UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Loss function analysis and optimization for spinal segmentation in Magnetic Resonance Imaging

Masterarbeit
zur Erlangung des Grades
MASTER OF SCIENCE
im Studiengang Computervisualistik

vorgelegt von

Manuel Hun

Betreuer: M.Sc. Ivanna Kramer Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: M.Sc. Ivanna Kramer, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im Juli 2021

Kurzfassung

Loss Funktionen beschreiben wie akkurat die Vorhersage des Modells im Vergleich mit der Grundwahrheit ist, und spielen daher eine essentielle Rolle im Trainingsprozess eines neuronalen Netzwerkes. Im Bereich der medizinischen Bildsegmentierung wurden viele unterschiedliche Loss Funktionen entwickelt, die für spezielle Teilbereiche im Vergleich zu standardmäßigen Loss Funktionen überlegene Ergebnisse erzielen, jedoch kann dieser Erfolg, wegen der fehlenden Generalisierung, nicht auf andere Teilgebiete übertragen werden. Deshalb beschäftigt sich diese Thesis mit der primären Forschungsfrage, welche Loss Funktionen mit der gegebenen Modellarchitektur von 3D U-Net die besten Segmentierungsergebnisse für die Magnetresonanztomographie Daten von Hals- und Lendenwirbel erzielt. Dafür werden die Unterschiede und Vorteile von selektierten Verlustfunktionen durch die Kategorisierung und einer Analyse aufgezeigt. Zusätzlich als sekundäre Forschungsfragen, um auf Automatisierungsmöglichkeiten des Trainingsprozesses einzugehen, werden Methoden vorgestellt, die das Ziel haben, Loss Funktionen direkt anhand der Daten zu lernen und vorhandene Loss Funktionen zu optimieren. Alle Ergebnisse der Loss Funktionen und der angewandten Methoden werden anschließend in einer umfassenden Studie zusammengefasst, präsentiert und verglichen.

Abstract

Loss functions describe how accurate the prediction of the model is compared to the ground truth, and therefore play an essential role in the training process of a neural network. In the field of medical image segmentation, many different loss functions have been developed that achieve superior results for specific subdomains compared to standard loss functions, but this success cannot be transferred to other subdomains due to the lack of generalisation. Therefore, this thesis deals with the primary research question, which loss function achieves the best segmentation results for the magnetic resonance imaging data of cervical and lumbar vertebrae with the given model architecture of 3D U-Net. For this purpose, the differences and advantages of selected loss functions are shown through categorisation and analysis. Additionally, as secondary research questions to address automation possibilities of the training process, methods are presented that attempt to learn loss functions directly from the data and to optimise existing loss functions. In a comprehensive study, all results of the loss functions and the applied methods are then summarised, presented and compared.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☒ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☒ nein ☐

Koblenz, den 22. Juli 2021

Contents

Nomenclature

Scalars

η_t	Current learning rate
$\eta_{t=0}$	Initial learning rate
e_m	Maximum training epochs
e_t	Current epoch
f_d	Validation dice value
f_i	Inspection value
f_p	Properties value

Loss functions

$\mathcal{L}_A$	Asymmetric
$\mathcal{L}_{BP}$	Pure Boundary
$\mathcal{L}_B$	Boundary with arbitrary regional loss function
$\mathcal{L}_{CE}$	Cross Entropy
$\mathcal{L}_{DCE}$	Dice-Cross Entropy
$\mathcal{L}_{DF}$	Dice-Focal
$\mathcal{L}_D$	Dice
$\mathcal{L}_E$	Effectiveness
$\mathcal{L}_{HD}$	Hausdorff distance with arbitrary regional loss function
$\mathcal{L}_H$	Pure Hausdorff distance

$\mathcal{L}_{TF}$ Tversky-Focal

$\mathcal{L}_{T}$ Tversky

Matrices

$\hat{\boldsymbol{Y}}$ Network prediction

$\boldsymbol{X}$ Input with dimension (N, C, Z, W, H)

$\boldsymbol{Y}$ Ground truth

Sets

$\hat{Y}$ Prediction domain

X Input domain

Y Target domain

Chapter 1

Introduction

1.1 Problem statement

In order to make health-critical diagnoses or plan medical operations, experts need information about the patient's condition. This information is usually provided for certain diseases by computer-based methods such as computed tomography (CT), positron emission tomography (PET) or magnetic resonance imaging (MRI). Each of these methods has its advantages and disadvantages. If there is a bone fracture or suspected internal bleeding, the CT scan may give a better indication, but the patient will be exposed to radiation. However, if a patient's tissues needs to be examined, MRI is more suitable and is generally a lower-risk method, but it requires a greater amount of time and the patient must not carry any metals in or on their body. In general, MRI also provides visual information about bone structures, but it is not equal in quality to a CT scan and is therefore not the standard for observing bone structures.

The manual processing of these medical image or volume data by an medical expert for health-critical operations or diagnoses can consume a significant amount of time. Computer-based methods can help to reduce this time disadvantage, make the prediction more precise and thus minimise the number of experts involved. The state of the art approach for processing data volumes for high accuracy and fast throughput represent neuronal networks trained in a supervised manner. However, in order to exploit the full potential of neural networks, the training must be optimally designed. In training, however, there are many decision options that have a direct influence on the performance of the neural networks. An important decision is therefore the selection of the loss function, which guides the neuronal network through the training process. Moreover, there exists no general heuristic for selecting a suitable loss function. Therefore, an analysis of existing loss functions or possibilities to optimize them automatically represents an interesting research question.

1.2 Research question

The primary and secondary research questions of this thesis arise from the observations made in the formulation of the problem:

- R1: With the given Magnetic Resonance Imaging data of cervical and lumbar vertebrae, which loss function with the 3D U-Net architecture yields the best segmentation results?
- R2: Is it possible to directly learn a loss function for segmentation tasks only based on the provided data?
- R3: Is it possible to optimize existing loss functions for segmentation tasks?

1.3 Thesis structure

An introduction to the topic of semantic segmentation as well as to the topic of loss functions is given in Chapter 2. Chapter 3 deals with the mathematical basics of loss functions, classifies existing loss functions and explains them in more detail. Chapter 4 deals with methods, that tackle the second research question on how to learn a new loss function directly based on the underlying datasets. Chapter 5 introduces the methods, that aim to optimize existing loss functions. An introduction to the provided datasets of cervical and lumbar spine, the preprocessing techniques and the used data augmentation methods follows in Chapter 6. The decisions for the selected model architecture are presented and justified in Chapter 7. Chapter 8 explains the design of the experiments, the validation strategy, and presents and discusses the results and limitations of the performed experiments. Finally, Chapter 9 provides a brief summary, outlook, and conclusion of the thesis.

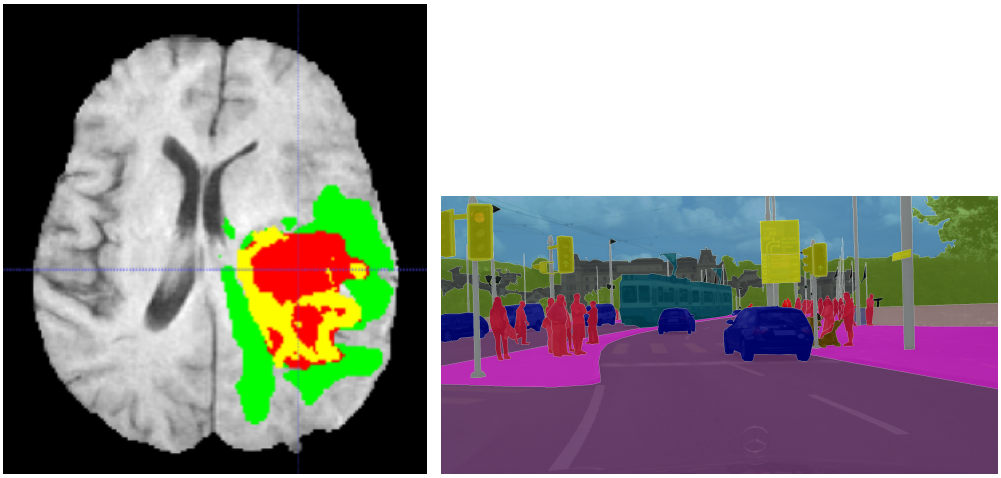
Chapter 2

Previous work

2.1 Semantic segmentation

Segmentation generally describes the process of dividing an entire unit into smaller segments according to an underlying heuristic. In computer vision, this process is guided by methods that attempt to recognise patterns in image or volume data. In semantic segmentation, each pixel or voxel is then assigned to a specific class on the assumption of the recognised pattern. The application areas of segmentation tasks range from the medical field, where, for example, malignant tumours must be segmented in specific regions of the human body as shown in Fig 2.1 (a), to industrial applications,where, for example, autonomous driving cars have to assess their environment in order to make decisions about their own driving behaviour based on this information as shown in Fig 2.1 (b). The history of groups of methods that attempt to identify patterns on the basis of which segmentation can be performed ranges from classical approaches such as thresholding, histogram-based, clustering, graph-based to modern approaches that utilise neural networks or machine learning techniques.

The first approaches of neural networks on image data were given by LeCun et al. [LeC+89], who used a neural network to classify handwritten zip codes. The foundation for medical image segmentation with neural networks was provided by Ronneberger et al. [RFB15], who proposed a new architecture called U-Net for the purpose of segmenting cell stuctures. This architecture has become the de facto standard for 2D medical image segmentation, with only minor modifications, such as in [Zho+18], where more nested skip pathways were added to the architecture. The generalisation of U-Net to volume data by Çiçek et al. [Çiç+16] followed, which is then called 3D U-Net. A new approach by Isensee et al. [Ise+18] called nnU-Net focuses less on making many new modifications to the standard U-Net architecture and more on the correct selection and optimal training of the architecture. The selection of the architecture components is conducted on the assumption of a heuristic, which obtains the information from a sta-



(a) (b)

Figure 2.1: Exemplary use cases of segmentation tasks in (a) is the determination of malignant tumour tissue[1] in the head area of a person and in (b) is the semantic assignment of an everyday scene in city traffic[2] for autonomous driving.

tistical analysis of the data. Isensee et al.'s suggestions were shown to lead to superior performance on several datasets, as described in [Ise+18].

2.2 Loss functions

The focus in the research area of deep learning, and in particular the classification and segmentation of images or volumes, is mainly on improving existing architectures in terms of accuracy, speed and complexity. However, publications such as [Ise+18] show that great value is also attached to optimising the actual training process. Loss functions serve an essential role in providing an evaluation of the correctness of the results produced during the training process of a neural network. In the areas where the datasets consist of natural images, such as in the ImageNet[3] [Den+09] dataset, loss functions that come from the area of distribution-based loss functions, such as the cross-entropy or the focal loss, which is a modified version of the cross-entropy, can be seen as the standard. This is partly because the datasets with natural images contain a relatively small class imbalance. However, in order to solve the problem of a possible class imbalance in datasets with natural images, the Focal loss [Lin+18] was proposed. Medical datasets, on the other hand, usually have a high class imbalance and therefore use loss functions that take the overlaps between regions as the basis of their calculation. Promi-

[2] Source: https://research.nvidia.com/publication/2018-09_3D-MRI-Brain [accessed: 09.04.2021]
[2] Source: https://www.cityscapes-dataset.com/examples [accessed: 10.04.2021]
[3] https://www.image-net.org/index.php

nent suggestions are the Dice loss or the Intersection over Union. In addition, other loss functions such as the Asymmetric loss or the Tversky loss have been proposed that generalise existing region-based loss functions into a single function. To address the diverse requirements that medical datasets have in their structure, other loss functions have been proposed that take into account the distance between prediction and ground truth. Well-known examples are the Hausdorff distance or the Boundary loss. Other proposals combine the properties of individual loss functions, which are called compound loss functions, such as the Tversky-Focal loss, which is based on a region-based approach and takes over the property of the Focal loss. An overview of the categories of loss functions mentioned can be seen in Figure 2.2.

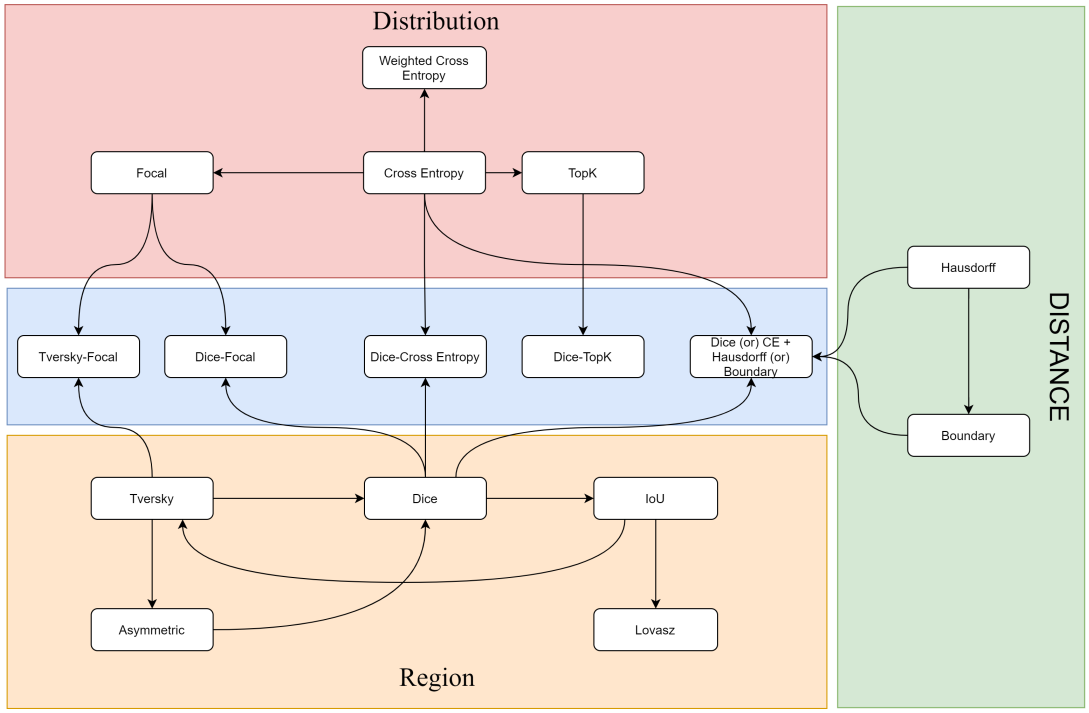


Figure 2.2: Overview of the different categories of loss functions and their relationship to each other, where the red area represents the distribution-based functions, the orange area represents the region-based functions, the green area represents the distance-based functions and the blue area represents the intersection between the categories known as compound functions.

Chapter 3

Theoretical background

3.1 Mathematical foundations

In general, learning algorithms like neural networks seek to find a optimal function $h_\theta : X \to Y$, where X and Y represent the input and output domain respectively. Therefore a loss function $\mathcal{L}$ describes the mapping of two or more values onto a nonnegative real number:

$$\mathcal{L} : h_\theta \times Y \to \mathbb{R}^{\geq 0}. \tag{3.1}$$

To find the optimal function h_θ, where θ denotes the parameterization, neural networks are trained to minimize the expected loss over a random sample set of X, a paradigm known as the Empirical Risk Minimization (ERM) [Vap92]:

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(h_\theta(X), Y). \tag{3.2}$$

The weights of the neural network are then updated proportional to the negative of the derivative of the loss function with respect to the corresponding weights and bias and the current value of the weight, known as gradient descent, which was first described by Cauchy [CAU47]:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{\partial \mathcal{L}}{\partial \theta_t}, \tag{3.3}$$

where η_t represents the learning rate and θ_t the weight at time step t. The learning rate η_t controls how much the derivative of the loss function with respect to the corresponding weights and bias influences the current weight value.

3.2 Distribution

3.2.1 Cross Entropy

In information theory, entropy [Wik21b] is defined as a measurement of uncertainty:

$$H(z) = -\sum_i p(z_i) log(p(z_i)), \tag{3.4}$$

where $p(z_i)$ represents different state probabilities. If the probability of a state is almost certain, the entropy of this state becomes low. In contrast, if the state is uncertain, the entropy value increases. The concept of entropy can be understood such that for certain events, lower effort is required to remove uncertainty, and for uncertain events, more effort is required to remove uncertainty. However, in supervised machine learning problems, the prediction of the model $\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$ and the ground truth $\boldsymbol{Y_i}$ are compared with each other for similarity. To measure the similarity or dissimilarity, the Kullback-Leibler divergence [KL51] is often used:

$$D_{KL}(\hat{Y}||Y) = \sum_i p_Y(z_i) log(p_Y(z_i)) - p_Y(z_i) log(p_{\hat{Y}}(z_i)), \tag{3.5}$$

where $p_{\hat{Y}}(z_i)$ and $p_Y(z_i)$ represent the probability functions of the prediction and ground truth domain respectively. The first term of Eq. (3.5) represents the entropy from the ground truth domain and the second term, also called cross entropy, describes how different $\hat{Y}$ is compared to Y from the perspective of Y. The definition of the cross entropy is then given by:

$$H(\hat{Y}, Y) = -\sum_i p_Y(z_i) log(p_{\hat{Y}}(z_i)), \tag{3.6}$$

The connection between cross entropy and Kullback-Leibler divergence [Wik21f] is then given by:

$$H(\hat{Y}, Y) = D_{KL}(\hat{Y}||Y) + H(Y, Y). \tag{3.7}$$

If the entropy of Y remains constant, cross entropy and Kullback-Leibler divergence are equivalent. The binary case of cross entropy is then given by:

$$\mathcal{L}_{CE}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y_i}) = -\frac{1}{M} \sum_{i=1}^{M} \boldsymbol{Y_i} log(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}) + (1 - \boldsymbol{Y_i}) log(1 - \hat{\boldsymbol{Y}}_{\boldsymbol{i}}). \tag{3.8}$$

3.2.2 Focal

Lin et al. [Lin+18] proposed the Focal loss to tackle the problem of class imbalance in datasets. Therefore, a modification of the standard cross entropy by adding a modulating factor γ was proposed:

$$\mathcal{L}_F(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}) = (1 - e^{-\mathcal{L}_{CE}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}})})^{\gamma} \mathcal{L}_{CE}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}), \tag{3.9}$$

where $\mathcal{L}_{CE}$ represents the cross entropy defined in Eq. (3.8). If γ takes the value zero, the Focal loss reduces to the cross entropy.

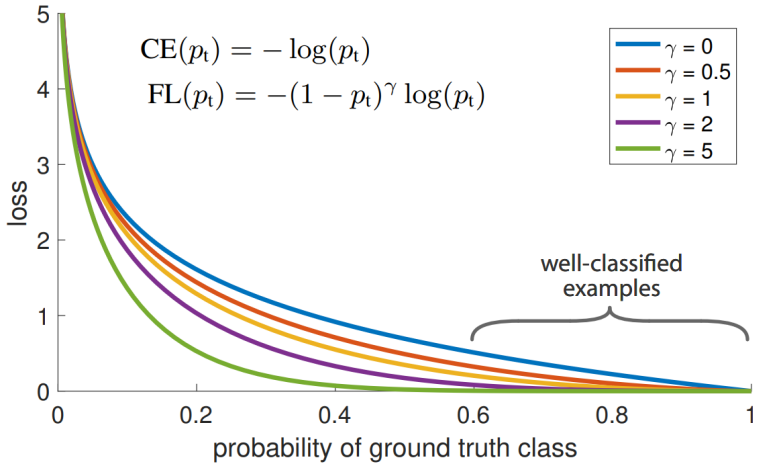


Figure 3.1: Visualization of the impact of different γ values of the Focal loss on examples compared to the standard cross entropy (CE).[1]

As shown in Fig. 3.1, as the γ value increases, the examples that have been classified well are weighted with a lower value, which results in the focus being shifted to the examples that are difficult to classify. In practice, the authors add another factor α to balance the importance of positive and negative examples:

$$\mathcal{L}_F(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}) = \alpha(1 - e^{-\mathcal{L}_{CE}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}})})^{\gamma} \mathcal{L}_{CE}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}). \tag{3.10}$$

Further, the authors showed that by correctly choosing the values of α (0.25) and γ (2.0), the performance can be increased compared to the cross entropy.

3.3 Region

In order to increase readability, the following definitions of the region-based loss functions omit the parameters that are intended to prevent a division by zero.

[1] Source: [Lin+18][accessed: 25.03.2021]

3.3.1 Dice

The Dice loss represents an approximated and differentiable variant of the Sørensen–Dice coefficient [Sør48]. The Sørensen–Dice coefficient measures the similarity between two sets by computing the overlap in relation to the sum of the cardinality of both sets:

$$D(\hat{Y}, Y) = \frac{2|\hat{Y} \cap Y|}{|\hat{Y}| + |Y|}. \tag{3.11}$$

In fact the Sørensen–Dice coefficient is a special case for the F_β score. The original formulation of the $F_{\beta=1}$ score was defined by van Rijs-bergen's [Rij79] Effectiveness function:

$$E(P, R) = 1 - \frac{1}{\frac{1}{2}P^{-1} + \frac{1}{2}R^{-1}}, \tag{3.12}$$

where P and R represent precision given by $\frac{|\hat{Y} \cap Y|}{|\hat{Y}|}$ and recall given by $\frac{|\hat{Y} \cap Y|}{|Y|}$, respectively. The full and general definition of the F_β score was later given by Chinchor [Chi92]:

$$F(P, R, \beta) = \frac{(1 + \beta^2)PR}{\beta^2 P + R} \quad (0 \leq \beta \leq +\infty). \tag{3.13}$$

The β parameter controls the balance between the precision and the recall. If $\beta < 1$ the balance shifts towards the precision, when $\beta > 1$ it favors to weight the recall more. Considering the case where $\beta = 1$, equation 3.13 reduces to Eq. 3.17, thus leading to the harmonic mean between precision and recall. Since the logical operators are non differentiable, the approximation is given by converting the predictions into probabilities and multiplying the prediction and ground truth matrix:

$$\mathcal{L}_D(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}},) = 1 - \frac{2\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \boldsymbol{Y}_{\boldsymbol{i}}}{\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} + \boldsymbol{Y}_{\boldsymbol{i}}}. \tag{3.14}$$

3.3.2 Intersection over Union

The Intersection over Union (IoU) also known as the Jaccard or Tanimoto index, as the name implies, relates the intersection of the sets to the union of the sets:

$$I(\hat{Y}, Y) = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cup Y|}. \tag{3.15}$$

In fact the IoU and the Sørensen–Dice coefficient have a monotonic connection to each other. This can be made clear by reformulating Eq. (3.17) and Eq. (3.15) so that both are dependent on the complements $\hat{Y} \backslash Y$ and $Y \backslash \hat{Y}$ respectively:

$$I(\hat{Y}, Y) = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cap Y| + |\hat{Y} \backslash Y| + |Y \backslash \hat{Y}|}, \tag{3.16}$$

$$D(\hat{Y}, Y) = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cap Y| + \frac{1}{2}|\hat{Y} \backslash Y| + \frac{1}{2}|Y \backslash \hat{Y}|}. \tag{3.17}$$

The dependence between the Sørensen–Dice coefficient and the Intersection over Union and vice versa is then given by:

$$I(\hat{Y}, Y) = \frac{D(\hat{Y}, Y)}{2 - D(\hat{Y}, Y)}, \tag{3.18}$$

$$D(\hat{Y}, Y) = \frac{2I(\hat{Y}, Y)}{1 + I(\hat{Y}, Y)}. \tag{3.19}$$

3.3.3 Tversky

The Tversky index [Tve77] generalizes both previous defined functions IoU and Dice into a single function by controlling the weightings of the asymmetric complements with parameters α and β:

$$T(\hat{Y}, Y, \alpha, \beta) = \frac{|\hat{Y} \cap Y|}{|\hat{Y} \cap Y| + \alpha|\hat{Y} \backslash Y| + \beta|Y \backslash \hat{Y}|}. \tag{3.20}$$

If α and β take the value 1/2 or the value 1, the Tversky index becomes the IoU or the Dice, respectively. By parameterizing the Tversky loss by α and β, the balance between false-positives and false-negatives can be flexibly adjusted. In order to use the Tversky index as a loss function, Salehi [SEG17] proposed the following approximation:

$$\mathcal{L}_T(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}, \alpha, \beta) = 1 - \frac{\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \boldsymbol{Y}_{\boldsymbol{i}}}{\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \boldsymbol{Y}_{\boldsymbol{i}} + \alpha \sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} (1 - \boldsymbol{Y}_{\boldsymbol{i}}) + \beta \sum_{i=1}^{M} (1 - \hat{\boldsymbol{Y}}_{\boldsymbol{i}}) \boldsymbol{Y}_{\boldsymbol{i}}}. \tag{3.21}$$

3.3.4 Asymmetric

Hashemi et al. [Has+19] proposed the Asymmetric loss, which, compared to the Dice loss, makes the weighting of the false-positives and false-negatives dependent on a parameter β:

$$A(\hat{Y}, Y, \beta) = \frac{(1 + \beta^2)|\hat{Y} \cap Y|}{(1 + \beta^2)|\hat{Y} \cap Y| + |\hat{Y} \backslash Y| + \beta^2|Y \backslash \hat{Y}|}. \tag{3.22}$$

In fact, the set formulation of the Asymmetric loss equals the formulation of the F_β score in Eq. 3.13 given by Chinchor [Chi92]. The Tversky index in Eq. 3.20 becomes the Asymmetric loss if the constraint $\alpha + \beta = 1$ holds. If the β value of Eq. 3.22 becomes zero, the Asymmetric loss changes to the definition of precision. Further if $\beta = 1$ the Asymmetric function changes to the Dice loss, if $\beta = 2$ the function takes the definition of F_2 score. For values $\beta \geq 2$, the Asymmetric loss weights recall more than precision. The differentiable loss is then given by:

$$\mathcal{L}_A(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}, \beta) = 1 - \frac{(1+\beta^2)\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \boldsymbol{Y}_{\boldsymbol{i}}}{(1+\beta^2)\sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \boldsymbol{Y}_{\boldsymbol{i}} + \sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}}(1 - \boldsymbol{Y}_{\boldsymbol{i}}) + \beta^2 \sum_{i=1}^{M}(1 - \hat{\boldsymbol{Y}}_{\boldsymbol{i}})\boldsymbol{Y}_{\boldsymbol{i}}}. \tag{3.23}$$

3.4 Distance

3.4.1 Boundary

Kervadec et al. [Ker+21] proposed the Boundary loss, which represents a distance metric on the space of approximated contours. In this approach, the integral over the boundaries is determined in comparison to the unbalanced integrals of the region-based loss functions. The differential formulation provided by the authors of the Boundary loss is given by:

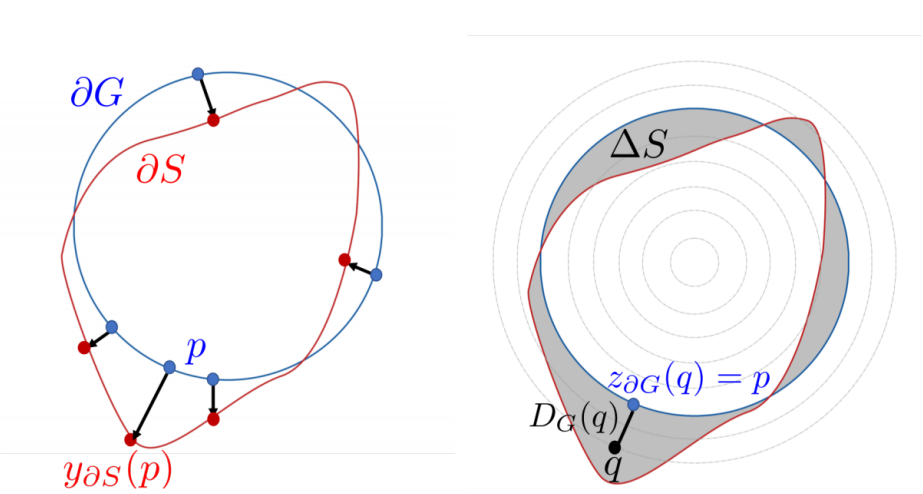
$$D(\partial G, \partial S) = \int_{\partial G} \|y_{\partial S}(p) - p\|^2 dp, \tag{3.24}$$

where ∂G and ∂S represent the boundary of ground truth and prediction respectively,p is an arbitrary point on ∂G and $y_{\partial S}(p)$ the corresponding point on ∂S and $\|.\|$ denotes the L_2 norm. Visually, one can imagine that each point p on the boundary ∂G is moved along the normal pointing outward with a velocity defined by the distance to the point $y_{\partial S}(p)$. The authors show by inspiration of the work of Boykov et al. [Boy+06] that the differential approach can be approximated by an integral approach:

$$D(\partial G, \partial S) \approx 2 \int_{\Delta S} D_G(q) dq, \tag{3.25}$$

where ΔS denotes the region between the two boundaries ∂G and ∂S and $D_G(q)$ represents the distance measure between one point q and the corresponding point based on the distance map D_G. The visual differences between the differential and integral approaches are shown in Fig. 3.2.

[2]Source: [Ker+21][accessed: 26.03.2021]



(a) Differential (b) Integral

Figure 3.2: Visualization of the difference between the differential and integral approach of the boundary loss.[2]

In the practical implementation, the distance map is computed by a level set function ϕ. This reduces the integral expression to an element-wise multiplication of the prediction of the network and the pre-computed level set function:

$$\mathcal{L}_{PB}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}, \phi) = \sum_{i=1}^{M} \hat{\boldsymbol{Y}}_{\boldsymbol{i}} \phi(\boldsymbol{Y}_{\boldsymbol{i}}). \tag{3.26}$$

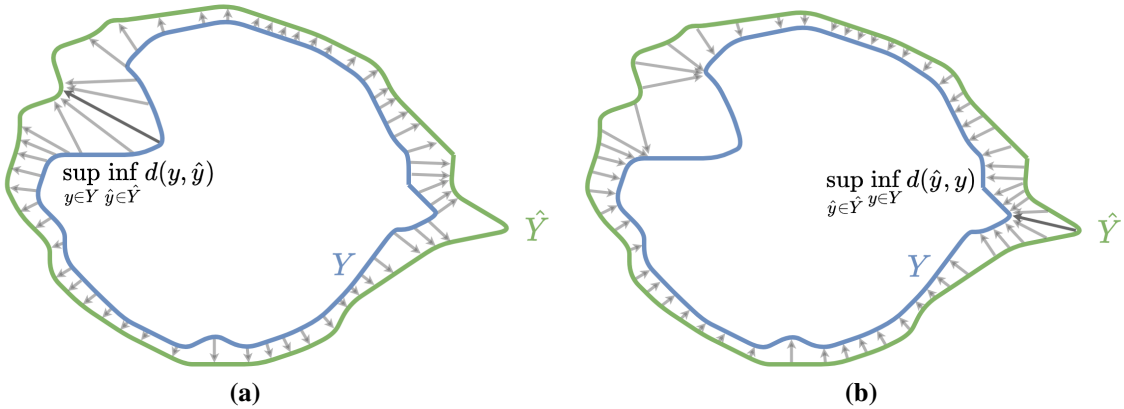
However, it should be mentioned that the Boundary loss, like almost all loss functions that are distance-based, only take a complementary role to region-based or distribution-based loss functions, because training only with distance-based loss functions leads to the network tending to predict only white areas. Therefore, the authors propose to combine boundary loss with a region-based loss function:

$$\mathcal{L}_B(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}, \phi_G) = \alpha \mathcal{L}_R(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}) + \beta \mathcal{L}_B(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y}_{\boldsymbol{i}}, \phi_G), \tag{3.27}$$

where $\mathcal{L}_R$ represents a arbitrary regional loss, $\mathcal{L}_B$ the Boundary loss and α and β are the corresponding weights.

3.4.2 Hausdorff distance

The Hausdorff distance [Wik21e] is a well-known metric to describe the distance between two sets which are subject to the assumptions of a metric space. Fig. 3.3 shows the largest distance pairs between two sets Y and $\hat{Y}$, which respectively represent the



(a) **(b)**

Figure 3.3: Visualisation of the largest distance (bold arrows) from set Y to $\hat{Y}$ (a) and vice versa from set $\hat{Y}$ to Y (b), where Y and $\hat{Y}$ represent the contour of the ground truth and prediction respectively.

ground truth and the prediction of the model. In Fig. 3.3(a) as well as in 3.3(b), but in reverse, the closest distances between the sets Y and $\hat{Y}$ are shown by grey arrows. Among the closest distances, the maximum distance, which is represented by the bold arrows, is then chosen. Finally, the Hausdorff distance is defined as the maximum distance of the two local maximum distances closest from one set to the other set:

$$HD = \max \left\{ \sup_{\hat{y} \in \hat{Y}} \inf_{y \in Y} d(\hat{y}, y), \sup_{y \in Y} \inf_{\hat{y} \in \hat{Y}} d(y, \hat{y}) \right\}. \tag{3.28}$$

In their publication [KS19], Karimi et al. presented three ways of approximating the Hausdorff distance in order to be able to use it as a differentiable loss function.

The first method uses morphological operations, for which several erosion operations with a cross-shaped structural element are applied to the symmetrical difference of the sets. The authors showed that with the right choice of the number of erosions (5 or 10), a good approximation of the Hausdorff distance is possible. This method is not computationally intensive and therefore does not impose a large overhead on the training process. However, the accuracy of this approach is strongly dependent on the number of erosions, which in turn can be different for each dataset. The second approach takes the difference of prediction and ground truth to terminate the regions that overlap and combines them with the distance maps of prediction and ground truth. The creation of the distance maps is computationally intensive and therefore leads to a longer training time, but this approach is more accurate in defining the actual distance compared to the morphological approach. The third approach presented by the authors to approximate the Hausdorff distance utilizes convolutional operations with a circular or spherical kernel, depending on the input dimension of the data. This approach also depends on several

parameters such as the radius of the kernel or a parameter α, which penalizes large error terms depending on the chosen value. Comparing the variants, the method of distance maps seems to be more intuitive, more precise and dependent on only a few parameters, although it increases the training time. Therefore, the choice fell on the second method, which uses distance maps to provide an approximation of the Hausdorff distance. The approximated pure Hausdorff distance is then given by:

$$\mathcal{L}_H(\hat{Y}_i, \boldsymbol{Y_i}, \phi) = \sum_{i=1}^{M} (\hat{\boldsymbol{Y}}_{\boldsymbol{i}} - \boldsymbol{Y_i})^2 \cdot (\phi(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}) + \phi(\boldsymbol{Y_i})), \tag{3.29}$$

where ϕ defines the level-set function given the prediction and the ground truth. Similar to the Boundary loss, the authors propose a combination of the Hausdorff distance and a region-based loss function $\mathcal{L}_R$ due to the instability of the training process only with the pure Hausdorff distance:

$$\mathcal{L}_{HD} = \alpha\mathcal{L}_R + \beta\mathcal{L}_H. \tag{3.30}$$

3.5 Compound

3.5.1 Tversky-Focal

The Tversky-Focal loss was proposed by Abraham [AK18] and combines the properties of the Tversky and Focal loss. As introduced in the previous chapters, the idea behind the focal loss is to shift the focus from easy examples to harder examples. The Tversky loss as seen in Sec. 3.3.3 generalizes regional loss functions such as Dice loss or IoU. To combine the properties, Abraham proposes the following:

$$\mathcal{L}_{TF}(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y_i}, \phi_G) = (1 - \mathcal{L}_T(\hat{\boldsymbol{Y}}_{\boldsymbol{i}}, \boldsymbol{Y_i}, \alpha, \beta))^{\frac{1}{\gamma}}, \tag{3.31}$$

where $\mathcal{L}_T$ is the Tversky loss defined in Eq. 3.21 and γ is a modulating factor as suggested in Eq. 3.9.

Fig. 3.4 shows the influence of the modulating factor γ on the resulting Tversky-Focal loss in relation to the Tversky loss. If γ takes the value one, the Tversky-Focal loss is reduced to the Tversky loss. For values $\gamma > 1$, the down-weighting of accurate predictions shifts the focus to the misclassified predictions. In general, it can be observed that adding the factor γ transforms the original linearity to a convex or concave function as the probability value increases, depending on the selection of the value. This observation is also applicable to the other region-based functions, since these also behave linearly as the ground truth probability increases. The authors showed on two

[3] Source: [AK18][accessed: 27.03.2021].

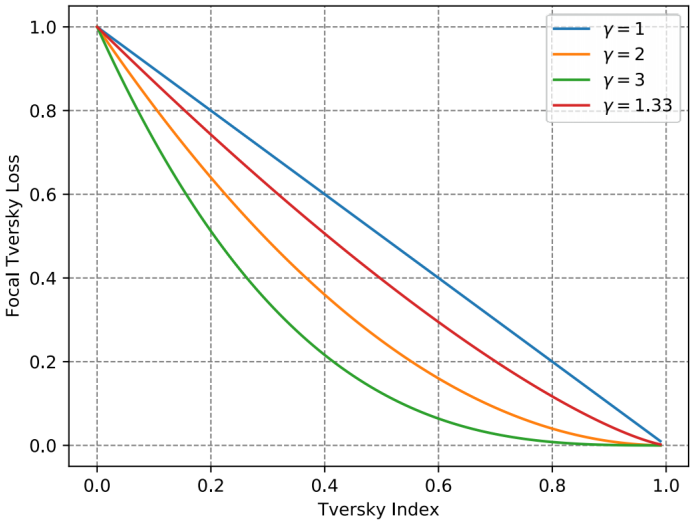


Figure 3.4: Visualization of the difference between the differential and integral approach of the boundary loss.[3]

datasets that superior results,compared to other loss functions like the Dice loss and the Tversky loss, can be obtained by the correct selection of the parameters of the Tversky-Focal loss. Therefore, the authors suggest setting $\alpha = 0.7$ and $\beta = 0.3$ to minimise false-negatives, and $\gamma = \frac{4}{3}$ in order to down-weight well segmented examples.

3.5.2 Dice-Cross Entropy

Isensee et al. [Ise+18] proposed in the publication "no new UNet" (nnU-Net) , in addition to the modification of the neural network architecture, the use of a combination of the Dice loss and the cross entropy:

$$\mathcal{L}_{DCE} = \alpha\mathcal{L}_D + \beta\mathcal{L}_{CE}, \tag{3.32}$$

where $\mathcal{L}_D$ is the Dice loss as in Eq. 3.14 and $\mathcal{L}_{CE}$ represents the cross entropy as in Eq. 3.8. The key idea is to combine the spatial information of the Dice loss and the per-pixel information of the cross entropy.

3.5.3 Dice-Focal

Iantsen [IVH21] proposed the combination of Dice and Focal loss by using the unweighted sum:

$$\mathcal{L}_{DF}) = \mathcal{L}_D + \mathcal{L}_F, \tag{3.33}$$

where $\mathcal{L}_D$ is the Dice loss as in Eq. 3.14 and $\mathcal{L}_F$ is the Focal loss as in Eq. 3.9. The authors demonstrated through their success in the "Automatic Head and Neck Tumor Segmentation (HECKTOR) Challenge" at MICCAI 2020 , initiated by Andrearczyk *et al*. [And+21], that the combination of Dice and Focal loss can produce robust results. However, it is worth mentioning that no comparison was made to other loss functions, and it is therefore difficult to assess how the loss function contributed to the good success. The idea is similar to the one in Sec. 3.5.2 that the distribution-based loss function, which in this case is the Focal loss, provides complementary information per pixel.

3.6 Analysis

In the following visual analysis of the loss functions based on several practical examples, the advantages and disadvantages of the previously categorised loss functions will be highlighted. For the sake of simplicity, the analysis only considers the case of binary segmentation where the focus is on foreground matching. In the following series of images, the ground truth is represented by blue pixels and a segmentation by an orange pixel; the blue edges in the prediction indicate the shape of the ground truth. The results of the individual loss functions are denoted in the corresponding tables. It is important to note that the analysis takes place on 10x10 images, and therefore the effects of the individual analyses scale with the increase in dimension size.

3.6.1 Shape size

In this example, the influence of different structure sizes on the output of the loss function is to be shown. Fig. 3.5 shows two ground truth structures with extreme differences in structure size and two predictions each containing one and two wrongly segmented pixels respectively.

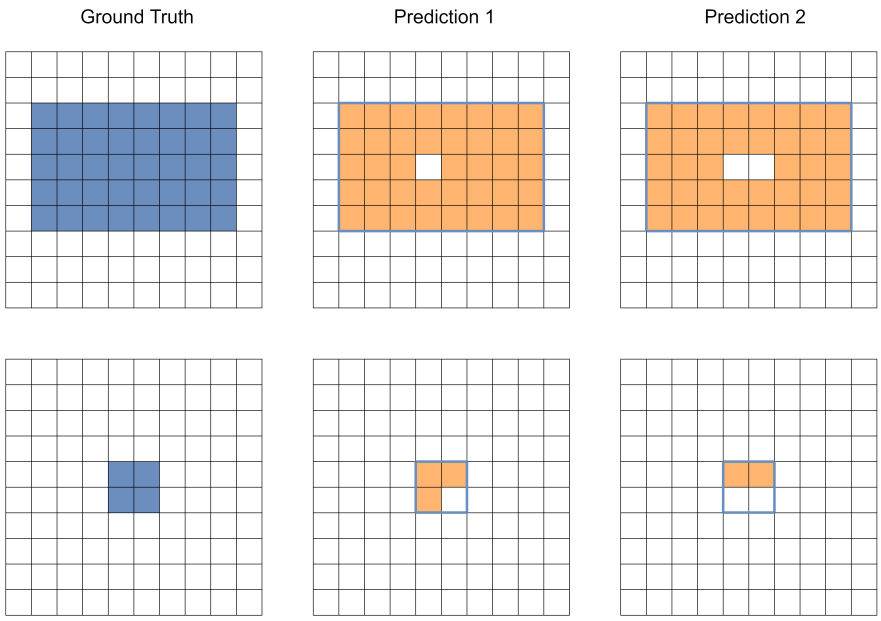


Figure 3.5: Exemplary use cases to investigate the influence of different shape sizes on the loss functions.

Table 3.1 shows the results of the loss functions for the different predictions of the two shape sizes. It is noticeable that for region-based loss functions the loss values show

a significant difference between large and small shape size. The direct comparison of the two predictions for the small structure sizes also shows that a wrong segmentation can lead to a high oscillation of the loss value. For the cross entropy it shows that a fault segmentation in a large structure has the identical value as in a small structure. However, the Focal loss function shows the effects of the modulating factor γ, which leads to a small difference in this use case. Due to the one-sided consideration of the Boundary loss function, the loss value of the associated predictions does not change, but a difference between the structural sizes is still recognisable. For the Hausdorff distance, only a minimal difference is shown for the example of the small shape size due to the two-sided distance consideration.

The more natural assumption for a loss function is the cross entropy, which evaluates each error equally regardless of structure size. This may explain why the cross entropy is preferred on datasets containing natural images, which usually have a proper balance between classes. The region-based loss functions, on the other hand, have a bias towards smaller structures or unbalanced class distributions, which is why they are preferred for medical data as these often contain a high class imbalance.

Shape size						
Loss Function	Large			Small		
	Pred. 1	Pred. 2	Difference	Pred. 1	Pred. 2	Difference
Asymmetric	0.0175	0.0354	0.0178	0.1887	0.4110	0.2223
BCE	1.0614	1.0804	0.0189	0.7470	0.7660	0.0189
Focal	0.1162	0.1179	0.0010	0.1840	0.1861	0.0020
Dice	0.0126	0.0256	0.0129	0.1428	0.3332	0.1904
DiceFocal	0.0377	0.0640	0.0263	0.2323	0.4386	0.2062
Tversky	0.0176	0.0355	0.0179	0.1891	0.4117	0.2225
TverskyFocal	0.0258	0.0440	0.0181	0.1655	0.3329	0.1673
Boundary	0.5146	0.5146	0.0000	1.5868	1.5868	0.0000
Hausdorff	0.1000	0.2000	0.1000	0.0199	0.0399	0.0200

Table 3.1: The individual loss values and the corresponding differences of different loss functions of the prediction and the ground truth for the analysis of the shape sizes.

3.6.2 Shape awareness

This example measures the influence of predictions that have differences in shape on the loss functions. For this purpose, two predictions have been generated in Fig. 3.6, whereby the first prediction is very similar to the actual ground truth and only one translation is necessary to match the ground truth. The second prediction in Fig. 3.6, however, does not resemble the actual ground truth in any respect to the shape. The left column of Table 3.2 shows the corresponding loss values of the individual loss functions for prediction 1 and prediction 2 of Fig. 3.6.

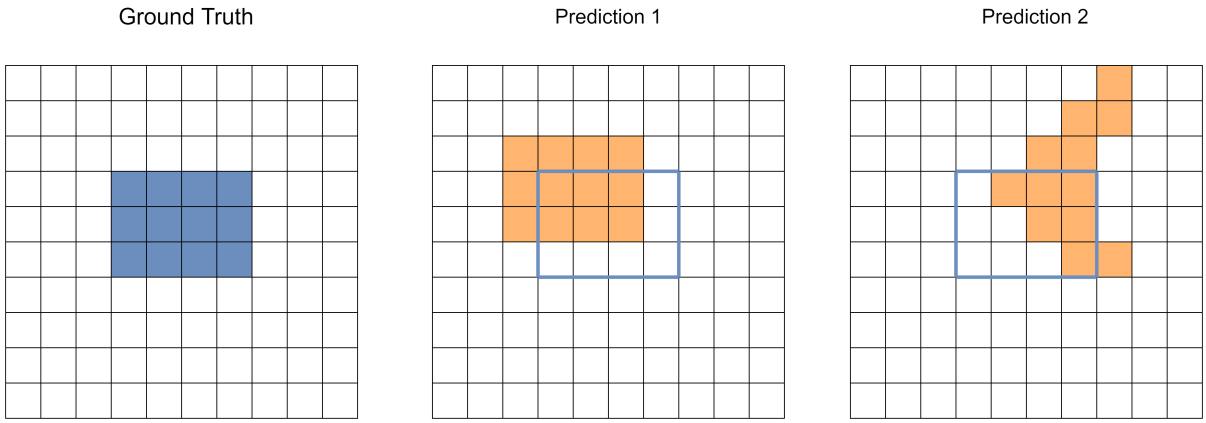


Figure 3.6: Exemplary use cases to investigate the influence of different shape structures on the loss functions.

It is noticeable that for region-based and distribution-based loss functions there appears to be no difference between prediction 1 and prediction 2. For distribution-based loss functions, this can be explained by the fact that only individual pixels are considered without spatial context. The region-based loss functions only measure the relative overlap, and since this is equal in area in prediction 1 and prediction 2, no difference is made here between prediction 1 and prediction 2. Considering the distance-based loss functions, it is noticeable that a difference is made between prediction 1 and prediction 2. The values of the predictions are closer to each other in the case of boundary loss because only a one-sided view is taken, and further apart in the case of the Hausdorff distance because a two-sided view is taken. The observations of this example show that distance-based loss functions can contribute to promoting forms that strongly resemble ground truth in terms of structure.

3.6.3 Over- and underfitting

This example is intended to show the influence of predictions on the loss functions, which either over- or underfit the shape of the ground truth. For this purpose, two pre-

Shape awareness			Shape fitting		
Loss Function	Pred. 1	Pred. 2	Loss Function	Pred. 1	Pred. 2
Asymmetric	0.5000	0.5000	Asymmetric	0.6767	0.2740
BCE	0.9490	0.9490	BCE	1.060	0.9568
Focal	0.1982	0.1982	Focal	0.1982	0.1982
Dice	0.4999	0.4999	Dice	0.5999	0.3846
DiceFocal	0.5945	0.5945	DiceFocal	0.6817	0.4883
Tversky	0.4999	0.4999	Tversky	0.6774	0.2727
TverskyFocal	0.5945	0.5945	TverskyFocal	0.5709	0.5646
Boundary	1.1806	1.1898	Boundary	0.2800	0.4399
Hausdorff	0.2599	0.4900	Hausdorff	1.0098	1.0599

Table 3.2: The individual loss values of different loss functions of the prediction and the ground truth for the analysis of the shape awareness (left column) and for the analysis of the shape fitting (right column).

dictions with corresponding ground truth were generated in Fig. 3.7, whereby the first prediction does not completely fill the ground truth, i.e. underfitting it, and the second prediction segments the complete area of the ground truth and also falsely segments surrounding pixels.

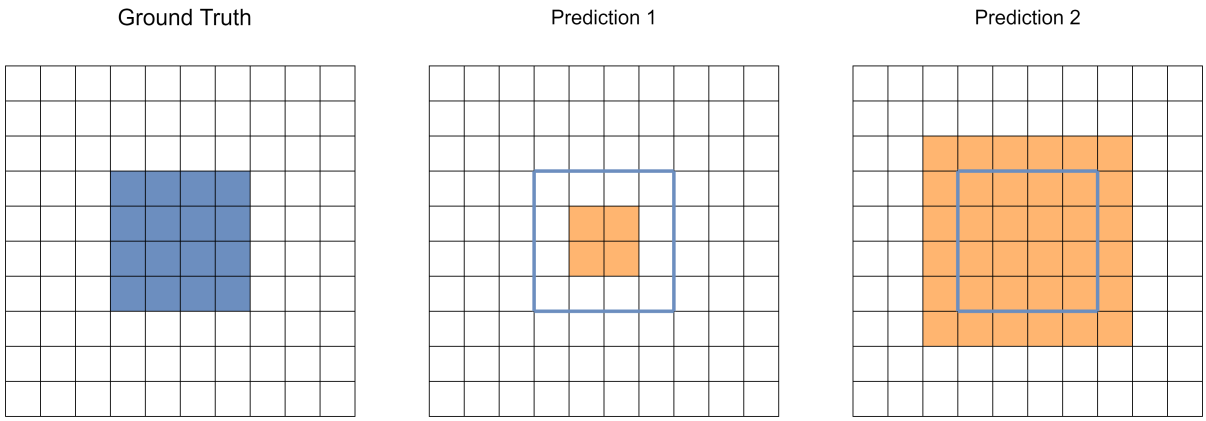


Figure 3.7: Exemplary use cases to investigate the influence of shape fitting on the loss functions.

The corresponding results of the loss functions are given in the right column of Table 3.2. When looking at the loss values, it is noticeable that the region-based loss functions generally prefer, albeit with differences, prediction 2, which oversegments the ground truth. Due to the different weights of the FP and the FN in the Asymmetric loss, it can be observed that prediction 2 is preferred with a large difference compared to prediction

1. The different weighting of the FP and the FN with the Tversky loss also leads to a preference for prediction 2 over prediction 1. If the unweighted variant, i.e. the Dice score, is now considered, it is noticeable that the differences between the loss values of prediction 1 and prediction 2 become smaller, this is because the pixels belonging to the FP in prediction 2 are still in the majority compared to the FN in prediction 1. For the Tversky-Focal loss, the difference between prediction 1 and prediction 2 is shown to be very small due to the change in linearity to a convex function.

Chapter 4

Learning loss functions

This chapter deals with the question of how loss functions can be learned directly from the given data. For this purpose, two approaches are examined and compared in the presentation of the results. The first approach uses genetic algorithms to search for suitable loss functions before the actual training. The second approach uses the idea of meta-learning to establish a helper network during training that takes over the task of the loss function.

4.1 Genetic algorithms

Genetic algorithms seek to find near optimal solutions by searching the possible space based on an underlying heuristic. It is inspired by Charles Darwin's "Theory of Natural Selection" [Dar59], in which the individual from a population that best adapts to the conditions of the environment has a higher probability of survival. The surviving individuals then produce offspring and the same procedure is repeated until the environment collapses or one individual dominates all others. Genetic algorithms simulate this process by deriving specific operators. The operators derived from evolutionary theory are selection, mutation, crossover and indirectly also evaluation.

Recently, attempts have been made to combine genetic algorithms with methods in the area of deep learning. Miikkulainen et al. [Mii+17] proposed to automate the architecture and hyperparameter search by using genetic algorithms . Others such as Loshchilov [LH16] compared state-of-the-art bayesian optimization algorithms and a special variant of the evolutionary strategy, the Covariance matrix adaptation evolution strategy (CMA-ES) method [HO01], for neural network hyperparameter optimization and showed that they yield comparable results. Stanley et al. [Sta+19] hypothesized that by applying genetic algorithms to the architecture search of neural networks, they could achieve human-level results in the next decades with the increase of performance capabilities. Gonzalez [GM20] proposed in his publication to use genetic algorithms

with the CMA-ES method not only for architecture or hyperparameter search, but much more for the search of new loss functions that improve the training of neural networks and their performance.

The proposed idea of Gonzalez [GM20] generates a population of syntax trees, also called individuals, which can be compiled into loss functions $\mathcal{L}_T$. These syntax trees can be modified by the genetic operators and evaluated. The syntax trees that produce the best functions are then selected and trained and evaluated against other loss functions. In the following, the individual processes of the genetic algorithm are defined in chronological order and on the basis of Fig. 4.1.

Primitive set represents the basic skeleton for the generation of the syntax trees. The set consists of different nodes. The internal nodes also called primitives are user-defined operators such as simple arithmetic rules. Terminal nodes split up into two subtypes, the constant nodes and the argument nodes. Constant nodes are predefined with either fixed values or randomly chosen values within a certain range. Argument nodes represent the input given by the user. The input parameters in the form of matrices consist of the prediction of the network $\hat{\boldsymbol{Y}}$ and the associated ground truth $\boldsymbol{Y}$. This offers the genetic algorithm the highest flexibility and freedom to generate suitable combinations, but due to the large search space this might not lead to convergence of an optimal value. Primitive sets can either be loosely or strongly typed. The difference is that the primitives and terminals of a strongly typed primitive set are associated with a special type, whereas this is not the case with the loosely type set. In addition, each primitive and terminal also receives the attribute of the probability of occurrence, which plays an important role in the initialization process. An example of the composition of the primitives and terminals of a primitive set into a syntax tree can be seen in Fig. 4.2.

Initialization describes the process of generating the initial population of individuals based on the previously defined primitive set. Typically the generation of the syntax trees is done by randomly sampling primitives and terminals from the primitive set, but with the addition of the probability of occurrence, the random sampling changes to a weighted random sampling. The depth of the syntax tree is constrained by a minimum and maximum depth of each leaf node. If the primitive set is strongly typed, the generation ensures that the type constraints are respected under the condition that a possible path from input type to output type exists.

Mutation describes the change of one component of the syntax tree. The mutation changes a node, taking into account the type restrictions, by sampling out of the primitive set. The aim of this change is to add randomness to the generation process, this plays an essential role, otherwise the possible combinations of primitive and terminal nodes would be limited to the space of the population created in the initialization step.

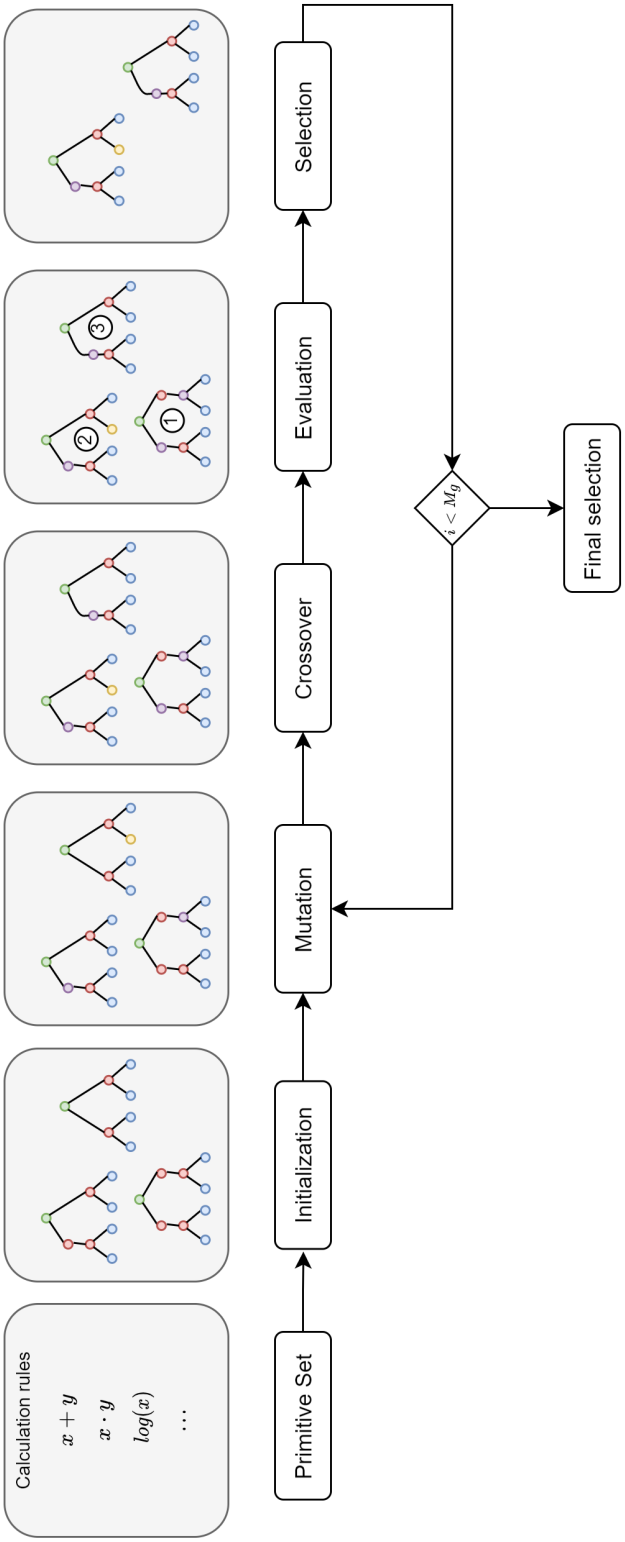


Figure 4.1: Overview of the simplified genetic algorithm process.

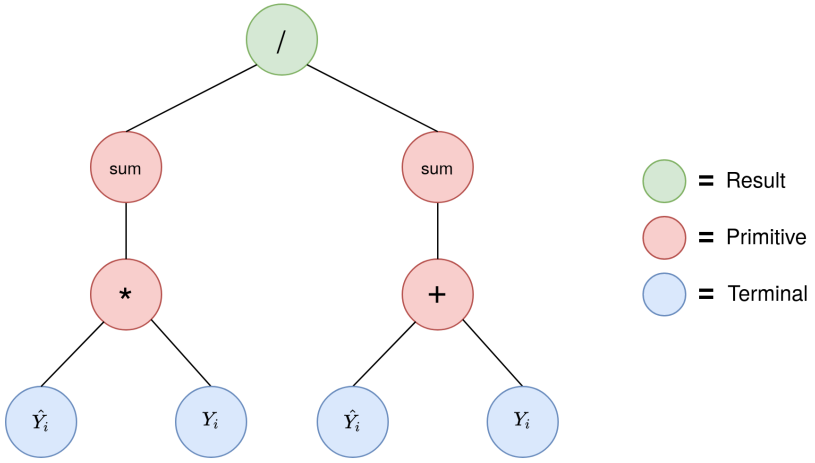


Figure 4.2: Visualisation of one randomly generated tree.

Crossover describes the mating of two individuals from the population. The mating produces two new children, which both contain sub trees of the parent trees. A position in both syntax trees is determined randomly, at which point the sub-trees are swapped in compliance with the type restrictions. The crossover process is illustrated in Fig. 4.3, the yellow arrow implies the randomly chosen position where the pairing takes place.

Selection describes the process of reducing the population based on an underlying strategy. The selected strategy takes the k best individuals from the population, where k is a predefined value. The aim of this selection is that only those individuals that have achieved an above-average result pass on their traits to their offspring.

Evaluation describes the process of assigning a fitness individual to each individual of the population. The fitness value expresses how well the individual is doing in relation to the problem. In the initialisation step the fitness value is set to -1.0 or 1.0, depending on whether it is a maximization or minimization problem. Normally, the fitness value is determined by a function, but in this case only an approximation is possible due to the inherent complexity. Therefore, the evaluation is divided into several sub-processes that test different hypotheses and assign corresponding fitness values. The general process of evaluation for a syntax tree can be seen in Fig. 4.4 and is defined chronologically below.

Inspection defines the examination of a syntax tree for certain predefined rules. These rules differ depending on the input parameters and provide a positive or negative contribution to the fitness value depending on the assessment. One of these rules ensures that at least one dependency must exist between different input parameters; if this is not

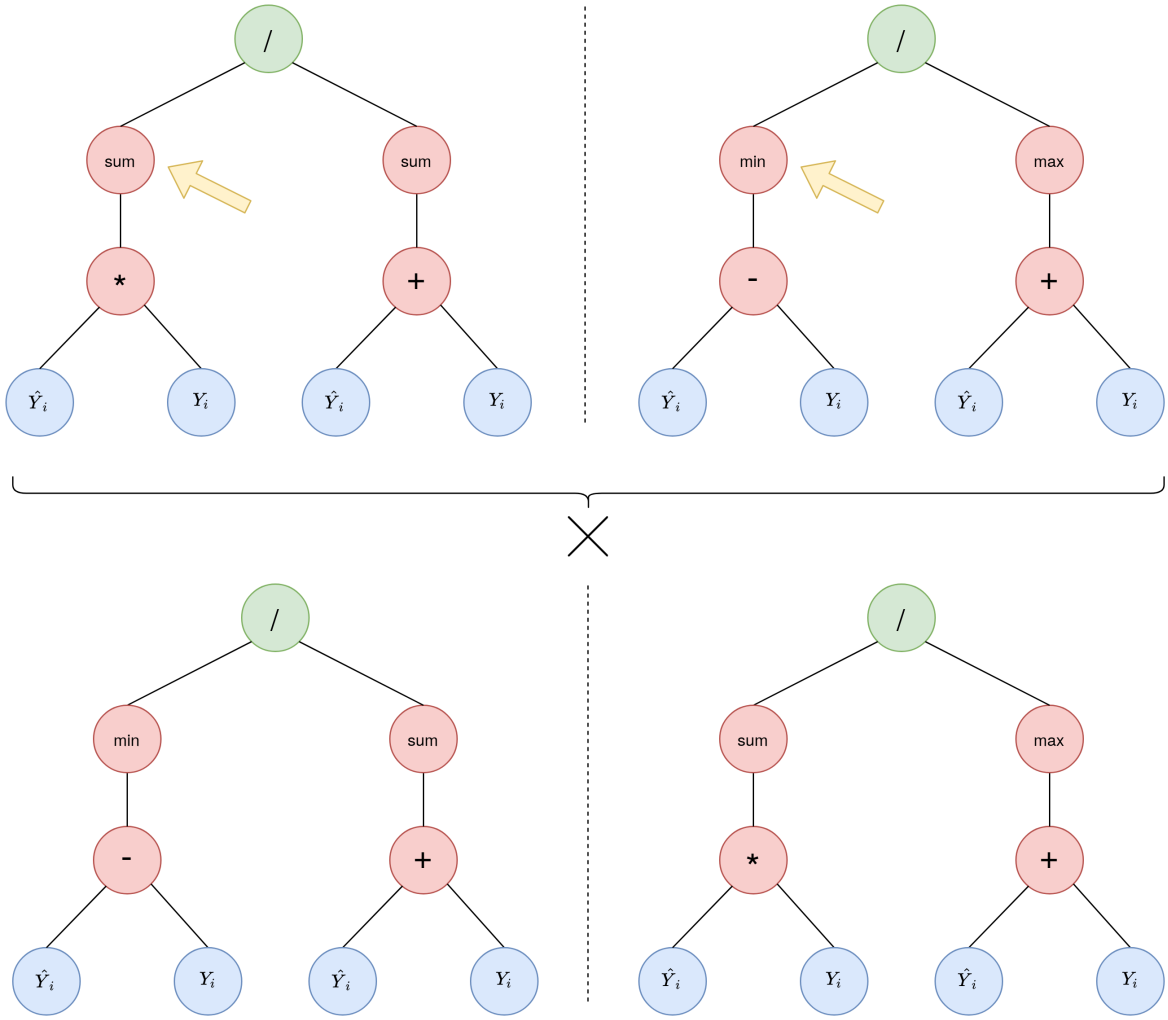


Figure 4.3: Exemplary application of the crossover operation to two arbitrarily generated trees.

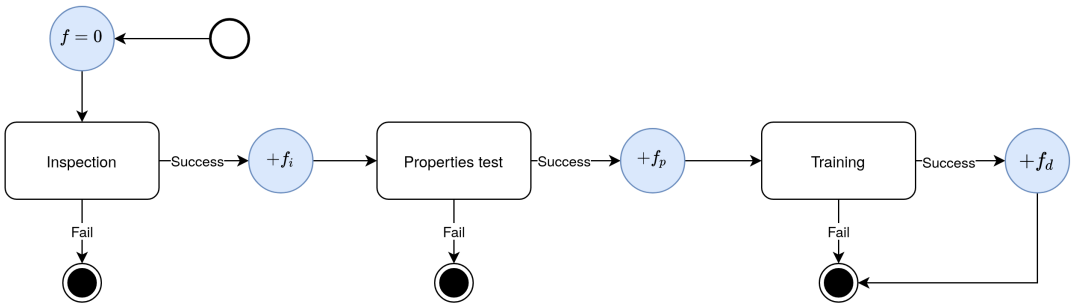


Figure 4.4: Procedure for the evaluation of one syntax tree.

the case, the entire evaluation process for this tree is terminated and the corresponding fitness value is assigned. Furthermore, the number of terminal and primitive operators is also evaluated; if an operator occurs disproportionately often compared to the other operators, this is expressed by a negative contribution to the fitness value. Each result is summed up to f_i at the end and added to the existing fitness value in case of a successful inspection.

Properties test describes the process of testing the compiled version of the syntax tree with synthetic data for mathematical properties. The synthetic data simulate in different ways the network prediction that improves over time to terminate in advance functions that do not satisfy the mathematical properties. The synthetic data is generated with noise. At the beginning, a random ground truth $\boldsymbol{Y}$ is selected from the set Y. Over a predefined period of time, the ground truth is multiplied by a noise matrix $\boldsymbol{N}$ that is regulated by the time parameter t, resulting in the synthetic data $\hat{\boldsymbol{Y}}_{\boldsymbol{n,t}}$:

$$\hat{\boldsymbol{Y}}_{\boldsymbol{n,t}} = (1 - t)\boldsymbol{N}\boldsymbol{Y}. \tag{4.1}$$

The compiled function of the syntax tree is then executed with the input parameters, which returns the value $l_{n,t}$:

$$l_{n,t} = \mathcal{L}_T(\hat{\boldsymbol{Y}}_{\boldsymbol{n,t}}, \boldsymbol{Y}). \tag{4.2}$$

Subsequently, the resulting data is checked for monotonicity properties, if the data has a monotonically decreasing property, the generated function has passed the test and an additional value in the form of f_p is added to the existing fitness value. In addition, the upper bound and lower bound are calculated to normalise the loss functions in the training step to the range of [0-1], as too high values would lead to no convergence of the network.

Training defines the process whereby the compiled function is tested in conjunction with the neural network. Therefore, the network is trained with the compiled function in a predefined period of time. In each epoch, the dice score that the network achieves on the validation data is stored. If the compiled function generates NaN or Inf values in the course of the training, the training is terminated immediately and the function is assigned the fitness value achieved up to that point. At the end, the mean value of the collected dice values is calculated and applied as f_d to the existing fitness value.

Statistics of each generation are created at the end after the offspring have been created.

The previously defined processes, with the exception of the generation of the primitive set and the initial population, are repeated M_g times. In the end, the k best individuals are selected from the population and undergo the same experiment as the other loss functions.

4.2 Meta-Learning

Meta-learning describes the idea of achieving a higher degree of generalisation on the basis of fewer training examples and by adaptively changing the training process based on sampling different tasks. The first and only ones to apply meta-learning to the field of loss functions were Bechtle et al. [Bec+21] with their proposal to integrate a helper network, which they denote as meta-loss network, that takes over the task of the loss function. The idea of Bechtle et al. can be seen in Fig. 4.5 and is explained on the basis of this figure.

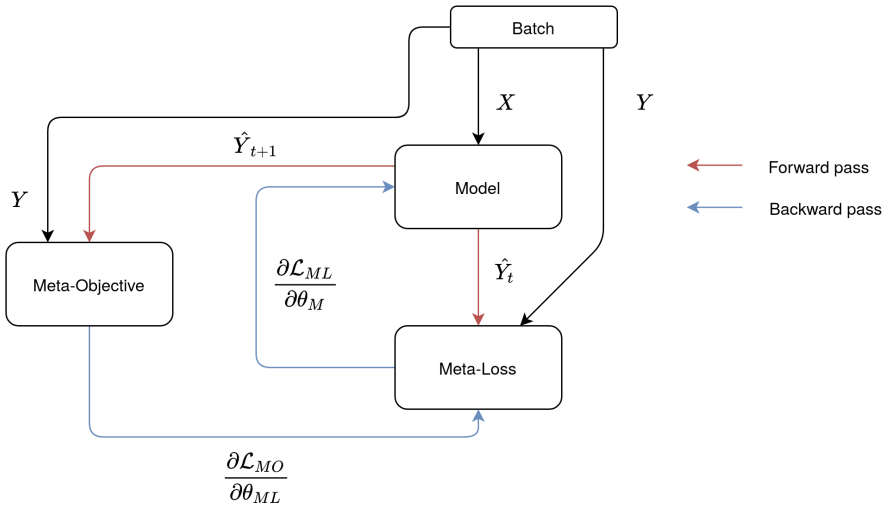


Figure 4.5: Overview of the meta-learning approach proposed by [Bec+21].

For each training step, the batch consisting of the input set X and the target set Y is randomly drawn. The input set X is transformed into the prediction set $\hat{Y}$ by the segmentation network h_θ. As with standard loss functions, the input to the meta-loss network is the prediction set $\hat{Y}$ and the target set Y. The output of the meta-loss network is a real positive number, which is ensured by the network architecture of the meta-loss network. The network architecture of the meta-loss network can be seen in Fig 4.6, it consists of a sequence of linear layers that reduce the input dimension to a real number, and two activation functions ReLU and SoftPlus[1] that ensure that the result does not become negative. It should be mentioned that a network architecture consisting of convolutional operations was also tested, but was not found to be a suitable choice due to lack of memory. Therefore, the final choice fell on the linear layers in combination with the activation functions.

[1] https://pytorch.org/docs/stable/generated/torch.nn.Softplus.html

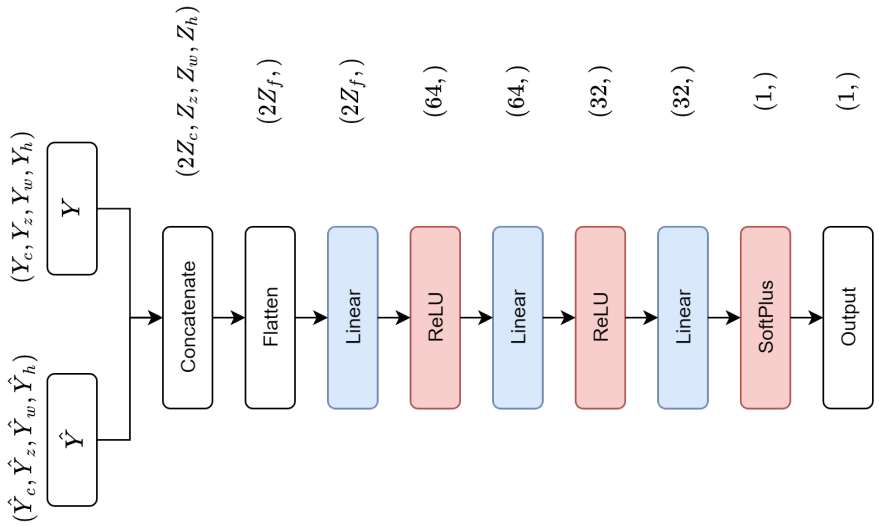


Figure 4.6: Visualization of the composition of the meta-loss network with corresponding dimension.

Based on the output of the meta-loss network, the parameters of the segmentation network are updated by a backward pass. In contrast to normal loss functions, the entire parameters of the meta-loss network must be derived on the basis of output. With the updated network h_θ, a new forward pass is performed with the same input data as in the first forward pass, resulting in the prediction set $\hat{Y}_{t+1}$. To evaluate the prediction, the authors introduce a meta objective to measure how accurate the prediction $\hat{Y}_{t+1}$ is. The authors suggest that the meta objective for regression tasks should be the mean-squared error loss, but since the given task , the loss functions presented in Chapter 3 are used in this case. On the basis of the output of the meta-objective, the parameters of the meta-loss network are updated in the backward pass.

Chapter 5

Optimize loss functions

This chapter deals with the question of how to optimise loss functions. A distinction is made between two groups of methods. The first group of methods deals with the question of how to adaptively change learnable parameters such as those of the Tversky loss, which was presented in Chapter 3. The second method group focuses on the question of whether the adaptive weighting of individual loss functions in a multi loss function can lead to better results.

5.1 Greedy algorithm

In their publication, Seo et al. [SBX20] proposed two methods for adaptively changing the learnable parameters of a loss function during the training process. For this purpose, the authors modelled a function that is a three-dimensional parameterized version of the Effectiveness function presented in Sec. 3.3.1:

$$\mathcal{L}_E = 1 - \frac{1+\gamma^2}{\frac{1}{P_\alpha} + \frac{\gamma^2}{R_\beta}}, \tag{5.1}$$

where P_α and R_β represents precision and recall parameterized by α and β, respectively, and γ a parameter to control the influence of the recall. The first method presented, which is also employed in this thesis, is based on a brute-force method and is therefore referred to as "greedy method". The second method is a heuristic approach derived from the results of the first method. Due to the fact that the second method is only a derived version of the results of the first method, the results cannot be transferred one-to-one to different data and therefore this approach is not considered further.

In the following, the idea of the greedy approach of Seo et al. is described and explained on the basis of Fig. 5.1. Discrete actions A are defined even before the actual training. These actions A act on the learnable parameters p of a loss function $\mathcal{L}$ such as increasing by 10%, decreasing by 10% or leaving the value unchanged. The

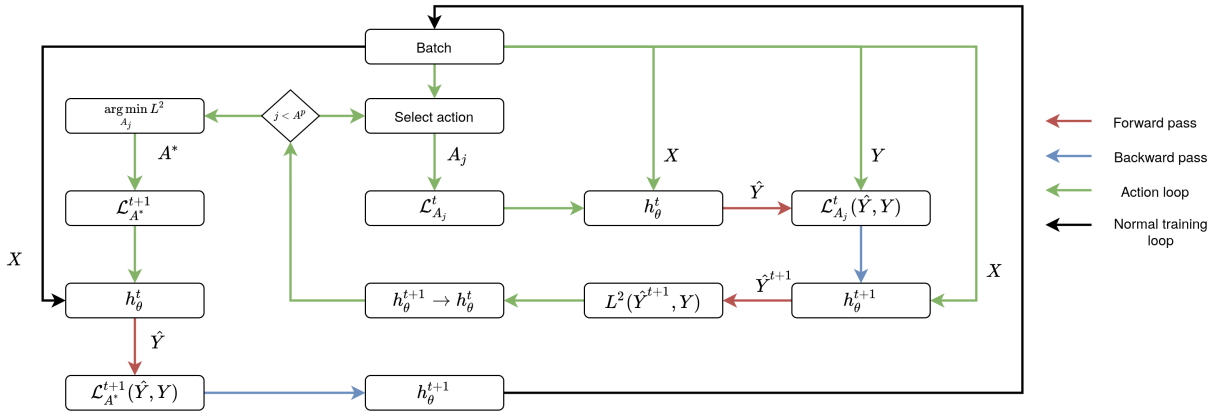


Figure 5.1: Overview of the greedy approach following the publication of Seo et al. [SBX20].

cartesian product provides all A^p possible combinations. For each training step t, all possible action combinations are tested. In the first step, an action A_j is selected and applied to the parameters of the loss function $\mathcal{L}^t_{A_j}$. Based on the raw input data, the model h^t_θ generates the prediction $\hat{Y}$. Then the loss value is formed by the prediction $\hat{Y}$ and the ground truth Y. The parameters of the model are updated based on the loss value in the backpropagation step, resulting in the model h^{t+1}_θ. Then, a new forward pass is performed with the same raw input data, which leads to the prediction $\hat{Y}^{t+1}$. The L^2 norm of prediction $\hat{Y}^{t+1}$ and ground truth Y is formed and stored. In the last step of the inner action loop, the parameters of the loss function and the parameters of the model are reset to the initial state so that the actions and the optimisation steps are not accumulated. After all possible combinations have been tested, the optimal action $A*$ that produced the minimum L^2 norm is now selected. The optimal action $A*$ is now finally executed on the parameters of the loss function, resulting in $\mathcal{L}^{t+1}_{A*}$. In the last step, another forward pass followed by a backward pass is performed, leading to the final optimization of the parameters of model h^{t+1}_θ. The process just described is repeated for each training step.

This approach is highly exploratory and exploiting, as all possible combinations are tested, and is therefore very costly in terms of runtime. The runtime for each training step increases by the factor of A^p.

5.2 Adaptive weighting

As already shown in Sec. 3.5, many loss functions are a composition of different loss functions. However, the individual weights of each loss function are usually approximated by Bayesian optimisation algorithms such as grid search or random search

[BB12], or by manual testing. However, these methods are very time-consuming and remain static among the training process. Adaptive weighting methods, on the other hand, aim to change the weightings of the loss function depending on a basic heuristic during the training process:

$$\mathcal{L}_{AW} = \sum_{i}^{M} \alpha_i \mathcal{L}_i, \tag{5.2}$$

where $\mathcal{L}_i$ is a arbitrary loss function and α_i the corresponding weight. In the following subsections, three methods are presented that use different approaches to calculate the weight α_i.

5.2.1 Rebalance

The rebalance strategy (RS) was additionally presented in the original publication [Ker+21] by the authors of the Boundary loss function, as introduced in Chap. 3.4. The strategy defines a linear interpolation by a parameter α which is increased by a constant value in each training step t:

$$\mathcal{L}_{RB} = \alpha \mathcal{L}_x + (1 - \alpha)\mathcal{L}_y, \tag{5.3}$$

where $\mathcal{L}_x$ and $\mathcal{L}_y$ are arbitrary loss functions, that complement each other, with the corresponding weightings of α and $1 - \alpha$. The constant value is calculated on the basis of the maximum number of training steps, taking care that the distance-based loss functions do not dominate the region-based loss functions in the initial phase of the training process. The idea behind the strategy is that in the initial phase, the dominance of the region-based loss functions creates a coarse segmentation, which is then improved by the dominance in the final phase of the distance-based loss functions with the segmentation of finer structures.

5.2.2 Soft Adapt

Heydari proposed the SoftAdapt (SA) method [HTM19], which makes use of the gradient and step information of the prior loss information to determine each future weight α_i. The adjustment by gradient information or also by step information in order to optimize the hyperparameters of the training process has already been addressed in various papers such as Adam [KB17] and Adagrad [DHS11]. The approach of Heydari changes the weights based on the approximation of the forward finite difference and normalises them by the sum of all weights:

$$\alpha_t^i = \frac{e^{\beta s_t^i}}{\sum_l^n e^{\beta s_t^l}}, \tag{5.4}$$

where t represents the time step, i is the ith-component of the compound loss, s_t^i is the approximation of the forward finite difference and β a modulating factor parameter to control the influence of the approximation of the rate of change. The name of the method is derived from the fact that the calculation of the weights generally represents the Softmax function [Wik21g]. The rate of change is calculated in Heydari's original paper by a simple subtraction of the current loss and the previous loss $\mathcal{L}_t^i - \mathcal{L}_{t-1}^i$, but more accurate approximations of the forward finite difference exist that can be used with a higher degree of prior information [Wik21d]. As the authors point out, for values $\beta > 0$ the focus is on the worst loss function, if $\beta < 0$ the focus is on the best loss function and for $\beta = 0$ the focus is uniformly distributed. The authors present another variant in which the current value of the loss function is included:

$$\alpha_t^i = \frac{\mathcal{L}_t^i e^{\beta s_t^i}}{\sum_l^n \mathcal{L}_t^l e^{\beta s_t^l}}. \tag{5.5}$$

5.2.3 Coefficient of Variations

Groenendijk et al. [Gro+20] proposed a method that is based on the on the coefficient of variations (COV) and utilizes information from past loss values. The authors develop their method on the basis of the hypothesis that "a loss term has been satisfied when its variance has decreased towards zero" [Gro+20, p. 3] . Therefore, the authors suggest using the coefficient of variations or also known as relative standard deviation:

$$c_{\mathcal{L}} = \frac{\sigma_{\mathcal{L}}}{\mu_{\mathcal{L}}}, \tag{5.6}$$

where $\sigma_{\mathcal{L}}$ is the standard deviation of the loss and $\sigma_{\mathcal{L}}$ the mean of the loss. The advantage of the relative standard deviation is that it is independent of the size and scale of the observations and can therefore also be used with unnormalised loss functions such as the distance-based loss functions from Sec. 3.4. The authors suggest that in order to calculate the relative standard deviation correctly, it is not derived directly from the loss values, but based on a ratio-scale:

$$\ell_t = \frac{\mathcal{L}_t}{\mu_{\mathcal{L}_{t-1}}}, \tag{5.7}$$

where $\mathcal{L}_t$ is the current loss value and $\mu_{\mathcal{L}_{t-1}}$ is the mean value of the loss function observed until timestep $t-1$. Compared to other methods such as GradNorm [Che+18], where $\ell = \frac{\mathcal{L}_t}{\mathcal{L}_0}$, or [LJD19] , where $\ell = \frac{\mathcal{L}_t}{\mathcal{L}_{t-1}}$, the authors find that the proposed method is more robust and cope more effectively with outliers. To calculate the weights α_i of the single loss functions, the authors propose to calculate the relative standard deviation based on the ratio scale ℓ_i:

$$\alpha_t^i = \frac{1}{z_t} c_{\ell_t^i} = \frac{1}{z_t} \frac{\sigma_{\ell_t^i}}{\mu_{\ell_t^i}}, \tag{5.8}$$

where t is the time component, i is the ith single-loss function in the multi-loss function and $z_t = \sum_i c_{\ell_t^i}$ is the sum of all relative standard deviations to ensure that all weights are normalized. To calculate the ratio-scale ℓ_t and the relative standard deviation c_ℓ, the authors propose to use an approximation of the mean and standard deviation of the loss function using the Welford's algorithm [Gro+20] [WW62]:

$$\mu_{\mathcal{L}_t} = (1 - \frac{1}{t})\mu_{\mathcal{L}_{t-1}} + \frac{1}{t}\mathcal{L}_t, \tag{5.9}$$

$$\mu_{\ell_t} = (1 - \frac{1}{t})\mu_{\ell_{t-1}} + \frac{1}{t}\ell_t, \tag{5.10}$$

$$M_{\ell_t} = (1 - \frac{1}{t})M_{\ell_{t-1}} + \frac{1}{t}(\ell_t - \mu_{\ell_{t-1}})(\ell_t - \mu_{\ell_t}), \tag{5.11}$$

where $\sigma_{\ell_t} = \sqrt{M_{\ell_t}}$ is the approximated standard deviation of ℓ_t. By calculating the mean and standard deviation of ℓ_t, each weight α_t can now be determined from Eq. 5.8 for each time step t.

Chapter 6

Datasets

In this chapter, the two given Magnetic Resonance Imaging (MRI) datasets from cervical and lumbar vertebrae are presented, the preprocessing steps and the chosen methods for data augmentation are justified.

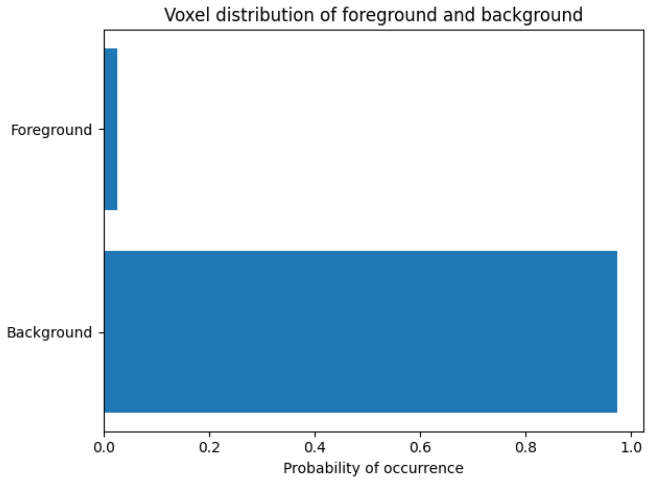
6.1 Cervical

The cervical vertebrae form the upper end of the entire spine. There are seven cervical vertebrae in total, the uppermost cervical vertebra is called atlas and connects the head with the spine, the lowermost cervical vertebra links to the first thoracic vertebra. The cervical dataset was provided by the VisSim[1] research group of the University Koblenz-Landau. The data base contains the MRI volume data of entire cervical vertebrae and the corresponding ground truth data of 13 subjects. The information on how the MRI experiment proceeded and the associated data of the subjects, are unfortunately not available due to an unknown data situation. The volume data was tri-linearly interpolated from the median dimension [256,512,512] to the uniform dimension of [32,224,224] for optimization and memory reasons. The volume data analysis as shown in Fig. 6.1 (a) shows that the dataset contains a high class imbalance between foreground and background. In addition, Fig. 6.1 shows the distribution of the individual cervical vertebrae. The visualization of the input data for the neural network and the corresponding ground truth can be seen on a randomly chosen volume in Fig. 6.3 (a) and Fig. 6.3 (b), respectively.

6.2 Lumbar

The lumbar vertebrae, together with the sacrum and the pubic bone, form the basis of the entire spine. There are a total of five lumbar vertebrae, with the fifth lumbar

[1] https://www.uni-koblenz-landau.de/en/campus-koblenz/fb4/icv/vissim

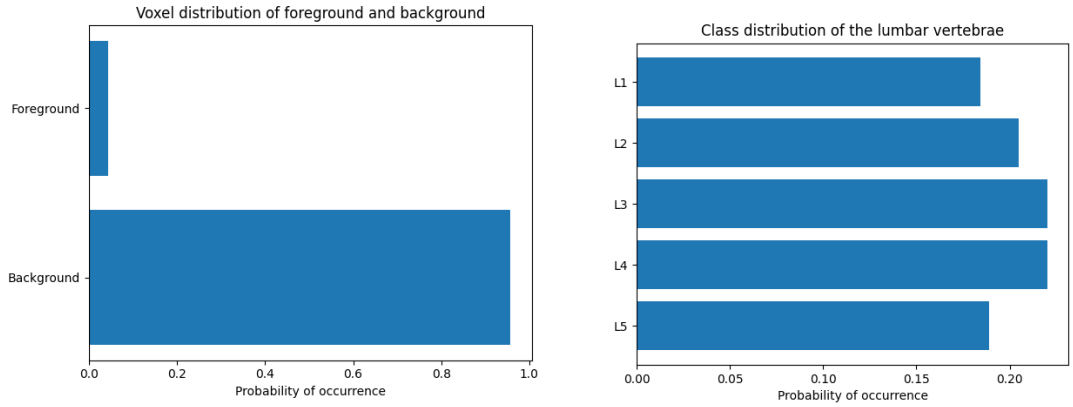


(a)

Figure 6.1: Voxel distribution of foreground and background on the cervical dataset.

vertebra ending at the sacrum and the first lumbar vertebra connecting to the twelfth thoracic vertebra. The exact segmentation of the lumbar vertebrae is crucial for an assessment by medical experts and later for a possible diagnosis, because the arrangement of the individual vertebrae can be used to precisely identify misalignments. The MyoSegmenTum database, provided by Burian et al. [Bur+19], consists of manually segmented lumbar muscle groups and lumbar vertebral bodies (L1 to L5) in chemical shift encoding-based water-fat MRI. Therefore, water, fat and proton density fat fraction (PDFF) volumes with the associated segmentation volumes 54 subjects were recorded. The proportion of female participants was 39 and that of male participants was 14, the average age and standard deviation among the participants were 51.6 ± 16.7 years. The MRI experiment was performed on a 3T system and individual scans were performed for the lumbar muscles and vertebral bodies. The collected data was then processed using mDIXON fat quantification method, resulting in T_2 weighted volume data. The segmentation process was performed by certified pathologists and took 50min each for the lumbar muscles and 1:40h for the lumbar vertebral bodies. For more information on the process of creation and on non-image related aspects, the reader is referred to the publication of Burian et al [Bur+19]. The volume data analysis of the ground truth as shown in Fig. 6.2 (a) shows that the dataset contains a high class imbalance between foreground, which resembles the lumbar vertebral bodies, and background. Further, it can be observed as in Fig. 6.2 (b) displayed that the distribution of the individual vertebrae is almost uniform, with vertebrae L3 and L4 occupying the largest proportions and vertebrae L1 taking up the smallest proportion. To get an idea of what the volume

data and the corresponding ground truth look like, Fig. 6.3 (c) and 6.3 (d) displays them from the identical rotated sagittal view.



(a) **(b)**

Figure 6.2: Voxel distribution of foreground and background (a) and the class distribution of each lumbar vertebrae (L1-L5) (b).

Figure 6.3: 3D rotated sagittal view of the normalized cervical (a) and (c) lumbar vertebrae volume data and the ground truth cervical (b) and (d) lumbar vertebrae volume data.

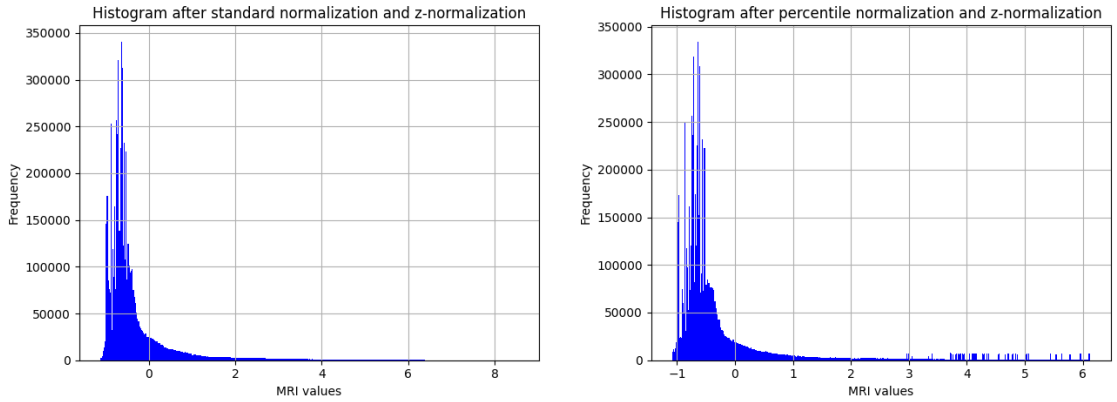
6.3 Preprocessing

The provided ground truth vertebrae data was merged into one *.mha file to speed up the process of loading it into the graphics card. Furthermore, the data was binarized during the merging process. The volume data of each individual were tri-linearly interpolated to match the dimensions [32,224,224]. The reduction of the dimension was done to increase the batch size, which in turn leads to a more stable and faster converging training process.

Due to the high intensity values, which can be seen in Fig. 6.5 (a) and Fig. 6.6 (a) for the cervical and lumbar dataset respectively, and the resulting exploding gradients, which would possibly occur in the backward pass through the neural network, the intensity values are normalized. As suggested in [Ise+18], the raw input data is normalized by clipping to the percentiles [0.5,99.5] of the intensity values. A comparison between the standard normalization and the normalization with consideration of the percentiles on the lumbar data can be seen in Fig. 6.4 in the form of the histograms. The normalization is then followed by a Z-normalization, as suggested in [Ise+18]:

$$\boldsymbol{Z}_i = \frac{\boldsymbol{X_i} - \mu_i}{\sigma_i}, \tag{6.1}$$

where μ_i and σ_i represent the mean and standard deviation of the input $\boldsymbol{X_i}$.

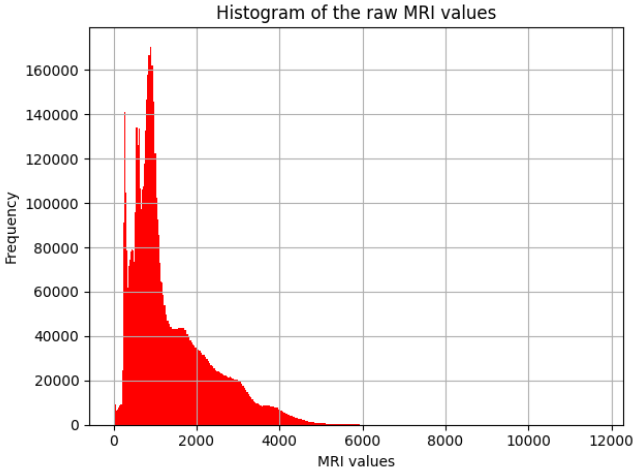


(a) **(b)**

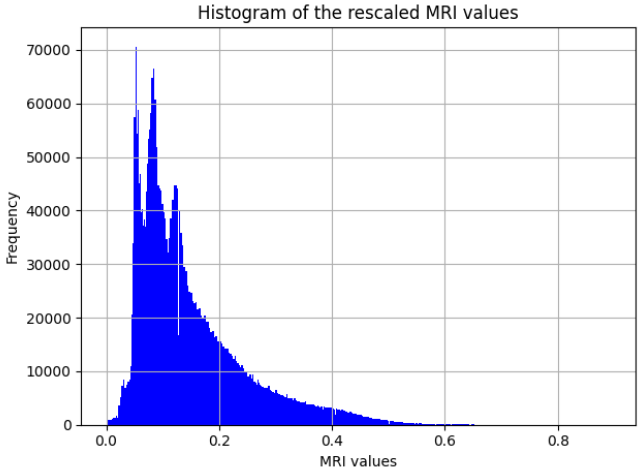
Figure 6.4: Comparison between the standard normalization (a) and the percentile normalization (b) applied on the lumbar data intensity values.

Due to the Z-normalization, the intensity values have a mean of zero and a variance of one. As described in [Lec+00], this leads to faster convergence because the backward pass updates of the network are not biased in a particular direction. The entire process of data modification from raw intensity values to training-ready intensity values of the

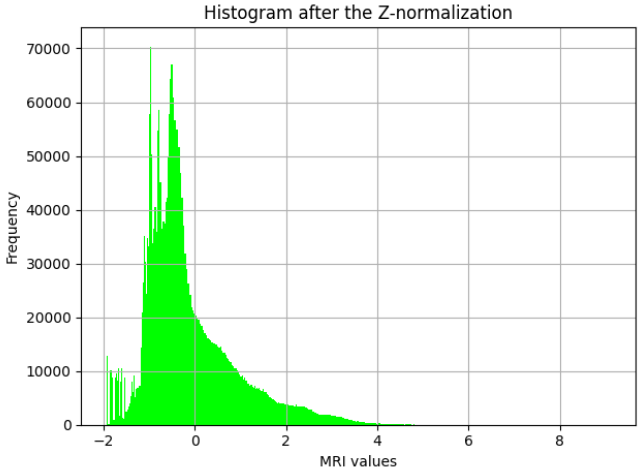
cervical and lumbar dataset can be seen in Fig. 6.5 and in Fig. 6.6, respectively. It is noticeable that the distributions of the raw MRI intensity values differ for the cervical and lumbar vertebrae. The distribution of raw intensity values of the cervical vertebrae, as shown in Fig. 6.5 (a), is more like a normal distribution with a few outliers that can be observed as peaks. Whereas the distribution of the raw intensity values of the lumbar vertebrae, shown in Fig. 6.6 (a) is more like an F-distribution [Wik21c]. After scaling and Z-normalisation, the initial F-distribution of the lumbar vertebrae intensity values more closely resembles a normal distribution. However, when transforming the cervical vertebra intensity values, it can be seen that binomial structures, as seen in Fig. 6.5 (b) and Fig. 6.5 (c), are enhanced.



(a)



(b)



(c)

Figure 6.5: Preprocessing steps of the cervical data from the raw MRI values (a), to the rescaled MRI values (b) and finally to the z-normalized MRI values (c)

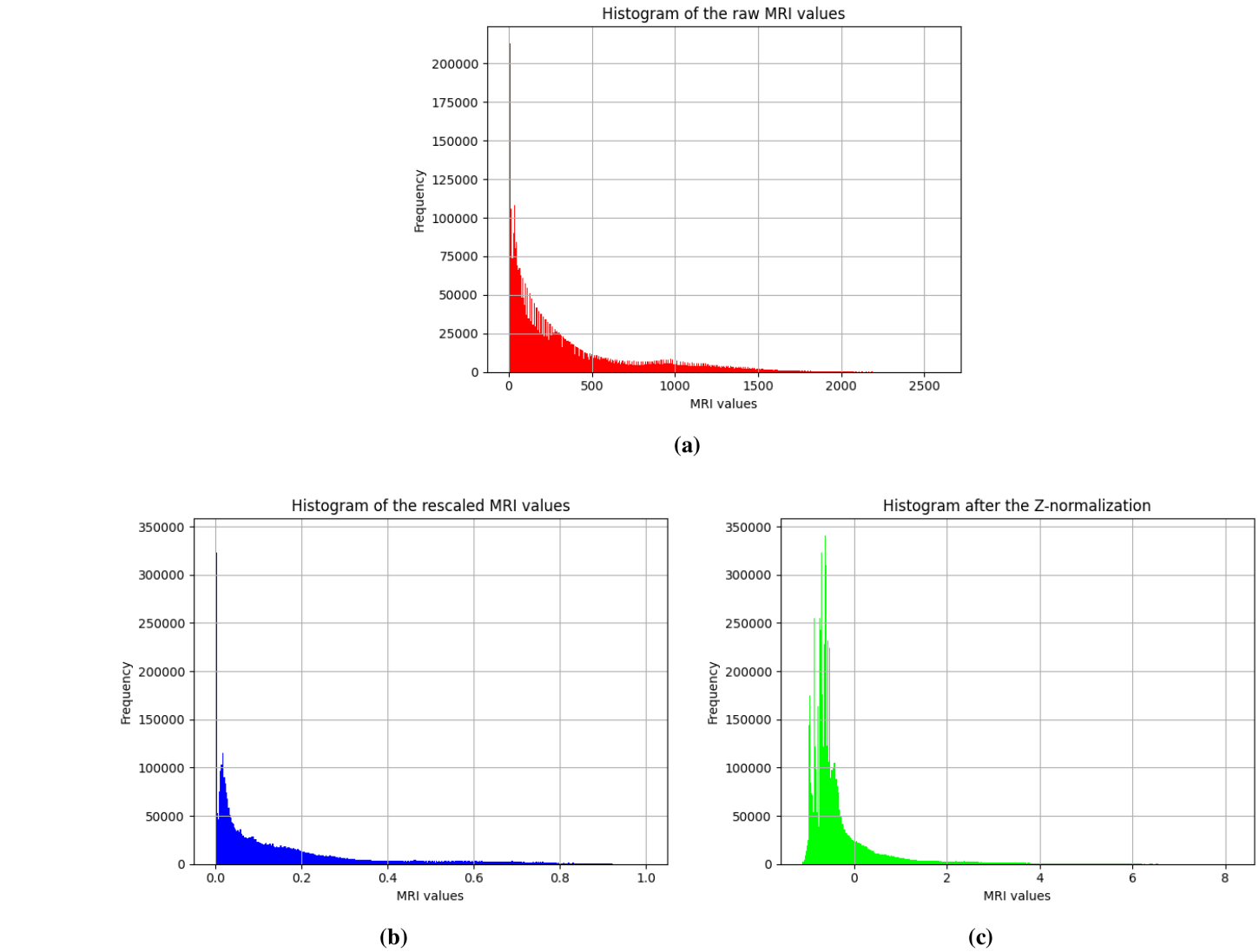


Figure 6.6: Preprocessing steps of the lumbar data from the raw MRI values (a), to the rescaled MRI values (b) and finally to the z-normalized MRI values (c)

6.4 Augmentation

Data augmentation represents a set of techniques that expands the inital input space X by creating new synthetic data based on the existing data. The objective of data augmentation is to act as a regularization method and to ensure that the model does not overfit the training data, which leads to a better performance on the validation set. To ensure reproducibility and to further promote standardisation, a library called Torchio devoloped by Perez-Garcia et al. [PSO20] , which specialises in data augmentation in the context of medical data, is used for the affine transformations. The transformations used include translation, scaling, rotation and the addition of noise, for the occurrence probabilities and the parameters of each function Table A.6 is referenced.

In addition to the standard methods, another method known as "MixUp", which was proposed by Zhang et al. [Zha+18], is used. The idea behind MixUp is that by interpolating the entire batch with a randomly chosen element of the batch, the network h_θ learns a linear correlation between the individual examples:

$$\boldsymbol{X_m} = \lambda \boldsymbol{X_t} + (1 - \lambda)\boldsymbol{X_r}, \tag{6.2}$$

$$\boldsymbol{Y_m} = \lambda \boldsymbol{Y_t} + (1 - \lambda)\boldsymbol{Y_r}, \tag{6.3}$$

where X_r and Y_r are the randomly chosen elements of the raw input data and the ground truth respectively, X_i and Y_i representing the entire batch of raw input data and ground truth at training step t respectively and $\lambda \sim Beta(\alpha, \alpha)$, for $\alpha \in (0, \infty)$. The probabilities of occurrence used for the methods just described can be seen in Table A.6 in the appendix.

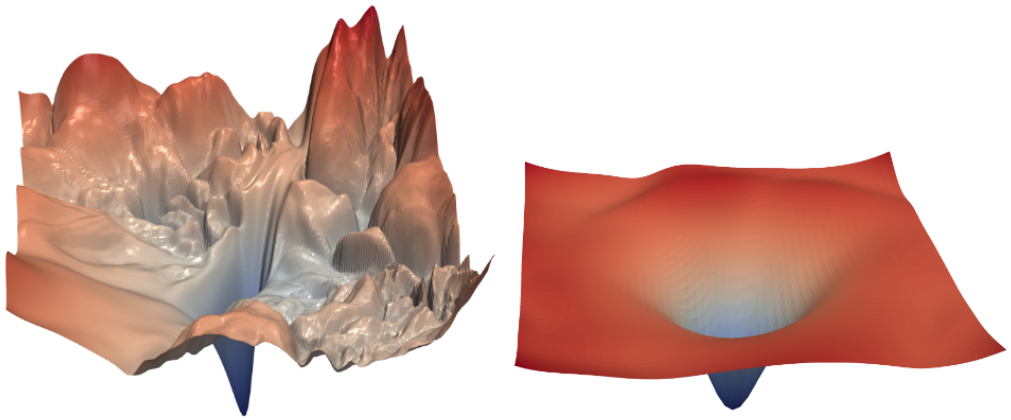
Chapter 7

Model architecture

In this chapter, the chosen 3D U-Net model architecture is presented and explained in more detail. In addition, individual modifications to the model architecture are addressed and justified.

7.1 3D U-Net

The U-Net architecture, proposed by Ronneberger [RFB15] in 2015 , is among the most robust and best performing network architectures for 2D biomedical image segmentation. The network architecture is composed of two major subsections, the encoder and decoder, which give the architecture its name by their combined path shape. The encoder compresses the input data into an internal feature representation that incorporates the global context, while the decoder uses this internal feature representation for precise estimation of local context. The encoder is composed of individual blocks called down-blocks, which are composed of different layers, which is shown in Fig. 7.2. The input of the down-block is a single multidimensional tensor. The input tensor is modified by two smaller blocks consisting of a sequence of convolution, normalization and activation operators. After the operators are applied, one result is saved for the use of the skip connection, and the other result is bisected by an application of the max pooling operator in the spatial dimensions and is considered as the input to the next down-block. The decoder also consists of individual blocks, the so-called up-blocks, the composition of which can be seen in Fig 7.3. The structure of the up-blocks is similar to the structure of the down-blocks, but the input and output differ. The input of the up-blocks consists of two tensors with the equal dimensions, which are concatenated before they are changed by the operators. To double the spatial dimension of the tensor, a transposed convolution operator is used at the end. The specific operators for normalization and activation are discussed further below. Unlike normal encoder-decoder structures such as Generative Adversial Networks (GAN) [Goo+14], which are connected via a latent



(a) without skip connections (b) with skip connections

Figure 7.1: Visualization of the ResNet model loss landscape without skip connections (a) and with skip connections (b).[1]

space, the encoder and decoder in the U-Net architecture are directly contiguous via skip connections. The skip connections ensure that low-level features of the encoder path are connected to the high-level features of the decoder path to generate finer and more accurate results in the output. The idea behind skip connections is to reduce the number of multiplications by the chain rule in the backward pass with very small value, leading to convergence of the weight update towards zero, by an alternative path for the gradient [Ada20]. Li et al. [Li+18] introduced a method to visualize the loss landscape in multiple dimensions and observed that skip connections in very deep networks act as a smoothing operator for very chaotic initial loss landscapes. The influence without and with skip connections on the loss landscape are respectively visualized in Fig. 7.1 using the example of the ResNet [He+15a] architecture.

The U-Net architecture for 2D biomedical image data was followed a year later by Çiçek et al. [Çiç+16] the generalization of the U-Net architecture, called 3D U-Net, for 3D biomedical volume data. In the meantime, many modifications to the original U-Net and 3D U-Net architecture like Res-UNet [Dia+20] or Attention U-Net [Okt+18] were proposed, which provided superior results in specific medical subsets, however, the adaptive framework "no new U-Net" (nnU-Net) by Isensee et al. [Ise+18] beat its counterparts on almost all medical benchmarks. The basic assumption made by Isensee et al. is that neural networks are not fully optimally trained due to lack of decisions regarding the correct choice of architecture based on inference from data analysis. Therefore, a heuristic was developed that provides decisions on the selection of the appropriate architecture based on the data analysis. In the end, the 3D U-Net architecture was chosen because the results of the nnUNet, which mainly use 3D U-Net with modifications, are

[1] Source: [Li+18][accessed: 27.03.2021]

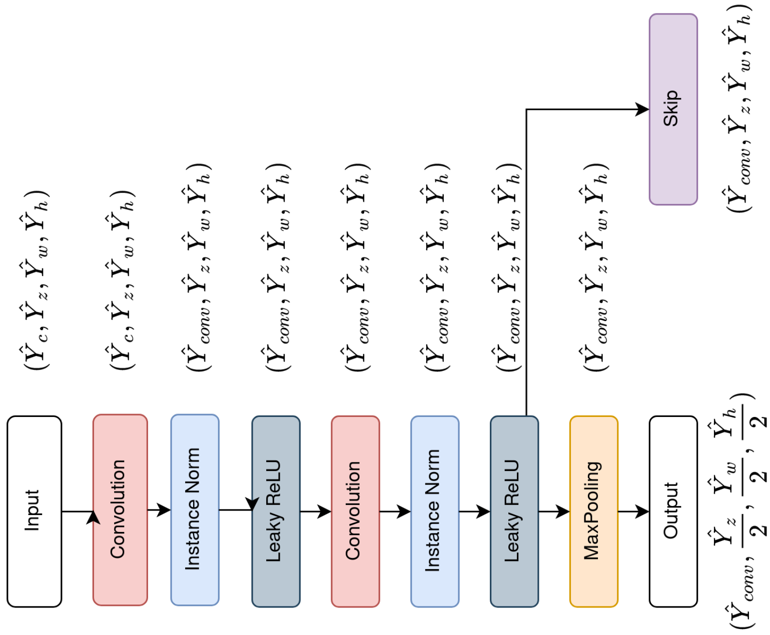


Figure 7.2: Visualization of the composition of the down-block with corresponding dimension.

superior to the other architectures after all. However, modifications to components of the standard 3D U-Net architecture are also made in the next chapter.

7.2 Activation

Non-linear activation functions allow the neural network to make a complex mapping between input and output. The derivative of a linear activation function is independent of the input and is therefore useless for backpropagation, the derivative of non-linear activation functions have a connection to the input and can therefore be used for backpropagation. Nair [NH10] proposed the now most common activation function Rectified Linear Units (ReLU). The ReLU function forms the identity equation in the positive quadrant and in the negative quadrant each value is set to zero:

$$f_r(x) = \begin{cases} x & for\ x > 0, \\ 0 & for\ x \leq 0. \end{cases} \tag{7.1}$$

Due to the almost linearity of ReLU, it can be optimized efficiently. However, the ReLU function in the negative quadrant does not produce useful information for backpropagation, since every value is set to zero. This effect is also known as "dead neurons". To prevent the dead neurons, Maas [MHN13] proposed a modification to the

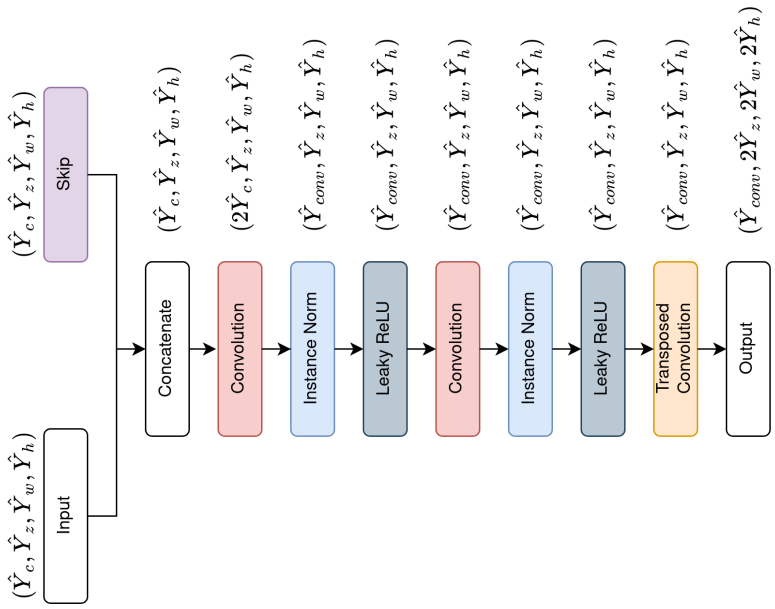


Figure 7.3: Visualization of the composition of the up-block with corresponding dimension.

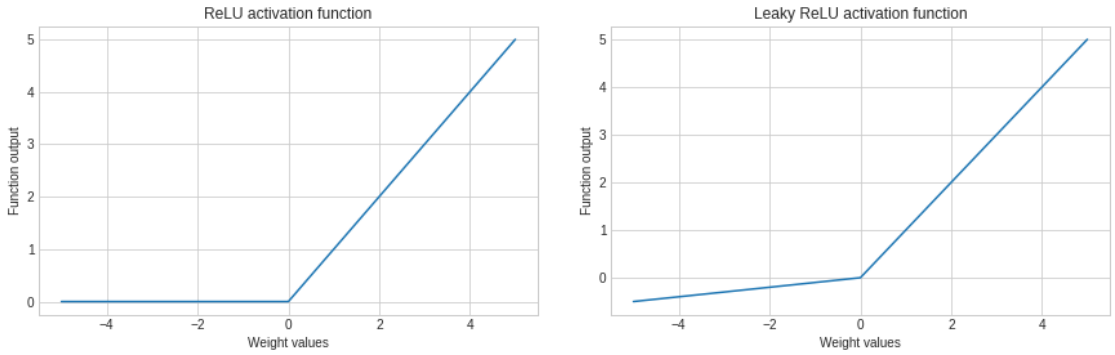
ReLU function that has become known as the "Leaky Rectified Linear Unit" (LReLU). The modification according to Maas is that instead of setting the values in the negative quadrant to zero, the values are instead multiplied by an associated slope α_{lr}:

$$f_{lr}(x) = \begin{cases} x & for\ x > 0, \\ \alpha_{lr} x & for\ x \leq 0. \end{cases} \tag{7.2}$$

Both the ReLU and the LReLU function are shown graphically in Fig. 7.4. Xu et al. have shown in an evaluation on convolutional neural networks [Xu+15] that replacing the null function in the negative quadrant with a non-zero function as proposed in LReLU or also "Parametric Rectified Linear Unit" (PReLU) [He+15b] leads to an improvement of the results. Due to the simplicity of LReLU and the described advantage over ReLU, it is used as the activation function for the proposed model architecture.

7.3 Initialization

The wrong initialization method of the weights of a neural network can lead to effects like vanishing gradients or exploding gradients. Glorot [GB10] introduced the Xavier initialization, which forms the values of the weights from a uniform distribution depending on the previous activation layer size:



(a) **(b)**

Figure 7.4: Comparison between the outputs of the ReLU activation function (a) and the Leaky ReLU activation function (b).

$$\theta_l \sim U[-\frac{1}{n_l}, \frac{1}{n_l}], \tag{7.3}$$

where U represents the uniform distribution with range of [-a,a] and n_l represents the size of the previous activation layer. The bias is initialized to a value of zero in all layers. He et al. [He+15b] introduced the Kaiming initialization, which considers the use of ReLU layers compared to Glorot, who assumes linear activations, in the modern networks. According to He et al. [He+15b] three basic properties are made:

1. the components of θ_i follow the identical distribution and are independent of each other
2. the components of the previous layer, a_{i-1} are independent of each other and follow the identical distribution,
3. θ_i and a_{i-1} are independent of each other.

From the previously defined properties, it follows that:

$$\frac{1}{2} n_l Var[\theta_l] = 1. \tag{7.4}$$

Which in turn leads to the initialization method:

$$\theta_l \sim \mathcal{N}[0, \frac{2}{n_l}], \tag{7.5}$$

where $\mathcal{N}$ represents the normal distribution with mean of zero and standard deviation of $\frac{2}{n_l}$. Since the selected architecture uses Leaky ReLU functions, the Kaiming initialization method is therefore preferred over the Xavier initialization method.

7.4 Normalization

Normalization reduces the training time and stabilizes the training by standardizing the input. In general, any normalization works by translating the mean value and scaling by the standard deviation:

$$\hat{\boldsymbol{Z}}_{\boldsymbol{i}} = \frac{1}{\sigma_i}(\boldsymbol{Y}_{\boldsymbol{i}} - \mu_i), \tag{7.6}$$

where $\boldsymbol{Y}_{\boldsymbol{i}}$ represents the input, μ_i and σ_i represent the mean and standard deviation of the input respectively and i represents the 5D-vector consisting of the axis (N, C, Z, W, H), where N is the batch dimension, C is the channel dimension and Z, W, H are the spatial dimensions.. The mean value and standard deviation of the input is determined as follows:

$$\mu_i = \frac{1}{n}\sum_{k \in S_i} \boldsymbol{Y}_{\boldsymbol{k}}, \tag{7.7}$$

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{k \in S_i}(\boldsymbol{Y}_{\boldsymbol{k}} - \mu_i)^2}, \tag{7.8}$$

where k is a element of the voxel set S_i and n is the size of the set. After normalization by mean and variance as shown in eq. 7.6 , the output is translated and scaled by the learnable parameters γ_i and β_i to avoid a loss in representation, resulting in the final output of the normalization layer $\boldsymbol{Z}_{\boldsymbol{i}}$:

$$\boldsymbol{Z}_{\boldsymbol{i}} = \gamma_i \hat{\boldsymbol{Z}}_{\boldsymbol{i}} + \beta_i. \tag{7.9}$$

Most normalization methods differ in the definition of the set S_i. Ioffe [IS15] introduced the "batch normalization", where S_i is defined as:

$$S_i = \{k | k_C = i_C\}, \tag{7.10}$$

where i_C denotes the normalization of voxels with the same channel. Ioffe claimed that by applying batch normalization, the Internal Covariate Shift (ICS), which describes the phenomenon that changing the parameters of one layer simultaneously changes the distribution of the input for the next layers, is reduced.

However, this claim is challenged by publications from Santurkar et al. [San+19], who hypothesizes that the performance gain is the result of smoothing the loss landscape, and Awais [AIB20], who shows in a comprehensive study that unnormalized networks with a small learning rate can achieve equivalent results in terms of performance and proposes variants of batch normalization. Nevertheless, normalizing the network

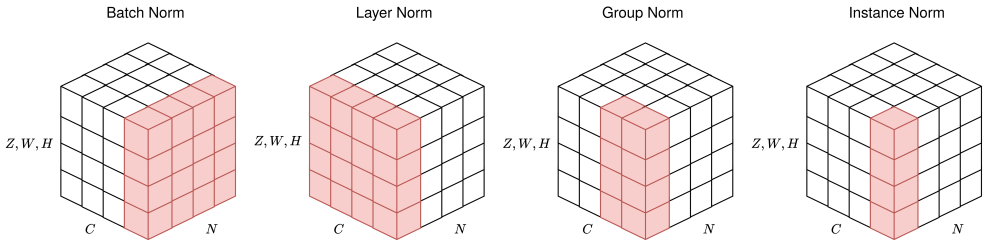


Figure 7.5: Effects of different normalization methods on the 5D input tensor displayed as a 3D cube, where the red marked mini cubes represent the selected set S_i.

by layers compared to the unnormalized version stabilizes the training and does not require finding a suitable learning rate. After batch normalization, the layer normalization proposed by Ba [BKH16] follows, where the set S_i is defined as follows:

$$S_i = \{k | k_N = i_N\}, \tag{7.11}$$

where i_N denotes the normalization of voxels that share the same batch. This variant is used in cases where the channel size is increased by the network. The layer normalization was in turn followed by the instance normalization propsed by Ulyanov [UVL17], which defines the set S_i as follows:

$$S_i = \{k | k_N = i_N, k_C = i_C\}, \tag{7.12}$$

where i_N and i_C denote the normalization of voxels along the spatial axis (Z, W, H). This variant is useful in applications where the channel size and the batch size are disproportionate to each other. The last proposed normalization method of Wu [WH18], describes the set of S_i as follows:

$$S_i = \{k | k_N = i_N, \frac{k_C}{C/g} = \frac{i_C}{C/g}\}, \tag{7.13}$$

where g is a pre-defined parameter and i_N and i_C denote the normalization of voxels along the spatial axis (Z, W, H), but with a fixed range along the channel axis, which forms a group. The relation among the normalization methods is shown in Fig. 7.5, inspired by the figure of Wu [WH18], through a 3D visualization of the effects of the normalization methods on the 5D input tensor, where the spatial dimensions have been collapsed into one dimension for a better visualization, displayed as a cube. Due to the fact that in the field of medical imaging and especially in MRI, the channel size is mostly one, the layer normalization and group normalization methods are not suitable for this purpose. Batch normalization compared to instance normalization is also only suitable to a limited extent due to the increasing number of batch sizes in the course of the forward pass through the network. The instance normalization, which is also

used by Isensee et al. [Ise+18], avoids this problem by normalizing along the spatial dimensions of each instance, which is independent of increasing batch sizes. Therefore, the instance normalization was used as a normalization method for the neural network.

Chapter 8

Experiments

In this chapter, the design of the experiments, the implementation and the results of the experiments are described. The validation technique "k-fold cross validation", the metrics and the learning rate scheduler are described and explained in more detail. The results of the experiments are presented statistically and visually according to the research question.

8.1 Setup and implementation details

The entire and uniform training sequence can be seen in Fig. 8.1 and is presented chronologically below. At the beginning, all relevant training variables such as the loss function to be used, the optional method or the dataset are defined. All other static training variables like the initial learning rate or the batch size can be found in Table A.6 in the appendix to ensure reproducibility. The dataset is then split into training and validation folds following the validation strategy presented in chapter 8.4. Then the split for the kth fold is selected and the model is trained on it. A training step includes training on all training folds with a predefined batch size. After a predefined value m_v of training steps, a validation step is perfomed on which the model is evaluated on the validation fold. The values of the evaluation and other information such as the 2D and 3D visualization of the results of the validation step are stored locally and on the platform "Weights and Biases" [Bie20]. After all k passes have been made for the dataset, the results of all models are cross-validated and also stored locally and online. In addition to storing the pure results, each model is also stored locally. To speed up the training process and to increase the batch size to fit more input data into the VRAM of the graphics card, the forward pass is performed in floating point 16, while the validation takes place in floating point 32. The models takes approximately 10GB of VRAM and one batch with size 4 needs also 10GB of VRAM. Each fold is trained M_e epochs and every m epochs the validation step is performed. The training time with three folds

takes approximately five hours. The training time with three folds is usually 5 hours, however, the time also depends on the selected method.

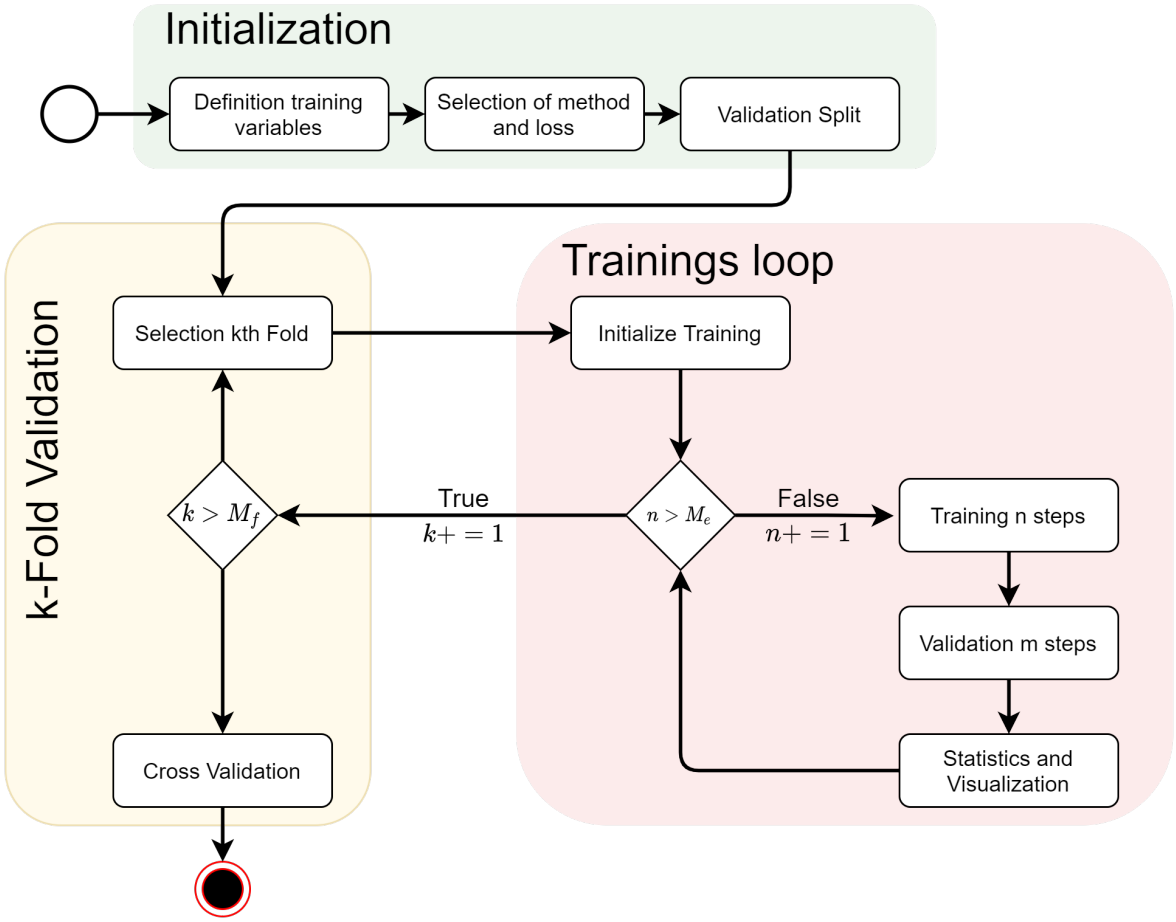


Figure 8.1: Visualisation of the training process with the k-fold validation strategy.

The implementation is done in Python and for all deep learning needs the framework PyTorch [Pas+19] was used. To make the implementation more modular and minimal, a combination of PyTorch Lightning [Fa19], which is a wrapper for PyTorch, and "Weights and Biases" [Bie20] were used. For the implementation of the genetic algorithms, the framework DEAP [For+12] was used, which was extended by adding additional attributes [1]. To reproduce the experiments, a system is required that has the modules listed in setup.py installed and has a CUDA capable graphics card that has at least 22GB of VRAM. The minimum VRAM can be reduced to 14GB by adjusting the

[1] https://github.com/mhun1/deap

batch size, but this may lead to unstable, slower and not comparable results. All experiments were conducted on an RTX 3090 with 24GB VRAM and an AMD Ryzen 7. The implementation is available on Github[2].

8.2 Learning rate scheduler

The learning rate influences how much impact the partial derivative of the loss function with respect to the weights has on the future values of the weight. Learning rate scheduler adjust the learning rate over the course of the training based on a underlying strategy. The PolyLR scheduler proposed by Chen et al. [Che+17], adjusts the learning rate based on the relation between the current count of epochs e_t and the maximal count of epochs e_m to train:

$$\eta_t = \eta_{t=0} \cdot (1 - \frac{e_t}{e_m})^p. \tag{8.1}$$

The influence of the factor p on the course of the learning rate in the training process, with an initial learning rate of 0.001 and a length of 100 epochs, can be seen in Fig. 8.2. For values between 0 and 1, the function transforms from an initially almost constant function to a concave function to a linear monotonically decreasing function. For values above 1, the function takes on a convex property. The idea behind using an LR scheduler like PolyLR in the context of segmentation is that a high learning rate is preferred in the initial phase of training so that the neural network learns coarse segmentations quickly, and a lower learning rate is preferred at the end of training in order to stabilise and thus also to improve the predictions. Therefore, the selected value for the factor p is 0.9.

8.3 Metrics

To evaluate each fold in the validation step, metrics are needed to value the predictions of the model. Each value of the prediction, if greater than 0.5, is clamped to one and if less than or equal to 0.5, is clamped to zero. In the following, true-positives, true-negatives, false-positives, and false-negatives are referred to as TP, TN, FP, and FN, respectively. Accuracy describes the ratio between all correctly segmented data and the total of segmentations:

$$A = \frac{TP + TN}{TP + TN + FP + FN}. \tag{8.2}$$

Precision measures the ratio between the positive segmentation data and the total of positive segmented data:

[2]https://github.com/mhun1/thesis

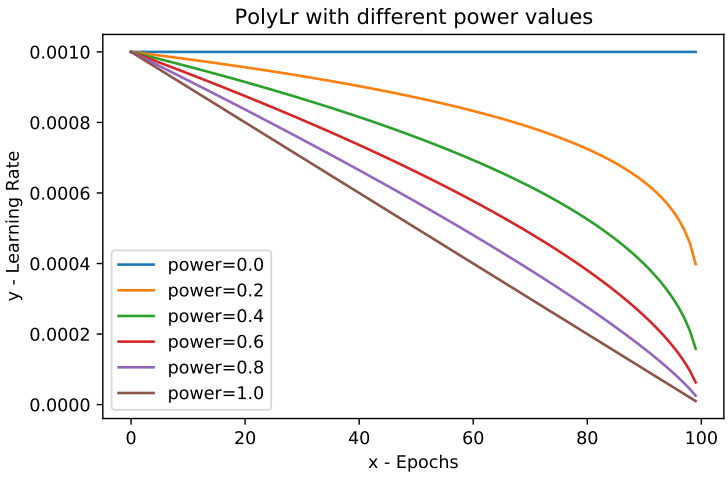


Figure 8.2: PolyLr scheduler with different values of p, initial learning rate $\eta_{t=0} = 0.001$ and $e_m = 100$.

$$P = \frac{TP}{TP + FP}. \tag{8.3}$$

Recall describes the ratio between the positive segmentation data and the total of correctly segmented data:

$$R = \frac{TP}{TP + FN}. \tag{8.4}$$

The F_β score generalizes the harmonic mean of recall and precision, the F_2 score favors to weight recall over precision:

$$F_2 = \frac{5PR}{4P + R}. \tag{8.5}$$

As already shown in Sec. 3.3.1, the Dice score, is a special case of the F_β score when $\beta = 1$, and therefore describes the harmonic mean of recall and precision:

$$D = \frac{2TP}{2TP + FP + FN}. \tag{8.6}$$

The definition of the Intersection over Union follows the definition presented in Sec. 3.3.2:

$$J = \frac{TP}{TP + FP + FN}. \tag{8.7}$$

The definition of the Hausdorff distance [Wik21e] is given in Millimetre (mm) and follows the standard implementation of the MedPy[3] library:

$$HD = \max \left\{ \sup_{\hat{y} \in \hat{Y}} \inf_{y \in Y} d(\hat{y}, y), \sup_{y \in Y} \inf_{\hat{y} \in \hat{Y}} d(y, \hat{y}) \right\}. \tag{8.8}$$

8.4 K-fold cross validation

To evaluate the performance of the models, a validation strategy called k-fold cross-validation [Wik21a] is used. For this purpose, the entire dataset is divided into training and validation folds by a pre-defined value k. Each split is based on a predefined seed value that is stored for each experiment to ensure comparability of results for each experiment. In each iteration, where a new model is trained, the trainings and validation folds are selected depending on the predefined value k. The training and the validation folds are then used for the new model. The schematic structure is shown in Fig. 8.3 using the example, where k equals four. During each iteration, the metrics presented in Sec. 8.3 are collected for each validation step. Then, depending on the metric, the minimum or maximum values are determined, and the mean and standard deviation of all iterations are computed. By alternating training and validation folds, a more accurate estimate of model prediction performance can be made compared to methods that do not alternate.

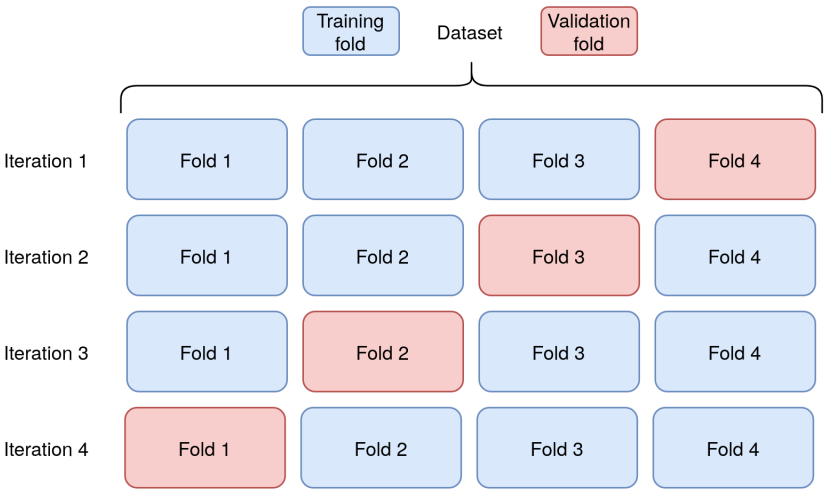


Figure 8.3: Example k-fold cross validation with k=4, where a blue rectangle indicates a training fold and a red rectangle a validation fold.

[3] https://github.com/loli/medpy

8.5 Results

Each table in the next sections shows the calculated mean and standard deviation of the best values over three runs, which are the maximum or minimum values depending on the metric, for each loss function. The best results of each metric among all loss functions are printed in bold. All results and the derived values in the tables can be viewed interactively for the cervical[4] and lumbar[5] vertebrae on the platform "Weights & Biases".

8.5.1 Research question 1

In order to answer the research question of which loss function provided the best segmentation results using the 3D U-Net architecture, the results of the training for the cervical and lumbar vertebrae are shown in Table 8.1. The visual results on the cervical and lumbar vertebrae datasets can be seen in two examples in Fig. 8.2 and 8.3 as a time series and in Fig. 8.4 and 8.5 as a volume representation, respectively. Further visual results can be found in the appendix.

In general, the accuracy of the results on the cervical dataset is very high, this is because the accuracy metric is the only one that takes true-negatives into account, which are disproportionately present in an imbalanced dataset as in Chapter 6 explained. The results for the cervical vertebrae show that the Tversky-Focal loss dominates four of the total seven metrics, which are highlighted in bold, with the highest scores. The Asymmetric loss is able to achieve the best result in the F_2 metric, with a gap of 1.74% to the next best loss function. In the Hausdorff metric, the Boundary loss achieves the best result, followed by the Asymmetric loss. The cross entropy scores the highest value in the Recall metric, followed by the Asymmetric loss with a difference of 1.33%. When examining the different categories of loss functions, several things stand out. The Focal loss performs worse than the cross entropy in five of the metrics, but the precision, with associated losses in recall, is shown to be higher. Looking more precisely at the region-based loss functions, it is noticeable that the balance between precision and recall is most consistent for the Tversky loss. As the values of precision and recall indicate, the focus on false-positives or false-negatives seems to be a direct trade-off, and can lead to different results, that are not fully captured by the Dice score.

Considering the results of the lumbar vertebrae, it is noticeable that the Dice loss dominates three of the seven metrics, closely followed by the Tversky-Focal loss, which is equal to the Dice loss in the accuracy metric and only performs better in the precision metric. In the recall metric, the cross entropy yields the best results, followed by the Asymmetric loss. Looking at the precision metric, the Tversky-Focal loss dominates,

[4] https://wandb.ai/mhun/MasterthesisCervical
[5] https://wandb.ai/mhun/MasterthesisLumbar

	Loss	Acc	Dice	F_2	IoU	Precision	Recall	Hausdorff
Cervical								
D	$\mathcal{L}_{CE}$	98.15 ± (0.17)	70.99 ± (3.17)	82.79 ± (1.37)	55.61 ± (3.70)	57.97 ± (4.44)	**95.11 ± (1.92)**	27.94 ± (8.12)
	$\mathcal{L}_{F}$	98.35 ± (0.28)	62.02 ± (1.77)	58.86 ± (4.06)	45.67 ± (2.03)	79.11 ± (1.46)	57.24 ± (5.42)	29.59 ± (2.93)
R	$\mathcal{L}_{D}$	99.00 ± (0.06)	80.55 ± (2.87)	84.39 ± (0.80)	67.80 ± (3.69)	77.65 ± (7.40)	89.30 ± (2.75)	24.65 ± (6.51)
	$\mathcal{L}_{T}$	99.03 ± (0.04)	79.69 ± (1.56)	79.34 ± (2.36)	66.59 ± (1.76)	83.62 ± (7.16)	79.91 ± (4.86)	31.40 ± (1.73)
	$\mathcal{L}_{A}$	98.79 ± (0.01)	78.50 ± (4.50)	**86.46 ± (2.27)**	65.08 ± (5.75)	69.24 ±(6.75)	94.63 ± (0.75)	20.13 ± (4.30)
Ds	$\mathcal{L}_{B}$	98.96 ± (0.17)	77.95 ± (7.52)	78.48 ± (7.43)	64.84 ± (8.93)	82.04 ± (6.19)	79.30 ± (7.23)	**19.02 ± (6.42)**
	$\mathcal{L}_{HD}$	98.39 ± (0.16)	65.17 ± (3.93)	63.48 ± (5.43)	48.97 ± (4.11)	75.66 ± (5.85)	62.84 ± (6.36)	25.34 ± (1.27)
C	$\mathcal{L}_{DCE}$	98.93 ± (0.02)	79.35 ± (2.22)	83.40 ± (2.26)	66.04 ± (2.72)	75.82 ± (5.97)	87.74 ± (4.58)	22.06 ± (1.87)
	$\mathcal{L}_{TF}$	**99.09 ± (0.01)**	**81.15 ± (2.53)**	80.83 ± (1.45)	**68.65 ± (3.16)**	**84.78 ± (6.21)**	81.77 ± (3.44)	23.95 ± (5.73)
	$\mathcal{L}_{DF}$	98.99 ± (0.01)	80.53 ± (2.81)	84.72 ± (1.56)	67.83 ± (3.43)	76.63 ± (4.92)	89.27 ± (1.85)	27.51 ± (3.84)
Lumbar								
D	$\mathcal{L}_{CE}$	98.09 ± (0.06)	83.27 ± (1.20)	89.16 ± (0.13)	72.09 ± (2.13)	75.80 ± (2.81)	**95.36 ± (1.64)**	19.01 ± (3.02)
	$\mathcal{L}_{F}$	98.34 ± (0.29)	83.07 ± (2.31)	80.90 ± (4.18)	71.82 ± (3.57)	90.55 ± (0.65)	79.99 ± (5.63)	13.80 ± (1.70)
R	$\mathcal{L}_{D}$	**98.79 ± (0.23)**	**88.63 ± (1.08)**	89.41 ± (1.35)	**80.35 ± (0.73)**	89.44 ± (1.45)	90.32 ± (2.04)	11.44 ± (1.74)
	$\mathcal{L}_{T}$	98.79 ± (0.24)	88.04 ± (1.14)	85.88 ± (1.06)	79.46 ± (0.92)	93.55 ± (2.15)	84.65 ± (1.06)	**10.89 ± (1.50)**
	$\mathcal{L}_{A}$	98.63 ± (0.17)	87.55 ± (0.66)	**90.61 ± (1.22)**	78.64 ± (0.32)	84.03 ± (1.75)	94.03 ± (2.03)	12.65 ± (3.47)
Ds	$\mathcal{L}_{B}$	98.64 ± (0.23)	86.79 ± (0.83)	86.42 ± (1.12)	77.40 ± (0.82)	90.43 ± (0.49)	86.26 ± (1.46)	12.78 ± (2.39)
	$\mathcal{L}_{HD}$	97.93 ± (0.31)	78.64 ± (2.72)	75.72 ± (5.29)	65.48 ± (3.53)	87.46 ± (1.47)	74.11 ± (6.89)	15.57 ± (0.37)
C	$\mathcal{L}_{DCE}$	98.76 ± (0.26)	88.35 ± (1.24)	88.87 ± (1.41)	79.88 ± (1.34)	89.12 ± (1.36)	90.09 ± (1.72)	12.45 ± (1.76)
	$\mathcal{L}_{TF}$	**98.79 ± (0.23)**	88.05 ± (0.87)	85.94 ± (0.59)	79.43 ± (0.48)	**93.56 ± (2.04)**	84.82 ± (0.85)	11.61 ± (0.70)
	$\mathcal{L}_{DF}$	98.78 ± (0.27)	88.46 ± (1.36)	89.35 ± (1.15)	80.06 ± (1.19)	89.88 ± (1.80)	90.95 ± (1.45)	12.57 ± (0.87)

Table 8.1: Baseline table for the cervical and lumbar dataset, grouped by the categories defined in Chapter 2, where D represents the distribution-based loss functions, R represents the regional-based loss functions, Ds the distance-based loss functions and C the compound loss functions.

followed by the Tversky loss with a small difference. The F_2 metric is dominated by the Asymmetric loss and in the Hausdorff metric the Tversky loss achieved the best result.

If the results are now considered across the board, several aspects stand out. In general, it is noticeable that there is a discrepancy between the metrics in the datasets, except for accuracy and recall. This is due to the fact that the segmentation of the lumbar vertebral bodies is structurally easier to consider than the segmentation of the entire cervical vertebrae. Looking at the cross entropy and the focal loss on both datasets, there is a direct trade-off between precision and recall, where the Focal loss dominates the precision and the cross entropy dominates the recall. When looking at the region-based loss functions, it can be observed on both datasets that precision is dominated by Tversky loss and recall by Asymmetric loss. However, there is no clear picture in terms of balance between precision and recall, which was most balanced on the cervical dataset with the Tversky loss and on the lumbar dataset with the Dice loss.

Table 8.2: Visual results of the cervical vertebrae as a time series using a random example from the first validation set.

Table 8.3: Visual results of the lumbar vertebral bodies as a time series using a random example from the first validation set.

Table 8.4: Visual results of the cervical vertebrae as a volume representation using a random example from the first validation set.

Table 8.5: Visual results of the lumbar vertebral bodies as a volume representation using a random example from the first validation set.

8.5.2 Research question 2

The results of the research question whether it is possible to learn loss functions for the segmentation tasks before training by a genetic approach or during training by a meta-learning approach are presented respectively. Figure 8.4 shows exemplary generated trees and the corresponding visual segmentation results. A closer look at the generated trees reveals that the genetic algorithm could not establish a meaningful or significant connection between the prediction of the neural network and the ground truth. In addition, the visual segmentation results also indicate that the neural network could not perform well in segmenting the vertebrae or the vertebral bodies as a result. Therefore, the genetic algorithms did not generate competitive results compared to the baseline in the previous Sec.8.5.1. The possible causes are discussed below. One possible reason could be the dimension of the input and output data. The authors of publication [GM20] use the method exclusively for regression tasks where the target output is only one-dimensional. In segmentation tasks, however, the input and output are multi-dimensional in nature. Secondly, the genetic algorithm is dependent on many parameters, such as the individual components of the genetic operators, which in turn can be a potential danger in terms of its outcome. Another factor is the time spent searching for the loss functions. In publication [GM20], the search was parallelised and could therefore be completed in a few days. In this use case, however, a parallel search is not possible due to the low VRAM storage capacity. The time spent searching for the loss functions was therefore terminated after one week without any prospect of further success.

The meta-learning approach, like the genetic approach, was unable to achieve competitive results compared to the results of the first research question. The training process proved to be unstable, which resulted in the segmentation network not providing usable segmentations. The possible reasons are similar to those for the genetic algorithms. The meta learning approach presented by Bechtle et al. only deals with tasks such as regression or classification that result in a one-dimensional output. For this reason, the results achieved by Bechtle et al. cannot be transferred one-to-one to the task area of segmentation. Furthermore, there is a difficulty in finding the right values for the training components, because not only one network, but two networks have to be optimised in one training step. The decision of the network architecture for the meta-loss network is already a difficult task. Bechtle et al. use a network architecture consisting of linear layers, which means that the prediction and the ground truth have to be concatenated and then flattened so that it can be used. A more natural approach would be to use convolution layers instead of linear layers, but this was found to lead to no improvement either. The research question of whether a loss function can be learned with the methods presented for segmentation tasks must therefore be answered in the negative at this point in time.

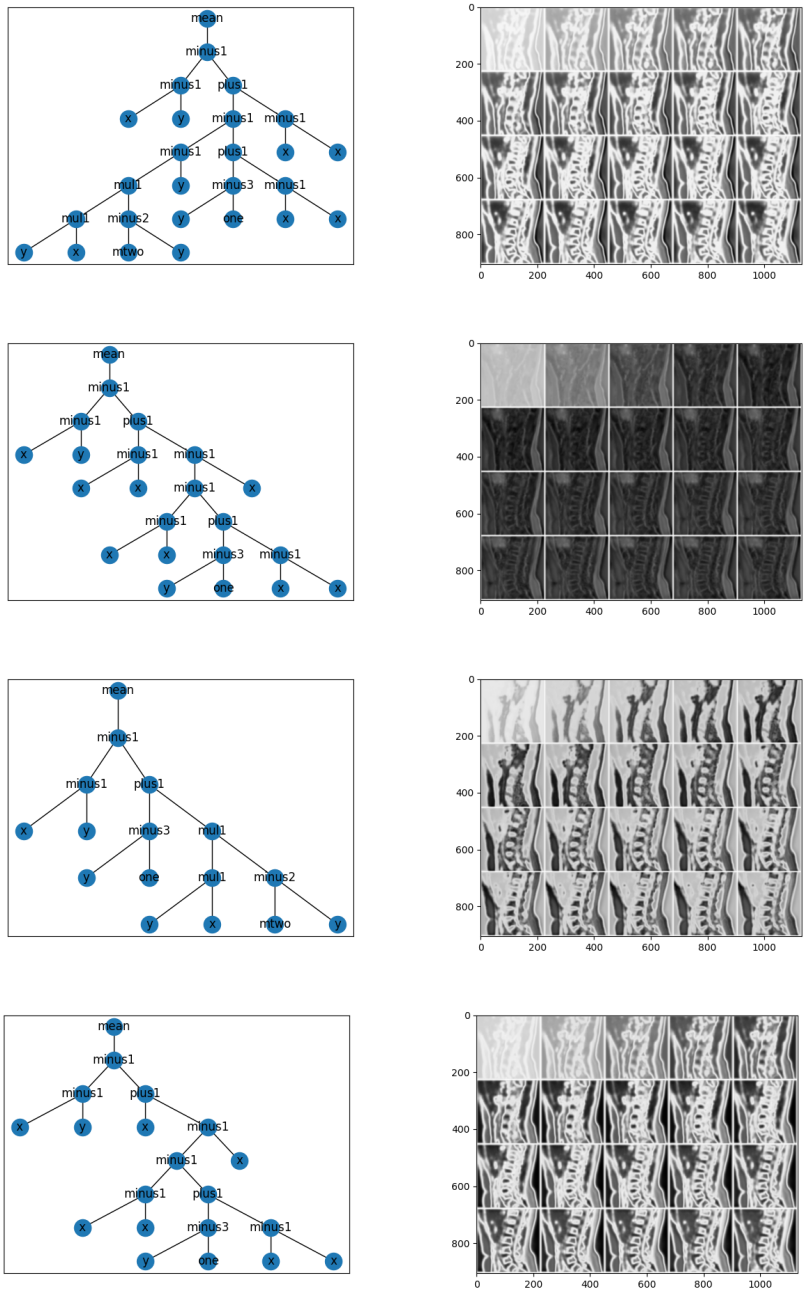


Figure 8.4: Results of the genetic algorithm, where the left column represents four exemplary trees and the right column represents the corresponding visual segmentation results.

8.5.3 Research question 3

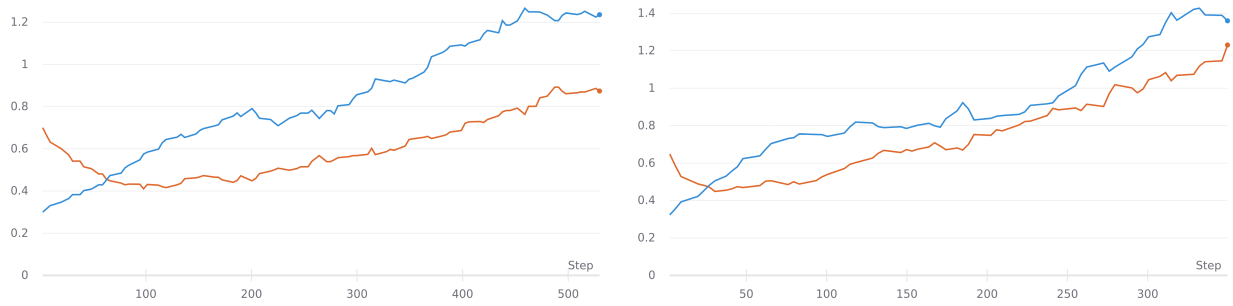
To address the third research question, the results of the greedy method are presented first, followed by the results of the adaptive weighting methods. Table 8.6 shows the results of the greedy method on the cervical and lumbar data compared to selected results from the first research question. On the cervical dataset, the greedy method applied to the Tversky loss is shown to achieve four of the seven metrics with the best results compared to the other loss functions. In direct comparison to the results of the normal Tversky loss, the greedy method optimises six of the seven metrics. In addition, it is noticeable that, compared to the normal Tversky loss, the recall has been greatly increased by 6.83% without a strong loss of precision. The Effectiveness loss function, proposed by the authors, was able to achieve good results, but did not lead to an improvement in any metric.

Loss	Acc	Dice	F_2	IoU	Precision	Recall	Hausdorff
Cervical							
$\mathcal{L}_D$	99.00 ± (0.06)	80.55 ± (2.87)	84.39 ± (0.80)	67.80 ± (3.69)	77.65 ± (7.40)	89.30 ± (2.75)	24.65 ± (6.51)
$\mathcal{L}_T$	99.03 ± (0.04)	79.69 ± (1.56)	79.34 ± (2.36)	66.59 ± (1.76)	**83.62 ± (7.16)**	79.91 ± (4.86)	31.40 ± (1.73)
$\mathcal{L}_A$	98.79 ± (0.01)	78.50 ± (4.50)	**86.46 ± (2.27)**	65.08 ± (5.75)	69.24 ±(6.75)	**94.63 ± (0.75)**	20.13 ± (4.30)
$\mathcal{L}_T/G$	**99.11 ± (0.05)**	**82.16 ± (1.96)**	83.81 ± (0.82)	**70.06 ± (2.47)**	83.22 ± (3.85)	86.74 ± (1.86)	**20.06 ± (0.37)**
$\mathcal{L}_E/G$	99.07 ± (0.06)	81.36 ± (1.61)	84.28 ± (0.66)	68.97 ± (1.75)	81.78 ± (5.60)	88.39 ± (2.31)	22.29 ± (2.51)
Lumbar							
$\mathcal{L}_D$	98.79 ± (0.23)	88.63 ± (1.08)	89.41 ± (1.35)	80.35 ± (0.73)	89.44 ± (1.45)	90.32 ± (2.04)	11.44 ± (1.74)
$\mathcal{L}_T$	98.79 ± (0.24)	88.04 ± (1.14)	85.88 ± (1.06)	79.46 ± (0.92)	**93.55 ± (2.15)**	84.65 ± (1.06)	**10.89 ± (1.50)**
$\mathcal{L}_A$	98.63 ± (0.17)	87.55 ± (0.66)	**90.61 ± (1.22)**	78.64 ± (0.32)	84.03 ± (1.75)	**94.03 ± (2.03)**	12.65 ± (3.47)
$\mathcal{L}_T/G$	**98.82 ± (0.20)**	**88.74 ± (0.63)**	89.11 ± (0.59)	**80.65 ± (0.18)**	91.32 ± (1.64)	89.47 ± (0.51)	11.39 ± (3.66)
$\mathcal{L}_E/G$	98.80 ± (0.26)	88.54 ± (1.21)	88.99 ± (0.97)	80.16 ± (0.99)	91.89 ± (1.13)	90.60 ± (2.30)	11.94 ± (2.69)

Table 8.6: Greedy algorithms results for the cervical and lumbar dataset.

Examining the results for the lumbar dataset in Table 8.6, it can be seen that the greedy method applied to the Tversky loss achieved the best results in three of the seven metrics. In direct comparison with the normal Tversky loss, the greedy method improved five of the seven metrics. In this case, too, compared to the normal Tversky loss, it can be seen that the recall was increased, which is accompanied by losses in precision, yet there is a greater balance between the two metrics. The Effectiveness loss was able to provide stable results, but these did not lead to an improvement in any metric, as in the cervical dataset.

What is striking across the datasets in terms of the balance between precision and recall after applying the greedy method to the Tversky loss, is that the difference between the two metrics was reduced or even reversed to a positive difference in the case of the cervical vertebrae data set. Looking at the trajectories of the weight values during the training step as seen in Fig. 8.5, several things stand out. Firstly, it can be seen that in both datasets a change in the initial weight ratio occurs after a few training steps. Furthermore, in both datasets there is a steady increase of the weight values after the



(a) **(b)**

Figure 8.5: The history of the distribution of the weight values of the Tversky loss per training step with the applied greedy method for the cervical (a) and lumbar (b) dataset, where α is represented by the orange line and β by the blue line.

crossing of the curves. This also causes the values of the training loss to increase in the middle of the training, however this did not lead to any decline in the metrics as previously shown. What is also noticeable is that the discrepancy of the two weight histories is smaller for the lumbar vertebra data than for the cervical vertebra data. This finding leads to a possible explanation as to why the balance between precision and recall is more balanced for the lumbar vertebrae than for the cervical vertebrae. In general consideration of the greedy method, advantages are shown over the standard method of setting the weights initially and leaving them static over the course of the training process.

Loss	Acc	Dice	F_2	IoU	Precision	Recall	Hausdorff
Cervical							
$\mathcal{L}_B$	98.96 ± (0.17)	77.95 ± (7.52)	78.48 ± (7.43)	64.84 ± (8.93)	**82.04 ± (6.19)**	79.30 ± (7.23)	**19.02 ± (6.42)**
$\mathcal{L}_B/R$	**98.97 ± (0.13)**	**80.45 ± (3.17)**	**84.59 ± (2.38)**	**67.83 ± (3.98)**	76.69 ± (4.35)	**89.68 ± (1.56)**	29.37 ± (7.71)
$\mathcal{L}_B/COV$	98.62 ± (0.06)	74.68 ± (2.02)	80.31 ± (0.62)	59.95 ± (2.19)	69.26 ± (3.48)	87.09 ± (3.28)	30.15 ± (1.47)
$\mathcal{L}_{HD}$	98.39 ± (0.16)	65.17 ± (3.93)	63.48 ± (5.43)	48.97 ± (4.11)	75.66 ± (5.85)	62.84 ± (6.36)	**25.34 ± (1.27)**
$\mathcal{L}_{HD}/R$	**98.40 ± (0.33)**	**69.29 ± (4.70)**	**74.47 ± (4.20)**	**53.95 ± (5.34)**	67.19 ± (0.75)	**79.67 ± (5.72)**	28.67 ± (6.30)
$\mathcal{L}_{HD}/COV$	98.15 ± (0.15)	64.98 ± (4.99)	68.97 ± (4.05)	48.71 ± (5.47)	64.10 ± (4.03)	72.79 ± (5.39)	33.22 ± (9.12)
$\mathcal{L}_{DCE}$	**98.93 ± (0.02)**	**79.35 ± (2.22)**	83.40 ± (2.26)	**66.04 ± (2.72)**	**75.82 ± (5.97)**	87.74 ± (4.58)	**22.06 ± (1.87)**
$\mathcal{L}_{DCE}/SA$	98.84 ± (0.06)	78.76 ± (2.09)	**84.50 ± (2.73)**	65.33 ± (2.13)	72.39 ± (3.30)	**90.12 ± (4.06)**	36.50 ± (10.70)
Lumbar							
$\mathcal{L}_B$	**98.64 ± (0.23)**	86.79 ± (0.83)	86.42 ± (1.12)	77.40 ± (0.82)	**90.43 ± (0.49)**	86.26 ± (1.46)	12.78 ± (2.39)
$\mathcal{L}_B/R$	98.53 ± (0.01)	**87.26 ± (0.19)**	87.29 ± (0.08)	**79.27 ± (0.31)**	88.57 ± (0.46)	87.73 ± (0.21)	**9.56 ± (0.62)**
$\mathcal{L}_B/COV$	98.63 ± (0.19)	86.89 ± (0.91)	**87.82 ± (0.93)**	77.59 ± (0.92)	88.22 ± (2.24)	**89.22 ± (1.75)**	14.14 ± (2.30)
$\mathcal{L}_{HD}$	97.93 ± (0.31)	78.64 ± (2.72)	75.72 ± (5.29)	65.48 ± (3.53)	87.46 ± (1.47)	74.11 ± (6.89)	15.57 ±(0.37)
$\mathcal{L}_{HD}/R$	98.42 ± (0.17)	84.23 ± (0.71)	83.58 ± (2.06)	73.61 ± (0.35)	89.20 ± (3.04)	83.57 ± (3.18)	14.08 ± (1.01)
$\mathcal{L}_{HD}/COV$	**98.51 ± (0.23)**	**85.52 ± (1.03)**	**85.10 ± (1.46)**	**75.54 ± (1.69)**	**89.73 ± (1.94)**	**85.12 ± (1.99)**	**11.98 ± (1.24)**
$\mathcal{L}_{DCE}$	**98.76 ± (0.26)**	**88.35 ± (1.24)**	88.87 ± (1.41)	**79.88 ± (1.34)**	**89.12 ± (1.36)**	90.09 ± (1.72)	12.45 ± (1.76)
$\mathcal{L}_{DCE}/SA$	98.65 ± (0.17)	87.46 ± (0.34)	**89.42 ± (1.28)**	78.50 ± (0.98)	86.31 ± (0.79)	**92.26 ± (1.93)**	**12.38 ± (4.15)**

Table 8.7: Adaptive weighting results for the cervical and lumbar dataset, where the extension R stands for the rebalance strategy, the extension COV for the Coefficient of Variations method and SA for the SoftAdapt method.

Table 8.7 provides the results of the adaptive weighting methods for the cervical and lumbar vertebrae data. Considering the results of the adaptive methods on the cervical vertebrae data, similar results can be seen among the loss functions used. For boundary loss and Hausdorff distance, the rebalance strategy showed improvements in five metrics compared to the results without the method applied and to the COV method. The SoftAdapt method applied to the combination of Dice loss and cross entropy only improved two of the metrics. Overall, the methods applied led to a decrease in precision, but to an improvement in recall. The results for the lumbar vertebrae do not indicate a clear result compared to the cervical vertebrae data. The rebalance strategy applied to the boundary loss led to an improvement in three metrics. However, for the Hausdorff distance, the COV method was found to improve all metrics. The SoftAdapt method produced an improvement in only three metrics.

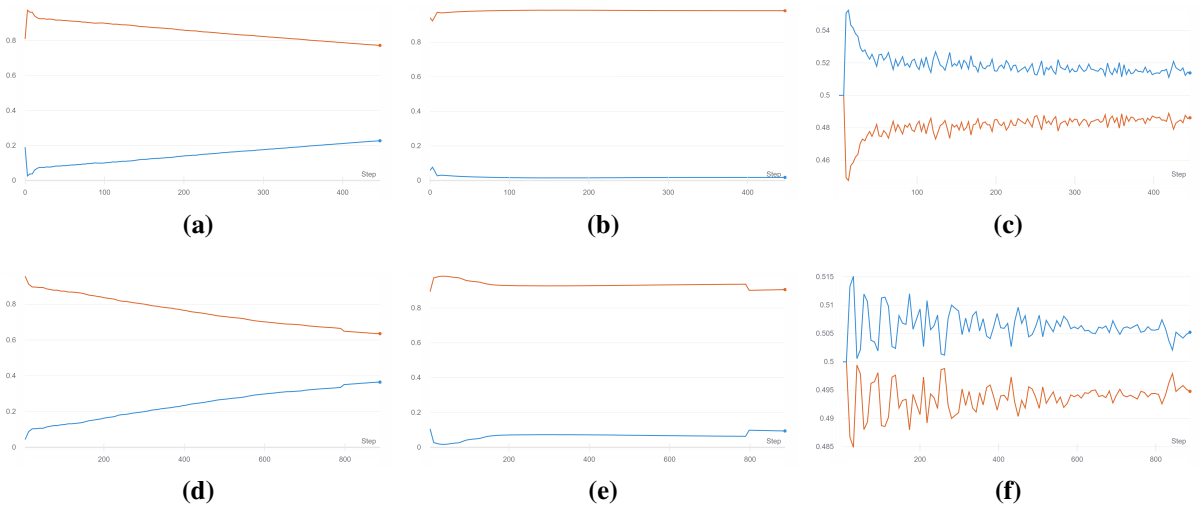


Figure 8.6: The history of the weight values with the adaptive methods applied during the training process, where the orange line represents α and the blue line represents β. The trajectories with the COV method for the $\mathcal{L}_B$ are shown in (a) and (d), the trajectories with the COV method for the $\mathcal{L}_{HD}$ in (b) and (e) and the trajectories of the SoftAdapt method applied to the $\mathcal{L}_{DCE}$ in (c) and (f). The upper and lower row displays the curves for the cervical vertebrae data and the lumbar vertebrae data, respectively.

Next, if the history of the weight values with the applied methods on both datasets in Fig. 8.6 are considered, several things stand out. When examining the COV method applied to the Boundary loss in (a) and in (d), it is noticeable that in the course of the training process the α value decreases monotonously and the β value increases monotonously. Whereby the slope and the decline in (d) are stronger compared to (a). When considering the COV method applied to the Hausdorff distance as shown in (b) and (e), it can be seen that the α parameter dominates the β parameter and remains al-

most constant over the entire training period. There are slight differences in the intensity of the parameters, in (b) β is close to zero and in (d) approximately 0.1, similarly for α where it is close to 1 in (b) and in (d) approximately 0.9.

For the SoftAdapt method applied to the Dice-Cross Entropy loss, both trajectories (c) and (f) show a balanced relationship during training, with β having a stronger weighting compared to α. Further, for the cervical data, the α value settles at 0.48 and for the β value at 0.52. For the lumbar data, there is a smaller difference between the values, α settles at 0.495 and β at 0.505. The rebalance strategy is not presented as it only displays a linear progression of both weights α and β, which are decreased or increased by a constant value per training step.

In a general observation, there are only minor differences in the courses of the weights between the datasets, however, these already led to different results as shown in Table 8.7. In conclusion, it can be said that the greedy method was able to improve the results from the baselines from the first research question and thus represents a general recommendation. The other adaptive weighting methods could only show improvements sporadically and in comparison to the respective static loss function. In addition, the observed results differ between the data sets, which does not lead to a clear recommendation of one method over the other.

8.6 Discussion

The limitations of this work in general and in the individual research questions are considered subsequently. A general limitation lies in the aspect of the available time, which directly affects the training duration and the restriction to selected loss functions and methods. The number of selected epochs was calculated based on the time, which on average takes five hours for a loss function, and on the amount of data seen during the training process. Increasing the epochs could lead to improvements, but they were expected to be relatively small in relation to the results already obtained. The selection of loss functions for answering the first research question are also capped by the time aspect and were therefore reduced to the most frequently used ones in order to establish comparability. Another possible limitation is the two datasets that were used. On the one hand, there is a possible limitation in the number of data subjects provided, which is, however, common in the medical field. To reduce this limitation and a simultaneous overfitting of the model, different data augmentation methods were used. Furthermore, the two datasets are very similar in the distribution of foreground and background, which led to similar results in the overall picture. Therefore, the assumptions and results obtained can only be applied to data that have similar distributions and a similar amount of data subjects. Another limitation is the hardware used, which caps the possibility of using more parameters. In addition, the question of possible modifications of the model architecture for the different structural requirements, which are either lumbar vertebral bodies or whole cervical vertebrae, is a subject for discussion. Also in the context of the used validation strategy, an increase from three to five folds could show a better comparability of the generalisation, but this is also constrained by the available time.

Chapter 9

Summary

This chapter provides a brief overview of the content of this thesis, gives an outlook on further developments in the field of medical segmentation and forms a final conclusion.

9.1 Overview

This thesis gives an overview and analysis of the loss functions used in medical image and volume segmentation. The loss functions were divided into different categories and the features were analysed based on specific scenarios. In addition, methods were explained and applied with the aim of learning loss functions before or during the training process. Subsequently, further methods were tested to adaptively change the parameters of individual components of a loss function or to adaptively adjust the weighting of individual loss functions in a multi loss function. All loss functions and the applied methods were used on two MRI data sets of neck and lumbar vertebrae, which were analysed in detail before the training. The 3D U-Net architecture used and the associated modifications based on the data analysis were also demonstrated and justified. The results were presented with a 3-fold cross validation strategy on a total of seven metrics and visualised in 2D as a time series and in 3D as a volume. The results were presented, discussed in the context of the accompanying analysis and the possible limitations were pointed out.

9.2 Outlook

When predicting the future developments in the field of segmentation or in the general field of deep learning, several things catch the eye that could also be exciting for medical application purposes. Currently, there seems to be an adaptation from the field of Natural Language Proccesing (NLP) into computer vision with regard to transformer architectures. As the publications [Che+21],[Pet+21] and [Cao+21] show, the adaptation

of mechanisms from the transformer architecture can further improve the performance of CNN. Other promising developments deal with the general question of how training can be further optimised. Chaitanya et al. [Cha+20] proposed a new training paradigm for architectures that consist of an encoder and decoder part, in which the encoder and decoder are first trained separately on different loss functions and then finally trained together again. This is based on the assumption that encoders and decoders fulfil different functions in the segmentation process and combines properties of the trainings process of self-supervised learning and supervised learning. Further, Xue et al. [Xue+19] suggest possible improvements by segmenting signed distance maps instead of the binary ground truth as these contain gradually finer information values. Further possible developments in the area of loss functions are offered by the topic of AutoML, which was originally developed for the search of optimal hyperparameters such as batch size or learning rate. AutoML techniques might provide a general heuristic that automatically selects a suitable loss function based on the specified data.

9.3 Conclusion

In order to complete this thesis, a conclusion is drawn in the following. In general, it was possible to achieve good segmentation results on the cervical vertebra data as well as on the lumbar vertebrae bodies. In addition, the time required for the segmentation process by a medical expert, which can take between one and two hours depending on the given data and the quality of the segmentation, was reduced to a couple of seconds. Furthermore, the results showed that the selection of a loss function is not trivial as it can lead to significantly different outcomes in individual metrics. For the cervical vertebrae data, the Tversky-Focal loss was shown to yield the best results. In comparison, the Dice loss provided the best results in three of the seven metrics on the lumbar vertebra data. The methods consisting of a genetic approach and a meta-learning approach to learn loss functions did not obtain comparable results to the results of the static loss functions. Among the adaptive methods, the greedy method was able to achieve sustained improvements on both datasets compared to the results of the static loss functions. The remaining adaptive methods could only achieve isolated and not cross-dataset improvements compared to the corresponding baselines and therefore do not draw a clear picture that lead to a general recommendation.

Appendix A

Hyperparameters

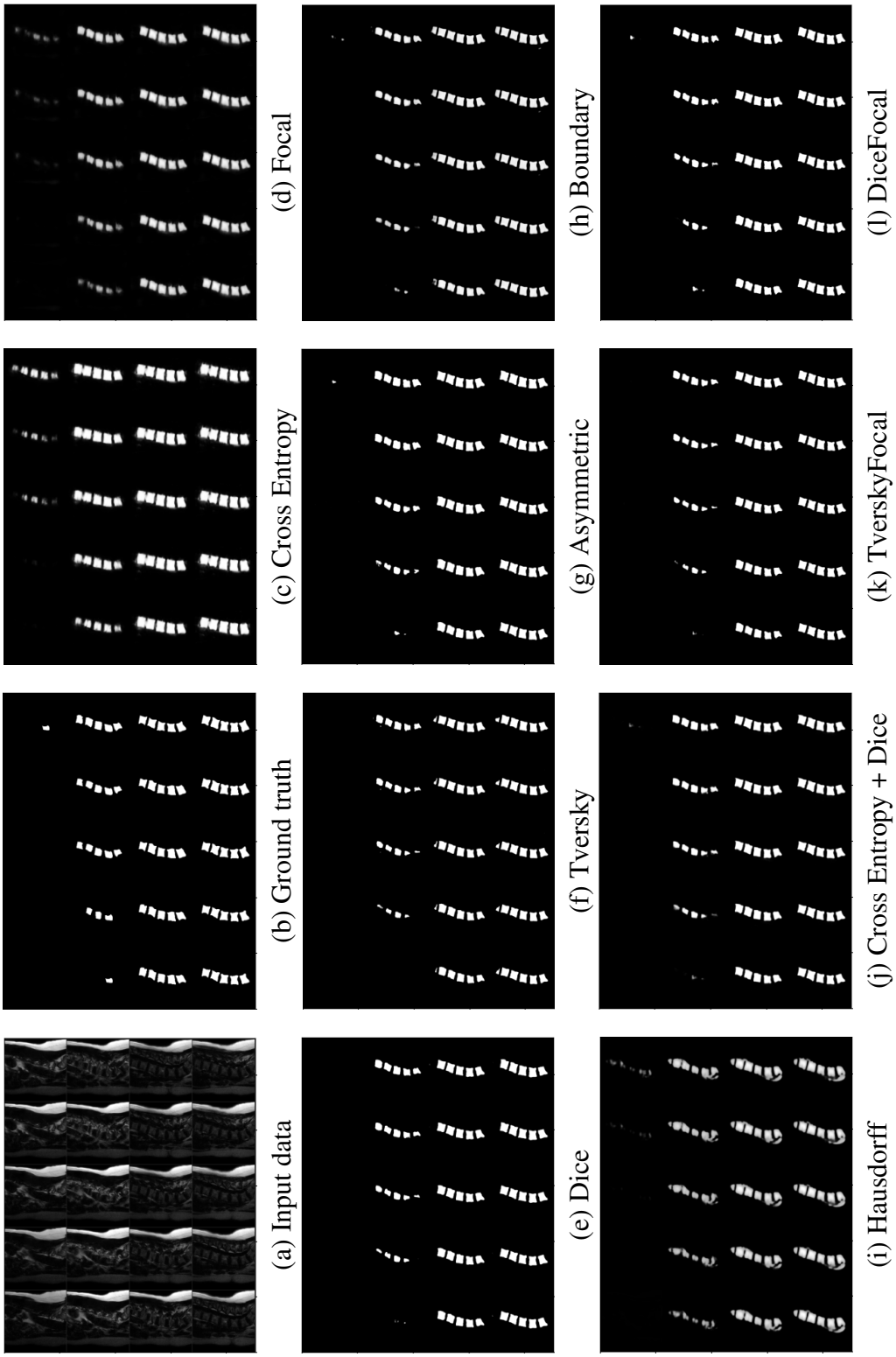
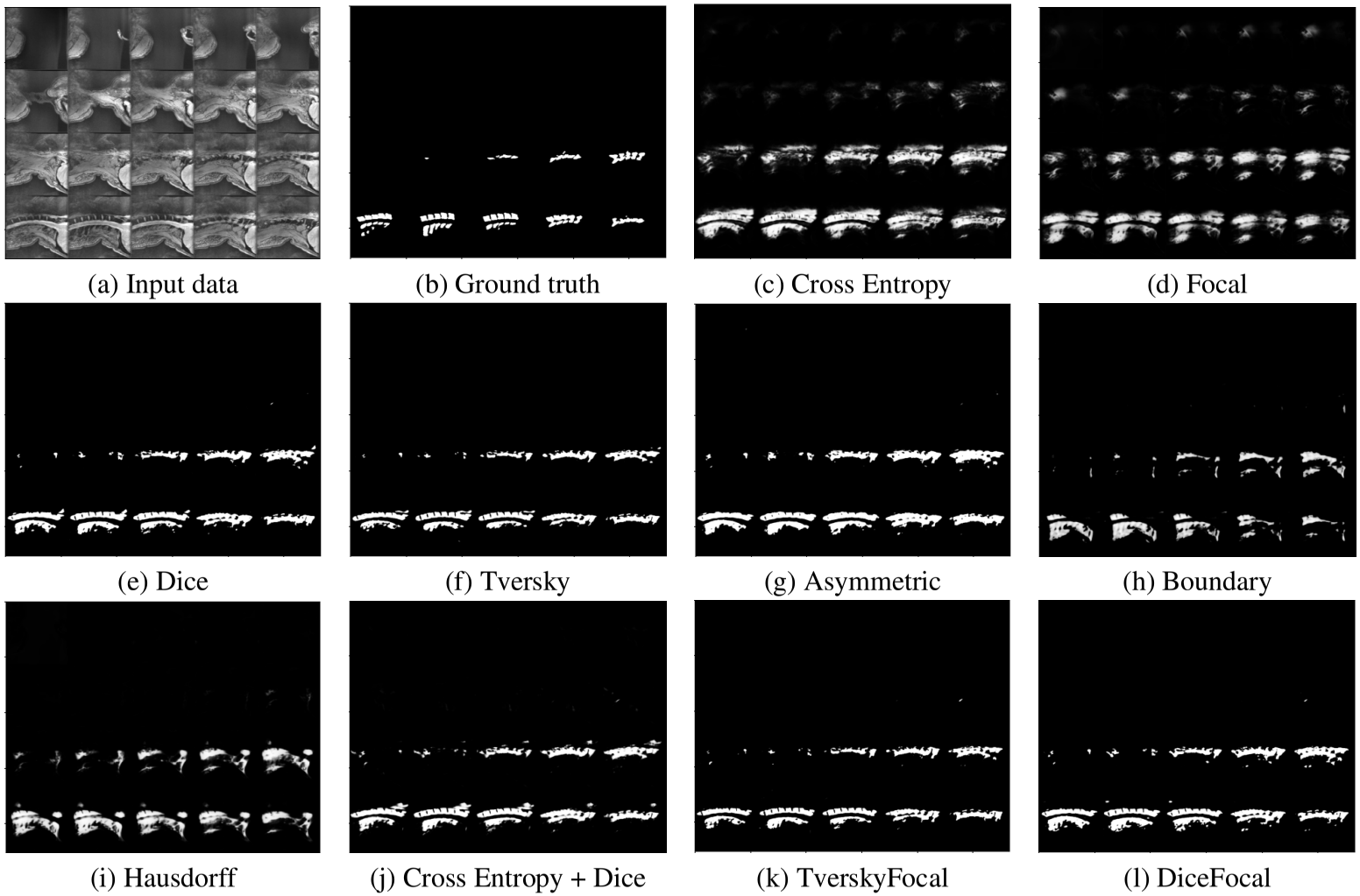


Table A.1: Visual results of the lumbar vertebral bodies as a time series using a random example from the second validation set.



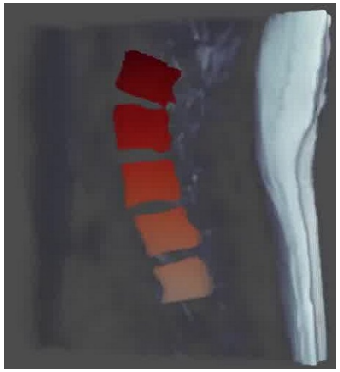
(a) Input data (b) Ground truth (c) Cross Entropy (d) Focal

(e) Dice (f) Tversky (g) Asymmetric (h) Boundary

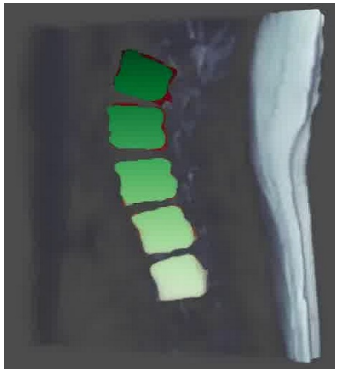
(i) Hausdorff (j) Cross Entropy + Dice (k) TverskyFocal (l) DiceFocal

Table A.2: Visual results of the cervical vertebrae as a time series using a random example from the second validation set.

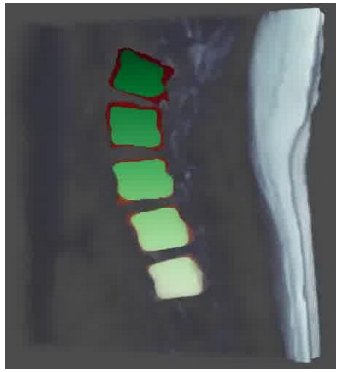
Table A.3: Visual results of the cervical vertebrae as a time series using a random example from the third validation set.



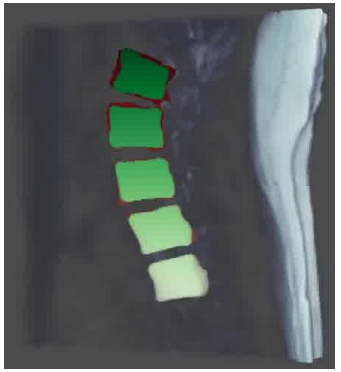
(a) Ground truth



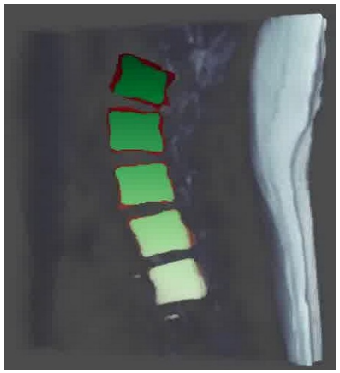
(b) Cross Entropy



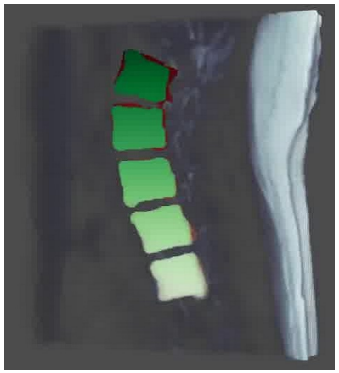
(c) Focal



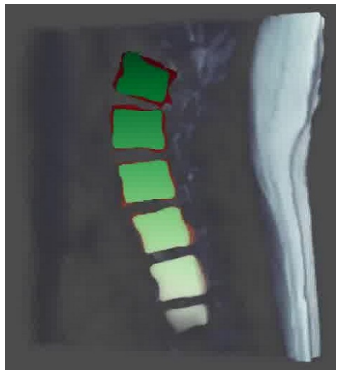
(d) Dice



(e) Tversky

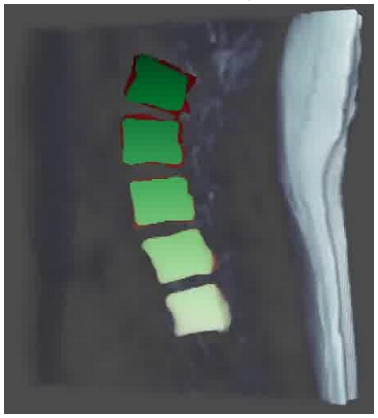


(f) Asymmetric

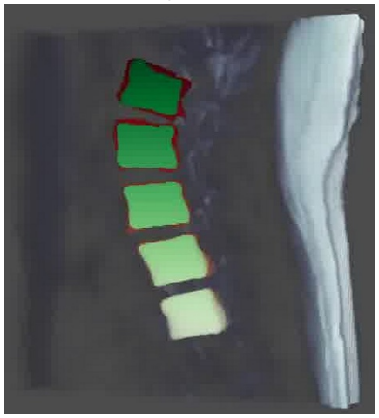


(g) Boundary

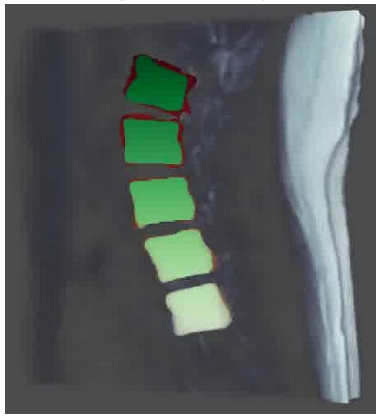
(h) Hausdorff



(i) Cross Entropy + Dice



(j) TverskyFocal



(k) DiceFocal

Table A.4: Visual results of the lumbar vertebral bodies as a volume representation using a random example from the second validation set.

Table A.5: Visual results of the cervical vertebrae as a volume representation using a random example from the second validation set.

	Cervical	Lumbar
Training variables		
k-fold	3	3
batch	4	4
max epochs (m_e)	150	100
learning rate (η_t)	0.002	0.001
validation step (m_v)	5	5
Augmentation		
Affine	40%	30%
Noise	30%	20%
Mixup	100%	100%

Table A.6: Hyperparameters for the training process of cervical and lumbar data.

Loss Function	Parameters
Asymmetric	$\beta = 1.52$
Focal	$\alpha = 0.25, \gamma = 1.45$
Dice-Cross Entropy	$\alpha = 0.5, \beta = 0.5$
DiceFocal	$\alpha = 0.5, \beta = 0.5, \gamma = \frac{5}{3}$
Tversky	$\alpha = 0.7, \beta = 0.3$
TverskyFocal	$\alpha = 0.7, \beta = 0.3, \gamma = \frac{4}{3}$
Boundary	$\alpha = 0.8, \beta = 0.2$
Hausdorff	$\alpha = 0.7, \beta = 0.3$

Table A.7: Parameter values of each loss function.

List of Tables

List of Figures

Bibliography

Published Literature

[Ada20] Nikolas Adaloglou. "Intuitive Explanation of Skip Connections in Deep Learning". In: *https://theaisummer.com/* (2020). URL: https://theaisummer.com/skip-connections/.

[AIB20] M. Awais, M. T. B. Iqbal, and S. -H. Bae. "Revisiting Internal Covariate Shift for Batch Normalization". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020), pp. 1–11. DOI: 10.1109/TNNLS.2020.3026784.

[AK18] Nabila Abraham and Naimul Mefraz Khan. *A Novel Focal Tversky loss function with improved Attention U-Net for lesion segmentation*. 2018. arXiv: 1810.07842 [cs.CV].

[And+21] Vincent Andrearczyk et al. "Overview of the HECKTOR Challenge at MICCAI 2020: Automatic Head and Neck Tumor Segmentation in PET/CT". In: Jan. 2021, pp. 1–21. ISBN: 978-3-030-67193-8. DOI: 10.1007/978-3-030-67194-5_1.

[BB12] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *J. Mach. Learn. Res.* 13.null (Feb. 2012), pp. 281–305. ISSN: 1532-4435.

[Bec+21] Sarah Bechtle et al. *Meta-Learning via Learned Loss*. 2021. arXiv: 1906.05374 [cs.LG].

[Bie20] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[Boy+06] Yuri Boykov et al. "An Integral Solution to Surface Evolution PDEs Via Geo-cuts". In: vol. III. May 2006, pp. 409–422. ISBN: 978-3-540-33836-9. DOI: 10.1007/11744078_32.

[Bur+19] Egon Burian et al. “Lumbar muscle and vertebral bodies segmentation of chemical shift encoding-based water-fat MRI: The reference database MyoSegmenTUM spine”. In: *BMC Musculoskeletal Disorders* 20 (Apr. 2019). DOI: 10.1186/s12891-019-2528-x.

[Cao+21] Hu Cao et al. *Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation*. 2021. arXiv: 2105.05537 [eess.IV].

[CAU47] A. CAUCHY. “Methode generale pour la resolution des systemes d’equations simultanees”. In: *C.R. Acad. Sci. Paris* 25 (1847), pp. 536–538. URL: https://ci.nii.ac.jp/naid/10026863174/en/.

[Cha+20] Krishna Chaitanya et al. *Contrastive learning of global and local features for medical image segmentation with limited annotations*. 2020. arXiv: 2006.10511 [cs.CV].

[Che+17] Liang-Chieh Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: 1606.00915 [cs.CV].

[Che+18] Zhao Chen et al. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks*. 2018. arXiv: 1711.02257 [cs.CV].

[Che+21] Jieneng Chen et al. “TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation”. In: *CoRR* abs/2102.04306 (2021). arXiv: 2102.04306. URL: https://arxiv.org/abs/2102.04306.

[Chi92] Nancy Chinchor. “MUC-4 evaluation metrics”. In: *MUC*. 1992.

[Çiç+16] Özgün Çiçek et al. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. arXiv: 1606.06650 [cs.CV].

[Dar59] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. or the Preservation of Favored Races in the Struggle for Life. London: Murray, 1859.

[Den+09] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12 (July 2011), pp. 2121–2159.

[Dia+20] Foivos I. Diakogiannis et al. “ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 162 (Apr. 2020), pp. 94–114. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2020.01.013. URL: http://dx.doi.org/10.1016/j.isprsjprs.2020.01.013.

[Fa19] WA Falcon and .al. "PyTorch Lightning". In: 3 (2019). URL: https://github.com/PyTorchLightning/pytorch-lightning.

[For+12] Félix-Antoine Fortin et al. "DEAP: Evolutionary algorithms made easy". In: *Journal of Machine Learning Research, Machine Learning Open Source Software* 13 (July 2012), pp. 2171–2175.

[GB10] Xavier Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 2010), pp. 249–256.

[GM20] Santiago Gonzalez and Risto Miikkulainen. *Improved Training Speed, Accuracy, and Data Utilization Through Loss Function Optimization*. 2020. arXiv: 1905.11528 [cs.LG].

[Goo+14] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[Gro+20] Rick Groenendijk et al. *Multi-Loss Weighting with Coefficient of Variations*. 2020. arXiv: 2009.01717 [cs.CV].

[Has+19] Seyed Raein Hashemi et al. "Asymmetric Loss Functions and Deep Densely-Connected Networks for Highly-Imbalanced Medical Image Segmentation: Application to Multiple Sclerosis Lesion Detection". In: *IEEE Access* 7 (2019), pp. 1721–1735. ISSN: 2169-3536. DOI: 10.1109/access.2018.2886371. URL: http://dx.doi.org/10.1109/ACCESS.2018.2886371.

[He+15a] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[He+15b] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[HO01] Nikolaus Hansen and Andreas Ostermeier. "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evolutionary Computation* 9 (June 2001), pp. 159–195. DOI: 10.1162/106365601750190398.

[HTM19] A. Ali Heydari, Craig A. Thompson, and Asif Mehmood. *SoftAdapt: Techniques for Adaptive Loss Weighting of Neural Networks with Multi-Part Loss Functions*. 2019. arXiv: 1912.12355 [cs.LG].

[IS15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[Ise+18] Fabian Isensee et al. *nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation*. 2018. arXiv: 1809.10486 [cs.CV].

[IVH21] Andrei Iantsen, Dimitris Visvikis, and Mathieu Hatt. "Squeeze-and-Excitation Normalization for Automated Delineation of Head and Neck Primary Tumors in Combined PET and CT Images". In: *Head and Neck Tumor Segmentation*. Ed. by Vincent Andrearczyk, Valentin Oreiller, and Adrien Depeursinge. Cham: Springer International Publishing, 2021, pp. 37–43. ISBN: 978-3-030-67194-5.

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[Ker+21] Hoel Kervadec et al. "Boundary loss for highly unbalanced segmentation". In: *Medical Image Analysis* 67 (Jan. 2021), p. 101851. ISSN: 1361-8415. DOI: 10.1016/j.media.2020.101851. URL: http://dx.doi.org/10.1016/j.media.2020.101851.

[KL51] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[KS19] Davood Karimi and Septimiu E. Salcudean. *Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks*. 2019. arXiv: 1904.10030 [eess.IV].

[Lec+00] Yann Lecun et al. "Efficient BackProp". In: (Aug. 2000).

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[LH16] Ilya Loshchilov and Frank Hutter. *CMA-ES for Hyperparameter Optimization of Deep Neural Networks*. 2016. arXiv: 1604.07269 [cs.NE].

[Li+18] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: 1712.09913 [cs.LG].

[Lin+18] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[LJD19] Shikun Liu, Edward Johns, and Andrew J. Davison. *End-to-End Multi-Task Learning with Attention*. 2019. arXiv: 1803.10704 [cs.CV].

[MHN13] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013.

[Mii+17] Risto Miikkulainen et al. *Evolving Deep Neural Networks*. 2017. arXiv: 1703.00548 [cs.NE].

[NH10] Vinod Nair and Geoffrey Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair". In: vol. 27. June 2010, pp. 807–814.

[Okt+18] Ozan Oktay et al. *Attention U-Net: Learning Where to Look for the Pancreas*. 2018. arXiv: 1804.03999 [cs.CV].

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[Pet+21] Olivier Petit et al. *U-Net Transformer: Self and Cross Attention for Medical Image Segmentation*. 2021. arXiv: 2103.06104 [eess.IV].

[PSO20] Fernando Pérez-García, Rachel Sparks, and Sebastien Ourselin. "TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning". In: *arXiv:2003.04696 [cs, eess, stat]* (Mar. 2020). arXiv: 2003.04696. URL: http://arxiv.org/abs/2003.04696 (visited on 03/11/2020).

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[Rij79] C. J. Van Rijsbergen. *Information Retrieval*. 2nd. USA: Butterworth-Heinemann, 1979. ISBN: 0408709294.

[San+19] Shibani Santurkar et al. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML].

[SBX20] Hyunseok Seo, Maxime Bassenne, and Lei Xing. "Closing the Gap Between Deep Neural Network Modeling and Biomedical Decision Making Metrics in Segmentation via Adaptive Loss Functions". In: *IEEE Transactions on Medical Imaging* PP (Oct. 2020), pp. 1–1. DOI: 10.1109/TMI.2020.3031913.

[SEG17] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. *Tversky loss function for image segmentation using 3D fully convolutional deep networks*. 2017. arXiv: 1706.05721 [cs.CV].

[Sør48] T. Sørenson. *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. Biologiske skrifter. I kommission hos E. Munksgaard, 1948. URL: https://books.google.de/books?id=rpS8GAAACAAJ.

[Sta+19] Kenneth Stanley et al. “Designing neural networks through neuroevolution”. In: *Nature Machine Intelligence* 1 (Jan. 2019). DOI: 10.1038/s42256-018-0006-z.

[Tve77] Amos Tversky. “Features of Similarity”. In: *Psychological Review* 84.4 (1977), pp. 327–352. DOI: 10.1037/0033-295X.84.4.327.

[UVL17] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization*. 2017. arXiv: 1607.08022 [cs.CV].

[Vap92] V. Vapnik. “Principles of Risk Minimization for Learning Theory”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson, and R. P. Lippmann. Vol. 4. Morgan-Kaufmann, 1992. URL: https://proceedings.neurips.cc/paper/1991/file/ff4d5fbbafdf976cfdc032e3bde78de5-Paper.pdf.

[WH18] Yuxin Wu and Kaiming He. *Group Normalization*. 2018. arXiv: 1803.08494 [cs.CV].

[WW62] Author(s) B. P. Welford and B. P. Welford. “Note on a method for calculating corrected sums of squares and products”. In: *Technometrics* (1962), pp. 419–420.

[Xu+15] Bing Xu et al. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. arXiv: 1505.00853 [cs.LG].

[Xue+19] Yuan Xue et al. *Shape-Aware Organ Segmentation by Predicting Signed Distance Maps*. 2019. arXiv: 1912.03849 [cs.CV].

[Zha+18] Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization*. 2018. arXiv: 1710.09412 [cs.LG].

[Zho+18] Zongwei Zhou et al. “UNet++: A Nested U-Net Architecture for Medical Image Segmentation”. In: *CoRR* abs/1807.10165 (2018). arXiv: 1807.10165. URL: http://arxiv.org/abs/1807.10165.

Internet Sources

[Wik21a] Wikipedia. *Cross-validation (statistics) — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Cross-validation%20(statistics)&oldid=1012087730. [Online; accessed 12-April-2021]. 2021.

[Wik21b] Wikipedia. *Entropy — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Entropy&oldid=1026721641. [Online; accessed 06-June-2021]. 2021.

[Wik21c] Wikipedia. *F-distribution — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=F-distribution&oldid=1015376810. [Online; accessed 07-June-2021]. 2021.

[Wik21d] Wikipedia. *Finite difference coefficient — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Finite%20difference%20coefficient&oldid=987174365. [Online; accessed 12-April-2021]. 2021.

[Wik21e] Wikipedia. *Hausdorff distance — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Hausdorff%20distance&oldid=1010268981. [Online; accessed 26-May-2021]. 2021.

[Wik21f] Wikipedia. *Kullback–Leibler divergence — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Kullback%E2%80%93Leibler%20divergence&oldid=1026917999. [Online; accessed 20-June-2021]. 2021.

[Wik21g] Wikipedia. *Softmax function — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Softmax%20function&oldid=1012399737. [Online; accessed 12-April-2021]. 2021.