Neural Architecture Search for Medical Image Segmentation

Martijn Bosma

M.Sc. Thesis

Neural Architecture Search for Medical Image Segmentation

by

Martijn Bosma

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday January 17th, 2021 at 09:30.

Student number:	4367340		
Project duration:	December, 2020 – January, 2022		
Thesis committee:	Prof. Dr. P. A. N. Bosman,	CWI, TU Delft	Chair and supervisor
	Dr. T. Alderliesten,	LUMC	Committee member and supervisor
	Dr. J. van Gemert,	TU Delft	Committee member
	M.Sc. M. Grewal,	CWI	Daily supervisor
	M.Sc. A. Dushatskiy,	CWI	Daily supervisor

Abstract

Deep Neural Networks (DNNs) have the potential to make various clinical procedures more time-efficient by automating medical image segmentation; largely due to their strong, in some cases human-level, performance. The design of the best possible medical image segmentation DNN, however, is task-specific. Neural Architecture Search (NAS), i.e., the automation of neural network design, has been shown to have the capability to outperform manually designed networks. However, the existing NAS methods for medical image segmentation have explored a quite limited range of types of DNN architectures that can be discovered, and, more importantly, do not evaluate the accuracy of performance estimation methods.

In this thesis, various performance estimation methods for DNNs are analysed for medical image segmentation tasks. Due to the use of different metrics, small datasets, and inter-physician variability, DNN performance values are susceptible to considerable noise. Through experiments on multiple datasets, it is shown that performance estimation needs to be more elaborate than proposed in previous literature on NAS for medical image segmentation. Only then can the noise induced by the problem be overcome. Based on evaluations of NAS performance with different levels of noise, a method is put forward to evaluate this noise, such that a more informed decision on performance estimation can be made.

The second contribution of this thesis, is the proposal of a novel NAS search space for medical image segmentation networks. This search space combines the strength of a generalised encoder-decoder structure, well known from U-Net, with network blocks that have been proven to have a strong performance in image classification tasks. The search is performed by looking for the best topology of the network, and simultaneously searching the configuration of each cell. This allows for interactions between topology- and cell-level attributes to be found. Experiments were performed on two publicly available datasets. The networks discovered by the proposed NAS method perform better than well-known handcrafted segmentation networks, and outperform networks found with other NAS approaches that perform only topology search, and topology-level search followed by cell-level search.

Finally, three search algorithms are compared for different performance estimation methods on a realistic clinical medical image segmentation task. The results show that the performance of these algorithms is very similar in noisy environments for initial runs, and show deterioration of performance for all algorithms when correlation with the validation performance values are low. This supports the findings that not adapting performance estimation to the task at hand will lead to poor NAS performance, no matter the chosen search algorithm.

Preface

This thesis is the final work of my MSc. Computer Science. On a larger scale, it represents the end of my career as a student at the TU Delft. It is with pride and fulfillment that I close what has been an exhilarating, educational and formative chapter.

From starting my studies in the United States with a year of Liberal Arts and Sciences, to pursuing a BSc. at the faculty of Mechanical Engineering in Delft, to doing a minor in Biomechanical Engineering in Sweden, it has felt like quite the roller coaster ride to finish (if all goes well) with a MSc. in Computer Science. I am delighted to say that I strongly feel that I have ended up in the right field. I have never felt more passionate about Computer Science and the role it will play in solving not only medical, but also environmental and societal problems.

This adventure has brought me countless and priceless memories. I hope my natural hard drive has enough room to store them all, and keep all the subject matter I have absorbed over the years on there as well. Thankfully, the matter I have read, struggled with, experimented with, and finally put on paper here, will stay with me for a long time.

This thesis discusses Neural Architecture Search for Medical Image Segmentation. It combines the topics of Evolutionary Algorithms and Computer Vision to take on the challenge of interpreting medical images. Having started with only one deep learning project in my locker, it has taken quite some reading up, literature exploration, coding, experiment setup, and interpretation, to get here. However, I am very proud of the result. I hope you enjoy the read!

Before proceeding with the contents of this paper, I would like to extend my gratitude to the people who encouraged me, and at moments, dragged me through the process known as research. Having started my thesis during the Covid-19 lockdown has brought along extra challenges, and I am very happy to say that my thesis and the lockdown is drawing to an end.

First of all, to Prof. Dr. Peter Bosman, thank you for inspiring me and pushing me to pursue my curiosity from the first minute. Your enthusiasm and thirst for knowledge truly rub off and have helped me get the most out of this thesis. Alongside your incredible amount of knowledge of so many different fields within Computer Science, I have felt the supervision could have been no better.

Second, I would like to thank Dr. Tanja Alderliesten, for providing a voice of reason when research ideas grew out of frame, and sharing her knowledge of the many medical aspects of the research. This always managed to put the research in perspective, such that I could pursue results that are relevant and understandable for the community that would benefit most from automated medical image segmentation.

To Dr. Jan van Gemert, I would like to thank you for taking out the time to evaluate my research, and for the lectures and labs that made me understand, and grow fond of the concepts of Deep Learning.

To the people I have to thank most for bearing with me: Thank you Monika Grewal, and thank you Arkadiy Dushatskiy. To Monika, your thorough approach to research, from understanding scientific concepts, to experimental setup, to writing papers, really helped to structure my work, something I need more often than I would like to admit. Arkadiy, your programming skills, the ability to glimpse over my code and help right away, and your knowledge of NAS and interesting papers to watch out for, made my life a whole lot easier. Thanks to both of you for keeping me moving at all times.

In the background, my support team needs mentioning as well! I want to thank my mother Patricia and my father Marco for always being there for me. Not only did you help lay all my foundations, but you have always continued to applaud, sponsor and challenge me throughout my life. I could never thank you enough. I also want to thank my brother, whose excellent performances have always inspired me to think bigger. Without you, I would never have achieved all this. Thanks Sebastian!

To Daphne, my incredible girlfriend, you were there to ride all my lows and highs during my thesis and the pandemic, and you managed to make them all unforgettable. Thank you for always being there and having my back.

Martijn Bosma
Delft, December 2021

Contents

List of Figures

List of Tables

List of Abbreviations

A list of all abbreviations and acronyms used in this thesis, alongside their definition, and the page on which they are first described.

Table 1: Abbreviations

Abbreviation	Meaning	Page found
(D)NN	(Deep) Neural Network	1
SotA	State-of-the-Art	1
NAS	Neural Architecture Search	1
CNN	Convolutional Neural Network	7
ReLU	Rectified Linear activation Unit	10
(S)GD	(Stochastic) Gradient Descent	12
DSC	Dice-Sorensen Similarity Coefficient	13
HD	Hausdorff Distance	14
SD	Surface Dice Similarity Coefficient	14
ADAM	Adaptive Moment Estimation	15
EA	Evolutionary Algorithm	16
SGA	Simple Genetic Algorithm	16
GOMEA	Gene-pool Optimal Mixing Evolutionary Algorithm	17
P3GOMEA	Parameterless Population Pyramid GOMEA	17
LT	Linkage Tree	17
NMI	Normalised Mutual Information	17
LS	Local Search	18
RL	Reinforcement Learning	21
RS	Random Search	21
MSD	Medical Segmentation Decathlon	25
AMC	Amsterdam University Medical Centre	25
MAE	Mean Absolute Error	32
MB-NAS	Mixed-block NAS	43
DARTS	Differentiable Architecture Search	50

1 Introduction

In recent years, a growing amount of clinical applications, such as computer-aided diagnostic systems for disease, have been benefiting from advances in automated medical image interpretation, most notably by using Deep Neural Networks (DNNs) [5, 13]. These artificial systems are inspired by human brains, and can map multi-dimensional inputs to corresponding multi-dimensional outputs. A DNN can learn this mapping by learning from examples, in a process known as neural network training.

In the case of medical image segmentation[1], a DNN is trained with medical scans and delineations provided by physicians. A properly constructed and trained DNN can then provide quick and accurate segmentations on cases it has not seen before, sometimes attaining human-level and clinically acceptable performance [34], and doing so in a matter of seconds.

These high-level performances, however, cannot be achieved by every possible DNN. Designing a State-of-the-Art (SotA) DNN, is often task-specific. Varying dataset sizes, data variability, different regions of interest, changes in data resolution, and different configurations of scanning machinery, can all lead to different DNN architecture[2] and training choices being optimal [20]. For DNNs, the amount of architecture possibilities is inconceivably large, and is impossible to manually navigate through in an exhaustive fashion, or even by means of intelligent design, while ensuring the best choices are being made.

Neural Architecture Search (NAS), i.e., the automated design of Neural Network architectures, can take on a large part of this task of architecture design. If implemented correctly, NAS algorithms can effectively and efficiently search through this space of possible network architecture designs, and find a network that is highly tailored to the task at hand [12]. While research on NAS for medical image segmentation has not been as elaborate as for natural image classification, it has already shown promising results by outperforming SotA architectures [49, 51, 55]. In my opinion, further research on NAS for medical image segmentation can make its contributions even more significant.

NAS involves three key components: (1) the performance estimation method, i.e., the choices made to score the performance of a DNN, such that the evaluated networks can be ranked; (2) the search space, which is the set of all possible networks given the specified architectural constraints; and (3) the search algorithm, or the algorithm used to navigate the search space. This thesis looks at improving all the different components separately, to create an improved NAS method for medical image segmentation.

1.1. Performance estimation

Estimating DNN performance within medical image segmentation starts by quantifying the similarity between the segmentations made by a NN and the segmentations made by physicians. Existing metrics rely on different segmentation properties [6, 9, 34]. This means that decisions have to be made on what property to optimise. Using different metrics will lead to different segmentations being evaluated as "good". The metrics are also noisy in comparison with metrics used for other tasks such as image classification. In medical image segmentation, the performance values per segmented class[3] in an image are computed using

[1]Medical image segmentation is the practice of delineating organs or regions of interest (e.g., tumors) within medical scans.
[2]Architecture is the name given to the blueprint of the DNN. It encompasses the design choices such as size, operation types used, and connections within the network.
[3]A class is a certain organ or region of interest to be segmented or classified in an image.

values per pixel/voxel, whereas for image classification performance values per classified class in an image are binary. This leads to a larger range of possible segmentation similarity scores, and makes it more likely that small changes in the output will affect the score. This larger sensitivity in the metrics, in combination with datasets being relatively small, and the possibility of differences between example cases, due to patient variation, human error and difference in physician interpretation, makes this task of quantifying similarity between segmentations more vulnerable to noise than that of quantifying correctly labeled classes for classification. Research on how noisy these metrics are, however, is understudied in literature, although possibly impacting DNN performance estimation quite significantly. It would be interesting to evaluate this noise and analyse its effect on performance estimation of DNNs.

Next to noise in the data, estimating performance of a DNN on any task has inherent noise, as the initialisation and training of the network is stochastic [42]. The most straightforward way to assess DNN performance after establishing what segmentation similarity metric to use, is to train the DNN and average the score of the provided segmentations on a validation set (see Section 2.3.1) of the training data. However, these network evaluations are very costly. For this reason, a lot of NAS research has gone into creating surrogate models, which try to use other Machine Learning methods [4, 29] or incomplete training predictors [32, 39], i.e., performance estimators that use properties of a partially trained or untrained network, to predict performance. However, these methods add inherent error to the estimated performance value, which could lead to sub-optimal networks being estimated as optimal by NAS algorithms. As there are many possible sources of noise for performance estimation of DNNs for medical image segmentation, it could be possible that these alternate methods, as well as single training run estimations might not reflect the true performance of a network. Literature has even shown that after validation, NAS results often do not outperform random search policies [52, 54], indicating performance estimation differences between the used evaluation method and validation. In Chapter 4, different performance estimation methods are analysed in order to gain insights on the noise in network performance estimation. The research question setup for this task is the following:

Research Question 1. *Can DNN performance estimation for medical image segmentation tasks be adapted to noise such that the performance values passed to search algorithms are accurate enough for them to outperform Random Search?*

1.2. Search space

The search space is the set of all possible networks that a search algorithm can evaluate. Its design largely determines the upper and lower limit performance bound of a NAS algorithm. To set this upper bound to be as least as high as the performance of SotA networks, it is important to include the known SotA networks in the search space. However, only including networks very similar to SotA networks introduces bias, which may prevent the NAS algorithm from finding novel architectures outside of current network ideas. So the search space must be flexible, but also constrained size-wise in a way that it can still be traversed effectively. These conflicting objectives make search space design complex. Several papers have proposed search spaces for medical image segmentation. These search spaces that can be split up into two categories: discrete and continuous. Discrete search spaces look at every network as a separate entity, and evaluate them one at a time. The most common approach using discrete search spaces in literature is to use a U-Net[4]-like architecture and adapt it and the operations within [3, 49, 55] to find the architecture that performs best on the task at hand. Continuous search spaces also use U-Net-like architectures, but agglomerate all the possible networks in the search space in a so-called Super-network [51]. The separate network performances can then be extracted from this Super-network. These continuous search spaces are disregarded in this thesis as some literature has shown issues with network evaluations when sharing components between networks [52, 54], along with the large computational requirements needed for the training of a Super-network.

In the scope of discrete U-Net-like architecture search, the search space is generally divided into two: a macro-level search space and a micro-level search space. The macro-level search space looks at connections within the DNN, the number of operations, as well as how these operations change the dimensions of the input. The micro-level search space looks at the possible configurations of operations that can achieve the desired dimensional changes from the macro search space. These concepts are further explored in Section 2.5.3 and Chapter 5. In literature, the configuration found in the micro search space is often repeated throughout the network [27, 28]. Instead of repeating this structure, research that allows various structures

[4]U-Net is a segmentation DNN that outperformed many of the SotA networks when it was published in 2015, and from which many DNNs still derive their structure today.

in a DNN, potentially resulting in a better performance, is scarce. Furthermore, existing knowledge about configurations could be drawn from networks with a strong performance on classification tasks. Allowing only configurations from a pre-selected pool (which are taken from advanced well-known classification networks [14, 18, 44]) instead of searching for a cell configuration from scratch every time, could allow for a stronger performance increase when using NAS. Apart from benefiting from advanced performance of these well known classification networks, the use of a pre-selected pool will also prevent explosive growth of the search space caused by searching the configurations from scratch at multiple locations in the network. Finally, recent research [55], does a sequential bi-level search, where macro-level search is followed by micro-level search. Searching through these two levels simultaneously could lead to finding possible interactions between the macro-level and micro-level search. This could potentially yield better performing networks. With these possible improvements in mind, it would be interesting to see whether a new search space could be designed and evaluated. For this, a second research question was drawn up:

Research Question 2. *Can a discrete bi-level search space be created such that the neural networks found by a NAS algorithm are better than hand-crafted SotA networks, as well as outperform networks found using different existing discrete search spaces, for medical image segmentation tasks?*

The details of this search space design and the performance of the new search space are found in Chapter 5.

1.3. Search algorithms

In the search algorithm domain, many techniques have been studied for image classification. The performance of these algorithms has mostly been compared on benchmark problems [36, 42, 53], which are confined to a few datasets and a small number of search spaces. These benchmarks contain a cached list of simple one-run network performances, and do not include noise, (with the exception of [42] that uses surrogates to model network performances instead of evaluating all the networks in the search space), making them quite different from realistic situations. No benchmarks exist for medical image segmentation unfortunately, and it is not clear how the performance of these algorithms translates to this task. It would therefore be interesting to compare multiple algorithms and their performance given a medical image segmentation dataset. In this thesis, two Evolutionary Algorithms - one Simple Genetic Algorithm and one Model-based Evolutionary Algorithm, as well as a Local Search algorithm, are compared. These experiments are done in Chapter 6. The research question for this is:

Research Question 3. *How do Evolutionary Algorithms and Local Search perform for NAS in different levels of noise and can a decision be made on which is better?*

Before diving into these three components of NAS, more information on relevant concepts is given in Chapter 2. The datasets used throughout this thesis are described in Chapter 3. Subsequently, Chapters 4 through 6 provide insights gathered on performance estimation, search spaces, and possible search algorithms for NAS for medical image segmentation. Finally, this thesis is concluded in Chapters 7 and 8 where results are discussed and conclusions drawn.

2 Background

This chapter contains information on all the relevant concepts used in this thesis. It starts by explaining medical image segmentation, and what is necessary to automate this process. This is followed up with the mechanics of Neural Networks, in particular the workings of Deep Convolutional Neural Networks. Next, the Search Algorithms considered in this thesis are explained. At the end of the chapter, the concept of Neural Architecture Search is introduced, alongside relevant research and how this thesis builds upon it.

2.1. Medical Image Segmentation

Medical imaging technologies, such as CT scanners, MRI scanners, X-ray generators, and ultrasound machines, have provided the ability to look inside the human body without invasive surgery. By using different tissue qualities, it is possible to make out different organs or regions of interest, e.g. tumors, from these images. This information can be valuable for multiple reasons, such as disease diagnosis [5, 13] and customised treatment planning [30, 48].

The information from these scans is stored spatially and is quite dense. When a scan is made with different scanning modalities, i.e., different measured values, the stored data contains multiple data entries for every pixel/voxel[1]. In the case of a 3D scan with multiple modalities, this results in 4D data, which is difficult to visualise for human interpretation. This high information density, can make interpreting these scans quite time-consuming.

Automating this process, such that a machine could interpret these scans, could lead to higher efficiency by increasing speed of interpretation. This is especially useful in treatment planning, where time is of the essence [30]. This thesis looks specifically at the field of automated medical image segmentation. This is the practice of having a machine interpret medical images such that contours are provided of an organ or a region of interest.

2.1.1. Automated medical image segmentation

To automate medical image segmentation, a machine needs to be able to process the multi-dimensional input image, and generate a multi-dimensional output mask from it. As the output is highly dependent on multiple input variables, i.e. the surroundings of the pixel in question are important for the class it is assigned to, segmentation methods need to be strongly connected such that many input variables can influence the output.

In recent years, model based approaches, which look at statistical data properties, have been surpassed by supervised machine learning methods, most notably Deep Neural Networks (DNNs) [41]. These tools have been achieving high, in some cases human-like, performance on several image segmentation tasks [34].

The strength of DNNs lies in their flexibility to learn non-linear mappings between a multi-dimensional input and output, by training on examples. Given elaborate data, DNNs can see scans from many different patients in quite a short time frame, and learn from delineations from multiple physicians. This process is known as network training. As the network can learn from scans made by different physicians, it could

[1]A voxel is the 3D equivalent of a pixel, carrying spatial information based on an x, y, z co-ordinate.

potentially also remove inter-physician variability that can occur when multiple professionals look at the same case [7]. Once trained, a DNN can provide a high quality segmentation within a matter of seconds. This capability of learning from examples to provide quick and accurate segmentations, is what makes these DNNs so adept at this task. An example segmentation of the prostate as provided by a DNN can be seen in Figure 2.1.

Although DNNs are already achieving impressive results, performance can still vary greatly per task, and research into how to create the best possible NN for different tasks is still actively ongoing.

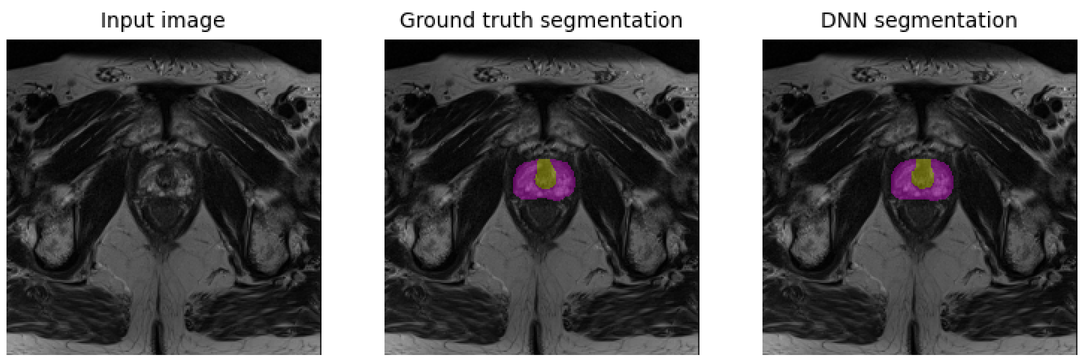


Figure 2.1: *On the left is a single modality of a slice from an MRI of the prostate area; in the middle is a physician's delineation of two prostate areas, the central gland in yellow, and the peripheral area in magenta; on the right an example of a DNN prediction of the two regions of interest. The MRI and ground truth segmentation were provided by the Medical Segmentation Decathlon [1].*

2.2. Neural Networks

As mentioned, DNNs have been achieving SotA results on many public medical image segmentation tasks [1, 26]. These DNNs are specifically created for the task of semantic segmentation, where every pixel/voxel is assigned a class. These DNNs are a subclass of NNs, and as all other NNs, work by stacking layers of artificial neurons.

2.2.1. Artificial Neural Networks

Neural Networks consist of artificial neurons that connect any number of input values to any number of output values. The output value of each neuron in a network is a weighted sum of its inputs along with a bias term. This output is then passed through a non-linear activation function and connected to a number of neurons in the next layer. By stacking layers, the network can generate a non-linear combination of weighted inputs as output. The layers between the input and output are called hidden layers. An example of a simple NN is given in Figure 2.2.

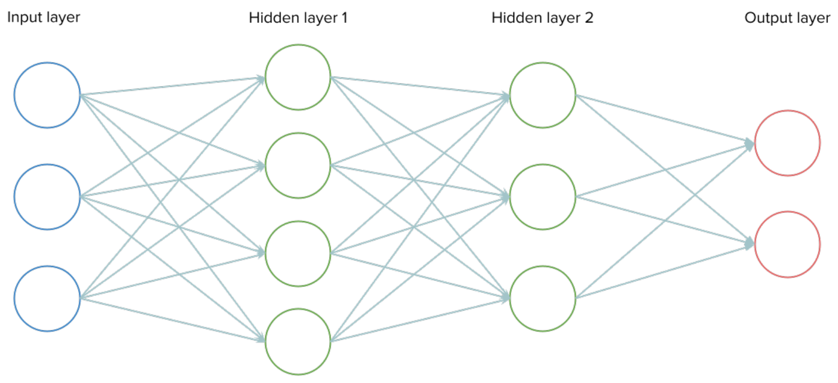


Figure 2.2: *An example of a Neural Network. It shows how any multidimensional input can be transformed to another number of dimensions. Here, two hidden layers connect the input to the output.*

The output of a neuron can be written as:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \phi \left(\sum_{i=1}^{n} \theta_i \cdot x_i \quad + b \right)$$

where x_i is an input and θ_i the assigned weight for $i \in 1, .., n$, ϕ the activation function, and b is the bias term.

To make sure the NN has the correct weights, it needs to be trained. Training can be done by doing a forward pass of an example image, i.e. calculating the output values given an example input, and then calculating the error between the generated output and the desired output using a loss function. This error is then used to re-calibrate the weights, in a process called back-propagation [40]. By doing this repeatedly the network will be optimised towards the task in the examples (more details on this process are given in Section 2.3). A trained NN is then able to generate similar outcomes from input values it has never seen before.

2.2.2. Deep Convolutional Neural Networks

A problem occurs when scaling the NN shown in Figure 2.2. For example, when given an image of resolution 128×128 as input, and using a network with one fully connected (meaning neurons are connected to all neurons in the previous layer) hidden layer with the same resolution, the number of connections, and therefore weights to optimise, is $2 * (128)^4 \approx 537$ million, which is an incredibly large number. Not only is the number of weights to optimise very large, but the network also only has a single hidden layer. This prevents the network from combining different patterns (i.e., combination of weights with a certain correlation to the output) found in previous layers, making it necessary to learn every pattern separately. An example of this, is that a network with single hidden layer would need to optimise weights to learn how to recognise a dog by learning entire dog features, such as ears and paws, while multiple hidden layers can first recognise curves, lines, or presence of fur, and build slowly upon these low level features. It has empirically been shown that learning these features hierarchically leads to better performance [25].

To increase the number of hidden layers, the number of weights per layer in the network needs to be reduced. In computer vision tasks, the most common method to reduce the number of weights to optimise, while still being able to capture information from a wide range of inputs, is to use convolutions (see Section 2.2.4). These operations have the capacity to highlight certain properties in an image, and allow networks to have many more hidden layers due to the low number of weights per layer. Networks using many stacked convolutional layers are known as Deep Convolutional Neural Networks, or deep CNNs.

2.2.3. Structure

The structure of a deep CNN consists of a combination of convolutional layers, that lead from the input layer to the output. For image segmentation, the input is generally a 2D image or a 3D scan, and the output is a mask showing where the region(s) of interest is/are. An example of a deep CNN, that is very similar to the architecture of U-Net, is given in Figure 2.3.

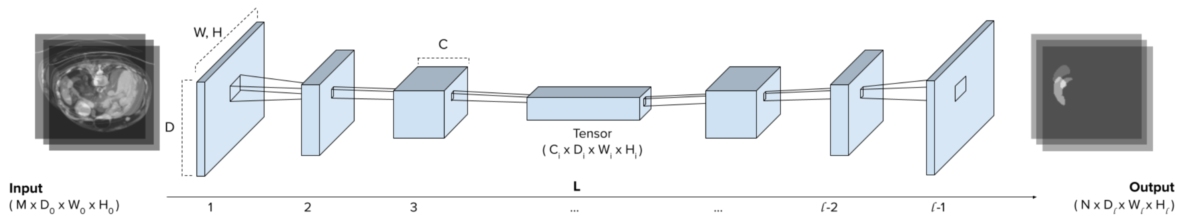


Figure 2.3: An example of a DNN for medical image segmentation. The blue cubes are the feature maps, i.e., the output of the layers in the network, and the connecting lines abstractions of the convolutional layers.

By default, every layer in a CNN is a combination of three operations: convolution, activation, and normalisation. These operations are explained in 2.2.4. By stacking these operations, a CNN can learn to identify various aspects within the image by assigning the proper weights to the connections between its layers. As more layers are applied, more elaborate features can be learned. This can be seen in Figure 2.4.

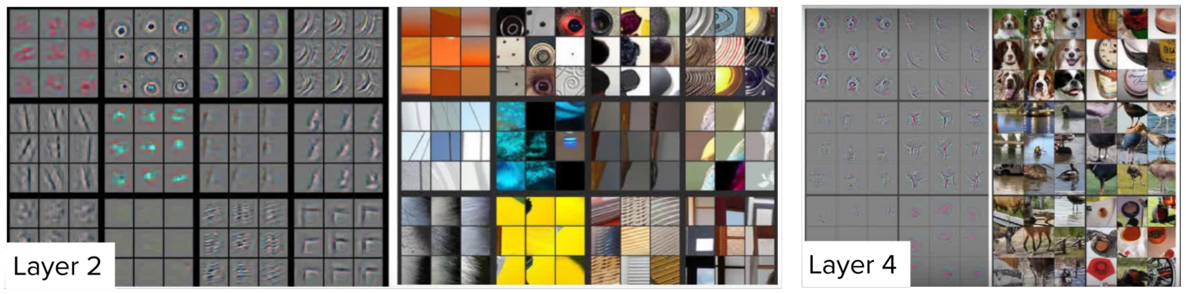


Figure 2.4: Examples of how convolutional layers extract features from images. The features that can be identified become more elaborate as more layers are added. A comparison can be seen between the features learned after 2 layers and after 4. These images are adaptations from Zeiler and Fergus' paper "Visualizing and Understanding Convolutional Networks" [56].

The total operation performed by a CNN for medical image segmentation can be represented as a mapping of $I(MxDxWxH) \rightarrow O(NxDxWxH)$, where I and O represent input image and output segmentation, respectively. The mathematical structure used to store the information in the network is called a tensor. Here the input tensor has four dimensions consisting of M, the number of modalities (e.g., 1 if greyscale, 3 if RGB[2], but can have any number of different scan parameters), D, the depth (1 if a single image, multiple if a 3D scan), W and H, the width and height of the image. Where the input tensor contains different modalities, the output tensor contains information on what class every pixel/voxel belongs to. So dimension N is the number of classes. At every convolutional layer $i \in 0, ..., l$, the tensor shape can be written as $\langle C_i x D_i x W_i x H_i \rangle$, where C_i is the number of channels in the feature map. Channels are the neuron outputs per pixel/voxel in a hidden layer.

2.2.4. Convolutional layers

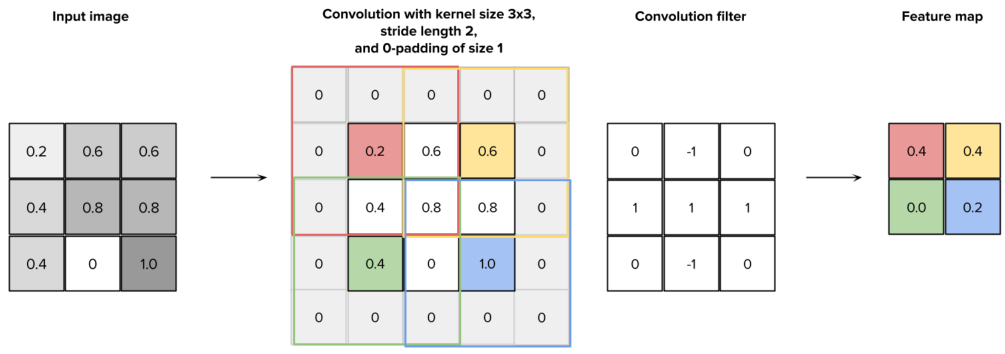
To get from input to output, image properties need to be learned for the region of interest. A very common way to do so, is by performing convolutions. A convolution is an operation that sums the input values of a grid around a pixel, known as a receptive field, multiplied by the weights of a convolutional kernel. This is different to the fully connected layers from the network shown in 2.2, as connections are only made between neighbouring neurons, instead of the entire input. Weights for a convolutional kernel are also shared throughout the image, drastically reducing the number of weights to optimise, allowing for deeper networks. The output values of a convolution are then stored within a feature map, which is the name given to the output tensor of a layer. By using the same kernel at every location in an image, patterns can be identified, e.g. vertical lines or other shapes (see Figure 2.4), throughout an image. The mathematical notation for this operation, when performed on a single channel (e.g. gray-scale image), is the following: for every position i, j where $i \in [0, W), j \in [0, H)$ given an image **X** and convolution kernel **K** with size k:

$$f_{i,j}(\mathbf{K},\mathbf{X}) = \sum_{n=0}^{k-1}\sum_{p=0}^{k-1}\mathbf{X}_{i-\lfloor k/2\rfloor+n,j-\lfloor k/2\rfloor+p}\mathbf{K}_{n,p}$$

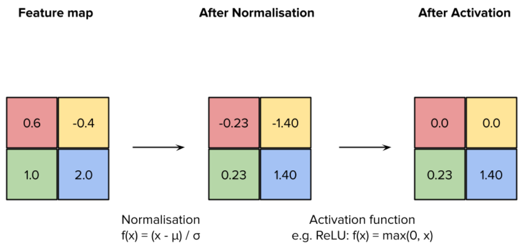
Depending on the convolutional kernel, the output image may be of a different dimension as the boundaries of the image prevent the kernel from performing convolution for positions at the edge. For these boundary locations, padding can be used. This padding serves as dummy values around the image such that convolution can still take place and information at the borders is not lost. Different padding methods exist, such as 0-padding, mirrored values, or replication of boundary values. An example of a convolution with a 3x3 kernel size, and 0-padding of size 1, is given in Figure 2.5a.

If the input has multiple channels, e.g. an RGB encoded image, convolution is performed with multiple kernels, one per input channel. A convolutional filter, is the name given to the collection of kernels[3]. The output of a convolution operation is the element-wise sum of the outcome of the convolution per channel. For every desired output channel, a different filter is used. The mathematical notation for a single channel output f is: for every position i, j where $i \in [0, W), j \in [0, H)$, where W, H are the width, height respectively, given an image **X** and convolution filter **H** with M kernels of size k.

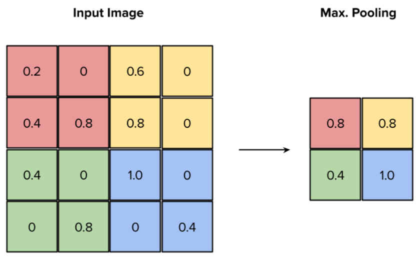
[2]Red, Green, Blue, oftentimes the encoding used for coloured images
[3]In the case of 1 channel, a filter and a kernel are identical.


Input image
Convolution with kernel size 3x3, stride length 2, and 0-padding of size 1
Convolution filter
Feature map

(a) Single-channel convolution with kernel size 3x3, stride length 2, and 0-padding of size 1 applied to the input image.



(b) Activation and Normalisation.



(c) Max Pooling.

Figure 2.5: *Different operations in a a convolutional cell. Figure 2.5a shows an example of a padded convolution with kernel size 3 and with a stride length of 2. The output feature map and the area of input that was part of the convolution is colour-coded. In Figure 2.5b an example of activation by ReLU is given, followed by instance normalisation. In Figure 2.5c an example of max-pooling is given.*

$$f_{i,j}(\mathbf{H},\mathbf{X}) = \sum_{m=1}^{M}\sum_{n=0}^{k-1}\sum_{p=0}^{k-1}\mathbf{X}_{i-\lfloor k/2\rfloor+n,j-\lfloor k/2\rfloor+p}\mathbf{H}_{m,n,p}$$

Increasing the number of convolutional filters that are applied to the input image, increases the number of channels in the feature map. This increases the number of weights to optimise, but also allows the convolutional layer to learn more features.

Changing the resolution of the feature maps can be interesting in order to learn more elaborate features that span more of the input image. The reason for this, is that the receptive field of future operations increases due to denser information. An example: a 3×3 convolution on an image of 128×128 pixels will encapsulate a smaller distance of surrounding data than a 3×3 convolution on the same image but with a resolution of 64×64. So decreasing the resolution can increase the receptive field. To change the resolution of the image, two methods are common: convolutional stride, or pooling layers.

Convolutional stride is the distance between positions that are evaluated in a convolution. For example, if the stride is 1, convolutional kernel size is 3x3, and there is padding of thickness 1, every position will be evaluated within the image and therefore the width W and height H of the feature map will not change. If in the same environment, we change the stride to 2 and remove the padding, one position will be skipped between convolutions, this leads to a new width and height, $W' = \frac{W}{2}$, $H' = \frac{H}{2}$. These smaller image dimensions lead to more compact information, meaning that future convolutional operations are connected to a larger portion of the input. An example of stride is given in Figure 2.5a.

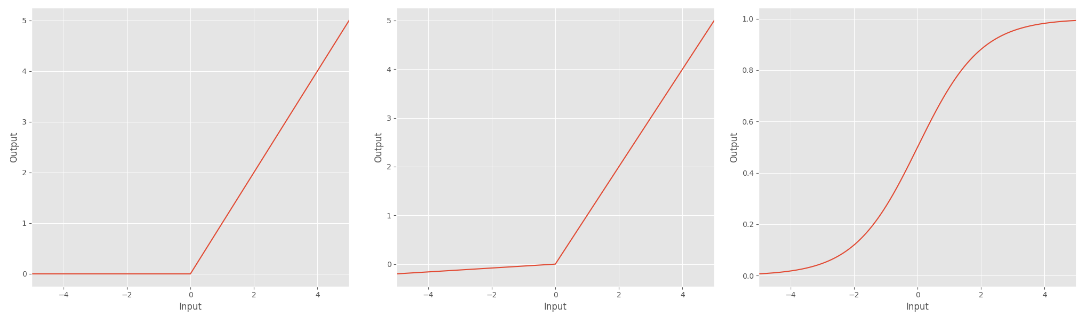
2.2.5. Pooling layers

Another way to decrease the resolution, is pooling. Pooling operations are fixed and therefore do not increase learnable weights in the network. These pooling operations apply a function to a window of values, trying to extract the most valuable information from the window. Two examples of pooling functions are max pooling and average pooling. The former outputs the maximum value of the window, while the latter outputs an average. An example of max-pooling is given in Figure 2.5c.

2.2.6. Activation and normalisation

Two operations are often added to both convolutional and pooling layers: activation functions and normalisation. These two operations generally succeed convolution and pooling operations.

Activation functions, can achieve multiple desirable traits in the output. Most commonly, non-linear activation functions are used to introduce non-linearity to an otherwise linear system of convolution operations. The most common functions are ReLU, sigmoid and LeakyReLU. In this thesis, ReLU and LeakyReLU are used to introduce non-linearity. The ReLU or Rectified Linear activation Unit, is simple to calculate as it is equal to the function $f = max\{0, x\}$. It also introduces non-linearity into the output, while the derivative is still easy to calculate. Also, the gradient does not vanish when x approaches 0. For this reason, it is the most common activation function in DNNs. An example of ReLU activation is given in Figure 2.5b. LeakyReLU is a variant of ReLU where, a small negative slope is added, to avoid the ReLU form dying, i.e. never activating as all input values are in the negative domain. The function for LeakyReLU is $f = max\{0, x\} + \alpha \cdot min\{0, x\}$, where α is a negative slope coefficient. Sigmoid functions are generally only used on the final layer. This function converts all values to a $[0, 1]$ domain. It is often used to change outputs to probability values to see which class the output belongs to. When multiple classes are involved, a softmax function is used, an extension of the sigmoid function that scales the probabilities such that they sum up to 1. Examples of these activation functions are given in Figure 2.6.



(a) ReLU. **(b)** LeakyReLU. **(c)** Sigmoid.

Figure 2.6: *Activation functions.*

Normalisation is used to stabilise these layers. The goal of normalisation is to make sure the data is 0 centered and and to down-scale large values. Centering the values around 0 makes sure that the activation function can be effectively used, as using activation functions in a domain away from 0 will just lead to multiplication with a constant value. Scaling values increases the speed of convergence of the weights in the network [19]. An efficient way to normalise the data is by using Z-normalisation. This makes sure the output has a mean at 0 and a variance of 1, thus achieving both goals. Normalisation can be performed on multiple levels, the most common ones in DNNs being instance normalisation, and batch normalisation. The former will take the mean and variance per image and per channel, while the latter will fix the mean and variance per channel within a layer, by averaging over a so-called mini-batch, which is a subset of the data used for a training iteration (see 2.3. This has been shown to speed up the training process, and give extra stability [19]. An example of Z-normalisation of instance-level is given in Figure 2.5. The mathematical notation of instance normalisation, where n is the image, c is the channel, μ is the mean, σ is the standard deviation and ϵ is a small constant added for numerical stability, is:

$$\hat{x} = \frac{x - \mu_{n,c}}{\sqrt{\sigma^2_{n,c} + \epsilon}} \qquad \mu_{n,c} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{n,c,j,k} \qquad \sigma^2_{n,c} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W}(x_{n,c,j,k} - \mu_{n,c})^2$$

For batch normalisation, the equation is similar but adds N, which is the amount of images in the mini-batch:

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma^2_c + \epsilon}} \qquad \mu_c = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{i,c,j,k} \qquad \sigma^2_c = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{k=1}^{W}(x_{i,c,j,k} - \mu_c)^2$$

Batch normalisation then also continues by allowing a scale and a shift parameter γ and β to be learned, which introduces a possibility to shift the mean away from 0 and increase or decrease the variance to a custom level. The rhetoric for this is that it would accelerate training [19]. The output then becomes:

$$\hat{x'} = \gamma\hat{x} + \beta$$

2.2.7. Architecture

When breaking down neural architectures, i.e., the building plans of a NN, the term cell is often used. This is a standardised group of operations, often consisting of one or multiple convolutional layers, that can be repeated. The configuration of a cell can be varied in multiple ways, e.g. what convolution filter sizes are used, whether pooling is applied, what activation function is used, whether residual connections[4] are used. In Figure 2.7, an example of a cell from U-Net [5] is given. This block is derived from the original VGG network [43].

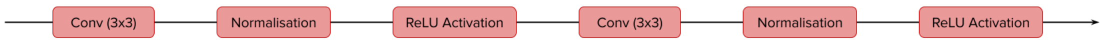


Figure 2.7: *A VGG cell, that is also used in U-Net. It performs two convolution operations followed by normalisation and activation. It does not contain pooling or residual connections.*

On a larger scale, the structure of how these cells connect, and what the dimensions of the feature maps are, is called the topology of the network. Networks can become wider, meaning more channels /neurons are added in a layer, or deeper, meaning the number of operations/layers is increased. The resolution discussed in Section 2.2.4 can also be changed not only for the input, but also for the feature maps. Also, the addition of skip-connections, or concatenating multiple previous outputs to form the input of the following cell can change the topology of a network.

2.2.8. U-Net

In the context of neural architectures for image segmentation, one architecture has been particularly influential: U-Net [37], and its encoder-decoder structure. U-Net achieved a significantly better performance in comparison to other segmentation networks when it appeared on the scene in 2015, and has been a solid baseline for all segmentation tasks since. Many upgrades have been suggested to this structure, such as the ones suggested by the nnUNet-team [20], who try to tune the network and the training processes based on the dataset, but these keep the same topology patterns. It is a good starting point for architectures when looking at medical image segmentation, and a great baseline for experiments.

What typifies U-Net is the encoder-decoder structure, where the input passes multiple convolutions and pooling operations such that the resolution is decreased and the number of channels increased, until a certain bottleneck. In this process, the network will learn more and more complex features from the input, a process visualised in Figure 2.4. The down-sampling of the image is done by using pooling layers, but can also be achieved with convolutional stride. After the bottleneck, the encoded features need to be localised in the original input, meaning that the features that were learned in the encoding, and are part of the eventual classes, need to be located such that they can be segmented. This is done by up-sampling with transposed convolutions in a reversed process to the encoding part. Every decoding cell is not only connected to its predecessor through a transposed convolution, but also to the output of the encoder cell where the feature map has the same dimensions and features are less complex. This combines the information from more elaborate features with simpler ones in order to segment the region of interest in question. In Figure 2.8 the architecture of U-Net is shown.

2.3. Network training

The previous sections show that the structure of a DNN can be elaborate, containing many layers and weights. All these weights need to be calibrated to achieve high accuracy segmentations. To be able to get these parameters to extract the information from the input, it is important that the network gets to see

[4]A residual connection is a type of bypass operation where the output is summed with the input such that every convolution changes the initial value without getting rid of it.

[5]U-Net is the architecture that is described in Section 2.2.8 and visualised in Figure 2.8.

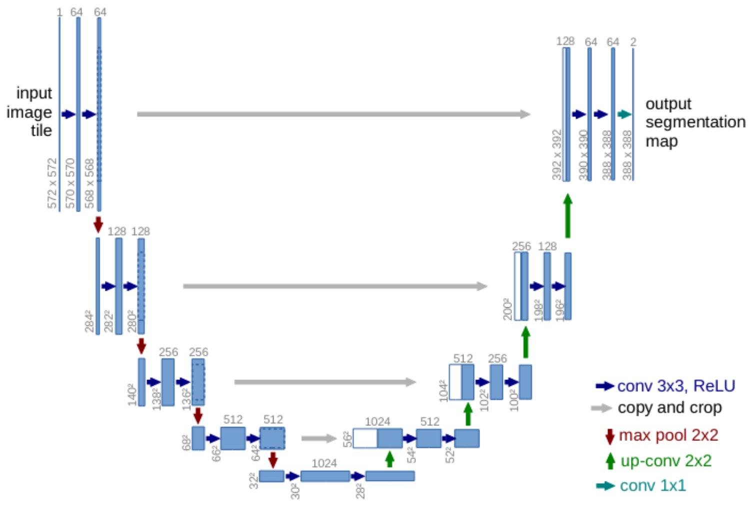


Figure 2.8: *The example given in Ronneberger et al. [37] paper on U-Net. It shows the underlying mechanism of U-Net to first gradually encode to a very small resolution with a gradually increasing number of channels, followed by the reverse process that is concatenated with the output of the encoding size.*

many different training samples, and gets enough training iterations. This can be troublesome, as gathering or generating data can be costly, but is also the strength of a NN, as it can see relatively many different cases in a short amount of time.

2.3.1. Data preparation and augmentation

In order to measure the performance of a NN, the dataset that is used is generally split into three groups, the training set, the validation set, and the test set. The training set consists of the images that are shown to the network and which the network learns from, the validation set is used to check if the network also performs well on cases it has not seen. For comparison with other methods, a separate test set is generally excluded from the rest of the data to analyse the final performance. However, in small medical datasets, this is often infeasible. The use of a validation set and test set allows for unbiased performance indication. For completeness, k-fold cross-validation can be performed. This means the experiment is repeated k times with k different splits such that every scan is in the validation set once. This process is visualised in Figure 2.9. When datasets are relatively small, i.e. different divisions of the dataset lead to very different results, it could be beneficial to add data augmentations to the training set. This tries to prevent potential over-fitting[6], by increasing the amount of unique training images. Data augmentations also have the potential to increase the performance of the network on scans that are different from the ones in the training set[20]. Many different ways of editing input images, such as translation, rotation, resizing, mirroring, brightness adjustment, contrast adjustment, etc. can be applied stochastically in order to increase the number of different training images available. When applied correctly, data augmentations will lead to a more robust network. However, they can also lead to the loss of valuable information when training cases become too unrealistic, deteriorating performance. More data augmentation schemes will also increase training time, so consideration is necessary when it comes to choosing what data augmentation methods to use.

2.3.2. Training procedure

The training procedure works by performing a large number of training iterations, until the network performance on the validation data stops improving. One iteration consists of one forward pass and one backward pass. The forward pass evaluates a certain amount of samples, also known as a batch of training images. The input images pass through the NN, returning a prediction. This is then compared with the reference masks, or ground truth. By comparing the predicted values to the ground truth, a loss can be calculated (see 2.3.4). The aim is then to minimise this loss. This is done with the backward pass, where an optimiser will try to follow the loss gradient with a Stochastic Gradient Descent (SGD) algorithm (see 2.3.5) and then

[6]Over-fitting is a situation where a model becomes more and more accurate on the training set, but where the accuracy on the validation set starts to decrease

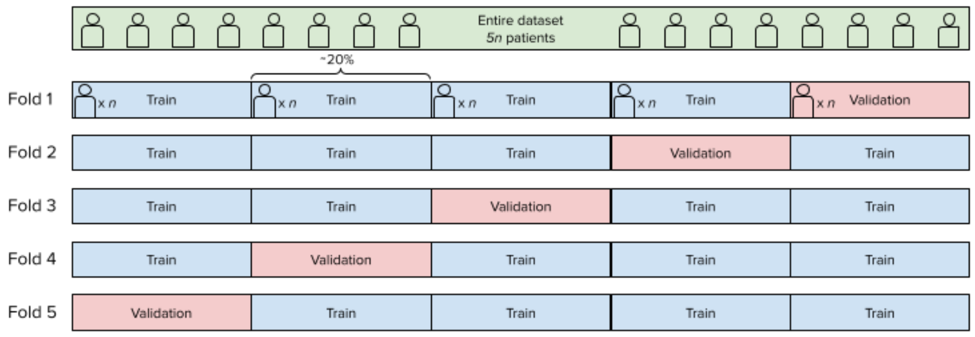


Figure 2.9: *An example of cross-validation. Here the data is split into 5 folds. In medical image segmentation, the split is done per scan/patient.*

adjust the weights in an attempt to decrease the loss given that batch. One epoch is generally the term used for one lap of all the training samples. After each epoch, the validation loss is calculated. If this validation loss no longer decreases, the training can stop. An example of a DNN training graph is given in Figure 2.10.

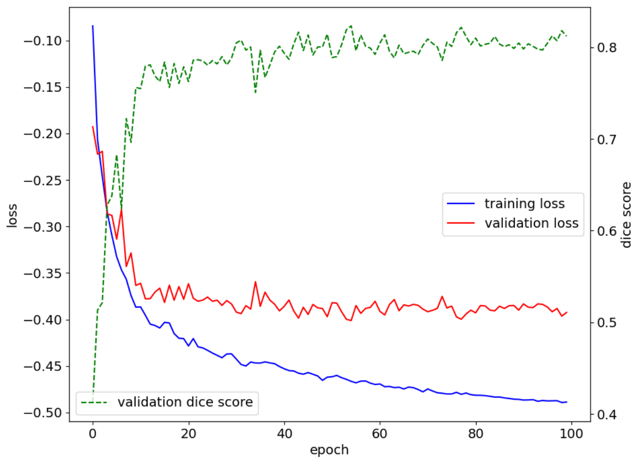


Figure 2.10: *A progress graph of a DNN training for a medical image segmentation task. In blue, the training Soft Dice loss per epoch, in red, the validation Soft Dice loss per epoch, and in green, the validation DSC per epoch.*

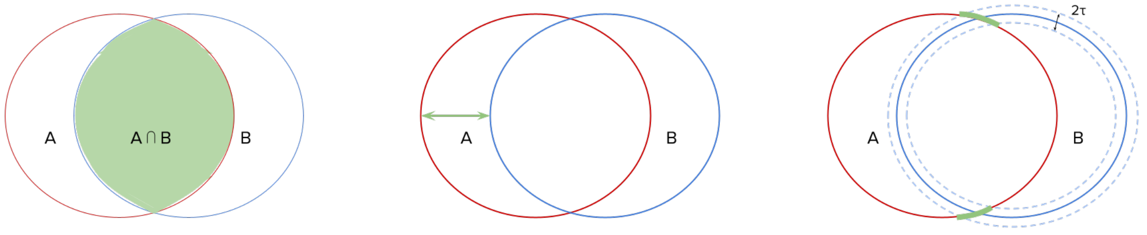
2.3.3. Network performance metrics

Metrics are needed to evaluate the performance of a DNN such that they can be trained. For medical image segmentation, this means a metric is needed to score the similarity between provided segmentations and the the segmentations output by a DNN. To quantify segmentation similarity, many metrics exist, but the three used in this thesis are: Dice-Sorensen Coefficient, Hausdorff distance, and Surface Dice. These are calculated for every image in the validation set and then averaged.

The Dice-Sorensen similarity Coefficient (DSC) [9] is a metric that uses overlap between prediction and reference mask. To calculate it, the true positives TP, false positives FP, and false negatives FN are computed per pixel/voxel. The DSC is computed using:

$$DSC = \frac{2|A \cap B|}{|A| + |B|} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Which when looking at a predicted mask and the reference mask, is equivalent to 2 times the surface of the overlapping area, over the sum of the segmented surface area and the reference surface area. The resulting domain is $[0,1]$ where 0 means the segmentations have no overlap, and 1 means identical segmentations. The Dice-Sorensen similarity coefficient is visualised in Figure 2.11a.



(a) Dice-Sorensen coefficient. The area $|A \cap B|$ is doubled and divided by $|A| + |B|$.

(b) Hausdorff distance.

(c) Surface Dice.

Figure 2.11: *An abstraction of a segmentation and a reference mask. Here A is the segmentation and B is the reference. The Dice-Sorensen similarity coefficient is calculated by taking two times the overlapping surface, seen in 2.11a, divided by the surface of the segmentation summed with the surface of the reference. The Hausdorff distance is the largest minimal distance, or supremum of the two sets, as seen in 2.11b. The Surface Dice similarity coefficient, does not look at the volume like DSC but at the surface area for which the segmentation and reference are within a certain tolerance τ, as seen in 2.11c.*

The Hausdorff distance (HD) [6] is a metric that calculates the maximum distance between one point in the two surfaces and the nearest point on the other surface, or in mathematical notation:

$$HD(A,B) = \max\left\{\sup_{a\in A} d(a,B), \sup_{b\in B} d(A,b)\right\}$$

The Hausdorff distance is visualised in Figure 2.11b. In order to make Hausdorff more robust, the 95^{th} percentile distance can be used instead of the maximum distance [21]. This removes the impact of exceptional outliers.

The Surface Dice similarity coefficient (SD) [34] measures the part of the surface of the segmented area where the distance between the segmented and reference surface is within a given tolerance, compared to the total surface of the segmentation. Mathematically this can be written as:

$$SD(A,B) = S_A - S_{A'}, \qquad S_{A'} = \left\{ S_A \middle| \sup_{a\in A} d(a,B) > \tau \right\}$$

where S is the surface, sup is the supremum, or maximum smallest distance, and τ is the tolerated distance. The Surface Dice similarity coefficient is visualised in Figure 2.11.

2.3.4. Loss functions

The metrics discussed above are useful to assess the similarity of output segmentations, which contain the assigned classes per pixel/voxel based on a probability. However, they are not very suitable as loss functions, because they do not distinguish how far the output is from the assigned class; the error per pixel/voxel is either 1, or 0.

For this reason, specific loss functions are needed. A commonly used loss function for medical image segmentation is the Soft Dice loss. Soft Dice loss is based on the Dice-Sorensen similarity coefficient, but has some fundamental differences. First, it uses the probabilities that are output by the network before they are made binary (e.g. A network outputs a probability of 0.6 that a pixel belongs to a class in a single class segmentation. This probability is used by the loss function to calculate the error. This is different to using the value in the output image, where 0.6 will be assigned to the class in question and rounded to 1). As these probabilities are not binary, true positives can not be calculated. Instead, the summed product of the output and the reference for every pixel/voxel is taken. Also a smoothing coefficient ϵ is added in both the numerator and the denominator such that if no surface needs to be segmented and there is no segmentation, the score is 1.

$$\mathbf{L} = -\frac{2\sum_{i=0}^{N} a_i b_i + \epsilon}{\sum_{i=0}^{N} a_i + \sum_{i=0}^{N} b_i + \epsilon}$$

where a_i and b_i are the pixel/voxel from the output and reference, respectively. N is the number of voxels.

In contrary to the Dice-Sorensen coefficient, which should be maximised to obtain the best segmentation, the Soft Dice loss works with negative values. This is then minimised by the optimiser, as Gradient Descent optimisers always work with minimisation.

2.3.5. Optimisers

Optimisers are a type of algorithm that modify the weights within a network during backpropagation. By performing some variation of gradient descent, they try to find weight values for which the loss is minimal. Many optimisers have been designed for this purpose, two of which are briefly explained here. The first one is Stochastic Gradient Descent (SGD). SGD updates the weights between layers iteratively by estimating the gradient of the loss based on a random subselection of the data. With this gradient, the weights are updated by using the following function:

$$\theta' = \theta - \alpha \cdot \nabla_\theta \mathbf{L}(x^{(i)}; y^{(i)}; \theta)$$

Here θ represents the weights, α is the learning rate, **L** is the loss, and i is the subset of samples to calculate the gradient over. The weights are updated by using the chain rule [40], computing the loss gradient in respect to the weight that is being updated. This is gradient in respect to each weight is written as $\nabla_\theta \mathbf{L}(x^{(i)}; y^{(i)}; \theta)$. SGD is the basis of other optimisers, as it can be extended with features such as learning rate decay, momentum and Nesterov, which increase the speed of the gradient descent.

ADAM, or Adaptive Moment Estimation [23], is a special type of Gradient Descent that uses a mechanism called momentum to smooth the noise in the direction it is moving in, which it does by calculating the exponentially weighted moving average of the average gradients, and smooths out the variance using the same principle but on the quadratic gradients. This second mechanism is known as RMSProp. With these two added features, ADAM has been shown to outperform SGD on numerous problems [23]. ADAM is used as optimiser of choice in this thesis.

2.4. Search algorithms

Search algorithms in the context of Neural Architecture Search, are methods developed to find the best possible network within a given solution space. No prior knowledge of network performance is given to these algorithms. It is their task to learn what performs well and what does not. In this thesis, two Evolutionary algorithms, namely a Simple Genetic Algorithm, and a Model-Based Genetic Algorithm, as well as Local Search are used. Before diving into the mechanisms of these algorithms, a description of is given on how networks within NAS can be represented as encodings. Also an explanation is given on why Evolutionary Algorithms were chosen for this problem.

2.4.1. Encodings

All individuals within a search space, are encoded in a genotype. This is a string of variables of length ℓ that can be continuous or discrete. Every variable x_i where $i \in [0, \ell - 1]$ has a given domain. This domain is the set of values the variable can take on, e.g. $\{0, 1, 2, 3\}$ or $\{C, G, T, A\}$. Every encoding can be evaluated with a fitness function, and this fitness can be used by the algorithm to rank the individuals. Search algorithms differ in how they rank individuals and what they do with this ranking to proceed.

In this thesis, an individual represents a DNN. In the encoding, each variable represents a different design choice. The fitness function is the chosen performance metric of an encoded DNN on a certain task.

2.4.2. Evolutionary Algorithms

Evolutionary Algorithms (EAs) are population-based optimisation algorithms, inspired by the principles of Charles Darwin's *Survival of the Fittest*. These algorithms use an evolving population of individuals to progress through a search space. By comparing the fitness of individuals, the individuals with the highest fitness (elites) survive. This process is called selection. The elite individuals are then used to generate new individuals (offspring) through variation. The generated offspring is merged with the elites to form the new population. One such cycle is called a generation.

EAs are population-based, and are more robust than point-based algorithms like Local Search in fitness landscapes with many local optima [35], as they tend to explore more of the search space. They also do not need any assumptions to work; all they require is a fitness function, mapping from individual to fitness. Their downside however, is that they tend to need more evaluations to find the optimum in simpler convex search spaces, compared to point-based approaches. Recent model-based EAs have shown to be competitive in discrete optimisation problems [47], and improve on classical EAs.

NAS for medical image segmentation is a discrete stochastic black-box[7] problem, with a fitness landscape dependent on many variables, such as task, architecture, optimiser, and initialisation. As assumptions are very difficult to make, Evolutionary Algorithms seem to fit the problem, and have already been used in previous NAS for image classification and image segmentation research [4, 12, 36]. However, Random Search and Local Search have been shown to perform similarly in comparison to these algorithms [36, 54] for other image processing tasks. It is interesting to see whether this also holds for medical image segmentation. To compare the different algorithms, a classical population-based EA, a model-based EA, and a point-based Local Search algorithm, were chosen to compare: SGA, P3GOMEA, and LS, respectively.

2.4.3. Simple Genetic Algorithm

The Simple Genetic Algorithm [16] (SGA) is a classical form of evolutionary algorithms. It uses mechanisms close to natural evolution to perform selection and variation. A visualisation of the algorithm is given in Figure 2.13, and pseudocode in Algorithm 1.

Selection is most often done with tournaments, where the individuals are matched up in groups and the ones with the highest fitness survive. Smaller tournament sizes, i.e., the groups in which individuals are compared, have a lower selection pressure. This means that there is a larger probability of survival for weaker individuals.

Variation, or the generation of offspring, is accomplished with crossover and mutation. Crossover is the process of generating a new individual with the genome of two parents. Mutation is the stochastic mechanism that changes a variable based on a chosen mutation probability. In Figure 2.12, examples of variation methods and mutation are given.

SGA does require parameter choices. Population size, tournament size, crossover mechanism, and mutation rate, all need to be chosen and affect performance of the algorithm. To chose the optimal values for these parameters, they would need to be tuned. However, this process can be very computationally expensive and therefore can quickly become infeasible for NAS.

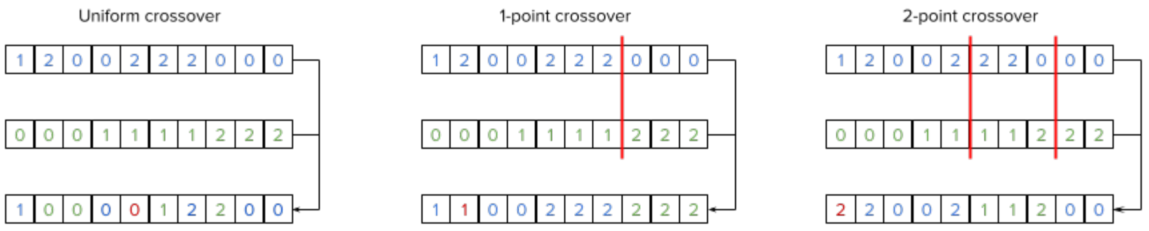


Figure 2.12: *Three different crossover methods, uniform, 1-point, and 2-point. Uniform crossover takes either the gene from parent 1 or parent two with probability $p = 0.5$. 1-point crossover decides on one position to cut the encoding, taking the front from one parent and the back from the other. 2-point crossover works similarly, but with 2 cut points, inserting a piece of the genome of one parent in the genome of the other. Mutation also takes place, shown with the red values. Mutation means a variable is switched with a certain probability, called the mutation rate.*

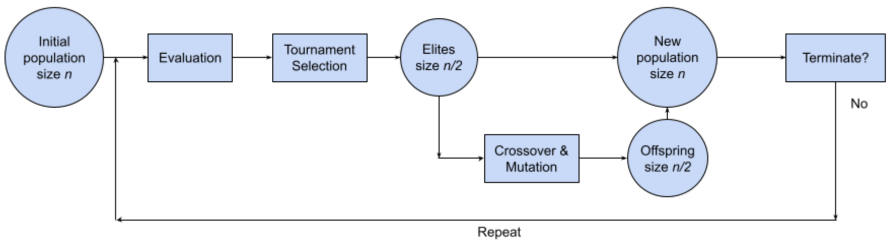


Figure 2.13: *The SGA algortihm visualised in a flow chart. After a random population is initialised and evaluated, selection takes place. The elites are then used to generate offspring. The elites and offspring then form the new population. This is repeated until a termination criterion is satisfied.*

[7]A black-box optimisation problem is often interpreted as a problem for which no gradient of the fitness landscape is available and the shape is unknown.

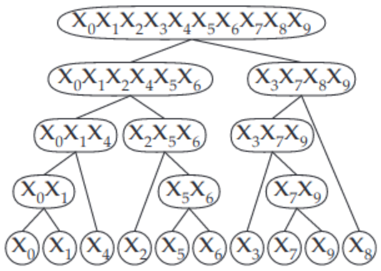
Algorithm 1 SGA algorithm

```
1: function SGA($fitnessFunction, crossoverType, mutationRate$)
2:     $population \leftarrow$ INITIALISERANDOMPOPULATION()
3:     while $\neg terminationCriterionSatisfied$ do
4:         $fitness \leftarrow$ EVALUATEPOPULATION($population, fitnessFunction$)
5:         $elites \leftarrow$ TOURNAMENTSELECTION($population, fitness$)
6:         $offspring \leftarrow$ GENERATEOFFSPRING($elites, crossoverType, mutationRate$)
7:         $population \leftarrow elites \cup offspring$
8:     return $elites, fitness$
```

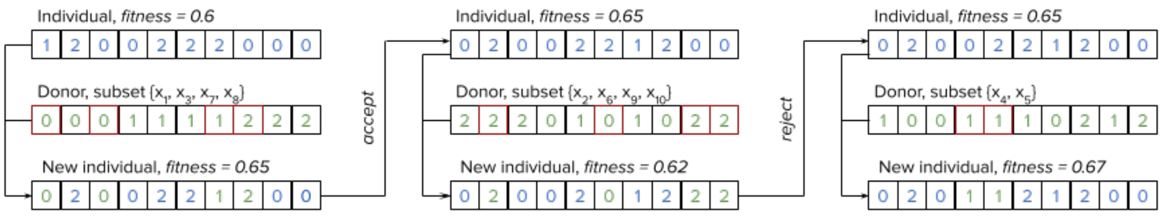
2.4.4. P3GOMEA

Parameterless Population Pyramid (P3) Gene-pool Optimal Mixing Algorithm (GOMEA) is a modern model-based Evolutionary Algorithm [11]. It does not only rely on stochastic mechanisms to do selection and variation, but also tries to learn which dependencies between variables could be exploited. These can then be propagated in new individuals, potentially increasing speed of improvement. The algorithm finds dependencies by learning a Linkage Model. This is defined as a structured set of subsets consisting of variables from the encoding. This structure is known as a Family of Subsets. The learned dependencies between variables guide the algorithm in making educated decisions and making variation more efficient.

The linkage model used in P3GOMEA is a Linkage Tree (LT). A LT is defined as a type of linkage model that is hierarchically structured. The dependencies in it are learned during the optimisation process by first estimating pairwise dependencies from the population and gradually building higher-order dependencies on top of these. Structure-wise, a LT is a binary tree with $2\ell - 1$ vertices where the leaves are singleton variables from the encoding, and the root is the set of all variables. All other vertices are subsets of these variables, and consist of unions of its children, which in turn are disjoint subsets. Subsets of variables closer to the leaves have stronger dependencies than the ones near the root. An example of a LT can be seen in Figure 2.14a.



(a) Linkage Tree.


Individual, fitness = 0.6
1 2 0 0 2 2 2 0 0 0
Donor, subset {x1, x3, x7, x8}
0 0 0 1 1 1 1 2 2 2
New individual, fitness = 0.65
0 2 0 0 2 2 1 2 0 0
accept
Individual, fitness = 0.65
0 2 0 0 2 2 1 2 0 0
Donor, subset {x2, x6, x9, x10}
2 2 2 0 1 0 1 0 2 2
New individual, fitness = 0.62
0 2 0 0 2 0 1 2 2 2
reject
Individual, fitness = 0.65
0 2 0 0 2 2 1 2 0 0
Donor, subset {x4, x5}
1 0 0 1 1 1 0 2 1 2
New individual, fitness = 0.67
0 2 0 1 1 2 1 2 0 0

(b) Gene-pool Optimal Mixing.

Figure 2.14: *A Linkage Tree is shown in 2.14a. It shows the leaves at the bottom, which all consist of singleton variables. Variables with high shared entropy are paired and the vertices grow as you get closer to the root. The root consists of all variables in the encoding. 2.14b shows three iterations of Gene-pool Optimal Mixing on an individual.*

Normalised Mutual Information (NMI) is used as a metric to create pairwise dependencies in the linkage tree. It looks at the Shannon entropy of both variable subsets, to see if knowing one subset would allow you to predict the other. A high NMI value between two subsets, will make the algorithm create a dependency by connecting these subsets in the LT, such that a union is the direct parent of the subsets. The equation for

NMI is the following:

$$NMI(X,Y) = \frac{H(X)+H(Y)-H(X\cup Y)}{H(X\cup Y)}$$
$$H(X) = \sum_{x\in\Omega_X} -P(X=x)\log(P(X=x))$$

where $H(X)$ is Shannon's entropy.

Variation is done by Gene-pool Optimal Mixing (GOM). Instead of doing crossover as explained for the SGA, GOM does multiple crossover operations per individual, accepting improvements from another individual, or donor. The linkage tree is used to decide in what order subsets of variables are donated. For every given subset a donor is selected, and the variables of the selected subset are tentatively donated. This is done iteratively until an improved genotype is found. When this happens, the new genotype replaces the old one. Forced Improvement makes sure that if no subsets from the chosen donors improved the individual, this process is repeated with the best-known individual as the donor. Due to this structure, individuals keep improving due to donated genes. An example of GOM is given in Figure 2.14b.

P3GOMEA is a parameterless algorithm, so e.g. population size does not have to be decided like for SGA. This is beneficial as population size tuning is not a viable solution for NAS due to computational requirements. P3GOMEA has been shown to be a competitive baseline for many benchmark problems [11]. In terms of number of function evaluations required to find an optimum, it is more efficient than other Model-Based EAs. Algorithm 2 shows the pseudocode of the algorithm in this thesis.

Algorithm 2 P3GOMEA algorithm [11]

```
1:  function EA(fitnessFunction)
2:      $iter \leftarrow 0$
3:      $Pyramid \leftarrow \{\emptyset\}$
4:      while $\neg terminationCriterionSatisfied$ do
5:          $elitist \leftarrow None$
6:          $p \leftarrow generateRandomSolution()$
7:          $Pyramid^0 \leftarrow Pyramid^0 \cup \{p\}$
8:          $solutionsAdded \leftarrow True$
9:          $currentTopLevel \leftarrow |Pyramid| - 1$
10:         $\mathcal{L} \leftarrow 0$
11:         $elitistBefore \leftarrow elitist$
12:         while $\mathcal{L} \leq currentTopLevel\ \&\ solutionsAdded$ do
13:             $\mathcal{F} \leftarrow learnLinkageModel(Pyramid^{\mathcal{L}})$
14:             $o \leftarrow GOM(p, Pyramid^{\mathcal{L}}, \mathcal{F})$
15:             if $compareSolutions(o,p)$ then
16:                 if $\mathcal{L} = currentTopLevel$ then
17:                     $Pyramid^{\mathcal{L}+1}.append(\emptyset)$
18:                 $Pyramid^{\mathcal{L}+1} \leftarrow Pyramid^{\mathcal{L}+1} \cup o$
19:                 $solutionsAdded \leftarrow True$
20:             $p \leftarrow o$
21:             $\mathcal{L} \leftarrow \mathcal{L} + 1$
22:         $elitistAfter \leftarrow elitist$
23:     return $elitist, fitness$
```

2.4.5. Local Search

Local Search (LS) is a very simple search algorithm that does not keep track of anything other than the best-found individual, or elite. It works by initialising a given or random network, and then iterating over every value of a certain variable to check whether the network improves. If it does, the new network becomes the elite. The algorithm will continue by passing every possible variable and every possible value trying to find improvements. As soon as it passes all variables without improving, a new architecture is chosen at random and the algorithm starts over. Local search has been shown to perform well on NAS problems [36]. Pseudocode is shown in Algorithm 3.

Algorithm 3 Local Search algorithm

```
function LS($fitnessFunction$)
    $converged \leftarrow True$
    $p \leftarrow$ GENERATERANDOMINDIVIDUAL()
    $elite \leftarrow p$
    $eliteFitness \leftarrow$ EVALUATE($p, fitnessFunction$)
    while $\neg terminationCriterionSatisfied$ do
        for $x \in$ GETRANDOMLYPERMUTATION($0, .., encodingLength$) do
            for $i \in alphabet_x$ do
                $o \leftarrow p$
                $o \leftarrow$ CHANGEVARIABLE($x, i$)
                $fitness \leftarrow$ EVALUATE($o, fitnessFunction$)
                if $fitness > fitnessElite$ then
                    $fitnessElite \leftarrow fitness$
                    $p \leftarrow o$
                    $elite \leftarrow p$
                    $converged \leftarrow False$
        if $converged$ then
            $p \leftarrow$ GENERATERANDOMINDIVIDUAL()
    return $elite, eliteFitness$
```

2.5. Neural Architecture Search

Now that both the concepts of search algorithms and DNNs have been described, the concept of Neural Architecture Search (NAS) can be further elaborated upon. NAS is the automated design of neural network architectures. The aim of NAS methods is to effectively and efficiently search through a space of possible network architecture designs and find a network that is highly tailored to the task at hand [12]. While research on NAS for medical image segmentation has not been as elaborate as for image classification, it has already shown promising results by outperforming the manually designed architectures[49, 51, 55].

2.5.1. NAS Structure

NAS involves three key components: (1) the performance estimation, which is the algorithm that scores a network's performance, such that these networks can be ranked by the search algorithm, (2) the search space, which is the set of all possible networks given the specified architectural constraints; and (3) the search algorithm, which is the algorithm chosen to navigate the search space. The structure of NAS and how the components work together is visualised in Figure 2.15.

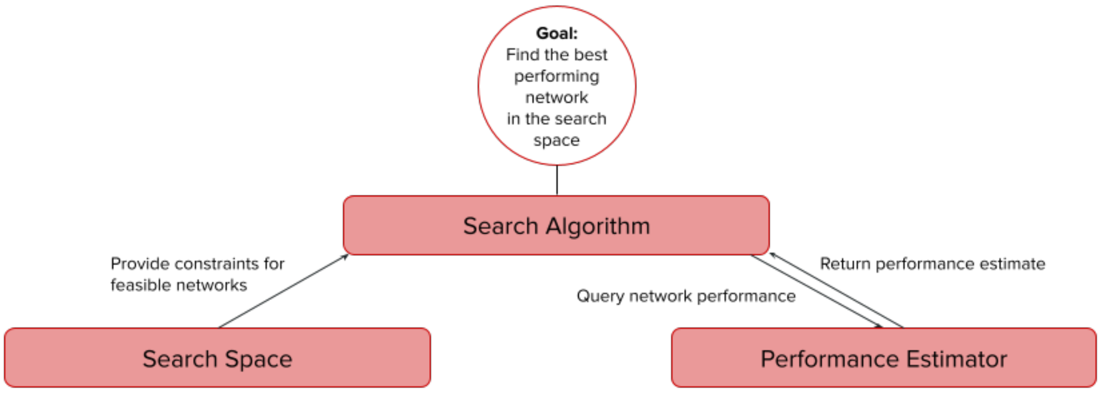

Goal:
Find the best performing network in the search space
Search Algorithm
Provide constraints for feasible networks
Return performance estimate
Query network performance
Search Space
Performance Estimator

Figure 2.15: *The structure of NAS. The search algorithm tries to find the best possible network, and generates networks from the search space to do so. A network is then queried to verify its performance, returning an estimate for the search algorithm to work with.*

2.5.2. Performance estimation

The goal of performance estimation is to give a search algorithm an accurate estimate of how well a network will perform on the task at hand. Two main challenges need to be tackled for performance estimation to work well: (1) NN training is a stochastic process, meaning that due to the large influence of stochastic methods in initialisation, data splits, data augmentation, and optimisation, a network might have varying performances in different runs. (2) Performance estimation is time-consuming. Training a network until saturation can take hours or even days, and repeating this for multiple iterations will only increase the time consumption. To circumvent this, research has looked into how to evaluate a network without having to train it. Four existing methods of performance estimation are: network evaluation, incomplete training [32, 38, 39], surrogate models [4, 29], and one-shot models [27, 36].

As mentioned, network evaluation is stochastic and time-consuming. A common approach is to evaluate for one random seed, which is a a certain value that will dictate all the random numbers generated during a run[8], and one fold of a data split [12]. Although this gives an estimate based on actual performance, this estimate could still be quite noisy due to dataset properties and other causes of noise. So far, no literature has looked into the noise of DNN performance for medical image segmentation tasks.

Incomplete training performance estimators are methods that predict performance based on a network that was not trained until saturation. This could be done using some secondary measurement of performance, like loss [39] on partially trained networks, or a metric based off of one forward pass on an untrained network [32]. However, when network evaluations are noisy, these metrics would generally have the same issues, only amplified. Observations on this can be seen in Chapter 4.

Surrogate models are another way of predicting the performance of a DNN in NAS. They aim to train a regression model on the performances from network evaluations, which are only done sparingly. Gaussian processes, and Random forests, are two popular methods for this [4, 29]. These regressors are very adept at learning functions in multidimensional spaces and will output a predicted score within seconds. This can vastly speed up NAS and is, therefore, an interesting concept.

One-shot models, are a third way of evaluating networks in a very quick fashion. The idea is to create a Super-network that contains all possible networks from the search space within it, and train it until saturation. The network chosen by the search algorithm to be evaluated, is then extracted from the Super-network and validated. This validation score is then used as a performance estimate. Although faster than evaluating networks separately, literature has shown that weight sharing degrades the performance of NAS. The extracted networks seem to under-perform and rank differently to the networks when trained individually [54].

These alternate performance estimation methods all attempt to speed up the process of performance estimation. However this speed-up mostly comes at a cost. It would be interesting to see how accurate and noisy some of these methods are in comparison with trained network evaluations.

2.5.3. Search space

The search space in NAS has many degrees of freedom. Every operation can be configured differently by using different convolution sizes, activation functions, and normalisation schemes. Additionally, every operation has consequences on the next. To make this search space traversable by a search algorithm, it is necessary to constrain and define this space. Often in NAS, the search space is divided into two different levels: a micro-level, and a macro-level [12].

The micro-level, or cell level, consists of the design choices on cell composition. This encompasses choices on operation types, e.g. pooling or convolutions, convolution kernel sizes, activation functions, etc. However the macro-level is usually fixed for this type of research so prior topology assumptions are required.

The macro-level, also known as the topology level, looks at the architectural decisions on how to stack cells, and what dimensions the feature maps should have at every position. It also looks at skip-connections, to see whether networks should become more dense, by allowing cells to use multiple concatenated previous outputs as input. The contents of every cell are fixed, however.

Bi-level search spaces, where macro- and micro-levels are searched consecutively, exist as well [55]. In this type of research, the levels are mostly searched sequentially, where the macro-level is first searched with fixed cells, and then the cell structures are optimised. However, the size of a combined search space can grow out of hand quickly. Therefore, thoughtful restrictions are necessary.

[8]random numbers are used for a plethora of different functions in CNN training, such as weight initialisation values, what augmentations to apply, what images are part of a batch, and the calculations of the gradient for the optimiser.

Instead of discrete search spaces, such as the ones explained above, one could also create a single network that contains every possible architecture in a search space, called a Super-network. Using continuous relaxation, the elements that contribute least to the network output can then be removed using gradient-based approaches [51]. However, this approach uses weight sharing, i.e., overlapping sub-networks with different architectures within the Super-network use the same weights at the overlapping location. According to Yu et Al. [54], this could deteriorate the quality of results. Also, this type of search is computationally very heavy due to the size of the Super-networks. Therefore, the decision was made to stick to discrete search spaces without weight sharing.

2.5.4. Search algorithms

NAS search algorithms can be categorised into Reinforcement Learning (RL), gradient-based algorithms (GD) and Evolutionary Algorithms (EA), and other search techniques such as Random Search (RS) and Local Search (LS). Reinforcement learning works on the basis of an agent that constructs architectures, and is rewarded for good architectures and punished for bad ones. The eventual policy learned by the agent should output the best possible network. Gradient-based algorithms are all methods that use an estimate of the gradient to optimise the performance of a network. These gradient-based search algorithms only work for continuous search spaces using Super-networks. Evolutionary algorithms are population-based optimisation methods. SGA and P3GOMEA, as described in sections 2.4.3 and 2.4.4 respectively, are part of this family of algorithms. The LS algorithm mentioned in section 2.4.5 is an example of a simple point-based improvement algorithm.

Evaluating these different approaches in real-life scenarios is quite costly. It is for this reason that benchmarks were created. These benchmarks are logs of performances of networks, such that performance estimation methods are no longer necessary. Benchmarks often used for image segmentation are NAS-Bench-101 [53], NAS-Bench-301 [42] and other benchmarks such MacroNAS-C10 and MacroNAS-C100 that were created for multi-objective NAS [36]. These benchmarks however do not contain any medical datasets, mostly focusing on CIFAR-10 and CIFAR-100, two natural image classification tasks. The search spaces involved are also very different, leading to very different optimisation problems.

Research on these benchmarks, has shown that random policies, i.e. Random Search, do not underperform compared to other algorithms [54]. A cause for this, could be that performance estimates of networks in a benchmark are based on a single evaluation, and these still contain much noise. The fitness landscape seen by the search algorithm could therefore be very different than the one when evaluating the same networks using cross-validation. This leads to the possibility of an elite network being found based on a peak induced by noise rather than a robust improvement. Other research has shown that simple Local Search can be a strong benchmark when doing multi-objective optimisation [36]. Due to these varying results, it is very difficult to decide on a single search algorithm that seems to perform particularly well on NAS problems.

Between RL, GD, and EAs, a choice was made to focus on EAs. As mentioned GD methods only work on Super-network approaches with available gradients and not on discrete search spaces, so they were left out. RL and EAs have both been shown to work for NAS [4, 36, 55, 57], however SotA RL methods rely on Neural Networks to learn policies for the problem, which has two issues: (1) these methods require relatively many hyperparameters choices, and tuning each of these parameters is very computationally expensive for NAS, (2) if a NN is used to learn the fitness landscape of the policy, should it not be optimised using NAS as well? Additionally, EAs have been shown to work well with NAS, and have a robust performance in fitness landscapes with many local optima [35]. Recent model-based EAs have also been shown to be competitive in diverse discrete optimisation problems [47]. This led to the decision to focus on EAs in this thesis.

2.6. Suggested improvements

The research on NAS has already been quite extensive for classification tasks [12]. As less research has been done on NAS for medical image segmentation, especially with clinical datasets, there is room for improvement and new insights in this field. Potentially, strong performing search algorithms could be translated from other domains to the medical image segmentation domain. However, multiple NAS papers [52, 54] indicate that a lot of search algorithms do not outperform random search policies after validation, and therefore need to be evaluated before choosing which one performs well on medical image segmentation tasks. The search space cannot be translated as it is specific to segmentation tasks. Bi-level search spaces have shown promising results in both classification and segmentation tasks [4, 45, 51, 55], but cell level variation is very small and the search space often very elaborate [52], which could lead to relatively small improvements being lost in the noise of performance estimation. Below are the improvements suggested by this thesis for the different components of NAS. They are linked to the research questions from Chapter 1.

2.6.1. Reduced noise performance estimation

Estimating DNN performance on medical image segmentation is dependent on the chosen segmentation similarity metric, the dataset and the network evaluation method. All these aspects can cause different amounts of noise in the value returned. Existing metrics rely on different segmentation properties [6, 9, 34], such that decisions have to be made on what property to optimise. These metrics are also noisy in comparison with metrics used for other tasks such as image classification. In medical image segmentation, the performance values per segmented class are computed using values per pixel/voxel, whereas for image classification performance values per classified class in an image are binary. This leads to a larger range of possible segmentation similarity scores, and more sensitivity.

Datasets in medical image segmentation are also relatively small, often containing less than 100 scans. The regions of interest in these scans, such as organs and tumors, can also vary a lot between patients. Next to patient variation, human error and difference in physician interpretation, also cause variation in the provided delineations. This variation within and small size of datasets, make the task of quantifying network performance more susceptible to noise, as datasplits can cause both favourable and unfavourable situations.

Research on how noisy metrics are and how much noise is induced by a dataset, is understudied in literature, although possibly impacting DNN performance estimation quite significantly. It would be interesting to evaluate this noise and analyse its effect on performance estimation of DNNs.

Next to noise in the data, estimating performance of a DNN on any task has inherent noise, as the initialisation and training of the network is stochastic [42]. Assessing DNN performance by training the DNN and averaging the score of the provided segmentations on a validation set is computationally expensive. For this reason, NAS research has tried to speed up the process by creating surrogate models [4, 29], or incomplete training predictors [32, 39], to predict performance. However, these methods add inherent error to the estimated performance value, which could lead to sub-optimal networks being found as elite by NAS algorithms. As there are many possible sources of noise for performance estimation of DNNs for medical image segmentation, it could be possible that these alternate methods, as well as single training run estimations might not reflect the true performance of a network. Literature has even shown that after validation, NAS results often do not outperform random search policies [52, 54], indicating performance estimation differences between the used evaluation method and validation.

In Chapter 4, different performance estimation methods are analysed in order to gain insights on the noise in network performance estimation. With these insights, a method is proposed to evaluate whether or not NAS will be able to robustly improve on SotA networks for a certain dataset.

2.6.2. Simultaneous Multi-Block search space

Multiple search spaces have been proposed for medical image segmentation. These include a topology search, where the search algorithm looks for a U-Net-like encoder-decoder topology [3, 49], a bi-level search where the topology search is followed by a cell search [55], or a combination of topology and cell search using continuous relaxation and a so-called Super-network [27, 51]. Within the scope of a U-Net-like topology search, there is room for making the search space more flexible. Instead of having an elaborate cell search space and repeating the same cell structure throughout the DNN, using various cell structures in a network could improve performance. In order to make this possible without the search space growing too much, the possible cell configurations can be limited. These configurations are selected using existing knowledge about DNNs with a strong performance on classification tasks. Using this pre-selected pool of configurations taken from advanced well-known classification networks [14, 18, 44], instead of searching for a cell configuration from scratch, can also lead to more structural differences in the cell search space. Finally, in contrast to recent bi-level search space research [55], where topology level search is followed by cell level search, these levels can be searched simultaneously. This allows algorithms to take possible interaction between the topology and cell search into account, potentially yielding better performing networks.

In Chapter 5, this new search space is evaluated and the results compared to SotA hand-crafted networks, as well as to networks obtained using different search spaces.

2.6.3. Robust search algorithms

As reports of how algorithms perform on benchmarks differ, it is interesting to compare several search algorithms on different levels of noise in performance estimation, which is inherent to the task of medical image segmentation. By comparing the performance of search algorithms on an unbenchmarked dataset, a performance comparison can be done.

In Chapter 6, a comparison is done between LS, SGA and P3GOMEA (see Section 2.4 for pseudocode) on an unbenchmarked clinically realistic medical image segmentation dataset. In these experiments, algorithm performances and robustness are compared.

3 Data

This chapter contains information on the datasets used throughout this thesis. It will cover the type of scans involved, the region of interest to be segmented and the properties of the data.

3.1. Datasets

Datasets in medical image segmentation contain sensitive information which make them quite restrictive in terms of privacy. However, in order to compare performance of NAS methods it is crucial that the dataset is publicly available. To accomplish this, the organisers of the Medical Segmentation Decathlon (MSD) [1], collected and anonymised medical imaging datasets and set-up an international open competition in order to promote the search for generalisable ML for medical image segmentation.

3.1.1. Medical Segmentation Decathlon

DNNs and/or NAS strategies that generalise well should be able to perform well on varying problems without human intervention. They should be able to operate well on different medical image segmentation dataset sizes, scan settings, amounts of classes to segment, and regions of interest. The MSD collection contains 10 datasets with varying properties and a leaderboard to see how methods compare. The different datasets have varying difficulties to overcome, that are often encountered in medical images, such as small datasets, unbalanced labels, multi-site data and small regions of interest. Teams from around the world are allowed to train networks on training data, and submit segmented test data to evaluate performance. The results of these segmentations were published in 2021, ranking teams per task [1]. A global leaderboard is also still kept to show what networks performed well on different tasks.

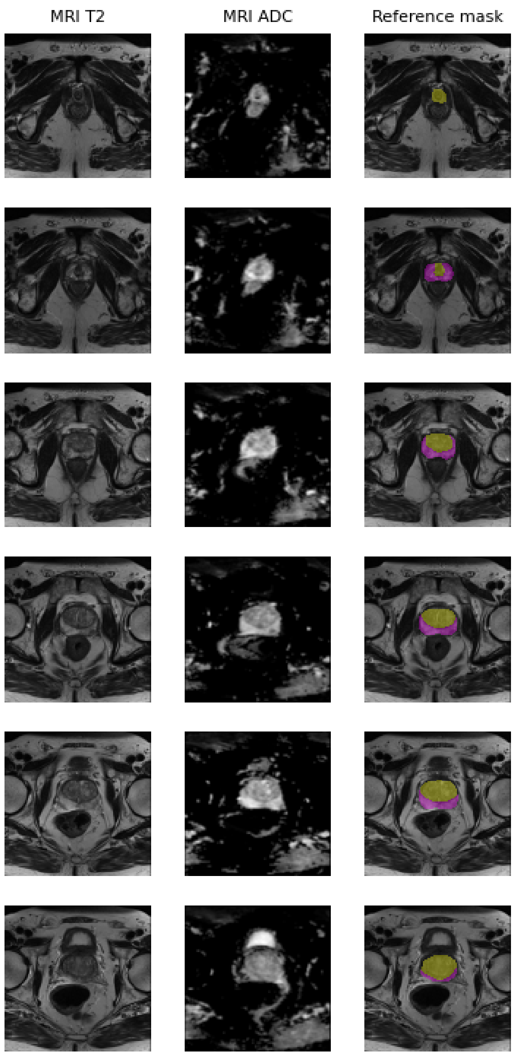
From the MSD datasets, two were chosen to experiment with, such that comparison is possible with other SotA NAS strategies and DNNs. A third dataset from Amsterdam University Medical Centre, location AMC, University of Amsterdam, was used for experiments where comparisons are only relative to other approaches in this thesis.

3.1.2. MSD Prostate

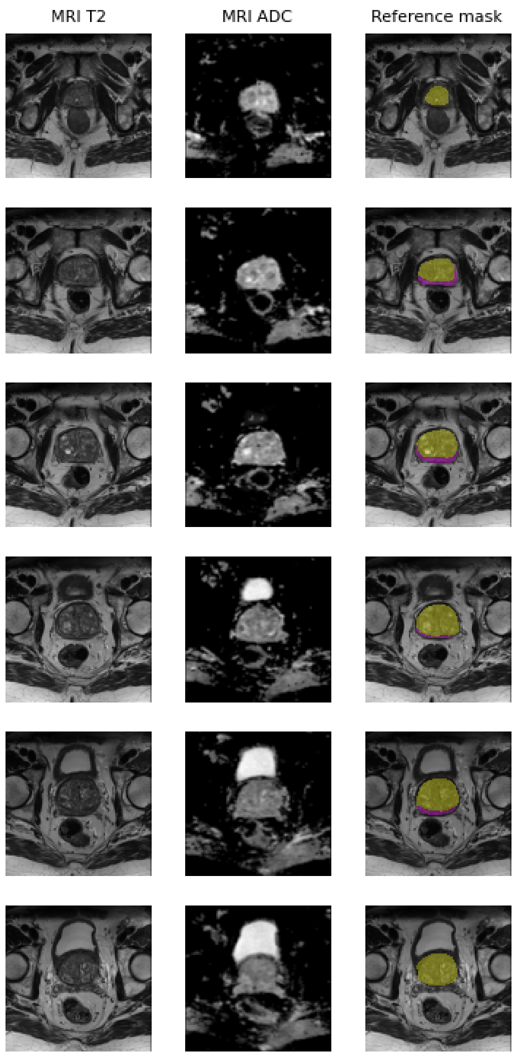
The first dataset that was chosen for experiments is the prostate dataset from the Medical Segmentation Decathlon. It contains 32 multi-modal MRI scans with two classes: the central gland and the peripheral zone of the prostate. What makes this segmentation task challenging is that there are two adjoint regions with large inter-subject variations [1]. The dimensions of the scans are approximately $2 \times 16 \times 320 \times 320$, with these exact values being the median. As some images contain more slices or have a slightly different width and height, the images were resampled to be 128×128 and stored as 2D images. This translates to 2 modalities, 13-20 slices, and a resolution of 128 by 128 pixels per slice.

The two modalities in the MRI scan are T2 and ADC. The former captures the time it takes for a magnetic resonance signal to irreversibly decay to $\frac{1}{e}$ (37%) of its initial value. It creates an image where fatty tissue and fluids have high intensity values and bone, air and proteins are dark. Apparent diffusion coefficient (ADC) images, are MRI images that show diffusion of water molecules in tissue. ADC images have high intensities when diffusion rates are high [2]. Examples of these two modalities are given in Figure 3.1.

The goal is to learn features from these two modalities such that the central prostate gland and the peripheral zone can be segmented accurately. These divisions seem to vary largely per patient, and an example of this can be seen in Figure 3.1. According to McNeal in his paper *The zonal anatomy of the prostate* [31], "the peripheral zone constitutes over 70% of the glandular prostate. It forms a disc of tissue whose ducts radiate laterally from the urethra lateral and distal to the verumontanum. [...] The central zone constitutes 25% of the glandular prostate. Its ducts arise close to the ejaculatory duct orifices and follow these ducts proximally, branching laterally near the prostate base. Its lateral border fuses with the proximal peripheral zone border, completing in continuity with the peripheral zone, a full disc of secretory tissue oriented in a coronal plane". In Figure 3.1 the different areas can be seen clearly in the masks as provided by physicians at the Radboud University Medical Centre. This dataset will be referred to as **MSD Prostate** throughout this thesis.



(a) Patient 1.



(b) Patient 2.

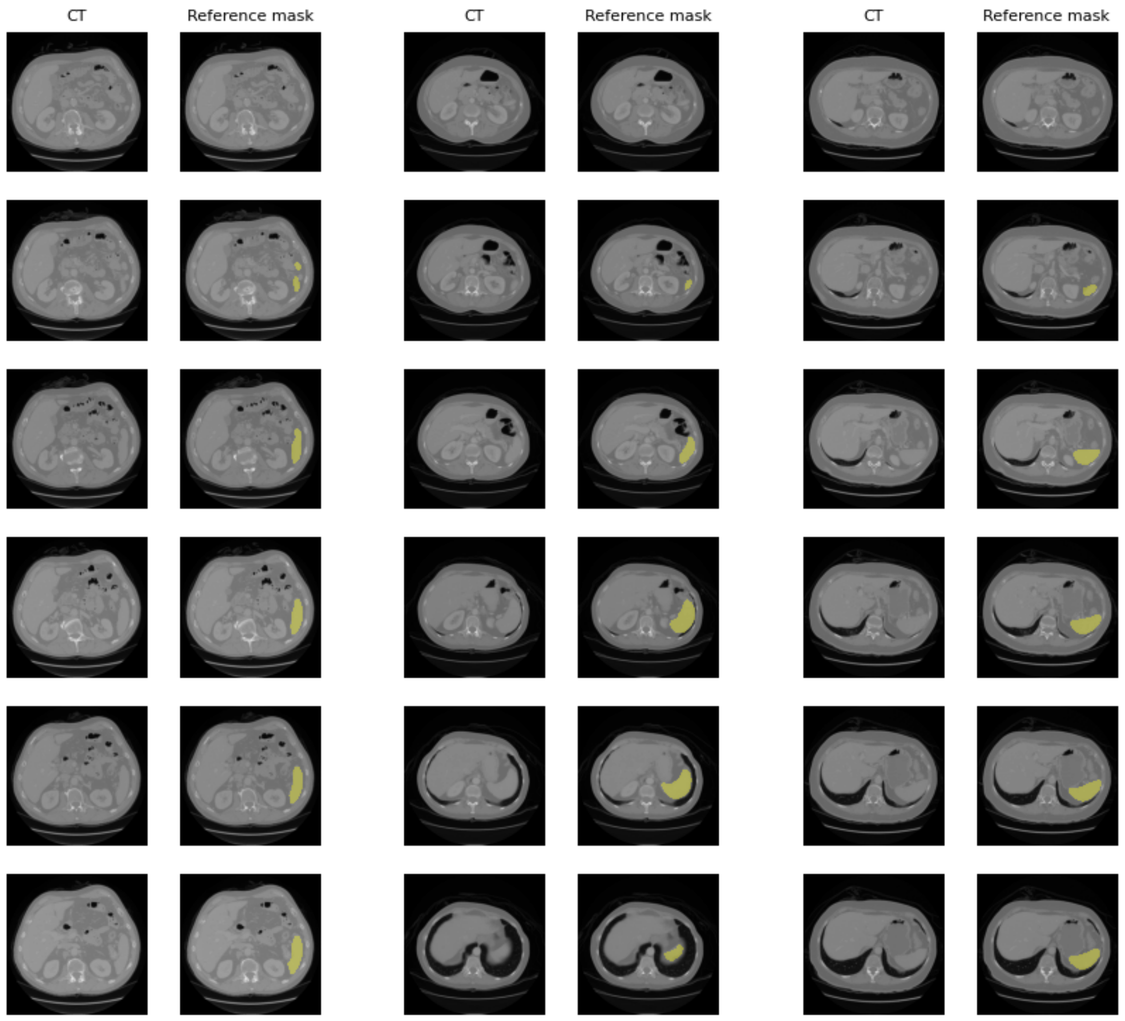
Figure 3.1: *A selection of slices of prostate MRIs from two patients within the MSD dataset. The first column contains the T2 modality, and the second is the ADC modality from the MRI scan. In the third column the T2 modality is layered with the reference mask showing the central gland in yellow and the peripheral zone in magenta. The difference in relative zone size is quite clear between patient 1 in Figure 3.1a and patient 2 in Figure 3.1b.*

3.1.3. MSD Spleen

The second dataset that was used is the spleen dataset from the MSD. In contrast to the prostate dataset, it has a single modality, generated using a CT scanner. It contains 41 scans with a single class, the spleen.

The foreground size of the spleen however varies largely throughout a scan. The dimensions of the scans are approximately $90 \times 512 \times 512$ with these exact values being the median. These were resampled to 2D images of with resolution 128×128. This translates to 30-167 slices per patient, and a resolution of 128 by 128 pixels per slice.

The CT scan uses X-ray technology to create cross-sectional slices of the human body. The CT-scanner calculates the attenuation of the material, based on of the decrease of electromagnetic radiation due to tissue absorption [17]. Denser tissues, such as bone, will have high intensity in a CT-scan while air, will be very dark. In Figure 3.2 the different organs can be seen in the scans and a clear reference mask as provided by physicians at the Memorial Sloan Kettering Cancer Center. This dataset will be referred to as **MSD Spleen**.



(a) Patient 1. **(b)** Patient 2. **(c)** Patient 3.

Figure 3.2: *A selection of CT scan slices from three patients within the MSD Spleen dataset. The first column contains the CT scan, the second the CT scan layered with the reference mask showing the spleen.*

3.1.4. AMC Prostate

The last dataset is not publically available, but was provided by the Amsterdam University Medical Centre, location AMC, University of Amsterdam, as part of the FEDMix project under supervision of Arkadiy Dushatskiy, of CWI. It contains single modality MRI scans. It contains 41 scans with a single class containing the entire prostate. The prostate in in these scans contain tumors and have been pierced with catheters, as the scans were collected during brachytherapy, where the prostate cancer is radiated. This makes the task different and potentially more difficult than general prostate segmentation. The dimensions of the scans are approximately $13 \times 128 \times 128$ with these exact values being the median. This translates to 13 slices per patient, and a resolution of 128 by 128 pixels per slice.

The benefit of this dataset is the easier task, due to it containing a single class, single modality input, with few slices without foreground. This shortens experiment time because DNNs converge faster on this task. It also was collected in clinical practice, mimicking a realistic situation. For these reasons, all experiments that do not compare results with SotA performances are run with this dataset. For privacy reasons, the dataset is not visualised in this thesis. It will be referred to as **AMC Prostate**. Before proceeding, a word of thanks to the Amsterdam University Medical Centre, location AMC, University of Amsterdam, for providing the dataset, especially Bradley Pieters.

4 Performance Estimation

This chapter describes the difficulties of performance estimation for medical image segmentation DNNs. It describes how similarity between two segmentations can be scored and how different metrics optimise different segmentation properties. It continues by elaborating on how network performance can be established, and what factors can induce noise in network performance values. Next, insights are given on whether network performance can be predicted in a quicker way than evaluating a trained network. It proceeds by showing how much noise network evaluations contain, and that rankings of the same group of architectures vary when training them for different seeds and on different folds. It also shows how noise propagates through the entire NAS procedure. Finally, it provides insights on whether or not NAS can effectively find networks that robustly improve on SotA, and what the computational cost is to do so.

4.1. Segmentation similarity metrics

In Chapter 2.3.3, an introduction is given to three different kinds of segmentation similarity metrics. These are the Dice-Sorensen Coefficient (DSC), Hausdorff distance (HD), and Surface Dice (SD). These metrics relate to different segmentation properties. Where DSC quantifies the ratio of correctly labeled pixels/voxels, HD finds the biggest difference between segmentation surfaces, and SD looks at the ratio of how much of the two segmentations are inside a certain threshold apart from each other. Optimising for one of these metrics could lead to different segmentation properties being favoured. In the example in Figure 4.1, it is clear to see that segmentations could score very differently for these metrics and could be considered good by one but poor by the other. Eventually it would be up to a physician to give a preference as to which metric should be optimised.

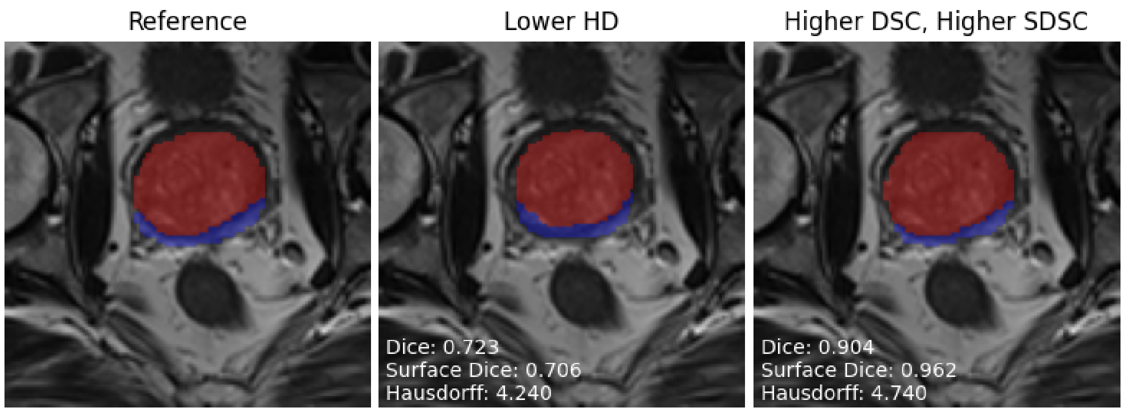


Figure 4.1: *The different metrics lead to optimising different properties in a segmentation. In the figure, the segmentation on the right has a better (higher) Dice-Sorensen Similarity Coefficient, and Surface Dice Similarity Coefficient, while the segmentation in the middle has a better (lower) Hausdorff Distance.*

For all these metrics, the score per segmentation is a real positive number ($\mathbb{R}+$). For DSC and SD, the values will range from $[0, 1]$, and for HD the values range from $[0, \infty\rangle$. This is different to the categorical output of a classification task, where the score of an image can only take on $C + 1$ different values, C being the number of possible classes. This means that for the metrics in image segmentation, scores are more sensitive to small changes in DNN output, as these scores are altered quicker per differently labeled pixel/voxel, where for classification scores are calculated per correctly labeled class and thus more robust.

To score the performance of a DNN, multiple segmentations need to be evaluated. The most logical way to accumulate these segmentation scores, is to average the score per image over the validation set images. As datasets in medical image segmentation are often quite small (AMC prostate and MSD spleen contain 41 scans, and MSD spleen contains 32), the validation images are part of a small set. This means that average performance between splits and also between folds can vary quite much, which can be seen in Section 4.2.2.

Another factor that makes scoring of a DNN difficult for medical image segmentation, is the fact that segmentations are often performed by different physicians, that delineate organs differently, as well as having varying delineations on different days. These phenomena are called inter- and intra-physician variability [7], resp., and cause noise within a dataset. Ideally the DNN would learn a perfect average of these delineations.

To conclude, the metrics in medical image segmentation use different properties to score similarity between segmentations. Using different metrics as performance estimate will lead to different segmentation properties being optimised by a DNN. The larger scoring range and sensitivity of the metrics, in combination with the relatively small size of the datasets, and inter- and intra-physician variability within the datasets, make measuring performance of a DNN for medical image segmentation both more subjective, and more sensitive to noise, than for image classification.

4.2. Noise in performance estimation

Noise, or variance in results when reproducing them multiple times, is inherent to all stochastic procedures. As Neural Networks fall into this category, noise in the optimisation and evaluation cannot be disregarded. However, if noise is relatively small compared to the possible performance values for a certain task, it can be less of an issue. Through varying outcomes in NAS experimentation throughout this chapter, the importance of understanding and quantifying noise becomes increasingly clear.

For NAS, the different sources of noise can be categorised into the noise induced by performance estimation and the noise caused by a stochastic search algorithm. In Figure 4.2, a breakdown is given of the different aspects within NAS that can induce noise, in particular for performance estimation.

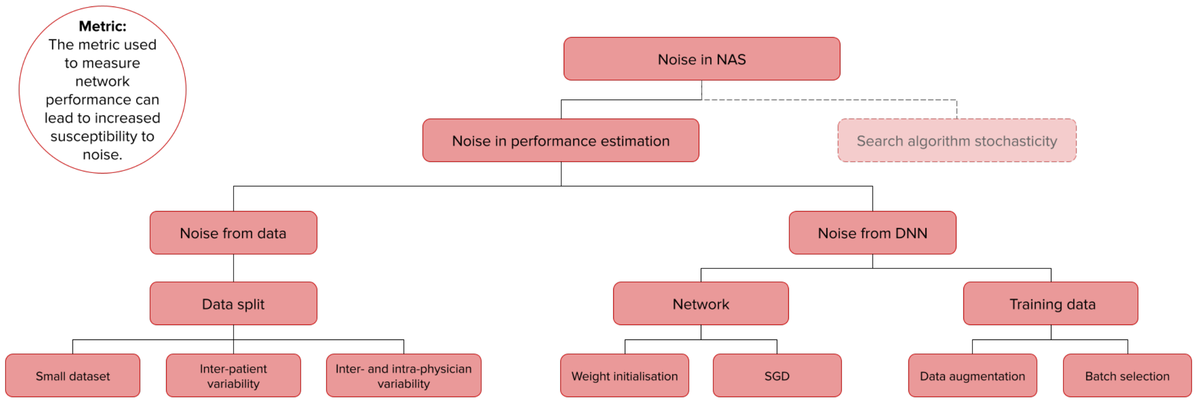

Metric:
The metric used to measure network performance can lead to increased susceptibility to noise.
Noise in NAS
Noise in performance estimation
Search algorithm stochasticity
Noise from data
Noise from DNN
Data split
Network
Training data
Small dataset
Inter-patient variability
Inter- and intra-physician variability
Weight initialisation
SGD
Data augmentation
Batch selection

Figure 4.2: *The different sources of noise in NAS, in particular for performance estimation of a network, are shown. While the performance metric does not directly induce noise, it influences the accuracy, and the susceptibility to noise, of the performance estimation. Noise sources for performance estimation are categorised in 2 groups in the figure. The first, is the noise generated by the data used. The second, is the noise caused by the optimisation of the DNN.*

The sources of noise for performance estimation can be split into two categories: (1) noise from the data, which can be ingrained in a dataset, and results in different performance values for different folds in a data split, and for different average performance values for different splits; and (2) noise from the DNN optimisation, which is caused by stochastic mechanisms within the DNN, such as random weight initialisation, SGD, and training batch composition (i.e. what order training data is loaded into the network). This noise can also be self-induced by data augmentations during training, something that is often necessary to increase training samples when using small datasets.

The third source of noise comes from the stochastic elements in search algorithms. These algorithms often generate (an) initial solution(s) randomly, and contain stochastic elements in the search approach, e.g. crossover, donor selection, and variable selection. However as this chapter looks at performance estimation of networks outside the NAS procedure, this will be disregarded until Chapter 6.

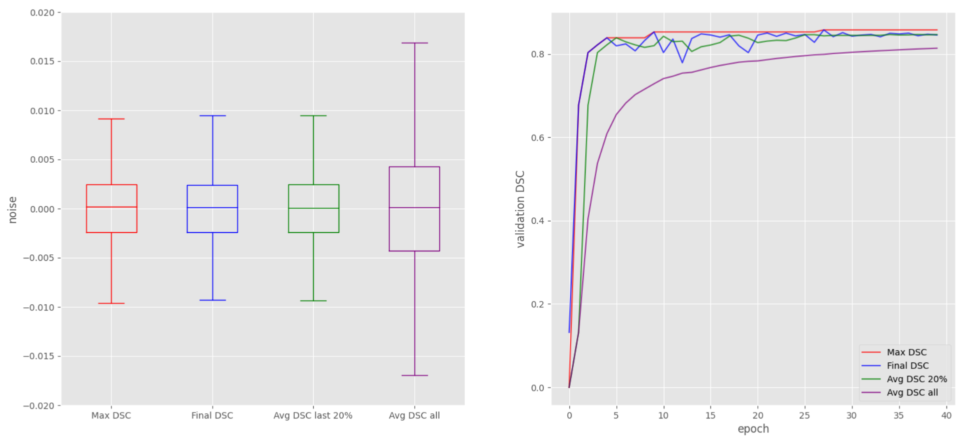
4.2.1. Measuring DNN performance

Before diving deeper into how to deal with different sources of noise, it is necessary to look at performance metrics, and how to translate from a single segmentation score to a performance estimate of a DNN. As explained in Section 4.1, similarity can be measured between segmentations and averaged for a validation set. But this average changes as a network is trained, and weights are optimised. This is a result of the noise caused by DNN optimisation.

First of all, after having decided on a validation metric, a decision needs to be made on what value to use for the performance of a network: the final validation accuracy, the maximum validation accuracy, or an average accuracy of multiple epochs. Other decisions, such as the number of epochs for training, which optimiser to use, and the learning rate of the optimiser, all affect the value of the outcome.

To simplify the decision of network setup, the optimiser and learning rate were fixed to ADAM and a learning rate of $1e-3$ with polynomial decay with exponent 0.9. This was done after experimenting with U-Net and manually searching for a high-performance optimiser. The AMC dataset was chosen for experiments, as it contains a simple single class segmentation task, and networks converge relatively quickly compared to the other datasets discussed in Chapter 3. It also contains scans collected in actual clinical circumstances, giving more realistic insights in NAS performance for clinical purposes. Here the DSC was used as validation metric. The networks that were evaluated were randomly chosen from a simplified version of MB-NAS (which is a simplified version of the search space described in Chapter 5), containing the same macro-level search space, but a micro-level search space that only includes the VGG block, and only varies convolution kernel size in the micro-level.

To find what metric would be the most stable, and would best reflect if a network performance estimate is robust, an experiment was done to evaluate how much noise the maximum, final, and average validation DSC contain. To measure the noise in the metric, 20 randomly chosen networks in the search space were evaluated. The noise was computed by looking at the difference between the average validation DSC of every network for 5 different seeds. This is visualised in a box-plot in Figure 4.3.



(a) Noise distributions for different training values of DSC. **(b)** Plots of different training values of DSC per epoch.

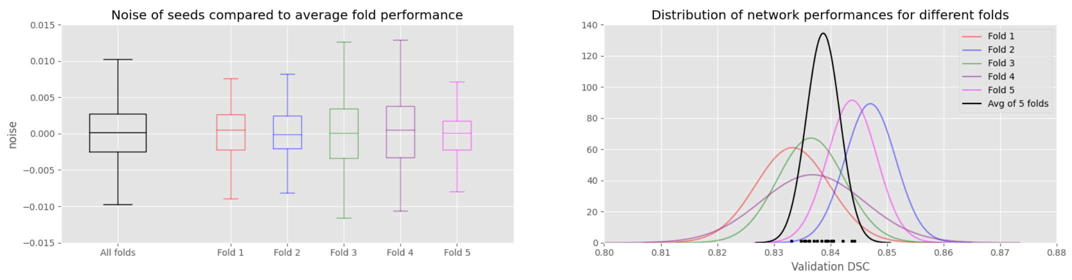
Figure 4.3: *Example noise distributions for maximum DSC, final DSC, average DSC of the last 20% of epochs, average DSC over all epochs, is given in 4.3a. The box-plots show the 1st quantile (Q1), median, and 3rd quantile (Q3) values. The whiskers show $Q1-1.5\times IQR$ and $Q3+1.5\times IQR$ values, where the interquantile range (IQR) is $Q3-Q1$. The first three box-plots are very similar. In 4.3b, example plots are given for a network evaluation with different stored values of the DSC metric.*

The outcome of the experiment was that the noise for the maximum DSC, the final DSC, and the averaged value of the last 20% of epochs is quite similar. This led to the decision of using the average last 20% of epochs as to not overestimate network performance. The number of epochs was decided by looking into the saturation of the network. For the AMC dataset, this was set at 40 epochs.

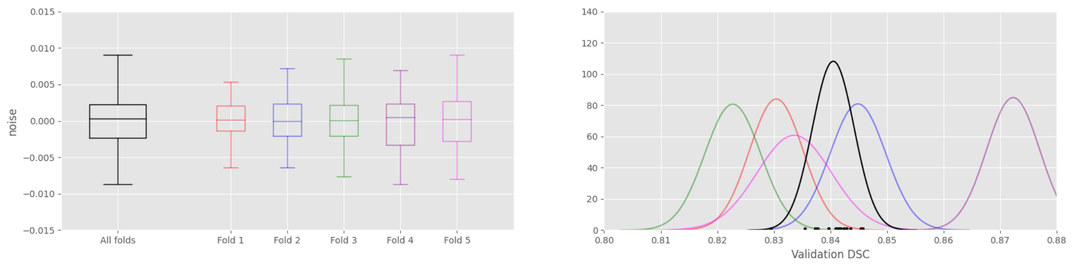
4.2.2. Evaluating noise

As can be seen from the box plots in Figure 4.3, there is noise between different seeds. To reduce this noise, it is necessary to sample outcomes multiple times. However, DNN network evaluations are expensive. Training a network until saturation can take hours, even days. So unfortunately, the amount of samples possible per network evaluation will lead to less possible network evaluations when running NAS due to computational budget. Therefore, it is important to find out how many samples are necessary without sampling excessively.

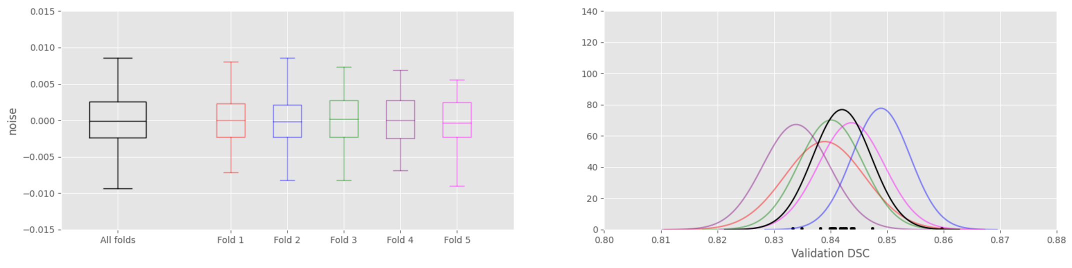
In the experiment visualised in Figure 4.4, 20 randomly selected networks were evaluated for 5 seeds, on 3 different 5-fold splits. The figure shows the distribution of noise per seed for different folds in box plots. It is clear that these noise levels, if evaluations are only done once, can cause very misleading perceptions of network performance. The figure also shows that the average and variance of performance levels for different folds, can be vastly different. Also, statistically significant differences in performance of networks can arise when using different dataset splits. This can be troubling when comparing results in different papers based on cross-validation scores.



(a) 5-fold data split 1.



(b) 5-fold data split 2.



(c) 5-fold data split 3.

Figure 4.4: *Noise and performance values in different data splits of the AMC Prostate dataset. On the left, box plots are shown that visualise the noise levels of different seeds based on the average network performance of a certain fold. This noise is induced by the stochastic optimisation of the DNN. The black box-plot is the average noise for a certain data split. It is clear to see that noise can vary per fold, as can be seen in Fold 2 and 3 having more noise relative to the other folds in 4.4a. The average noise is also relatively large for the data split in 4.4a vs. 4.4b and 4.4c. On the right, the distribution of network performances for different folds in a split are shown when averaged over 5 seeds, as well as the performance of the networks when averaging over all the folds. These distributions show the noise induced the data, when using different folds and data splits. The mean of performance values can be different between splits, with split 3 obtaining a higher performance than split 1, with a statistical significance of* $p = 0.017$ *for an independent t-test. Also, the performance distribution between folds can be very different, as seen by the different shapes and locations of the 'bells'.*

The results of this experiment show that noise caused by data can be observed when looking at average performance estimates on various folds and data splits. Using multiple data splits and folds could reduce this noise. The noise caused by DNN optimisation can be observed by looking at the difference in performance estimates of the same network on the same fold. This could be reduced by averaging over multiple seeds.

These insights are valuable, as the noise and the average performance value of a network, might help predict how networks are ranked when sampled. To obtain more insights on difference in ranking, another experiment was conducted. In it, the difference in ranking was compared by looking at the average correlation between rankings using different evaluation methods. These methods look at averaging over different amounts of seeds, folds, and epochs, and are compared to the ranking when averaging over scores after 40 epochs for 5 seeds and 1 data split of 5 folds. For example, how does the ranking based on a 1 seed, 1 fold, 20 epoch evaluation of a group of networks, compare to the ranking of a 5 seeds, 5 folds, 40 epoch evaluation of the same networks. This is visualised in Figure 4.5. To compare the network rankings the Spearman rank correlation [33] is used. This metric compares the rank of a number of random DNNs in a search space, on one side ranked by the performance estimator, i.e., the evaluation method using a different amount of seeds, folds and epochs, on the other side ranked based on performance score, i.e. the ranking after 40 epochs averaging over 5 seeds and 5 folds. The Spearman rank correlation is calculated using:

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

where, d_i is the difference in rank of i, and n is the number of participants. To get a feeling of this metric, example correlation levels and ranking differences are visualised in Figure 4.6. In Figure 4.5, the rank comparison shows that the ranking based on a single seed, single fold network evaluation, has a very low correlation with the ranking based on the performance value of the same networks when averaging over 5 seeds and 5 folds. This means that doing NAS with single seed, single fold evaluations, something very common in NAS for classification tasks, does not work for the medical image segmentation tasks at hand, as it leads to networks being found that might not perform well on the entire dataset or for different seeds.

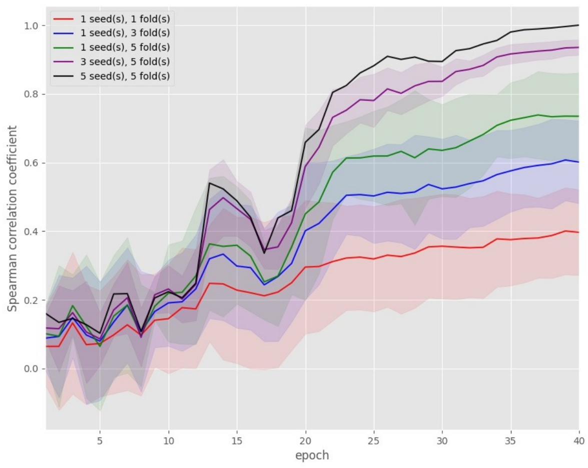


Figure 4.5: *Spearman correlations for the ranking per epoch when compared to the final ranking of a 5 seeds, 5 folds evaluation. These correlations are shown for different multiple seed and fold runs. The solid lines represent the mean while the shaded area represents the standard deviation. The ranking is based on the average DSC for the last 20% of epochs, which is averaged over every possible combination of multiple seeds and folds that complies with the amount of seeds and folds shown in the legend. This is done for 20 random architectures.*

This finding is reinforced when calculating the Mean Absolute Error (MAE) for a classification task and a medical image segmentation task. This is done by finding the mean difference between the performance value of a single seed vs. a validation value using multiple seeds. In classification, for networks in an example search space, NAS-Bench-101 [53], where single seed evaluations on CIFAR-10 and CIFAR-100 are used and evaluated on a validation set, the MAE is $\approx 4.5e-3$ [42], when validating with two other seeds. In comparison, the MAE for the networks evaluated for the AMC dataset and the simplified MB-NAS search space, is $\approx 9e-2$, which is 20 times larger than for CIFAR-10 with a simple search space.

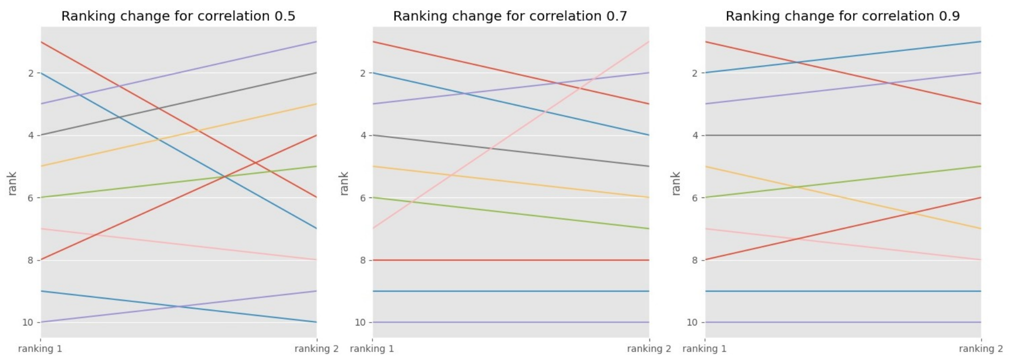


Figure 4.6: *Three examples of rank comparison for correlation levels 0.5, 0.7 and 0.9 for qualitative comparison.*

This has quite an implication, because it means that cheap network evaluations are not going to work well with NAS for medical image segmentation. The search algorithm would not be able to rank the networks such that they reflect robust performance.

4.3. Alternate performance estimators

Before abandoning cheaper evaluations, such as single network evaluations, other methods to estimate network metrics are attempted. Because of the computational cost of network evaluation, a lot of research has gone into finding ways to accurately predict architecture performance without having to perform full network evaluations. Three proposed methods from image classification are incomplete training performance predictors, surrogate-assisted performance estimators, and one-shot models. To analyse if these estimators are useful for NAS for medical image segmentations, their performance prediction is compared to full evaluation performance scores.

4.3.1. Incomplete training performance predictors

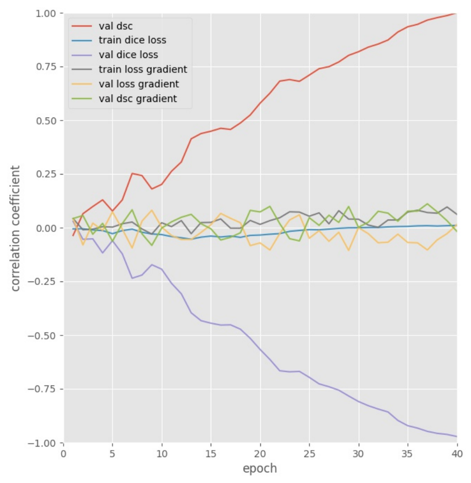
In several papers [32, 38], authors find high correlations in metrics from DNNs that are not trained until saturation, and the validation accuracy of the same DNNs. In [38] the training loss, in particular the sum of the training gradient is used to achieve Spearman rank correlations of > 0.8 for all the evaluated classification datasets and search spaces. As no literature for such performance estimators exists for medical image segmentation tasks, these methods first have to be translated. To evaluate if this also holds for medical image segmentation tasks, different metrics and their correlations with the final validation score are analysed. These experiments can be seen in Figure 4.7.

The experiments look at the AMC prostate dataset, and use a simplified version of the MB-NAS search space, using only VGG blocks and only varying convolution kernel size per cell. In a group of 20 random networks, the training loss, validation loss, validation accuracy, the training loss gradient, and the validation loss gradient are shown. The values are also smoothed with different methods, showing a sum of all previous values, sum of the last 50%, sum of the last 20%, and no smoothing. Unfortunately, all these metrics show poor rank correlations at early stages and do not come close to values useful for prediction. What seems to work for CIFAR in image classification, does not seem to translate to the medical image segmentation domain, at least not for the AMC Prostate dataset.

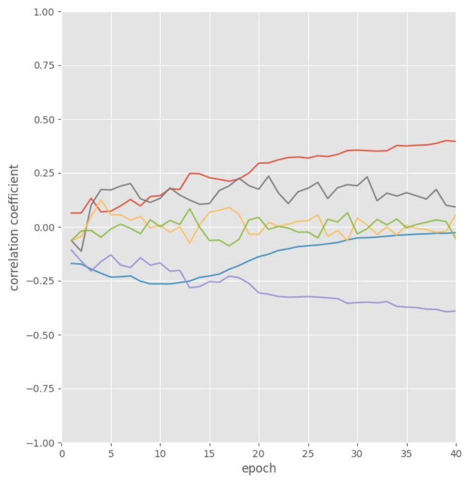
4.3.2. Surrogate models and One-shot models

Another approach to creating cheaper performance estimation are surrogate models. These methods train regression models with a set of full evaluations to guide the search to another possible high performance network [4, 12]. Only sparingly are networks evaluated in order to improve the surrogate. In order to evaluate if surrogates work for medical image segmentation, it is necessary to know if NAS without surrogates can work, to know with what type of evaluations to train the surrogate. Only after that, can a surrogate be designed. As this task is difficult and search space specific, surrogates cannot be immediately translated from classification tasks and require elaborate research. For this reason surrogate models are left out in this thesis.

One-shot models, using weight sharing, try to bypass retraining networks by training a single so called



(a) Spearman correlation coefficients of different metrics during network training with the ranking after evaluation of that same seed and fold after 40 epochs.



(b) Spearman correlation coefficients of different metrics during network training with the ranking after evaluation averaged over all seeds and folds after 40 epochs.

Figure 4.7: *Spearman Correlation coefficients of multiple metrics during the training of a network. Other than the validation Soft Dice loss and the validation DSC, none of the metrics have a visible correlation with the final performance of the 1 seed, 1 fold run, as shown in 4.7a. None of the metrics have a strong correlation with the averaged final performance of 5 seeds, 5 folds evaluation, although the correlation of the train loss and train loss gradient seems slightly higher than for the 1 seed, 1 fold case, as seen in 4.7b.*

supermodel. These models unfortunately have been shown to have deteriorated performance of evaluated networks, and the correlation with the same networks trained without weight-sharing is low [54]. This low correlation with eventual performance is the purpose of performance estimation. For this reason, and computational restraints, supermodels are left out of this thesis as well.

4.4. Probability of finding the best network

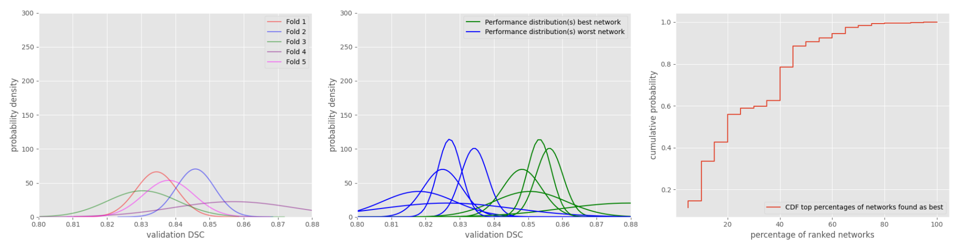
The previous sections show that, due to the noise in medical image segmentation tasks, multiple network evaluations are necessary in order to overcome noise and obtain an accurate performance indication of how a network performs. To give an indication of how many are needed for the AMC dataset, an estimation of the probability of finding the best network is given in Figure 4.8. The probability is based on the Gaussian distributions shown in Figure 4.4. The network ranking is created by using the mean DSC from a 10 seeds, 5 folds evaluation for a 5-fold split. The difference in performance distribution is then shown based on the scores of these networks. For example, for 1 seed, 5 folds evaluations, a distribution is created based on the 10 seeds that were evaluated for that network. This is done for both the worst and best network in order to visualise the difference in distribution. By using the same method to create a performance distribution for every network, a CDF of the probability of the best sampled network being in the top X percent of networks (based on the 10 seed, 5 folds evaluation) is created. This is based on results originating from sampling $1e6$ times from the estimated distributions.

The probabilities in the different CDFs show the added value of multiple samples when doing network evaluations. Using 3 seeds, 5 folds evaluations, leads to higher correlations in this situation, and therefore higher probabilities of finding the best network, at least if the seeds and folds used in the evaluation are also part of the seeds and folds used to validate the network. The only issue is that it is not good practice to validate the results with the same seeds and folds when evaluating stochastic algorithms. How this changes performance estimation is covered in Section 4.5.

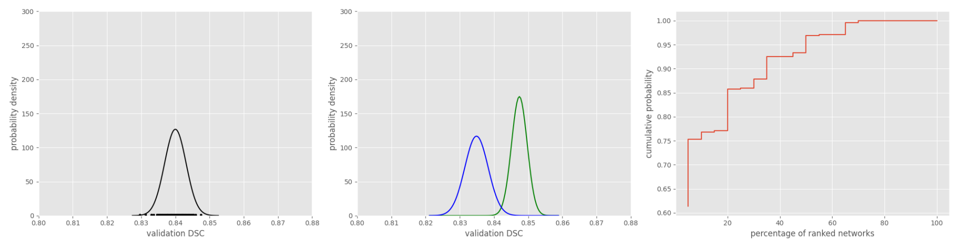
4.5. NAS with noise

In the previous sections, experiments show that there is noise in performance estimation, and that it affects the ranking of networks. The differences in rankings could influence the performance of a NAS algorithm.

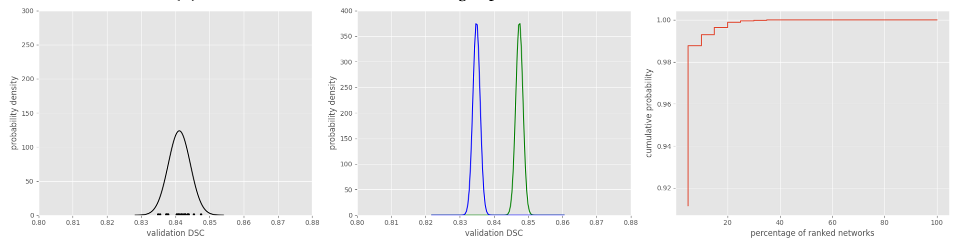
To observe how noise propagates through NAS, an experiment with a Local Search algorithm was done. LS has been shown to be a strong baseline for NAS [36]. Specifically, a first-improvement approach with a variable neighbourhood of 1 is used. The pseudocode for this algorithm can be seen in Algorithm 3. The



(a) 1 seed, 1 fold evaluations, average Spearman correlation coefficient: 0.43.



(b) 1 seed, 5 folds evaluations, average Spearman correlation coefficient: 0.80.



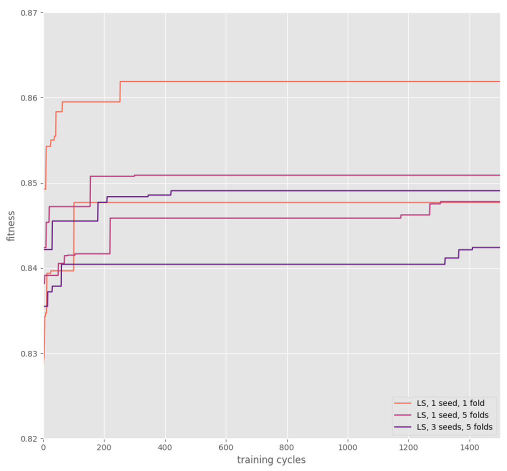
(c) 3 seeds, 5 folds evaluations, average Spearman correlation coefficient: 0.92.

Figure 4.8: *On the left, distributions are shown for the evaluations that can be sampled. For 1 seed, 1 fold evaluations, the performance values are split per fold. This is because the network evaluations to compare will all be sampled from the same fold. For the 1 seed, 5 folds and 3 seeds, 5 folds evaluations, the performance is averaged over 5 folds giving a single distribution, only differing in amounts of seeds used for an evaluation. In the center, the distributions are shown for a 1 seed, 1 fold evaluation for different possible folds, a 1 seed, 5 folds evaluation, and a 3 seeds, 5 folds evaluation, of the best and worst network when evaluating all 10 seeds and 5 folds. Here, one can clearly see that the probability of the worst network being sampled as better than the best network decreases as more samples are taken. On the right, cumulative distribution functions (CDFs) are given for the probability of the best found network being in the top X percentage of ranked networks. For example, the best network found using a 1 seed, 1 fold evaluation of 20 networks, has $\approx 78\%$ chance of being one of the top 40% of networks. For 3 seeds, 5 folds evaluations, the probability of finding the best network as the best network is $\approx 99\%$. The Spearman correlation coefficients given in the captions are taken between the 10 seeds, 5 folds average performance rank, and the average rank of all possible possibilities of the different evaluation methods.*

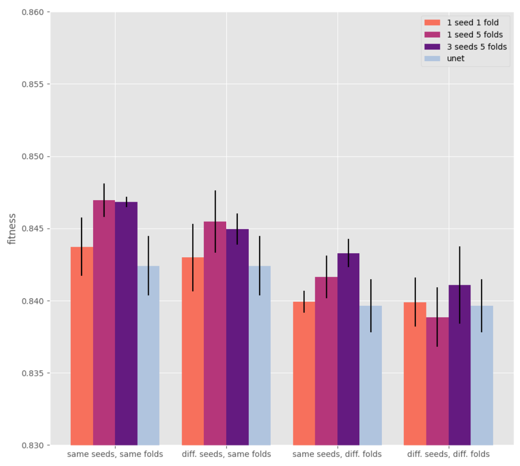
AMC dataset with the same simplified version of MB-NAS, used throughout this chapter, was used for the experiment. The algorithm was run with a 1500 network evaluation sample budget. Three different evaluation methods were used, 1 seed, 1 fold evaluations, 1 seed, 5 folds evaluations, and 3 seeds, 5 folds evaluations. Every network evaluation for 1 seed, 1 fold, uses 1 network evaluation sample, every 1 seed, 5 folds evaluation uses 5 evaluation samples, and 3 seeds, 5 folds uses 15 evaluation samples. The different amount of noise per method will give a sense as to whether noise reduction is more important than exploration, due to limited budget. The elite networks throughout the search were validated in four different ways: same range of seeds and same folds, same range of seeds and different folds, different seeds and same folds, and different seeds and different folds. This performance after validation is shown in Figure 4.9.

The performance increase over U-Net, of the elite networks after validation is shown in Table 4.1. These values are side by side the Spearman rank correlation coefficient and the Pearson correlation coefficient calculated using the networks found and evaluated during NAS, and the same networks after validation. The Pearson correlation coefficient does not look at specific rankings as its Spearman counterpart, only evaluating how much the performance values per network are correlated. To calculate the Pearson correlation coefficient the following equation is used:

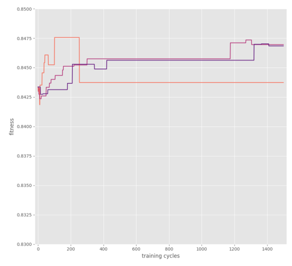
$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$



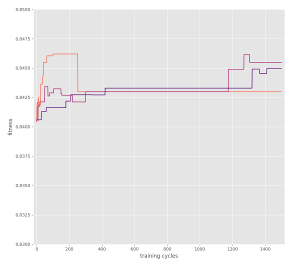
(a) NAS as seen by algorithm (2 runs).



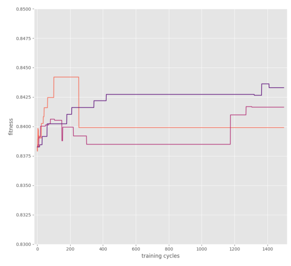
(b) Performance of different evaluation methods on NAS elite compared to simple U-Net performance, for different validation practices.



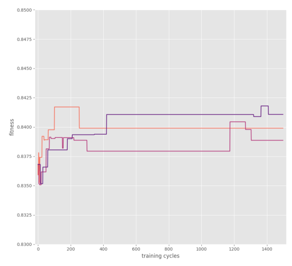
(c) 5 seeds, 5 folds validation.



(d) 5 seeds, 5 folds validation different seeds.



(e) 5 seeds, 5 folds validation different folds.



(f) 5 seeds, 5 folds validation different seeds, different folds.

Figure 4.9: *In 4.9a, the elite performances as seen by the algorithm are shown. Two runs of each evaluation method were done. This unfortunately is too low for statistical significance. The runs can also not be compared as different data splits were used, something that can change the performance score. However, indications can be seen of the deterioration of the search runs as the correlation with the original seeds and folds used during NAS decreases. Also noteworthy is that the average performance of the 5-fold split in 4.9e and 4.9f is lower than for 4.9c - 4.9d. This is also shown in Figure 4.4, where this lower performance is shown for the first data split. From Table 4.1, correlation values can be compared to the bar graph from 4.9b, to see that higher correlations tend to lead to better results. The performance increase is very limited however, as the search space used is quite simple and not very different form U-Net.*

where ρ is the Pearson correlation coefficient, cov is the covariance, and σ is the standard deviation. It can be observed that higher correlation coefficients indicate a higher performance increase.

The results of the experiment suggest that the 3 seeds, 5 folds evaluations seem to overcome issues with noise slightly, having a more stable performance than NAS runs with shorter evaluation methods and more exploration budget. Its performance however is still deteriorated after validation on a different data split, when comparing the performance increase shown in the initial NAS results. The low rank correlation between different data splits, seems to also deteriorate the correlation such that even when increasing the amount of seeds, the correlation does not increase as much as when using the same data split. Because of this, increasing only the number of seeds on the same data split, will stop having the desired effect of increasing correlation, as this will strengthen the correlation with a different data split.

4.6. Validation runs and outperforming SOTA

As validation should be done with different seeds and a different data split, and should not be replaced with the alternate methods shown above[1], it is necessary to realise that very expensive performance estimation

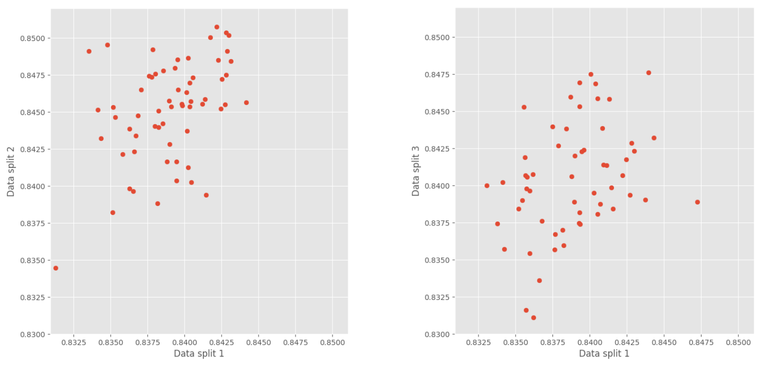
[1]In none of the found literature are multiple data splits validated. The networks are generally evaluated as trained for a certain seed, and evaluated on the same folds [3, 51, 55] when doing cross-validation. This appears like optimising towards your validation problem, something that should be avoided. The best practice seems to be using a separate test set, where in most literature, the network performance for a certain seed and fold is ensembled with the networks trained for different folds in the data split, and then used to measure performance on the test set [4, 55]. As shown in Figure 4.4, this test set can have vastly different performance ranges and noise levels than the training set. Using a test set also decreases the size of the dataset even further, which is undesirable.

Table 4.1: Average Spearman rank correlation coefficients, average Pearson correlation coefficients, and average performance increases (PI) over U-Net, for different performance estimation methods.

Performance estimation method	Correlation with validation set and performance increase compared to U-Net											
	Same seeds, same folds			Diff. seeds, same folds			Same seeds, diff. folds			Diff. seeds, diff. folds		
	Spearman	Pearson	PI	Spearman	Pearson	PI	Spearman	Pearson	PI	Spearman	Pearson	PI
1 seeds, 1 fold	0.650	0.787	0.002	0.611	0.752	0.001	0.543	0.700	0.000	0.610	0.702	0.000
1 seed, 5 folds	0.733	0.880	0.005	0.610	0.878	0.003	0.713	0.773	0.003	0.605	0.720	−0.001
3 seeds, 5 folds	0.814	0.914	0.005	0.724	0.851	0.003	0.752	0.859	0.003	0.627	0.776	0.001

will be needed to reduce the noise of NAS such that algorithms can work as expected on the AMC dataset. As seen from Table 4.1, the outcome of NAS with noise levels found in 3 seeds, 5 folds performance estimation for the used search space and the AMC dataset is still not going to have a large chance at finding a top 10% network in the search space, and increase performance robustly. The correlations between evaluated network rank and validation rank are still relatively low at 0.627 for the Spearman rank correlation coefficient and 0.776 for the Pearson correlation coefficient (see Table 4.1). Linearly interpolating the probabilities from examples in Figures 4.8a and 4.8b indicates a probability of around 0.4. If this cannot be increased between NAS evaluation and validation, NAS will not be able to traverse the search space effectively.

To make matters worse, when calculating the Pearson correlation coefficient between the network performances found by NAS and validated on different data splits, these values are low. This indicates that using different data splits will always cause trouble when doing validation, having to not only use multiple seeds, but also multiple data splits to estimate performance robustly. The correlations of the network performances can be seen in Figure 4.10.



(a) Data split 1 vs data split 2, $\rho = 0.61$.

(b) Data split 1 vs data split 3, $\rho = 0.48$.

Figure 4.10: *Pearson correlation of network performances on two different data splits of the AMC dataset.*

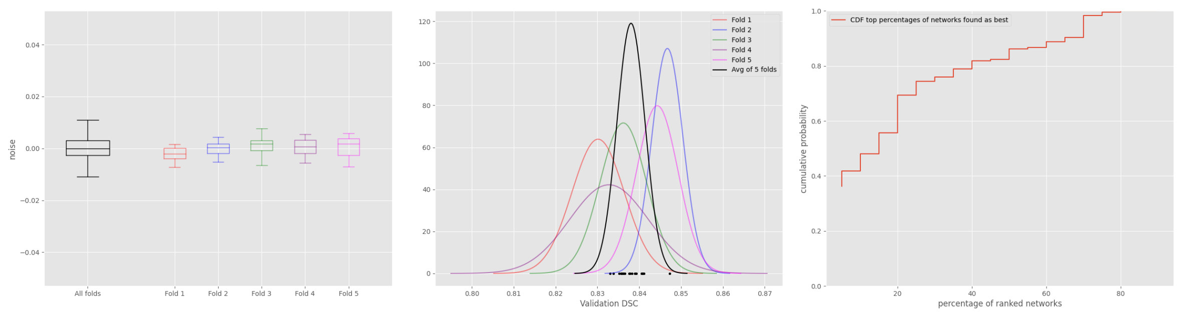
4.7. Other tasks and possible improvements

The findings of the AMC dataset are not necessarily generalisable for all datasets. Therefore, the other datasets are also evaluated for noise, to see if performance estimation can be cheaper for these other tasks.

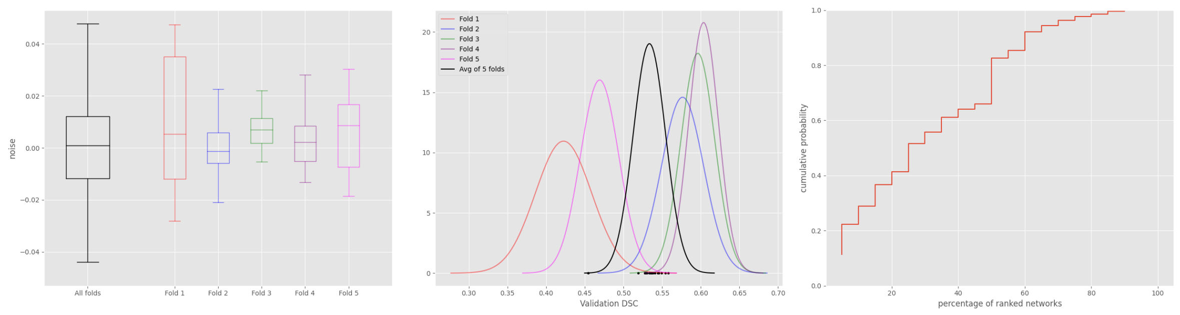
4.7.1. Other datasets

In Figure 4.11, the noise levels and performance distribution of the AMC Prostate, MSD Prostate and MSD Spleen dataset are shown. As can be seen, the MSD Prostate dataset, although noisier per seed, has a larger distribution of different performances. The MSD Spleen dataset has very little noise but a smaller range of performances. A reason for this could be the different levels in interpatient variation between datasets. The MSD Spleen dataset shows little difference between patients in comparison to the MSD Prostate. When looking at the modelled CDFs, the MSD Prostate dataset seems to have the lowest probability of finding a top percentage network.

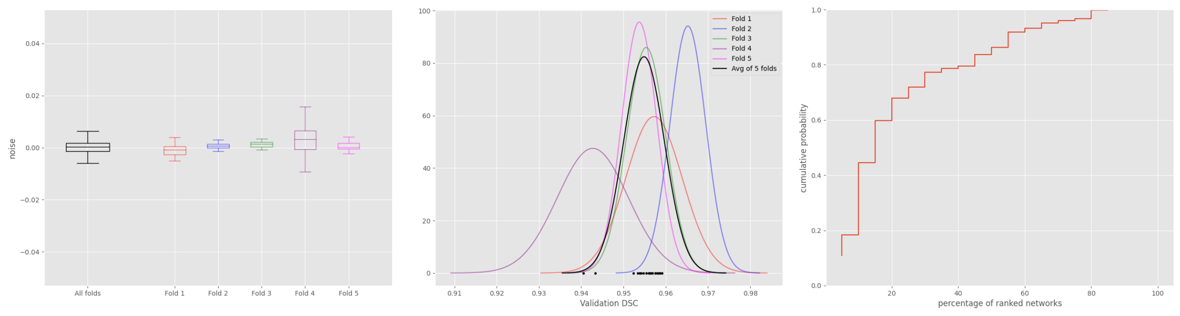
The figure shows that medical image segmentation tasks contain different levels of noise, and therefore



(a) Noise and network performance distribution for AMC Prostate.



(b) Noise and network performance distribution for MSD Prostate.



(c) Noise and network performance distribution for MSD Spleen.

Figure 4.11: *The noise levels and network performance distributions of the three datasets in this thesis. The datasets seem to be similar in ranking ability. However, due to the cluster of network performances being closer together, the CDFs show that the probability of finding a top percentile network is lower, given the 20 networks evaluated. Note that the axes in the subplots are not equal to increase visibility.*

NAS requires different performance estimation methods for different tasks, with more or less samples based on the noise and distribution of performances in a search space. However, it is not conclusive as to which dataset is better suited for NAS. It actually shows that in terms of network ranking, these datasets are quite similar. The relative noise compared to the performance spread is important when it comes to ranking, not absolute noise levels. This means that different levels of noise can be acceptable for NAS for different datasets. By increasing the number of seeds and folds for all these datasets, the networks can be ranked better, as shown in Figure 4.8. However, as seen from the AMC dataset, NAS will only work if the correlation between data splits used for training and for validation is high as well.

4.7.2. Evaluating a dataset before NAS

To finalise this chapter, a tentative solution is put forward that evaluates the correlation between possible training and validation data splits, to make sure the correlation is high enough such that the results hold up after validation, regardless of noise.

The suggested method for the number of data splits to use is the following. With a simple U-Net architecture, or another baseline network, the correlation between dataset splits can be measured. Here, U-Net is trained using 10 seeds on 5 different data splits. The Pearson correlation between data splits will indicate how many different k-fold splits you need to evaluate and what the correlation is that can be achieved robustly. By trying to find a Pearson correlation higher than approximately 0.8 (see Table 4.1 to see why 0.8 was chosen as an initial threshold) between the training data split(s) and the evaluation data split(s), the probability that the NAS algorithm will find a network that performs similarly on the validation dataset, and being one of the top networks, is high.

For the AMC dataset a Pearson correlation 0f 0.82 is found when using two data splits for training and two for validation, in this case split 1 and 4, and 2 and 3, respectively. This can be seen in Figure 4.12. The correlation when different networks are used, is expected to only get larger due to actual architectural differences.

Once correlated data splits are found, the relative noise of the estimation can be decreased by increasing the number of seeds to evaluate the network with. This will lead to more accurate network rankings.

It could be the case that correlations are never high enough for a search algorithm to rank networks correctly, or that network evaluations become too costly for NAS to be feasible. For these datasets, NAS is not recommendable.

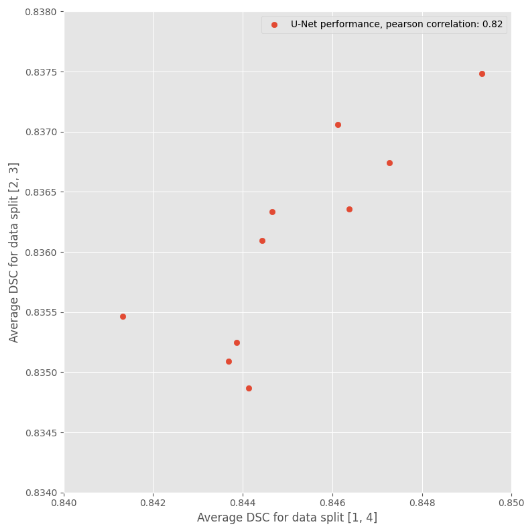


Figure 4.12: *The average network scores for data split 1 and 4 vs. 2 and 3. The different networks are differently seeded U-Net architectures. Here a Pearson correlation of 0.82 is found.*

4.8. Conclusions

Performance estimation for NAS for medical image segmentation is more complex and susceptible to noise than for image classification. Small datasets, inter-patient variability, and intra- and inter-physician variability, can lead to different DNN performance values for different folds and data splits. In addition to this, the noise inherent to the stochastic mechanisms in DNNs add another layer of noise. Because of this low correlations can be seen between different evaluations of the same network. Predicting network performance with metrics from early stopping, which seems to work on classification tasks such as CIFAR, does not work with medical image segmentation tasks due to low correlations with actual performance, for the AMC dataset. To find a network that robustly improves on other networks, it is important to perform NAS such that the correlation between the evaluated network performance and the validation network performance is high, without using the same seeds and k-fold split. For this repeated network evaluations, and high correlations between dataset splits are needed. By increasing the amount of samples in a network evaluation, and increasing the different amount of dataset splits seen by the NAS algorithm, the more robust performance of the found network will be, and the higher the probability of the NAS algorithm finding a top ranked network in your search space.

By using the proposed dataset analysis method, which evaluates the correlation between network performance on different data splits, one can increase the probability of NAS performing as expected. After deciding on what data splits to use, a decision can be made to further reduce noise using multiple evaluations to estimate network performance robustly.

5 Search Space

This chapter describes the design of a novel search space for medical image segmentation. It describes what the goals are, what has already been done, and how this work improves on the research already published. It proceeds by comparing different search spaces on two public datasets. The networks found using NAS are also compared to SotA hand-crafted networks.

5.1. Learning from SotA

In Section 2.2.3 and 2.2.7, the endless possibilities of DNN design are described. For NAS to work effectively, it is necessary to constrain these possibilities such that the search algorithms, that usually have a low evaluation budget[1], can find high-performance networks efficiently. It is also important to leave enough room for the search algorithms to find new networks, as well as include known SotA architectures in the search space. The goal is to find the architectural choices that can influence performance (such that different tasks need different configurations), and to include a large range of possibilities in these dimensions. As the SotA architectures are included, the resulting architectures should not under-perform in comparison to these SotA hand-crafted networks. In order to assess what architecture choices could influence network performance, and to know what the current performance levels are, it is important to look at SotA networks to find trends. Luckily, multiple competitions have been held to evaluate the performance of different networks and training approaches. [1, 15, 22, 26]. A very recent one is the Medical Segmentation Decathlon (MSD) [1], as discussed in Chapter 3.

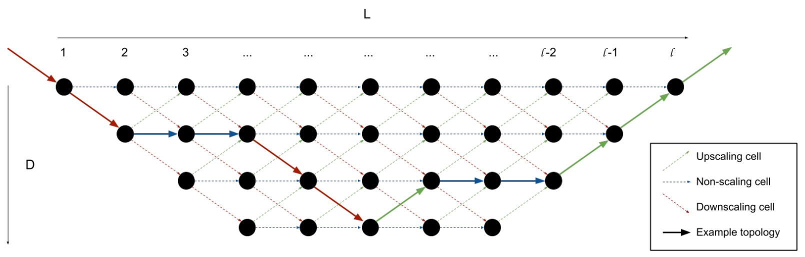
At the top of the leaderboard of the MSD, multiple teams use a U-Net backbone [1]. In particular, the best performing two teams - nnU-Net, holding the first position, and K.A.V.athlon, holding second - both use a variation of U-Net to obtain these high level performances. This underlines the fact that U-Net-like architectures can achieve great results for medical image segmentation tasks. However, it is crucial to also see that both nnU-Net [20] and K.A.V.athlon used methods to vary these U-Net-like network architectures based on dataset properties. nnU-Net uses ”a fully automated dynamic adaptation of the segmentation pipeline, done independently for each task in the MSD, based on an analysis of the respective training dataset” [1]. So, not only were network topologies adapted, but the pre- and post-processing methods were also determined based on these properties. It is important to notice that the nnU-Net pipeline scales the number of down- and up-scaling cells[2] in the network based on the dataset resolution. Using out-of-the-box U-Net does not perform as well as these task-specific network architectures [20]. This makes the case for topology level search interesting, where freedom is given such that the resolution and number of channels can be varied.

5.2. Search spaces in literature

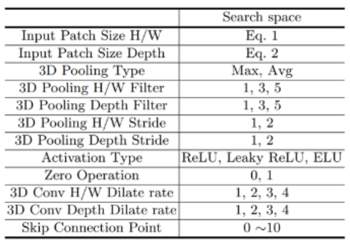
Other research has also concluded that adapting the amount of down- and up-scaling cells, can increase the performance on certain tasks [3, 27, 29, 55]. In NAS research, when a search algorithm can adapt these

[1]This low budget is a result of the computationally expensive evaluation of a DNN, that requires training the network until saturation.
[2]A down-scaling or up-scaling cell is a network cell that changes the feature map dimensions. For down-scaling the resolution of the image is decreased and the amount of channels increased, and vice-versa for up-scaling.

aspects of a network, this is known as topology level, or macro-level search, as described in Section 2.2.7. This topology level looks at three main categories of network architecture: (1) the dimensions of the feature maps - networks that have more channels and contain more down-scaling operations will have the capacity to capture more elaborate features, and are easier to train [45], however extracting and localising the features, i.e. tracing the features back to the respective pixels/voxels, could be harder; (2) the density of the network, i.e. the skip-connections used - these skip-connections could alleviate the vanishing-gradient problem, strengthen feature propagation and encourage feature reuse [18]; (3) the depth of the network, i.e., the amount of layers used - increasing depth could make it easier to capture complex features. This capacity of learning more complex features saturates after a certain level of depth [14]. As making a network deeper increases the search space size exponentially, this parameter is often fixed in NAS [27, 51, 55]. In Figure 5.1 a few examples are given of discrete macro-level search spaces. The elements of topology search in these examples are all based on the structure of U-Net. No matter the outcome, the network will consist of a combination of down-scaling, non-scaling and up-scaling layers. This allows the feature maps to be condensed to a lower resolution, enabling the network to learn more and larger features, and then extracting the learned features and scaling up the resolution again. The non-scaling layers, that do not change the feature map dimensions, do allow learning more intricate features at a certain resolution. Possible skip-connections allow less complex feature map weights to be concatenated with the output weights of up-scaling layers.

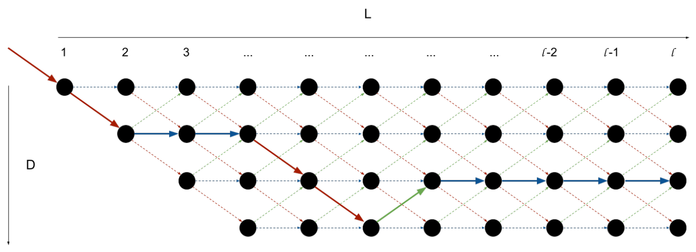


(a) Macro search space C2F [55].



	Search space
Input Patch Size H/W	Eq. 1
Input Patch Size Depth	Eq. 2
3D Pooling Type	Max, Avg
3D Pooling H/W Filter	1, 3, 5
3D Pooling Depth Filter	1, 3, 5
3D Pooling H/W Stride	1, 2
3D Pooling Depth Stride	1, 2
Activation Type	ReLU, Leaky ReLU, ELU
Zero Operation	0, 1
3D Conv H/W Dilate rate	1, 2, 3, 4
3D Conv Depth Dilate rate	1, 2, 3, 4
Skip Connection Point	0 ∼10

(b) Macro search space RONAS [3].

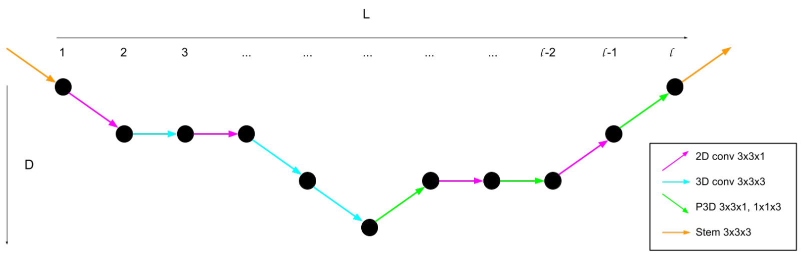


(c) Macro search space AutoDeepLab [27].

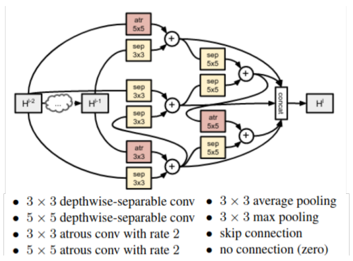
Figure 5.1: *Examples of macro search spaces used for medical image segmentation as can be found in literature. The Coarse2Fine approach in 5.1a and the RONAS approach in 5.1b are examples of discrete search spaces. The RONAS approach also contains elements that would be considered micro-search in this thesis, e.g. activation layers, pooling type used. The AutoDeepLab approach, seen in 5.1c uses continuous relaxation and the finds the path from input to output with the highest weights. In examples 5.1a and 5.1c, L is the number of cells in the network, and D is the sum of amount of down-scaling operations used prior to the given position.*

Next to the topology search space, a common search space in NAS for all tasks, is a micro-level, or cell level, search [27, 51, 55]. This search space looks at the possibilities within a given topology, so within the cells, of a network. As the input and output feature maps configurations, i.e., resolution and number of channels, are already fixed, the degrees of freedom for this search level are the operations used, their parameters, and the order of these operations. Examples of such a search space could include convolution types and kernel sizes, pooling sizes and types, activation layers, normalisation layers, and the order in which all these operations are used. As the options can be very extensive for just one cell, and the number of cells in a DNN that can all be optimised differently can be relatively large, this search space needs to be constrained. This avoids it becoming too large to navigate. A common way to achieve this, in both discrete and continuous search spaces, is by repeating a certain possible cell-structure throughout the network. An example of such an optimised cell that is repeated throughout the network is given in Figure 5.2b [27]. Alongside is an example of a simpler micro-level search space from the Coarse-to-Fine approach [55].

Oftentimes, these search spaces are combined into a bi-level search space [27, 51, 55]. These contain



(a) Micro search space C2F [55].


• 3 × 3 depthwise-separable conv
• 5 × 5 depthwise-separable conv
• 3 × 3 atrous conv with rate 2
• 5 × 5 atrous conv with rate 2
• 3 × 3 average pooling
• 3 × 3 max pooling
• skip connection
• no connection (zero)

(b) Micro search space AutoDeepLab [27].

Figure 5.2: *Two examples of micro search spaces from literature. The Coarse-to-Fine (C2F) cell search space seen in 5.2a is a discrete search space that includes only different convolution types at every position in the network. The AutoDeepLab search space is based on continuous relaxation, and the cell structure repeated at every edge in 5.1c [27].*

both a macro-level and a micro-level search space. These can either be searched consecutively, or simultaneously. Consecutive search avoids the search space increasing in size multiplicatively. Simultaneous search increases the search size, but allows for possible interactions between macro- and micro-level search spaces to be exploited. Simultaneous bi-level search has only been performed using continuous relaxation during the evaluation of a super-network [51].

Continuous relaxation based methods [27, 51], use one very large super-network that combines all possible networks in a search space into one. During the training of this network, the elements that contribute least to the output are gradually removed. This can be done by including a separate gradient descent algorithm to find what elements to remove, and including it after the existing one that is used to optimise the weights. By continuously removing components until a certain threshold, you are left with a network that fulfills your goals, e.g. in amount of total parameters.

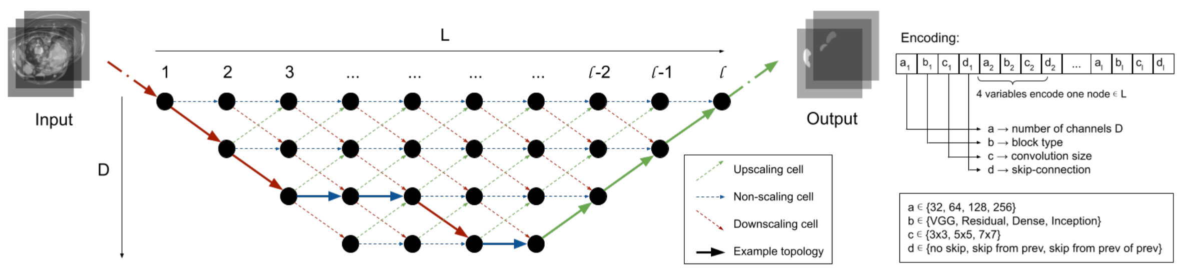
This approach was not chosen in this thesis as there are alleged issues with weight sharing methods not representing actual network performance [54], as well as that training super-networks is very computationally expensive.

Bi-level search, consecutive or simultaneous, has the possibility to create highly tailored networks, due to the large amount of degrees of freedom. This is what makes it so attractive for NAS. However, knowing that performance estimation of DNNs is notoriously expensive computationally, it is necessary to constrain a bi-level search space in size. Not doing so might lead to deterioration of results if algorithms cannot effectively traverse the search space. In an attempt to create a search space that is more flexible than previous literature, a bi-level search space with a different approach on the micro-level is proposed. Instead of repeating the same cell structure, various cell structures are allowed in a network. This gives the network the freedom to use cells with different properties at different locations. The intuition is that this will give more possibilities in the features that can be captured. Further, the amount of cell configurations can be restricted, by using existing knowledge about networks with a strong performance on classification tasks. By using a pre-selected pool of configurations (which are taken from advanced well-known classification networks [14, 18, 44]) instead of searching for a cell configuration from scratch, the number of possibilities per location in the network is restricted. This will help avoid the explosive growth of the search space caused by searching the configuration of each cell from scratch. On the topology level, skip-connections can be added from the previous two cells with the same feature maps dimension, increasing flexibility compared to other discrete macro search spaces. Finally, in contrast to recent research [51, 55], where topology level search is followed by cell level search, both topology as well as the configuration of each cell is searched simultaneously. This allows for possible interaction between the topology-level search and cell-level. The combination of these improvements will further be referenced to as Mixed-Block NAS (**MB-NAS**).

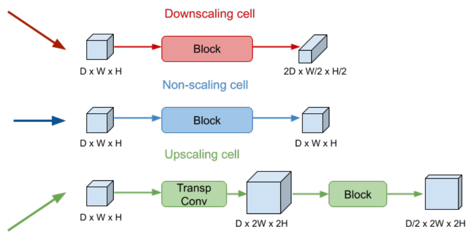
5.3. Mixed-block NAS

The search space proposed in this thesis can be seen in Figure 5.3. It is searched in a discrete way. This avoids weight sharing and circumvents the computational cost of training a super-network.

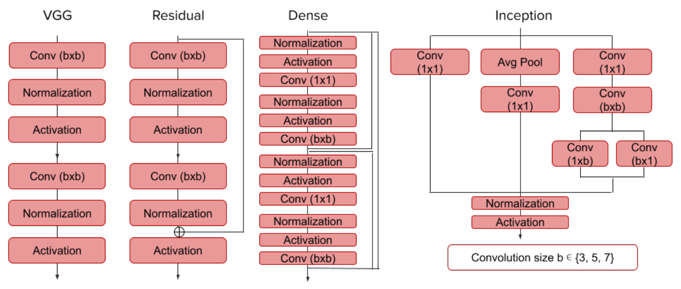
At **topology-level** (Figure 5.3a), the search space contains all possible network architectures resulting from varying connections between cells of different types. Three possibilities for a cell are considered (Figure 5.3b): a down-scaling, an up-scaling, or a non-scaling cell; similar to [27, 51, 55]. A down-scaling cell means that the input image resolution is halved, while the number of channels is doubled. Up-scaling is the



(a) Topology-level search and encoding.



(b) Cell types.



(c) Block types.

Figure 5.3: *All possible topologies in the search space are shown in 5.3a. A possible topology consists of one arrow per layer from left to right. This is a combination of down-scaling, non-scaling, and up-scaling cells, as indicated by the red, blue, and green arrows, respectively. The non-dashed, bold line is an example of a possible topology. The number of layers for the experiments presented below is* $l = 10$ *and the base number of features* $d = 32$*. In 5.3b the structure of the cells is illustrated, alongside 5.3c where the possible blocks (VGG, Residual, Dense, Inception) that can be placed in the cells are schown. The convolution sizes can be either* 3×3, 5×5, *or* 7×7, *based on the encoding.*

opposite operation, i.e., doubling the resolution and halving the number of channels, while a non-scaling cell changes neither the resolution nor the number of channels in the feature maps. The input to a cell can be either the feature maps from the preceding cell (no skip-connection), or the feature maps of the preceding cell concatenated with the feature maps from one of two cells prior to the preceding cell, that have the same feature maps spatial dimensions (a single skip-connection).

At **cell-level**, different configurations of each cell are searched. The configuration of a cell is encoded by two variables: the type of block[3] within the cell, and the convolutional kernel size within the block. Instead of searching for the topology of the blocks within a cell, which would make the search space incredibly large, predefined blocks derived from previously SotA architectures for classification are used (Figure 5.3c). Here, VGG blocks which are standard in U-Net, as well as Residual blocks, Dense blocks and Inception blocks, are considered. In this way, the search space includes different cell configurations at every edge (instead of repeating one throughout the network) while preventing the further growth of the search space.

The network architecture is represented by connections between a fixed number of nodes (fixed to 10 in the experiments in this thesis). Each node l (Figure 5.3a) is represented by 4 categorical variables: a_l = number of channels, b_l = the block type, c_l = convolution size, and d_l = skip-connection source. The cell type is derived by the difference in number of channels between two nodes. The topology of the neural network is encoded by variables a_l and d_l at each node. The cell-level search is represented by variables b_l and c_l at each node. The maximum number of channels is constrained to four times the initial number of channels in these experiments. Note that the standard U-Net shape is included in the topology-level search space. The resulting search space contains $1.14 * 10^{18}$ possible networks.

5.4. Experimental setup

In order to evaluate the potential of the created search space, a performance estimation method and a search algorithm must be chosen to run experiments with.

Local Search (LS) is used as search algorithm for the experiments in this chapter, as it has been shown to be a strong baseline for NAS [36]. Specifically, a first-improvement approach with a variable neighbourhood

[3]a block is an organised structure consisting of multiple convolution and normalisation layers, as well as activation functions

of 1 is used. This means that starting from a certain solution, each possibility is iterated over for every variable. The variables are chosen in a random order. When a solution is better than the previous best solution, it becomes the new best solution. Once all variables are considered, the process is restarted, until no improvements can be found. The pseudocode for this algorithm can be seen in Algorithm 3.

As some network architectures are infeasible when generating random encodings, a decision needs to be made on whether to reject or repair these architectures. In this LS algorithm, infeasible architectures are rejected and removed from the possibilities.

Due to limited computational resources, the NAS was run for only 150 network evaluations. Each network was evaluated during NAS using the validation Dice score averaged over the last 20% of training epochs. Averaging over the last epochs, helps to reduce the noise in performance values, which is shown in Section 4.2. The number of epochs (100 epochs for prostate, 50 for spleen) was decided such that saturation was ensured based on preliminary runs. An average score of 5-fold patient-level cross-validation, repeated for 3 network initialisations, was used as performance estimator. The use of multiple folds decreases noise caused by data splits, while using multiple seeds reduces the noise caused by the network optimisation. The decreased noise makes for more reliable information to be passed to the search algorithm. More information on this approach is given in Chapter 4.

For network training, the ADAM optimiser [23] was used with a learning rate 10^{-3} and polynomial decay with an exponent of 0.9. The loss function was foreground Soft Dice. The batch size was 32 and input image size was 128x128. The data was augmented using scaling, shifting, rotating, flipping, and brightness adjustment.

The experiments are used not only to compare the obtained networks from the **MB-NAS** search space against SotA networks, but also to evaluate the search space against the following alternative approaches. In the first considered alternative search space (**Macro-NAS**) only topology search is performed. All block types are fixed to be standard U-Net blocks (VGG). The second considered search space (**Micro-NAS**) has the U-Net topology, where only the block type and convolution sizes are subject to the search. In the third alternative search space (**Bilevel-NAS**), a sequential bi-level approach was used which means that first the topology-level is searched, and then the convolution size, similar to the Coarse-to-Fine approach [55].

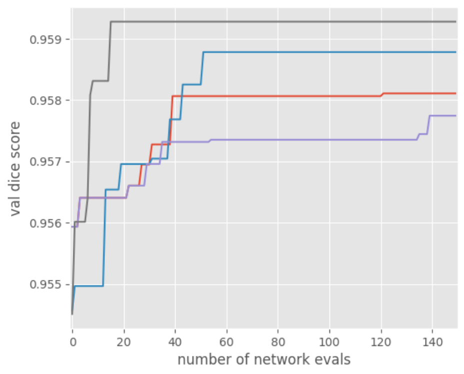
The best networks found by the proposed search spaces were evaluated against two hand-crafted neural network architectures: standard **U-Net**, and U-Net with a ResNet-50 encoder (**ResU-Net**). The chosen size of these networks, is the size advised by the nnU-Net guidelines. This is larger than the NAS topologies can get. The implementations are taken from the Pytorch Segmentation Models library [50].

5.5. Results

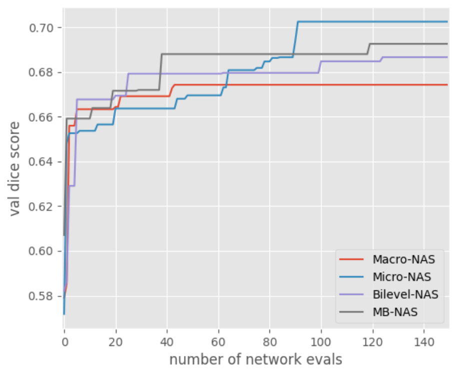
The progress made by the LS algorithm in the different search spaces (**MB-NAS**, **Macro-NAS**, **Micro-NAS**, **Bilevel-NAS**) can be seen in Figure 5.4a and 5.4b. It shows the performance value of the best network found by the search algorithm. The architectures of the best networks from MB-NAS are visualised in Figure 5.5. The varying performances of the best networks from different NAS approaches, **U-Net**, and **ResU-Net** is summarised in Table 5.1. Example segmentations for qualitative comparison are given in Figure 5.6.

Table 5.1: Performance values of U-Net, ResU-Net, and the best networks from different NAS approaches. The metrics are validated by averaging over 5 seeds on a 5-fold cross-validation. Standard deviations are calculated based on single seed scores on all 5-folds. Best values in each column are highlighted in bold. DSC: Dice-Sorensen similarity coefficient, HD: Hausdorff distance (95% cutoff), SD: Surface Dice (2mm threshold), MMAC: Mega Multiply–ACcumulate operations, Params: number of parameters, $\times 10^6$.

Model	Prostate dataset					Spleen dataset				
	DSC	HD	SD	MMAC	Params	DSC	HD	SD	MMAC	Params
U-Net	0.6702 ± 0.004	8.833	0.6046	302	18.4	0.9578 ± 0.002	1.412	0.9174	302	18.4
ResU-Net	0.6580 ± 0.004	9.441	0.5705	**166**	32.5	0.9464 ± 0.004	1.625	0.9047	**166**	32.5
Macro-NAS	0.6593 ± 0.004	8.606	0.5977	256	3.39	0.9566 ± 0.001	1.467	0.9167	255	**3.39**
Micro-NAS	$\mathbf{0.6796 \pm 0.007}$	**8.394**	**0.6203**	1,295	22.7	0.9567 ± 0.002	1.388	0.9177	795	2.83
Bilevel-NAS	0.6702 ± 0.005	8.492	0.6134	414	6.77	0.9553 ± 0.001	1.449	0.9145	415	6.78
MB-NAS	0.6760 ± 0.010	8.419	0.6192	644	**3.04**	$\mathbf{0.9592 \pm 0.002}$	**1.385**	**0.9189**	1,294	22.7



(a) NAS spleen.



(b) NAS prostate.

Figure 5.4: *The progress of the LS for the (a) spleen and (b) prostate datasets, respectively. As can be seen from the plots,* ***Micro-NAS*** *and* ***MB-NAS*** *perform better than* ***Macro-NAS*** *and* ***Bilevel-NAS*** *on both datasets. The performances at the end of the plot also differ from the performances in Table 5.1 as the performance in the Table is validated over different seeds than used by the algorithm.*

5.5.1. Spleen dataset

The best performing network on the MSD Spleen dataset was found by the proposed search space, MB-NAS. In Table 5.1, one can see that this network shows the best performance by all three considered performance metrics. Note that the topology (Figure 5.5, left) is quite different from the standard U-Net. The network is shallower, potentially indicating that smaller, more intricate features, are more important towards the segmentation of the spleen, than larger, more elaborate ones. This could be due to the relatively fixed location of the spleen in the data, as well as the small size of the organ in relation to the image. Furthermore, blocks of all four types, are included in the architecture, as well as different convolution sizes. This means that both the topology-level and cell-level search space parts were utilised to find this network, an indication towards the performance increase from the designed search space.

5.5.2. Prostate dataset

Different than for the spleen dataset, the best network (Figure 5.5, right) for the MSD Prostate dataset is very similar to the U-Net architecture. This provides additional evidence to the argument that the architecture of the best DNN is task-specific. This also indicates that the U-Net topology is best suited for the underlying task, giving an added advantage to Micro-NAS, wherein different blocks and convolution sizes are searched within U-Net topology. Consequently, Micro-NAS yields the best performance. However, it should be noted that MB-NAS could find the given network or another network with comparable performance, despite the fact that the search was performed for both topology as well as cell configuration.

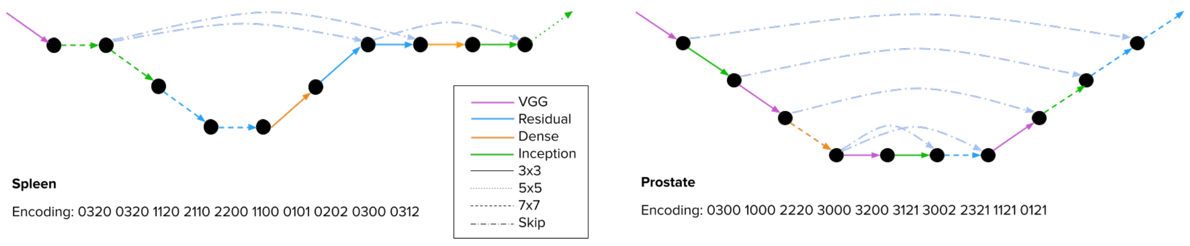
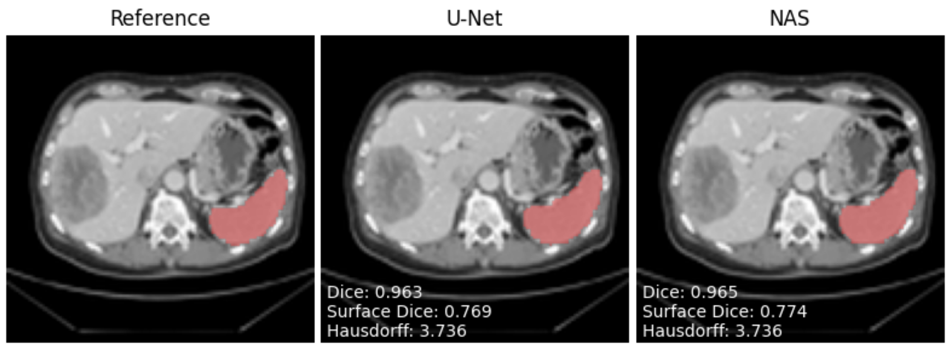

Spleen
Encoding: 0320 0320 1120 2110 2200 1100 0101 0202 0300 0312
VGG
Residual
Dense
Inception
3x3
5x5
7x7
Skip
Prostate
Encoding: 0300 1000 2220 3000 3200 3121 3002 2321 1121 0121

Figure 5.5: *The elite network found by the LS algorithm in the compared search spaces. Left, the network found for the Spleen dataset. Right, the network found for the Prostate dataset. The network for the Spleen is shallower than for the Prostate, potentially indicating that smaller, more intricate features, are more important towards the segmentation of the Spleen, than larger, more elaborate ones. This could be due to the quite fixed location of the spleen in the data as well as its small size. The network found for the Prostate dataset has a similar topology than U-Net. However, the blocks used throughout the network are different than the VGG blocks used in U-Net at many locations, as is the found convolution kernel size. This means that different feature sizes can be learned.*

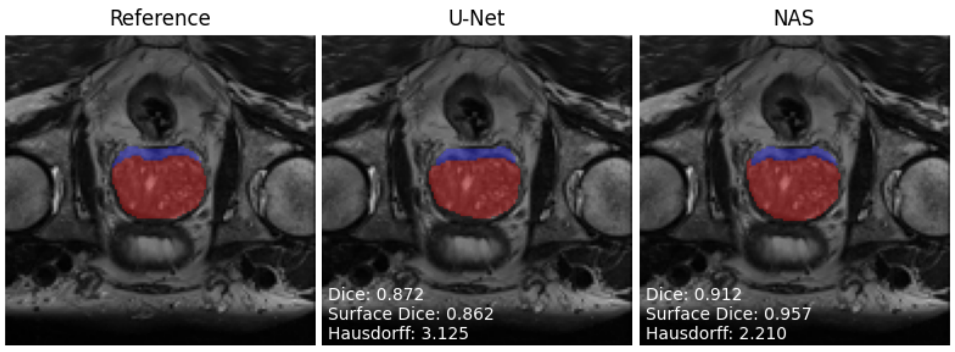
5.5.3. General results

It is worth noting that no block seems to be ultimately preferable to other blocks, i.e. outperforms other blocks at every location of the network. This indicates the advantage of searching for a mixed-block configuration such that different blocks can be used throughout the network. Additionally, the differences in magnitude for the validation Dice-Sorensen Coefficient of poorly performing networks compared to the best networks is much larger for the MSD Prostate dataset, indicating a more difficult task. Further, Table 5.1 shows that the best networks from NAS outperform the manually hand-crafted U-Net and ResU-Net networks, but only slightly, after validation with different seeds.

An important remark to make is that these results are not statistically significant. Only one run of the NAS algorithm was done per dataset due to GPU-time constraints. It would be interesting to observe the performance when repeating this search multiple times. This could lead to a clear conclusion as to how the search spaces compare.



(a) Example segmentations on the spleen dataset.



(b) Example segmentations on the prostate dataset. red: central gland, purple: peripheral zone.

Figure 5.6: *Example segmentations of both the MSD Prostate dataset and MSD Spleen dataset for qualitative comparison.*

5.5.4. Comparison with SotA

It can be argued that the results for the MSD Prostate dataset in Table 5.1 are not at the same level as the SotA results given by e.g., nnU-Net [20]. This is due to higher image resolution used by nnU-Net, additional pre-processing, carefully chosen data augmentations, post-processing, and advanced inference method, that are all different from the setup used in NAS in this thesis. Therefore, the network performance when using the nnU-Net training and evaluation setup, was also evaluated with the found networks from NAS. For MSD Prostate, the 5-fold cross-validation Dice-Sorensen coefficient is $\mathbf{0.7325}$ for MB-NAS vs. $\mathbf{0.7315}$ for nnU-Net. It should be noted that the comparison is done between the network found by NAS, trained in the nnU-Net environment, and a U-Net architecture that is tailored to the data according to the nnU-Net heuristics. The architectures found by NAS were found using entirely different settings. This gives an unfair advantage to nnU-Net. Nevertheless, the architecture found by NAS still performs slightly better to nnU-Net in these settings, which is remarkable. This could indicate that small levels of resolution scaling could be a viable option for NAS, as the found network generalises well to a resolution that is approximately two and a half times larger. For the MSD Spleen dataset, the performance level is quite similar to the results reported by nnU-Net. Nevertheless, the network was validated in this environment too. The 5-fold cross-validation Dice-Sorensen coefficient is **0.9467** for MB-NAS vs. **0.9466** for nnU-Net, again being very comparable. It does show that the more elaborate nnU-Net method does not generalise well to this task as performance scores are lower.

5.6. Conclusions

A novel search space and simultaneous topology- and cell-level search strategy for medical image segmentation NAS was proposed in this chapter. In the cell-level search, existing knowledge from networks with high performance in image classification tasks, i.e. ResNet, DenseNet and InceptionNet, was used to create a pool of possible block configurations. The experiments show the added value of this approach. Unfortunately, due to limited computational resources, only one run of NAS was performed per search space, and for a limited number of network evaluations. Longer experiments with multiple runs would have helped draw more definitive conclusions. Running NAS experiments for more datasets may also provide more insights about the specific characteristics of network architecture design required for good performance across datasets. Overall, the results indicate that further research into search space refinement, allowing to exploit key features of what accounts for good deep learning performance, may yet push the boundaries of what can be achieved with deep neural networks for medical image segmentation.

6 Search Algorithms

This chapter analyses several search algorithms used for NAS. It starts off by elaborating on the existing research introduced in Chapter 2, it then proceeds by explaining what the difficulties are when comparing some different types of algorithms. It follows up with a comparison of 3 algorithms and their performances with different levels of noise. Finally, the chapter ends with some possible improvements and conclusions.

6.1. Algorithm types

Search algorithms most used for NAS are often categorised into Reinforcement Learning (RL), Gradient-based algorithms (GD), Evolutionary Algorithms (EA), and simple first-improvement based search techniques such as Random Search (RS) and Local Search (LS). In this thesis, the focus was on Evolutionary Algorithms and whether they could improve on simple search techniques like RS and LS.

As explained in Chapter 5, gradient-based algorithms are often used for NAS, especially for medical image segmentation. However, they can only be used in continuous search spaces [27, 51], as discrete search spaces do not have gradients readily available. As the search spaces are all discrete in my thesis, Gradient Descent algorithms are left out of comparisons.

6.1.1. Evolutionary Algorithms

Evolutionary algorithms, as introduced in Section 2.4, have been used often in NAS research [4, 12, 55]. This is due to several factors: (1) networks are easily adaptable to encodings used in these algorithms, (2) they lend themselves well to adapt them to incorporate surrogate models, and (3) evolutionary algorithms have the capability of exploiting possible interactions between variables in an encoding, or cells in a network. In this thesis two Evolutionary Algorithms are compared: SGA [16] and P3GOMEA [11]. The pseudocode for these algorithms, along with a description of the working mechanisms is given in Section 6. These algorithms are compared to an LS algorithm, as this was shown to be a strong benchmark [36].

In order to focus on learning interactions between network components, P3GOMEA [11], tries to evolve networks by learning what variables can be best paired in a Family of Subsets. By learning what variables have high Mutual Information, the networks can be altered such that strongly performing combinations are selected together. GOMEA variants have been shown to work well on various optimisation problems [46]. It is for this reason that P3GOMEA is chosen as one of the EAs in this thesis.

Simple Genetic Algorithms are the simplest group of EAs, and a good benchmark to compare P3GOMEA with. Where P3GOMEA tries to learn linkage in the search space, SGA does not keep track of any structures other than the population and the performance of the individuals. By combining parts of strongly performing networks, it tries to find new combinations that might work even better. By using this as a benchmark, potential exploitation of higher-order variable interactions by P3GOMEA could be observed. SGA requires a couple of input parameters, such as selection method, crossover type, and population size. For optimal performance these need to be tuned. For P3GOMEA, no parameters are necessary, which is very convenient for expensive searches where hyperparameter tuning is out of computational budget reach.

In [36], not only is it shown that GOMEA performs very well in a multi-objective NAS for classification networks, but also that LS performs remarkably well at that same task. As the task of NAS for medical image segmentation of clinical data is more complex and noisy than for classification on a benchmark, it is interesting to see if this strong baseline performs equally well for NAS in medical image segmentation.

Due to the large amount of noise in the tasks in medical image segmentation, particularly for the AMC dataset, it is interesting to see how these algorithms deal with noise, and if they manage to overcome the problem that performance estimation creates for medical image segmentation tasks.

6.2. Benchmarks

Search algorithms and approaches are being thoroughly tested on image classification benchmarks, where network performance has been saved for a certain dataset and search space. The datasets used for this are CIFAR-10 and CIFAR-100 [24]. Multiple search spaces have been proposed, all based on DARTS (Differentiable Architecture Search) [28] with restrictions in order to limit the size of the search space. These benchmarks store performances in tabular data, or by using a surrogate trained on a part of the search space to predict the rest. As no benchmarks exist for medical image segmentation, it is impossible to use them for search algorithm comparison, making performance estimation necessary to compare networks.

As also seen in Chapter 4, in the given medical image segmentation task, noise in performance estimations is a lot higher when using common estimation methods, making it much harder for search algorithms to navigate. Unfortunately, that means that the results of search algorithms from other research cannot just be adopted without consideration, and that they need to be compared in this new environment to check whether performance is increased. For this, an experiment was devised to be able to evaluate the performance of the algorithms with different levels of noise, and also varying levels of correlation with the validation evaluation. The expectation is that all algorithms will have trouble learning in the given search space due to the noise and low computational budget. However, if the noise seems to be low enough for P3GOMEA and LS to be able to improve steadily, these algorithms should perform slightly better than SGA.

6.3. Experimental setup

In order to evaluate the search algorithms, a performance estimation method is chosen, and a search space and dataset. For the dataset and search space the choice was made for the simple variant of MB-NAS, and the AMC dataset was chosen, just like in Chapter 4. Network training is done in an identical fashion to the described method in Chapter 5.

6.3.1. Performance estimation

Three performance estimation methods were chosen: 1-seed 1-fold evaluations, 1-seed-5-fold evaluations, and 3-seed 5-fold evaluations. The algorithms were run with a 1500 network evaluation sample budget. Every network evaluation for 1-seed 1-fold uses 1 network evaluation sample, every 1-seed 5-fold evaluation uses 5 evaluation samples, and 3-seed 5-fold uses 15 evaluation samples. Every algorithm was run two times, due to computational constraints. The outcome of these two results were averaged to be able to visualise performances better. These runs were done with identical seeds per algorithm, and on 2 different data splits. As increasing correlation between the performance estimation and validation performance to high levels, as shown in Section 4.7.2, is too computationally expensive, the validation is done in multiple ways to be more similar to the evaluation in the search algorithm. For this 4 approaches are used: (1) same seeds and folds - meaning that the evaluated seeds and folds are all in the validation range, (2) different seeds, same folds - meaning that the random seeds used are all different, but the dataset split and folds are the same, (3) same seeds, different folds, where random seeds used in NAS are also used in validation, but with a different data split, (4) different seeds and folds, such that both random seeds and data split are different. All validation is done with 5-seed 5-fold evaluation of networks. As shown in Chapter 4, if performance estimation is too noisy and contains a correlation that is too low with the proper validation methods, NAS will not lead to expected results. By changing the validation, environments with higher correlations and less noise are simulated. The search algorithms are compared to see if one of the search algorithms manages to outperform the others in these different environments.

6.3.2. Algorithm description

The algorithms used in this experiment are implemented as described in Chapter 2. However, some decisions have to be made on the specific parameters and repair choices for these algorithms.

For the LS algorithm, the implementation as described in Chapter 5 is used. It implements a first-improvement approach with a variable neighbourhood site of 1. It does not repair infeasible networks at it evaluates the feasibility of every encoding change before iterating through them. It therefore always stays in the feasible domain.

For P3GOMEA, the decision was made to repair networks right after a solution is generated. It is then evaluated by the algorithm. The repair operation makes sure the network is feasible by correcting all layers that have jumps larger than doubling or less than halving of the number of input channels. Also, all infeasible skip-connections are removed.

SGA is implemented with a similar approach, except with an additional element of not allowing identical networks in the population. The population size is 16 and 1-point crossover is used. Selection is done with tournaments, with tournament size of 2.

6.4. Results

The performance of these algorithms, in the search environments with the lowest correlation to the validation space, are very similar. This means that the algorithms have just as much trouble/success in finding local optima in the validation search space. What is surprising to see, is that SGA performs very well in the actual search space, having the best performance in all but one run. The results are visualised in Figure 6.1, and also summarised in Table 6.1 along with the validation scores. Here you can see that the performance of every algorithm degrades equally when the correlations decrease, something that is to be expected. For the performance estimation method with 3-seed-5-fold evaluations and higher correlations the algorithms also only have 100 networks that can be evaluated, which is very few considering the size of the search space. It is therefore hard for P3GOMEA to make informed decisions, which is needed to make it perform better than the other two algorithms.

Table 6.1: Performance increase of different search algorithms when using different performance estimation methods with different noise values

Performance estimation method	**Performance increase of elites found by the Search Algorithms compared to U-Net performance**											
	Same seeds, same folds			Diff. seeds, same folds			Same seeds, diff. folds			Diff. seeds, diff. folds		
	LS	P3GOMEA	SGA	LS	P3GOMEA	SGA	LS	P3GOMEA	SGA	LS	P3GOMEA	SGA
1 seed, 1 fold	0.002	0.003	0.004	0.001	0.001	0.003	0.000	0.000	0.003	0.000	0.000	0.000
1 seed, 5 folds	0.005	0.004	0.003	0.003	0.003	0.002	0.002	0.004	0.002	−0.001	0.001	0.000
3 seeds, 5 folds	0.005	0.004	0.005	0.003	0.003	0.004	0.003	0.003	0.005	0.001	0.000	0.001

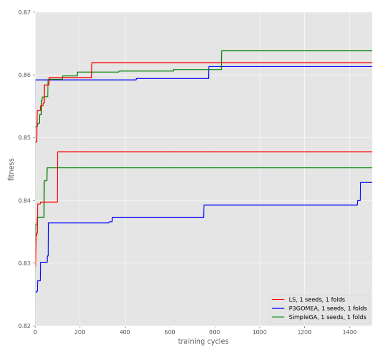
6.5. Discussion

The results indicate that LS and P3GOMEA do not outperform SGA, even performing slightly worse. A possible explanation of this is that both algorithms only take into improvements of single samples of a network, albeit sampled differently. Due to noise, this could 'fool' the algorithm into moving away from a direction with many more potential improvements. Other algorithm types, such as Estimation of Distribution Algorithms (EDAs), could be better at such a task, where noise degrades the reliability of network comparisons. These algorithms search for an optimum by estimating probabilistic models, e.g., Gaussian distributions, of candidate solutions, and do not make greedy decisions. By building up a larger base of information to decide on the best network, EDAs evade the trap caused by Greedy acceptance in high levels of noise. By not discarding valuable information on noise, the probability of finding top networks could be increased.

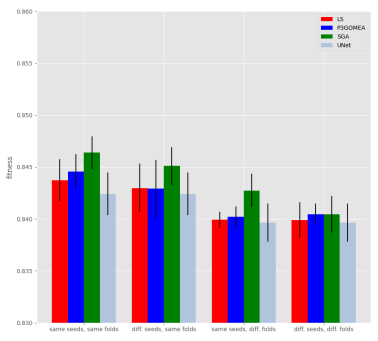
Another factor that could cause the lack of difference between search algorithm performance, is the small evaluation budget. Due to the computational cost of performance estimation, the amount of evaluations that every algorithm can perform is far lower than what these algorithms usually need to converge to an optimum [36, 46]. By increasing this budget, something that was not feasible in this thesis, more insights could be gathered on search algorithm performance.

6.6. Conclusions

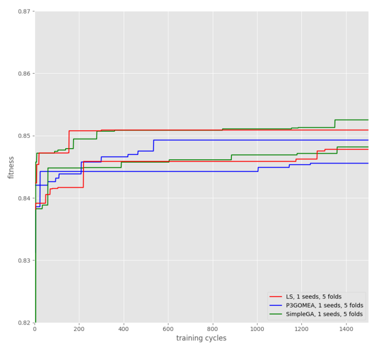
The results indicate that the algorithms perform very similarly, with a remarkable performance of SGA, outperforming the other algorithms for all but one run. This could indicate that the other algorithms get stuck in local optima quicker than SGA. The results however are not statistically significant due to the large computational cost of experiments.



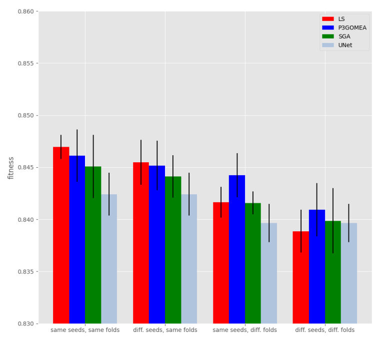
(a) 1-seed-1-fold NAS runs.



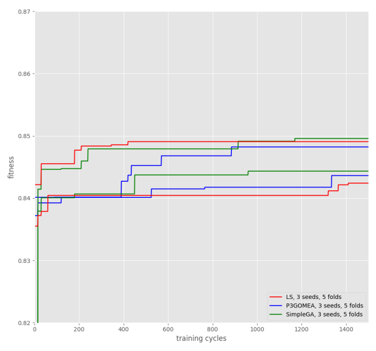
(b) Average performance increase of 1-seed-1-fold runs in different validation settings.



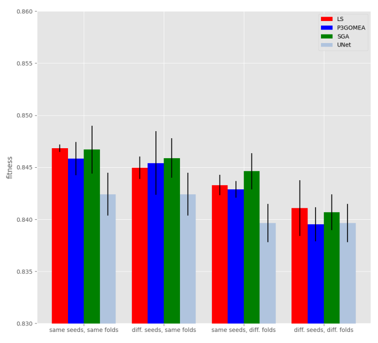
(c) 1-seed-5-fold NAS runs.



(d) Average performance increase of 1-seed-5-fold elites in different validation settings.



(e) 3-seed-5-fold NAS runs.



(f) Average performance increase of 3-seed-5-fold elites in different validation settings.

Figure 6.1: *The different NAS runs on the left show the performance on two data splits of the algorithms with identical estimation methods per subfigure. As can be seen by the clear division in performance of the two runs per evaluation method, one data split has a significantly lower average performance score than the other. The validation performance is shown on the right for the 4 different validation types to see how the performance is affected by the different validation types.*

The results also reiterate that when performing validation correctly, the performances of the elites found by all the algorithms with the chosen evaluation methods deteriorate severely. For this reason, it can be concluded that when choosing the search algorithms used in this paper, if performance estimation methods are too noisy, correlations with the validation search space low, and the number of evaluations limited, these algorithms will perform poorly and return an elite performing worse than the best network in the search space, and no better than a similarly sized U-Net.

7 Discussion

7.1. Search space experiments

In Chapter 5, the novel search space, MB-NAS, is shown to contain promising networks that can be found with the use of a LS algorithm and a 3 seeds, 5 folds validation technique. However, in Chapter 4, it is shown that, although for a different dataset, this could be deteriorated after validation. Luckily, the validation does hold up, although the results are worse than found during the NAS and not a lot better than the baselines. As can be seen in the experiments performed in both Chapter 4 and 6, if a higher correlation was achieved with the validation performance landscape, performance of the networks could be higher. The networks found as elites in the experiments in Figure 5.4, probably have a lower validation performance than other networks in the search space. This means that the search space has more potential than the chosen search algorithm and performance estimator can exploit. This reiterates the fact that both search space design and search algorithm choice are only interesting when proper performance estimation methods exist. Without proper performance estimation it is very difficult to find the best network and to know what the added value is of different search spaces and algorithms.

7.2. Statistical significance

Throughout this thesis many experiments are only performed a small number of times, such that statistical significance tests, such as t-tests, would require many assumptions to have value. The reason for the low number of experiment repetitions is the large number of GPU-hours that is necessary for NAS. A single NAS run of 1500 samples on the simple version of the search space, as run in Chapters 4 and 6, takes 10 days, where the longer runs, such as in Chapter 5, took 20 GPU days. Having to perform them with multiple search algorithms, datasets, and estimation methods, and do enough repetitions to calculate statistical significance was unfortunately infeasible[1]. The results however did present multiple insights that can help progress NAS for medical image segmentation in areas where research was lacking.

7.3. Noise and validation

Throughout Chapter 4, the noise in several medical image segmentation tasks, as well as the noise in DNN optimisation is studied. Additionally, the effect of different correlation levels between the performance landscape used in NAS, and the validation performance landscape, is analysed. What was surprising during this research is the large effect of these phenomena and how it is largely understudied in literature. Also, validation practices differ in literature. Many papers are fixated on higher scores, even when only fractional. When looking at cross-validation scores, these fractional differences in medical image segmentation datasets like MSD Prostate and MSD Spleen could be caused by different datasplits, not having to be related to increased performance of the method. Also, many networks found using NAS are used to validate performance without retraining. This is impractical as it largely reduces reproducibility of results, as well as optimising the architecture towards a certain initial state. Especially when using continuous relaxation,

[1]I would like to extend my gratitude to the CWI and my peers for all the GPU time I used up as it is. Everyone was quite patient with me, so thank you for bearing with me and providing the resources.

the same model is not likely to be achieved when retraining the network without it ever having been part of a super-network.

The insights provided in Chapter 4 could contribute to a better understanding and comparison of results for medical image segmentation. As DNN optimisation is largely stochastic, providing context in noise and validation methods can make comparison more statistically significant, even if computational costs makes this difficult. It can also potentially make results obtained by SotA algorithms actually outperform Random Search, which has been shown to sometimes not be the case [52, 54].

7.4. Further research

NAS, although young, has already delivered impressive results. As a field of research pursued by academic institutions and powerful corporations, e.g. Google and NVIDIA, alike, it has led to some revolutionary work, like being used in the development of EfficientNet [45]. For classification networks, many benchmark leaderboards contain multiple Neural Architecture Search networks among them. However, there is still room for further research in NAS. Multiple research topics seem interesting to dive in: further noise reduction in performance estimation, NAS for image transformers, and network scaling.

7.4.1. Noise reduction in performance estimation

Although quite some experimentation was done to obtain less noisy scores in performance estimation, there are still many tracks that could be explored more. Three of them seem promising to me, enough so to be mentioned here. The first is experimenting more with metrics. Although 95th % Hausdorff only made validation accuracy noisier in my initial experiments, using different metrics for validation accuracy such as Surface Dice where a threshold value can be chosen, or experimenting with Hausdorff percentiles could lead to less noise. It should be kept in mind that information is lost when making these thresholds too large, however the trade-off might prove effective.

Another possibility that could be pursued is the use of deterministic weight initialisation methods, such that seeds have less impact on the performance. This would reduce the effect of different seeds in training and validation.

The third path is to see if surrogate models could decrease noise for performance estimation of medical image segmentation CNNs such that the predictions made by the regression model are better than cheap network evaluations. This would be able to speed up the process while taking along the understanding of noise in the taks at hand.

7.4.2. NAS for image transformers

Image transformers, a Neural Network type based on a different image processing operation called attention (en lieu of convolution), seem to be outperforming CNNs for many classification tasks over the last couple of months [10], as well as medical image segmentation tasks [8]. It would be interesting to see if these transformers could be exploited by NAS for medical image segmentation. These networks are larger than most CNNs so computational restraints have to be taken into account in the feasibility of such a project. However performance increase seems substantial so it could lead to very promising results.

7.4.3. Network scaling

As explained in Chapter 5, an issue with search spaces is that they quickly grow exceedingly large. This makes performing NAS on large networks difficult. As shown for EfficientNet and ResNet, scaling networks is a promising way to increase performance. By performing NAS with a smaller image size and a smaller search space, and then scaling up to the original image size together with the network, NAS can be faster and more feasible for large scans. saving time in NAS performance estimation as smaller networks are faster to train.

8 Conclusions

Convolutional Neural Networks perform well on medical image segmentation tasks, sometimes achieving human-like performance. However, manually designing these networks is arduous and time-consuming. Neural Architecture Search has been shown to have the capability of finding CNNs that outperform manually designed networks for various tasks. However, for medical image segmentation, NAS research is limited. For this reason, this thesis looked at the different aspects of NAS for the purpose of medical image segmentation.

First, performance estimation methods were analysed in terms of noise. It appeared that these methods need to be a lot more elaborate for medical image segmentation tasks than for image classification. Through experiments, it is also shown that the correlation between the training and the validation performance landscape needs to be high. Low correlations lead to deteriorated results such that NAS does not longer outperform U-Net. A tentative method was put forward to analyse a dataset before NAS such that more informed choices can be made for performance estimation.

Next, a novel search space was proposed, which makes use of existing knowledge of high-performance classification networks, and seems to outperform U-Net, and two alternate search methods, on two datasets.

In the final chapter three search algorithms, SGA, P3GOMEA and LS, are compared and shown to perform very similarly for the task at hand. This could be due to to relatively high levels of noise and low computational budget. These findings align with papers showing that elaborate search strategies do not improve on Random Search for several classification tasks after validation if there is a low correlation between the two performance landscapes. It is concluded that before elaborate search algorithms are implemented, it is crucial to know what the noise levels are between evaluations and what the correlation levels are between dataset splits.

8.1. Performance estimation

To summarise the findings for every aspect of NAS, the research questions from Chapter 1 are reiterated here. For performance estimation the question was:

> **Research Question 1.** *Can DNN performance estimation for medical image segmentation tasks be adapted to noise such that the performance values passed to search algorithms are accurate enough for them to outperform Random Search?*

Although this question proved to be more difficult to answer than expected, many insights and conclusions were drawn during the analysis of performance estimation for medical image segmentation. First, performance estimation for NAS for medical image segmentation was found to be more complex and susceptible to noise than for image classification, due to the metrics used. Also, the small size of datasets, inter-patient variation, and intra- and inter-physician variability lead to considerable noise when using different folds and data splits for training. Next to noise from the data, network optimisation also has inherent noise. Different evaluations of the same network on the same task, have different performance values. Predicting network performance with metrics from early stopping, which seems to work on classification tasks such as CIFAR, does not work with medical image segmentation tasks, at least not for the AMC dataset. To find a network that robustly improves on other networks, it was found that NAS needs to be performed with performance estimations such that the relative noise is low, and the correlation between the evaluated network performance and the validation network performance is high. To do this without undermining validation, so without using the same seeds and k-fold split, repeated network evaluations, and high correlations between dataset splits are needed. By increasing the amount of samples in a network evaluation, and tailoring the amount of dataset splits seen by the NAS algorithm and in validation, the more robust performance of the found network will be, and the higher the probability of the NAS algorithm finding a top ranked network in your search space.

By using a proposed dataset analysis method, which evaluates the correlation between network performance on different data splits, one can make a more educated decision on how many evaluations are needed to achieve a selected correlation value and estimate network performance robustly. This does mean that if the dataset chosen to perform NAS on has a low correlation between dataset splits, and the search space evaluated is large, it will be very expensive to perform NAS. Not only to allow search algorithms enough evaluations to be able to progress through the search space, but to also estimate and validate performance in a way that reflects network performance independent of chosen seeds and datasplit.

8.2. Search space

For the search space the research question was the following:

Research Question 1. *Can a discrete bi-level search space be created such that the neural networks found by a NAS algorithm are better than hand-crafted SotA networks, as well as outperform networks found using different existing discrete search spaces, for medical image segmentation tasks?*

In Chapter 5, a novel NAS search space for medical image segmentation networks was proposed. The search space combines a simultaneous topology- and cell-level search strategy. The cell-level search, uses existing knowledge from networks with high performance in image classification tasks through means of a restricted pool of possible block configurations. This allows for unique blocks to be implemented at various locations without making the search space exceedingly large. The experiments indicate an added value of this approach. The elite networks found by a LS algorithm in the novel search space, are slightly better than the ones found by an only a macro-level search and a sequential macro-level search followed by a micro-level search. They also have a slightly higher performance than U-Net and ResU-Net. It is noted that due to limited computational resources, only one run of NAS was performed per search space, and for a limited number of network evaluations. Longer experiments with multiple runs should be performed to obtain statistical significance. Also, running NAS experiments for more datasets may provide more insights about the specific characteristics of network architecture design required for good performance across datasets. Overall, the results indicate that further research into search space refinement, allowing to exploit key features of what accounts for good deep learning performance, may push the boundaries of what can be achieved with DNNs for medical image processing.

8.3. Search algorithms

The last research question was on the performance of search algorithms:

Research Question 2. *How do Evolutionary Algorithms and Local Search perform for NAS in different levels of noise and can a decision be made on which is better?*

As performance estimation was shown to need very elaborate methods to obtain high correlations with the validation performance, the NAS algorithms were evaluated with different levels of noise, and different validation methods to vary the correlation with the validation performance values.

The results indicated that the algorithms perform very similarly, with a remarkable performance of SGA, outperforming the other algorithms for all but one run. This could indicate that the other algorithms get stuck in local optima quicker than SGA. The results however are not statistically significant due to the large computational cost of experiments.

The results also reiterate that when performing validation correctly, the performances of the elites found by all the algorithms with the chosen evaluation methods deteriorate severely. If performance estimation methods are too noisy, and correlations with the validation performance landscape are low, the algorithms in this thesis will all perform poorly and on average return elites performing far worse than the best network in the search space, and no better than a similarly sized U-Net. This explains why Random Search is shown to perform equally to other SotA algorithms after being validated, as decisions made by the algorithms do not lead to better validation scores.

8.4. Recommendation for NAS for medical image segmentation

When looking at the multiple aspects of NAS for medical image segmentation, there is one recommendation that I would like to end with. I would suggest that before designing more and more complex search spaces and analysing search algorithms, it is vital that performance estimation is further researched, such that the noise in network performance is reduced, even if this is done by gathering larger and ”cleaner” datasets. By being able to evaluate the noise and trying to understand how this relates to the performance of NAS and the elite network found by the algorithm, performances of NAS on different datasets can be compared with more context of the task at hand. In my opinion, noise analysis in performance estimation is very under-represented in literature. The most research into performance estimation is generally done with the aim of speed-up so that more exploration can be done by the algorithm. However, as shown in my experiments, the low correlation in cheaper evaluations mean the elitist performance will likely be very different when validated. By looking into reducing noise, such that the performance of the network is properly reflected, more robust results will be achieved, hopefully leading to new insights in Deep Learning, NAS, and the use of EAs for this task.

Appendix A

Mixed-Block Neural Architecture Search for Medical Image Segmentation

Martijn M.A. Bosma, Arkadiy Dushatskiy[1], Monika Grewal[1], Tanja Alderliesten, Peter A. N. Bosman

ABSTRACT

Deep Neural Networks (DNNs) have the potential for making various clinical procedures more time-efficient by automating medical image segmentation. Due to their strong, in some cases human-level, performance, they have become the standard approach in this field. The design of the best possible medical image segmentation DNNs, however, is task-specific. Neural Architecture Search (NAS), i.e., the automation of neural network design, has been shown to have the capability to outperform manually designed networks for various tasks. However, the existing NAS methods for medical image segmentation have explored a quite limited range of types of DNN architectures that can be discovered. In this work, we propose a novel NAS search space for medical image segmentation networks. This search space combines the strength of a generalised encoder-decoder structure, well known from U-Net, with network blocks that have proven to have a strong performance in image classification tasks. The search is performed by looking for the best topology of multiple cells simultaneously with the configuration of each cell within, allowing for interactions between topology and cell-level attributes. From experiments on two publicly available datasets, we find that the networks discovered by our proposed NAS method have better performance than well-known handcrafted segmentation networks, and outperform networks found with other NAS approaches that perform only topology search, and topology-level search followed by cell-level search.

1. INTRODUCTION/DESCRIPTION OF PURPOSE

A growing amount of clinical applications, such as computer-aided diagnostic systems, are benefiting from recent advances in automated medical image segmentation, most notably from Deep Neural Networks (DNNs).[2] Given a medical scan, a DNN can provide contours of organs or regions of interest (e.g., tumors), with clinically acceptable segmentation quality within a matter of seconds.[3] Designing a State-of-the-Art (SotA) DNN, however, is often task-specific. In order to design a DNN, choices have to be made for the topology of the network such as depth, and connections between cells (a cell is a group of operations that transform the feature maps in a DNN), as well as the configuration of each cell, e.g., convolutional kernel size, or activation function. This gives rise to an inconceivably large amount of network architecture design possibilities, which is impossible to manually navigate through in an exhaustive fashion, or even by means of intelligent design, while ensuring the best choices are made.

Neural Architecture Search (NAS), i.e., the automated design of neural network architectures, can effectively and efficiently search through this space of possible network architecture designs and find a network that is highly tailored to the task at hand.[4] While research on NAS for medical image segmentation has not been as elaborate as for natural image classification, it has already shown promising results by outperforming the SotA architectures.[5] In our opinion, further research on NAS for medical image segmentation can make its contributions even more significant.

NAS involves three key components: (1) The search space (the set of all possible networks given the specified architectural constraints); (2) the search algorithm (the algorithm to navigate the search space); (3) performance estimation (the choices made to score a network's performance, such that these networks can be ranked by the search algorithm). So far, NAS research has been more elaborate for image classification tasks.[6] Potentially, strong performing search algorithms, and fast and accurate performance estimation methods can be translated to the medical image segmentation domain. The search space, however, is specific to segmentation tasks. Several papers have proposed search spaces for medical image segmentation. These include searching for a U-Net like encoder-decoder topology i.e., by adapting only the cells within,[7] by searching for best topology followed by the best convolution size within cells,[8] or a combination of topology and cell search using continuous relaxation and a so-called Super-network.[9]

We believe that within the scope of a U-Net-like topology search, there is a room for making the search space more flexible – instead of repeating the same cell structure, we propose to allow various cell structures in a network, potentially resulting in a better performance. Further, we believe that the configuration of cells can benefit from existing knowledge about networks with a strong performance on classification tasks. Therefore, we propose to search through a pre-selected pool of configurations (which are taken from advanced well-known classification networks[10])

[1]equal contribution

[2]Bernard et al., "Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis"; Fauw et al., "Clinically applicable deep learning for diagnosis and referral in retinal disease".

[3]Nikolov et al., "Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy".

[4]Elsken, Metzen, and Hutter, "Neural architecture search: A survey".

[5]Yu et al., "C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation"; Yan et al., "MS-NAS: Multi-scale neural architecture search for medical image segmentation"; Weng et al., "NAS-UNet: Neural architecture search for medical image segmentation".

[6]Elsken, Metzen, and Hutter, "Neural architecture search: A survey".

[7]Weng et al., "NAS-UNet: Neural architecture search for medical image segmentation".

[8]Yu et al., "C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation".

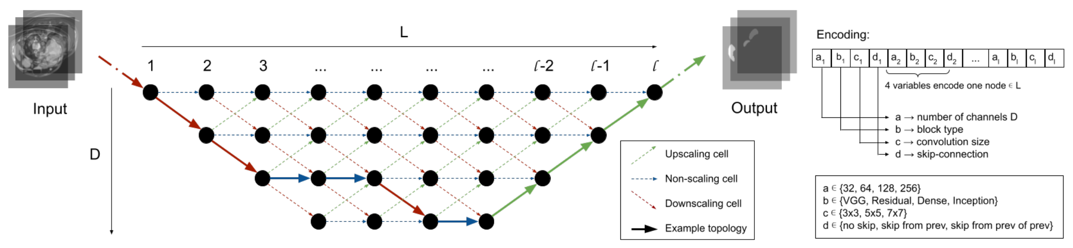
[9]Yan et al., "MS-NAS: Multi-scale neural architecture search for medical image segmentation".

[10]He et al., "Deep residual learning for image recognition"; Huang et al., *Densely Connected Convolutional Networks*; Szegedy et al., "Inception-v4, inception-ResNet and the impact of residual connections on learning".

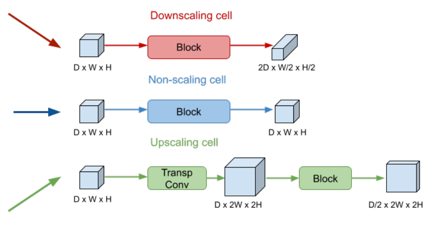
instead of searching for a cell configuration from scratch. Apart from benefiting from advanced performance of these well known classification networks, this proposed improvement will also help avoiding the explosive growth of the search space caused by searching the configuration of each cell from scratch. Finally, in contrast to recent research,[11] where topology level search is followed by cell level search, we search for both topology as well as the configuration of each cell simultaneously. This allows to take into consideration the possible interaction between the topology-level search and cell-level, potentially yielding better performing networks. The combination of these improvements results in the proposed approach, which we will further refer to as Mixed-Block NAS (**MB-NAS**).

2. METHOD

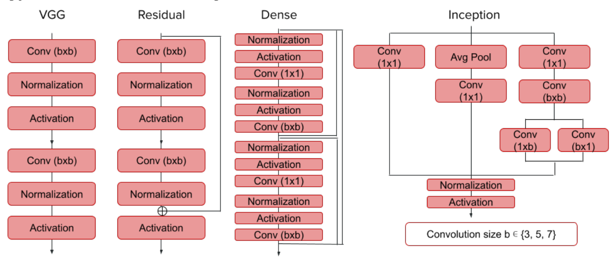
2.1 Search space



(a) Topology-level search and encoding.



(b) Cell types.



(c) Block types.

Figure 1: *Description of MB-NAS search space.*

At **topology-level** (Figure 1a), the search space contains all possible network architectures resulting from varying connections between cells of different types. We consider three possibilities for a cell (Figure 1b): a downscaling, an upscaling, or a non-scaling cell. A downscaling cell means that the input image resolution is halved, while the number of channels is doubled. Upscaling is the opposite operation, i.e., doubling the resolution and halving the number of channels, while a non-scaling cell changes neither the resolution nor the number of channels in the feature maps. The input to a cell can be either just the feature maps from the preceding cell (no skip-connection), or the feature maps of the preceding cells concatenated with the feature maps from any previous cell with the same feature maps spatial dimensions (a single skip-connection).

At **cell-level**, different configurations of each cell are searched. The configuration of a cell is encoded by two variables: the type of block (a block is an organised structure consisting of multiple convolution and normalisation layers, as well as activation functions) within the cell, and the convolutional kernel size within the block. Instead of searching for the topology of the blocks within a cell, which would make the search space incredibly large, we used predefined blocks derived from previously SotA architectures for classification (Figure 1c). In this work we consider VGG blocks which are standard in U-Net, as well as Residual blocks, Dense blocks and Inception blocks. In this way, we allow the search space to chose a different cell configuration at every edge (instead of repeating one throughout the network) while preventing the further growth of the search space.

The network architecture is represented by connections between a fixed number of nodes (fixed to 10 in our experiments). Each node l (Figure 1a) is represented by 4 categorical variables: a_l = number of channels, b_l = the block type, c_l = convolution size, and d_l = skip-connection source. The cell type is derived by the difference in number of channels between two nodes. The topology of the neural network is encoded by variables a_l and d_l at each node. The cell-level search is represented by variables b_l and c_l at each node. Note that the standard U-Net shape is included in the topology-level search space. The resulting search space contains $1.14 * 10^{18}$ possible networks.

[11] Yu et al., "C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation"; Yan et al., "MS-NAS: Multi-scale neural architecture search for medical image segmentation".

2.2 Datasets

In our experiments, we used two datasets that can be found on the Medical Decathlon[12] challenge website[13]. The first is a collection of 3D MRI-scans of the prostate, alongside their multi-class (Central and Peripheral prostate zones) segmentations. The second is the Spleen dataset, containing 3D-CT scans and single class segmentations of the spleen. Both datasets are represented as a collection of 2D slices (no resampling was used as preprocessing).

2.3 Implementation details

We use Local Search (LS) as the search algorithm as it has been shown to be a strong baseline for NAS.[14]Due to limited computational resources, the NAS was run for only 150 network evaluations. Each network was evaluated using the validation Dice score averaged over the last 20% of training epochs. The number of epochs (100 epochs for prostate, 50 for spleen) was decided such that saturation was ensured based on preliminary runs. We used the average score of 5-fold patient-level cross-validation, repeated for 3 network initialisations, as performance estimator. Multiple initialisations reduce the noise in scoring, providing more reliable information to the search algorithm, making it easier to navigate the search space. The use of multiple folds decreases noise caused by data splits.

For network training, we used the Adam optimiser with a learning rate 10^{-3} and polynomial decay with an exponent of 0.9. The loss function was foreground soft Dice. The batch size was 32 and input image size was 128x128. The data was augmented using scaling, shifting, rotating, flipping, and brightness adjustment to avoid overfitting.

2.4 Experimental setup

We performed experiments to compare the proposed search space design (Mixed-Block, or **MB-NAS**) against the following alternative approaches. In the first considered alternative search space (**Macro-NAS**) only topology search is performed. All block types are fixed to be standard U-Net blocks (VGG). The second considered search space (**Micro-NAS**) has the U-Net topology, where only the block type and convolution sizes are subject to the search. In the third alternative search space (**Bilevel-NAS**), a bi-level approach was used which means that first the topology-level is searched, and then the convolution size, similar to the Coarse-to-Fine approach.[15]

The best networks found by the proposed search space were evaluated against two hand-crafted neural network architectures: standard **U-Net**, and U-Net with a ResNet-50 encoder (**ResU-Net**). Implementations are taken from the Pytorch Segmentation Models library.[16]

3. RESULTS AND DISCUSSION

The performance of the different search spaces (**MB-NAS**, **Macro-NAS**, **Micro-NAS**, **Bilevel-NAS**) can be seen in Figure 2a and 2b. The the architectures of the best networks from MB-NAS are visualised in Figure 2c. The performance of the best networks from different NAS approaches, **U-Net**, and **ResU-Net** is summarised in Table 1.

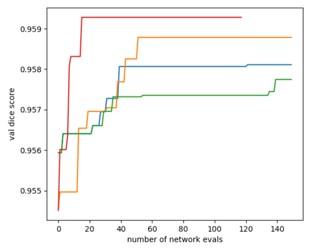
Table 1: *Performance comparison of U-Net, ResU-Net, and the best networks from different NAS approaches. The metric are averaged for 5 runs of 5-fold cross-validation. Best values in each column are highlighted in bold. HD: Hausdorff distance (95% cutoff), SD: Surface Dice (2mm threshold), MMAC: Mega Multiply–ACcumulate operations, Params: number of parameters,* $\times 10^6$.

Model	Prostate dataset					Spleen dataset				
	Dice	HD	SD	MMAC	Params	Dice	HD	SD	MMAC	Params
U-Net	0.6702	8.833	0.6046	302	18.4	0.9578	1.412	0.9174	302	18.4
ResU-Net	0.6580	9.441	0.5705	**166**	32.5	0.9464	1.625	0.9047	**166**	32.5
Macro-NAS	0.6593	8.606	0.5977	256	3.39	0.9566	1.467	0.9167	255	**3.39**
Micro-NAS	**0.6796**	**8.394**	**0.6203**	1,295	22.7	0.9567	1.388	0.9177	795	2.83
Bilevel-NAS	0.6702	8.492	0.6134	414	6.77	0.9553	1.449	0.9145	415	6.78
MB-NAS	0.6760	8.419	0.6192	644	**3.04**	**0.9592**	**1.385**	**0.9189**	1,294	22.7

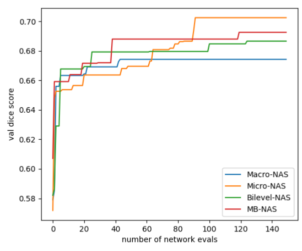
Spleen dataset. The best performing network was found by the proposed search space, MB-NAS. In table 1, we see that this network shows the best performance by all three considered performance metrics. We note that the topology (Figure 2c, upper row) is quite different from the standard U-Net. Furthermore, we note that blocks of all four types are included in the architecture, as well as different convolution sizes, meaning that both the topology-level and cell-level search space parts were utilised to find this network.

Prostate dataset. Different than for the spleen dataset, the best network (Figure 2c, bottom row) is very similar to the U-Net architecture. This provides additional evidence to the argument that the architecture of the best DNN is task-specific. This also indicates that the U-Net topology is best suited for the underlying task, giving an added advantage to Micro-NAS, wherein different blocks and convolution sizes are searched within U-Net topology. Consequently, Micro-NAS yields best performance. However, it should be noted that MB-NAS could find a network with comparable performance despite the fact that the search was performed for both topology as well as cell configuration.

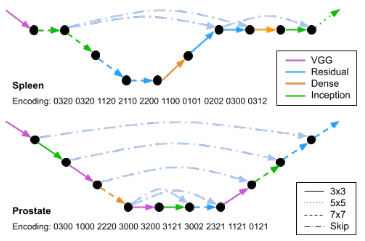
[12] Antonelli et al., *The Medical Segmentation Decathlon*.
[13] http://medicaldecathlon.com/
[14] Ottelander et al., *Local search is a remarkably strong baseline for neural architecture search*.
[15] Yu et al., "C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation".
[16] Yakubovskiy, *Segmentation Models*.



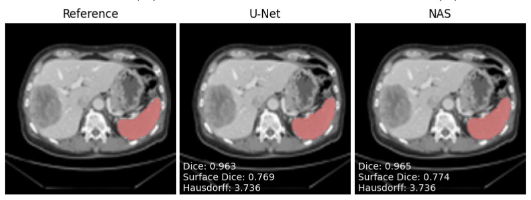
(a) NAS spleen.



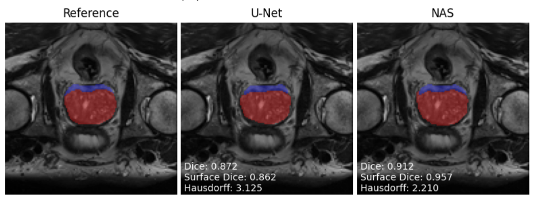
(b) NAS prostate.


Spleen
Encoding: 0320 0320 1120 2110 2200 1100 0101 0202 0300 0312
VGG
Residual
Dense
Inception
Prostate
Encoding: 0300 1000 2220 3000 3200 3121 3002 2321 1121 0121
3x3
5x5
7x7
Skip

(c) Best networks from MB-NAS.



(d) Example segmentations on the spleen dataset.



(e) Example segmentations on the prostate dataset. red: central gland, purple: peripheral zone.

Figure 2: *The progress of the LS for the (a) spleen and (b) prostate datasets, respectively. The best networks found by LS with MB-NAS are shown in (c). (d) and (e) show example segmentations.*

It is also worth noting that no block seems to be ultimately preferable to other blocks in the best performing networks indicating the advantage of searching for a mixed-block configuration. Further, we note that the best network from NAS outperforms the manually hand-crafted U-Net and ResU-Net networks.

Comparison with SotA. It can be argued that the results for the Prostate dataset in Table 1 are not at the same level as the SotA results given by e.g., nnU-Net.[17] This is due to higher image resolution used by nnU-Net, additional preprocessing, carefully chosen data augmentations, post-processing, and advanced inference method, that are all different from the setup we use in NAS. Therefore we add results using the nnU-Net framework with the found networks from NAS. For prostate, the 5-fold cross-validation Dice score is **0.7325** for MB-NAS vs **0.7315** for nnU-Net. It should be noted that while all the settings discussed above were specifically chosen to work well with a tailored U-Net in the nnU-Net framework, the architectures found by NAS were found using entirely different settings, which gives an unfair advantage to nnU-Net. Nevertheless, the architecture found by NAS still performs comparable to nnU-Net in these settings, which is remarkable.

4. NEW OR BREAKTHROUGH WORK TO BE PRESENTED

We consider NAS for medical image segmentation with deep learning. The main contribution is a novel search space that combines the strengths of the encoding-decoding structure of U-Net with the strength of blocks adapted from well-known high-performing classification networks. Additionally, we perform both the topology and cell-level search simultaneously. The networks found with the proposed approach perform better than the manually designed SotA networks. They also have an indicative advantage over networks found using other search approaches from related literature, namely topology search and consecutive topology and convolution size search.

5. CONCLUSIONS

We have proposed an approach for medical image segmentation NAS, which involves a novel search space and simultaneous topology- and cell-level search strategy. In the cell-level search, we used existing knowledge from networks with high performance in image classification tasks, i.e. ResNet, DenseNet and InceptionNet, to create a pool of possible block configurations. The experiments in this paper show the added value of this approach. We note that due to limited computational resources, only one run of NAS was performed per search space, and for a limited number of network evaluations. Longer experiments with multiple runs would have helped draw more definitive conclusions. Next to it, running NAS experiments for more datasets may provide more insights about the specific characteristics of network architecture design required for good performance across datasets. Overall, the results indicate that further research into search space refinement, allowing to exploit key features of what accounts for good deep learning performance, may yet push the boundaries of what can be achieved with deep neural networks for medical image processing.

Note. This work has not been submitted for publication in any other conference proceedings or journal.

[17]Isensee et al., "nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation".

Bibliography

[1] Michela Antonelli et al. The medical segmentation decathlon, 2021.

[2] Yahya Baba and Mohammad Niknejad. Apparent diffusion coefficient, February 2013. URL `https://doi.org/10.53347/rid-21759`.

[3] Woong Bae, Seungho Lee, Yeha Lee, Beomhee Park, Minki Chung, and Kyu-Hwan Jung. Resource optimized neural architecture search for 3d medical image segmentation, 2019.

[4] Maria Baldeon Calisto and Susana K. Lai-Yuen. Emonas-net: Efficient multiobjective neural architecture search using surrogate-assisted evolutionary algorithm for 3d medical image segmentation. *Artificial Intelligence in Medicine*, 119:102154, 2021. ISSN 0933-3657. doi: https://doi.org/10.1016/j.artmed.2021.102154. URL `https://www.sciencedirect.com/science/article/pii/S0933365721001470`.

[5] Olivier Bernard et al. Deep learning techniques for automatic mri cardiac multi-structures segmentation and diagnosis. *IEEE Transactions on Medical Imaging*, 37(11):2514–2525, 2018. doi: 10.1109/TMI.2018.2837502.

[6] Henry Blumberg. Book review: Grundzüge der mengenlehre. *Bull. Amer. Math. Soc. 27*, 27(3):116–130, December 1920. doi: 10.1090/s0002-9904-1920-03378-1. URL `https://doi.org/10.1090/s0002-9904-1920-03378-1`.

[7] Luigi Franco Cazzaniga, Maria Antonella Marinoni, Alberto Bossi, Ernestina Bianchi, Emanuela Cagna, Dorian Cosentino, Luciano Scandolaro, Marica Valli, and Milena Frigerio. Interphysician variability in defining the planning target volume in the irradiation of prostate and seminal vesicles. *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*, 47(3):293–296, June 1998. doi: 10.1016/s0167-8140(98)00028-0. URL `https://doi.org/10.1016/s0167-8140(98)00028-0`.

[8] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation, 2021.

[9] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945. doi: 10.2307/1932409. URL `https://doi.org/10.2307/1932409`.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[11] Arkadiy Dushatskiy, Tanja Alderliesten, and Peter A. N. Bosman. A novel surrogate-assisted evolutionary algorithm applied to partition-based ensemble learning, 2021.

[12] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[13] De Fauw et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*, 24(9):1342–1350, 2018.

[14] Kaiming He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] T. Heimann, B. van Ginneken, M.A. Styner, Y. Arzhaeva, V. Aurich, C. Bauer, A. Beck, C. Becker, R. Beichel, G. Bekes, F. Bello, G. Binnig, H. Bischof, A. Bornik, P. Cashman, Ying Chi, A. Cordova, B.M. Dawant, M. Fidrich, J.D. Furst, D. Furukawa, L. Grenacher, J. Hornegger, D. Kainmuller, R.I. Kitney, H. Kobatake, H. Lamecker, T. Lange, Jeongjin Lee, B. Lennon, Rui Li, Senhu Li, H.-P. Meinzer,

G. Nemeth, D.S. Raicu, A.-M. Rau, E.M. van Rikxoort, M. Rousson, L. Rusko, K.A. Saddi, G. Schmidt, D. Seghers, A. Shimizu, P. Slagmolen, E. Sorantin, G. Soza, R. Susomboon, J.M. Waite, A. Wimmer, and I. Wolf. Comparison and evaluation of methods for liver segmentation from CT datasets. *IEEE transactions on medical imaging*, 28(8):1251–1265, August 2009. doi: 10.1109/tmi.2009.2013851. URL `https://doi.org/10.1109/tmi.2009.2013851`.

[16] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0262082136.

[17] Jiang Hsieh. *Computed tomography : principles, design, artifacts, and recent advances*. SPIE Optical Engineering Press, Bellingham, WA, 2003. ISBN 978-0-8194-4425-7.

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[20] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: A self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2): 203–211, 2021.

[21] Davood Karimi and Septimiu E. Salcudean. Reducing the hausdorff distance in medical image segmentation with convolutional neural networks, 2019.

[22] Ali Emre Kavur, M. Alper Selver, Oğuz Dicle, Mustafa Barış, and N. Sinem Gezer. CHAOS - Combined (CT-MR) Healthy Abdominal Organ Segmentation Challenge Data. *Medical Image Analysis*, April 2019. doi: 10.5281/zenodo.3362844. URL `https://doi.org/10.5281/zenodo.3362844`.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[25] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin. Exploring strategies for training deep neural networks. *Journal of machine learning research*, 10(1), 2009.

[26] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, Robin Strand, Filip Malmberg, Yangming Ou, Christos Davatzikos, Matthias Kirschner, Florian Jung, Jing Yuan, Wu Qiu, Qinquan Gao, Philip "Eddie" Edwards, Bianca Maan, Ferdinand van der Heijden, Soumya Ghose, Jhimli Mitra, Jason Dowling, Dean Barratt, Henkjan Huisman, and Anant Madabhushi. Evaluation of prostate segmentation algorithms for MRI: The PROMISE12 challenge. *Medical Image Analysis*, 18(2):359–373, February 2014. doi: 10.1016/j.media.2013.12.002. URL `https://doi.org/10.1016/j.media.2013.12.002`.

[27] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation, 2019.

[28] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.

[29] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search, 2020.

[30] Ngoc Hoang Luong, Tanja Alderliesten, Arjan Bel, Yury Niatsetski, and Peter A.N. Bosman. Application and benchmarking of multi-objective evolutionary algorithms on high-dose-rate brachytherapy planning for prostate cancer treatment. *Swarm and Evolutionary Computation*, 40:37–52, 2018. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2017.12.003. URL `https://www.sciencedirect.com/science/article/pii/S2210650217306685`.

[31] John E. McNeal. The zonal anatomy of the prostate. *The Prostate*, 2(1):35–49, 1981. doi: 10.1002/pros.2990020105. URL `https://doi.org/10.1002/pros.2990020105`.

[32] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training, 2021.

[33] Jerome Myers. *Research design and statistical analysis.* Lawrence Erlbaum Associates, Mahwah, N.J, 2003. ISBN 978-0-8058-4037-7.

[34] Stanislav Nikolov et al. Deep learning to achieve clinically applicable segmentation of head and neck anatomy for radiotherapy. *arXiv preprint arXiv:1809.04430v3*, 2018.

[35] V. Nissen and J. Propach. On the robustness of population-based versus point-based optimization in the presence of noise. *IEEE Transactions on Evolutionary Computation*, 2(3):107–119, 1998. doi: 10.1109/4235.735433.

[36] T. Den Ottelander, A. Dushatskiy, M. Virgolin, and P. A. N. Bosman. Local search is a remarkably strong baseline for neural architecture search, 2020.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[38] Binxin Ru, Clare Lyle, Lisa Schut, Miroslav Fil, Mark van der Wilk, and Yarin Gal. Speedy performance estimation for neural architecture search, 2021.

[39] Binxin Ru, Clare Lyle, Lisa Schut, Mark van der Wilk, and Yarin Gal. Revisiting the train loss: an efficient performance estimator for neural architecture search, 2021. URL `https://openreview.net/forum?id=XvOH0v2hsph`.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

[41] Neeraj Sharma, AmitK Ray, KK Shukla, Shiru Sharma, Satyajit Pradhan, Arvind Srivastva, and LalitM Aggarwal. Automated medical image segmentation techniques. *Journal of Medical Physics*, 35(1):3, 2010. doi: 10.4103/0971-6203.58777. URL `https://doi.org/10.4103/0971-6203.58777`.

[42] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search, 2020.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[44] Christian Szegedy et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI Conference on Artificial Intelligence*, 2017.

[45] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[46] Dirk Thierens and Peter A.N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 617–624, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305570. doi: 10.1145/2001576.2001661. URL `https://doi.org/10.1145/2001576.2001661`.

[47] Dirk Thierens and Peter AN Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 617–624, 2011.

[48] Marjolein C. van der Meer, Tanja Alderliesten, Bradley R. Pieters, Arjan Bel, Yury Niatsetski, and Peter A. N. Bosman. Better and faster catheter position optimization in hdr brachytherapy for prostate cancer using multi-objective real-valued gomea. In *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference, pages 1387–1394. Association for Computing Machinery, Inc, 2018. doi: 10.1145/3205455.3205505. 2018 Genetic and Evolutionary Computation Conference, GECCO 2018 ; Conference date: 15-07-2018 Through 19-07-2018.

[49] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[50] Pavel Yakubovskiy. Segmentation models. `https://github.com/qubvel/segmentation_models`, 2019.

[51] Xingang Yan et al. Ms-nas: Multi-scale neural architecture search for medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 388–397. Springer, 2020.

[52] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard, 2020.

[53] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/ying19a.html`.

[54] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search, 2019.

[55] Qihang Yu et al. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *IEEE*, pages 4126–4135, 2020.

[56] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

[57] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2017.