Efficient Algorithms for Medical Image Segmentation

José Miguel Ferreira Henriques de Oliveira Martinho

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Arlindo Manuel Limede de Oliveira

Examination Committee

Chairperson: Prof. David Manuel Martins de Matos
Supervisor: Prof. Arlindo Manuel Limede de Oliveira
Member of the Committee: Prof. Ana Catarina Fidalgo Barata

October 2022

Acknowledgments

In the first place, I would like to thank Professor Arlindo Oliveira for all the advice and the opportunity to work with him. I would also like to express gratitude to Instituto Superior Técnico for these years and to INESC-ID for providing all the necessary conditions to carrying out this work.

Last but not least, I would like to thank my family, my friends and all the people close to me who gave me the right support whenever I needed.

Abstract

With the growth in cancer cases and the increasing expenditures in the healthcare system, it is necessary to automate processes, aiming for a faster diagnostic and decrease in expenses. Although current technologies enable to capture high-resolution 3D images of organs, manual segmentation of organs and tumours is still a complex process that requires high expertise.

State-of-the-art algorithms are already very accurate. However, they are very compute-intensive tasks, leading to the need for expensive hardware and energy wasting. Coupling state-of-the-art efficient feature extraction algorithms to the *nnUNet* segmentation framework, this work proposes novel efficient architectures for medical image segmentation. For some tasks, similar results were achieved using around 30% less Floating Point Operations (FLOPs) than the baseline *nnUNet*, also decreasing the inference time. Morevover, a better performance then *nnUNet* was achieved using architectures with slightly longer inference time.

Keywords

Deep Learning; Medical Imaging; Segmentation; Efficiency.

Resumo

Com o crescimento do número de casos de cancro e com o aumento dos gastos no sistema de saúde, é necessário automatizar processos, com vista a tornar mais rápido o diagnóstico e a diminuir gastos. Apesar de as tecnologias mais recentes permitirem a captura de imagem médica 3D de alta resolução, a segmentação manual de órgão e tumores a partir dessas imagens continua a ser um processo que requer muito tempo e qualificações.

Os algoritmos do estado da arte da segmentação através de algoritmos de Aprendizagem Profunda são já bastante exatos. No entanto, consomem muitos recursos, levando à necessidade de *hardware* caro e desperdícios energéticos. Juntando algoritmos eficientes do estado da arte de extração de *features* (ou características) à conhecida framework de segmentação de imagem médica *nnUNet*, este trabalho apresenta novos algoritmos mais eficientes para segmentação de imagem medica. Para algumas tarefas, foram alcançados resultados semelhantes aos da rede original, usando 30% menos Floating Point Operations (FLOPs), também diminuindo o tempo de inferência. Foram também alcançados resultados melhores do que a *nnUNet* original, com um tempo de inferência ligeiramente superior.

Palavras Chave

Aprendizagem Profunda; Imagem médica; Segmentação; Eficiência.

Contents

List of Figures

List of Tables

Acronyms

CNN	Convolutional Neural Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KiTS	Kidney Tumour Segmentation
BraTS	Brain Tumour Segmentation
FCN	Fully-Convolutional Network
MICCAI	Medical Image Computing and Computer Assisted Intervention Society
RoI	Region of Interest
GPU	Graphics Processing Unit
VAE	Variational AutoEncoder
DSC	Dice Similarity Coefficient
sDSC	surface Dice Score
CT	Computed Tomography
FLOP	Floating Point Operation

1

Introduction

Contents

1.1 Problem Statement and Motivation

Tumours are masses of tissue formed by cells that grow and divide more than they should. When malignant (also called cancer), tumours may invade and spread into nearby tissues or other parts of the body. According to PORDATA©[1], malignant tumours are a relatively growing cause of death in Portugal, being the second greatest in this country. In fact, data from the World Health Organization (WHO), indicates that 1 in 6 world deaths is due to cancer [1].

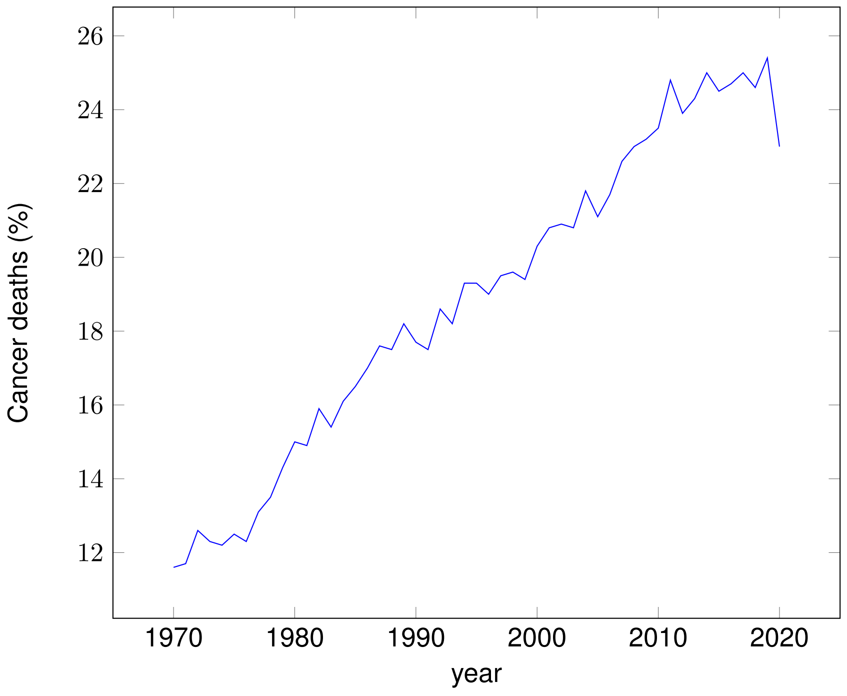


Figure 1.1: Percentage of deaths caused by cancer, in Portugal. Data by PORDATA©

According to some studies and oncologic reports [2], [3], the survival rate of cancer is increasing at a good pace since the 70s and was in 2010 fixed at an average of 70.5% for a 1-year survival period, while 54.3% patients survived for at least 5 years. In Portugal, the mortality rate of cancer was near 27% in 2018.

Despite these advances in medicine, Figure 1.1 shows that the deaths caused by cancer do not stop growing each year, and represent now about 1/4 of deaths in Portugal. Trivially, we can conclude that the number of cancer cases is growing. This growth is mainly explained by the ageing population [4]. This demographic problem is making health expenses grow each year and cancer treatment represents around 6% of the total Portuguese Health Service expenditure [5].

Early diagnosis of tumours plays a significant role in the treatment of malignant tumours and increases the survival rate of patients [6]. To this end, medical images should be acquired through ra-

[1] www.pordata.pt

diological means and analyzed, with the goal of extracting information about the clinical situation of the patient. Cancer screening images are generally acquired through a computational process called Computed Tomography (CT). This process consists of X-Rays and computational operations and outputs highly detailed 3-dimensional images of a specific part of the body. One of the most essential steps in acquiring valid information from these medical images is image segmentation. This process transforms original raw medical images of parts of the human body into meaningful spatially structured information that can be used by doctors and medical assistants [7]. Since the segmentation of tumour areas is a truly specialized and time-consuming task requiring a fair amount of expertise, the automation of this process is considered advantageous.

This field of computer vision is in extensive development and the advances are evident every year. However, frequently, improvement is related to the increase of computational resources needed to evaluate each image. These computing operations, apart from representing excessive energy consumption, are not always available for health institutions.

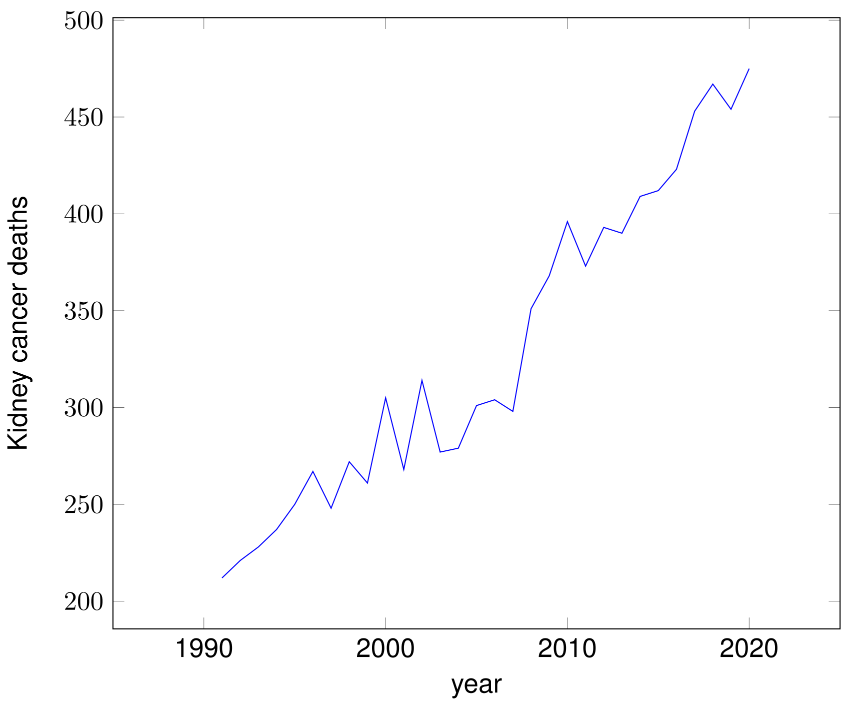


Figure 1.2: Absolute number of deaths caused by kidney cancer in Portugal. Data by INE (Portuguese Statistics Insitute)

It is, therefore, crucial to automate cancer segmentation, aiming at early detection of this disease as well as decreasing the specialized human resources needed, reducing costs. The goal of this dissertation project is to go through an extensive study in order to improve the performance of current algorithms, as well as decrease the computing power needed for them to evaluate each image. The chosen dataset was Kidney Tumour Segmentation (KiTS), an open database of kidney cancer CTs and annotations done by specialists. Kidney tumours are among the top 10 most frequent cancers in men [3] and have a survival rate near the average cancer [2]. The growing trend follows the general malignant tumour growth already covered in this section, as shown in Figure 1.2.

1.2 Approach

Looking at the leaderboard from the main image segmentation challenges, one can observe that most of the state-of-the-art approaches consist of convolutional neural network architectures divided into the encoder and decoder stages. The encoder is responsible for transforming the input into a series of features (also called the ”latent space”) and the decoder uses these features to obtain the final segmentation.

Using state-of-the-art computationally efficient feature extraction algorithms, the goal of this project is to try to improve a baseline encoder-decoder network (*U-Net*), by comparing the performance and resource consumption of different encoder-decoder combinations. For this assessment, the KiTS dataset is being used. This dataset will be used for training and evaluation of the developed algorithms.

1.3 Thesis structure

- Chapter 2 - Overview of state-of-the-art scientific work on the area of image segmentation, in specific on medical imaging.
- Chapter 3 - Detailed explanation of the implementation of the different architectures that were implemented, in specific the encoders and decoders that were used and developed. Resource consumption of each network is also assessed.
- Chapter 4 - Analysis of the training decisions and methodologies that were used, specifically, the loss function, hyperparameters and data preprocessing.
- Chapter 5 - Comprehensive comparison between all the architectures that were implemented. They will be evaluated by their Floating Point Operations (FLOPs) requirements as well as the number of parameters and inference time. Counterintuitively, these 3 metrics were revealed not to be directly related among them.
- Chapter 6 - Discussion about the contributions and proposal of future work to be developed.

2

Related Work

Contents

2.1 Convolutional Neural Networks

Deep learning is a branch of machine learning that tries to imitate the human brain's neurons in order to learn the structure and meaning of data through algorithms. These algorithms, called artificial neural networks, have been the subject of intense research since the mid-20th century [8], [9]. With the growing development of technology and the increase in the quantity and quality of data worldwide, the accuracy of these algorithms and the data to be processed has grown over the years.

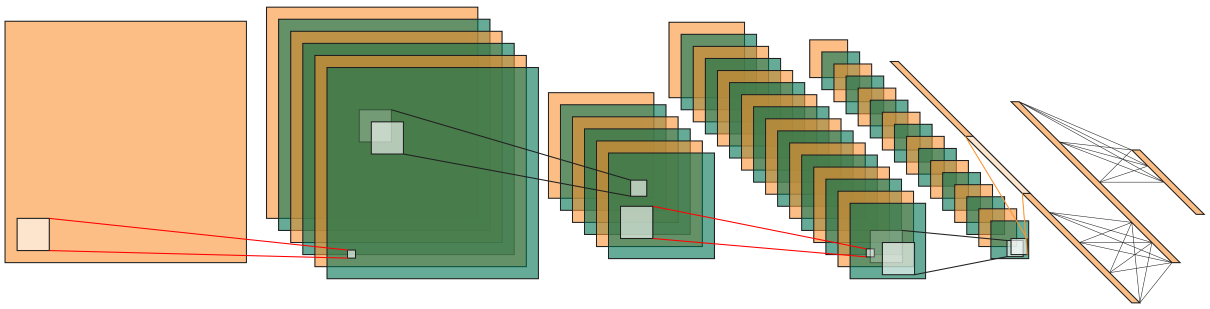


Figure 2.1: Architecture of *LeNet-5*. This network consists of alternated convolutions (represented in red) and subsampling operations (in black), followed by several fully-connected layers, for the inference. Adapted from Lecun *et al.* [10]

Convolutional Neural Networks (CNNs) are a specific type of deep learning algorithms targeted at spatially structured data, such as images. Quickly these algorithms turned into the state-of-the-art architecture family for visual tasks. Trying to simulate the visual perception process of living beings, CNNs learn small details during the first stages and use those details to learn more high-level features and output valid information, such as segmentation, classification or other semantic characteristics of the input. Inspired by Fukushima's Neocognitron [11] that emerged in 1980, the first scientists to introduce the concept of convolutional neural networks were Lecun *et al.* [10] in 1998, with their acclaimed *LeNet*, which is the first notable work about this kind of networks. The name makes reference to the mathematical convolution operation, the basis for the feature extraction operations used by these networks. In general, CNNs are composed of various convolutional layers, alternated with pooling layers and non-linear activation functions. Convolutional layers comprehend various kernels, which will learn different features of their input. Let I be the input of the convolution, O be its output and k be the $N \times N$ convolution kernel. The convolution operation is expressed by:

$$O[m,n] = (I * K)[m,n] = \sum_{l}^{N} \sum_{j}^{N} K[l,j] \times I[m-l, n-j] \tag{2.1}$$

Convolutions may be seen as the weighted sum of neighbour pixels of an image, whose weights are the kernel and are learned through back-propagation. An example convolution is visually represented on Figure 2.2. Subsequent pooling layers help reduce the complexity of the network, maintaining computa-

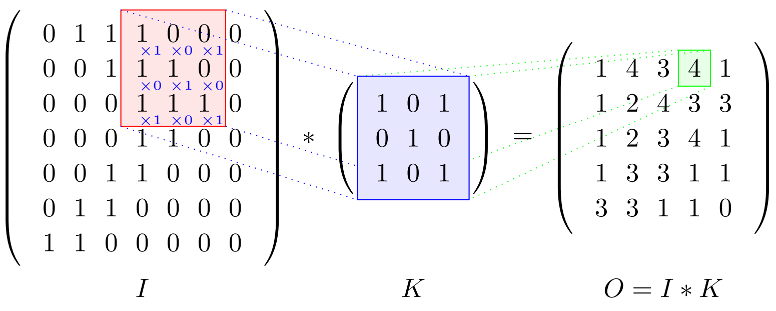


Figure 2.2: Convolution operation

tional efficiency, by decreasing the resolution of the input to the following layers.

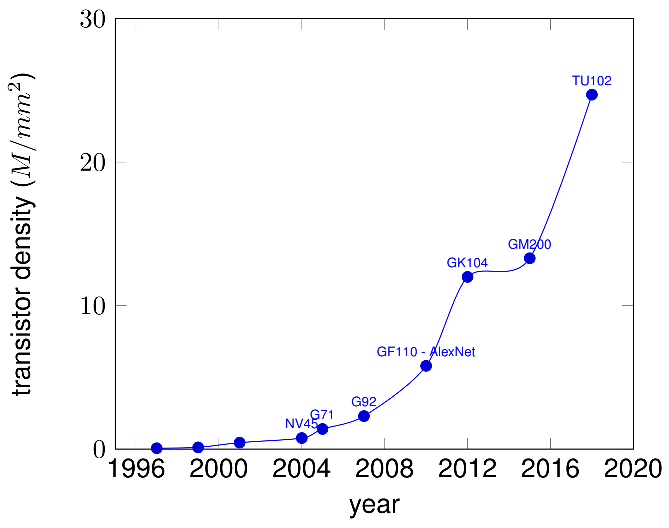


Figure 2.3: *AlexNet* emerged during a period of intense development of the graphics card industry

The continuous research on CNNs only took steps further more than a decade later, with the increase of the computational power of Graphics Processing Units (GPUs). One of the first GPU-based CNNs used in such tasks was presented in 2012, by Krizhevsky *et al.* [12], making them win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Their proposed *AlexNet* consisted of a sequence of 5 convolutional and 3 fully connected layers. The main particularity of their approach is the parallelization technique used. Aiming to be able to run the model in low memory GPUs, they divided the kernels between 2 processing units, which shared outputs among them only on certain layers.

Further research was made during the following years. The increasing of computational power of existent hardware paved the way for the emergence of more and more complex algorithms. This complexity, however, brings some drawbacks: besides the need for powerful hardware and the increase in inference and training time, the increase in complexity does not always mean an increase in performance. Hence, scientific works on CNNs started to make an effort to amortize these hitches.

In 2015, Szegedy *et al.* [13] introduced the Inception modules. These modules consisted of a set of convolutions with different kernel sizes, whose outputs were concatenated with others'. This architecture would enable the network to extract features at different dimensions, being able to scale the network without blowing up in computational complexity. The developed network, *GoogLeNet*, made up of these modules, was able to perform state-of-the-art results on ILSVRC 2014 with a top-5 error rate of 6.67%.

Comparing the performance of shallow with deeper networks, He *et al.* [14] realised that deeper layers in deep networks hardly learn identity mapping. This leads to the possibility of shallow networks performing better than their deeper counterparts in the case of the first layers being optimal for the problem. Aiming to combat this degradation problem, they introduced in 2016 the *ResNet*, a remarkable deep learning framework that learns residual functions. In order to do this, outputs from early layers are directly summed to outputs from deeper layers, through *residual connections*. Mathematically, while classical stacked layers have to learn a mapping $H(x)$, where x is the input to the first layer, in residual networks that mapping is given by $H(x) = R(x)+x$, where $R(x)$ is called the residual and is the learnable function of that network. In extreme cases, the residual would be pushed to zero, which proved to be easier to learn than identity mapping. Using this technique, a 152-layer architecture was developed, which achieved state-of-the-art performance on the *ImageNet* dataset, with a 4.49% top-5 error rate, using considerably less computational resources than previous architecture as the *GoogLenet* [13].

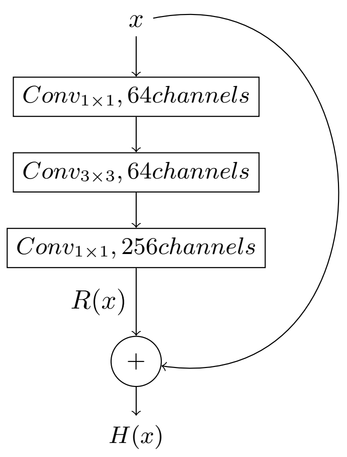

x
$Conv_{1\times1}, 64channels$
$Conv_{3\times3}, 64channels$
$Conv_{1\times1}, 256channels$
$R(x)$
$+$
$H(x)$

Figure 2.4: Building block of the *ResNet* architecture. The input is summed to the output and stacked layers aim to learn the residual function $R(x) = H(x) - x$

Given the success of residual learning, aiming to further exploring this concept, Huang *et al.* [15] proposed in 2017 the *DenseNet*. This architecture makes use of residual connections to connect every layer to every following layer. This approach proved to reduce the vanishing gradient problem, strengthen feature propagation and also reduce the number of parameters. This architecture enabled the researchers to achieve state-of-the-art performance on CIFAR, SVHN and ILSVRC datasets while using a smaller number of parameters than *ResNet* [14].

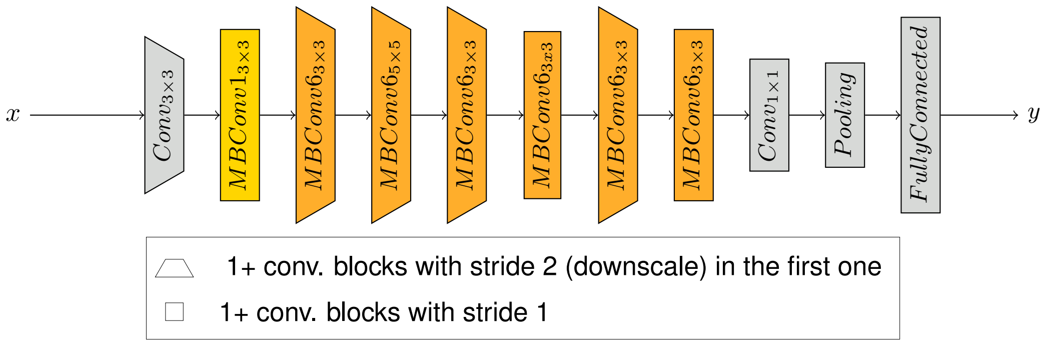


Figure 2.5: *EfficientNet* architecture. This network comprises 9 phases, that consist of a first convolution, followed by 7 *MBConv* blocks and a final mapping phase. The number of repetitions of each *MBConv* block, their number of channels and the input resolution are scaled following the method proposed by the authors [16]

Recently, in 2020, motivated by the need of creating scaling criteria for CNNs, Tan *et al.* presented the *EfficientNet* [16]. The proposed method combines width (number of channels), depth (number of convolutions) and resolution scaling to adapt the size of the networks to the usage requirements. Besides applying this method to existing architectures (such as the *MobileNets* [17], [18] and ResNet [14]), authors have also proposed their own series of CNNs (a baseline and its scaled variations), which achieved state-of-the-art results on the ImageNet dataset, using at least 4 times fewer FLOPs and 2 times fewer parameters than previous state-of-the-art architectures, such as the *ResNet* [14], *DenseNet* [13] AmoeabNet [19]. Heavily influenced by the *MobileNets* [17], [18], this architecture comprises an initial 3×3 convolution used to expand the number of channels, followed by 7 scalable phases composed of *MBConv* blocks, and the final output phase, as we can observe in Figure 2.5. The resolution, number of channels and number of convolutions used in each *MBconv* phase are determined by the scale factor, which varies in each *EfficientNet* variation.

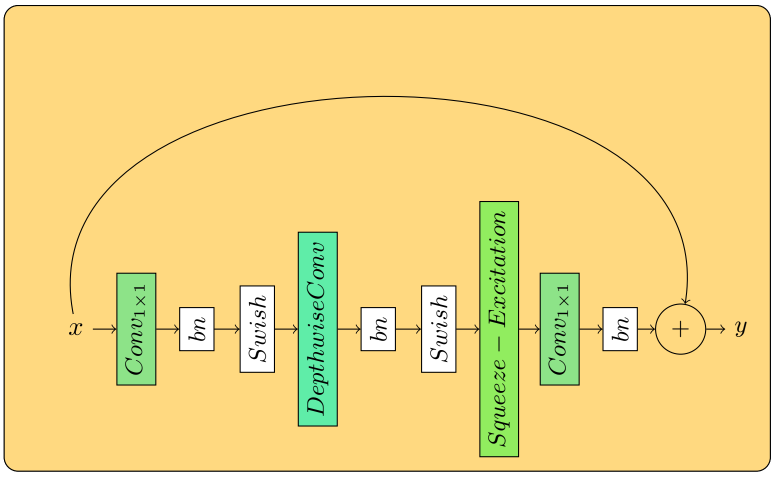


Figure 2.6: *MBConv* block

The main building blocks of the *EfficientNet* are *MBConvs*. These are convolutional blocks originally presented by Sandler *et al.* [18], characterized by their inverted residual shape. Firstly, an optional 1×1 convolution is applied to increase the number of channels, creating a wide intermediary shape. Due to this feature map's great number of channels, a "normal" convolution would be too computationally expensive, therefore a depthwise convolution is applied instead. At the end of the block, a *Squeeze-Excitation* module [20] is applied to the output of the depthwise convolution. This latter module comprehends a first convolution that decreases the number of channels to a very small number, followed by a second convolution that increases the number of channels to the same as the input of the block. A final convolution is applied and the output is then added to the input of the block, as with the *Resnet* [14]. Every convolutional operation is followed by a batch normalization layer and a swish [21] activation function.

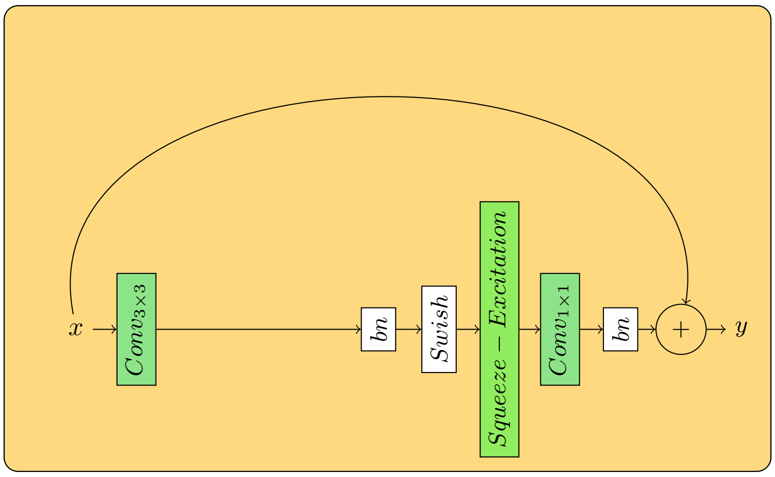
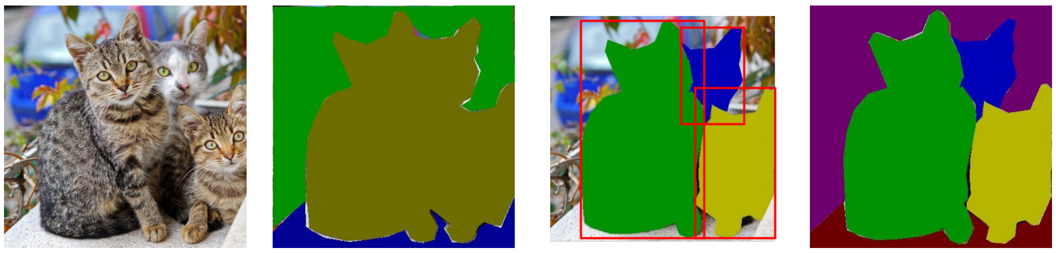


Figure 2.7: Fused *MBConv* block

Later, in 2021, the same authors [22] proposed an improved model of this network, *EfficientNetv2*. The main architectural difference relative to the former is the use of an alternative building block: the *Fused-MBConv*. This convolutional block switches the first expanding convolution and the depthwise convolution by an expanding 3x3 convolution. *EfficientNetV2* was able to reduce the computational complexity in comparison to its original counterpart, maintaining comparable accuracy.

2.2 CNN Architectures for Image Segmentation

Image segmentation is the process of assigning a class to each pixel of an image, creating different regions of pixels that ideally correspond to different objects or different classes of objects. There are different types of image segmentation: semantic segmentation (where objects of the same class are assigned the same label), instance segmentation (where different objects of the same class are assigned different labels), and panoptic segmentation (a combination of the previous two).



(a) input image (b) semantic segmentation (c) instance segmentation (d) panoptic segmentation

Figure 2.8: Different segmentation tasks. Semantic segmentation (b) aims to label each pixel with the class of object that it belongs to. Every pixel that belongs to a cat will be given the same label. Instance segmentation (c) labels separately instances of a specific class, cats are given distinct labels. Panoptic segmentation (d) aims to perform instance segmentation on the target class and to perform semantic segmentation on the rest of the ”stuff”.

Image segmentation has umpteen use cases, such as autonomous vehicles, medical image analysis and digital marketing [23], making it a very important problem in the computer vision field.

The first approaches to image segmentation originated in the 1980s, being divided in three categories: thresholding, edge-based and region-based [24]. Thresholding techniques chose the class of each pixel based on whether its intensity was greater or less than that threshold. Edge-based segmentation consisted of applying a filter on the image (such as Prewitt's filter) to detect edges that would be used to segment the image. Finally, region-based segmentation groups pixels that have similar characteristics. Later, some more complex techniques emerged, such as the usage of Markov Random Fields [25], Bayesian Classifiers [26], clustering [27]–[29] or image registration [30].

Following the success of CNNs on image feature extraction, and motivated by the relevancy of image segmentation in some fields of science such as medicine, this technique rapidly become standard for semantic tasks of this kind.

The first authors to present relevant work in this field were Ciresan *et al.* [31]. The proposed approach consisted of a ”classical” CNN, composed of convolutional and max pooling Layers, followed by several fully connected layers. The inference and training were done to the neighbouring region (patch) of a pixel and the output represented the class that each pixel belongs to, from among 2 classes. Although disruptive, this network has shown some drawbacks. Firstly, as the network must be run in a different patch for each pixel, it can make the inference quite slow and complex. Moreover, larger patches mean more max-pooling layers, which reduce localization accuracy, whilst small patches mean a loss of context of the image [32].

Aiming to increase the segmentation accuracy of the method proposed by Ciresan *et al.* [31], the first Fully-Convolutional Networks (FCNs) have surfaced [33]. Given the success of convolutions, authors modified some existing classification network architectures, such as *AlexNet* [12], *VGG16* [34] and

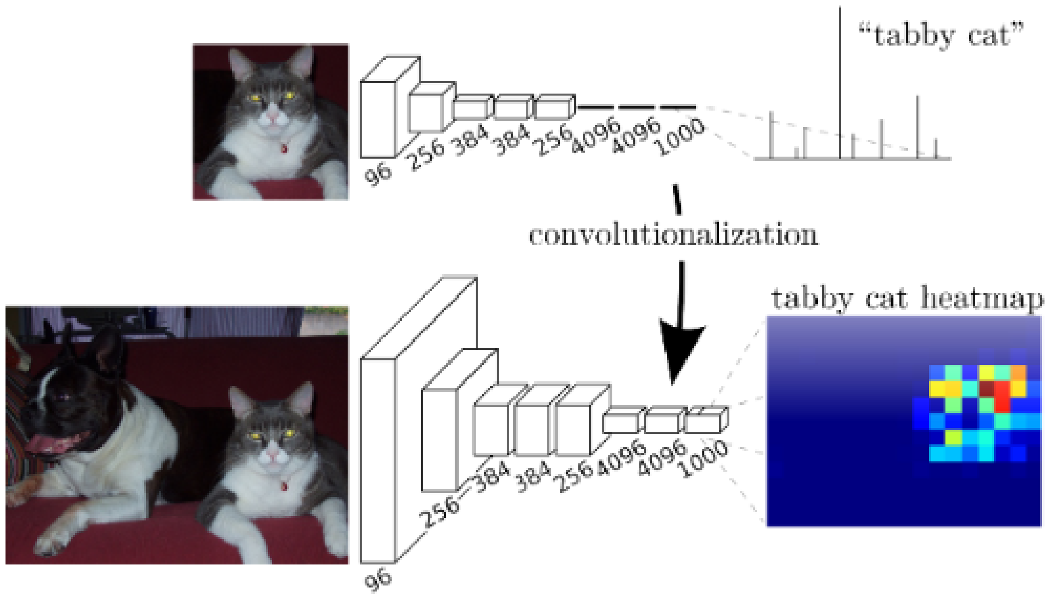


Figure 2.9: Conversion of state-of-the-art classification neural networks into fully convolutional ones (”convolutionalization”). The fully connected layers are discarded to give place to convolutional ones, enabling a space-aware output map and decreasing inference times.

GoogLeNet [13]. Originally, these networks comprise convolutional layers followed by fully connected ones for classification. This latter type of layer makes the network be limited to a specific input resolution and to lose the spatial information. Through a process they called ”convolutionalization” (Figure 2.9), the authors propose to remove the fully connected networks, replacing them with convolutional ones. This makes the networks resolution-agnostic and enables them to output a spatial output map, making them fit for segmentation problems. Moreover, the computation is highly amortized over the overlapping regions of input patches. As an example, *AlexNet* [12] took 1.2ms to infer classification in an image, its fully convolutional version was proven to take 22ms to produce a 10×10 grid of outputs, which is 5 times faster, taking in account the number of outputs. Although this method works, the results were not satisfactory, which motivated the same authors to propose a novel network architecture, that introduced skip connections to combine coarse with finer information.

Based on the work on FCNs [33] and other scientific papers that suggested to combine finer with coarser feature maps to produce the output [35], [36], Ronneberger *et al.* [32] came up with one of the most relevant works on image segmentation, the *U-Net* (Figure 2.10). Originally conceived to be applied to biomedical imaging, it is now the basis for the vast majority of state-of-the-art works on semantic segmentation. This architecture consists of two branches: a contractive and an expansive path. The former (also known as the encoder) follows a typical convolutional network architecture. It is a sequence of blocks of two convolutions followed by max pooling for down-sampling. At each down-sampling step, the number of feature maps is doubled. The expansive path (also known as the decoder)

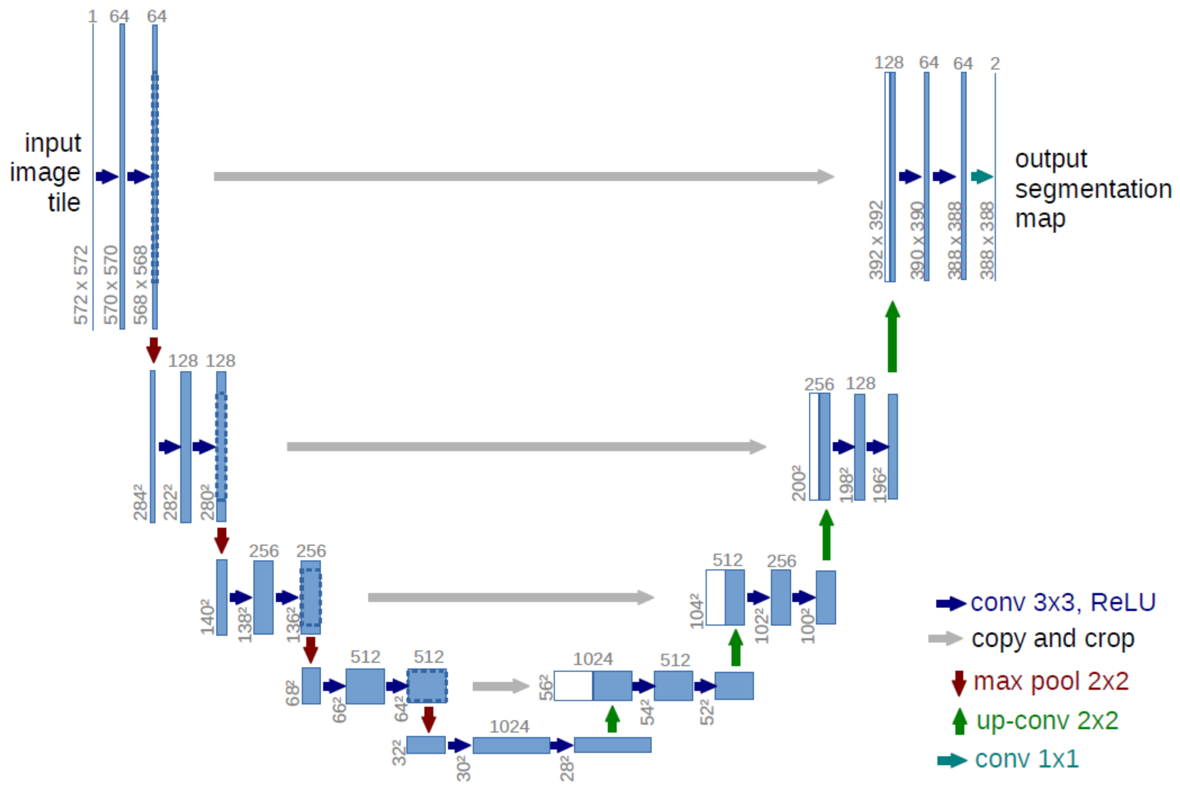


Figure 2.10: The *U-Net* architecture consists of a contractive and an expansive path, where feature maps' dimensions are halved and doubled respectively, while the number of feature maps increases in the former and decreases in the latter. Also, the contractive path's feature maps are concatenated with their expansive path counterpart to recover spatial information.

consists of the inverse operations: transposed convolutions to perform the up-sampling, followed by convolutions used to halve the feature map size. At the end of the decoder, a 1×1 convolution is applied in order to map each pixel to the desired class. At each level of the encoder, there is a ”skip connection” to the corresponding decoder resolution step. These connections consist of the concatenation of the feature maps in the contraction path to the corresponding ones in the up-sampling path. This process enables the decoder to recover the spatial information that may have been lost during the contraction step. To overcome the problem of the scarcity of data in biomedical problems, the authors apply data augmentation, specifically random elastic deformations following a Gaussian distribution. Moreover, a *dropout* layer exists at the end of the contracting path, which works as implicit data augmentation. *U-Net* outperformed state-of-the-art architectures in several biomedical imaging datasets, such as PhC-U373 and DIC-HeLa. Prompted by the good results that *U-Net* has shown, many scientific works have used it as a base to improve and explore.

Although the *U-Net* has been implemented for 2D image segmentation, nowadays many biomedical images are tridimensional. Hence, some 3D variants have been proposed [37] [38]. The main idea

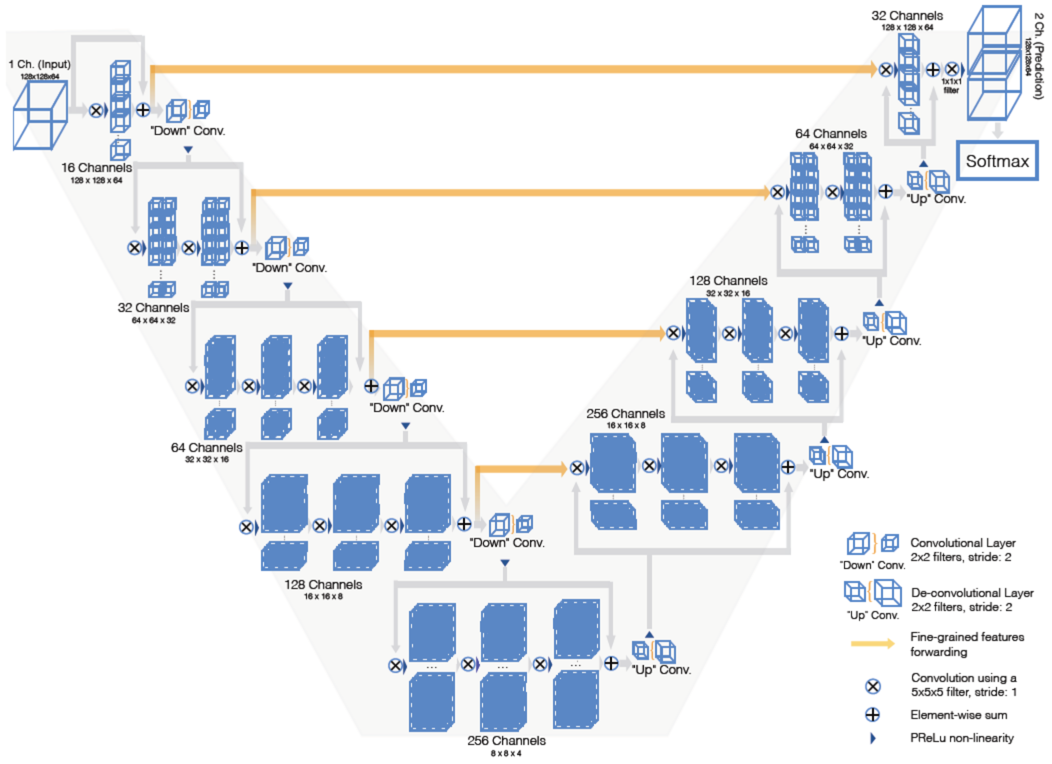


Figure 2.11: *V-Net* architecture [37]. 2-dimensional convolutions are replaced by 3-dimensional ones and residual connections [14] are introduced

behind these approaches is to substitute normal convolutions for 3D ones, instead of processing slices of the image. The *V-Net* model [37] goes even further, by implementing residual connections [14] in the encoder path of the architecture. Aiming to overcome the difficulty of labelling 3D human images slice by slice, the approach by Çiçek *et al.* [38] is able to learn 3D segmentation with annotated 2D slices.

In 2018, Oktay, Schlemper, Folgoc, *et al.* introduced attention gates (AG) to the *U-Net*'s skip connections [39]. The aim of this technique is to suppress feature responses in irrelevant background regions, in order to reduce false positives. These gates consist of weighted sums of each encoder layer with its decoder counterpart's immediately lower layer, the result of which is piece-wise multiplied by the original input. This model was revealed to be much more accurate than the original *U-Net*, although being computationally more expensive.

A different approach is followed by *UNet++* [40]. The authors propose introducing an expansive path for each contractive level, creating as many full-resolution feature maps as the number of down-sampling operations. This enables deep supervision to be used on every full-resolution feature map. This way, at inference time, this network can be pruned in order to use only the smaller encoder-decoder paths (fast inference) or an average of all of them (more accurate). It is important noticing that every feature at a certain resolution level is concatenated with the other feature maps at the same level. This architecture

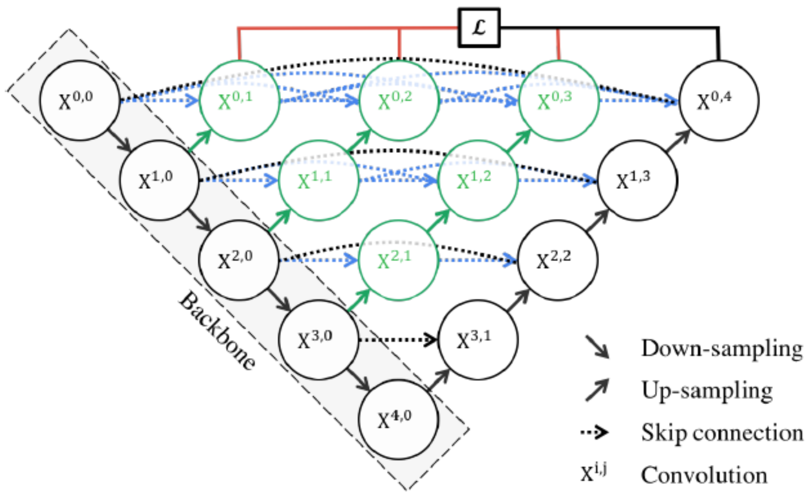


Figure 2.12: *U-Net++* architecture [40]. Every feature map is connected to the others, through skip connections.

has shown big improvements in comparison to *U-Net*, even when deep supervision was not used.

Aiming to overcome data scarcity, in addition to the aforementioned, the error function is also adjusted to handle class imbalance (background is often more frequent than foreground labels). Çiçek *et al.* [38] propose a weighted cross entropy error, whereas Milletari *et al.* [37] suggests to maximize the Dice Coefficient Function. Other approaches apply a combination of both binary cross-entropy loss and dice coefficient [7], [40].

The encoder-decoder model used by the *U-net* has been the target of extensive studies and improvements. While the *V-net* [37] uses residual connections in the encoder convolutions, improving learning speed and achieving better results than state-of-the-art architectures, Jegou *et al.* [41] propose the substitution of the encoder convolutions for *DenseNets* [15]. This showed promising results in different datasets. On the other hand, Lourenço-Silva *et al.* proposed to apply a bottleneck architecture similar to *EfficientNet* [16] at both the encoder and the decoder of the *Unet++*, creating the *EfficientUnet++* [42].

2.3 Medical Image Segmentation

Compared to natural images, medical images require a much greater level of accuracy. Otherwise, automatic segmentation it can lead to poor user experience [40]. Medical datasets feature many characteristics that differentiate them from other datasets, such as the low number of classes [43], data scarcity [44] or class imbalance [37]. Other factors make these tasks challenging, such as the variation in the appearance of certain organs and medical images and the pollution of medical images with artefacts and distortions [37].

The *U-Net* has undoubtedly played a crucial role in medical imaging segmentation. We can clearly note this by looking at the leaderboard of the 2019 edition of KiTS, one of the biggest segmentation challenges hosted by the Medical Image Computing and Computer Assisted Intervention Society (MICCAI): all the 15-top methods are *U-Net* like architectures [7]. Throughout this section, I will present some state-of-the-art methods for Medical Imaging Segmentation, presented at some worldwide challenges. I will give emphasis to a specific model, the *nnU-Net*, which has obtained promising results on many different datasets and has been used by a large variety of authors.

2.3.1 *nnU-Net*

In 2018, the second place on the Brain Tumour Segmentation (BraTS) (Brain Tumor Segmentation) Challenge's leaderboard went to a CNN model proposed by Isensee, Kickingereder, Wick, *et al.*, called No New-Net [45]. This model assumes that a well trained *U-Net* or 3D *U-Net* [32], [38] is more dataset agnostic and may show better results than other *U-Net* variations (such as residual connections [14], [37], dense connections [15], [41] or attention gates [39]). Later on, the same authors proposed the *nnU-net* [7], a self configuring deep learning method for medical image segmentation. The self-configuration enables less experienced users to train the network without great knowledge and ensures the best approach for each dataset. The key idea behind this approach is to capture the *dataset fingerprint*, which describes the dataset used for training and to elaborate the *pipeline fingerprint* based on it.

The *pipeline fingerprint* consists of 3 types of parameters: *Blueprint Parameters*, *Inferred Parameters* and *Empirical Parameters*. *Blueprint Parameters* are key network choices, that won't change among different datasets. It features key details about the network architecture and training parameters. *Inferred Parameters* are variable depending on the dataset. On data preprocessing, they specify the target spacing desired for the training samples and normalization and resampling techniques to be applied to the training set. They also adjust batch and patch size to fit hardware limitations, as well as the number of downsampling/upsampling steps to be performed by the encoder/decoder. A 2D *U-Net*, a 3D *U-Net* and a cascade 3D *U-Net* are then trained and the best ensemble is chosen by cross-validation. This ensemble constitutes the *Empirical Parameters*.

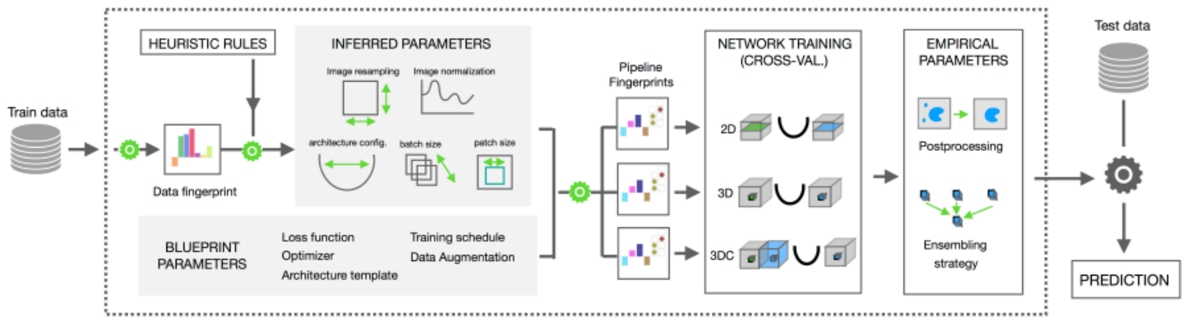


Figure 2.13: The *nnU-Net*'s parameters consists of Inferred Parameters (inferred through the data fingerprint), Blueprint Parameters (key choices) and Empirical Parameters (chosen by cross-validation)

For training, the authors propose deep supervision at every but the two lowest resolutions and as a loss function, the sum of cross-entropy and Dice loss is proposed.

nnU-net proved to be very impactful on medical imaging segmentation research. Many applications and variations of this approach have been proposed both by the same authors as well as by other researchers.

2.3.2 State-of-the-Art Tumor Segmentation Techniques

KiTS and BraTS are MICCAI's biggest segmentation challenges and hence can be regarded as the state of the art in medical image segmentation. Editions from the last years of these two competitions will be in focus throughout this section.

A large portion of KiTS 2019 and 2021's approaches is based on the successful *nnU-Net* and its variants. Hou *et al.* [46] propose to use the *nnU-Net* on a 3 stage approach: after the pre-processing, first and second stage use a *nnU-Net* in order to localize and segment the kidney. The tumour segmentation is then performed using a custom-made 3D *U-Net*. Other similar 2-stage approaches have also been followed [47], [48], where they take advantage of 2 *nnU-Nets* in order to first localize the kidneys and then segment the tumour. The approach from Zhao *et al.* [49] goes even further, using 4 distinct *nnU-Nets*: one for firstly segment the Region of Interest (RoI) of the kidney and then the other 3 to finely segment the kidney, the tumour and the mass, respectively. The latter 2 receive as input both the RoI and the finely segmented kidney. The authors also propose a novel loss function, the Surface Dice Loss, based on the Surface Dice Coefficient. This loss penalizes the model based on the distance between the wrongly classified pixels and the boundary of the ground-truth region. Golts, Khapun, Shats, *et al.* [50] also use the *nnU-Net* architecture, and propose a loss function that penalizes the output of a pixel based on its neighbouring pixels. On the other hand, Yang *et al.* [51] propose to train a model on a large medical imaging dataset and then use the best weights to initialize the training on the KiTS dataset. All the datasets are pre-processed with the methods suggested by the *nnU-Net* and the model used for training is a *U-Net* with residual connections.

Despite the predominance of the use of *nnU-Net* by the top places in this challenge, other promising approaches are worth noting. Myronenko *et al.* [52] presented an encoder-decoder architecture consisting of a larger encoder and a smaller decoder, with a series of convolutions and residual connections. The output of the decoder is concatenated with the output of a parallel boundary stream, which consists of a series of convolutions and attention gates, similar to the approach proposed by Oktay *et al.* [39].

In 2019, the same author proposed another asymmetrical encoder-decoder architecture with two parallel decoders. Here, besides UNet's classic decoder that outputs the segmentation mask, an auxiliary Variational AutoEncoder (VAE) decoder is implemented in the other branch and used during training only, trying to reconstruct the original, ”regularizing” the learning process of the encoder. The novel loss function presented is a weighted sum of VAE loss and segmentation loss and is given by:

$$L = L_{dice} + 0.1 \times L_{L2} + 0.1 L_{KL} \tag{2.2}$$

where L_{dice} is the dice loss of the output segmentation, L_{L2} is the $L2$ loss of the VAE output and L_{KL} is the KL divergence between the estimated distribution $\mathcal{N}(\mu, \sigma^2)$ and a prior distribution $\mathcal{N}(0, 1)$.

This approach has won the first place in BraTS 2018, outperforming the original *nnU-Net* [45].

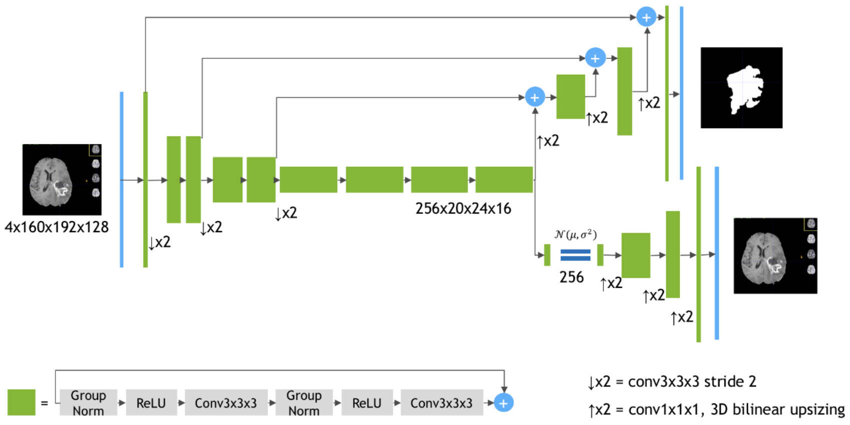


Figure 2.14: Architecture proposed by Myronenko [53]. Two different decoders are attached at the end of the encoder: One for performing the segmentation and a VAE branch for "regularizing" encoder training.

A similar architecture is implemented by Jiang *et al.* [54]. They propose a two-stage segmentation approach, where the first stage is performed by a *U-Net* with a larger encoder and the second stage is performed by a similar network, but with a double decoder, one of which is only used during training. This way, the loss function and the gradient can be applied based on 2 decoders instead of just one. This was awarded first place in BraTS 2019 challenge.

Another asymmetric *U-Net* is presented by Yuan [55]. In this architecture, classical convolution blocks are replaced by residual connections with squeeze-and-excitation [20] modules. Another singularity about this network is that skip connections are done from every encoder level onto every decoder level, by a Scale Attention Block.

Similarly to the approaches by Myronenko and Jiang *et al.*, also the approach presented by Wang *et al.* [56] makes use of 2 parallel branches on a *U-Net* like architecture. However, the latter makes use of both for inference and not just one as the others do. The several input channels are divided among the two branches and a series of connections are made between both of them. This approach was awarded second place in BraTS 2020 challenge, only beaten by a variation of the *nnU-Net* [57] presented by the original authors. This variation focuses especially on data augmentation and an increase in training batch size.

Aside from U-Net architectures, other approaches have been proved relevant for Brain Tumor Segmentation among them the H^2NF-*Net* [58]. This 2 stage approach makes use of 2 networks composed of several fully connected residual networks to process the input images on several resolutions. The outputs of the final stage are then merged using a special module. This has also been awarded second place in BraTS 2020 competition. Also McKinley *et al.* [59] proposed a non-*U-Net* architecture, making use of residual connections and Attention Gates. One of the interesting characteristics of this approach is the outputs of the network: besides the pixels' classes prediction, it also outputs the prediction that a certain classification disagrees with the Ground Truth. This enables the formulation of a novel loss function, that combines the output classification, the output disagreement probability and the ground truth.

2.4 Discussion

Encoder-decoder architectures with skip connections (such as the *U-Net*) are dominant in the image segmentation field. The use of extra skip connections [40], [55] or auxiliary decoder branches [52]–[54], [56] have been widely studied and the results have been satisfactory. Different encoders have also been proposed based on state-of-the-art convolutional frameworks, such as *ResNet* [37], [51] or *DenseNet* [15], [41]. However, *EfficientNet* has not been yet very explored in the encoder-decoder segmentation procedures, despite the good performance it shows on the original paper as a feature extraction network. To my knowledge, the only approach that used *EfficientNet* for medical image segmentation was *EfficientUNet++* [42], which has shown promising results. Other approaches make use of Attention Mechanisms [39], [55], [59], which also performed well.

nnU-Net is undoubtedly the most influential work on state-of-the-art biomedical image segmentation works. A vast majority of recent approaches make use of it either for pre-processing or as a baseline network for their architectures.

3 Architecture Implementation

Contents

3.1 Overview

As the previous chapter has shown, the *nnU-Net* [7] has been established as a reference framework for the implementation of *U-Net* based 3D image segmentation architectures. Its powerful dataset analysis enables it to apply a solid pre-processing pipeline and hyperparameter estimation. Moreover, its modular implementation makes it a perfect fit for building upon it and for the development of novel *U-Net*-like architectures.

Also, some works show that *U-Net* and similar architectures can be successfully modified to work with different feature extraction operations in the encoder path [15], [37], [41], [51]. Given that the *EfficientNet* family is currently one of the state-of-the-art image classification networks, and being it yet under-explored as an encoder in encoder-decoder architectures, during this work several modified *EfficientNet* and *EfficientNetV2* variations were extensively tested as an encoder in the *nnU-Net* framework. Moreover, given that this architectural family use an inferior number of computational resources in comparison to other state-of-the-art algorithms, it makes sense to apply it to high-resolution 3D images. Furthermore, a novel decoder was developed based on the *MBConv* blocks [16]–[18], [22].

This work focuses exclusively on the 3D model of the *nnU-Net* framework.

3.2 *EfficientNet* as an encoder

The original *U-Net* [32] proposes for the encoder a series of classical convolutions on each resolution stage, followed by a max-pooling layer for the downsizing for the next stage. At each stage, the resolution is half of the previous one and the number of channels is doubled. However, the *nnU-Net* framework suggests performing the inter-stage downsampling with strided convolutions instead. Specifically, each stage in both the encoder and the decoder comprises 2 convolutional blocks, each block followed by an instance normalization layer and leaky ReLU activation function. The *nnU-Net*'s encoder architecture for its 3D *U-Net*to be applied on the KiTS dataset is depicted in Figure A.1.

One of the challenges to integrating classification architectures into the *U-Net* is to divide them into resolution stages so that they may be inserted into the encoder. To this end, some changes were made to the original proposed *EfficientNet* model. Firstly, the initial convolution proposed in the original *EfificentNet* publication suggests a fixed-size input strided convolution (whose input size changes among *EfificientNet* variations). However, as the first stage of the U-Net requires a feature map with the exact resolution as the input but wider (with more channels), the stride of this first convolution was changed to 1. Also, the input to the first convolution was made variation agnostic, and always fixed to the patch size. To compensate for the stride change, the first *MBConv* block is implemented with stride 2. The last stage from the *EfficientNet* is pruned, as we don't need to output a mapping, only the latent map created by the convolutions.

Stage	Operator	#Channels	#Layers	Stride
1	$Conv_{3\times3}$	32	1	~~2~~1
2	$MBConv1_{3x3}$	16	1	~~1~~2
3	$MBConv6_{3x3}$	24	2	2
4	$MBConv6_{5x5}$	40	2	2
5	$MBConv6_{3x3}$	80	3	2
6	$MBConv6_{5x5}$	112	3	1
7	$MBConv6_{5x5}$	192	4	2
8	$MBConv6_{3x3}$	320	1	1
~~9~~	~~$Conv_{1x1}\&Pooling\&FC$~~	~~1280~~	~~1~~	~~-~~

Table 3.1: Changes made to the *EfficientNet-b0* classification architecture to create the *EfficientNet-b0* encoder. The strides from the first and second stages were changed, and the last output stage was pruned.

With these changes made, it is now possible to divide this architecture into several resolution stages to match the *U-Net*'s architecture. The first stage corresponds to the first convolution, which will expand the input into more channels. After that, each encoder stage corresponds to the *EfficientNet* block or sequence of blocks until the last convolution before the resolution is halved. Each encoder stage does not necessarily correspond to one *EfficientNet* stage. The *EfficientNet*-b0 encoder is explicitly depicted in Figure 3.1 and replicated in Figure A.2, in a more visually intuitive version.

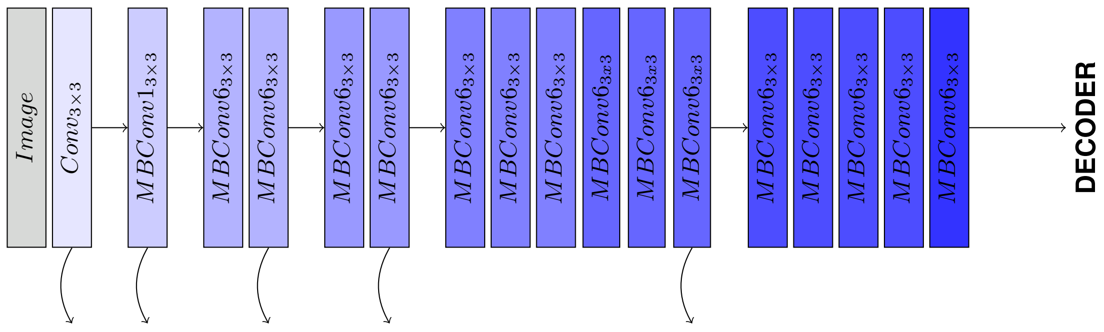


Figure 3.1: *EfficientNet-b0* encoder. Skip connections are marked with a bent arrow. Different color tonality means different *EfficientNet* stage. Note that each encoder stage may contain convolutions from different *EfficientNet* stages.

Despite the change of the encoder, some of the *nnU-Net*'s Inferred Parameters are followed. For the KiTS dataset, *nnU-Net* determines a patch size of $128 \times 128 \times 128$ and therefore it proposes to perform 5 downsamples in the encoder, as the rule of the *nnU-Net* is to perform downsamples until the shape is $4 \times 4 \times 4$. We can observe in Figure 3.1 that this verifies for the proposed encoder. As the *EfficientNet* family is invariant on the number of downsampling operations, this condition verifies for every encoder belonging to the family.

A similar approach was followed for the *EfficientNetV2*, whose architecture is depicted in Appendix A, for legibility due to its high dimension. This newer variant is very similar to the original *EfficientNet*, but makes use of *Fused-MBConv* blocks, already covered in Chapter 2 and represented in Figure A.3 as *FMBConv*.

3.3 Implementation of a novel decoder

Trying to further explore the powerful capacities of *MBConvs*, proven by the *EfficientNets* [16], [22] and *MobileNets* [17], [18], a novel decoder was developed, replacing the generic convolutions proposed by the *nnU-Net* with *MBConvs*. These blocks are repeated twice each stage, such as proposed by the original *nnU-Net*, and have kernel sizes also defined by the *nnU-Net* inferred parameters.

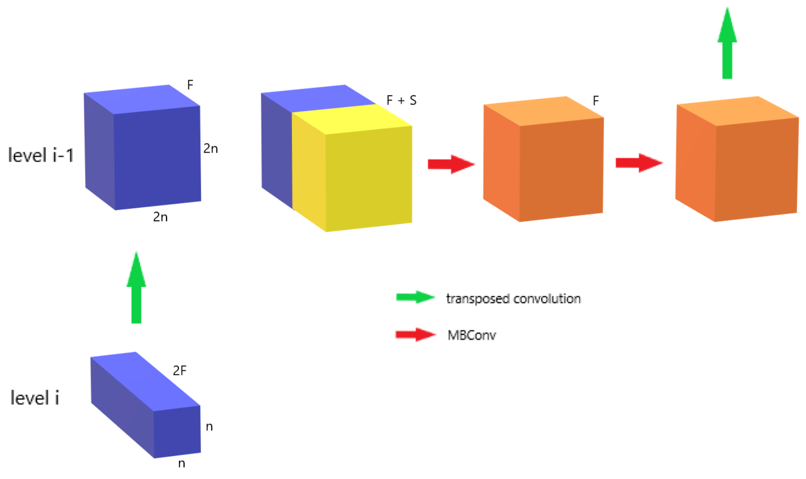


Figure 3.2: *FullyEfficient* decoder. The yellow block corresponds the concatenation from the skip connection. The transposed convolution remains unchanged from the *nnU-Net* implementation. *MBConvs* are the main building block.

As we can observe in Figure 3.2, the block from the deeper level of the encoder is upsampled using the transposed convolution proposed in the original *nnU-Net* implementation. It is important to note that the concatenated volume may not contain the same number of channels as the upsampled one: whereas the original encoder follows the rule of duplicating the number of channels for each level, the *EfficientNet* does not follow a similar pattern. Therefore, besides the fact that the encoder and decoder are asymmetric in the number of channels, the block originated from the concatenation of the two varies in shape among different variations of the *EfficientNet* encoder. This new decoder was named *FullyEfficient-UNet*.

3.4 VAE normalization

A strategy similar to the one followed by Myronenko [53] has also been implemented. A VAE decoder was attached at the end of the encoder and is used during training to regularize the training of the encoder. The loss function is described in Chapter 2.

However, due to the time constraints of this dissertation, aggravated by the excessive amount of time taken by network training, hyperparameters were not finetuned and the results are not included in this work.

3.5 Computational Comparison

	TFLOPs	
Encoder	UNet	FullyEfficient-UNet
$Original$	0.958	0.861
$EficientNet - b0$	0.608	0.507
$EficientNet - b1$	0.610	0.509
$EficientNet - b2$	0.611	0.510
$EficientNet - b3$	0.655	0.553
$EficientNet - b4$	0.690	0.588
$EficientNet - b5$	0.701	0.599
$EficientNet - b6$	0.749	0.647
$EficientNet - b7$	0.797	0.695
$EficientNetV2 - S$	0.678	0.576
$EficientNetV2 - M$	0.724	0.622
$EficientNetV2 - L$	0.924	0.822

Table 3.2: Number of TeraFLOPs required by a forward pass for batch of 2 $128 \times 128 \times 128$ volume of the *U-Net* with different encoders

Aiming to measure the computational impact of these encoders in the network, the open-source tool *fvcore*[1] was used. This tool, besides many other features, includes a powerful flop counter for machine learning algorithms. As expected, the architectures using *EfficientNet* use fewer computational resources than the baseline. Moreover, the novel *FullyEfficient-UNet* decoder also decreases the number of FLOPs. This is due to the efficiency of *MBConvs* in comparison to sequences of classical convolutions.

[1] https://github.com/facebookresearch/fvcore

4 Model Training Methodology

Contents

4.1 Overview

The *nnU-Net* original paper [7] proposes to use batch learning with 5-fold cross-validation for training. However, due to the excessive training time, it was unfeasible to follow the same strategy. Also, as the ground-truth of the test set of the KiTS dataset is not publicly accessible, the training set was split into train, validation and test sets.

Every model was trained using an NVIDIA®Tesla®V100S with 32GB VRAM.

The loss function, hyperparameters and preprocessing are the same as the proposed by the *nnU-Net Blueprint* and *Inferred parameters* and will be addressed in the chapter.

4.2 Loss Function

As suggested in the *nnU-Net* original paper, the loss function that was used was the sum of Generalized Dice Loss and Cross Entropy Loss:

$$L = L_{Dice} + L_{CE} \tag{4.1}$$

4.2.1 Soft Dice Loss

In 2016, Milletari *et al.* [37] proposed to use the Dice Similarity Coefficient (DSC) as an objective function for segmentation, along with a 3D *U-Net*, which they called *V-Net*. This coefficient is a statistical score to measure the similarity between two sets A and B:

$$DSC = \frac{|A \cap B|}{|A| + |B|} \tag{4.2}$$

This metric can be used to measure the disparity between the predicted segmentation masks of each class and the corresponding ground truth mask. As the output is a set of probabilities and not the mask itself, the Soft Dice Metric can be used, where instead of using thresholding to get the predicted mask and intersect with the ground-truth mask, we can make use of the probabilities to make a weighted mask. Hence, the used metric is given by:

$$D = \frac{2TP}{2TP + FP + FN} = \frac{2\sum_n^N \hat{p}_n * y_n}{2\sum_n^N \hat{p}_n * y_n + \hat{p}_n * (1 - y_n) + (1 - \hat{p}_n) * y_n} \tag{4.3}$$

where $\hat{p}$ is the output probability matrix, y is the ground-truth and $*$ represents the element-wise multiplication operation. This metric is calculated for each class and then averaged.

Note that these coefficients are always between 0 and 1, with values close to 1 indicating that the predicted map is very close to the segmentation ground-truth. As the goal is to approximate the function's maximum, the loss function must be the negative coefficient:

$$L_{Dice} = -D \tag{4.4}$$

4.2.2 Cross-entropy Loss

As Dice loss may lose accuracy with batch-based learning and does not deal well with class oversampling [7], authors of *nnU-Net* empirically note that these hitches may be overcome by combining it with cross-entropy loss. This loss is used in many deep learning tasks and is given by:

$$L_{CE} = -\sum_{n}^{N} y_n \cdot \log \hat{p}_n \tag{4.5}$$

4.3 Hyperparameters

Most of the hyperparameters are decided on the *nnU-Net* pipeline, as explained in Chapter 2. During learning, each batch comprises two $128 \times 128 \times 128$ patches. Patches are randomly sampled from training cases, assuring that one-third of them contains foreground voxels.

nnU-Net also proposes to train the models for 1000 epochs. Each epoch comprises 250 training iterations, each composed of a forward and backward pass of a mini-batch. The learning rate is initialized at 0.01 (with a Nesterov momentum of 0.99) and decayed using the *polyLR* policy. This policy generates a decaying learning rate given by:

$$lr = lr0 * (1 - epoch/maxepoch)^{0.9} \tag{4.6}$$

where $lr0$ is the initial learning rate (0.01) and $maxepoch$ is the last epoch. Figure 4.1 shows the learning rate variation over the epochs.

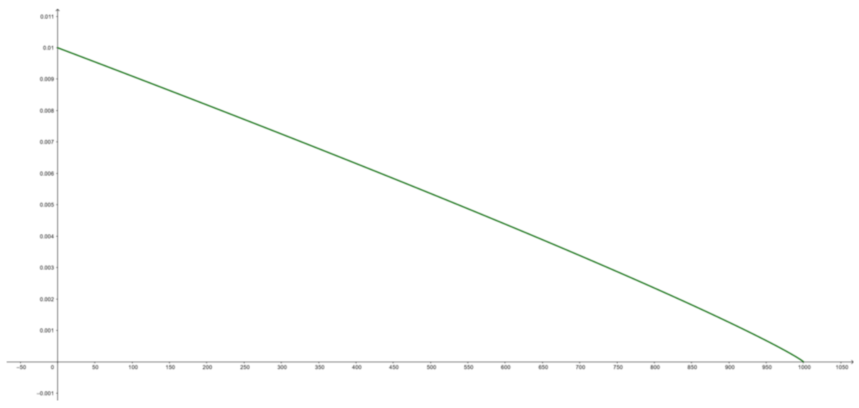


Figure 4.1: Learning Rate as a function of the epoch number

However, this strategy could not always be followed, as covered in the next chapter.

4.4 Data Preprocessing and Augmentation

One of the most useful properties of the *nnU-Net* is its preprocessing planning and pipeline. Two main operations are performed during *nnU-Net*'s preprocessing pipeline: intensity normalization and resampling.

For CT images, which is the case of KiTS, the strategy proposed by the *nnU-Net*'s authors, and followed in this work is to use the 0.5 and 0.95 percentiles of the foreground voxels for clipping and then normalizing using the global foreground mean and standard deviation:

$$I' = \frac{I - \mu_{foreground}}{\sigma_{foreground}} \tag{4.7}$$

where I is the original, clipped intensity, I' is the normalized intensity and $\mu_{foreground}$ and $\sigma_{foreground}$ are the foreground mean and foreground standard deviation of all images, respectively. The usage of the global mean and standard deviation is due to the fact that the voxels' intensity reflects tissue properties in CT images and hence images shouldn't be independently normalized.

For resampling, third-order spline interpolation is used to resample images to the target spacing defined in *nUNet*'s Inferred parameters. Target spacing corresponds to the median of all spacings found in the dataset, which for KiTS corresponds to 0.789. In some images where there is a high discrepancy between the highest and lowest resolution axis, resampling on the lowest resolution one(s) is performed with nearest-neighbour interpolation. For the ground-truth segmentation mask, linear interpolation is used.

Data augmentation is done on the fly during training and is dataset-independent. The following augmentations are applied:

- Rotation
- Scaling
- Gaussian Noise
- Gaussian Blur
- Brightness
- Contrast
- Simulation of low resolution

- Gamm augmentatin
- Mirroring

Every detail about preprocessing and augmentation is described in the original *nnU-Net* paper [7].

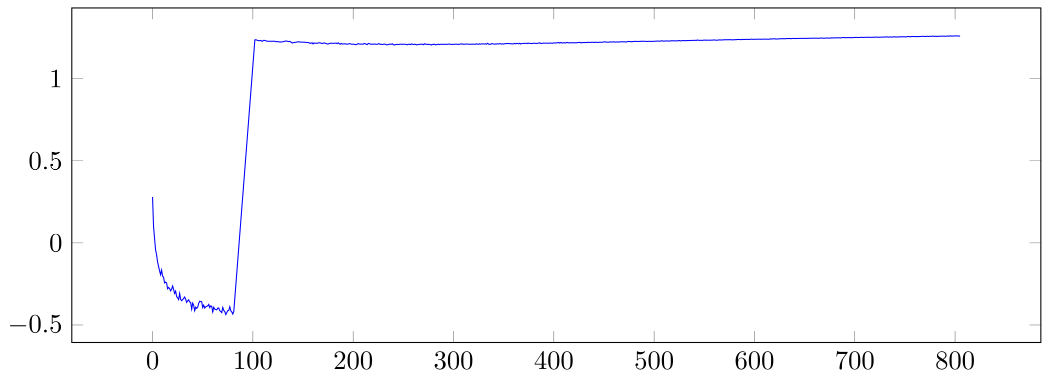
5 Experimental Results

Contents

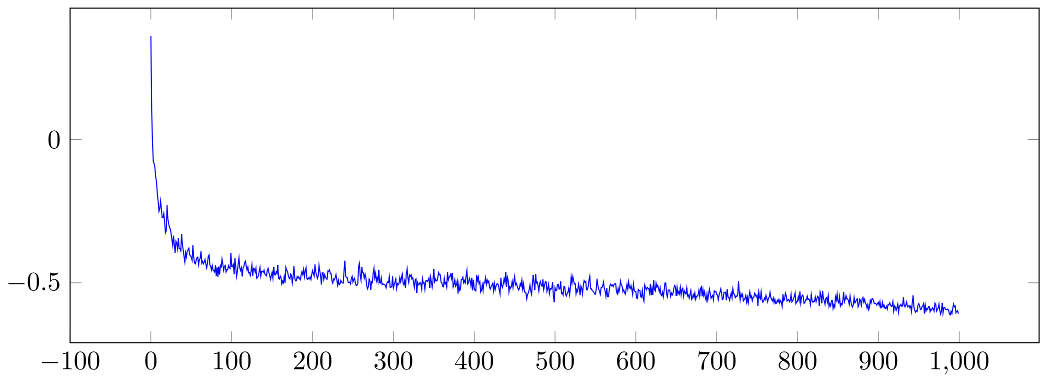
5.1 Overview

As explained in Chapter 4, *nnU-Net* proposes training during 1000 epochs with an initial learning rate of 0.01, which is decayed following the *polyLR* policy. However, some architectures were revealed to be untrainable with these settings, due to gradient explosion. The evaluation metrics are the ones proposed by the challenges that provide the datasets and these are the metrics used for comparison between different architectures. In this chapter, all the challenges that had to be overcome during the training of the networks will be addressed and the results of different architectures will be assessed.

5.2 Challenges



(a) Training loss during training of EfficientUNet



(b) Training loss during training of FullyEfficientUNet

Figure 5.1: Gradient explosion during the training of EfficientUNet-b4. When the decoder is replaced with the novel decoder, originating the FullyEfficientUNet-b4, this problem disappears.

As already mentioned, during the training process of some architectures, gradients and weights exploded, which made the training non-viable with the original parameters. This happened when *Effici-*

enNet-b1 and over replaced the original encoder maintaining the original decoder. If the initial learning rate were decreased to 0.001, this problem would disappear. However, it is important to perceive the extra training epochs that it would take to train models until convergence with a reduced learning rate.

Another problem that was faced was the excessive training time. As each epoch could take up to 15 minutes, the training process with 1000 epochs could take up to more than one week. For architectures with a double decoder (where a VAE is implemented to regularize the encoder training, as mentioned in Chapter 3) each epoch could last up to around 30 minutes. The excessive training time was a consequence not only of the forward passes, but also because of on-the-fly data access and augmentation.

If the number of epochs were increased, the models' learning time would grow to amounts unfeasible for the short time that was available for this work. Hence, the decision to train the networks with the original parameters has been taken and is the explanation for not including results of networks that led to this phenomenon in Chapter 5.

With the novel decoder, presented on Chapter 3, however, the exploding gradient problem disappears, and was also a way to overcome this hitch.

5.3 Metrics

The metrics used for evaluation are the ones proposed by the organizations of the challenges the datasets were obtained from. These are the metrics used for ranking and the most fair for comparison among different models.

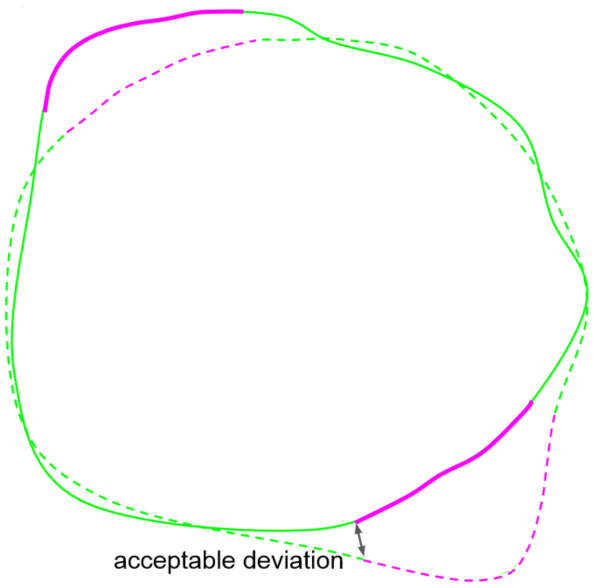


Figure 5.2: Visual representation of surface Dice Score (sDSC). Contours of the predicted segmentation whose distance to the groundtruth is greater than a threshold are considered ”unacceptable”. This metric measures the relation between the ”unacceptable” surface and the whole groundtruth.

KiTS challenge proposes an evaluation that comprises separate evaluations of different foreground classes: kidney, cysts and tumours. However, these are not evaluated individually. Aiming to avoid double penalization in some cases, the evaluation is performed on hierarchical classes:

- Kidney and Masses (Kidney + Tumour + Cyst)
- Kidney Mass (Tumour + Cyst)
- Tumor

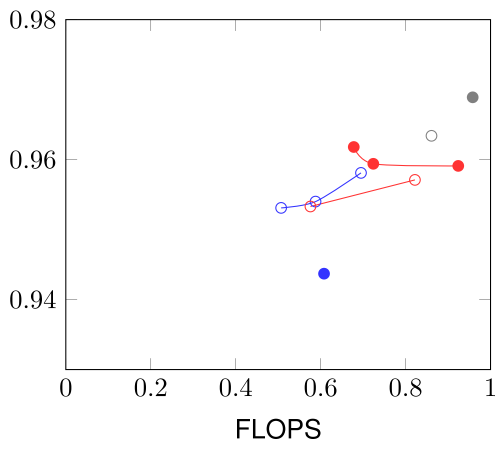
Each of these hierarchical classes is evaluated with the DSC (presented in Chapter 4) and sDSC [60]. The latter (Figure 5.2) firstly analyzes the multiple different segmentation groundtruths to calculate the ”acceptable deviation”, a distance that defines the limit to classify the predicted segmentation’s surface as ”acceptable or ”unacceptable”. This limit corresponds to the 95th percentile of the distances of those segmentations done by professionals. For KiTS this metric is viable, given that for each case there are 3 different annotations done by different professionals.

5.4 Performance Analysis

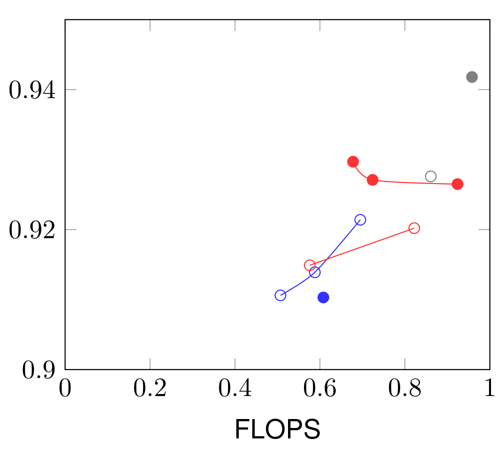
Due to time constraints, not every developed architecture was evaluated. Besides the baseline, the evaluated networks were the following:

- EfficientUNet-b0
- EfficientUNetV2-s
- EfficientUNetV2-m
- EfficientUNetV2-l
- FullyEfficientUNet-b0
- FullyEfficientUNet-b4
- FullyEfficientUNet-b7
- FullyEfficientUNetV2-s
- FullyEfficientUNetV2-l

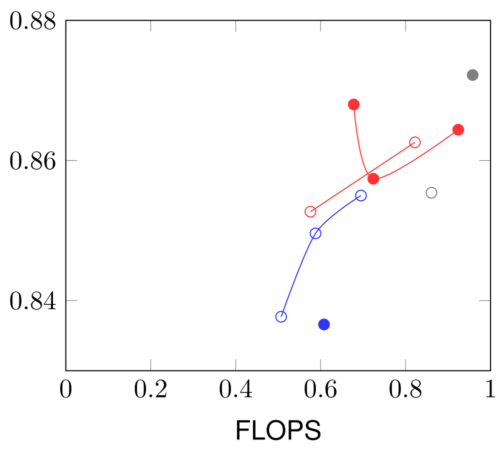
Graphs of Figure 5.3 and B.1 show the results of different encoder-decoder combinations as a function of the number of FLOPs performed in a forward pass and its number of parameters, respectively.



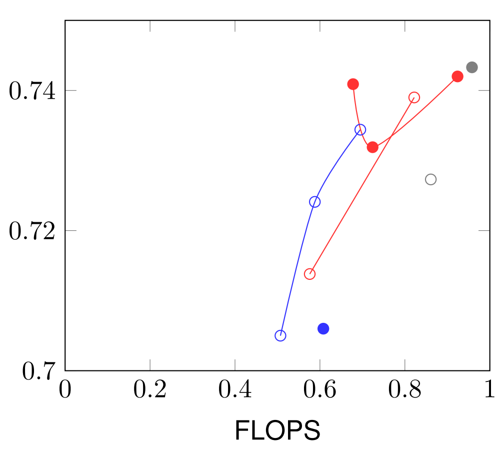
(a) Kidney segmentation DSC



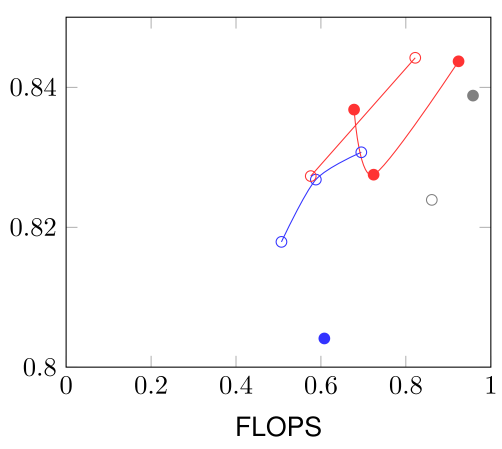
(b) Kidney segmentation sDSC



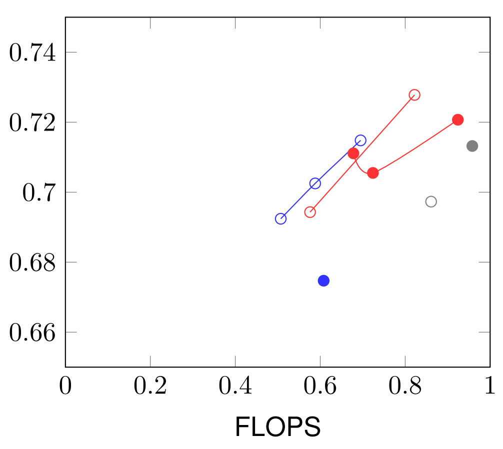
(c) Masses segmentation DSC



(d) Masses segmentation sDSC



(e) Tumour segmentation DSC



(f) Tumour segmentation sDSC

Figure 5.3: Comparison of different encoder-decoder combinations as a function of the number of the number of FLOPs. Architectures with the *EfficientNet* encoder are represented in blue, whereas architectures with the *EfficientNetV2* encoder are represented in red. Original encoder is depicted in gray. Filled circles and open circumferences represent the original and the novel decoder, respectively.

5.4.1 FLOPs vs. performance tradeoff

As it is possible to notice in Figure 5.3, every trained architecture executes fewer FLOPs than the original *nnU-Net* network (depicted in a filled grey circle), maintaining the results very similar: score metrics only vary at most 0.04 among different architectures.

For the whole kidney segmentation (5.3(a), 5.3(b)), the baseline network is clearly the most performative. Among more efficient architectures, *EfficientNetV2* is the family that performs the best. However, an odd event occurs with these encoders: smaller networks perform better than larger ones. This can be explained with overfitting, due to the great number of parameters these networks comprise, as shown in Figure B.1.

Notably, the smaller the region to be segmented, the better the results of efficient architectures in comparison with the baseline. For masses segmentation (5.3(c), 5.3(d)) and tumour segmentation (5.3(e), 5.3(f)), EfficientUNetV2-S shows very similar results to the baseline executing 40% less FLOPs. Moreover, FullyEfficientUNetV2-L shows better results than the baseline in the tumour segmentation task, with 18% less FLOPs.

The impact of the *FullyEfficient* decoder is not clear. In conjunction with the *EfficientNet* encoder, despite the lack of results with the original decoder due to the gradient problem presented in Section 5.2, both FullyEfficientUNet-b0 and FullyEfficientUNet-b1 have shown better results than EfficientUNet-b0 on almost all the tasks. However, for the *EfficientNetV2* encoder family results are not so clear: despite the novel decoder worsening results in the kidney segmentation task, the most performative architecture for tumour segmentation is the FullyEfficientUNetV2-L.

Analysing every graph, the main highlight is possibly the EfficientUNetV2-S architecture, which with less FLOPs and a similar number of parameters is able to achieve similar results to the original *nnU-Net*.

5.4.2 Number of parameters

The number of parameters is a metric used often when comparing different deep learning models and addressing efficiency. However, the number of parameters is not directly linked with efficiency. As a naive example, let's imagine a specific layer from two similar CNNs, with kernels of the same size. The first network applies a convolution with a stride of 2 and the second one applies the same convolution but with a stride of 1. It is trivial to realise that despite having the same number of parameters (kernels were of the same size), the number of FLOPs will be higher in the latter architecture.

The number of parameters may however be related to *overfitting* [14], [61]. It is straightforward to think that the larger the number of parameters, the more the model can be shaped to a particular set of data, not generalizing for other samples.

Graphs that show the performance of each network as a function of the number of parameters are depicted in Appendix B. Results are not so favourable to the efficient architectures. *fvcore* allows developers to analyze whole networks in order to discern the regions of the network which are using more parameters and requiring more FLOPs. As shown in C.1, the required FLOPs and number of parameters in the decoder remains the same among different encoder variations (which counter-intuitively may not be trivial, due to skip connections). Then, the discrepancy in the number of FLOPs comes from the encoder. As an example, the second level of the encoder of the EfficientUNetV2-M comprises 5 *MBConv* blocks. On average, each of these blocks was composed of around 220k parameters and required around 14 GFLOPs. On the other side, the original *nnU-Net* network's second level of the encoder comprised only 2 convolutional blocks with 83k parameters and 43.5 FLOPs each, on average. As each *MBConv* block has more parameters than the original convolutional ones and is repeated more than the former, the number of parameters will rapidly grow for efficient architectures, even though this does not correspond to an increase in the number of FLOPs required.

As far as *overfitting* is concerned, there were no symptoms of it on tasks that envolved the segmentation of smaller volumes, as some architectures with a large number of parameters have shown better results at some tasks than the original *nnU-Net* on masses and tumour segmentation tasks. The absence of *overfitting* may be due to *MBConv*'s residual architecture, that has already proved in the past to hinder this common problem [14], [16]–[18], [22].

5.4.3 Inference Time

Figure 5.4 depicts a comparison of the time different architectures take to execute a forward pass of a single $128 \times 128 \times 128$ volume on an NVIDIA® Tesla V100S. The simulations were run 500 times and averaged, in order to obtain a great level of accuracy.

Contrary to expectations, the decrease in the number of FLOPs does not linearly correspond to a decrease in inference time. Although there is no evident explanation for this, the discrepancy between the variation in the number of FLOPs and the variation of inference time may be explained by varioations on the efficiency of the hardware. As shown in Figure A.3 and Figure 3.1, the introduced encoders comprise more layers than the original one. Although these layers are more compute-efficient than the original ones, as each layer's operations are dependent on the output of the previous layer, the ”parallelizability” of the network tends to decrease. This could explain the fact that EfficientUNetV2-M and EfficientUNetV2-L show a larger inference time than the baseline despite their smaller number of required FLOPs.

However, the novel decoder apparently also increases the running time while reducing the number of parameters and FLOPs. This can also be explained by the fact that *MBConv* blocks perform more sequential operations than the original convolutional blocks. This significant disparity may have other causes, out of the scope of this work.

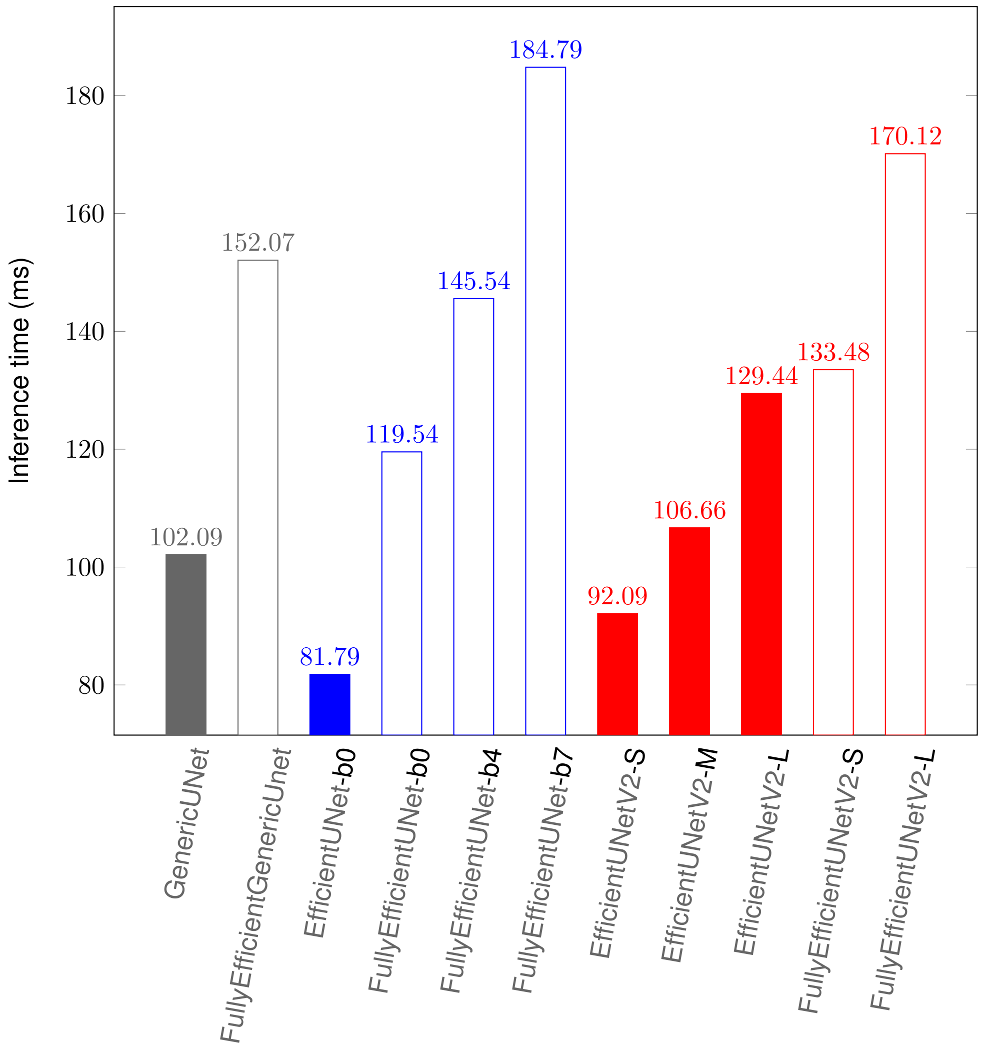


Figure 5.4: Inference time comparison for different architectures, in milliseconds. The time represented corresponds to the time that each network takes to perform a forward pass of a single $128 \times 128 \times 128$ volume. As in Figure 5.3, filled bars represent architectures with the novel decoder, whereas filled ones represent networks with the original decoder. Colours also correspond to previously used ones

6 Conclusions

Contents

6.1 Discussion

This thesis aimed to contribute to the efficiency of medical image segmentation algorithms. Most existing state-of-the-art approaches, although already very accurate, lack in efficiency, requiring a large number of FLOPs to execute and, consequently, a longer inference time. To achieve efficiency, multiple encoders and decoders have been comprehensively tested and attached to the *nnU-Net*, a popular biomedical image segmentation framework.

Every developed architecture revealed to require less FLOPs than the original one, although this is not directly related to a decrease in inference time or in the number of parameters. In fact, some architectures with an *EfficientNet* backbone have shown to require more parameters than the original network, but this didn't lead to *overfit*, possibly due to the residual connections comprised in *MBConv*s and *Fused-MBConv*s. Actually, for the masses segmentation and tumour segmentation tasks, EfficientUNetV2-L and FullyEfficientUNetV2-L performed better than the original *nnU-Net* despite having more parameters and less FLOPs than the latter.

EfficientUNetV2-S, with 30% less FLOPs than the original *nnU-Net* and also a lower inference time is able to achieve very similar results to the baseline. For masses segmentation, this architecture achieves a DSC of 0.868 and a sDSC of 0.7409, very near to the 0.8722 and 0.7433 achieved by the baseline. For the tumour segmentation, EfficientUNetV2-S reaches a DSC of 0.8368 and a sDSC of 0.7111, against the 0.8388 and 0.7132 achieved by the baseline. The best score for the tumour segmentation task was achieved by the FullyEfficientUNetV2-L, with an achieved DSC of 0.8442 and sDSC of 0.7278. This architecture requires less FLOPs to run, despite its larger inference time.

6.2 Summarized Contributions

Summing up, the contributions of this thesis were:

1. Assessment of several architectures with a greater level of computational efficiency than the baseline *nnU-Net* without great loss of accuracy

2. Development of an architecture with better results than the baseline *nnU-Net* at some segmentation tasks, requiring a smaller number of FLOPs

3. Proposal of a new *U-Net* decoder that reduces the number of FLOPs and achieves better results in some cases.

4. Thorough comparison of different encoder-decoder architectures with respect to the required FLOPs, the number of parameters and inference time.

6.3 Future Work

Due to time constraints, some models that were aimed to be trained and tested were not assessed. One of the techniques that performed well in previous works is the VAE regularization [23]. The introduction of a VAE in parallel with the decoder in order to regularize the encoder's training radically increases training time but is able to produce more accurate gradients in order to optimize the encoder. As the VAE is only used during training time, the inference time and the number of FLOPs would remain the same.

Also, more encoder-decoder combinations should be addressed. As it has been shown in this dissertation, architectures that led to gradient explosion have been discarded, because decreasing the learning rate would result in more epochs to achieve convergence and, hence, longer training.

It is also of interest to validate these experiments on other datasets. Other tasks related to medical image segmentation and even other image segmentation may require efficiency and the architectures presented in this work may also be a pathway to achieve it

Finally, this research segment is in rapid development with new algorithms emerging every day, specifically classification architectures that may be used as segmentation backbones. Hence, the assessment of these novel technologies is of interest for the development of segmentation algorithms.

Bibliography

[1] F. de Almeida Fernandes, “Cancro mata mais do que sida, malária e tuberculose juntas,” *Diário de Notícias*, Feb. 4, 2021. [Online]. Available: https://www.dn.pt/sociedade/cancro-mata-mais-do-que-sida-malaria-e-tuberculose-juntas-13312922.html (visited on 02/04/2021).

[2] M. Quaresma, M. P. Coleman, and B. Rachet, “40-year trends in an index of survival for all cancers combined and survival adjusted for age and sex for each cancer in england and wales, 1971-2011: A population-based study,” *The Lancet*, vol. 385, no. 9974, pp. 1206–1218, Mar. 2015. DOI: 10.1016/s0140-6736(14)61396-9. [Online]. Available: https://doi.org/10.1016/s0140-6736(14)61396-9.

[3] A. da Costa Miranda, A. Mayer-da-Silva, L. Glória, and C. Brito, “Registo oncológico nacional de todos os tumores na população residente em portugal, em 2018,” 2020. [Online]. Available: https://ron.min-saude.pt/media/2196/2021-0518_publica%C3%A7%C3%A3o-ron_2018.pdf.

[4] *Risk factors: Age*, Mar. 2021. [Online]. Available: https://www.cancer.gov/about-cancer/causes-prevention/risk/age.

[5] J. M. Lopes, F. Rocha-Gonçalves, M. Borges, P. Redondo, and J. Laranja-Pontes, “Custo do tratamento do cancro em portugal,” 2017. [Online]. Available: https://ecancer.org/en/journal/article/765-the-cost-of-cancer-treatment-in-portugal/pdf/pt.

[6] D. Crosby, S. Bhatia, K. M. Brindle, *et al.*, “Early detection of cancer,” *Science*, vol. 375, no. 6586, Mar. 2022. DOI: 10.1126/science.aay9040. [Online]. Available: https://doi.org/10.1126/science.aay9040.

[7] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, “nnU-net: A self-configuring method for deep learning-based biomedical image segmentation,” *Nature Methods*, vol. 18, no. 2, pp. 203–211, Feb. 2021, ISSN: 1548-7105. DOI: 10.1038/s41592-020-01008-z.

[8] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. DOI: 10.1007/bf02478259. [Online]. Available: https://doi.org/10.1007/bf02478259.

[9] A. G. Ivakhnenko and V. G. Lapa, *Cybernetic Predicting Devices*. CCM Information Corporation, 1965.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[11] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, Apr. 1980. DOI: 10.1007/bf00344251. [Online]. Available: https://doi.org/10.1007/bf00344251.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[13] C. Szegedy, W. Liu, Y. Jia, *et al.*, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298594. [Online]. Available: https://doi.org/10.1109/cvpr.2015.7298594.

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.90. [Online]. Available: https://doi.org/10.1109/cvpr.2016.90.

[15] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.243. [Online]. Available: https://doi.org/10.1109/cvpr.2017.243.

[16] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” *arXiv:1905.11946 [cs, stat]*, Sep. 11, 2020. arXiv: 1905.11946. [Online]. Available: http://arxiv.org/abs/1905.11946 (visited on 10/06/2021).

[17] A. G. Howard, M. Zhu, B. Chen, *et al.*, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017. DOI: 10.48550/ARXIV.1704.04861. [Online]. Available: https://arxiv.org/abs/1704.04861.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00474. [Online]. Available: https://doi.org/10.1109/cvpr.2018.00474.

[19] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, *Autoaugment: Learning augmentation policies from data*, 2018. DOI: 10.48550/ARXIV.1805.09501. [Online]. Available: https://arxiv.org/abs/1805.09501.

[20] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00745. [Online]. Available: https://doi.org/10.1109/cvpr.2018.00745.

[21] P. Ramachandran, B. Zoph, and Q. V. Le, *Searching for activation functions*, 2017. DOI: 10.48550/ARXIV.1710.05941. [Online]. Available: https://arxiv.org/abs/1710.05941.

[22] M. Tan and Q. V. Le, “Efficientnetv2: Smaller models and faster training,” 2021. DOI: 10.48550/ARXIV.2104.00298. [Online]. Available: https://arxiv.org/abs/2104.00298.

[23] NVIDIA. “Image segmentation.” (2021), [Online]. Available: https://catalog.ngc.nvidia.com/orgs/nvidia/collections/imagesegmentation (visited on 09/09/2022).

[24] C. A. Glasbey and G. Horgan, *Image analysis for the biological sciences*, en, ser. Wiley Series in Statistics in Practice. Chichester, England: John Wiley & Sons, Mar. 1995.

[25] A. Şengür, İ. Türkoğlu, and M. C. İnce, “Unsupervised image segmentation using markov random fields,” in *Artificial Intelligence and Neural Networks*, Springer Berlin Heidelberg, 2006, pp. 158–167. DOI: 10.1007/11803089_19. [Online]. Available: https://doi.org/10.1007/11803089_19.

[26] T. Kapur, W. Eric, L. Grimson, R. Kikinis, and W. M. Wells, “Enhanced spatial priors for segmentation of magnetic resonance imagery,” in *Medical Image Computing and Computer-Assisted Intervention — MICCAI’98*, Springer Berlin Heidelberg, 1998, pp. 457–468. DOI: 10.1007/bfb0056231. [Online]. Available: https://doi.org/10.1007/bfb0056231.

[27] G. Coleman and H. Andrews, “Image segmentation by clustering,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, 1979. DOI: 10.1109/PROC.1979.11327.

[28] J. C. Bezdek, L. O. Hall, and L. P. Clarke, “Review of MR image segmentation techniques using pattern recognition,” *Medical Physics*, vol. 20, no. 4, pp. 1033–1048, Jul. 1993. DOI: 10.1118/1.597000. [Online]. Available: https://doi.org/10.1118/1.597000.

[29] Z. Liang, J. MacFall, and D. Harrington, “Parameter estimation and tissue segmentation from multispectral MR images,” *IEEE Transactions on Medical Imaging*, vol. 13, no. 3, pp. 441–449, 1994. DOI: 10.1109/42.310875. [Online]. Available: https://doi.org/10.1109/42.310875.

[30] J. Maintz and M. A. Viergever, “A survey of medical image registration,” *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, Mar. 1998. DOI: 10.1016/s1361-8415(01)80026-8. [Online]. Available: https://doi.org/10.1016/s1361-8415(01)80026-8.

[31] D. Ciresan, A. Giusti, L. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/459a4ddcb586f24efd9395aa7662bc7c-Paper.pdf.

[32] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28. [Online]. Available: https://doi.org/10.1007/978-3-319-24574-4_28.

[33] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298965. [Online]. Available: https://doi.org/10.1109/cvpr.2015.7298965.

[34] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. DOI: 10.48550/ARXIV.1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556.

[35] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, *Hypercolumns for object segmentation and fine-grained localization*, 2014. DOI: 10.48550/ARXIV.1411.5752. [Online]. Available: https://arxiv.org/abs/1411.5752.

[36] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen, “Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks,” in *2013 IEEE International Conference on Computer Vision*, IEEE, Dec. 2013. DOI: 10.1109/iccv.2013.269. [Online]. Available: https://doi.org/10.1109/iccv.2013.269.

[37] F. Milletari, N. Navab, and S.-A. Ahmadi, *V-net: Fully convolutional neural networks for volumetric medical image segmentation*, 2016. arXiv: 1606.04797 [cs.CV].

[38] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, “3d u-net: Learning dense volumetric segmentation from sparse annotation,” in *Medical Image Computing and Computer-Assisted Intervention — MICCAI 2016*, Springer International Publishing, 2016, pp. 424–432. DOI: 10.1007/978-3-319-46723-8_49. [Online]. Available: https://doi.org/10.1007/978-3-319-46723-8_49.

[39] O. Oktay, J. Schlemper, L. L. Folgoc, *et al.*, “Attention U-Net: Learning Where to Look for the Pancreas,” *arXiv:1804.03999 [cs]*, May 2018, arXiv: 1804.03999. [Online]. Available: http://arxiv.org/abs/1804.03999 (visited on 12/14/2021).

[40] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, Springer International Publishing, 2018, pp. 3–11. DOI: 10.1007/978-3-030-00889-5_1. [Online]. Available: https://doi.org/10.1007/978-3-030-00889-5_1.

[41] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional DenseNets for semantic segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jul. 2017. DOI: 10.1109/cvprw.2017.156. [Online]. Available: https://doi.org/10.1109/cvprw.2017.156.

[42] J. Lourenço-Silva, M. N. Menezes, T. Rodrigues, B. Silva, F. J. Pinto, and A. L. Oliveira, "Encoder-decoder architectures for clinically relevant coronary artery segmentation," in *Computational Advances in Bio and Medical Sciences*, Springer International Publishing, 2022, pp. 63–78. DOI: 10.1007/978-3-031-17531-2_6. [Online]. Available: https://doi.org/10.1007/978-3-031-17531-2_6.

[43] T. J. Jun, J. Kweon, Y.-H. Kim, and D. Kim, "T-net: Nested encoder–decoder architecture for the main vessel segmentation in coronary angiography," *Neural Networks*, vol. 128, pp. 216–233, Aug. 2020, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.05.002. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2020.05.002.

[44] D. C. Castro, I. Walker, and B. Glocker, "Causality matters in medical imaging," *Nature Communications*, vol. 11, no. 1, Jul. 2020. DOI: 10.1038/s41467-020-17478-w. [Online]. Available: https://doi.org/10.1038/s41467-020-17478-w.

[45] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "No new-net," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International Publishing, 2019, pp. 234–244. DOI: 10.1007/978-3-030-11726-9_21. [Online]. Available: https://doi.org/10.1007/978-3-030-11726-9_21.

[46] X. Hou, C. Xie, F. Li, and Y. Nan, "Cascaded Semantic Segmentation for Kidney and Tumor," in *Submissions to the 2019 Kidney Tumor Segmentation Challenge: KiTS19*, University of Minnesota Libraries Publishing, 2019. DOI: 10.24926/548719.002. [Online]. Available: https://kits.lib.umn.edu/cascaded-semantic-segmentation-for-kidney-and-tumor/ (visited on 12/21/2021).

[47] Y. Zhang, Y. Wang, F. Hou, *et al.*, "Cascaded volumetric convolutional network for kidney tumor segmentation from CT volumes," in *Submissions to the 2019 Kidney Tumor Segmentation Challenge: KiTS19*, University of Minnesota Libraries Publishing, 2019. DOI: 10.24926/548719.004. [Online]. Available: https://doi.org/10.24926/548719.004.

[48] Y. George, “A coarse-to-fine 3d u-net network for semantic segmentation of kidney CT scans,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2022, pp. 137–142. DOI: 10.1007/978-3-030-98385-7_18. [Online]. Available: https://doi.org/10.1007/978-3-030-98385-7_18.

[49] Z. Zhao, H. Chen, and L. Wang, “A coarse-to-fine framework for the 2021 kidney and kidney tumor segmentation challenge,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2022, pp. 53–58. DOI: 10.1007/978-3-030-98385-7_8. [Online]. Available: https://doi.org/10.1007/978-3-030-98385-7_8.

[50] A. Golts, D. Khapun, D. Shats, Y. Shoshan, and F. Gilboa-Solomon, “An ensemble of 3d u-net based models for segmentation of kidney and masses in CT scans,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2022, pp. 103–115. DOI: 10.1007/978-3-030-98385-7_14. [Online]. Available: https://doi.org/10.1007/978-3-030-98385-7_14.

[51] X. Yang, J. Zhang, J. Zhang, and Y. Xia, “Transfer learning for KiTS21 challenge,” in *Lecture Notes in Computer Science*, Springer International Publishing, 2022, pp. 158–163. DOI: 10.1007/978-3-030-98385-7_21. [Online]. Available: https://doi.org/10.1007/978-3-030-98385-7_21.

[52] A. Myronenko and A. Hatamizadeh, “3D Kidneys and Kidney Tumor Semantic Segmentation using Boundary-Aware Networks,” *arXiv:1909.06684 [cs, eess]*, Sep. 2019, arXiv: 1909.06684. [Online]. Available: http://arxiv.org/abs/1909.06684 (visited on 12/21/2021).

[53] A. Myronenko, “3d MRI brain tumor segmentation using autoencoder regularization,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International Publishing, 2019, pp. 311–320. DOI: 10.1007/978-3-030-11726-9_28. [Online]. Available: https://doi.org/10.1007978-3-030-11726-9_28.

[54] Z. Jiang, C. Ding, M. Liu, and D. Tao, “Two-Stage Cascaded U-Net: 1st Place Solution to BraTS Challenge 2019 Segmentation Task,” en, in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 231–241, ISBN: 978-3-030-46640-4. DOI: 10.1007/978-3-030-46640-4_22.

[55] Y. Yuan, “Automatic brain tumor segmentation with scale attention network,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International Publishing, 2021, pp. 285–294. DOI: 10.1007/978-3-030-72084-1_26. [Online]. Available: https://doi.org/10.1007/978-3-030-72084-1_26.

[56] Y. Wang, Y. Zhang, F. Hou, *et al.*, “Modality-pairing learning for brain tumor segmentation,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International

Publishing, 2021, pp. 230–240. DOI: 10.1007/978-3-030-72084-1_21. [Online]. Available: https://doi.org/10.1007/978-3-030-72084-1_21.

[57] F. Isensee, P. F. Jäger, P. M. Full, P. Vollmuth, and K. H. Maier-Hein, “nnU-net for brain tumor segmentation,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International Publishing, 2021, pp. 118–132. DOI: 10.1007/978-3-030-72087-2_11. [Online]. Available: https://doi.org/10.1007/978-3-030-72087-2_11.

[58] H. Jia, W. Cai, H. Huang, and Y. Xia, “H2nf-net for brain tumor segmentation using multimodal MR imaging: 2nd place solution to BraTS challenge 2020 segmentation task,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, Springer International Publishing, 2021, pp. 58–68. DOI: 10.1007/978-3-030-72087-2_6. [Online]. Available: https://doi.org/10.1007/978-3-030-72087-2_6.

[59] R. McKinley, M. Rebsamen, R. Meier, and R. Wiest, “Triplanar Ensemble of 3D-to-2D CNNs with Label-Uncertainty for Brain Tumor Segmentation,” en, in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 379–387, ISBN: 978-3-030-46640-4. DOI: 10.1007/978-3-030-46640-4_36.

[60] S. Nikolov, S. Blackwell, A. Zverovitch, *et al.*, *Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy*, 2018. DOI: 10.48550/ARXIV.1809.04430. [Online]. Available: https://arxiv.org/abs/1809.04430.

[61] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, *Understanding deep learning requires rethinking generalization*, 2016. DOI: 10.48550/ARXIV.1611.03530. [Online]. Available: https://arxiv.org/abs/1611.03530.

Glossary

EfficientUNet

U-Net architecture with the *EfficientNet* encoder and the original *nnUNet* decoder . . . . . ix, 37, 39, 41, 43, 66

EfficientUNetV2

U-Net architecture with the *EfficientNetV2* encoder and the original *nnUNet* decoder 39, 41–43, 47, 63, 65, 66

FullyEfficientGenericUnet

U-Net architecture with the original *nnUNet* encoder and the novel *FullyEfficient* decoder 43, 66

FullyEfficientUNet

U-Net architecture with the *EfficientNet* encoder and the novel *FullyEfficient* decoder . . . . . ix, 37, 39, 41, 43, 66

FullyEfficientUNetV2

U-Net architecture with the *EfficientNetV2* encoder and the novel *FullyEfficient* decoder 39, 41, 43, 47, 65, 66

GenericUNet

Original *nnUNet* network. 43, 66

A Encoders

This appendix aims to give a visual representation of the architecture of the different encoders that were tested along with the *nnU-Net*. Only the smallest variation of each encoder is depicted, as the others are very similar, only changing the number of convolutions, in most cases. As described in Chapter 3, *EfficientNet* and *EfficientNetV2* encoders consist of the *EfficientNet* and *EfficientNetV2* networks with some small changes: the first convolution is done without downsizing (stride 1), in order to fit in the first encoder layer; the second convolutional block is performed with stride 2 and not 1, to compensate for the first convolution; and the last convolutional block is pruned, because it is used is for classification purposes only.

It is impportant to notice that different *EfficientNet* and *EfficientNetV2* stages do not necessarily correspond to different encoder stages in the *U-Net*. This happens because there are some *EfficientNet*'s levels without downsizing and therefore there isn't a change of encoder level.

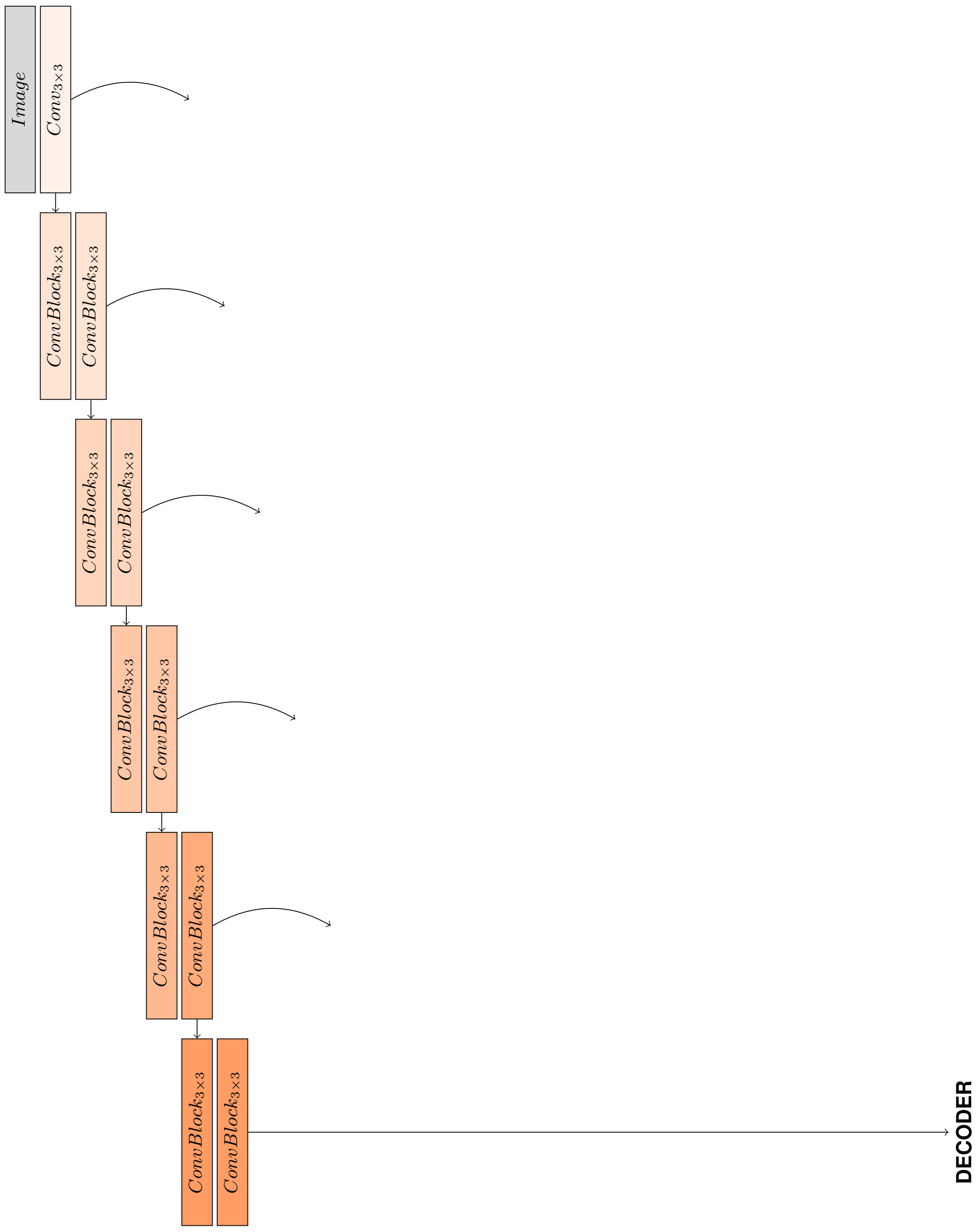


Figure A.1: Original *nnUNet* encoder. Skip connections are marked with a bent arrow. The first convolutional block of each encoder level has stride 2. Each convolutional block corresponds to 2 convolutions followed by a dropout normalization layer and a ReLU activation function.

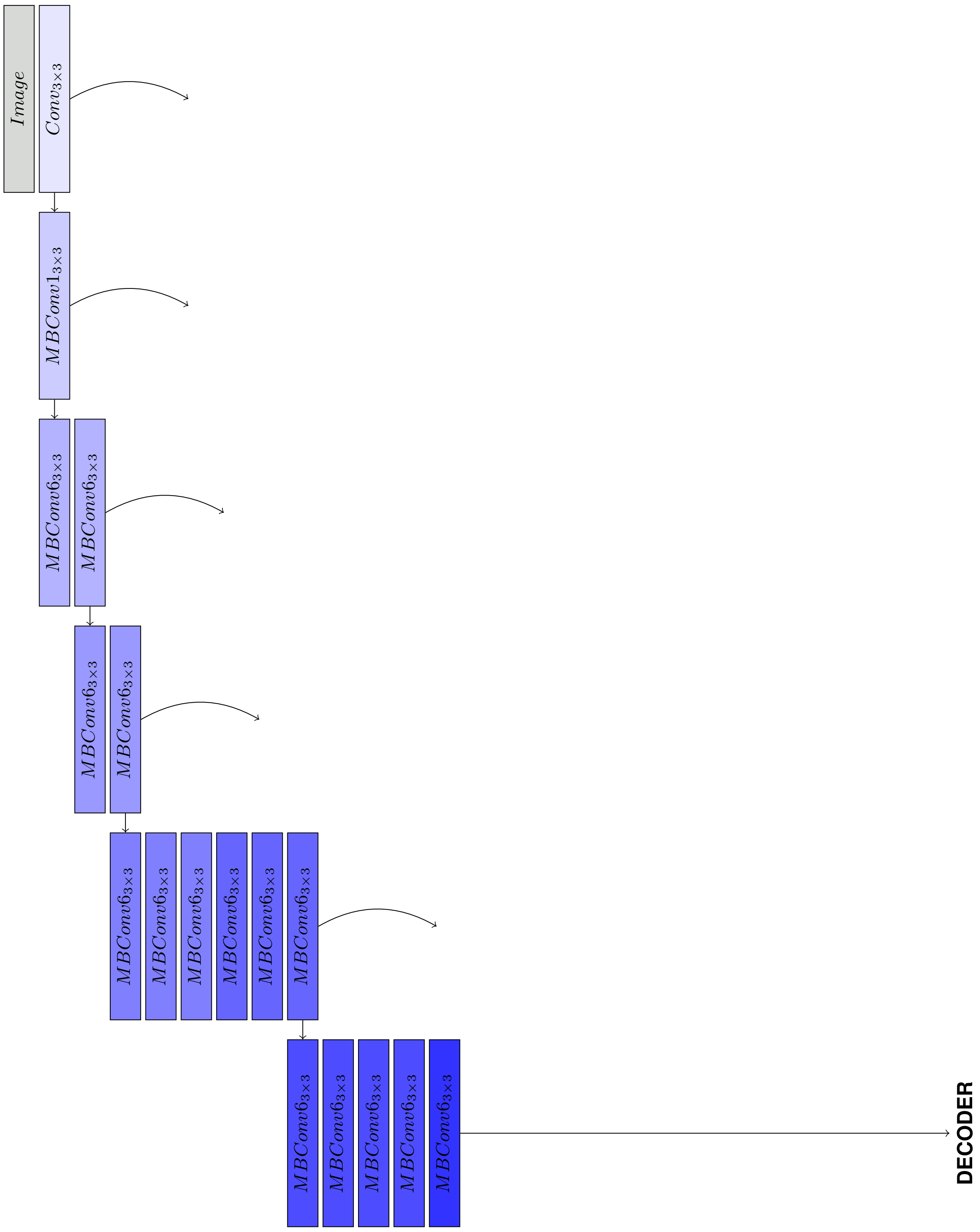


Figure A.2: *EfficientNet*-b0 encoder, as also depicted on Figure 3.1. Skip connections are marked with a bent arrow. The first convolutional block of each encoder level has stride 2. Different colour tonalities means distinct *EfficientNet* stages. Note that each encoder stage may contain convolutions from different *EfficientNetV2* stages. The main building blocks are *MBConv*s.

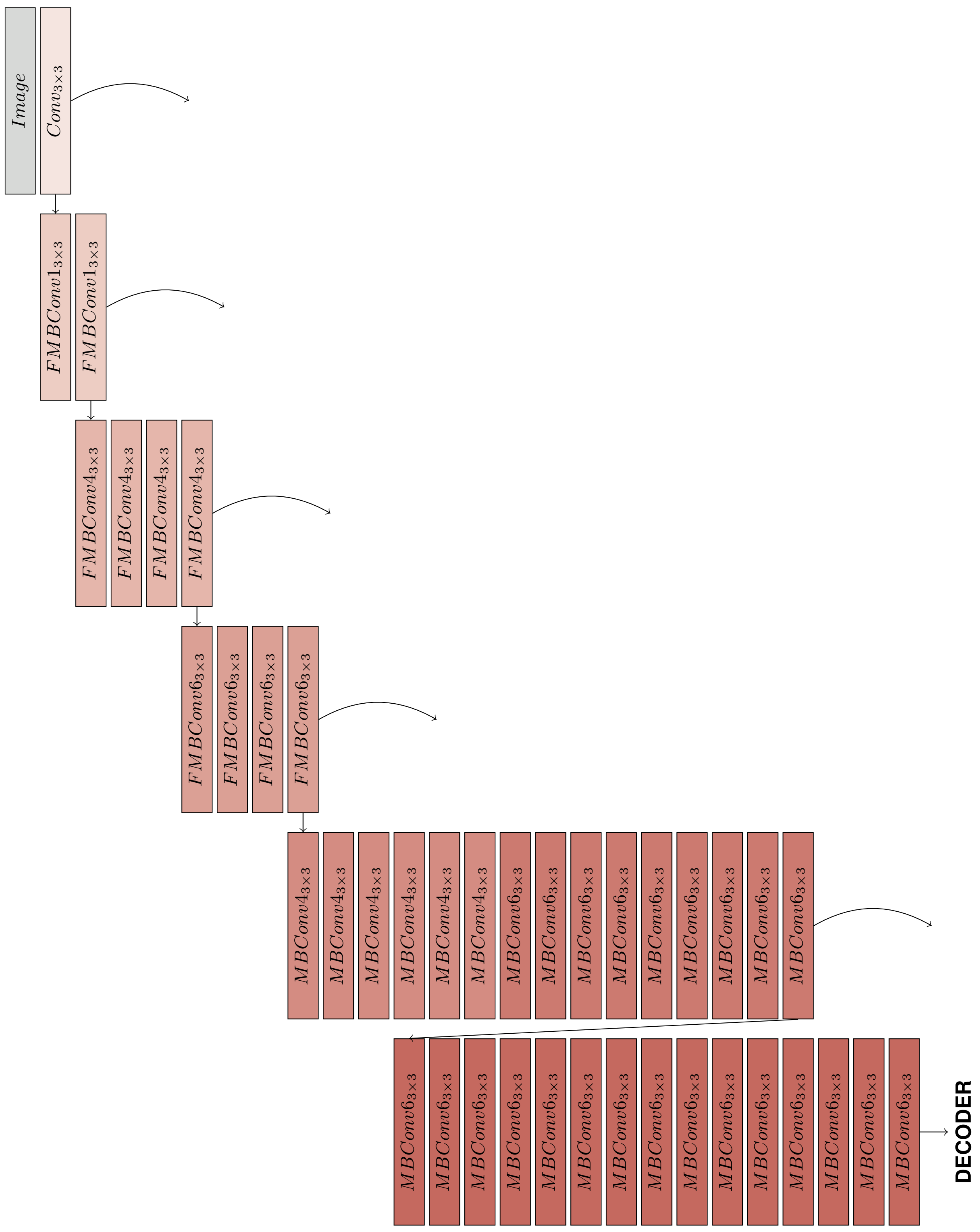
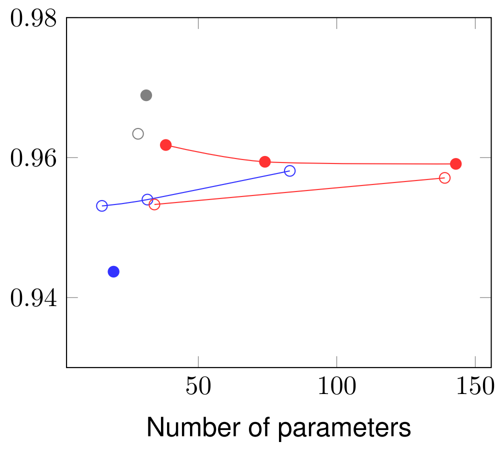


Figure A.3: *EfficientNetV2*-S encoder. Skip connections are marked with a bent arrow. The first convolutional block of each encoder level has stride 2, Different colour tonalities means distinct *EfficientNetV2* stages. Note that each encoder stage may contain convolutions from different *EfficientNetV2* stages. This encoder comprises *MBConv* blocks as well as *Fused-MBConv* ones.

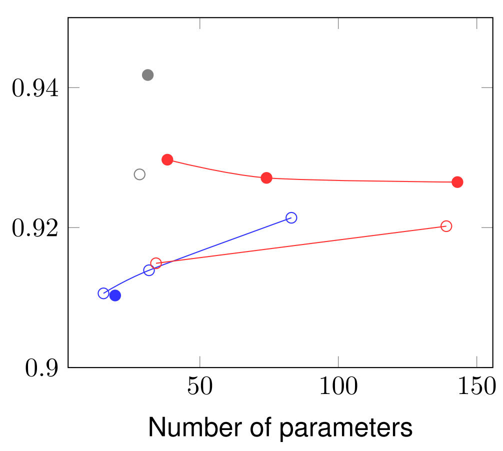
B Number of Parameters Comparison

Notably, a considerable part of the efficient architectures have more parameters than the original *nnUNet* network. This happens because *MBConv* blocks comprise a large number of parameters and moreover are repeated many times. For example, the *EfficientNetV2*-M encoder is composed of 57 *MBConv* blocks, a lot more than the 12 convolutional blocks used in the original *nnUNet*. Specifically, the second level of the encoder of the EfficientUNetV2-M comprises 5 *MBConv* blocks. On average, each of these blocks is composed of around 220k parameters and required around 14 GFLOPs. On the other side, the original *nnUNet* network's second level of the encoder comprised only 2 convolutional blocks with 83k parameters and 43.5 FLOPs each, on average. As each *MBConv* block has more parameters than the original convolutional ones and is repeated more than the former, the number of parameters will rapidly grow for efficient architectures, even though this does not correspond to an increase in the number of FLOPs required.

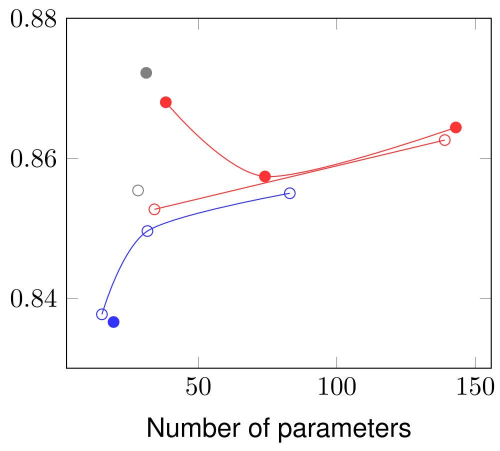
Although sometimes an excessive number of parameters may lead to overfitting, the only situation where that scenario may have been the reality is with the EfficientUNetV2 family, on the kidney segmentation task, where accuracy degrades with more parameters.



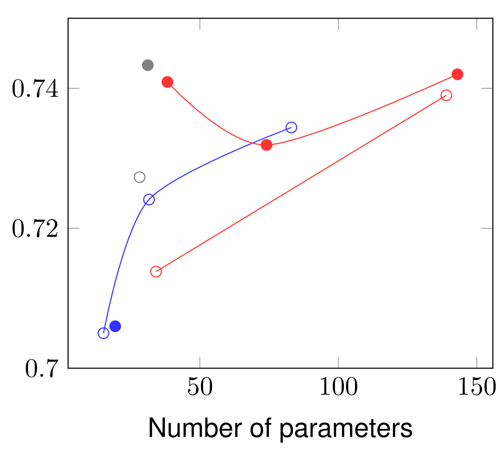
(a) Kidney segmentation DSC



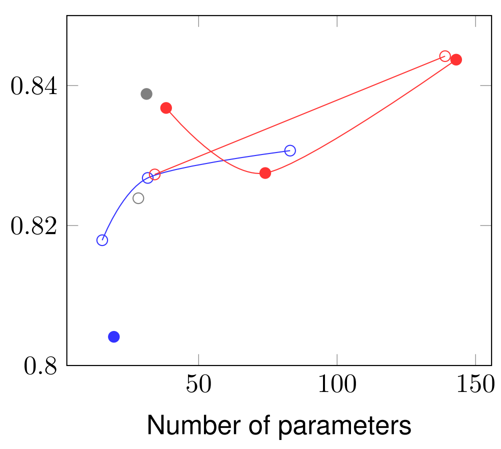
(b) Kidney segmentation sDSC



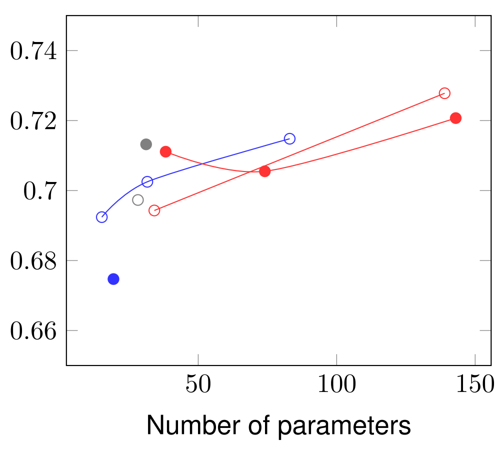
(c) Masses segmentation DSC



(d) Masses segmentation sDSC



(e) Tumour segmentation DSC



(f) Tumour segmentation sDSC

Figure B.1: Comparison of different encoder-decoder combinations as a function of the number of the number of parameters. Architectures with the *EfficientNet* encoder are represented in blue, whereas architectures with the *EfficientNetV2* encoder are represented in red. Original encoder is depicted in gray. Filled circles and open circumferences represent the original and the novel decoder, respectively.

C

Complete Comparison

This appendix contains the complete comparison of the different networks on the KiTS dataset. The FullyEfficientUNetV2-L and EfficientUNetV2-L are more performative than the baselineon the tumour segmentation task, despite the lower number of FLOPs required.

Name	# FLOPs			# Parameters			Inference Time	Performance (DSC)			Performance (sDSC)		
Network	Network	Enc.	Dec.	Network	Enc.	Dec.	Network	Kidney	Masses	Tumour	Kidney	Masses	Tumour
GenericUNet	0.958T	0.279T	0.663T	31.196M	14.025M	15.349M	102.09s	**0.9689**	**0.8722**	0.8388	**0.9418**	**0.7433**	0.7132
FullyEfficientGenericUnet	0.861T	0.279T	0.565T	28.246M	14.025M	12.399M	152.07s	0.9634	0.8554	0.8239	0.9276	0.7273	0.6973
EfficientUNet-b0	0.608T	9.658G	0.581T	19.408M	5.969M	11.617M	81.79s	0.9437	0.8366	0.8041	0.9103	0.706	0.6747
EfficientUNet-b1	0.61T	11.926G	0.581T	22.165M	8.727M	11.617M	-	-	-	-	-	-	-
EfficientUNet-b2	0.611T	12.503G	0.582T	23.768M	10.124M	11.741M	-	-	-	-	-	-	-
EfficientUNet-b3	0.655T	17.916G	0.62T	27.509M	13.595M	11.928M	-	-	-	-	-	-	-
EfficientUNet-b4	0.69T	23.469G	0.65T	35.753M	21.406M	12.198M	-	-	-	-	-	-	-
EfficientUNet-b5	0.701T	32.024G	0.652T	47.829M	33.097M	12.419M	-	-	-	-	-	-	-
EfficientUNet-b6	0.749T	42.989G	0.689T	61.875M	46.697M	12.702M	-	-	-	-	-	-	-
EfficientUNet-b7	0.797T	59.897G	0.721T	86.96M	71.32M	12.999M	-	-	-	-	-	-	-
FullyEfficientUNet-b0	0.507T	9.658G	0.481T	15.201M	5.969M	7.41M	119.54s	0.9531	0.8377	0.8179	0.9106	0.705	0.6924
FullyEfficientUNet-b1	0.509T	11.926G	0.481T	17.958M	8.727M	7.41M	-	-	-	-	-	-	-
FullyEfficientUNet-b2	0.51T	12.503G	0.481T	19.574M	10.124M	7.547M	-	-	-	-	-	-	-
FullyEfficientUNet-b3	0.553T	17.916G	0.519T	23.344M	13.595M	7.763M	-	-	-	-	-	-	-
FullyEfficientUNet-b4	0.69T	23.469G	0.65T	35.753M	21.406M	12.198M	145.54s	0.954	0.8496	0.8268	0.9139	0.7241	0.7025
FullyEfficientUNet-b5	0.599T	32.024G	0.55T	43.756M	33.097M	8.346M	-	-	-	-	-	-	-
FullyEfficientUNet-b6	0.647T	42.989G	0.587T	57.874M	46.697M	8.702M	-	-	-	-	-	-	-
FullyEfficientUNet-b7	0.695T	59.897G	0.618T	83.044M	71.32M	9.083M	184.79s	0.9581	0.855	0.8307	0.9214	0.7344	0.7148
EfficientUNetV2-S	0.678T	94.991G	0.567T	38.267M	24.321M	12.287M	92.094s	0.9618	0.868	0.8368	0.9297	0.7409	0.7111
EfficientUNetV2-M	0.724T	0.139T	0.568T	74.065M	59.216M	12.536M	106.66	0.9594	0.8574	0.8275	0.9271	0.7319	0.7055
EfficientUNetV2-L	0.924T	0.327T	0.58T	0.143G	0.128G	13.131M	129.44s	0.9591	0.8644	0.8437	0.9265	0.742	0.7207
FullyEfficientUNetV2-S	0.576T	94.991G	0.465T	34.163M	24.321M	8.183M	133.48s	0.9533	0.8527	0.8273	0.9149	0.7138	0.6943
FullyEfficientUNetV2-M	0.622T	0.139T	0.466T	70.008M	59.216M	8.48M	-	-	-	-	-	-	-
FullyEfficientUNetV2-L	0.822T	0.327T	0.478T	0.139G	0.128G	9.244M	170.12s	0.9571	0.8626	**0.8442**	0.9202	0.739	**0.7278**

Table C.1: Complete comparison of different networks. Missing data is due to lack of time or gradient explosion. Every network was trained during 1000 epochs with an initial training rate of 0.01.