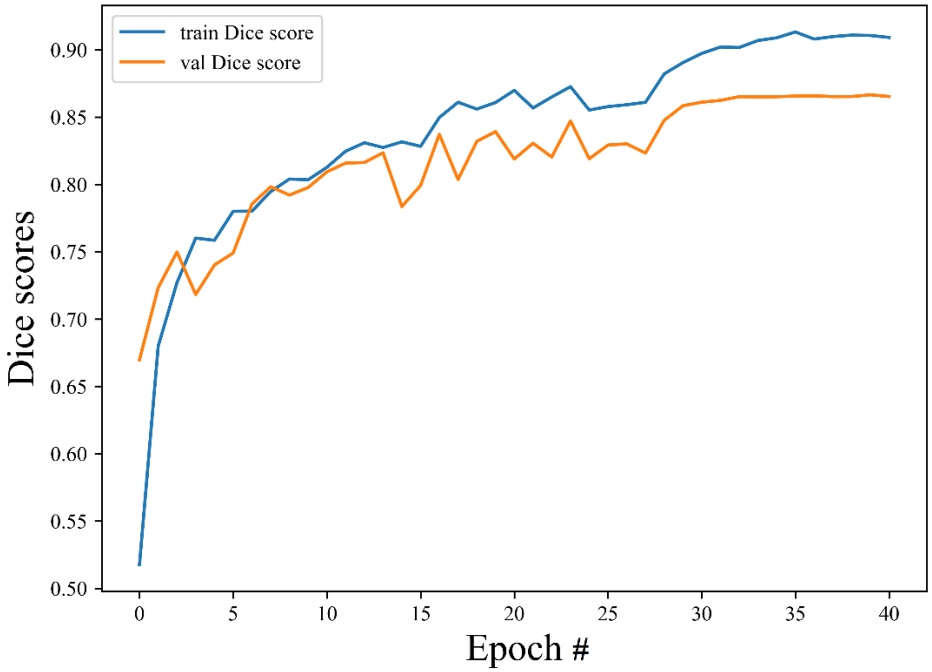


Figure 6.2 Dice scores for the case of *omni-comprehensive* loss and ResNet18-UNet architecture.

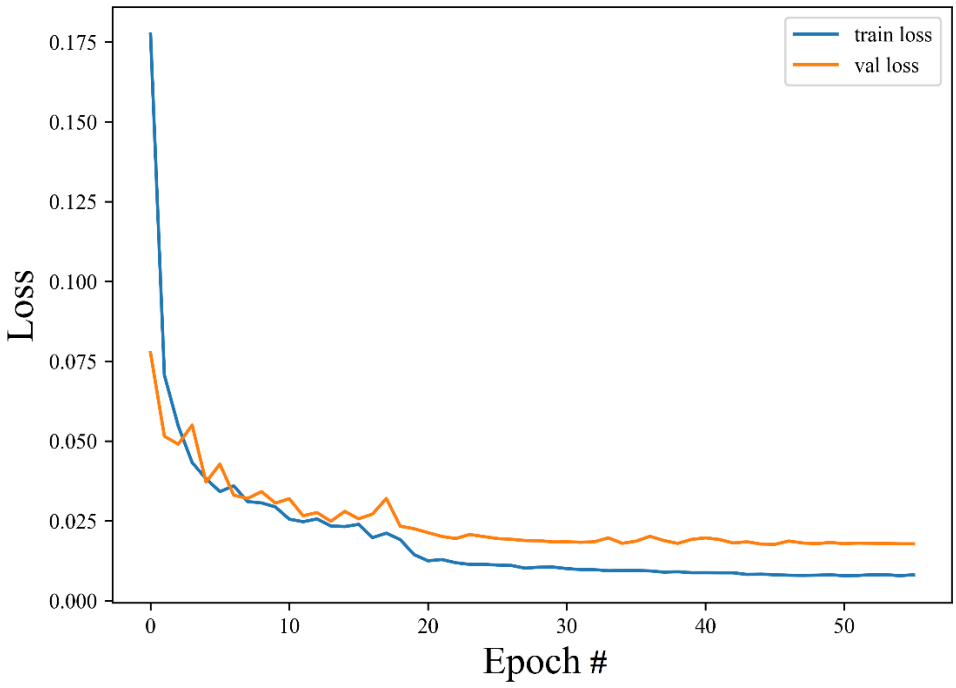


Figure 6.3 Loss scores for the case of *omni-comprehensive* loss and ResNet34-UNet architecture.

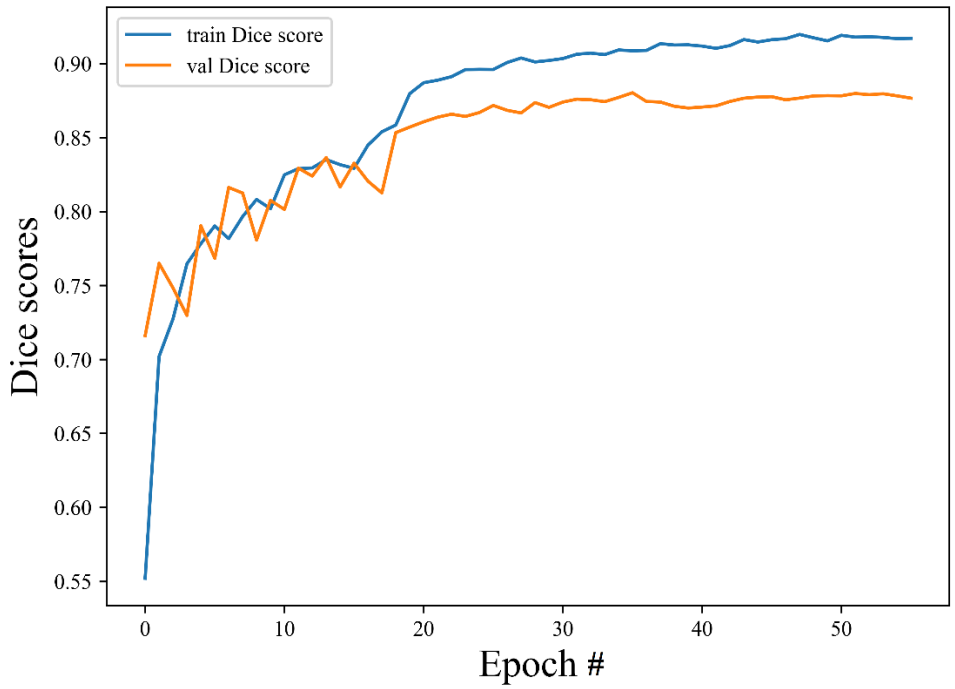


Figure 6.4 Dice scores for the case of *omni-comprehensive* loss and ResNet34-UNet architecture.

During the implementation and development of our software, giga floating point operations per second (GFLOPS) of the two CNN architectures are calculated to observe the computation-intensive of the models. 5-fold stratified cross validation is used when setting up the learning rate and early stopping criteria to prevent overfitting during training. The training process is terminated if there is no improvement on validation scores of dice and IoU at the end of each ten epochs. For our model development, the training process is performed using Nvidia GTX 1070 Max-Q design graphics card, and Python programming language is used for code development.

Table 6.5 Test performance comparison using unseen CVC-Clinic DB and CVC-Colon DB.

	External DB test scores (%)															
	CVC-Clinic DB								**CVC-Colon DB**							
	mDice	mIoU	mAcc	mSens	mSpec	mPrec	mF2	mMCC	mDice	mIoU	mAcc	mSens	mSpec	mPrec	mF2	mMCC
BLE-net [Tae et al. 2022]	-	-	-	-	-	-	-	-	73.1	63.8	-	-	-	-	-	-
SwinE-Net [Park et al., 2022]	-	-	-	-	-	-	-	-	**80.4**	**72.5**	-	-	-	-	-	-
Ours	**70.66**	**61.66**	95.54	70.83	98.71	83.16	70.02	86.40	78.71	70.03	96.53	81.81	99.05	83.30	79.94	89.52

mDice=mean dice, mAcc=mean accuracy, mSens=mean sensitivity, mSpec=mean specificity, mF2=mean F2 score, mMCC=mean Matthews correlation coefficient.

Table 6.6 Test performance comparison using unseen Kvasir-SEG and ETIS Larib DB.

	External DB test scores (%)															
	Kvasir-SEG DB								**ETIS Larib DB**							
	mDice	mIoU	mAcc	mSens	mSpec	mPrec	mF2	mMCC	mDice	mIoU	mAcc	mSens	mSpec	mPrec	mF2	mMCC
BLE-net [Tae et al., 2022]	-	-	-	-	-	-	-	-	67.3	59.4	-	-	-	-	-	-
SwinE-Net [Park et al., 2022]	-	-	-	-	-	-	-	-	**75.8**	**68.7**	-	-	-	-	-	-
Ours	**81.58**	**73.18**	94.17	86.20	96.82	84.12	82.88	87.85	71.31	62.5	98.07	79.71	98.55	73.34	75.34	87.10

mDice=mean dice, mAcc=mean accuracy, mSens=mean sensitivity, mSpec=mean specificity, mF2=mean F2 score, mMCC=mean Matthews correlation coefficient.

The aim of using small step size is to guarantee finding the global minimum of a loss function and to avoid updating learning rate many times. For the learning rate, it is advantageous to initialize with a small value. Hereby, initial learning rate is set to $5x10^{-4}$. Weight decay regularization using AdamW optimizer is also performed. The value of the weight decay is chosen as a small value, which is optimally determined as $1x10^{-5}$. Learning decay rate is used as the default value assigned as 0.1 in PyTorch. Also, beta momentum values are assigned as $\beta_1 = 0.9$ and $\beta_2 = 0.999$ in AdamW optimizer. For testing the results of the model, original

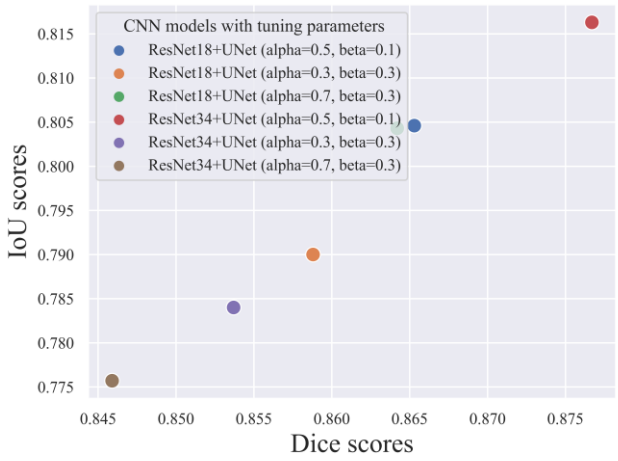


Figure 6.5 Evaluation results with different tuning parameters used in *omni-comprehensive* loss.

databases are used for making predictions and binarization. Some of the results obtained are illustrated in Figure 6.6. Our deep learning model can segment

multiple polyps, even small sized ones, at once. In the testing phase, our algorithm checks the connected contours as polyp region for each predicted binary image. In addition, our deep learning model is aware of difficult imaging conditions even with white light saturation that can be exposed on a polyp, air bubbles and so on. All evaluation metrics as shown in Table 6.4 are calculated as the average values of each unseen dataset as seen the scores in Table 6.5 and Table 6.6. It is also considered Matthews correlation coefficient (MCC) in this study. MCC is a special form of Pearson's correlation coefficient and can be used as a measure of association for binary classification in the case of imbalanced classes (Hicks et al., 2022; Chicco and Jurman, 2020). The MCC takes a value in a range of -1 and 1, where 1 means perfect prediction, 0 means random prediction with no relationship between target and predicted class, -1 means worst prediction. In Table 6.5 and Table 6.6, it is seen that the state-of-the-art models don't include many other evaluation metrics and our study has comparable good results among these studies. In Figure 6.6, It is seen that the overlaid images obtained with blending of original images and related prediction masks. The adding or blending operation can be applied using $g(x)=(1-\alpha)f_0(x)+\alpha f_1(x)$, where $g(x)$ is the blended image, f_0 is the original image, f_1 is the prediction mask, and alpha (α) is a weight factor that is assigned to 0.8 in this study. OpenCV library is used for the blending operation.

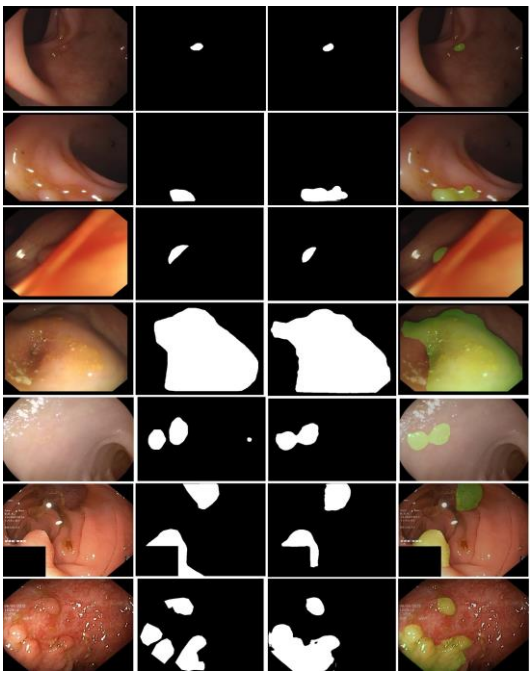


Figure 6.6 Evaluation results for the case of *omni-comprehensive* loss and ResNet18-UNet architecture. First column: original images. Second column: Ground truths. Third column: Predicted masks. Fourth column: Overlaid segmented images between original images and predicted masks.

7 PRODUCTION

The production phase of this study includes the integration of the developed polyp segmentation model into clinical application. For data acquisition, “Fujinon EPX-4450D” endoscopy processor, which is used in a colonoscopy device, can be considered. Image data can be captured using a frame grabber device compatible with this processor that has a HD-SDI type digital output. As seen in Figure 7.1, frames can be acquired from colonoscopy device by using ultra studio mini-recorder as a frame grabber. Captured frames can be read using OpenCV library thanks to an external AI server. Code snippet of frame capture using OpenCV library is included in the Appendix E section. Then, the obtained frames are given as input to the trained model to make polyp segmentation is provided. To visualize the polyp segmentation stage, the frames transferred to the server can be processed and monitored via the web application by using a monitor connected to the AI server. A mobile-friendly web application can be developed using the “streamlit” library in the Python programming language. In addition, OpenCV, PIL and PyTorch libraries are used for capturing and processing the images. Thus, such an architecture is developed from research to production. It is included the full code in the CD for both real-time video and a single image.

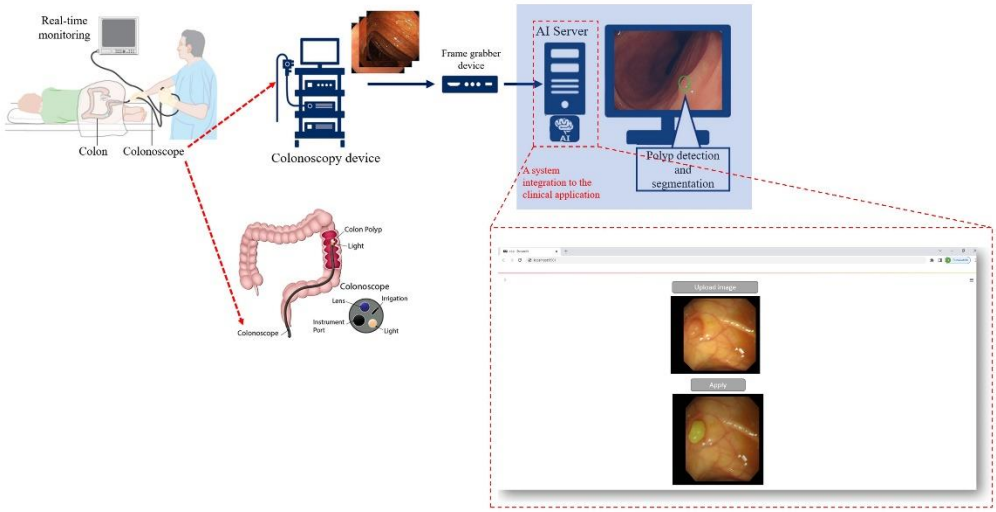


Figure 7.1 An architecture for production phase.

In this thesis, a standalone mobile-friendly web application is also developed for a second application model as a device-independent and integrable solution using streamlit library in Python programming language as seen in Figure 7.2 and Figure 7.3 (see APPENDIX F for source code).

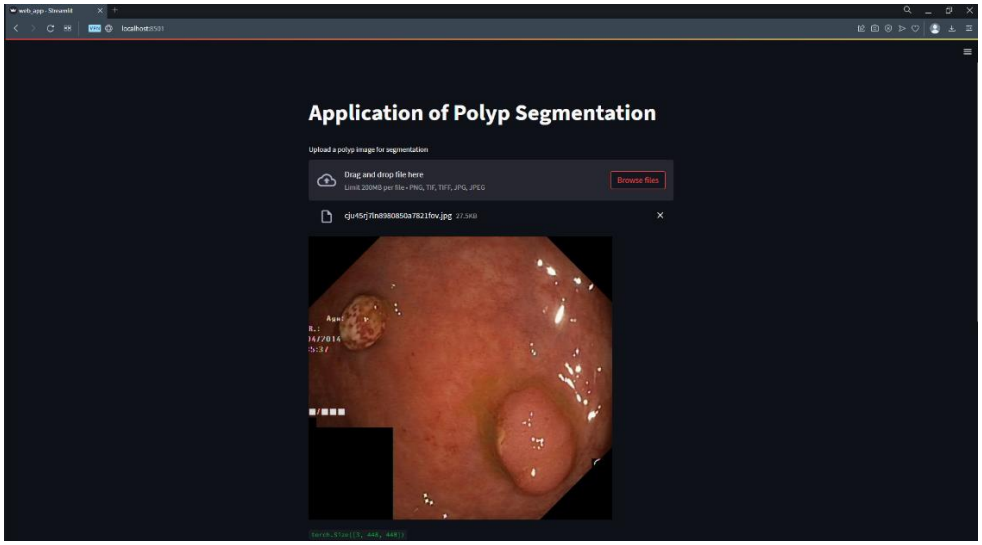


Figure 7.2 Standalone web application and polyp image choosing step.

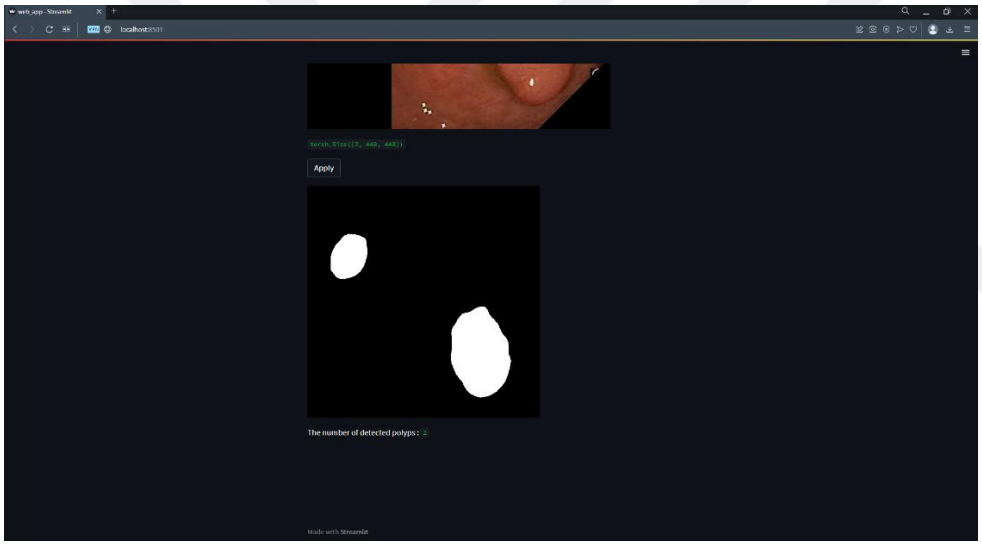


Figure 7.3 The result of both polyp segmentation and the number of detected polyps via web application.

8 DISCUSSION

Colorectal cancers may occur as a result of late detection of polyps. Colonoscopists use a colonoscopy device to remove polyps by excisional biopsy. The morphology of polyps has specific patterns that can dynamically change and may cause misdiagnosis of lesions due to the characterization of mucosal tissue from person to person. In addition, colonoscopic lightning conditions, inadequate bowel preparation, mucus on the lesion, polyps that form behind the folds, and

blind spots can increase the missed polyp rate. Despite such difficult imaging conditions, accurate and precise segmentation of polyps captured by the colonoscopy camera has a critical importance.

The aim of this study is to develop a web application that provides a polyp segmentation model that contributes to the clinical application. For polyp segmentation, a new imbalance-aware loss function, i.e. omni-comprehensive loss, has been developed to be used in deep neural networks to overcome both imbalanced datasets and the vanishing gradient problem. The second crucial reason for developing a new loss function is to be able to produce a more comprehensive one with the evaluation capabilities of region-based, shape-sensitive and pixel-wise distribution loss approaches at once. For doing such binary segmentation (as polyp and non-polyp regions), two different architectures have been conducted for observing the performance of our proposed new loss function. First, an 18-layer residual network as the backbone with UNet as the decoder is implemented. Second, a 34-layer residual network as the encoder and a UNet as the decoder are combined to perform a segmentation model.

Tunable hyperparameters play an important role. Weighting the false positives and false negatives used in the loss functions is also critical. For omni-comprehensive loss, α parameter weighs the NCC and the TI to tune the shape sensitive case and semantically asymmetric similarity case, respectively. θ parameter weighs FPs and FNs due to the class imbalance issue and it controls the trade-off between precision and recall. To do this, the parameter θ is selected by considering the class imbalance ratio (89.6% for 0-labeled class and 10.4% for 1-labeled class) and weigh more on false negatives. It is observed that using lower θ in our proposed loss function in training led us to decrease FNs and to boost recall.

Multiple publicly available datasets are used for this thesis, 5-fold cross validation, and testing steps. In addition to the original data in these datasets, augmented versions of these datasets have also been generated by flipping,rotating and contrast-limited adaptive histogram equalization operations. While the augmented samples are used both in the training and validation phases, the original datasets are tackled in the testing phase.

The networks trained with our proposed loss function achieved re-validation scores using all seen original datasets such as 92.61% dice, 87.28% IoU, 93.81% sensitivity, 92.60% precision and 93.21% F2 scores by using ResNet18+UNet model. Likewise, ResNet34+UNet has achieved re-validation scores as 93.09% dice, 88.09% IoU, 93.27% sensitivity, 93.78% precision and 93.11% F2 score. For the performance comparison of state-of-the-art mathematical models used during training, obtaining the best performance metrics among the popular imbalance-aware losses are provided as the emphasis of this study as shown in Table 6.1 and Table 6.2.

9 CONCLUSION

This thesis is focused on creating a new loss function, i.e., omni-comprehensive loss, that aims to make instance segmentation of polyps to determine the precise localization of them using colonoscopic white light images. At this point, our new loss function is proposed to overcome both the imbalanced dataset and the vanishing gradient problem. In conclusion, in this way, when the loss we proposed is used, it is possible for a simpler architecture (i.e., ResNet18+UNet) to achieve very close and even higher performance compared to a more complex architecture (i.e., ResNet34+UNet). These convolutional neural network architectures powered by our new imbalance-aware omni-comprehensive loss also exhibit the best performance compared to the other popular state-of-the-art loss functions such as dice loss, BCE loss, focal loss, focal Tversky loss and Tversky loss.

Based on the results obtained, it is hoped that this system would be helpful to colonoscopists during polyp removal. Therefore, it can be put into clinical use. As future work, it is aimed to achieve higher success rates by developing a new convolutional architecture with the following strengths: 1-) customizing the combination of residual neural network and UNet with novel methods. 2-) keeping the model complexity lower to boost the performance of the segmentation model.

REFERENCES

Abraham, N., Khan, M. N., 2019, A NOVEL FOCAL TVERSKY LOSS FUNCTION WITH IMPROVED ATTENTION U-NET FOR LESION SEGMENTATION, IEEE 16th International Symposium on Biomedical Imaging (ISBI), doi: https://doi.org/10.48550/arXiv.1810.07842

Anthony, Wolf, F., L., Kanding B., Selvan R., 2020, Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models,ICML workshop, doi: https://doi.org/10.48550/arXiv.2007.03051

Attouch, H., Lucchetti, R. and Wets, R. J. B., 1991, The topology of the ρ-hausdorff distance., *Annali di Matematica pura ed applicata* 160, 303–320. https://doi.org/10.1007/BF01764131

Bernal, J., Sánchez, F. J., Fernández-Esparrach, G. Gil D., Rodríguez, C., Vilariño, F., 2015, WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians., Computerized Medical Imaging and Graphics, doi: 10.1016/j.compmedimag.2015.02.007

Bernal, J., Sánchez, J., and Vilarino, F., 2012, Towards automatic polyp detection with a polyp appearance model. *Pattern Recognition*, *45*(9), 3166-3182pp.

Bressler, B., Paszat, LF., Chen, Z., et al., 2007, Rates of new or missed colorectal cancers after colonoscopy and their risk factors: a population-based analysis., Elsevier, Gastroenterology, doi: 10.1053/j.gastro.2006.10.027

Cassinotti, A., Fociani, P., Duca, P. et al., 2020, Modified Kudo classification can improve accuracy virtual chromoendoscopy with FICE in endoscopic

surveillance of ulcerative colitis, Endoscopy International Open, vol.8, doi: 10.1055/a-1165-0169

Chicco, D., Jurman, G., 2020, The advantages of Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, *BMC Genomics* 21, 6, https://doi.org/10.1186/s12864-019-6413-7.

Doorn, Van, C. S., Hazewinkel, Y., East, E. J. et al, 2015, Polyp Morphology: An Interobserver Evaluation for the Paris Classification Among International Experts, The American Journal of Gastroenterology, doi: 10.1038/ajg.2014.326

Eelbode, T., Bertels, J., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R., Blaschko, B. M., 2020, Optimization for Medical Image Segmentation: Theory and Practice when evaluating with Dice Score or Jaccard Index, IEEE Transactions on Medical Imaging, 3679-3690pp.

He, K., Zhank, X., Ren, S., Sun, J., 2016, Deep Residual Learning for Image Recognition., IEEE CVPR conference, doi: 10.1109/CVPR.2016.90.

Hicks, A. S., Strümke, I., Thambawita, V., Hambou, M., Riegler, A. M., Halvorsen, P., Parasa, S., 2022, On evaluation metrics for medical applications of artificial intelligence, *Sci Rep* 12, 5979, https://doi.org/10.1038/s41598-022-09954-8.

Huang, Y., Yang, X., Huang, X., Liang, J., Zhou, X., Chen, C., Dou, H., Hu, X., Cao, Y., Ni, D., 2022, Online Reflective Learning for Robust Medical Image Segmentation, MICCAI conference, doi: https://link.springer.com/chapter/10.1007/978-3-031-16452-1_62.

Jadon, S., 2020, A Survey of loss functions for semantic segmentation., IEEE, doi: 10.1109/CIBCB48159.2020.9277638.

Jha, D., Smedsrud, H. P., Johansen, D., Lange, d. T., Johansen, D. H., Halvorsen, P., Riegler, A. M., 2021, A Comprehensive Study on Colorectal Polyp Segmentation With ResUNet++, Conditional Random Field and Test-Time Augmentation, IEEE, Journal of Biomedical and Health Informatics, vol.25, doi: https://doi.org/10.1109/jbhi.2021.3049304.

Jha, D., Smedsrud, H. P., Riegler, A. M., Halvorsen, P., Lange, d. T., Johansen, D., and Johansen, D. H., 2020, Kvasir-SEG: A Segmented Polyp Dataset., In Proc. of the international conference on Multimedia Modeling, 451–462pp.

Jha, D., Ali, S., Tomar, K. N., Johansen, D. H., Johansen, D., Rittscher, J., Riegler, A. M., and Halvorsen, P., 2021, Real-Time Polyp Detection, Localization and Segmentation in Colonoscopy Using Deep Learning., IEEE access, vol. 9, doi: 10.1109/ACCESS.2021.3063716

Jia, X., Mai, X., Yuan, Y. Y., Xing, X., Seo, H., Xing, L., and MengQ, H. M., 2020, Automatic Polyp Recognition in Colonoscopy Images Using Deep Learning and Two-Stage Pyramidal Feature Prediction, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, doi: 10.1109/TASE.2020.2964827

Jha, D., Smedsrud, H. P., Riegler, A. M., Johansen, D., 2019, ResUNet++: An Advanced Architecture for Medical Image Segmentation, IEEE, doi: 10.1109/ISM46123.2019.00049

Lin, T-Y., Dollar, R., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017, Feature pyramid networks for object detection, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, Jul., 4pp.

Lin, Y., Goyal, P., Girshick, R., He, K., and Dollar, P., 2017, Focal loss for Dense Object Detection, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2980-2988pp.

Li, W., Zhao, Y., Li, F., and Wang, L., 2022, MIA-Net: Multi-information aggregation network combining transformers and convolutional feature learning for polyp segmentation. Elsevier, Knowledge-Based Systems, doi: https://doi.org/10.1016/j.knosys.2022.108824.

Long, J., Shelhamer, E., and Darrell, T., 2015, Fully convolutional networks for semantic segmentation, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 3431–3440pp.

Loshchilov, I. and Hutter, F., 2019, DECOUPLED WEIGHT DECAY REGULARIZATION, ICLR conference.

Mahmud, T., Paul, B., Fattah, A. S., 2020, PolypSegNet: A modified encoder-decoder architecture for automated polyp segmentation from colonoscopy images, Elsevier, Computers in Biology and Medicine, doi: https://doi.org/10.1016/j.compbiomed.2020.104119.

Mehrotra, A., Morris, M., Gourevitch, A. R., Carrell, S. D., et.al., 2018, Physician characteristics associated with higher adenoma detection rate., Elsevier, Gastrointestinal Endoscopy, vol.87, doi: 10.1016/j.gie.2017.08.023

Meijer, José, A., Visser, A., 2019, A Residual Neural-Network Model to Predict Visual Cortex Measurements, *BNAIC/BENELEARN*

Park, Beom-K., Lee, Y. J., 2022, SwinE-Net: hybrid deep learning approach to novel polyp segmentation using convolutional neural network and Swin Transformer, *Journal of Computational Design and Engineering*, Volume 9, Issue 2, 616–632pp, https://doi.org/10.1093/jcde/qwac018.

Rodriquez, N. A., Carbajales, D. R., Fernandez, L. H. et al, 2021, Deep Neural Networks approaches for detecting and classifying colorectal polyps, Elsevier Neurocomputing, doi: https://doi.org/10.1016/j.neucom.2020.02.123

Ribera, J., Güera, D., Chen, Y., Delp, J. E., 2019, Locating Objects Without Bounding Boxes, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6479-6489pp.

Ronneberger, O., Fischer, P., Brox, T., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention, MICCAI., Lecture Notes in Computer Science, vol 9351. Springer, Cham., doi: https://doi.org/10.1007/978-3-319-24574-4_28.

Salehi, S. S. M., Erdogmus, D., Gholipour, A., 2017, Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. In: Wang Q., Shi Y., Suk HI., Suzuki K. (eds) Machine Learning in Medical Imaging. MLMI 2017. Lecture Notes in Computer Science, vol 10541. Springer, Cham., doi: https://doi.org/10.1007/978-3-319-67389-9_44

Ta, N., Chen, H., Lyu, Y., Wu, T., 2022, BLE-Net:boundary learning and enhancement network for polyp segmentation, *Multimedia Systems*, doi: https://doi.org/10.1007/s00530-022-00900-2.

Wen, H., Shi, J., Chen, W., Liu, Z., 2018, Deep Residual Network Predicts Cortical Representation and Organization of Visual Features for Rapid Categorization. *Sci Rep* 8, doi: https://doi.org/10.1038/s41598-018-22160-9

Yeung, M., Sala, E., Schönlieb, B-C., Rundo, L., 2021, Unified Focal Loss: Generalising Dice and cross entropy-based losses to handle class

imbalanced medical image segmentation, Elsevier, Computerized Medical Imaging and Graphics, doi: https://doi.org/10.1016/j.compmedimag.2021.102026.

Zhao, S., Wang, S., Pan, P. et al, 2019, Magnitude, Risk Factors, and Factors Associated with Adenoma Miss Rate of Tandem Colonoscopy: A Systematic Review and Meta-analysis, Elsevier, Gastroenterology, vol.156, doi: 10.1053/j.gastro.2019.01.260

ACKNOWLEDGEMENT

I sincerely would like to acknowledge and give my warmest thanks to my supervisor Prof. Dr. Mehmet KUNTALP, for his guidance, valuable suggestions, criticism, and support during this study. His contributions and advices motivated me through all the stages of doing my thesis.

I would also thank to my committee members for letting me defense be an enjoyable and motivational.

Finally, I am deeply grateful to my parents for their support, appreciation, encouragement and keen interest in my academic achievements.

RESUME

Mahmut Ozan GÖKKAN

EDUCATION

Ege University, Bornova, Izmir, Turkey
Ph.D. in Biomedical Tecnologies Deparment,
Institute of Natural and Applied Sciences
2018-2023

Ege University, Bornova, Izmir, Turkey
M.Sc. in Electrical & Electronics Engineering Department,
Institute of Natural and Applied Sciences
2012-2014

Başkent University, Etimesgut, Ankara, Turkey
B.S. in Computer Engineering Department,
2005-2011

PUBLICATIONS

O. Gökkan, M. Kuntalp, "A new imbalance-aware loss function to be used in a deep neural network for colorectal polyp segmentation " Computers in Biology and Medicine, Elsevier, doi:https://doi.org/10.1016/j.compbiomed.2022.106205, 2022.

M. O. Gökkan, S. Tozburun, "Automatic classification of melanocytic skin tumors based on hyperparameters optimized by cross-validation using support vector machines" SPIE Photonics West, doi:https://doi.org/10.1117/12.2542161, 2020.

INTERNATIONAL CONFERENCES

Ozan Gökkan, Mehmet Kuntalp, “Identification of Colorectal Polyp Region using Optimized Deep Convolutional Encoder-Decoder Network”, Journal of Artificial Intelligence Theory and Applications (AITA), 2021, Turkey.

Ozan Gökkan, Mehmet Kuntalp, “Performance Comparison of Polyp Segmentation Model using Imbalance-Aware Losses in Deep Neural Networks”, Journal of Artificial Intelligence Theory and Applications (AITA), 2021, Turkey.

APPENDIX A

Python code for ResNet18+UNet CNN model

```
import torch
import torch.nn as nn
from torchvision import models

def conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True)
)

def double_conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True),
nn.Conv2d(out_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels)
)

class Res18Unet(nn.Module):
def __init__(self, net_out_ch=1):
super().__init__()
encoder = models.resnet18(pretrained=True)
self.encoder_layers = list(encoder.children())

self.block1 = nn.Sequential(*self.encoder_layers[:3])
self.block2 = nn.Sequential(*self.encoder_layers[3:5])
self.block3 = self.encoder_layers[5]
self.block4 = self.encoder_layers[6]
self.block5 = self.encoder_layers[7]

self.up_6 = nn.ConvTranspose2d(512, 512, kernel_size=2, stride=2)
self.double_6 = double_conv_bn_relu(768, 512)
self.up_7 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
self.double_7 = double_conv_bn_relu(384, 256)
self.up_8 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
self.double_8 = double_conv_bn_relu(192, 128)
self.up_9 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
self.double_9 = double_conv_bn_relu(128, 64)
self.up_10 = nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2)
self.conv10 = nn.Conv2d(32, 16, kernel_size=3)
self.conv11 = conv_bn_relu(16, net_out_ch)

def forward(self, x):
block1 = self.block1(x)
block2 = self.block2(block1)
block3 = self.block3(block2)
block4 = self.block4(block3)
block5 = self.block5(block4)

x = self.up_6(block5)
x = torch.cat([x, block4], dim=1)
x = self.double_6(x)
```

```
48.  x = self.up_7(x)
49.  x = torch.cat([x, block3], dim=1)
50.  x = self.double_7(x)

51.  x = self.up_8(x)
52.  x = torch.cat([x, block2], dim=1)
53.  x = self.double_8(x)

54.  x = self.up_9(x)
55.  x = torch.cat([x, block1], dim=1)
56.  x = self.double_9(x)

57.  x = self.up_10(x)
58.  x = self.conv10(x)
59.  x = self.conv11(x)

60.  return x

61.  model = Res18Unet().cuda()
```

APPENDIX B

Python code for ResNet34+UNet CNN model

```
import torch
import torch.nn as nn
from torchvision import models

def conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True),
)

def double_conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True),
nn.Conv2d(out_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
)

class Res34Unet(nn.Module):
def __init__(self, net_out_ch=1):
super().__init__()
encoder = models.resnet34(pretrained=True)
self.encoder_layers = list(encoder.children())

self.block1 = nn.Sequential(*self.encoder_layers[:3])
self.block2 = nn.Sequential(*self.encoder_layers[3:5])
self.block3 = self.encoder_layers[5]
self.block4 = self.encoder_layers[6]
self.block5 = self.encoder_layers[7]

self.up_6 = nn.ConvTranspose2d(512, 512, kernel_size=2, stride=2)
self.double_6 = double_conv_bn_relu(768, 512)
self.up_7 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
self.double_7 = double_conv_bn_relu(384, 256)
self.up_8 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
self.double_8 = double_conv_bn_relu(192, 128)
self.up_9 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
self.double_9 = double_conv_bn_relu(128, 32)
self.up_10 = nn.ConvTranspose2d(32, 32, kernel_size=2, stride=2)
self.conv10 = conv_bn_relu(32, net_out_ch)

def forward(self, x):
block1 = self.block1(x)
block2 = self.block2(block1)
block3 = self.block3(block2)
block4 = self.block4(block3)
block5 = self.block5(block4)

x = self.up_6(block5)
x = torch.cat([x, block4], dim=1)
x = self.double_6(x)

x = self.up_7(x)
```

```
48.  x = torch.cat([x, block3], dim=1)
49.  x = self.double_7(x)

50.  x = self.up_8(x)
51.  x = torch.cat([x, block2], dim=1)
52.  x = self.double_8(x)

53.  x = self.up_9(x)
54.  x = torch.cat([x, block1], dim=1)
55.  x = self.double_9(x)

56.  x = self.up_10(x)
57.  x = self.conv10(x)

58.  return x

59.  model = Res34Unet().cuda()
```

APPENDIX C

Python code for *omni-comprehensive* loss function

```
import numpy as np
import torch.nn as nn
from torch.nn import functional as F

# Normalized Cross Correlation
def NCC(inputs, targets):
    c = (inputs - inputs.mean()) / (inputs.std() * len(inputs))
    d = (targets - targets.mean()) / (targets.std())
    c = c.cpu().detach().numpy()
    d = d.cpu().detach().numpy()
    ncc = np.correlate(c, d, 'valid')
    return ncc.mean()

# Calculation of Tversky Index
def TverskyIndex(inputs, targets, adding_term=1, beta=0.1):
    #True Positives, False Positives & False Negatives
    TP = (inputs * targets).sum()
    FP = ((1-targets) * inputs).sum()
    FN = (targets * (1-inputs)).sum()
    Tversky = (TP + adding_term) / (TP + beta*FP + (1-beta)*FN + adding_term)
    return Tversky.mean()

class omni_comprehensive_loss(nn.Module):
    def __init__(self):
        super(omni_comprehensive_loss, self).__init__()
    def forward(self, inputs, targets):
        alpha = 0.5
        #sigmoid activation layer
        inputs = F.sigmoid(inputs)
        #flatten label and prediction tensors
        inputs = inputs.view(-1)
        targets = targets.view(-1)
        #compute binary cross-entropy
        BCE = F.binary_cross_entropy(inputs, targets, reduction='mean')
        omni_comprehensive = (1 - (alpha*NCC(inputs,targets)+\
                        (1-alpha)*TverskyIndex(inputs, targets)))*BCE
        return omni_comprehensive
```

APPENDIX D

Python code for carbon emission measurement

```
def start_training(self):
    for epoch in range(self.num_epochs):
        self.tracker.epoch_start()
        self.phase(epoch, "train")
        state = {
            "epoch": epoch,
            "best_loss": self.best_loss,
            "state_dict": self.net.state_dict(),
            "optimizer": self.optimizer.state_dict(),
        }
        val_loss = self.phase(epoch, "val")
        self.scheduler.step(val_loss)
        if val_loss < self.best_loss:
            self.val_no_improve = 0
            print("******** Saved optimum case ********\n")
            state["best_loss"] = self.best_loss = val_loss
            torch.save(state, "./model.pth")
        else:
            self.val_no_improve +=1
              # Check early stopping condition
            if self.val_no_improve == self.n_epochs_stop:
                print('Early stopping!' )
                early_stop = True
                break
            else:
                continue
            if early_stop:
                print("Stopped")
                break
        self.tracker.epoch_end()
    print("----------------------------------------------------\n")
    self.tracker.stop()
```

APPENDIX E

Python code for data capturing from colonoscopy device using frame grabber

---

```
1. import cv2
2. import streamlit as st
3. from PIL import Image

4. st.title("Application of Polyp Segmentation")
5. FRAME_WINDOW = st.image([])

6. # Blackmagic --ultra studio mini recorder (frame grabber) for video capturing stage using
   opencv
7. cam = cv2.VideoCapture('decklinksrc mode=7 connection=0 ! videoconvert ! appsink')

8. #image upload
9. ret, frame = cam.read()

10. #real-time video streaming, measuring frames per second (FPS) and making polyp
    segmentation
11. cam.set(cv2.CAP_PROP_FPS, 30)
12. fps = int(cam.get(5))
13. print("fps:", fps)

14. if st.button("Apply"):
15. while True:
16. ret, frame = cam.read()
17. frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
18. prediction = model_prediction(frame, model) # model prediction function
19. st.image(Image.open(prediction))
20. FRAME_WINDOW.image(frame)
21. else:
22. st.write('Stopped')
```

APPENDIX F

Python code for standalone and device-independent web application

---

```
import cv2
import streamlit as st
from PIL import Image
import torchvision.transforms as T
import torch
import numpy as np
import torch.nn as nn
from torchvision import models

best_threshold=0.5
min_size=5
image_size=448

def post_process(probability, threshold, min_region_size,size):
mask = cv2.threshold(probability, threshold, 1, cv2.THRESH_BINARY)[1]
num_component, component = cv2.connectedComponents(mask.astype(np.uint8))
predictions = np.zeros((size, size), np.float32)
num = 0
for c in range(1, num_component):
p = component > 0
if p.sum() > min_region_size: # if region of polyp is greater than min_region_size
predictions[p] = 1
num += 1
return predictions, num


def conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True)
)

def double_conv_bn_relu(in_channels, out_channels):
return nn.Sequential(
nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
nn.ReLU(inplace=True),
nn.Conv2d(out_channels, out_channels, kernel_size=3, stride=1, padding=1),
nn.BatchNorm2d(out_channels),
)

class Res18Unet(nn.Module):
def __init__(self, net_out_ch=1):
super().__init__()
encoder = models.resnet18(pretrained=True)
self.encoder_layers = list(encoder.children())

self.block1 = nn.Sequential(*self.encoder_layers[:3])
self.block2 = nn.Sequential(*self.encoder_layers[3:5])
self.block3 = self.encoder_layers[5]
self.block4 = self.encoder_layers[6]
self.block5 = self.encoder_layers[7]

self.up_6 = nn.ConvTranspose2d(512, 512, kernel_size=2, stride=2)
```

```
self.double_6 = double_conv_bn_relu(768, 512)
self.up_7 = nn.ConvTranspose2d(512, 256, kernel_size=2, stride=2)
self.double_7 = double_conv_bn_relu(384, 256)
self.up_8 = nn.ConvTranspose2d(256, 128, kernel_size=2, stride=2)
self.double_8 = double_conv_bn_relu(192, 128)
self.up_9 = nn.ConvTranspose2d(128, 64, kernel_size=2, stride=2)
self.double_9 = double_conv_bn_relu(128, 64)
self.up_10 = nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2)
self.conv10 = nn.Conv2d(32, 16, kernel_size=3)
self.conv11 = conv_bn_relu(16, net_out_ch)

def forward(self, x):
block1 = self.block1(x)
block2 = self.block2(block1)
block3 = self.block3(block2)
block4 = self.block4(block3)
block5 = self.block5(block4)

x = self.up_6(block5)
x = torch.cat([x, block4], dim=1)
x = self.double_6(x)

x = self.up_7(x)
x = torch.cat([x, block3], dim=1)
x = self.double_7(x)

x = self.up_8(x)
x = torch.cat([x, block2], dim=1)
x = self.double_8(x)

x = self.up_9(x)
x = torch.cat([x, block1], dim=1)
x = self.double_9(x)

x = self.up_10(x)
x = self.conv10(x)
x = self.conv11(x)

return x

model = Res18Unet().cuda()
st.title("Application of Polyp Segmentation")

img = st.file_uploader("Upload a polyp image for segmentation",
    type=["png","tif","tiff","jpg", "jpeg"])
img = Image.open(img)
st.image(img)
model.eval()
state = torch.load("C:/Users/ozangokkan/Desktop/PhD/thesis/codes/model.pth",
    map_location=lambda storage, loc: storage)
model.load_state_dict(state["state_dict"])

preprocess = T.Compose([
T.Resize([448,448]),
T.ToTensor(),
T.Normalize(
mean = [0.485, 0.456, 0.406],
std = [0.229, 0.224, 0.225]
)
])
```

```
x = preprocess(img)
x.shape

transformed = torch.unsqueeze(torch.tensor(x), 0)
print(transformed.shape)
out = model(transformed)
out = np.squeeze(out)
out = out.detach().numpy()
predict_image, num_predict_ = post_process(out, best_threshold, min_size, image_size)
if st.button("Apply"):
st.image(predict_image)
st.write('The number of detected polyps :', num_predict_)
```