Slovak university of technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-182905-92143

Bc. Sandro Rybárik

Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí

Master Thesis

Supervisor: prof. Ing. Vanda Benešová, PhD.

2022, May

Slovak university of technology in Bratislava

Faculty of Informatics and Information Technologies

FIIT-182905-92143

Bc. Sandro Rybárik

Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí

Master Thesis

Study programme: Intelligent Software Systems

Study field: Computer Science

Training workplace: Institute of Computer Engineering and Applied Informatics

Thesis supervisor: prof. Ing. Vanda Benešová, PhD.

2022, May

Slovak University of Technology in Bratislava
Ústav počítačového inžinierstva a aplikovanej informatiky

Faculty of Informatics and Information Technologies
Academic year: 2021/2022
Reg. No.: FIIT-182905-92143

MASTER THESIS TOPIC

Student:	**Bc. Sandro Rybárik**
Student's ID:	92143
Study programme:	Intelligent Software Systems
Study field:	Computer Science
Thesis supervisor:	prof. Ing. Vanda Benešová, PhD.
Head of department:	Ing. Katarína Jelemenská, PhD.

Topic: **Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí**

Language of thesis: English

Specification of Assignment:

V oblasti praktickej medicíny sa sníma veľké množstvo vizuálnych dát, ktoré je potrebné následne odborne spracovať, pričom tento proces je časovo náročný a vyžaduje prácu vysokokvalifikovaných odborníkov. Pri spracovaní vizuálnych dát v medicíne, ako sú napríklad dáta z počítačovej tomografie (CT) alebo magnetickej rezonancie (MRI), nachádzajú metódy počítačového videnia stále významnejšie uplatnenie. Významnými problémovými oblasťami sú napríklad detekcia a segmentácia anomálií, registrácia multimodálnych dát a analýza dát za účelom diagnostiky alebo liečby. Analyzujte súčasný stav problematiky využitia počítačového videnia pri spracovaní vizuálnych dát v medicíne. Zamerajte sa predovšetkým na metódy využívajúce hlboké neurónové siete pre spracovanie CT alebo MRI snímkov za účelom diagnostiky nemoci diagnostikovateľnej na jednom anatomickom orgáne, napr. na pľúcach. Navrhnite metódu spracovania CT alebo MR snímok pľúc, ktorú následne budete realizovať softvérovým prototypom s využitím knižníc a rámcov na spracovanie medicínskych dát. Riešenie overte experimentom s reálnymi dátami z medicínskych modalít. Vyhodnoťte presnosť, robustnost a časovú efektívnosť spracovania. Výsledky porovnajte s publikovanými riešeniami v tejto oblasti.

Deadline for submission of Master thesis:	16. 05. 2022
Approval of assignment of Master thesis:	27. 04. 2022
Assignment of Master thesis approved by:	prof. Ing. Ivan Kotuliak, PhD. – Study programme supervisor

Declaration of Honor

I honestly declared that I have prepared this work independently, on the basis of consultations and using the cited literature.

Bc. Sandro Rybárik

Acknowledgment

I would like to thank my parents for ever lasting support during my study and during writing this thesis. Next I would like to thank Lucia Malinová for support during lockdowns and harder times during writing this thesis.

I would like to thank my supervisor prof. Ing. Vanda Benesova PhD. for valuable feedback during experiments and for guidance during writing this thesis.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Intelligent Software Systems

Autor: Bc. Sandro Rybárik
Diplomová práca: Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí
Vedúci diplomového projektu: prof. Ing. Vanda Benešová, PhD.
Máj 2022

Cieľom tejto práce bolo navrhnúť riešenie automatickej segmentácie pľúcnych infekcií na CT snímkach. V prvej časti sme analyzovali existujúce štúdie a metódy riešenia tohto problému. Navrhované riešenie sme vo veľkej miere zakladali na prácach s podobným zameraním a známych architektúrach v tejto oblasti. Orezali sme vstupné dáta pomocou natrénovaného 2D U-Net modelu, tak aby na výstupe boli zachytené len pľúca a tým sme zjednodušili proces trénovania. Veľkosť datasetu sme zredukovali o 58,9%. Vysokú mieru rozdielu jednotlivých tried pri segmentácii sme adresovali našou modifikovanou stratovou funkciou DiceDiceLoss, ktorá minimalizovala počet falošne pozitívnych a falošne negatívnych tried pri výstupnej segmentácií. Stratovú funkciu sme navhrli, tak aby sa dala modifikovať podľa datasetu a prítomnej miery nerovnováhy segmentačných tried. Navrhli sme prototyp založený na architektúre 3D U-Net v základnej konfigurácií a tento natrénovaný model sme vyhodnotili pomocou trojitej krížovej validácie. Výsledky sme porovnali voči DiceFocalLoss, kde sme dosiahli signifikantne lepšie výsledky na metrike Dice.

Annotation

Slovak University of Technology Bratislava
Faculty of Informatics and Information Technologies
Degree Course: Intelligent Software Systems

Author: Bc. Sandro Rybárik
Master's Thesis: Medical image processing using methods of computer vision and deep neural networks
Supervisor: prof. Ing. Vanda Benešová, PhD.
2022, May

This work aimed to design a solution for the automatic segmentation of pulmonary infections in CT images. In the first part, we analyzed existing studies and methods to solve this particular problem. We based our proposed solution on work with a similar focus and well-known architectures in this area. We cropped the input images using a trained 2D U-Net model, so that only the lungs are captured on the output, thus simplifying the training process. We have reduced the size of the dataset by 58.9%. We addressed the imbalance of segmentation classes during segmentation by the modified loss function DiceDiceLoss, which minimized the number of false-positive and false-negative classes in output segmentation. We designed the loss function so that it could be modified according to the dataset and the present degree of imbalance of segmentation classes. We designed a prototype based on the 3D U-Net architecture in the basic configuration and evaluated this trained model using three-fold cross-validation. We compared the results against DiceFocalLoss achieving significantly better performance on the Dice metric.

Contents

Chapter 1

Introduction

Many challenges in medicine have risen recently. The need to scale healthcare is more necessary than ever. The recent COVID-19 pandemic shows how much more work there is to be done in terms of speeding up diagnostics. At the end of January 2020 WHO organization declared COVID-19 as the Public Health Emergency of International Concern [45]. As of 3 December 2021, the WHO reported 263,563,622 worldwide cases and 5,232,562 confirmed deaths and administrated 7,864,123,038 doses of vaccine[45]. Reverse-transcription polymerase chain reaction (RT-PCR) became one of the widely used methods to detect viral nucleotides from specimens obtained with the oropharyngeal swab, nasopharyngeal swab, bronchoalveolar lavage, or tracheal aspirate [4]. As COVID-19 disease has infected millions of patients and produced an enormous number of datasets such as numerical data of pandemic, CT scans, and other modalities; see Fig. 1.1. It turns out that a CT scan is a reliable method of COVID-19 diagnostics with a high prediction rate of a disease [39, 1]. On the other hand, this method requires a radiology doctor to examine findings and make the judgment on whether a patient is sick or healthy. Although we can't replace doctors with a deep learning model, we can help to

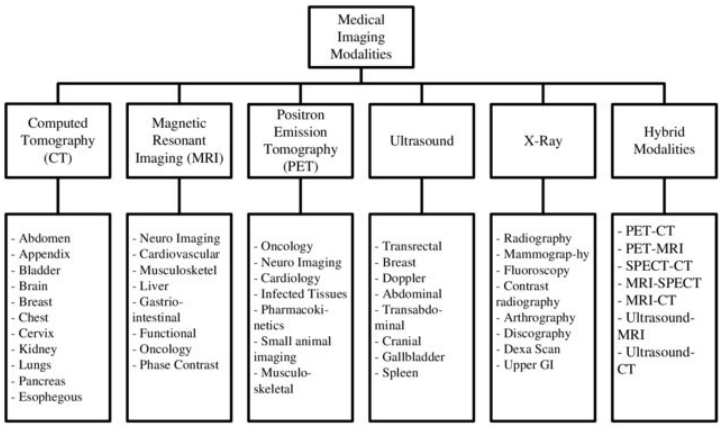

Figure 1.1: Topology of Medical Imaging Modalities [3].

identify regions of interest to fasten examinations of these findings.

Our focus was primarily on lesion segmentation inside the lungs. The goal is to provide quality healthcare to the masses with help of automation tools such as machine learning. Hospitals already gather data on diagnostics and patient treatments. We can take advantage of these datasets and provide supporting automation tools for doctors in specific fields. Development of such tools quickly as possible using existing work as a starting point is the key to helping as soon as possible. Such tools should be generally applicable to a range of patients with similar conditions. Deep learning has recently shown its potential in medical imaging analysis. We examined deep learning models and the state of art approaches to better understand them. These models are a combination of numerous parameters and components which can be modified to achieve better results on a given task. We run experiments to benchmark our modifications against state-of-the-art models.

Such help can be offered with help of computer vision and deep learning methods to guide doctors' attention on specific regions of CT scans where machine learning models predicted a higher probability of disease occurrence. Medical imaging

analysis as the field is being heavily improved upon as [42] is suggesting.

1.1 COVID-19 Segmentation Challenge

The widespread disease also generated datasets and competitions in which we participated. Common imaging modalities for evaluation of COVID-19 infections are chest radiographs (CT), chest computerized tomography, and ultrasound which is used less often. The most accurate modality that identifies infection is chest CT [5, 50]. Frequent findings in the chest of patients were ground-glass opacities (GGO) and pneumonic consolidations [38]. COVID-19 challenge[38] brought access to 199 CT images annotated by experts which we used during building our prototype solution. Over a thousand teams registered for the challenge and 98 completed it through the test phase.

This allowed participants and us to utilize quality data as the number of larger datasets is not publicly accessible and available.

Chapter 2

Analysis

Modern image segmentation methods build upon the foundation laid out by convolutional neural networks (CNN) in 2012. In past years CNNs overtake the image segmentation field yielding better results. We focused on relevant publications in this research area and provide important and influential works that shaped the field.

2.1 Traditional Approach of Computer Vision

To betterx understand and appreciate modern methods we should compare them with their predecessors.

These methods can be marked as descriptive analysis. As it involves defining an understandable mathematical model which describes the event that we wish to observe. A further collection of data about the process, forming hypotheses according to the data, and validating these hypotheses through comparing results of the model with real outcomes[6].

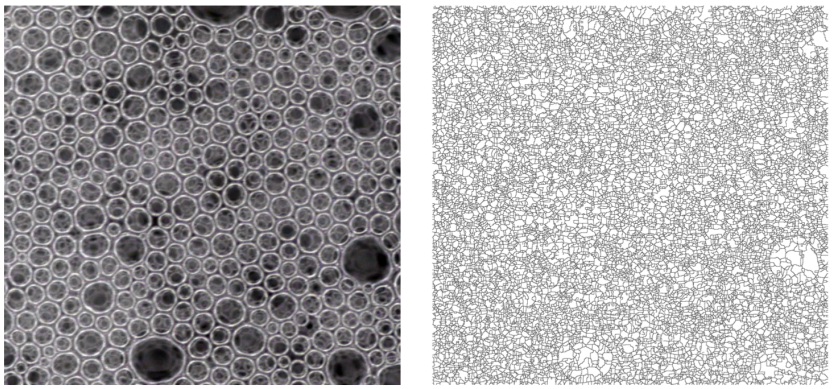

Figure 2.1: Example of using watershed transform for segmenting bubbles [43].

This requires expertise and a good understanding of a problem. As an architect of such a model must be well informed about a domain of a problem we can safely assume that it is also time demanding. This step was fully automated as machine learning models firstly appeared and shifted focus more on datasets than carefully engineering mathematical models by hand.

Traditional computer vision employs morphological operations, applying adaptive thresholds, and watershed transform (Fig. 2.1 to form a segmentation algorithm for specific input images. These operations are primarily applicable to gray-scale images and require careful tuning of parameters to achieve the desired result.

Active Contours In 1998 Kass et al. introduced "Snakes: Active Contour Models"[21] where they proposed an algorithm for finding the contours of an object. Using an edge-detector that depends on the gradient of the image. Similar work [9, 8, 30] used gradient-based technique to detect the contours of an object.

These algorithms are parametric and parameters must be adjusted to specific input images. Requires testing and trying before useful results are obtained.

2.2 Deep Learning

What is deep learning? According to authors L. Deng and D. Yu deep learning is a class of machine learning algorithms. Firstly, uses a cascade of multiple layers of nonlinear processing units for feature extraction and transformation where each successive layer uses the output from the previous layer as input. Secondly, learns multiple levels of representation that correspond to different levels of abstraction [12].

Deep learning involves searching for rules that describe a phenomenon and forming a predictive model that is trying to minimize the error between the real and the predicted result [6].

Convolutional Neural Networks (CNN) In 2012 breakthrough happened at ImageNet Large Scale Visual Recognition Challenge where AlexNet[24], a convolutional neural network won the competition by a huge margin. This started a new generation of convolutional neural network designs and improvements and it lasts until today. Machine learning as a field started to rise in recent years due to accessibility to computation power primarily to cheaper graphic cards. Generally, convolutional neural networks (CNNs) mostly consist of convolutional layers, nonlinear layers, and pooling layers. Convolutional layers apply convolution operation on input producing feature maps as an output. These feature maps contain encoded information about the image or volume. Convolution operation involves filters with size, padding, and stride which defines the output of convolution. Filters also called kernels can be thought of as weights in a multi-layer perceptron neural network. This Convolution is defined by a rectangular size kernel for example 3×3. With more filters come higher computational complexity and more power is required. But in a high-level view convolution traverse input image or volume and outputs feature maps. Nonlinear layers apply activation function on

feature maps usually element-wise which enables modeling of non-linear functions. Nwankpa et al. in 2018 [34] summarised numerous activation functions such as ReLU, Sigmoid, leaky ReLU. Each activation function solves a different kinds of problems during training. There are also domain-specific tasks that require tuning the activation function to accomplish goals. Pooling layers replace the rectangular neighborhood of values usually with max or mean values. These pooling operations are applied on feature maps to reduce their spatial dimension and reduce computational complexity on further convolution layers. We demonstrated the simplified version of basic convolutional network architecture in Fig. 2.2.

ResNet Architecture At the end of 2015, He et al. published an influencing paper about Deep Residual Learning for Image Recognition and introduced ResNet architecture [15]. ResNet has been primarily used for classification tasks but it was used in DeepLabV3[11] semantic segmentation model as a basis. As the authors stated, deeper neural networks should in theory perform better as they should better approximate the function describing the learned dataset. But they empirically showed that deeper networks have a higher error rate than shallower ones. This was previously solved by other techniques such as normalization and

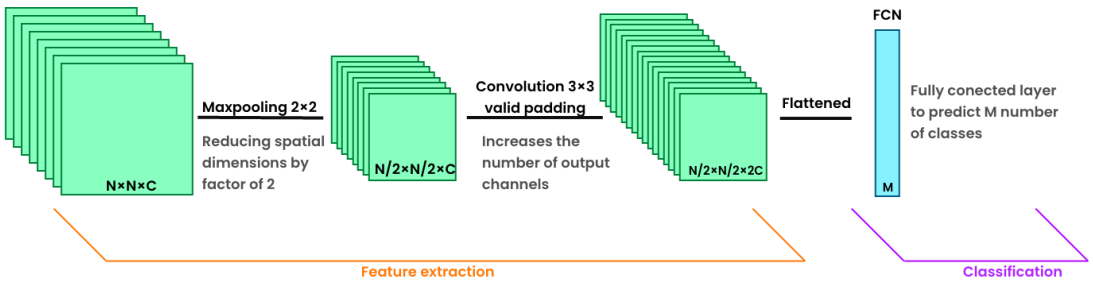

Figure 2.2: Oversimplified version of the most basic operations in convolutional neural networks. These basic operation in feature extraction part are often stacked to increase feature extraction ability.

dropout. Their solution did not increase the number of operations required.

They introduced skip connection as an answer to overcoming difficulties during the training of deeper networks. Skip connection passed input from the previous layer to end of the N-layer block summing with an output of N-layer block output and passing it into activation function. This whole block with a skip connection is called a residual block. The presented network architectures were composed of these residual blocks and compared to so-called plain networks without skip connections. They compared them on classification tasks and proved residual blocks indeed increased the performance of the model without sacrificing performance. Skip connections also helped shallower networks to converge faster. They used skipped connection every three layers and created architecture ResNet-152 with a total of 152 layers and won 1st place at ILSVRC 2015.

2.3 Influential Architectures for Segmentation Tasks

2.3.1 Fully Convolutional Network architecture

Long et al. 2014 proposed Fully Convolutional Networks for semantic segmentation (FCN) [27]. With the novelty of taking an image of arbitrary size as an input. This paper proved that it is possible to train the model with convolutional layers end to end and get state-of-the-art results. The ability to segment arbitrary input image size was due to the replacement of flattening layers with 1×1 convolution to perform segmentation. A convolutional layer is defined as a three-dimensional array of size $h \times w \times c$ where c is a number of color channels and $h \times w$ is the spatial size of an image. It is assumed that convolutional networks are translation invariant. This claim was challenged in a recent paper [22] and authors had shown that convolutional networks encoded absolute spatial information of objects. This

could be further analyzed from the perspective of dataset augmentation to create more generalized models.

They also introduced transposed convolution referred to as deconvolution in the paper for up-sampling. These layers can be learned too. The combination of finer and coarser information resulted in higher quality segmentation due to additional spatial information provided during up-sampling. Up-sampling process benefited from previous feature maps which led to the model making predictions with respect to global structure.

2.3.2 U-Net architecture

Ronneberger et al. in 2015 introduced U-net convolutional network for biomedical image segmentation which laid the foundation for future work in the image segmentation field [37]. U-Net architecture consisted of two parts, a contracting path, and an expansive path; see Fig. 2.3. The contracting path involved a double 3×3 convolution followed by a ReLU activation function followed by 2×2 max pooling layer with stride 2 and an increasing number of filters by a factor of two. Each convolution layer was followed by the ReLU activation function. This basic block was repeated four times. At the end of the contracting path was the final 3×3 convolution. The expansion path was composed by up-sampling of the feature map followed by 2×2 convolution which halved the number of features every block and concatenating with previous feature maps by skipped connections followed by two 3×3 convolutions. An important step is to copy and crop feature maps from contracting path blocks into expansion path blocks. This is called skipped connection to introduce spatial information from previous layers and achieve better up-sampling results. Final layer a 1×1 convolution mapped 64 feature vectors to the number of output classes in this case task was binary classification therefore

outputting the number of filters was 2. To train U-Net they used the so-called energy function computed pixel-wise softmax over the final feature map combined with a cross-entropy loss function.

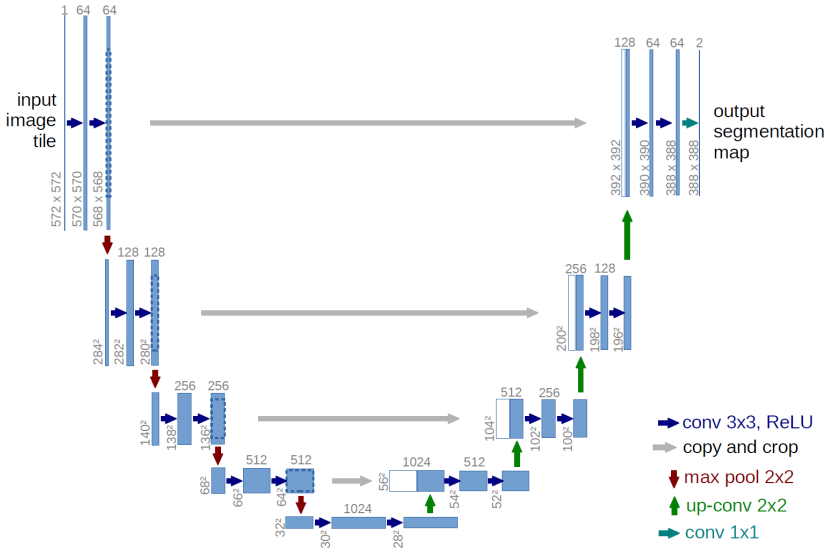

Figure 2.3: Overview of U-Net architecture that influenced segmentation architectures[37].

2.3.2.1 3D-Unet architecture

In 2016 Çiçek et al. [51] introduced 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation and extended successful U-Net architecture for a 3D volumetric dataset. They modified convolutions, max pooling from 2D to 3D; see Fig. 2.4. The Contracting path had four steps. Each layer consisted of two $3 \times 3 \times 3$ convolutions each followed by ReLU activation function, and then $2 \times 2 \times 2$ max pooling with stride of 2. The expansion path similarly used two $2 \times 2 \times 2$ convolutions by the stride of 2, followed by two $3 \times 3 \times 3$ convolutions each followed by ReLU. The last layer in the network did $1 \times 1 \times 1$ convolution in order to reduce the number of output channels to the number of output classes. They also avoided bottlenecks by doubling the number of channels before max pooling

operation. Batch normalization was used before each ReLU to speed up convergence. The model trained on 2D annotated slices and performed segmentation on 3D 3D-Unet achieved performance equivalent to the 2D U-Net network with their fully-automated setup in the Xenopus Kidney segmentation task.

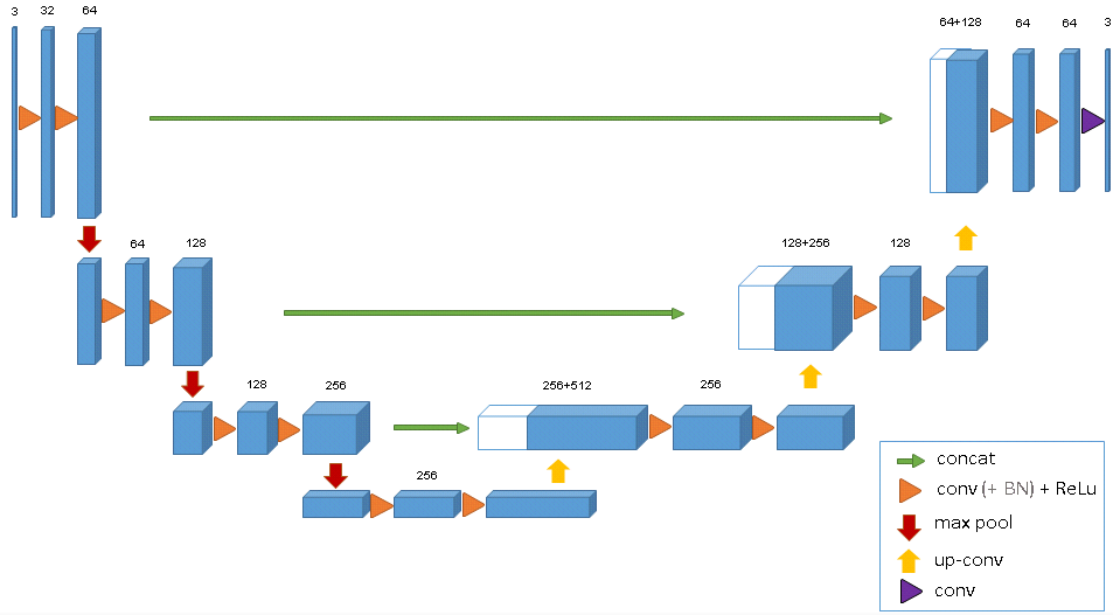

Figure 2.4: Architectural overview of 3D-Unet network[51]. Based on origin 2D UNet version of Ronneberger et al.[37]

2.3.2.2 UNet++ architecture

In 2018 Zhou et al. introduced new architecture called UNet++ for medical image segmentation [49]. An encoder-decoder architecture with added dense skip pathways reduced the gap between the feature maps of the encoder and decoder. The main addition to U-Net architecture was the re-design of skipped connections between encoder-decoder; see Fig. 2.5. The main difference is in delivering feature maps from encoder to decoder. The number of skipped connections row-wise is referenced as semantic level. In U-Net they were cropped and copied, in U-Net++ they were passed through additional convolution layers where each

convolution is preceded by a concatenation layer that joined input with the corresponding up-sampled output of the lower dense block. Next, they proposed to use deep supervision [25] which enabled their ability to operate the model in two modes. An accurate mode and a fast mode. The accurate mode averages the final segmentation branches. The fast mode chooses only one segmentation map from segmentation branches. The selection of branch in fast mode is determined by model pruning and speed gain. All convolutions used kernel size of 3×3 or $3 \times 3 \times 3$ for 3D data. An Adam optimizer with a learning rate of $3e-4$ was used during training. The model was trained with a loss function consisting of binary cross-entropy and dice coefficient for each semantic level. In summary, UNet++ is different from U-Net in two major ways. First Unet++ has convolution layers between skipped pathways. And second Unet++ is using deep supervision to fine-tune performance gains. They also benchmark the model against U-Net and Wide U-net and UNet++ achieved an average IoU gain of 3.9 and 3.4 points over others.

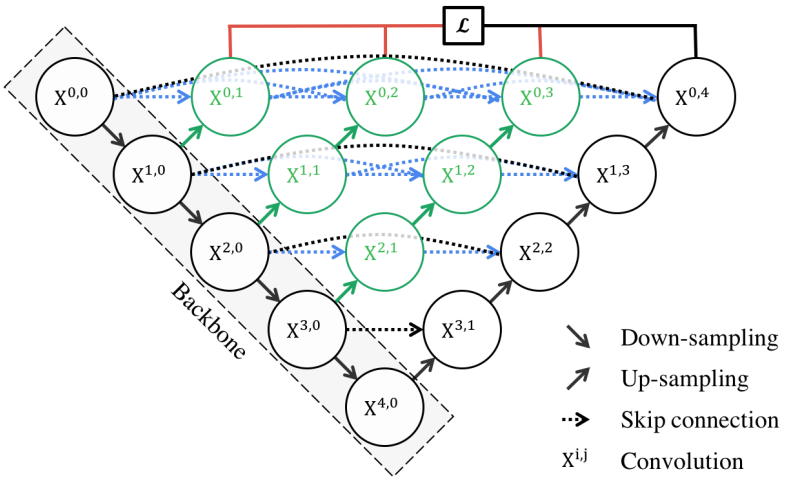

Figure 2.5: U-Net++ interconnected architecture overview[49].

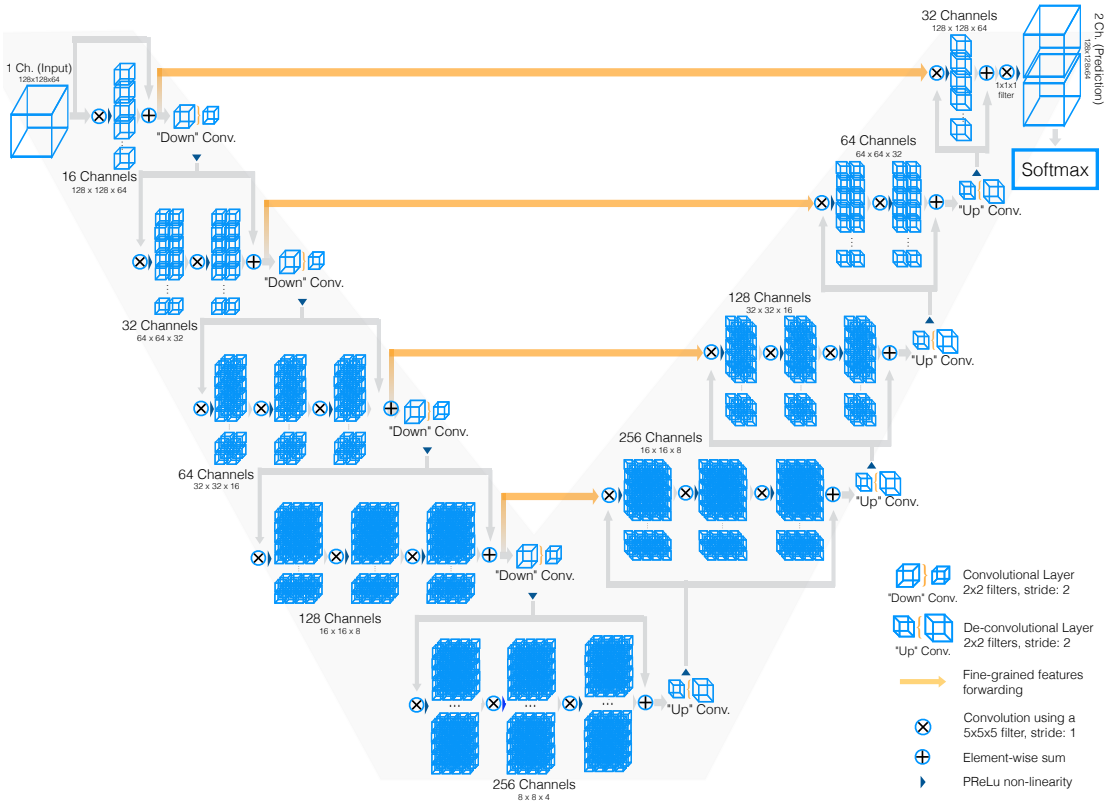

Figure 2.6: Architectural overview of V-Net[31] built upon foundation laid by Ronneberger et al. [37] with UNet.

2.3.3 V-Net Architecture

In 2016 Milletari et al. proposed a fully convolutional neural network for volumetric medical image segmentation called V-Net [31]. V-Net addressed volumetric segmentation because of its already huge interest in 2D segmentation tasks. The v-Net design was inspired by U-Net architecture. The network was trained on MRI prostate scans. All volumes had fixed size of $128 \times 128 \times 64$ voxels and spatial resolution of $1 \times 1 \times 1.5$ mm.

The model consisted of two parts, a left part was divided into different stages, and each stage contained one to three convolutional layers; see Fig. 2.6. Each stage also learned residual function. They used kernel sizes of $5 \times 5 \times 5$ voxels and reduced resolution through $2 \times 2 \times 2$ voxels wide convolution with stride 2. This resulted in halved resolution. An activation function called PReLU is used throughout the network. The right part of the network used transposed convolution operation in order to increase the size of the inputs, as in the left part each input was followed by one to three convolutional layers. The initial input of a stage also received halved feature maps from previous layers; see orange lines in figure Fig. 2.6. Similar to the left also residual functions were learned in the convolutional stages. Soft-max layer was used to output the probability of each voxel belonging to either foreground or background class. They introduced a new loss function based on the Dice similarity coefficient.

2.4 Recent Work: State Of The Art

Publication	Modality	Architecture	Loss Function	Evaluation Metrics
Müller et al.[33]	CTs, 3D	3D-Unet	Categorical CE	DSC, Spec., Sen.
Shan et al. [41]	CTs, 3D	VB-Net	-	DSC
Fan et al. [14]	CTs, 2D	Inf-Net	Custom	DSC, Sen., Spec., Prec.
Yan et al. [47]	CTs, 3D	COVID-SegNet	Dice Loss + CE	DSC, Sen., Prec.
Jin et al. [18]	CTs, 3D	U-Net++	Dice Loss	AUC, Spec., Sen.

Table 2.1: Overview of important aspects of other publications. DSC denoting Dice Similarity Coefficient, Spec. denoting Specificity, Sen. denoting Sensitivity, Prec. denoting Precision, AUC denoting AUC over ROC. and CE denoting Cross-Entropy.

Last year the number of publications has risen up due emergency of the global pandemic COVID-19. We analyzed state-of-the-work in order to get useful insights from these recent publications; see Table. 2.1.

All recent publications built upon well-known architectures, activation functions, dataset augmentation, and evaluation metrics. The most prevalent architecture was U-Net due to its simple design, customization, and solid performance across a variety of tasks. The U-Net architecture was also used as a baseline in the grand challenge [35]. U-Net inspired modifications UNet++ and V-Net. V-Net adjusted U-Net architecture for 3D volumetric datasets where UNet++ modified architecture to gain extra accuracy. The popular activation function is rectified linear unit (ReLU). Next were evaluation metrics where dice similarity coefficient (DSC), sensitivity, and specificity were primarily used. These metrics are commonly used in the medical field.

2.4.1 3D U-Net Architecture with augmentation pipeline[33]

Müller et al. proposed standard 3D U-Net based solution with emphasis on data augmentation pipeline [33]. They trained on a publicly available dataset [20] con-

sisting of 20 CT COVID-19 CT volumes with annotated infection masks. Half of the dataset had a resolution of 512×512 (Coronacases Initiative) and the other half had 630×630 (Radiopaedia) with a number of slices by mean of 176. They used 5-fold cross-validation on the dataset. There was no test, train, or validation splitting due to the limited size of the dataset and no hyperparameter tuning after validation/testing results. They choose standard 3D U-Net in order to minimize the space of hyperparameters and focus on extensive data augmentation. In preprocessing phase they clipped values of Hounsfield units (HU) to interval -1250 as minimum and +250 which transformed all values lower than -1250 to -1250 and bigger than +250 to 250 respectively. This clipping was possible only on half of the dataset from Coronacases Initiative as the other half was already normalized to grayscale range 0 to 255. They state that varying signal intensity drastically influences the fitting process and model performance. In order to reduce the size, they resampled all CT volumes to a target spacing of $1.58 \times 1.58 \times 2.70$ resulting in a median volume shape of $267 \times 254 \times 104$ and a huge positive impact on model performance. Authors pinpoint the batch generators package as an API for state-of-the-art data augmentation in the medical domain. They implemented spatial augmentation by mirroring, elastic deformations, rotations, and scaling; color augmentation by brightness, contrast, and gamma alternations. And the last type of augmentation was noise augmentation by adding Gaussian noise.

All data augmentation was performed on-the-fly on each image right before being inputted into the model. To decrease the probability of generating the same augmented image by applying the same augmentations each augmentation had a 15% probability to be applied to each image. To decrease the risk of overfitting they decided to slice the volume into smaller cuboid patches and forward randomly only a single cropped patch from the image to the fitting process. Volumes were sliced into patches of shape $160 \times 160 \times 80$. For inference, they introduced overlap between

patches of size $480 \times 80 \times 40$ to increase prediction performance. After inference, all patches were assembled into the original volume shape whilst overlapping regions were averaged. The model architecture had kernel sizes of $3 \times 3 \times 3$ with stride of $1 \times 1 \times 1$, exception was up-sampling and downsampling (achieved via transposed convolution) convolutions with kernel size of $2 \times 2 \times 2$ with stride of $2 \times 2 \times 2$; see Fig. 2.7. The architecture had 32 feature maps at the highest resolution and 512 at its lowest. To fight strong bias in class distribution background - lungs - infection as 89% - 9% - 1% respectively they used the sum of the Tversky index and categorical cross-entropy as loss function. Adam optimization was used with initial weight decay of 1e-3 with a dynamic learning rate. The maximum number of epochs was 1000, but a single epoch was defined as an iteration over 150 training batches. Three standard evaluation metrics were used: dice similarity coefficient, sensitivity, and specificity. Results of infection segmentation were as follows DSC: 0.761, sensitivity: 0.730, specificity: 0.999. The authors arrived at the conclusion that the model can be trained without overfitting on small datasets without novel complex network architecture and achieve state-of-the-art performance.

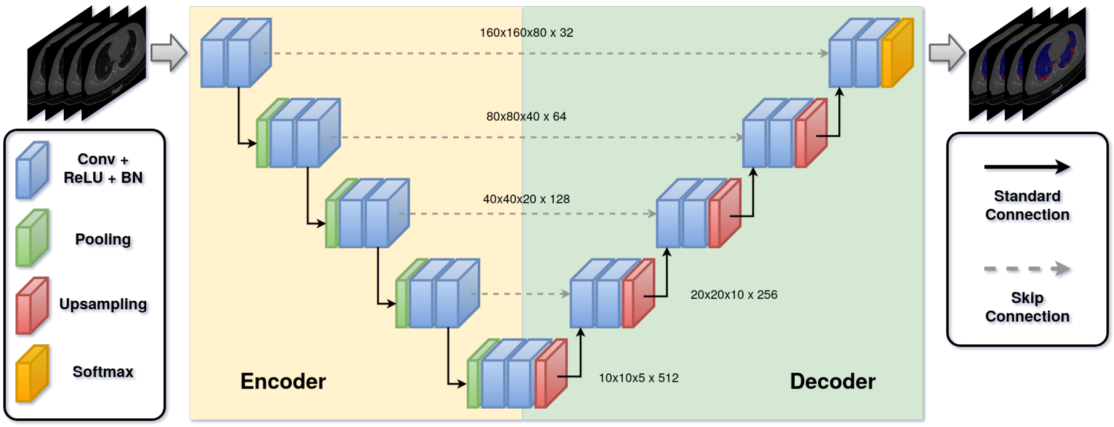

Figure 2.7: Architectural overview of standard 3D U-Net architecture proposed by Müller et al. with emphasis on data augmentation[33].

2.4.2 Improved V-Net: VB-Net Architecture[41]

Shan et al. [41] proposed VB-Net a deep learning segmentation model processing entire CT scans based on V-Net[18]. The model also estimated shapes and percentages of infection referred to as POI. The training set consists of 249 CT scans and the test set was 300 CT scans of COVID-19 patients. Each CT scan was from a different patient and data wasn't augmented before training. There is a bottle-neck structure in place designed as a stacked structure of 3 layers; see Fig. 2.8. The first layer uses $1 \times 1 \times 1$, the second use $3 \times 3 \times 3$ and the last $1 \times 1 \times 1$ convolution kernels, where the first layer with $1 \times 1 \times 1$ kernel reduces the number of channels and feeds the data for a regular $3 \times 3 \times 3$ kernel layer processing, and then the channels of feature maps are restored by another $1 \times 1 \times 1$ kernel layer. VB-Net is more capable of dealing with large 3D volumetric data than traditional V-Net due to reducing and combining feature map channels.

Evaluation metrics included dice similarity coefficients and the Pearson correlation coefficient was used to measure linear dependence between POI and pneumonia severity index (PSI).

They concluded that segmentation of infections through CT scans of patients yielded high accuracy. The model achieved a 91.6% mean Dice similarity coefficient on 300 testing samples.

2.4.3 Using an Attention Modules: Inf-Net Architecture[14]

Fan et al. proposed in their paper Inf-Net a deep neural network for segmentation of CT slices [14]. They introduced also a semi-supervised version of Inf-Net trained on unlabeled CT slices, but we are interested only in the former model. Inf-Net used parallel partial decoder (PPD) and leveraged recurrent reverse attention modules; see Fig. 2.9.

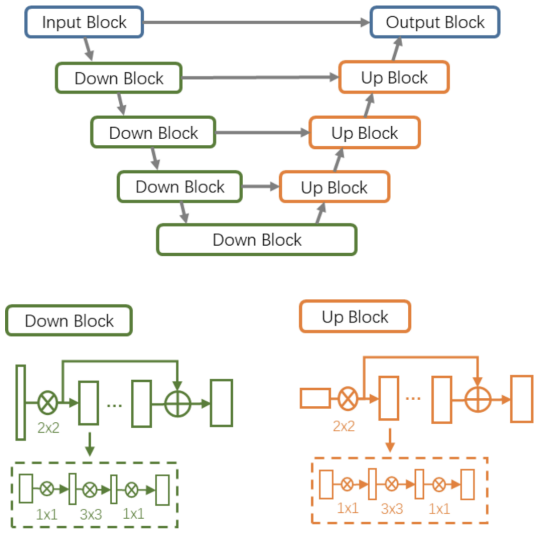

Figure 2.8: VB-Net architecture overview proposed by Shan et al. [41]. The Down and Up Blocks are modified version of V-Net blocks which they described as faster due to $1 \times 1 \times 1$ kernel size convolution layers that reduces number of channels.

The model was trained on 100 axial CT slices from 19 patients with annotations of COVID infected regions. There was no additional dataset augmentation before the training phase. The network used 3 important modules an Edge Attention Module, Parallel Partial Decoder, and Reverse Attention Module. Edge Attention Module learns edge-attention representation. Edges can provide useful information for feature extraction [48]. This module produces an edge map that can be compared to the ground truth mask. These terms were then used in the binary cross-entropy loss function. The idea behind Parallel Partial Decoder (PPD) is an aggregation of only high-level features because as Wu et al. [46] shown that low-level features demand computational power due to high resolution and contribute less to model performance. We want to emphasize that this is amplified in volumetric data segmentation where data has usually a larger spatial size due to more dimensional space. In the end output of PPD serves as guidance for RA modules. The last module Reverse Attention is inspired by clinical practice in-

fection segmentation done by doctors via two steps. In the first step, a rough location of infection is found and in the second step, the accurate procedure is done in order to segment the infection by closer tissue inspection. A PPD acts in this manner as a rough locator. Next, they defined custom loss function which is a combination of weighted IoU loss, weighted binary cross-entropy loss, edge map versus ground truth binary cross-entropy with the term from up-sampled side outputs. Dice similarity coefficient, specificity, sensitivity, and precision were used as evaluation metrics during the benchmark. The model was compared to U-Net, Attention-UNet, Gated-UNet, Dense-UNet, and U-Net++ and outperformed most of them.

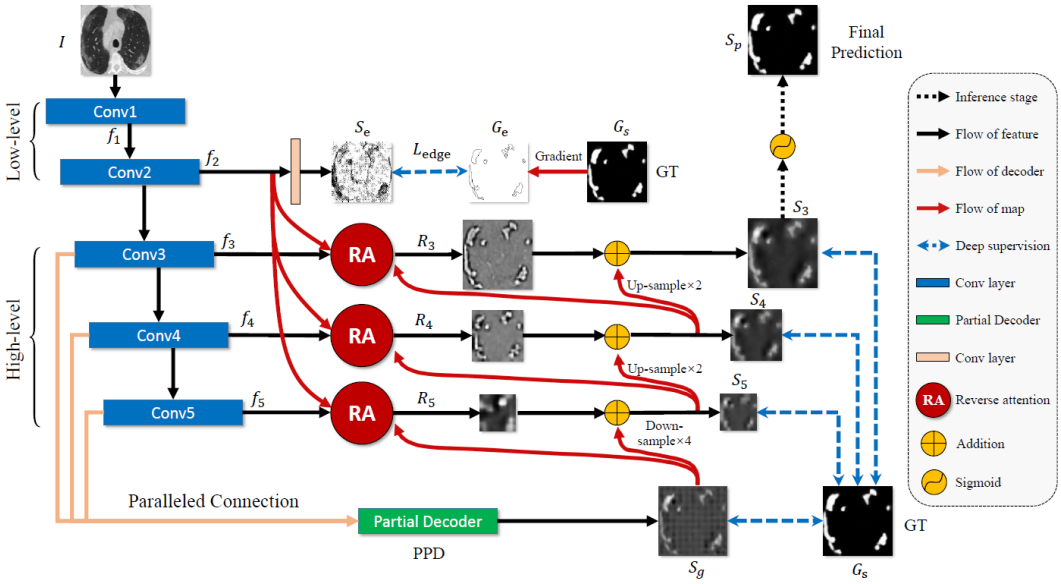

Figure 2.9: Architectural overview of Inf-Net [14]. Lower level features from the top and also higher levels from the bottom are fed into higher-level features where Reverse attention (RA) modules reside. The Partial parallel decoder (PPD) guides the RA model to produce an accurate prediction on a given level of features. These outputs are up-sampled from the higher level into lower levels and to the sigmoid layer output.

2.4.4 COVID-SegNet Architecture[47]

Yan et al. proposed COVID-SegNet a deep convolutional neural network with feature variation block (FV) and progressive atrous spatial pyramid pooling (PASPP) block [47]. The model handles the segmentation of COVID infection as well as lung segmentation. The dataset contained chest CT images of 861 patients with COVID-19. The training set was created as a random selection of 731 CT images and the other 130 were used as a testing set. A medium sharp reconstruction algorithm with a thickness of 0.625 - 10 mm was used in order to reconstruct CT images. No additional augmentation of CT images was done. The size of the minibatch was 2. Encoder-decoder network was employed with 4 encoding layers and 3 decoding layers. Four building blocks were used: FV block (Fig. 2.11),convolutional block, PASPP block and softmax function. These two parts were connected through the PASSP block with different dilate rates as can be seen in Fig. 2.10. All convolution layers used kernels of sizes $3 \times 3 \times 3$ each followed by batch normalization and ReLU activation function. The number of channels is doubled at each layer during encoding from 64 up to 512 and halved from 512 down to 64 during decoding. The loss function was the combination of cross-entropy loss and dice loss. They highlighted feature capturing through the FV block.

FV block consists of 3 branches a contrast enhancement branch, a position-sensitive branch, and an identity branch. The motivation behind the PASSP block was fact that infected regions have different sizes. This requires features to be a different size as well. Dice similarity coefficient, sensitivity, and precision were used to evaluate the performance of the model. Authors conducted experiments and benchmarks with other state-of-the-art models and results have shown that their model outperformed all models in both tasks, lung segmentation, and infection segmentation. Other models were FCN, UNet, UNet++, and V-Net.

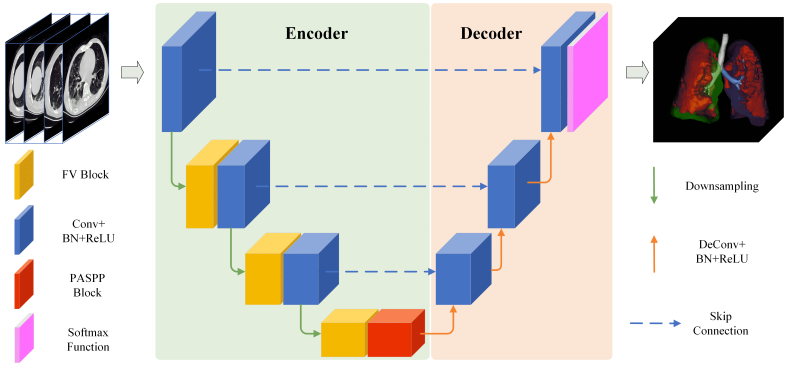

Figure 2.10: Architectural overview of COVID-SegNet proposed by Yan et al. [47]. The architecture consists of two parts an encoder and a decoder.

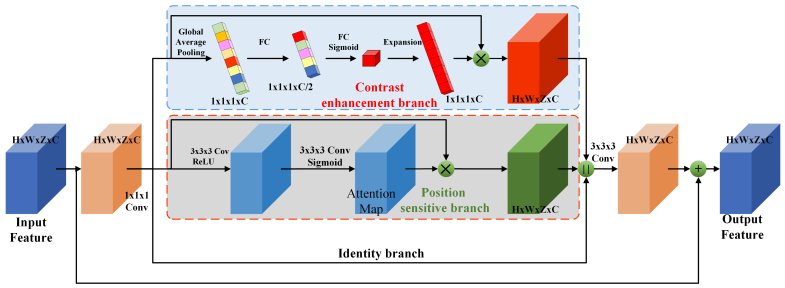

Figure 2.11: Details of FV Block used in COVID-SegNet proposed by Yan et al. [47]. Idea of FV Block is to enhance the contrast and adjust the intensity in feature level automatically for different images employing also channel attention.

2.4.5 Improved U-Net: the UNet++ Architecture[18]

Jin et al. proposed model performing classification and segmentation tasks [18]. The model was trained on 1136 training cases (723 COVID-19 positives). Achieved sensitivity of 0.974 and specificity of 0.922 on the test dataset consisting of also pulmonary diseases. This model has been deployed in 16 hospitals performing 1300 screenings per day. The dataset covers a variety of different cases such as samples gathered from different CT equipment, some samples also contained other diseases. CT images were normalized to (1,1, 2.5) mm using standard interpolation algorithms. Window width and window level were adjusted for each model.

Dataset augmentation was present, typical augmentation operations were used such as random flip, zooming, and panning to improve variability. All values were normalized to [0, 1]. All segmentation was done on the (256, 256, 128) patch which was also the input image size of the model. Preprocessing step also included lung segmentation on which model did classification and segmentation.

The model was divided into 2 stages: 3D segmentation and classification. The model architecture was chosen by empirical tests of different architectures such as FCN-8s, U-Net, V-Net, and 3D-Unet++ for segmentation tasks and for classification task models like DPN-92, Inception-v3, ResNet-50, and Attention ResNet-50. The 3D-Unet++ model achieved the highest Dice coefficient. Attention ResNet-50 model was chosen as the best model for classification task achieving area under the curve (AUC) of 0.991. The Dice coefficient was used to evaluate the performance in the segmentation task and AUC was used to evaluate performance in the classification task. Specificity and sensitivity were also measured.

2.4.6 An Ensable Using the nnUNet Architecture[17]

Isensee et al. proposed UNet based pipeline with automatically adjusted parameters from training data [17]. There are 3 types of parameters: fixed parameters, rule-based parameters, and empirical parameters. Fixed parameters are picked before and did not change according to input data. Fixed parameters are learning rate, loss function, architecture template, optimizer, data augmentation, training procedure, and inference procedure. They used dice loss and cross-entropy function, SGD optimizer with Nesterov momentum set to 0.99. Rule-based parameters are calculated based on input data in form of heuristics based on the expert knowledge of the authors. These parameters included batch size, patch size, image resampling strategy, and such. The last type of parameters were em-

pirical ones where the right model or combination of two was picked according to cross-validation performance. By default, nnUNet generates three different U-Net configurations, 2D-Unet, 3D-UNet, and cascade 3D-UNet. The cascade 3D-UNet is the combination of 2 3D-UNets operating in two stages. In the first stage, the standard encoding-decoding takes place and in the second stage, the outputs from the first stage are concatenated as one-hot encoding to the full resolution and further refined by the second 3D-UNet. They showed that this strategy is suited for various medical imaging tasks and outperformed specialized pipelines.

2.4.7 Challenge Submissions: Top Performers

Roth et al. summarised the competition results of top 10 participants[38]. Participants used fully-automated methods, which all were based on some modified version of U-Net[37, 51], a fully convolutional network [27] with skip connections[27, 13]. Most of the participants used the particular U-Net variant the nnU-Net[17], which was already used in other challenges. Popular segmentation function used was Dice Loss[31] along with additional cross entropy, top-k[28] and focal loss[26]. The winning segmentation model utilized model ensembling, where predictions are fused from independently trained models. Majority of methods used 5-fold cross validation and model ensemble for creating final prediction.

2.4.7.1 Rank 1: "Semi-supervised Method for COVID-19 Lung CT Lesion Segmentation"

The team of Shishuai et al. used an additional unlabeled dataset for improvement in model generalization. They sued nnU-Net architecture as a basis for segmentation network and trained it with labeled data first. After training, they generated the pseudo lesion masks of annotated and not annotated infected CT images. Lastly, the model was trained as fully supervised with the original dataset and generated

pseudo labels.

2.4.7.2 Rank 2: “nnU-Net for Covid Segmentation”

The team of Fabian Isensee et al. also used nnU-Net and used it to implemented five 3D U-Net configurations. Starting with a low-resolution residual U-Net with extensive data augmentation and batch normalization. The next four configurations were all high-resolution U-Nets with either residual blocks or extensive data augmentation. They trained a total of 10 models for each configuration and created an ensemble with softmax averaging for predictions.

2.4.7.3 Rank 3: "Automated Ensemble Modeling for COVID-19 CT Lesion Segmentation"

Claire Tang developed U-Net based automated pipeline with data processing and with various loss functions. During data preprocessing both 2D and 3D were created. For 3D images, a downsampling technique was used to produce also low-resolution images. For each dataset, U-Net models were trained. The author pointed out a 3D cascade U-Net which was firstly trained on low-resolution 3D images and then it used its prediction to further train a full-resolution U-Net.

Rank	Value	ID #	Fully Automated	Extra Data	Pretrained	Ensemble	Data Dim.	Network Arch.	Authors
1	2.6	53	✓	✓	✗	✗	3D	nnU-Net	S. Hu et al.
2	6.0	38	✓	✗	✗	✓	3D	nnU-Net	F. Isensee et al.
3	7.7	65	✓	✗	✗	✓	2D/3D	nnU-Net	C. Tang
4	8.4	58	✓	✗	✗	✓	3D	nnU-Net	Q. Yu et al.
5	8.5	31	✓	✗	✗	✓	3D	nnU-Net	J. Sölter et al.
6	9.2	50	✓	✗	✗	✓	2D/3D	nnU-Net	T. Zheng & L. Zhang
6	9.2	68	✓	✗	✓	✗	2D/3D	VGG16 Hybrid MONAI	V. Liauchuk et al.
8	9.4	95	✓	✗	✗	✓	3D	nnU-Net	Z. Zhou et al.
9	10.6	29	✓	✗	✗	✗	3D	nnU-Net	J. Moltz et al.
10	11.3	15	✓	✗	✗	✗	3D	U-Net	B. Oliveira et al.

Table 2.2: Summary of top-10 participants in the COVID-19 segmentation grand challenge from [38].

Chapter 3

Proposed solution

Based on our analysis we chose UNet architecture. It has been widely used and has provided a solid performance on medical imaging datasets. The ability to modify and experiment with the bare minimum was our requirement to explore possible modifications to adjust architecture to our needs. We focused on loss function modifications and dataset augmentation therefore we wanted to minimize the number of parameters involved in experiments.

3.1 Research Objectives

We aimed to solve one of the Grand Challenge competitions about volumetric segmentation of COVID-19 disease inside the lungs. To solve this challenge we set up 2 research objectives. Dataset preprocessing and loss function tuning.

Objective 1: Spatial Reduction Due to large images with unnecessary data we wanted to compare the performance of our proposed architectures on the cropped dataset.

Objective 2: Loss Function Modification Due to the large class imbalance which affects the performance of the loss function we wanted to propose an adjusted loss function for this problem.

3.2 Solution Overview

To achieve both objectives we have set, we created a two-step process. In the first step, we trained a 2D UNet model for lung detection. We used external dataset[29] for training 2D U-Net. Next, the whole dataset was cropped by a bounding box that surrounded the lungs by an axial axis. In the second step, we trained the 3D UNet model for COVID-19 lesions segmentation on the cropped dataset. Cropped dataset improved training speed and allowed quicker experiments and iterations. This high-level overview can be seen on Fig. 3.1. Methods datasets are further described in further particular sections.

3.3 COVID-19 Dataset - Segmentation Challenge

We worked with the provided dataset of CT volumes of COVID-19 patients from [35] and participated in the Grand Challenge segmentation task. Dataset consisted

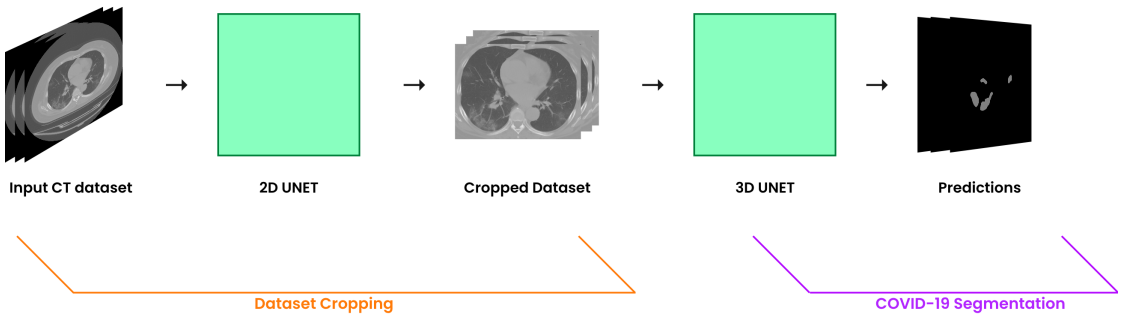

Figure 3.1: Overview of our solution that is divided into two steps. Firstly, cropping the CT image dataset. Secondly, performing COVID-19 segmentation.

of 199 unenhanced chests CTs. Confirmation of the presence of SARS-CoV-2 was tested with Reverse Transcription Polymerase Chain Reaction (RT-PCR). Dataset included ground truth annotations of COVID-19 lesions in the lung. Each dataset file was in NIFTI file format compressed with gzip. The test set wasn't publicly available at the moment of writing. CT images have variable depth sizes, but width and height (512×512) are the same across all of them. When we looked at all CT scans and a calculated ratio of ground truth masks 0 and 1 pixels. There was 1 positive class (have a disease) to 242 negative class (without disease). A positive class is rare.

3.4 Dataset Cropping

When we looked closer at the dataset we identified that almost half of the volumetric data are irrelevant such as background or other non-lungs regions. Another observation was that nodule annotations could be sparse and scattered across the lungs; see Fig. 3.2. In later sections we targeted these smaller regions with loss function modification to increase performance.

3.4.1 2D Lungs Dataset for Lungs Segmentation

To train our spatial reducing UNet model we used publicly available data hosted on the Kaggle platform published by K. Mader [29]. As this dataset contained exactly what we needed to segment lungs from the slice of a CT image. This dataset contains 267 CT slice as shown examples in Fig. 3.3 and Fig. 3.4. We set aside 15 slices for the test and the rest split 80% for the training set and the rest 20% for the validation set. We did not apply any data augmentations to the dataset.

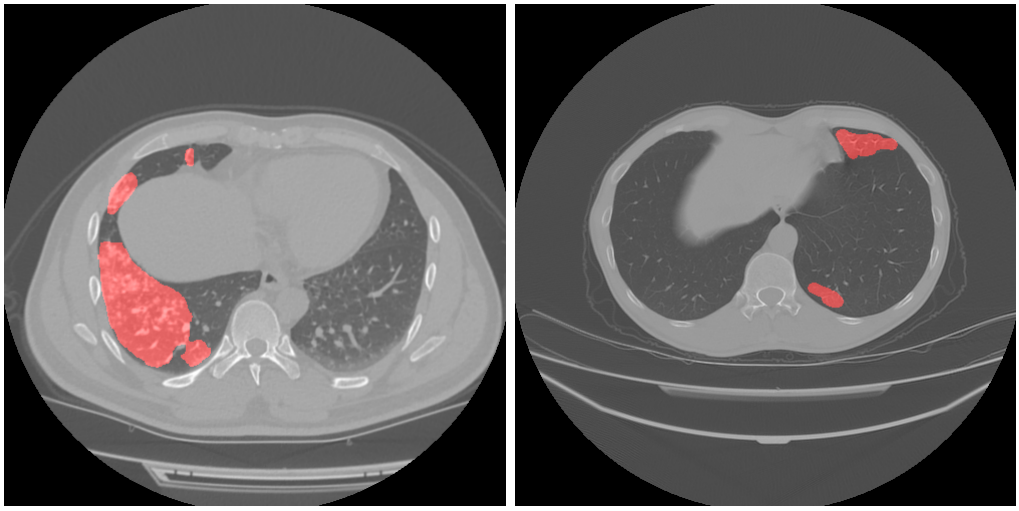

Figure 3.2: Example of single slices of lung nodules of 0003 and 0072 patients. Red area signalizes the ground truth annotation. We can see there can be variance between annotations resulting in smaller or bigger regions.

3.4.2 Method: using U-Net Model

We approached the problem of volumetric reduction as a lung segmentation problem. We wanted to train the model on publicly available datasets online with ground truth masks. Instead of looking for 3D volumetric lung segmentation, we looked after the 2D dataset to segment lungs per CT slice.

Architecture: 2D UNet We implemented standard 2D UNet architecture with a number of filters (32, 64, 128, 256, 512). We used the ReLU activation function in all layers except the output layer. As an output layer, we have chosen the sigmoid layer. Adam optimizer with a learning rate set to 0.0001 with weight decay set to 0.00001. We trained with a batch size of 4 and up to 40 epochs.

Volume reduction: Slicing lungs out of CT volume We selected 3 middle slices from the original volume, the middle slice was referred to as the m slice and additional slices were the $m-10$ and $m+10$ slice. Before segmentation, these slices

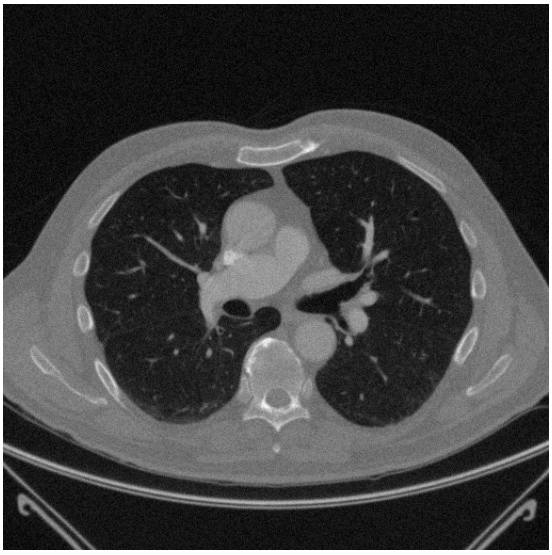

Figure 3.3: CT slice example from lungs CT dataset.

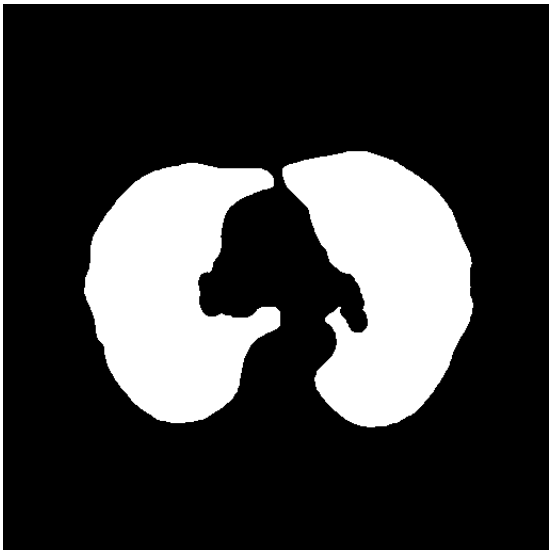

Figure 3.4: Ground truth mask example from lungs CT dataset.

were rotated by 270 deg due to the lung segmentation model where dataset images had this exact position. These axial slices were fed into the segmentation model and the resulting masks were cleaned. The cleaning method consisted of finding contours smaller than 6000 pixels squared and leaving them out. This proves as a good heuristic for cleaning resulted masks. In the next stage, the mask was added together with the XOR operation forming one single *XORMask*. Next, we found a bounding box of contents of XORMask which were used to slice the original CT image. When we were slicing the original CT image only the depth dimension was preserved. Both width and height were cropped to match the bounding box of the XORMask. Ground Truth masks were cropped as well to match the XORMask bounding box. Both CT image and ground truth mask were rotated back to their default rotation.

After reducing all CT images the dataset size was reduced by 58.9%. Output CT images weren't perfectly cropped as some unnecessary background voxels were still present but all essential data were present after volume reduction.

3.4.3 Training 2D U-Net

We trained for 40 epochs, we achieved peak dice mean of 0.9168 at 36 epochs and we used this model for lungs segmentation; see Fig. A.4. When we cropped volume we had problems with cropped sides to solve this issue we added padding to the bounding box before cropping. All 70 CT volumes we cropped were examined by selecting middle and checking if important segment was not cropped; see examples on Fig. 3.5 and Fig. 3.6.

Figure 3.5: Slice of original CT example before spatial reduction.

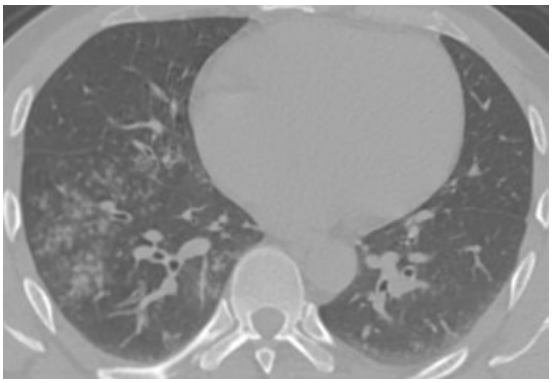

Figure 3.6: Slice of cropped CT example after spatial reduction.

3.5 Using 3D Lesion Segmentation

To solve the Grand Challenge segmentation task[35] we focused firstly on creating a baseline model to further improve upon. We built upon the baseline model created by MONAI Consortium the team behind MONAI framework[32] built on well known Pytorch framework. This baseline implementation provided all fundamental blocks such as handling dataset loading and data augmentation. We

named it CovidSeg.

3.5.1 Dataset Augmentation of COVID19 Dataset

We have already introduced the dataset in Sec. 3.3. Intensity values were clipped to minimal value of -1000 and maximum 500 as the authors of dataset suggested to do so; see Fig. 3.7 and Fig. 3.8.

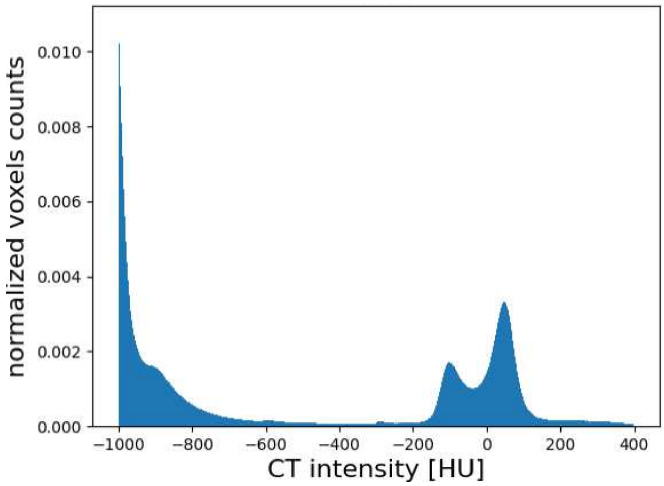

Figure 3.7: Normalized histogram of intensities [38].

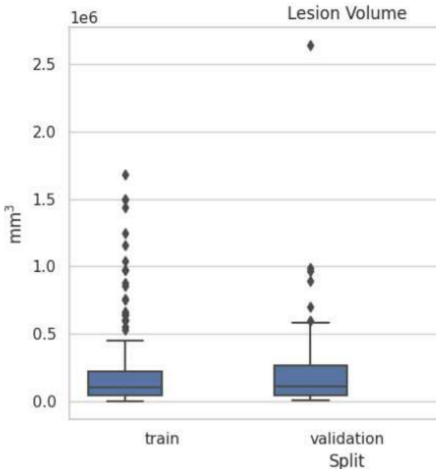

Figure 3.8: Volume size of COVID-19 lesions [38].

Dataset samples augmentations were re-scaled from original value range into $[0, 1]$ range. Next during training, we also applied spatial padding to width and height dimensions omitting depth dimensions to ensure size of at least 192×192. Next, we applied random affine transform to slightly alter the scale by 0.1 and rotation by $0.05rad$ both of width and height dimension axes with a probability of 15%. As the next step, we picked random 3 windows of size $192 \times 192 \times 16$ were picked as training examples. This allowed us to work with arbitrary input data which we leveraged in reduced dataset samples. After that, we applied Gaussian noise with $std = 0.01$ with a probability of 15% and flipped each axis with a probability of 50%.

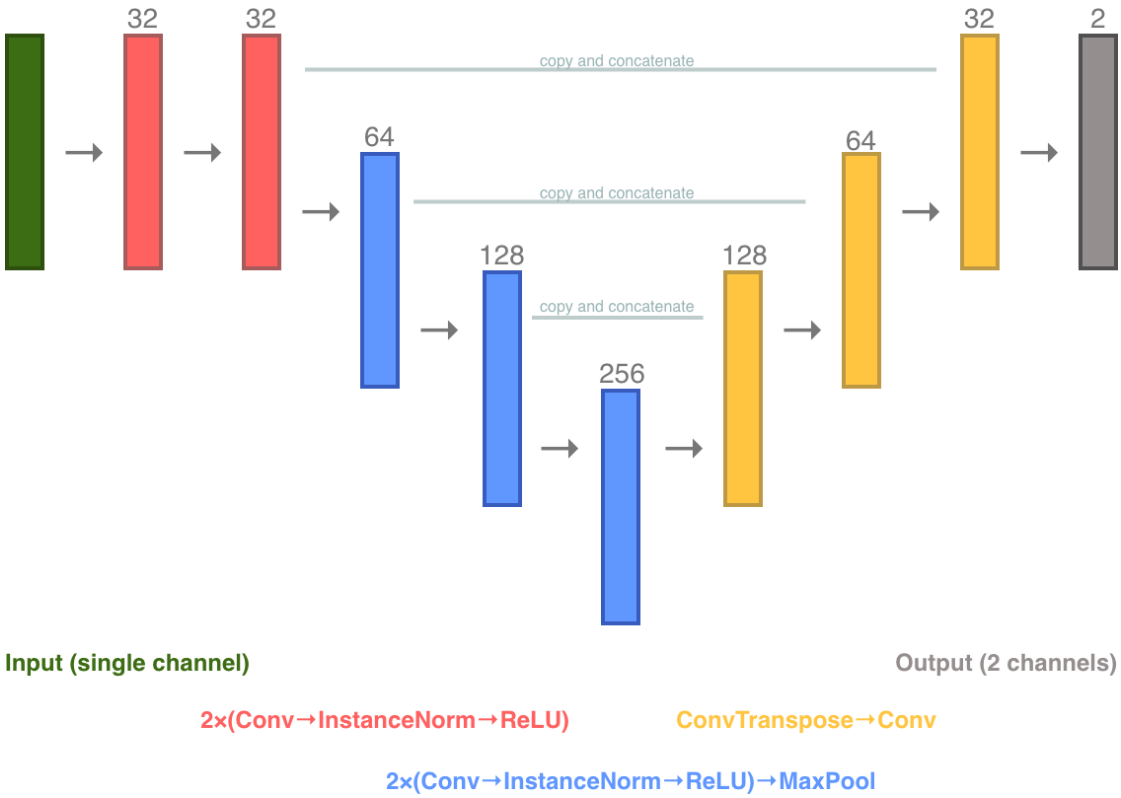

Figure 3.9: Architecture overview of our CovidSeg model we used from MONAI[32] framework. Numbers above rectangles denoting number of output channels.

3.5.2 Method: using Basic 3D Unet

For network architecture we have chosen standard 3D UNet architecture provided from MONAI[32]; see Fig.3.9. With the number of channels starting from 32, 32, 64, 128, 256, 32 where 256 is the bridge layer between contracting and expanding path. We also used dropout with the value of 0.1. Each layer used $3 \times 3 \times 3$ kernel size and padding set to 1. As an activation function, we started with ReLU in each layer. We used sliding window inference with the size of $192 \times 192 \times 16$ with an overlap of 50% and replicating padding mode. This inference method took 2 slices at a time.

In every run, we used the Adam optimizer function. The number of epochs was

100 and the batch size was set to 2. Before calculating loss function on input was applied softmax function and one-hot encoded. The background channel was omitted and the loss and also other metrics were calculated only on the positive class of the second channel. We omitted the background channel during metrics calculation as the background was majority class and a huge class imbalance was present.

First experiment: Training with original dataset We started with the original dataset to create baseline results. What we wanted to achieve with the first experiments are identifying problems during segmentation and the impact of larger CT images with more irrelevant data within them.

In this run, we tried the sum of Dice loss and Cross Binary Entropy as a loss function. We set the learning rate to 0.0001. We used the ReLU activation function in all layers. In this run, we used original CT images. After 100 epochs the training resulted in a Dice metric of 0.435 with a maximum peak of 0.464. With an average of 0.397.

According to the results we further analyzed loss function performance and opportunities to tweak it to increase performance.

3.5.3 Loss Function Modification

When we looked at inference on the test set we observed a problem with false-positive and false-negative results. We looked closely at the Dice coefficient used in Dice + Cross-Entropy Loss function in the first experiment. The Dice coefficient defined as:

$$DSC = \frac{2TP}{2TP + FP + FN}$$

The true positive cases drive the result of DSC higher and on the other hand, false positive and false negative cases aren't that punishing if present. To counterweight this imbalance the Tversky Loss function was introduced[40]. Where parameters α and β are used to weight the punishment for false negative and false positive respectively.

$$Tversky = \frac{TP}{TP + \alpha FN + \beta FP}$$

When we experimented with this function in our settings we got similar results to DSC because of the large class imbalance. During the training, we want to reward models for true positives because of their rarity, but we also wanted to reduce false negative and false positive rates too. We experimented with the idea of adaptive loss function where the importance of false-negative and false-positive vary as in already mentioned Tversky Loss, but we did not reach any conclusion. This may be reexamined in future work.

Another idea that we experimented with is the balancing DSC with Jaccard Index defined as:

$$Jaccard = \frac{TP}{TP + FN + FP}$$

Our loss function was defined as:

$$Loss = 1 - \frac{DSC + Jaccard}{2} + CELoss \tag{3.1}$$

During the training we observed that our loss function performed better on sensitivity metric and worse on mean dice; see Fig. 3.11 and Fig. 3.12. This loss

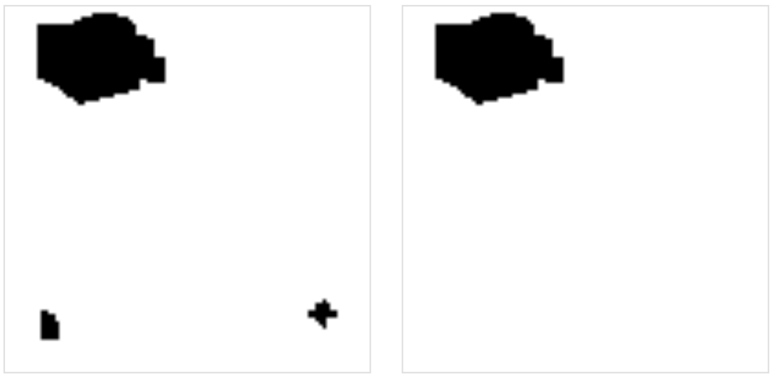

Figure 3.10: On the left side we created a ground truth image, on the right side is prediction. Dice coefficient calculated on these images equals to 0.9964.

function modification improved upon the Dice coefficient problem. When Dice coefficient is used upon segmentation with one larger region and smaller regions that are spread out. The Dice loss function will quickly learn to segment larger regions but will have a harder time segmenting the smaller ones. Our modification was aimed at the problem with smaller regions and punished the Dice coefficient further to improve performance. To see this problem; see Fig. 3.10. As Dice favors true positive results it skews results. What may seem as almost perfect segmentation is in reality lacking.

3.5.3.1 Training With Cropped Dataset

After we created the baseline of our work we cropped our samples of 60 CTs (48 train and 12 validation samples) and have done another run of experiments with the same networks and parameters, but on the cropped dataset. We used the same method of dataset augmentation and processing, network architecture, and loss functions.

Firstly we trained with standard loss function $DSC + CE$. After 100 epochs the training resulted in Mean Dice metric 0.400 with maximum value of 0.413. With

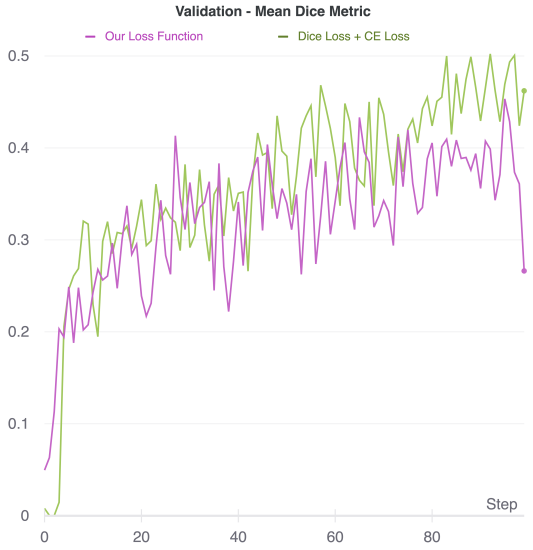

Figure 3.11: Comparison of performance of our loss function (purple) and Dice Loss + Cross Entropy Loss (green) on mean dice metric. Both models were trained on cropped dataset.

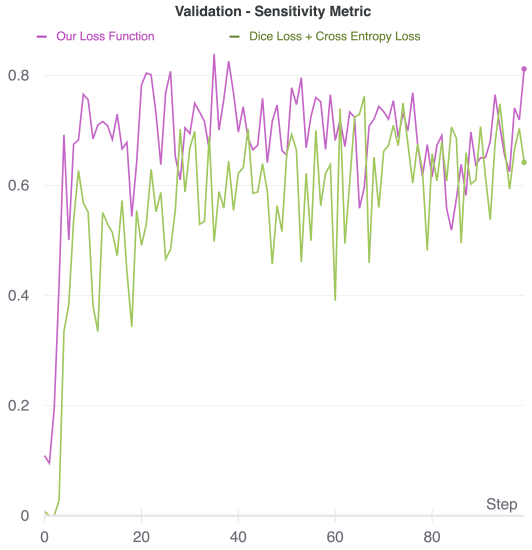

Figure 3.12: Comparison of performance of our loss function (purple) and Dice loss + Cross Entropy Loss (green) on sensitivity metric. Both models were trained on cropped dataset.

an average of 0.392; see Fig. 3.13. The validation sensitivity metric resulted in 0.598 with maximum value of 0.711 and an average of 0.594; see Fig. 3.14. For validation loss; see Fig. A.1.

Secondly we trained with our modified loss function (Eq. 3.1) After 100 epochs the training resulted in Dice metric 0.423 with the maximum value of 0.466. With an average of 0.431; see Fig. 3.13. The validation sensitivity metric resulted in 0.734 with maximum value of 0.786 and an average of 0.694; see Fig. 3.14.

Preliminary Results: Overview We compared all runs together and applied them to smooth in form of a running average of 10 values. Our loss function performed worse on the mean dice metric; see Fig. 3.15. By contrast, the performance on the sensitivity metric was better in all cases than Dice Loss + Cross-Entropy Loss; see Fig. 3.14. The validation loss in case of our loss function is converging

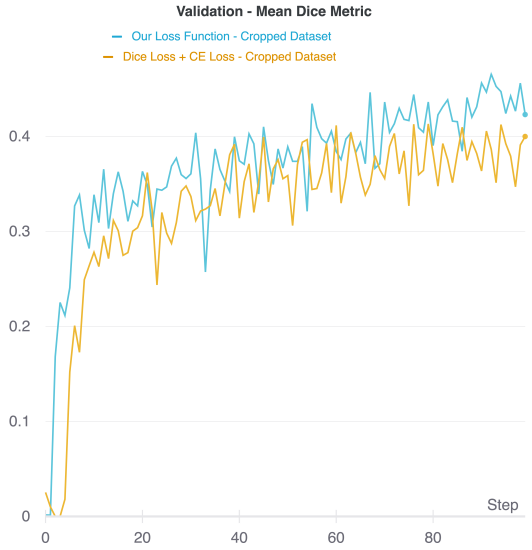

Figure 3.13: Comparison of performance of our loss function (blue) and Dice Loss + Cross Entropy Loss (yellow) on mean dice metric. Both models were trained on cropped dataset.

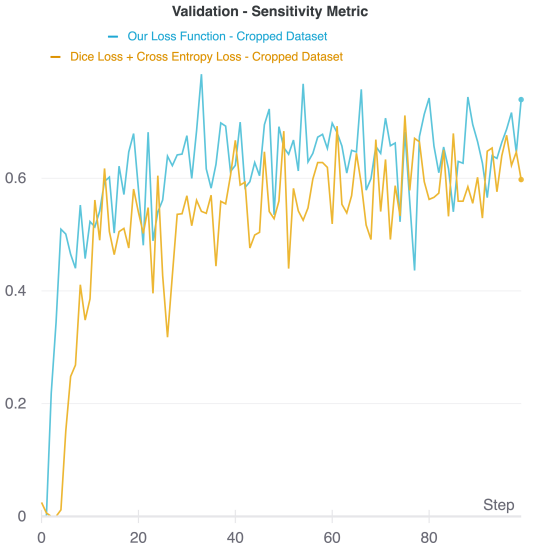

Figure 3.14: Comparison of performance of our loss function (blue) and Dice loss + Cross Entropy Loss (yellow) on sensitivity metric. Both models were trained on cropped dataset.

worse the one explanation could be large class imbalance and the punishment from false positive and false negative cases slows down the training; see Fig. A.2.

3.5.4 Further Addressing the Class Imbalance

The class imbalance problem was explored in the 1990s by Anand et al. [2] where authors showed that the majority class negatively affected the backpropagation algorithm. The majority class had a higher impact on gradient updates leading to reduced error of the majority class but the increasing error of the minority class.

We use the same grouping of methods as authors Johnson and Khoshgoftaar [19] in their survey paper on this topic. They grouped methods as data-level, algorithm-level, and hybrids.

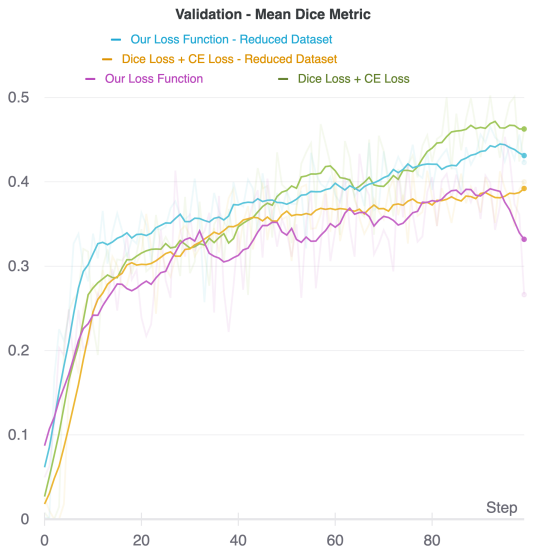

Figure 3.15: Performance comparison between all experiments on mean dice metric smoothed with running average over last 10 values.

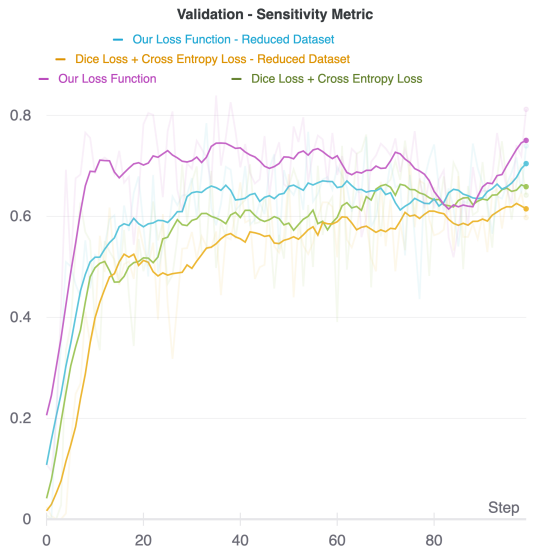

Figure 3.16: Performance comparison between all experiments on sensitivity metric smoothed with running average over last 10 values.

Data-level methods focus on the dataset and try to balance samples of the dataset or change underlying distribution with methods like random over-sampling (ROS) or random under-sampling (RUS) [10, 7, 36, 16]. Where ROS duplicates examples from the minority class and RUS discards random examples from the majority class. The focus was on minimizing the overfitting problem that naturally stems from the highly imbalanced dataset were learning the majority class yields accuracy of the majority class.

Algorithm-level methods focus on adjusting the learning process on the level of loss function or architecture. Such a concept was used by Lin et al. where they proposed focal loss which weights the classification loss based on the prediction [26]. They introduced a loss function that adds a factor $(1 - p_t)^\gamma$ to the standard cross-entropy loss function. Whenever the model incorrectly classifies example the p_t is small and the term is closer to 1; see Eq. 3.2[26]. As the authors pointed out

the γ reduces the loss contribution from easy to classify examples and extends the range in which an example results in a lower loss.

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{3.2}$$

Dang et al. defined class imbalance problem as a cost-sensitive classification problem [44]. To address this problem they used the so-called cost matrix which encoded penalties of classifying samples from one class as another. This cost matrix was used as a part of the loss function where larger the value in the cost matrix meant a larger penalty.

Khan et al. proposed a loss function by estimating the uncertainty of each class [23]. Khan et al. proposed a method of calculating loss function by estimating the uncertainty of each class. Authors based this method on extending classification boundaries further away from a more uncertain class to avoid over-fitting and modeled each sample as multi-variate Gaussian distribution with a mean vector and covariance matrix.

The last method of dealing with class imbalance is the combination of the previous two. We used this strategy as our dataset had a large class imbalance. The original dataset had a ratio of 1 positive voxel to 242 negatives and the cropped dataset we later used during training had a ratio of 1 positive voxel to 76. And we also modified the loss function with terms that penalize the model as the class imbalance is higher.

We experimented with the information on the balance ratio and we decided to create a single number that represented discounting factor γ; see Eq. 3.3. In search of an exponential discounting factor as in Focal loss, we naturally constructed such an equation based on logarithms.

$$\gamma = \log\left(-\log\alpha\right) \tag{3.3}$$

The base of the logarithm can be also taken into account, we used a natural logarithm. The γ for our dataset is 1.4567. We used this parameter to introduce another cost to our existing loss function.

We also created an extension of our first loss function the sum of Jaccard index and Dice Loss and divided by 2 on the same principle. Adding Dice coefficient to the power of γ and averaging this sum Eq. 3.4.

$$L = \frac{DSC + DSC^{\gamma}}{2} + CE \tag{3.4}$$

DiceDiceLoss function further generalizes the principle of using class imbalance ratio to apply a global penalty for incorrectly classified classes. This helped during training compared to standalone Dice Loss.

We did experiments to determine whether this heuristic γ calculation is superior to other numbers and we found out in small series of experiments, that the higher the gamma, the better performance we achieved; see Fig. 3.17. The $\gamma = 2.2$ yielded the best performance and we chose this value for the evaluation process.

3.5.4.1 Balanced Dataset Splits

When we split the dataset into training, validation, and testing set we ensured that all these sets have had a similar distribution of classes within them. We accomplished that with an iterative algorithm that shuffled the dataset until ratios of positive classes to negative between each set were smaller than 0.001.

Figure 3.17: Experiments with loss function modification (Eq. 3.4) with different γ values. Smoothing function was applied.

Adjusted Loss Function: Summary To minimize the impact of high-class imbalance we applied the hybrid method. Firstly we cropped the dataset to contain only lungs. Secondly, we modify the loss function to be more punishing during model training. These results led us to modify DiceDiceLoss function which uses γ parameter to improve the function. We further evaluated DiceDiceLoss function and compared it to baseline DiceFocalLoss.

Chapter 4

Evaluation

To evaluate our models, we used the standard evaluation technique K-Fold cross-validation for model performance estimation. We chose 3 folds for each model configuration and compared the results. Each fold consisted of 120 training samples and 60 test samples. The number of maximum epochs was set to 150 with an early stopping with patience set to 10. When loss on the test set wasn't improving in 10 consequent epochs the particular fold was stopped. The number of folds was determined by hardware and time constraints. The average run-time of a single cross-validation run was 38.36 hours. As for baseline, we run the same cross-validation configuration with DiceFocalLoss function from MONAI[32] framework with default parameters.

4.0.1 Robustness

Our model may accept any CT as we applied data augmentation that ensures size at least 192×192 and preserves depth dimension. We did not implement input data validation as we worked only with a provided dataset that was prepared for training. Before we started our experiments we ensured that the dataset was with-

out flaws that could affect results. We manually examined the datasets in both the COVID segmentation pipeline and dataset cropping pipeline before training.

4.0.2 Evaluation Metrics

We chose Precision(Eq. 4.1), Sensitivity(Eq. 4.2), Dice coefficient(Eq. 4.3) to better understand model performance with a particular loss function. We omitted specificity due to ignoring the majority class (background) during metric calculation.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.2}$$

$$Dice = \frac{2TP}{2TP + FN + FP} \tag{4.3}$$

4.0.3 Quantitative Evalution

After performing a 3-fold cross-validation we gathered evaluation metrics per fold. In the end, averaging every metric across all folds gave us an average model performance on each metric. Training during cross-validation with time was less and less stable, but validation loss steadily decreased. Our model outperformed baseline; see Fig. 4.2.

Our model performed better on dice, precision with an average of 0.5377 and 0.6159; see Tab. 4.2. We used an independent t-test to evaluate these metrics and both resulted in statistically significantly higher with a p-value of 0.001 for dice

Fold	Dice	Sensitivity	Precision	Epochs
1	0.4772	0.8739	0.5390	70
2	0.5002	0.9108	0.5320	91
3	0.4833	0.9090	0.5343	78
mean	0.4869	**0.8979**	0.5351	79.6667

Table 4.1: Summary table of 3-fold cross validation of DiceFocalLoss configuration.

Fold	Dice	Sensitivity	Precision	Epochs
1	0.5336	0.7796	0.6212	102
2	0.5363	0.8202	0.6091	92
3	0.5432	0.7843	0.6175	92
mean	**0.5377**	0.7947	**0.6159**	95.3333

Table 4.2: Summary table of 3-fold cross validation of DiceDiceLoss configuration.

metric and 2.0e−5 for precision. Due to a low number of folds the power of these tests is low. Baseline was better in sensitivity with an average values of 0.8979; see Tab. 4.1.

4.0.4 Test Set Evaluation

To evaluate our prototype on the test set we picked the best performing model from each cross-validation run. After that, we run an evaluator with the test set as input for each model. Finally, we averaged these values which gave us an average performance as each model from cross-validation was trained on a different part of the dataset. Our test set contained 19 CT images that our model have never seen before. We put the results of our test set evaluation compared to baseline and other participants in the challenge in Tab. 4.3.

We also put our results into relation with the top 3 competitors from COVID-19 Lung CT Lesion Segmentation Challenge[38]. This comparison was just for demonstration purposes as we did not get access to the same datasets. The training datasets were similar, but the testing dataset used in Tab. 4.4 were different. They

Dice	mean	std	median
DiceDice	**0.5752**	0.0182	0.5839
DiceFocal	0.5032	0.0059	0.5039
Precision			
DiceDice	**0.6002**	0.0676	0.5715
DiceFocal	0.5161	0.0303	0.5082
Sensitivity			
DiceDice	**0.5409**	0.034	0.5589
DiceFocal	0.4735	0.0191	0.4696

Table 4.3: We picked the best model from each cross validation run and we made evaluation run for each model and averaged results.

Model / Competitor	mean	std	median
DiceDice	0.5490	0.1443	0.5321
DiceFocal	0.5167	0.1492	0.4825
Rank 1. Team (see Sec. 2.4.7.1)	0.5980	0.2640	0.7000
Rank 2. Team (see Sec. 2.4.7.2)	0.5930	0.2580	0.6770
Rank 3. Team (see Sec. 2.4.7.3)	0.5590	0.2910	0.6860

Table 4.4: Dice coefficient comparison table with our results and results from challenge[38]. These results compared relative performance as conditions during evaluation and training weren't same as ours.

tested on 46 CT images and we did on 19. These images weren't the same. We showed these results to get a general idea of where performance was on the Dice metric. Therefore we couldn't draw any conclusions from this comparison.

We statistically tested with independent t-test equality of means of DiceDice and DiceFocal results; see Tab. 4.4. With an alternative hypothesis of DiceDice mean as greater. We rejected the null hypothesis with a resulting p-value of 0.003.

4.0.5 Qualitative Evaluation

We picked slices from the test set and depicted differences between models. As we run the inference loop for each example we also calculated the Dice coefficient for each example. According to these results, we picked the worst, and average the

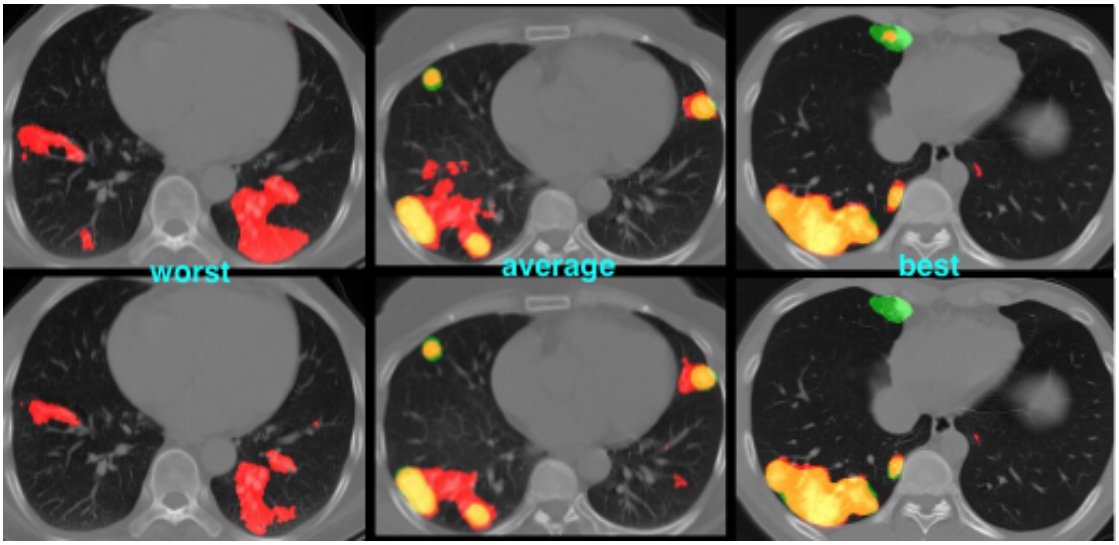

Figure 4.1: In first row are samples from our model using DiceDiceLoss function and below are inference examples from model using DiceFocal Loss. Red is model segmentation, green is ground truth and yellow signalizes correct segmentation.

best predictions. We marked ground truth with green color and the model's segmentation with red. Overlapping regions indicated correct segmentation resulted in yellow color. We can see there are differences between the performance in Fig. 4.1.

We looked closer to the value distribution of positive classified values and we found out that our model and baseline in the worst-case failed to predict the underlying distribution and classified wrongly all voxels on a depicted slice in Fig. 4.1. In the average case, distribution was hard to learn for our model due to similar values in positive and negative cases. The interval from -800 to -600 has roughly 4 times more values than ground truth segmentation. That means our model tried to classify these values as positive ones, but in reality, they were negative; see Fig. A.3. In the best case, both models predicted well across all CT slices achieving an above 0.80 dice score.

Figure 4.2: Results of 3-fold cross validation. Each graph shows averaged results for each epoch.

Chapter 5

Conclusion

Developing a system to detect COVID-19 disease was a challenging task on multiple levels. We run more then 70 experiments total as we progressed through this work. We encountered a memory problem and we proposed solutions to partially eliminate these tighter constraints. We cropped the dataset which allowed us to train and experiment with available hardware and also speed up the training process. Cropping the dataset along the axial axis consisted of training the 2D UNet model on a publicly available dataset of annotated CT lung slices. Finally, the cropped dataset was reduced in size by 58.9%.

As our dataset contained highly imbalanced data we explored the loss functions which perform well in this regard. After the first experiment, we found working averaging the Dice coefficient with the Jaccard index which slightly improved performance. After that, we further improved the loss function to the DiceDice loss function with the parameter to account for class imbalance. After numerous experiments, we proposed loss function modification as we wanted further explore the possibilities of performance improvement through loss function modification. We experimented with the idea of counterbalancing loss function with

exponential terms to punish inaccurate classifications. After performing 3-fold cross-validation and comparing our loss function with Dice Focal loss with default parameters results showed that our model performed better on dice and precision metrics with statistical significance. We evaluated the best models from each cross-validation run on the test set of CT images. Our DiceDice model resulted in $mean = 0.5490$, $std = 0.1443$ and $median = 0.5321$ on Dice coefficient against baseline $mean = 0.5167$, $std = 0.1492$ and $median = 0.4825$. We also put these results into relation with competition[38] results which indicated a good enough performance of our model. As the best competitor achieved 0.5980 Dice coefficient with $std = 0.2640$ and $median = 0.7000$. The evaluation process of the competition was not the same as ours as they used a slightly larger testing dataset with different CT images. We concluded our results as good enough in relation to the competition results as weren't far behind in the Dice coefficient metric. We also pointed out the fact that we used the most basic UNet model without special optimizations and were solely focused on dataset augmentation and loss function modifications.

There are two main branches of our work that may be explored further loss function modification based on dataset class imbalance and architecture tuning with respect to class imbalance. We showed that with a simple model good enough results can be achieved. Our model may be further improved by utilizing either other deep learning blocks or building more deeper network. Imbalanced data are prevalent and we need more tools to handle this problem.

Our limitation was the runtime of running experiments and high hardware requirements. There is an opportunity to explore our model performance after more cross-validation folds.

Resumé

Vznik pandemickej situácie v roku 2019, sposobená ochorením COVID-19, podnietila svet k výskumu vakcíny a metód ako tento vírus odhaliť. Objavili sa nové výzvy v oblasti medicíny, vďaka ktorým vznikli medzinárodne výzvy[38] na detekciu ochorenia COVID-19. Výzvy a súťaže sprístrupnili nazhromaždené dáta verejnosti a výskumníkom. My sme sa do tejto výzvy[38] zapojili, aby sme vytvorili riešenie na nimi poskytnutých dátach. V analytickej časti sme zmapovali momentálny stav v tejto výskumnej oblasti a zhrnuli dôležité prístupy. Na základe analytickej časti sme sa rozhodli postaviť náš prototyp na U-Net architektúre. V ďalšej časti práce sme navrhli riešenie na protyp, kde sme sa držali našich 2 výskumných cieľov - úprava datasetu a úprava stratovej funkcie. V poslednej časti práce sme naše riešenie overili na testovacoch datasete.

V našej práci sme sa venovali problematike segmentácie pľúcnych lézií zo snímok CT pacientov s týmto ochorením. Cieľom práce bolo vytvoriť prototyp na segmentáciu týchto lézií a vyhodnotiť výsledky. V analytickej časti práce sme analyzovali rôzne relevantné architektúry ako sú U-Net[37], V-Net[31] a ich rôzne variacie [17, 51], na źaklade čoho sme sa rozhodli využiť túto architektúru na riešenie nášho problému. U-Net architektúra je stále veľmi známa a často využívaná v oblasti medicínskej domény. Počas analýzy oblasti segmentácie CT snímkov, ktoré môžeme označiť aj ako 3D dáta, sme narazili na problém s veľkosťou a

nevyváženosťou pozitívnych a negatívnych segmentačných tried. Veľkosť CT snímok sa prejavila pri práci s pamäťou počas výpočtov na grafickej karte. Toto obmedzovalo rýchlosť trénovania a teda aj našu rýchlosť tvorby prototypu, keďže bolo trénovanie časovo náročné a tým aj iterácie tvorby prototypu. Na základe prvých experimentov sme našu pozornosť zamerali na modifikáciu samotného datasetu a redukovanie priestorovej veľkosti CT snímok a v druhom rade sme upravovali výlučne stratovú funkciu, pričom sme nemenili architektúru a neoptimalizovali jej hyper-paramatre. Sústredili sme sa najmä na úpravu stratovej funkcie, aby sme maximalizovali efektívnosť na našich dátach. Následne sme pozorovali, aký vplyv má úprava stratovej funkcie a zredukovanie priestorovej veľkosti datasetu na výsledky.

Na riešenie problému sme zostrojili prototyp pomocou vývojového rámca MONAI[32]. Problém sme rozdelili na dve časti: orezanie CT snímok z datasetu a segmentácia COVID-19 pľúcnych lézií. Využili sme teda 2 hlboké neurónové sieťe. Na orezanie CT snímok sme využili 2D U-Net, ktorú sme natrénovali na axiálnych výrezoch CT snímkov z verejne dostupných dát aj s maskami pľúc[29]. Následne sme z každého CT snímku vybrali 3 stredné výrezy, na ktorý sme vysegmentovali pľúca a zostrojili sme z nich orezovú masku pozdĺž axialnej osi. Táto maska bola očistená o malé artefakty pomocou tradičných metód počítačového videnia. Následne sme pomocou tejto masky orezali celý CT snímok pozdĺž axialnej osi. Vo výsledku sme mali snímky rôznych veľkostí so zachytenou oblasťou pľúc. Následne sme na experimentovanie využili tento orezaný dataset, ktorý mal o 58.9% menšiu veľkosť ako pôvodný dataset. Úpravu stratovej funkcie sme prispôsobili k našej úlohe a datasetu. Vzhľadom na nerovnosť jednotlivých tried sme využili samotný pomer na odhad parametra γ, ktorý sa v stratovej funkcií nachádza, aby sme generalizovali využitie našej stratovej funkcie aj na iné datasety. Tento parameter bol inšpirirovaný z práce, ktorá predstavila Focal loss[26]. Stratová funkcia

mala tvar $L = \frac{DSC+DSC^{\gamma}}{2} + CE$. Kde DSC je Dice koeficient a DSC^{γ} je trest, ktorý v sa pomocou priemeru prejaví v stratovej funkcií pri chybnej segmentácií. Tento trest sa následne prejavil v priemere signifikantne lepšej segmentácii o 0,07 na metrike Dice voči porovnávanej funkcií stratovej funkci DiceFocalLoss so základným nastavením na testovacích dátach. Na vyhodnotenie sme použili metódu krížovej validácie, aby sme čo najviac potlačili zaujatosť trénovacích a testovacích dát. Zvolili sme 3-krížovú validácie, kde v každej časti bolo 120 trénovacích a 60 testovacích vzoriek. Tým sme dostali 3 najlešie modely, ktoré sme následne ešte použili na zvyšných 19 vzoriek, ktoré slúžili ako testovacie dáta. Vyhodnotli sme výsledky a záver ich dali do relácie s výsledkami zo súťaže, keďže sme nemali prístup k testovacím dátam ako ostatní súťažiaci.

V práci sa nám podarilo dosiahnuť stanovené výskumné cieľe. Zkonštruovaným prototypom sme potvrdili prínos orezania vstupných dát a úpravu stratovej funkcie na náš dataset. Upravená stratová funkcia signifikantne zlepšila výsledok voči porovnávanej funkcií. Z čoho vyplynulo, že má zmysel optimalizovať a prispôsobovať stratovú funkciu na náš problém pre dosiahnutie lepších výsledkov. Tiež chceme zdôrazniť orezanie vstupného datasetu, čo malo tiež pozitívny vplyv na rýchlosť učenia a pracovanie s datasetom v kontexte náročnosti na grafickú pamäť. Tým, že sme vybrali tú najzákladnejšiu 3D U-Net architektúrú, ktorú sme v priebehu experimentov nemenili, môžeme konštatovať, že experimentami s architektúrou by sa dalo výsledky zlepšíť ešte viac. Optimalizácia architektúry, hyper-parametrov, ale bolo mimo nášho výskumného rámca a zamerania.

Našim prínosom bolo definovanie metódy na orezávania dát a aj samotný odhad parametra γ z nerovnosti dát, ktorý sa dá využiť na podobných datasetoch, kde veľkosť dát a nerovnosť sťažuje proces učenia. Náše metódy sú aplikovateľné s malými úpravami aj na iné datasety, ktoré obsahuje CT snímky hrudníka. Taktiež

je možné využiť nami definovanú stratovú funkciu na iných problémoch a upraviť si ju podľa potrieb.

Bibliography

[1] Tao Ai et al. "Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases". In: *Radiology* 296.2 (2020). PMID: 32101510, E32–E40. DOI: 10.1148/radiol.2020200642. eprint: https://doi.org/10.1148/radiol.2020200642. URL: https://doi.org/10.1148/radiol.2020200642.

[2] R. Anand et al. "An improved algorithm for neural network classification of imbalanced training sets". In: *IEEE Transactions on Neural Networks* 4.6 (1993), pp. 962–969. DOI: 10.1109/72.286891. URL: https://doi.org/10.1109/72.286891.

[3] Syed Anwar et al. "Medical Image Analysis using Convolutional Neural Networks: A Review". In: *Journal of Medical Systems* 42 (Oct. 2018), p. 226. DOI: 10.1007/s10916-018-1088-1.

[4] Harrison X. Bai et al. "Performance of Radiologists in Differentiating COVID-19 from Non-COVID-19 Viral Pneumonia at Chest CT". In: *Radiology* 296.2 (2020). PMID: 32155105, E46–E54. DOI: 10.1148/radiol.2020200823. eprint: https://doi.org/10.1148/radiol.2020200823. URL: https://doi.org/10.1148/radiol.2020200823.

[5] Cuiping Bao et al. "Coronavirus Disease 2019 (COVID-19) CT Findings: A Systematic Review and Meta-analysis". In: *Journal of the American College*

of Radiology 17.6 (June 2020), pp. 701–709. DOI: 10.1016/j.jacr.2020.03.006. URL: https://doi.org/10.1016/j.jacr.2020.03.006.

[6] Giuseppe Bonaccorso. *Machine Learning Algorithms: Popular algorithms for data science and machine learning*. Packt Publishing Ltd, 2018.

[7] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2009, pp. 475–482.

[8] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. In: *International Journal of Computer Vision* 22.1 (1997), pp. 61–79. DOI: 10.1023/a:1007979827043. URL: https://doi.org/10.1023/a:1007979827043.

[9] Vicent Caselles et al. "A geometric model for active contours in image processing". In: *Numerische Mathematik* 66.1 (Dec. 1993), pp. 1–31. DOI: 10.1007/bf01385685. URL: https://doi.org/10.1007/bf01385685.

[10] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[11] Liang-Chieh Chen et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. 2017. DOI: 10.48550/ARXIV.1706.05587. URL: https://arxiv.org/abs/1706.05587.

[12] Li Deng and Dong Yu. "Deep learning: methods and applications". In: *Foundations and trends in signal processing* 7.3–4 (2014), pp. 197–387.

[13] Michal Drozdzal et al. *The Importance of Skip Connections in Biomedical Image Segmentation*. 2016. DOI: 10.48550/ARXIV.1608.04117. URL: https://arxiv.org/abs/1608.04117.

[14] Deng-Ping Fan et al. "Inf-Net: Automatic COVID-19 Lung Infection Segmentation From CT Images". In: *IEEE Transactions on Medical Imaging* 39.8 (2020), pp. 2626–2637. DOI: 10.1109/TMI.2020.2996645.

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

[16] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. "Experimental perspectives on learning from imbalanced data". In: *Proceedings of the 24th international conference on Machine learning - ICML '07*. ACM Press, 2007. DOI: 10.1145/1273496.1273614. URL: https://doi.org/10.1145/1273496.1273614.

[17] Fabian Isensee et al. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation". In: *Nature Methods* 18.2 (Dec. 2020), pp. 203–211. DOI: 10.1038/s41592-020-01008-z. URL: https://doi.org/10.1038/s41592-020-01008-z.

[18] Shuo Jin et al. "AI-assisted CT imaging analysis for COVID-19 screening: Building and deploying a medical AI system in four weeks". In: *medRxiv* (2020). DOI: 10.1101/2020.03.19.20039354. eprint: https://www.medrxiv.org/content/early/2020/03/23/2020.03.19.20039354.full.pdf. URL: https://www.medrxiv.org/content/early/2020/03/23/2020.03.19.20039354.

[19] Justin M. Johnson and Taghi M. Khoshgoftaar. "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6.1 (Mar. 2019). DOI: 10.1186/s40537-019-0192-5. URL: https://doi.org/10.1186/s40537-019-0192-5.

[20] Ma Jun et al. *COVID-19 CT Lung and Infection Segmentation Dataset*. Version Verson 1.0. Apr. 2020. DOI: 10.5281/zenodo.3757476. URL: https://doi.org/10.5281/zenodo.3757476.

[21] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. "Snakes: Active contour models". In: *International Journal of Computer Vision* 1.4 (Jan.

1988), pp. 321–331. DOI: 10.1007/bf00133570. URL: https://doi.org/10.1007/bf00133570.

[22] Osman Semih Kayhan and Jan C. van Gemert. *On Translation Invariance in CNNs: Convolutional Layers can Exploit Absolute Spatial Location*. 2020. arXiv: 2003.07064 [cs.CV].

[23] Salman Khan et al. "Striking the right balance with uncertainty". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 103–112.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, 1097–1105.

[25] Chen-Yu Lee et al. *Deeply-Supervised Nets*. 2014. arXiv: 1409.5185 [stat.ML].

[26] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[28] Siwei Lyu et al. "Average Top-k Aggregate Loss for Supervised Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.1 (Jan. 2022), pp. 76–86. DOI: 10.1109/tpami.2020.3005393. URL: https://doi.org/10.1109/tpami.2020.3005393.

[29] Kevin Mader. *Finding and Measuring Lungs in CT Data*. 2017. URL: https://www.kaggle.com/kmader/finding-lungs-in-ct-data.

[30] Ravikanth Malladi, James A Sethian, and Baba C Vemuri. "Topology-independent shape modeling scheme". In: *Geometric Methods in Computer Vision II*.

Vol. 2031. International Society for Optics and Photonics. 1993, pp. 246–258.

[31] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[32] MONAI Consortium. *MONAI: Medical Open Network for AI*. 2022. DOI: 10.5281/ZENODO.4323058. URL: https://zenodo.org/record/4323058.

[33] Dominik Müller, Iñaki Soto Rey, and Frank Kramer. *Automated Chest CT Image Segmentation of COVID-19 Lung Infection based on 3D U-Net*. 2020. arXiv: 2007.04774 [eess.IV].

[34] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. arXiv: 1811.03378 [cs.LG].

[35] An P et al. *CT Images in Covid-19 [Data set]*. 2020. DOI: 10.7937/tcia.2020.gqry-nc81. URL: https://doi.org/10.7937/tcia.2020.gqry-nc81.

[36] Enislay Ramentol et al. "SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory". In: *Knowledge and information systems* 33.2 (2012), pp. 245–265.

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

[38] Holger Roth et al. "Rapid Artificial Intelligence Solutions in a Pandemic - The COVID-19-20 Lung CT Lesion Segmentation Challenge". In: (June 2021). DOI: 10.21203/rs.3.rs-571332/v1. URL: https://doi.org/10.21203/rs.3.rs-571332/v1.

[39] Geoffrey D. Rubin. "CT Diagnosis of COVID-19: A View through the PICOTS Lens". In: *Radiology* 301.1 (2021). PMID: 34184939, E375–E377. DOI: 10.1148/radiol.2021211454. eprint: https://doi.org/10.1148/radiol.2021211454. URL: https://doi.org/10.1148/radiol.2021211454.

[40] Seyed Sadegh Mohseni Salehi, Deniz Erdogmus, and Ali Gholipour. "Tversky loss function for image segmentation using 3D fully convolutional deep networks". In: *CoRR* abs/1706.05721 (2017). arXiv: 1706.05721. URL: http://arxiv.org/abs/1706.05721.

[41] Fei Shan et al. "Abnormal lung quantification in chest CT images of COVID-19 patients with deep learning and its application to severity prediction". In: *Medical Physics* 48.4 (Mar. 2021), 1633–1645. ISSN: 2473-4209. DOI: 10.1002/mp.14609. URL: http://dx.doi.org/10.1002/mp.14609.

[42] Sofie Tilborghs et al. *Comparative study of deep learning methods for the automatic segmentation of lung, lesion and lesion type in CT scans of COVID-19 patients*. 2020. arXiv: 2007.15546 [eess.IV].

[43] Elena Šikudová et al. *Počítačové videnie Detekcia a rozpoznávanie objektov*. Praha: Wikina,Praha, 2013, p. 397. ISBN: 978-80-87925-06-5. URL: http://vgg.fiit.stuba.sk/kniha/.

[44] Haishuai Wang et al. "Predicting Hospital Readmission via Cost-Sensitive Deep Learning". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15.6 (Nov. 2018), pp. 1968–1978. DOI: 10.1109/tcbb.2018.2827029. URL: https://doi.org/10.1109/tcbb.2018.2827029.

[45] W.H.O. *Coronavirus disease COVID-19 pandemic*. https://www.who.int/emergencies/diseases/novel-coronavirus-2019. 2020.

[46] Zhe Wu, Li Su, and Qingming Huang. *Cascaded Partial Decoder for Fast and Accurate Salient Object Detection*. 2019. arXiv: 1904.08739 [cs.CV].

[47] Qingsen Yan et al. *COVID-19 Chest CT Image Segmentation – A Deep Convolutional Neural Network Solution*. 2020. arXiv: 2004.10987 [eess.IV].

[48] Jia-Xing Zhao et al. *EGNet:Edge Guidance Network for Salient Object Detection*. 2019. arXiv: 1908.08297 [cs.CV].

[49] Zongwei Zhou et al. *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*. 2018. arXiv: 1807.10165 [cs.CV].

[50] Jieyun Zhu et al. "CT imaging features of 4121 patients with COVID-19: A meta-analysis". In: *Journal of Medical Virology* 92.7 (Apr. 2020), pp. 891–902. DOI: 10.1002/jmv.25910. URL: https://doi.org/10.1002/jmv.25910.

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