Eötvös Loránd University

Faculty of Informatics

Department OF COMPUTER ALGEBRA

**Volumetric left ventricle segmentation of MPI SPECT images with self-supervised learning**

*Supervisor:* *Author:*

Ádám István Szűcs Kristóf Máté Horváth

Doctorate candidate MSc in Computer Science

*Budapest, 2022*

**Text, letter

Description automatically generated**

**EÖTVÖS LORÁND UNIVERSITY**

**FACULTY OF INFORMATICS**

**Thesis Registration Form**

**Student’s data:**

**Student’s Name:** Kristóf Máté Horváth

**Student’s Neptun code:** NIQY2L

**Course Data:**

**Student’s Major:** Computer Science for Autonomous Systems MSc

I have an internal supervisor.

***Internal Supervisor’s Name:*** *Ádám István Szűcs*

* Supervisor’s Home Institution: *Eötvös Loránd University*
* Address of Supervisor’s Home Institution: *1053 Budapest, Egyetem tér 1-3*
* Supervisor’s Position and Degree: *Research assistant, MSc*

**Thesis title:** **Volumetric left ventricle segmentation of MPI SPECT images with self-supervised learning**

**Topic of the Thesis:**

Successful real-world applications of machine learning techniques utilize a large amount of manually labeled data to achieve satisfiable performance. Latest research diverged from the traditional approach since on-pair performance was obtained with self-supervised learning methods. Self-supervised learning techniques require only a small amount of labeled data to fine tune machine learning models after unlabeled data was incorporated into the training procedure to extract useful features. Single-photon emission computed tomography (SPECT) is a nuclear medicine tomographic imaging technique which works with γ rays, which is a widely used procedure to detect heart diseases. Many previous works approached the problem of heart ventricle segmentation, but most of them concentrated on labeled data or have not utilized the volumetric nature of the data that SPECT reconstruction provides. The thesis will concentrate on left ventricle segmentation of the heart, based on 3-dimensional SPECT images and various self-supervised learning methods, which will be compared to the state-of-the-art and evaluated quantitatively. The thesis will build on many medical and machine learning results, which aimed volumetric segmentation of organs.

Budapest, 2022.12.09.

**Contents**

[**1. Introduction to the medical domain** 3](#_Toc103020571)

[**1.1. The human heart and the left ventricle** 3](#_Toc103020572)

[**1.2. Myocardial perfusion SPECT** 4](#_Toc103020573)

[**1.3. Digital Imaging and Communications in Medicine** 5](#_Toc103020574)

[**1.4. Coordinate systems** 5](#_Toc103020575)

[**1.5. Thesis hypothesis** 6](#_Toc103020576)

[**2. Building blocks of Convolutional Neural Networks** 7](#_Toc103020577)

[**2.1. Convolution and Up-Convolution** 7](#_Toc103020578)

[**2.2. Max-pooling** 9](#_Toc103020579)

[**2.3. Activation functions** 10](#_Toc103020580)

[**2.4. Rectified Linear Unit and Parametric Rectified Linear Unit** 10](#_Toc103020581)

[**2.5. Receptive Field of Convolutional Networks** 11](#_Toc103020582)

[**2.6. Effective Receptive Field of Convolutional Networks** 12](#_Toc103020583)

[**2.7. Dropout** 14](#_Toc103020584)

[**2.8. Normalization layers** 15](#_Toc103020585)

[**2.9. Batch Normalization** 16](#_Toc103020586)

[**2.10. Instance Normalization** 18](#_Toc103020587)

[**2.11. Group Normalization** 19](#_Toc103020588)

[**2.12. Skip Connections** 20](#_Toc103020589)

[**3. Loss functions** 22](#_Toc103020590)

[**3.1. Binary Cross-Entropy** 22](#_Toc103020591)

[**3.2. Weighted Binary Cross-Entropy** 22](#_Toc103020592)

[**3.3. Dice loss** 22](#_Toc103020593)

[**3.4. Focal loss** 23](#_Toc103020594)

[**3.5. Log-Cosh Dice loss** 24](#_Toc103020595)

[**4. Metrics** 26](#_Toc103020596)

[**4.1. Accuracy** 26](#_Toc103020597)

[**4.2. Precision** 26](#_Toc103020598)

[**4.3. Recall** 27](#_Toc103020599)

[**4.4. Intersection over Union** 27](#_Toc103020600)

[**5. Architectures for segmentation** 28](#_Toc103020601)

[**5.1. U-Net for image segmentation** 28](#_Toc103020602)

[**5.2. 3D U-Net for volumetric segmentation** 29](#_Toc103020603)

[**6. Frameworks for self-supervised representation learning** 31](#_Toc103020604)

[**6.1. Visual representation learning from unlabeled data by Context Prediction** 31](#_Toc103020605)

[**6.2. Solving jigsaw puzzles as a pretext task** 34](#_Toc103020606)

[**7. Experiments** 38](#_Toc103020607)

[**7.1. Labeling** 38](#_Toc103020608)

[**7.2. Exploratory Data Analysis** 39](#_Toc103020609)

[**7.3. Preprocessing** 40](#_Toc103020610)

[**7.4. Representation learning from context prediction** 43](#_Toc103020611)

[**7.5. Representation learning from solving jigsaw puzzles of SPECT volumes** 46](#_Toc103020612)

[**7.6. Supervised fine-tuning** 50](#_Toc103020613)

[**7.7. Supervised fine-tuning after the relative patch location pretext task** 51](#_Toc103020614)

[**7. 8. Supervised fine-tuning after the jigsaw puzzle pretext task** 53](#_Toc103020615)

[**8. Conclusion** 61](#_Toc103020616)

[**9. Future work** 62](#_Toc103020617)

[**10. Acknowledgement** 63](#_Toc103020618)

[**11. References** 64](#_Toc103020619)

# **1. Introduction to the medical domain**

## **1.1. The human heart and the left ventricle**

The human heart’s role is to move the blood throughout the human body, while it controls the rhythm and speed of its rate and maintains blood pressure [31]. The heart works together with the nervous and the endocrine systems. The nervous system sends signals to the heart to control its pump rate. The Endocrine system sends out hormones which influence the blood vessels and consequently the blood pressure. The heart weighs 200 to 425 grams and on average it is a little larger than a human fist. Figure 1 illustrates the anatomy of the human heart. The heart consists of 4 chambers. The upper chambers are called the left and right atria and the lower chambers are called left and right ventricles. The chambers are separated by wall of muscles, which is called septum.

Diagram

Description automatically generated

Figure 1: Anatomy of the human heart [30].

The left ventricle is the strongest chamber, which pushes blood into the body through the aortic valve. The blood flow of the heart is regulated by the tricuspid-, pulmonary-, mitral- and aortic valve. The blood flow between the right atrium and the left ventricle is regulated by the tricuspid. The blood that is directed to the lungs flows through the pulmonary articles from the right ventricle and it is controlled by the pulmonary valve. The oxygen rich blood that flows from the lungs passes the left atrium and ends in the left ventricle, influenced by the mitral valve. The aortic valve controls the way of the oxygen-rich blood that flows from the left ventricle into the body’s largest artery, the aorta.

The cardiovascular system is comprised of the heart and the circulatory system. The pumping of the heart pushes blood to the tissues, organs and cells of the body. The blood delivers nutrients and oxygen to the cells while it also removes carbon dioxide and waste products made by the cells.

The conduction system is responsible for the delivery of electronical impulses that causes the heart to contract. A natural pacemaker called the sinoatrial node, that is located at the top of the right atrium sends electrical signals at a certain rate to influence the contraction of the atria and ventricles. Other factors, such as physical demand, stress and hormonal factors can affect the contraction.

## **1.2. Myocardial perfusion SPECT**

The quantity of blood that flows through one cm3 of an organ or tissue per unit time is called perfusion [[1]](#footnote-1). Sufficiently decreased perfusion in an organ or in a region of an organ can lead to the necrosis of the affected tissue. The myocardium can take up radioactive products in proportion to the blood that streams through it, which facilitates the examination of the myocardium. Single-photon emission computerized tomography (SPECT) is a nuclear type imaging test, which uses radioactive substance and specialized camera to create 3D images. The left ventricular function is assessed through one of the most important imaging modalities, called Myocardial perfusion SPECT (MPS) [30]. A radioactive perfusion tracker is injected, and it is taken up by the left ventricular myocardium. The perfusion tracker emits photons, which are detected to reconstruct perfusion images. MPS provides 8 or 16 volumetric perfusion image sets through electrocardiographic gating. These image sets correspond to different phases of the cardiac cycle. For the diagnosis or prognosis of coronary artery disease or patient risk assessment, left ventricular contractile functional indices are derived from the MPS images.

## **1.3. Digital Imaging and Communications in Medicine**

Digital Imaging and Communications in Medicine (DICOM) is the standard of storing and transmitting medical images. Modern medical imaging modalities like computed tomography (CT), magnetic resonance imaging (MRI) and SPECT images are stored as DICOM, which can integrate medical modalities that are manufactured by different vendors [31]. The DICOM file format stores medical image data and data sets with attributes of patient and diagnosis, such as name, birth date and acquisition data related to the modality and its settings. The DICOM network protocol defines a transport layer protocol that is used by medical imaging applications in hospitals. The protocol facilitates the information exchange between hospitals and the search for image studies in archives.

## **1.4. Coordinate systems**

In the medical domain there are three commonly used coordinate systems for imaging applications. These are the world, the anatomical and the medical image coordinate systems [31].

The medical imaging modality is positioned in the world coordinate system, which is a Cartesian coordinate system, and it defines the position and orientation of the modality.

The medical imaging coordinate system is also called voxel space, and it contains 3D arrays or otherwise tensors.

The anatomical space is also called patient coordinate system. The standard anatomical position of a human is described by three planes: the axial, the sagittal and the coronal plane, which together constitute the anatomical space. Figure 2 illustrates the three planes. The green plane is the axial, which is a top-view plane. On the axial plane, near the head a slice is referred to as superior, while closed to the leg a slice is referred to as interior.

Diagram

Description automatically generated

Figure 2: Medical imaging coordinate systems. From left to right - World, anatomical and medical imaging coordinate systems [31].

The sagittal plane is a side view as it is visualized by the pink plane on Figure 2. The view can be left or right and the positive direction depends on the coordinate system. The coronal plane is the purple plane on Figure 2. It is called the anterior plane, when the viewer looks into the eye of the patient, and it is called posterior plane, when the viewer looks at the back of the patient.

## **1.5. Thesis hypothesis**

The quantification accuracy of the left ventricular myocardium volume from the MPS directly relates to the fidelity of the left ventricular function assessment [31]. The manual segmentation is tedious and depends on the observer’s expertise. Many supervised machine learning procedures require large number of labels, which further require human involvement and in the medical field specialized experts are needed. In this work, the hypothesis is that it is possible to train a general feature detector for left ventricle segmentation, with relatively new self-supervised learning algorithms that reduce the label requirement of neural networks.

# **2. Building blocks of Convolutional Neural Networks**

Convolutional Neural Network (CNN) is a subclass of Artificial Neural Networks, which achieved outstanding results in the fields of computer vision and even outperformed human level performance on image classification, object-detection and many other tasks. Medical field is no exception, and in the recent years many networks approximated the performance of human experts on disease detection, classification and organ segmentation [28]. CNN is typically applied but not limited to the processing of images or image sequences. CNN is capable to learn spatial hierarchical features through convolutional, pooling and fully connected layers, which realize mathematical operations. Convolutional and pooling layers perform feature extraction and image down or up-sampling, while dense layers are utilized to map the extracted features to the true labels [28].

## **2.1. Convolution and Up-Convolution**

A Convolutional layer implements the mathematical linear convolution operator, and it is associated with a kernel (also called filter), which contains real numbers in an array like fashion. Kernels can be one, two or three dimensional.

The parameters of the kernel are learnable weights. The kernel is placed on the image on every possible location in order to convolve the kernel with the image, which means that for every possible location the elementwise dot-product of the image and the kernel is computed. The possible locations for the kernel are defined by a predefined set of hyperparameters, such as kernel size, padding and stride. The kernel size defines the shape of the kernel, and it typically has squared size, like 5x5 and 7x7 for 2D convolutional layers. Convolution reduces the input data shape and therefore, padding is usually applied to the input data. Padding is also called zero-padding because it is most often used to pad the image with zeros along specified dimensions to control the shape of the output feature map. The input of a neural network layer is usually called tensor, while the output of the Convolutional layer is called a feature map, which are both multidimensional arrays. The stride defines the number of unit length steps e.g., pixels that the kernel moves between each neighboring position. Stride typically has a value of 1, but values greater than 1 can be used for down-sampling purposes. The number of kernels hyperparameter defines how many kernels are applied in a Convolutional Layer for feature extraction. Each filter is capable to extract unique and independent features. Figure 3 illustrates how a certain output feature is computed with the convolutional layer from an input tensor and a filter. The kernel is size of 3x3, and it overlaps a certain area of the input tensor. From the overlapping area the element-wise dot product is computed to produce a certain output value on the feature map. The same operation is repeated for every possible position with respect to the input tensor.

Diagram, engineering drawing

Description automatically generated

Figure 3: Convolution of an input tensor and a kernel and the resulting position on the feature map [28].

The shape of the output feature map for a convolutional layer with an input shape of *W1xH1xD1* can be computed with the Equations (1) - (**3**) [[2]](#footnote-2),

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |

where *W*, *H* and *D* refers to the width, height and depth dimensions of the input and output data, *F* is the filter size, *P* is the padding, *S* is the stride and *K* is the number of filters. The depth of the produced feature map is equal to the number of filters. The Convolutional layer is translation invariant with respect to its input, due to its parameter or weight sharing properties [28], [[3]](#footnote-3). Weight sharing means that the same kernel is applied to every position on the input data, which dramatically reduces the complexity of the model in terms of parameters.

## **2.2. Max-pooling**

Pooling layers serve as down-sampling layers to decrease the spatial dimensions of the input data. Pooling layers increase the network’s invariance to small shifts and distortions, and they reduce the number of learnable parameters of subsequent layers [28].

Diagram, table

Description automatically generatedMax-pooling layer partitions the input tensor into non-overlapping subregions, which are called patches. The resulting patches are usually have squared size. For each patch, the max-pooling operation takes the maximum of the values and produces an output value at a certain index for the output tensor. Figure 4 illustrates the max-pooling layer with an input tensor of size 4x4, kernel size of 2x2 and the resulting output tensor of size 2x2. The 2x2 kernel reduces the spatial dimensions in each direction by a factor of 2.

Figure 4: Max-pooling operation with a kernel size of 2x2 and the resulting output tensor.

Alternatives for Max-pooling exist such as Average-pooling, which instead of taking the maximum of each patch, it takes the average of the values.

## **2.3. Activation functions**

The purpose of activation functions is to add non-linearity to the neural network. Activation functions are the basic building blocks of neural networks that enable them to approximate non-linear functions, or in other words facilitate the learning of highly complex tasks. An activation function is usually applied after the convolution and max-pooling operations. Commonly used activation function are the Rectified Linear Unit (ReLU) and its variants.

## **2.4. Rectified Linear Unit and Parametric Rectified Linear Unit**

According to [29], an activation function can be formally defined by Equation (4)

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

, where *yi* is the input of the nonlinear activation function *f* on the *i*th channel, and *ai* is a coefficient that controls the slope of the negative part of the function. The index *i* indicates that the activation function can vary on different channels [29]. When *ai* is 0, Equation (4) becomes the ReLU function. Equation (4) can be equivalently written as .

Parametric Rectified Linear Unit (PReLU) is a generalization of the original ReLU function, which was introduced in [29], as a partial solution to train deep neural networks. PReLU introduced *ai* as a learnable parameter in Equation (4), which introduces a small number of extra parameters compared to the number of feature channels. If *ai* is a small number that avoids zero gradients, the activation function is called Leaky Rectified Linear Unit (LReLU). Figure 5 demonstrates the shape of the ReLU and PReLU activation functions.

**Diagram, text

Description automatically generated**

Figure 5: Relu (left) and PReLU (right) activation functions [29].

## **2.5. Receptive Field of Convolutional Networks**

Understanding the Receptive Field (RF) of convolutional neural networks is essential to design architectures that can perform well on certain tasks. Typically, in object-detection tasks, if the span of a convolutional feature is small, the network might not be able to detect small objects and vice-versa [26]. The RF associates an output feature with an input region that produced the certain output feature [26]. In other words, the computation of the RF aims to measure the extent of the effect of the input signals on the output features, while it also traces back the output features to the inputs that produced them. The work of [26] presented closed-form solutions in the form of recurrence equations for the computation of the RF of modern convolutional neural networks, both for single-path (AlexNet [4]) and multiple path (ResNet [9]) cases. A feature map of a fully-convolutional network (FCN) [8] is denoted by , where the index *l* specifies that the feature map is the output of the *l*th layer, *hl*, *wl* and *dl* represents the height, width and depth dimensions, respectively. The input of the FCN is denoted by *f0,* and the output feature map of the last layer is denoted by *fL*, thus the FCN has *L* layers. Additional parameters that are considered in the following equations are *kl*, *sl*, *pl*, and *ql,* which represent the kernel size, stride and the padding of the left and right side of the input feature map, respectively. Figure 6 illustrates the notations and shows how one position of the kernel affects a certain position of the output feature map.

Diagram

Description automatically generated

Figure 6: Illustration of the parameters used for receptive field computation [26].  
Here, kl=2, pl=1, ql=1 and sl=3.

The size of the RF of the final output layer *fL* is denoted by *rl*. To generate one feature in *fL*, there are *rl* number of features that contribute to the feature map *fl*. The recurrence equation for single-path networks is presented by Equation (5), which is a first-order, non-homogeneous equation with variable coefficients [26].

|  |  |
| --- | --- |
|  | (5) |

The recurrence equation can be solved according to the study of [26], which can be further referenced for a more detailed explanation. Equation (6) formulates the solution for the recurrence Equation (26) in terms of *kl*’s and *sl*’s.

|  |  |
| --- | --- |
|  | (6) |

One can see that when all kernels are of size 1, the RF is also of size 1. When all strides are 1, then the RF is the sum of (*kl - 1*) over all layers plus 1, and when the stride is greater than 1 for a certain layer, for consecutive layers the region of the RF increases proportionally [26].

## **2.6. Effective Receptive Field of Convolutional Networks**

The concept of the Effective Receptive Field (ERF) was introduced in the research of [27], which also empirically showed that it has a Gaussian distribution, and it occupies a fraction of the theoretical RF. In theory, the size of the theoretical RF increases linearly as more layers are stacked together to make the network deeper, while sub-sampling increases the size of the receptive field multiplicatively. It was discovered that not all values of the receptive field contribute equally to the output unit’s response, because values at the center of the RF have a much larger impact. Figure 7 shows that the ERF grows linearly by , while the relative ratio of the ERF over the theoretical RF decreases by , with respect to the number of convolutional layers, where *N* is the number of layers [27]. It was also observed how skip-connections affect the ERF of a network. Like in ResNet [9], the ERF should be significantly smaller than the theoretical RF, because skip-connections have a decreasing effect on the ERF. However, researchers observed that the initially small ERF at the beginning of the training evolves into a much larger one during the training [27].

Chart, line chart

Description automatically generated

Figure 7: The ERF grows linearly as more convolutional layers are stacked together (left), while the relative ratio of the ERF over the theoretical RF shrinks linearly (right) [27].

The theoretical receptive field of a ResNet-17 model with 17 residual blocks has a theoretical receptive field of 74x74. The authors of [27] conducted an experiment on the CIFAR-10 [[4]](#footnote-4). data set and measured the ERF of a ResNet-17 model at the beginning of the training. They found that the ERF had a size of 32x32, which potentially indicated a bad initialization bias. After the model was trained the ERF’s size grew much larger and it almost reached the diameter of the image, as it is shown on Figure 8.

A picture containing logo

Description automatically generated

Figure 8: ERF of a ResNet-17 architecture before and after training on the CIFAR 10 data set [27].

As it was previously stated, the center pixels of the receptive field have a much larger influence on the output feature map. Therefore, the authors of [27] proposed that after any weight initialization the weights should be scaled with a distribution that has lower scale at the center and higher scale on the outside.

## **2.7. Dropout**

Diagram

Description automatically generatedDropout is a regularization technique which facilitates the training of deep neural networks. Training a neural network with dropout can be interpreted as an approximation of an equally weighted geometric mean of the predictions of an exponential number of models that share parameters [13]. Dropout prevents overfitting and effectively combines many different neural network architectures. For this reason, the trained model can be regarded as a model of ensembles. Figure 9 illustrates the concept of dropout. For each neuron in the network, there is a fixed probability *p* for the neuron to be retained during each training epoch. If the neuron is not retained, it is temporarily removed by dropping its input and output connections.

Figure 9: Illustration of dropout in a neural network of dense layers.

Srivastava, Nitish, et al. [13] reported that the optimal dropout rate in the hidden layers is close to 0.5, while for the input layer the dropout rate is close to 0.2. A neural network of *n* neurons can be possibly comprised of *2n* smaller networks which share their weights extensively and many of them have a small chance to be trained. At test time the outgoing weights of each hidden neuron is multiplied by the retention probability *p* to ensure that the expected output of the hidden unit is the same. Dropout reportedly yields a much lower generalization error in a wide range of applications. The reason why dropout works effectively is, because it prevents the co-adaptation of the neurons or feature detectors [14]. Co-adaptation means that the neurons are only capable to effectively detect features in the presence of other active neurons. A model with a high number of parameters can mimic the training data, but when its neurons are co-adapted, the model typically performs poorly on the held-out data.

## **2.8. Normalization layers**

Data normalization is the transformation of the data in the pre-processing stage before the data is fed to a machine learning model or neural network layers. The goal of data normalization is to transform the underlying features in a way to be distributed in the same range, typically in a small interval of [-1, 1] or [0, 1]. Equation (7) formulates a normalization technique called Min-Max scaling.

|  |  |
| --- | --- |
|  | (7) |

Min-Max scaling transforms the data set into a range of [0, 1]. Evidently, when the data is not normalized, features that span larger intervals can influence the training of the model much more than features on smaller scales.

Standardization or Z-Score Normalization transforms the data set features by subtracting the mean from the samples and dividing the result by the standard deviation of the data set. Equation (8) formulates Z-Score Normalization.

|  |  |
| --- | --- |
|  | (8) |

The standardized data set has approximately zero mean and unit variance. Therefore, it closely approximates a Gaussian Bell curve, which has favorable characteristics with respect to the training convergence of machine learning models.

It was proven long before that the training of neural networks converges faster towards an optimal loss value when the inputs are linearly transformed to have zero means and unit variances and when the features are decorrelated [12], [23].

## **2.9. Batch Normalization**

During the training of deep neural networks, the distribution of each layer’s inputs changes, which can cause the training to diverge, while at the same time it also slows down by a great margin [12]. Given a layer of a deep neural network, small changes to the preceding layers can have a huge impact on the distribution of the inputs for the given layer and this is amplified as the layer is located deeper in the network. The layers continuously need to adapt to the distribution changes of each layer’s input data, which is referred as *Internal Covariate Shift* [12]*.* Batch Normalization was proposed as a modification of the neural network’s architecture in order to stabilize the training and facilitate the application of larger learning rates, which also speeded up the convergence. Batch Normalization aims to eliminate the phenomenon of internal covariate shift by fixing the means and variances of each layer’s inputs. Batch Normalization also affects the gradient flow of the network, because the gradients are less dependent on the scale of the parameters or their initial values [12]. It also reduces the rate of dropout that needed be applied, because it has regularization characteristics, and it benefits the model’s generalization.

Text

Description automatically generatedThe algorithm on Figure 10 describes how Batch Normalization and its required statistics are computed over a mini batch of training data.

Figure 10: The algorithm for the computation of Batch Normalization [12].

For each mini batch of examples, the mean and variance of the mini-batch are computed in order to normalize each training sample separately in the mini-batch. The transformed data is indicated with *yi* and it is computed by scaling and translating the normalized data with *γ* and *β*learnable parameters. *γ* and *β* are learnable parameters, because it possible that the optimal transformation should represent the *identity transformation* [12]. ϵ is introduced to ensure that the computation of the normalized data is always numerically stable. After Batch Normalization, each layer that previously received its input as x, it receives its input as BN(x). Figure 11 further illustrates the computation of Batch Normalization.

**Diagram

Description automatically generated**

Figure 11: Illustration of Batch, Layer, Instance and Group Normalization layers [19]. H, W and C represent the image dimensions of height, width and channels, and N represents the batch size. Layer Normalization is not discussed in this work, but one can refer to [24].

During training, the population or the whole training data set running mean and variance are computed through moving averages. This is required, because Batch Normalization is not applied during inference or test time. Instead, the running statistics of the whole population are used to normalize the test samples, as it is formulated by Equation (9).

|  |  |
| --- | --- |
|  | (9) |

Batch Normalization achieves that in each layer the neural network produces activations that always closely approximates the desired Gaussian distribution, which consequently stabilizes and speeds up the training.

## **2.10. Instance Normalization**

Instance Normalization or “contrast normalization” was introduced by Ulyanov et al. [20] to achieve qualitatively better results for image stylization [22] with Generative Adversarial Networks [21]. Ulyanov et al. argued that replacing Batch Normalization layers in the Generator network for Instance Normalization layers allows the network to remove instance-specific contrast information from the content image, because instance-specific mean and covariance shift do not simplify the learning process [20]. Instance Normalization layers are also kept during test time in contrast to Batch Normalization layers, for which the parameters that were learned during the training are frozen and applied during test time. Equation (10) - (12) formulates Instance Normalization

|  |  |
| --- | --- |
|  | (10) |
|  | (11) |
|  | (12) |

, where is an input tensor with a batch size of *T,* and in each batch, an input image has *C* channels and its width and height are denoted by *W* and *H*, respectively. It is noticeable that Instance Normalization computes the mean and standard deviation along the *W* and *H* dimensions, while it fixes the dimensions of *T* and *C*, as it is illustrated on Figure 11 (with the exception that T = N). As a conclusion, Instance Normalization normalizes each training sample separately, therefore, it is beneficial to apply it for large volume segmentations.

## **2.11. Group Normalization**

Group Normalization was introduced as an alternative for Batch Normalization when the training batch size should be small because of memory limits, such as it commonly happens in object detection and segmentation tasks. Therefore, researchers often must compromise between model complexity and batch size. The computation of the statistics for Batch Normalization becomes inaccurate and introduces rapidly increasing error as the batch size becomes smaller [19]. Group Normalization computes the mean and variance for normalization by dividing the channels into groups and normalizes the features within each group separately. Its computation is independent of the batch size. Equation (13) formulates how *Si*, the set of pixels in each group is selected

|  |  |
| --- | --- |
|  | (13) |

, where *G* is a pre-defined hyperparameter and represents the number of groups, *C/G* is the number of channels within each group and [.] is the floor operator. The mean µ and standard deviation σ are computed along the axes of *H, W* and *C/G* channels. Group Normalization also learns the *γ* scale and *β* translation parameters during training. Figure 11 illustrates Group Normalization, where the channels are divided into 2 groups and each of them has 3 channels.

## **2.12. Skip Connections**

Diagram

Description automatically generatedAlthough the concept of Skip or Shortcut Connections were long studied, it was He et al. [9] who successfully applied them for deep neural networks for the first time. Many research including the study of He et al. [9] experimentally verified that as the network grows deeper, the training error increases and therefore, the accuracy of the model degrades. He et al. [9] introduced a *deep residual framework* to address the problem of degradation. Figure 12 visualizes a Residual Block, which contains an identity mapping of the input x, through a skip connection. The Residual Block is built into feed forward neural networks to realize the mapping of . The skip connection performs the *identity mapping*, and its output is added to the stacked layers through element-wise addition. Therefore, the dimensions of *F(x)* and *x* must match.

Figure 12: Residual block of ResNet [9].

The identity mappings do not add extra parameters or computational complexity to the architecture. The authors of [9] call the resulting architectures Deep Residual Networks or ResNets in an abbreviated form. ResNets experimentally verified that even deep neural networks with a thousand layers can be trained efficiently, while the learned features are applicable to other domains as well, and they provide excellent generalization properties.

A picture containing text, underpants

Description automatically generatedSkip connections produce loss functions for which the training procedure becomes easier, while proper choice of training parameters like batch size, learning rate and optimizer can affect the generalization capabilities of the model [25]. However, it is not well understood how they exactly influence the underlying loss space. Li, Hao, et al. [25] developed methods to effectively visualize the loss landscape and they experimentally observed that architectures that include skip connections produce smoother, convex loss landscapes instead of chaotic ones as the network becomes deeper. Figure 13 visualizes the loss landscape of a 56-layer ResNet with and without skip connections. The study of [25] demonstrated that architecture design can cause extreme, non-convex loss landscape structures which directly has an impact on the model’s generalization error. Residual connections prevent the explosion of non-convexity as the network grows deeper [25].

Figure 13: Loss surfaces of ResNet-56 with and without skip connections [25].

# **3. Loss functions**

Loss function is a method to evaluate the performance of a machine learning model. Loss function is also called cost function or objective function. The error between the true labels and the predictions of the model is computed with the loss function. Typically, the goal of the model is to minimize the objective function with respect to the parameters of the model. In the following, loss functions that are commonly used for segmentation problems are discussed, based on the study of Jadon et al. [15].

## **3.1. Binary Cross-Entropy**

Binary Cross-Entropy (BCE) is derived from the Bernoulli distribution, and it measures the difference between two probability distributions for a random variable. It is commonly used as a cost function for classification. As segmentation is a pixel level classification, BCE can be applied. Equation (14) describes the BCE loss function

|  |  |
| --- | --- |
|  | (14) |

, where *y* is the true label and is the predicted one. BCE is a distribution-based loss function, and its best exploitation appears when the data set is balanced.

## **3.2. Weighted Binary Cross-Entropy**

Weighted Binary Cross-Entropy (WBCE) is a variant of Cross-Entropy (CE), which weights the positive samples by some coefficient [15]. Equation (15) formulates WBCE

|  |  |
| --- | --- |
|  | (15) |

, where *β* is a parameter to tune the number of resulting false negatives and false positives. WBCE’s best application is when the data set is skewed.

## **3.3. Dice loss**

The Dice Score Coefficient (DSC) is a measure of overlap between two samples. DSC is formulated by Equation (16)

|  |  |
| --- | --- |
|  | (16) |

, where *pi* is the predicted pixel and *gi* is the ground truth pixel. To formulate a loss function that can be minimized, the DSC is subtracted from 1, as Equation (17) describes it

|  |  |
| --- | --- |
|  | (17) |

, where *y*, are the true and the predicted labels, and 1 is added in both the numerator and the denominator to handle edge case scenarios.

## **3.4. Focal loss**

The Focal loss function was developed for dense object detection tasks, which usually involve extreme amount of foreground-background class imbalance during training [17]. The Focal loss function reshapes the standard CE loss in a way that it down-weights the loss for easy and well-classified examples. Focal loss is basically a dynamically scaled CE loss function where the scaling factor decays to zero as the confidence in the correct class increases, which helps the model to focus on the hard examples. Focal loss has a *γ* parameter which can control the magnitude of the relative loss between the well-classified easy examples and incorrectly classified hard examples. According to Lin, Tsung-Yi, et al. [17], it facilitates the training of dense object detectors such as RetinaNet [17], by preventing the easy examples to overwhelm the model during training. Figure 14 presents the shape of the Focal loss function with different *γ* parameters. The blue curve corresponds to *γ* equal to 0, in which case the Focal loss becomes CE loss. Apparently, CE loss can become a large number when the loss from the frequently occurring easy examples (*pt > 0.5*) are summed up. Therefore, these frequently occurring examples can dominate the training. Lin, Tsung-Yi, et al. [17] call *(1 - pt) γ* a modulating factor, where *γ* is a tunable focusing parameter and it is ≥ 0. When *γ* increases, the loss becomes larger down to a certain point as the confidence in the prediction decreases. At the same time, for increasing *γ* parameters, the loss for predictions with high confidence scores decreases relative to the loss for the hard examples. One can see that a prediction for which the model has a confidence of *pt = 0.6*, the function yields decreasing loss values as *γ* increases. For a prediction with a confidence of *0.2*, the loss also decreases as *γ* increases, but the difference between the low and high confidence predictions increases. If the confidence of the prediction is very low (*pt < 0.1*), the function yields almost identical values for different *γ* parameters. Equation (18) formulates the Focal loss, which can be also seen on Figure 14.

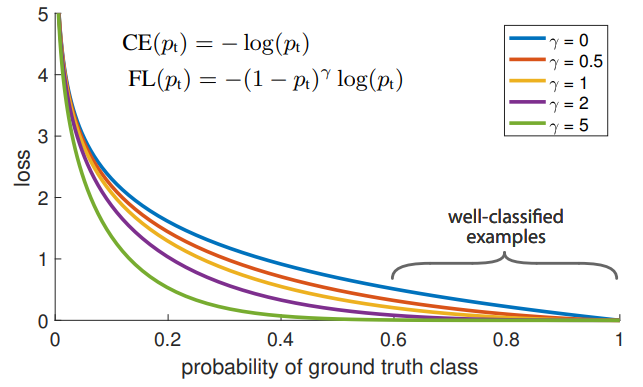


Figure 14: Shape of the Focal loss function for different *γ parameters* [17].

|  |  |
| --- | --- |
|  | (18) |

## **3.5. Log-Cosh Dice loss**

Jadon et al. [15] proposed a new loss function called the Log-Cosh Dice loss function, which aims to preserve the featured properties of the Dice loss function while it also eliminates its non-convex nature, in order to facilitate the convergence more easily toward optimal results. Figure 15 shows the shape of the function and how it is computed.

Diagram

Description automatically generatedThe computation of the derivation is straightforward, as Equation (19) - (20) show.

Figure 15: Coshx function, which is the average of and [15].

|  |  |
| --- | --- |
|  | (19) |
|  | (20) |

The logarithm of the function is taken to better constrain its infinite range [15]. The final formulation of the loss function is described by Equation (21).

|  |  |
| --- | --- |
|  | (21) |

# **4. Metrics**

Metrics are functions, which take the predictions of a machine learning model and the true labels as input, and they produce a value which describes the performance of the model from different statistical aspects. Several metrics that are applied for classification problems to evaluate the performance of a model can also be applied for segmentation tasks, as segmentation can be regarded as a pixel-level classification problem. True Positive elements are those that a machine learning model correctly classifies as positive values when the true labels are also positive. False Positive predictions are those that a model classified as positives, but the true labels are negative. The same logic is applied to determine the number of True Negatives and False Negatives. In the following, segmentation metrics are introduced. The output range of these metrics spans the interval of [0, 1], inclusive.

## **4.1. Accuracy**

Accuracy computes the ratio of correctly classified examples over the number of predictions. It is computed as the fraction of the sum of the True Positive and True Negative values divided by the sum of the True Positive, True Negative, False Positive and False Negatives, as Equation (22) describes it.

|  |  |
| --- | --- |
|  | (22) |

Accuracy can be a very misleading metric when the data set is imbalanced, as the majority classes can have a higher influence on the model, and they can dominate its performance.

## **4.2. Precision**

Precision is a widely used metric to determine the ratio of correctly classified positive examples over the amount of positively predicted ones. Precision is computed as the fraction of True Positive predictions divided by the sum of all predicted positives [18]. Equation (23) formulates the computation of Precision.

|  |  |
| --- | --- |
|  | (23) |

## **4.3. Recall**

Recall is usually computed hand in hand with Precision to characterize the performance of a machine learning model on a classification problem. Recall answers the question of how many of the true positive examples are found by the model? It is computed as the fraction of True Positives divided by the sum of the True Positives and False Negatives [18], as Equation (24) shows.

|  |  |
| --- | --- |
|  | (24) |

Recall can be optimized when the data set is highly imbalanced and one of the classes constitutes the bulk of the examples, why the main goal is to correctly classify the minority class most of the time.

## **4.4. Intersection over Union**

Intersection over Union (IoU) measures the area of overlap between the predictions and the true labels. IoU is also called Jaccard-Index. It is a commonly used metric to evaluate a model for objection detection and segmentation tasks, as it measures how much the predicted area, and the target area overlap. A perfect overlap indicates an IoU value of 1. IoU is computed as the ratio of the area of intersection divided by the union of the two areas, as Equation (25) describes.

|  |  |
| --- | --- |
|  | (25) |

# **5. Architectures for segmentation**

Image segmentation is a per pixel classification process. There are two subsets of image segmentation, namely semantic and instance segmentation. Semantic segmentation classifies each pixel to a certain class. Instance segmentation uniquely differentiates between each object within the classes. In the following sections, two neural network architectures are introduced for image and volume segmentation.

## **5.1. U-Net for image segmentation**

The U-Net architecture was developed for biomedical image segmentation [7]. The goal of the researchers was to develop a network and a training framework which can tackle the lack of annotated training samples, therefore they relied heavily on data augmentation techniques.

Chart

Description automatically generatedData augmentations are transformations that increase the diversity of the training samples, but at the same time they preserve the realistic nature of the data. The latter property is especially important in the medical domain, where the transformed data should necessarily reflect attributes of real organs. Elastic deformations are the most common form of variation in tissues and by introducing these transformations to the training data, the network can learn invariance and robustness to them without the existence of such data in the annotated image corpus [7]. The U-Net architecture is a fully convolutional network that extends the results of [8]. Figure 16 presents the U-Net architecture which consists of an encoding path on the left side, a bottleneck layer at the bottom and a decoding path on the right side. The encoding path of the network provides down-sampling with the max pooling layers while at the same time, the number of feature channels are doubled at each level. The encoding path is also called contractive path. The decoding path or expansive path increases the resolution of the down-sampled image by up-sampling, which reduces the number of feature maps at each level by half. The high-resolution feature maps from the same level of the contractive path are concatenated together with the up-sampled feature maps. The information propagation from the contracting path to the expansive path serves localization purposes. The information flow is realized through skip-connections [9]. The number of feature maps at the end of the expansive path is reduced through 1x1 convolutions [10]. The output segmentation maps of the network have the same resolution as the input image, and they consist of as many channels as the number of classes are. The architecture design of U-Net yields a roughly symmetric architecture [7], in which case the skip connections are called long skip connections. Long skip connections are suited to recover fine-grained details and to predict dense estimations, which are common requirements in the medical domain.

Figure 16: U-Net architecture [7].

## **5.2. 3D U-Net for volumetric segmentation**

The U-Net architecture was introduced by Ronneberger et al. [7] and it was extended by Çiçek, Özgün, et al. [11] to generalize from 2D image segmentation to 3D volumetric segmentation. As most of the medical images consist of several slices, which build up a volume, the 3D spatial context must be understood to provide accurate 3D segmentations. The 3D U-Net proposed by Çiçek, Özgün, et al. [11] replaces the 2D convolutional layers to their 3D counterparts. The network takes a 3D volume as its input, which is passed through its 3D convolutional, max pooling and up-sampling layers. Figure 17 shows the resulting architecture. Batch normalization layers [12] were also introduced before each ReLU activation function in the convolutional blocks to ensure faster convergence. Batch normalization layers normalize the data in each batch with respect to the mean and standard deviation of the batch. Global statistics are updated throughout the training and are used for Diagram

Description automatically generatednormalization during test time.

Figure 17: 3D U-Net architecture [8]

# **6. Frameworks for self-supervised representation learning**

Supervised learning utilizes labeled data to train a parametric model. Unsupervised learning methods aim to bypass costly labeling that require human work and build upon techniques that exploit different labeling methods that are freely available within the data. Pretext tasks generally synthesize labels from the unlabeled data that can be used to train the network, which can consequently learn general spatial features. The general features can be used to optimize the network on a target task using a small fraction of its original amount of label requirement. The features obtained during self-supervised training can be transferred to a target task, like classification and object detection with comparable or even better performance, compared to the purely supervised training paradigm. Algorithms that fall into the category of self-supervised learning are also called representation or feature learning algorithms. Representation learning algorithms extract intermediate representations from the raw data itself, to later build upon the concept of transfer learning, to solve general machine learning problems, like classification and object detection, by repurposing the learned features. Transfer learning follows the scheme of pre-training and fine-tuning, which can be directly translated to solve the pretext task and then fine-tune the model on the target task during self-supervised learning. In the following sections, two self-supervised learning techniques are introduced, which are commonly used in medical imaging [35].

## **6.1. Visual representation learning from unlabeled data by Context Prediction**

Embeddings or feature vectors for text or an image can be learnt through unsupervised representation learning. In the embedding or feature space the semantically similar data points should be closer, while the semantically different data points should be further away from each other. Doersch et al. [2] explored the utilization of spatial context to extract free labels from unlabeled data, in order to learn visual representations in a self-supervised manner. The idea to explore the information content of the spatial context of images arose from the study of Mikolov et al. [6], where the Skip-gram architecture was introduced. The Skip-gram architecture was originally developed to create word embeddings by mapping sparse one-hot encoded word vectors to a dense sub-space, through a process in which the network’s task is to predict for each word its probable context (n preceding and n succeeding words).The Skip-gram architecture and its training framework convert an unsupervised task to a self-supervised one by forcing the model to learn good word embeddings, meaning that semantically similar words are mapped closer to each other, with respect to a chosen vector distance metric. Doersch et al. [2] builds upon the context exploitation idea of Mikolov et al. [6], as Figure 18 shows. A grid of 3x3 patches is defined on an image with a random sampled offset from a given range, which defines the starting coordinates of the grid.

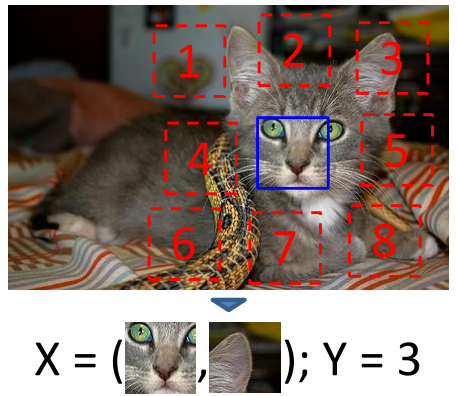


Figure 18: Recognition of a relative patch location with respect to a central patch, as a self-supervised pretext task [2].

Gaps are introduced between the patches and the patches are also randomly shifted with respect to each other, but both parameters are controlled through predefined ranges. The location of the center patch is indicated with blue color on Figure 18. The center patch always corresponds to the same relative position. A random patch is selected from the remaining eight positions (showed as red dashed lines on Figure 18), and without any further information, the task of the model is to correctly predict the relative location of the random patch with respect to the center patch. As illustrated on Figure 19 the network is fed with the central and the random patch in this order, and the network’s task is to assign a probability to each of the eight spatial configurations. Doersch et al. [2] uses the AlexNet [4] architecture for the visual context prediction Diagram

Description automatically generated with low confidenceproblem, as it is shown on Figure 19.

Figure 19: AlexNet style architecture, where the dashed lines represent shared weights [2].

The network processes each patch separately one at a time until the *fc6* layer, then it fuses the two embeddings before passing the fused information through the *fc6-fc9* layers. The architecture that is shown on Figure 19 is misleading in a sense that the subnetwork that processes the patches separately uses the same layers with shared weights, as it is indicated by the dashed lines. The parentheses in each block indicate the size of the kernels, the number of output filters and the stride of the convolution.

In the experiments conducted by Doersch et al. [2], researchers realized that the design of the pretext task greatly influences the quality of the extracted features and the model’s subsequent performance on the target task. Researchers discovered that the model was able to find shortcuts, which were used to exploit the low-level statistics of the data to solve the pretext task. However, these shortcuts prevented the model to generalize well on the target task. Shortcuts included boundary patterns, like edge continuity and similar textures between the patches. Therefore, random gaps between the patches and random shifts in each direction between the relative positions of the patches were introduced. Afterall, a trivial existing solution was discovered that the network could exploit. Chromatic aberration is the side effect of the camera lens focusing behavior at different wavelengths. Certain cameras usually shrunk the green channel relative to the others towards the center of the image. Consequently, the network learns to localize the center patch relative to the camera lens, after which the prediction of the relative position becomes trivial. Several techniques were introduced that tried to eliminate the existence of these shortcuts, which can be further studied in the research of Doersch et al.[2].

Consecutive research like the work of Noroozi et al. [1] presented the ambiguities that may arise from the relative patch location pretext task and its adverse effect on the network’s performance. However, Doersch et al. [2] still introduced state of the art results compared to previous research.

## **6.2. Solving jigsaw puzzles as a pretext task**

Noroozi et al. [1] introduced the context-free network (CFN), a siamese-ennead CNN, which was trained to solve jigsaw puzzles as its pretext task. Afterwards, the trained model was fine-tuned to solve general object detection tasks. Image tiles were fed to the network one tile at a time to explicitly limit the receptive field of its early processing units. The network was reportedly able to learn the mapping of object parts to their spatial arrangement. The jigsaw puzzle reassembly problem aimed to prove that much more information can be extracted from single snapshots of data, than it was previously achieved by supervised learning methods. Noroozi et al. [1] stated that the association of each separate puzzle tile to a precise object part can be ambiguous, but these ambiguities can be eliminated when all tiles are observed at once, due to their mutual exclusiveness. Training the network to solve jigsaw puzzles reduced their training time on the same data set to 2.5 days from 4 weeks compared to previous methods, and it also eliminated the problem of chromatic aberration [2]. Self-supervised learning algorithms exploit the free labels within the data that comes with the data itself. The network that solves the jigsaw puzzle obtains the labels from the structures of the data by exploiting the pixel arrangements. Noroozi et al. [1] also indicated that the color and texture similarity between the tiles do not help the localization of the tiles, therefore the features learned by solving the jigsaw puzzle also apply to different instances corresponding to the same class, as the network will learn features that are invariant to shared patterns. Noroozi et al. [1] built upon previous studies like the work of Richardson et al. [3], which presented how the visuospatial processing capabilities of humans can be assessed by jigsaw puzzles. The goal of Noroozi et al. [1] was to develop visuospatial representations of objects in CNNs, while at the same time avoiding cues or so-called shortcuts that were used by previous algorithms and humans as well, like the shape of the tiles or texture similarity at the border of the tiles. Shortcuts are not useful to learn general features as they solve the problem based on low-level statistics instead of the complete understanding of the global context. Researchers intentionally designed the CFN to compute features separately for each tile in the CNN part of the network, then observe the arrangements of the parts by concatenating the extracted features and passing them through the dense layers of the network [1]. The design forces the network to learn both representative and discriminative features for each part of the given object, in order to determine their relative location to each other. The CFN is presented on Figure 20. Noroozi et al. [1] originally used the AlexNet architecture (described in Krizhevsky et al. [4]) up to the fc6 layer with shared weights. As Figure 20. describes, the tiles are extracted from the 3x3 grid. Then, the tiles are reordered according to a randomly chosen permutation from a pre-computed set of permutations. Subsequently, each tile is passed through the network one tile at a time until the *fc6* layer using shared weights, meaning the same weights are used for the computation of the low-level features. This process is exactly the reason why Noroozi et al. [1] called the network context-free network, because they separated the data flow of the patches until the fully connected layers, which handled the contextual arrangement of the patches.



Figure 20: Jigsaw puzzle generation and the Context Free network [1].

As Figure 20 shows, a 3x3 grid is defined on the image. The starting coordinate of the grid is defined by the offset, which is relative to the left border, and it is randomly chosen from a predefined range. The size of the tiles is initially 75x75 pixels, which are reduced to 64x64 pixels by randomly choosing an offset within the initial tile, to reduce the size of the tile to the target size. This procedure creates a gap between the tiles, while they are randomly shifted relative to each other, which aims to prevent shortcuts that may emerge from texture similarities near the borders. Shortcuts prevent the network to learn general features and instead the network learns features which are suited to solve the pretext task, but not the target task. Shortcuts can emerge when the network is only fed with one jigsaw puzzle permutation per an input image, in which case the network could learn the absolute position of a tile and not the relative position to the other tiles. Shortcuts can also emerge when the network exploits low-level statistics such as edge-continuity, the pixel intensity and color distribution, or chromatic aberration [1]. The tiles are then reordered according to a certain permutation, which is randomly chosen from a pre-computed set of permutations. A 3x3 grid of patches indicates 9! = 362,880 possible permutations that can be used to rearrange the patches. Noroozi et al. [1] explicitly stated that choosing the right set of permutations to train the network highly influences the generalization capabilities of the network after training. The set of permutations is precomputed by maximizing the average Hamming distance between the permutations, which ensures that the tiles are shuffled sufficiently, and each tile of the same image has a chance to be assigned to different positions during the training epochs. Noroozi et al. [1] empirically proved that the ambiguity of the task was reduced, while the performance of the model increased by increasing the cardinality of the set of permutations up to a certain point and increasing the average Hamming distance between the permutations. They used a subset of 1000 permutations by selecting permutations based on increasing average Hamming distance between permutations. It was observed that the increase in the average Hamming distance between the permutations correlates with increasing model performance both on the pretext and the target task. For further details about the greedy algorithm that generates permutations, one can refer to the research of Noroozi et al. [1]. It was also concluded from the experiments that a good self-supervised task is neither simple nor ambiguous. The gap between the tiles discourages the CFN to learn low-level statistics such as edge continuities. The gaps may range from 0 to a certain number of pixels, chosen randomly from a range when the grid definition happens.

Adjacent tiles or patches may include similar low-level statistics such as mean and standard deviation of pixel intensities, therefore Noroozi et al. [1] normalized each patch by computing the mean and standard deviations for each patch independently. To avoid shortcuts that may arise from edge continuities, Noroozi et al. [1] introduced the gaps and random shifts between the patches.

According to Noroozi et al. [1] the output of the CFN represents a conditional probability density function of the spatial arrangement of object parts, represented by Equation (26)

|  |  |
| --- | --- |
|  | (26) |

, where *S* is the configuration of the tiles, *Ai* is the *i-*th part appearance of the object and *{Fi}* *i = 1, …, 9* forms the intermediate feature representation. The objective of the CFN is to evolve such *Fi* features that have semantic attributes capable to detect relative positions of object parts to each other. Noroozi et al. [1] also built upon the research of Yosinski, Jason, et al. [5], by showing that as one goes deeper in the convolutional layers, the layers produce task specific features, while the initial layers are general purpose. They showed that the last convolutional layers are specialized on solving the jigsaw puzzle and they achieve better performance by training previous convolutional layers up to a certain layer. Although a subset of the layers that were trained on the pretext task were also optimized during the training on the target task, the training time is still a fraction of most of the supervised learning methods.

# **7. Experiments**

## **7.1. Labeling**

The data set that was obtained from Mediso-ELTE Laboratory [[5]](#footnote-5), [[6]](#footnote-6) contained 102 patients, who were admitted to hospital examination to investigate their underlying heart condition based on SPECT imaging. The SPECT volumes and other metadata that described the condition of the patients were stored in DICOM files. The data set did not contain any labels, so I randomly selected 10 patients to manually annotate the SPECT volumes for fine-tuning after self-supervised training. To annotate the volumes, I used an open-source software called 3D Slicer [32], which is widely used in medical research. A built-in tool called “Grow from Seeds” in 3D Slicer was used to annotate the unlabeled volumes. Grow from Seeds requires two segments to be defined on an image. One is to determine the area of interest and the other is to define the region of pixels that belong to the background. From the two segments, a binary mask (or multi-label mask with more segments) can be exported to different file formats. Figure 21 illustrates the process of annotation. The green colored pixels are seeds that indicate the area of interest. The second segment is indicated by the yellow pixels, which must demarcate the pixels that belong to the background. In order to create a mask for both the volume of interest and the background, the two segments do not have to be defined precisely on every slice of the SPECT volume, because the algorithm is capable to incrementally expand the seeds based on the pixel intensities.

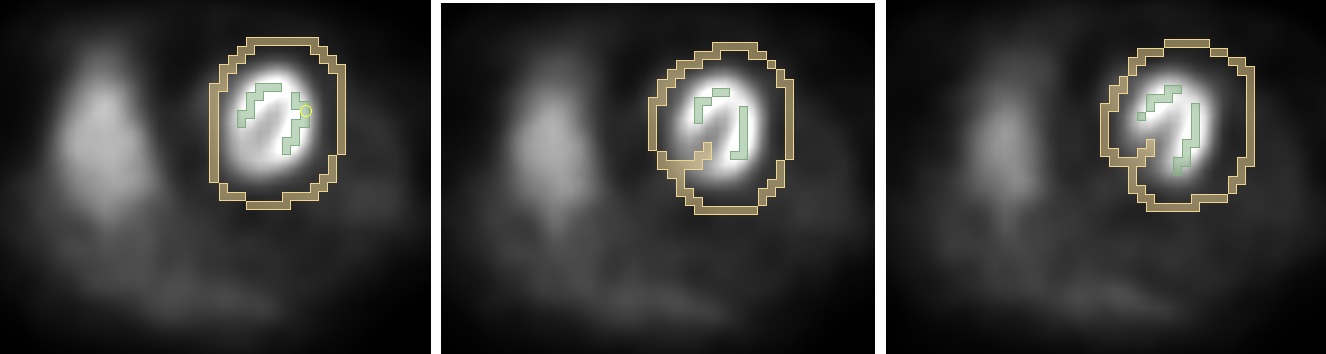


Figure 21: 2 layers of segments in 3D Slicer.

From the area of interest, the Grow from Seeds algorithm expands the area on the slices. Note that the whole SPECT volume can contain even 128 slices, but as Figure 22 shows, it is usually sufficient to define the seeds on a few slices. The resulting 3D mask for the left ventricle is illustrated on Figure 22 from 3 different angles.

A picture containing schematic

Description automatically generated

Figure 22: segmented 3D left ventricle after the application of grow from seeds in 3D slicer.

Specialist could argue that the obtained volume segmentation as annotation for training is far from perfect. For this reason, Ádám István Szűcs, who has plenty of experience in the medical domain, supervised every annotation, and where the results were not satisfiable, I manually corrected the segmentation on every slice.

## **7.2. Exploratory Data Analysis**

As it was previously mentioned, the obtained data set contained 102 patients. The data was stored in DICOM files. There were 61, 39, 1 and 1 volumes where the width and height dimensions had values of 64, 128, 52 and 81, respectively. The minimum and maximum number of slices in the volumes were 17 and 128, while the mean number of slices was roughly 56. Regarding the annotated volumes, the positive slices are where there is at least one pixel that belongs to the target class (which is the left ventricle) and not to the background. The ratio of positive slices was 26.5% on average, while the minimum and maximum values were 12.5% and 64.7%, respectively. The imbalance of the positive pixels compared to the total number of pixels in the volume was much larger. The ratio of positive pixels was 0.7% on average, while the minimum and maximum values were 0.1% and 2%, respectively. The positive slice and pixel ratios clearly demonstrated that the training procedure should be designed to handle extreme class imbalance.

Data collection in hospitals usually produces DICOM files that contain a lot of noise. Medical images are constructed based on different parameter settings and by using a diverse set of medical imaging devices. Indeed, this was the case with the obtained data set as well, because at least the order of the axes that correspond to the channel, height, width and depth dimensions differed for several volumes. For this reason, the slices of the volumes were displayed, and the axes were manually reordered, where it was necessary. The standardized data set was saved to have an axes order of channels, width, height and depth, in this order.

## **7.3. Preprocessing**

The preprocessing stage could be separated into offline and online phases. During the offline phase, the data is preprocessed before the training of the neural network begins. Online data preprocessing takes place during the training, typically in a randomized way for each training sample.

In the offline phase, each volume was cropped by detecting the minimum and maximum values along the axes of width and height, where the pixels were greater than zero. Each cropped volume was then resized to have a shape of 128x128 along the width and height dimensions using linear interpolation. This naïve procedure successfully removed some rows or columns in a few cases, but due to noise, the overall impact was negligible. On the left side of Figure 23, there is an example SPECT volume and on the right side of Figure 23, there is the corresponding mask that demarcates the left ventricle. It is visible that on every slice, there is a relatively large number of pixels with apparently zero values that correspond to empty space and therefore these areas do not contain useful information. The number of slices where the left ventricle is presented is relatively small, compared to the total number of slices.

The online preprocessing phase involved the extensive usage of augmentations that was provided by the TorchIO library [34]. TorchIO is an open-source Python library that was built on top of Pytorch [33]. It provides efficient data loading, preprocessing and augmentations for 3D medical images. Augmentations improve the model’s robustness by increasing the size of the training data set through the generation of new realistic images.

Graphical user interface

Description automatically generated with medium confidence

Figure 23: An Example SPECT Volume (left) and the corresponding Mask (right) with the left ventricle.

The exact description of the preprocessing pipeline is not explained in this thesis, but it is provided with the code repository. Both for self-supervised learning and for supervised learning fine-tuning, standardization or Z-normalization was applied. For the relative patch location and jigsaw puzzle pretext tasks, each 3D patch was normalized separately based on the statistics of each patch volume before concatenating them together, as it is described in [1], [2]. For supervised training and supervised fine-tuning each 3D volume was normalized based on batch statistics. Cropping and padding were applied on every volume by specifying the maximum depth as 32 slices. TorchIO provides the “CropOrPad” method that either crops or pads the volume, depending on the number of slices. When a target mask is provided, the cropping happens by taking into consideration the matching indices of the input data and the positive mask slices. Mirroring across all the width, height and depth dimensions was applied through the “RandomFlip” method. Additional augmentations that were used are random blur and Gaussian noise, random affine and elastic transformations with different parameter combinations. The augmentation pipelines were further randomized and complicated with certain probabilities given to each method to be applied, and the pipelines were further separated to subparts. From each of these subparts, always one of the included methods was selected and applied, again based on assigned probabilities. Figure 24 TorchIO augmentations. Reference image with grid (left), random affine transformation (center) and random elastic transformation (right). Figure 24 visualizes a reference image on the left, for which a grid was plotted to better illustrate the effects of augmentations. At the center of Figure 24, there is an example for the application of the “RandomAffine” transformation, while the right of Figure 24 is an example for “RandomElasticDeformation”. The augmented images can be regarded as new training samples and the realistic nature of the original images is preserved.

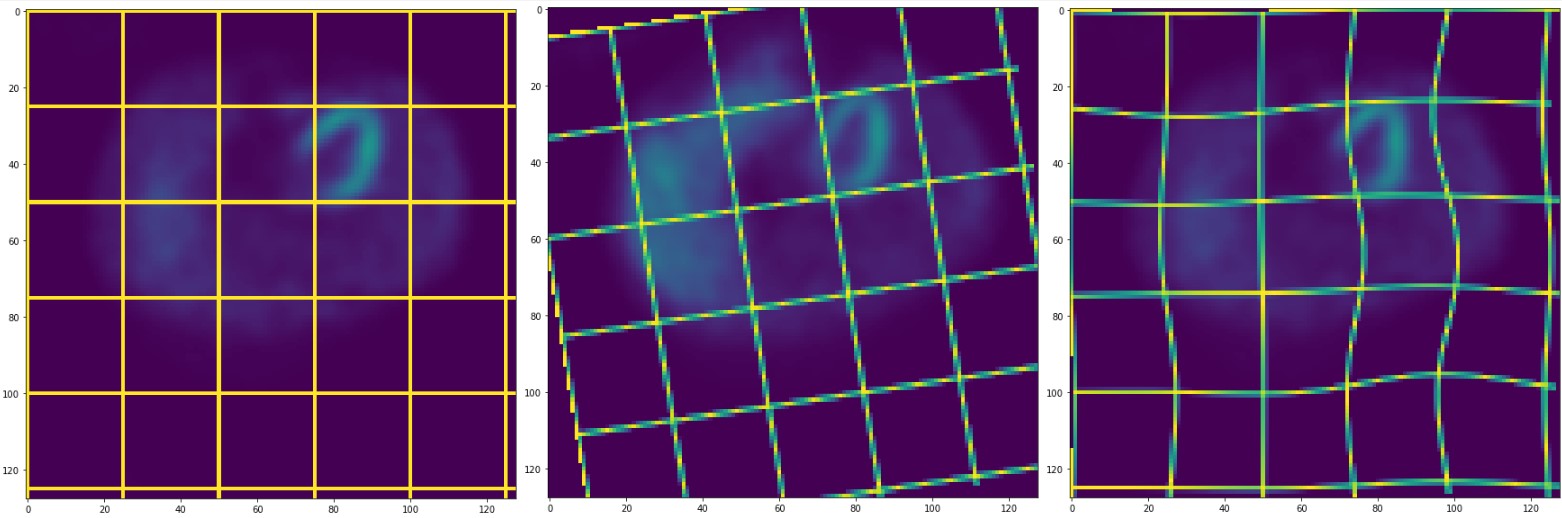


Figure 24: TorchIO augmentations. Reference image with grid (left), random affine transformation (center) and random elastic transformation (right).

Figure 25 presents an example for the output of the whole augmentation pipeline. The 3D volume was cropped to 32 slices and a subset of the previously described augmentations were applied to produce a new, realistic training sample. When a mask is provided as input with the 3D volume, it is guaranteed that the same augmentations are applied for both. Although augmentations can be used during validation and test-time, in this work they are only applied during training.

A screenshot of a computer

Description automatically generated with low confidence

Figure 25: Application of the augmentation pipeline on the input data (left) and target mask (right).

## **7.4. Representation learning from context prediction**

The unlabeled SPECT volumes that corresponded to 92 patients were utilized for unsupervised representation learning. The methodology that was described in the research of Doersch et al. [2] was closely followed.

In every epoch, for each training and validation SPECT volume a grid was defined. A random offset was chosen from the interval of [0, 16], which defined the starting coordinates of the grid along the axes of width and height. The shape of the patches was configured to be 28x28x32 along the width, height and depth dimensions. The randomized grid defined the location of the 3x3 patches, which were extracted from the volume. The reference or center patch was always located at the middle of the grid, and a patch was chosen randomly from the remaining 8 positions. Figure 26 demonstrates the procedure of grid definition on the left and the extraction of the center and random patch on right, at positions (2, 2) and (3, 1), respectively. Although Figure 26 visualizes only a slice of the given SPECT volume, in Shape, square

Description automatically generatedreality 3D volumes were extracted, which are referenced as 3D patches or simply patches in the following sections. The patches were normalized independently of each other, as it was described by Doersch et al. [2]. The network’s task was to correctly identify the location of the random patch, with respect to the center patch. As the study of Doersch et al. [2] demonstrated, the context prediction pretext task facilitates for the network to learn useful spatial features, because the network is forced to recognize the relative position of spatial objects parts. Noroozi et al. [1] argued that the prediction of the relative patch location can be ambiguous, since only a subset of the object parts is fed to the network. The relative locations of the object parts are mutually exclusive only when they are observed at the same time. The task can be ambiguous as well when observing the SPECT volumes. The left ventricle appears partly at the position of the center patch on Figure 26 and its shape closely resembles a horseshoe. However, the location of the left ventricle on SPECT volumes is not constant. The training data consisted of many SPECT volumes where the position of the left ventricle appeared around the 4 corners. The applied “RandomFlip” augmentation could also mirror the volumes across each of the spatial axis.

Figure 26: Context prediction pretext task. Grid definition (left) and extraction of the center and random 3D patch (right).

The extraction of the 3D patches did not necessarily cover the left ventricle. The appearance of the left ventricle covered 15 slices on average, while the average number of slices in the SPECT volumes was 56. However, the goal of the pretext task was not to precisely localize the left ventricle, but to recognize the shape of the organs that appear on the SPECT images.

The experiments that were conducted utilized a 3D U-Net architecture with 3 encoding blocks, 8 output kernels at the first encoding block (which is doubled at every level downward), Instance Normalization layers, PReLU activation functions, and different levels of dropout were applied. As the pretext task was formulated as a classification task and the decoder of the 3D U-Net architecture was used for segmentation, the decoding block of the 3D U-Net architecture was replaced by 2 Dense layers with PReLU activation functions. 1D Batch Normalization was applied in the first Dense layer before the activation function. The model was configured with a so called “Reduction Factor” parameter, which controlled the number of output neurons in the first Dense layer with respect to the number of input neurons, which depended on the encoder configuration of the 3D U-Net. The reduction factor of 64 meant that the number output neurons of the first Dense layer were 1/64 of the number of input neurons. The values of 32 and 64 were used as reduction factors for most of the experiments that produced the best results, in which cases the models consisted of 5,098,282 and 2,522,940 trainable parameters, respectively. For the training procedure the AdamW optimizer from PyTorch was used with a learning rate of 1e-3 and default parameter settings. The AdamW algorithm modifies the Adam stochastic optimization method, by decoupling the weight decay from the gradient update [36]. The AdamW optimizer operated on the Cross-Entropy loss function during the experiments.

Figure 27 presents the results of the experiments, where each model was configured with a reduction factor of 32 and with different dropout rates. It is visible that regarding the validation results, the model with a dropout rate of 0.0 was the first that achieved a validation accuracy of 1.0. However, a few epochs later the model with a dropout rate of 0.5 could also reach the same performance on the validation set. It is also visible on Figure 27 that while the convergence of the training loss and accuracy is relatively stable, the validation loss and accuracy oscillate by a huge amount. The validation accuracy collapsed several times by 30-40% due to weight updates. The weights of each model were saved whenever the accuracy reached the value of 1.0 on the validation set, to utilize the model on the target task later.

**Chart

Description automatically generated with medium confidence**

Figure 27: Experiment results with reduction factor of 32 and dropout rates of 0.0, 0.25 and 0.5.

The number of parameters of each model was reduced by half with a reduction factor of 64 and comparable results were obtained, as Figure 28 demonstrates it. Both 3 models with different dropout rates were able to achieve the accuracy score of 1.0, while the oscillation of the validation results was also observable.

A screenshot of a computer

Description automatically generated with low confidence

Figure 28: Experiment results with reduction factor of 64 and dropout rates of 0.0, 0.25 and 0.5.

## **7.5. Representation learning from solving jigsaw puzzles of SPECT volumes**

The training and validation sets were consisted of the same unlabeled SPECT volumes that corresponded to the 92 patients, as it was described previously. The methodology that was described by Noroozi et al. [1] was closely followed for training on the jigsaw puzzle pretext task during self-supervised learning.

Permutation sets of size 100, 500 and 1000 were precomputed by maximizing the average Hamming distance between the permutations. The permutation sets of size 100 and 500 proved to be quickly learnable by the networks. For this reason, based on the set of 1000 permutations experiment results are summarized, which was harder to learn for the networks.

Figure 29 demonstrates an example slice of a SPECT volume that contains the horseshoe shaped left ventricle around the top right part of the image. A grid of 3x3 patches is randomly placed based on the specified parameters. The size of each patch was defined as 28x28 pixels along the width and height dimensions. The gap between the patches was defined as 14 pixels. The offset parameter, which defined where the grid starting position is located was randomly chosen from the interval of [0, 16], separately for each dimension. The volumes that were demarcated by the patches were extracted from each of the 3x3 positions, as it is illustrated on the middle image on Figure 29. Although Figure 29 illustrates only one slice for each patch, 3D volumes were extracted. Then, the volumes were reordered based on a randomly selected permutation from the 1000 precomputed permutations.

A picture containing background pattern

Description automatically generated

Figure 29: Generating jigsaw puzzle from SPECT volume. A Grid of 3x3 patches is defined and its location is randomized (left), patch extraction (center) and permutation of the patches (right).

The procedure of the jigsaw puzzle creation was uniquely performed for each training sample in each training epoch. The augmentation pipeline was applied before the jigsaw puzzle definition. The extracted volumes of patches were then normalized independently based on the patch statistics. The 3D U-Net architecture was utilized with 3 encoding blocks and with 8 output kernels in the first encoding block. The decoder of the 3D U-Net architecture was replaced for the same 2 Dense layers as previously, with PReLU activation functions and with Batch Normalization in the first Dense layer before the activation function. The training procedure always relied upon the AdamW optimizer [36] with a learning rate of 1e-3, while the other parameters of the optimizer were set to their default values. Figure 30 presents the result of the experiments that tested the effect of the reduction factor, by values of 16, 32, and 64, in which cases the models consisted of 10,479,462, 5,098,282 and 2,522,940 trainable parameters. It appears on Figure 30 that each model achieved the accuracy score of 1.0 on the validation set. The performance of the models oscillated by a relatively large margin in terms of accuracy, but whenever the perfect accuracy score was reached, the weights of each model were saved to utilize the trained model on the target task later. Since every model could reach comparable performance sooner or later, the reduction factor of 64 was selected for the following experiments.

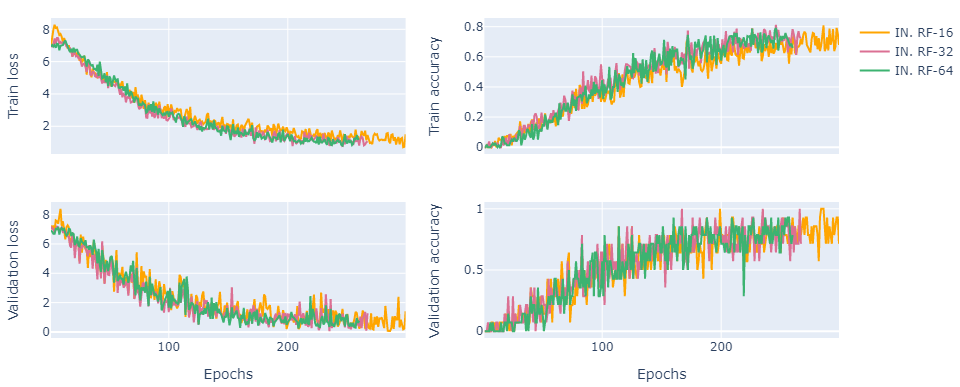


Figure 30: Experiments based on different reduction factors of 16, 32 and 64.

Figure 31 demonstrates how different normalization layers, such as Batch Normalization, Instance Normalization and Group Normalization layers affect the training of the networks. Batch Normalization proved to be the least effective in terms of training stability. The model with Batch Normalization layers could also reach the accuracy score of 1.0, but the performance collapsed several times. The most effective layers were Instance Normalization, and Group Normalization with 2 groups, which produced the most stable convergence results and reached the accuracy score of 1.0 for the first time. For the following experiments, Instance Normalization layer was applied.

Figure 32 presents the loss and accuracy curves for various dropout rates. The model with a dropout rate of 0.25 was the first that achieved a validation accuracy of 1.0, while the model with a dropout rate of 0.15 and 0.3 could also reach the same performance, as it is shown on Figure 32. However, as the dropout rate of 0.5 eliminates half of the connections for each neuron in the network, for this rate the network could not reach the perfect performance, as it is presented on Figure 32.

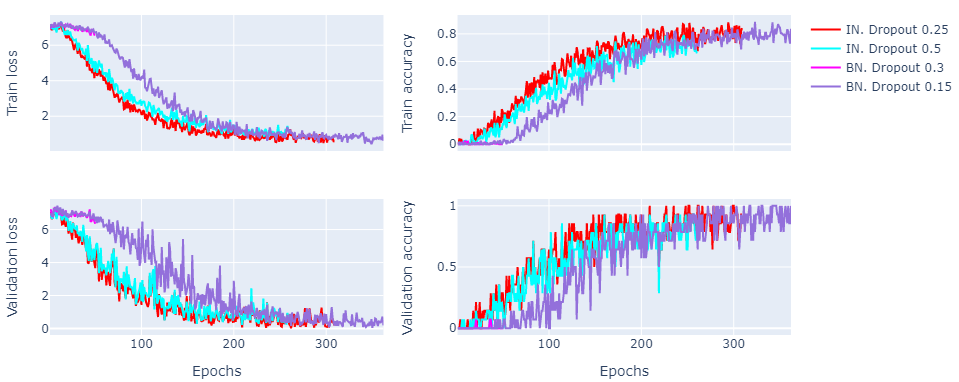


Figure 32: Experiment results based on various dropout rates

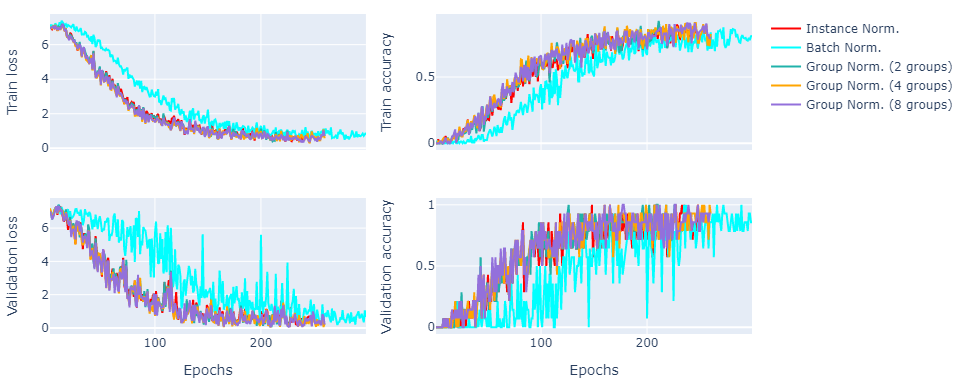


Figure 31: Effect of normalization layers on the training convergence towards optimal results.

To demonstrate the results of additional experiments, Table 1 describes the name of the experiments with the parameter configuration of each model and other properties. The number of encoding blocks was increased to 4 from 3, while the number of output channels (filters) of the 1st encoding block (which doubles at every level downward) was also increased to 16 and 64, from 8.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Experiment name | Num. encoder blocks | 1st layer output channels | Reduction factor | Normalization | Dropout | Num. parameters |
| exp-1 | 3 | 16 | 8 | Instance | 0.0 | 3,445,900 |
| exp-2 | 3 | 16 | 8 | Instance | 0.25 | 3,445,900 |
| exp-3 | 3 | 16 | 8 | Instance | 0.5 | 3,445,900 |
| exp-4 | 3 | 64 | 32 | Instance | 0.25 | 17,838,028 |
| exp-5 | 3 | 64 | 32 | Instance | 0.5 | 17,838,028 |

Table 1: Parameter configurations for the jigsaw puzzle pretext task.

Figure 33 summarizes the results that of the experiments that Table 1 presented. The models that had 64 and 16 output channels, with dropout rates of 0.25 and 0.0 were able to achieve the errorless validation accuracies at epochs of 173 and 215, respectively, as it can be seen on Figure 33. The other models were not able to reach the **Chart

Description automatically generated**same performance and they were struggling around the validation accuracy score of ~0.93.

Figure 33: Experiments based on various dropout rates.

## **7.6. Supervised fine-tuning**

The previously introduced models that were trained on the relative patch location and jigsaw puzzle pretext tasks were fine-tuned on the target task to test the underlying hypothesis of the thesis. The target task comprised of the randomly selected 10 patients, for whom the annotations were created with 3D Slicer. During supervised fine-tuning on the target task, the parameters of each of the pre-trained models that constituted the encoder part of the 3D U-Net were loaded to a new 3D U-Net, which adapted the same parameter configurations as the pre-trained models. The hypothesis of the thesis claims that it is possible to train a model in a self-supervised manner on SPECT volumes, which can be fine-tuned in a supervised way to provide accurate 3D segmentations for the left ventricle. In order to prove the hypothesis of the thesis, many of the following experiments relied upon the pre-trained encoder by freezing its weights during the fine-tuning, and training only the bottleneck layer and the decoder.

## **7.7. Supervised fine-tuning after the relative patch location pretext task**

The extracted features from the relative patch location pretext task did not prove to be robust, unfortunately. Many experiments were conducted, but only one is presented, because the jigsaw puzzle pretext task yielded more efficient feature representations, than the relative patch location pretext task. For supervised fine-tuning, the pre-trained model with 2 encoding blocks, 8 output channels, Instance Normalization layers, and with a dropout rate of 0.25 was utilized. For the training, the AdamW optimizer was used with a learning rate of 1e-3. The Dice loss function was the objective function of the model. The encoding blocks of the model were frozen, and only the bottleneck and the decoder layers were trained. The data set of the 10 patients was randomly split to 6 and 4 patients to comprise the training and validation sets, respectively. Figure 34 presents noisy Dice loss and IoU curves on the training set, and smoother Dice loss and IoU curves on the validation set. This was the result of the heavy augmentation pipeline that was applied on the training set. The maximum IoU score that the model could reach on the validation set was around ~0.25%, according to Figure 34, which is very low.

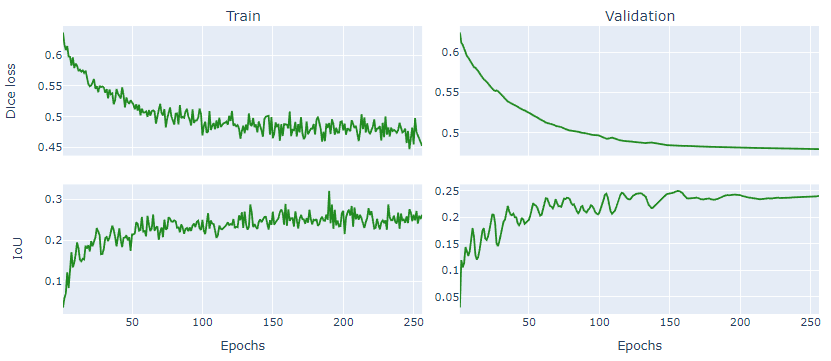


Figure 34: Results of the supervised fine-tuning after self-supervised training on the relative patch location pretext task.

When the model reached a higher IoU score on the validation set, its weights were saved. After a defined number of epochs, if the model could not reach a higher IoU score, than the all-time high, the training was stopped. Then, the saved weights were loaded to the model to perform the segmentation of the left ventricle on the validation set. On the left side of Figure 35, the ground truth masks, and on the right side, the predictions are presented for a certain patient from the validation set, for the given SPECT volume. With the frozen encoder, the model was not able to differentiate between the left ventricle and other organs that were presented on the input SPECT volume. It is reflected that a lot of false positive predictions appear on the right side of Figure 35, which were foreseeable from the low IoU score. Figure 36 demonstrates the predictions for a randomly selected slice (for the same input data), from the perspective of the 3 anatomical planes, where the same inaccurate segmentation is also A picture containing text

Description automatically generatedobservable.

Figure 35: Left ventricle segmentation. Ground truth mask (left) and prediction (right).

Although the model was able to extract the shapes of the organs from the SPECT volumes, the learned feature representations from the relative patch location pretext task were not sufficient to provide accurate segmentations for the left ventricle.

## **Graphical user interface, application Description automatically generated7. 8. Supervised fine-tuning after the jigsaw puzzle pretext task**

Figure 36: Predictions illustrated on a slice of the 3D SPECT volume in the anatomical space. Slice (LEFT), ground truth mask (center) and prediction/segmentation (RIGHT).

The jigsaw puzzle pretext task yielded much better feature representations compared to the relative patch location pretext task, as it was observed after each model was fine-tuned on the target task. Experiments that resulted in the highest IoU scores are presented in this section. The data set of 10 patients with left ventricle annotation was split as previously. The AdamW optimizer was applied with an initial learning rate of 0.01. The learning rate was reduced by 0.01 in each epoch, following an exponential learning rate scheduling. The parameter configuration of each experiment is described in Table 2. Each experiment is identified by a number for clarification. The pre-trained encoders with 2 or 3 encoder blocks, 16, 32 and 64 output channels, and with dropout rates of 0.0 or 0.25 were utilized.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Experiment name | Num. encoder blocks | 1st layer output channels | Dropout | Criterion | Num. encoder blocks frozen | Num. parameters | Gamma |
| exp-1 | 2 | 32 | 0.0 | Dice | 2/2 | 3,927,052 | - |
| exp-2 | 2 | 16 | 0.0 | Dice | 2/2 | 982,028 | - |
| exp-3 | 3 | 64 | 0.25 | Dice | 2/3 | 65,251,344 | - |
| exp-4 | 3 | 64 | 0.25 | Dice | 1/3 | 65,251,344 | - |
| exp-5 | 3 | 64 | 0.25 | Dice | 0/3 | 65,251,344 | - |
| exp-6 | 3 | 64 | 0.25 | Focal | 3/3 | 65,251,344 | 0.5 |
| exp-7 | 3 | 64 | 0.25 | Focal | 3/3 | 65,251,344 | 2.0 |
| exp-8 | 3 | 64 | 0.25 | Focal | 3/3 | 65,251,344 | 3.0 |
| exp-9 | 3 | 64 | 0.25 | Focal | 3/3 | 65,251,344 | 4.0 |
| exp-10 | 3 | 64 | 0.25 | Focal | 3/3 | 65,251,344 | 1.0 |
| exp-11 | 3 | 64 | 0.25 | LCD | 2/3 | 65,251,344 | - |
| exp-12 | 3 | 64 | 0.25 | LCD | 1/3 | 65,251,344 | - |
| exp-13 | 3 | 64 | 0.25 | LCD | 0/3 | 65,251,344 | - |

Table 2: Experiment configurations for the fine-tuning of the models that were pre-trained on the jigsaw puzzle pretext task.

During the training, the number of encoder blocks that were frozen changed from zero to the number of encoding blocks, to take conclusions about the usefulness of the learned feature representations. The models were optimized based on the Dice, Focal and Log-Cosh Dice (LCD) loss functions. Figure 37 visualizes the train and validation performance of each model that was trained with the Dice loss function. The models of the experiments “exp-4” and “exp-3” reached the best IoU scores on the validation set, around 0.678 and 0.662 at epoch 73 and 91, respectively. At these IoU scores, the Precision of each model was around 0.748 and 0.698, and the Recall score was around 0.885 and 0.889, respectively.

Chart

Description automatically generated with medium confidenceThe Precision scores of 0.748 and 0.698 indicate that the positively classified pixels by the models were 75% and 70% right, while 25% and 30% of the positively classified pixels were wrong (false positives). Both Recall scores were around 0.88, which indicate that the models found 88% of the pixels on average that corresponded to the left ventricle. Figure 38 demonstrates the predictions of the model of “exp-4”. Compared to the previous results, the model was able to filter out the irrelevant information and provide segmentation for the left ventricle, exclusively. It is observable that the predictions can be characterized by more realistic and smoother contours on the right side, than the ground truth masks at the center, on Figure 38. This is truly the case if one compares the predictions of the model to the contours of the left ventricle on the left side on Figure 38. The model was able to find the pixels that belong to the left ventricle by a much higher accuracy, than the annotation indicated. In this case, the model was forced to learn bad feature representations when it was trained on the target task and after it successfully learned general features on the pretext task. The number of fine-tuned encoder bocks for the model of “exp-4” was 1 and for the model of “exp-3” it was 3, which empirically proves the conclusions of Yosinski, Jason, et al. [5], that the last layers of the pre-trained network are task-specific, and it is usually beneficial to fine-tune those layers as well during transfer learning.

Figure 37: Train and validation performances of models that were trained with the Dice loss function.

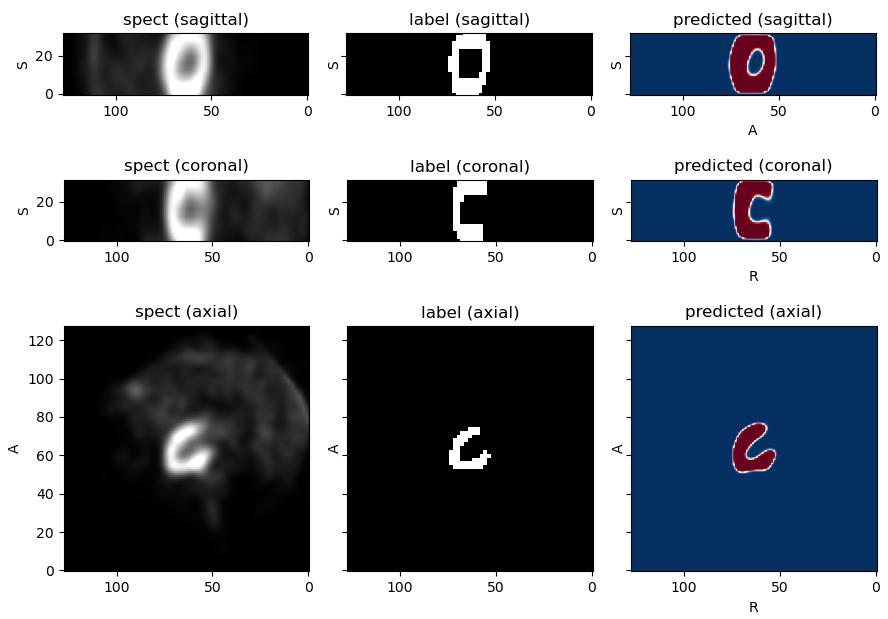


Figure 38: Left ventricle segmentation with the model of “exp-4”, illustrated from the 3 aspects of the anatomical space.

The effectiveness of the pre-training on the jigsaw puzzle pretext task is also observable by the experiment results described in Figure 37, and the parameter configurations demonstrated by Table 2. The models of “exp-2” and “exp-1” contained roughly 1% and 6% the number of trainable parameters, than the models of “exp-3” and “exp-4”, and their encoder blocks were fully frozen, while at the same time they both reached IoU scores of ~0.6. Figure 39 presents the predictions of the model of “exp-4” for the whole SPECT volume, for the same patient as on Figure 38. The left image represents the ground truth annotations, while the right image shows the segmentation of the model on Figure 39. With respect to the ground truth labels, the model detected the left ventricle one slice later, and finished the segmentation one slice earlier. The model successfully recognized the shape of the left ventricle, and the contours of its predictions were more realistic and accurate, than the annotations. At this point, I conclude that the performance of the model is highly restricted by the quality of the labels and there is a high probablity that much better results could be obtained.

A picture containing application

Description automatically generatedThe experiment results from “exp-6” to “exp-10” are summarized on Figure 40. Each model consisted of 3 encoding blocks, 64 output channels at the first encoding layer, and a dropout rate of 0.25 was applied. The encoding blocks were frozen for all experiments. Many experiments were conducted, where the encoding blocks were trained as well, but these did not result in better performance, therefore they are excluded, and only the best results are presented. The models roughly consisted of ~65M trainable parameters and each of them was trained with the Focal loss function, with different *γ* parameters. As it can be seen on Figure 40, both experiments resulted in comparable IoU scores, around ~0.6 - 62. The Precision and Recall scores oscillated between ~0.6 - 0.8 and ~0.8 - 0.98, respectively, towards the end of the training. The highest IoU score was obtained by the model of “exp-7”, which reached the IoU, Precision and Recall scores of 0.624, 0.658 and 0.925, respectively, at epoch 194. Regarding the overall conclusion of the experiments, the application of the Focal loss function did not result in higher overall IoU scores and better segmentation results, but it effectively reduced the number of false negatives and it improved on the Recall scores.

Figure 39: Segmentation of the left ventricle with the model of “exp-4”, for a given SPECT volume. Ground truth annotations (left) and predicted segmentation (right).

Figure 41Graphical user interface, application

Description automatically generated demonstrates the results of the experiments from “exp-11” to “exp-13”. The same configuration with respect to the model was applied as previously, in the case of the experiments that trained the models with the Focal loss function. Except, that the models of these experiments were trained with the LCD function, and the best results were obtained by training a varying subset of the encoding blocks. Each model was able to reach comparable performance at different phases of the training procedure. The model of “exp-13” reached the highest IoU score of 0.673, slightly below the best performing model of the Dice loss experiments. The Precision and Recall scores of 0.693 and 0.958 corresponded to the highest IoU score on the validation set.

Figure 40: Train and validation performances of models that were trained with the Focal loss function.

Chart

Description automatically generated

Figure 41: Train and validation performances of models that were trained with the LCD loss function.

The high Recall scores indicate that the LCD loss function also effectively removed the number of false negatives, so the models were able to find the pixels that corresponded to the left ventricle. The Precision scores also signify the importance of accurate annotations. The models classified several pixels as positives, while the ground truth labels were negatives. This phenomenon is attributable to the low quality of the labels, which directly has an impact on the IoU score as well. Figure 42 visualizes the segmentation of the left ventricle for 3 slices, from the perspective of the anatomical planes. The fragmented contours of the annotations are observable at the middle column of Figure 42. The same can be noted as previously, that the contours of the predictions are more realistic at the right column on Figure 42, compared to the annotations, while they also accurately demarcate the pixels of the left ventricle, that appears at the left column on Figure 42.

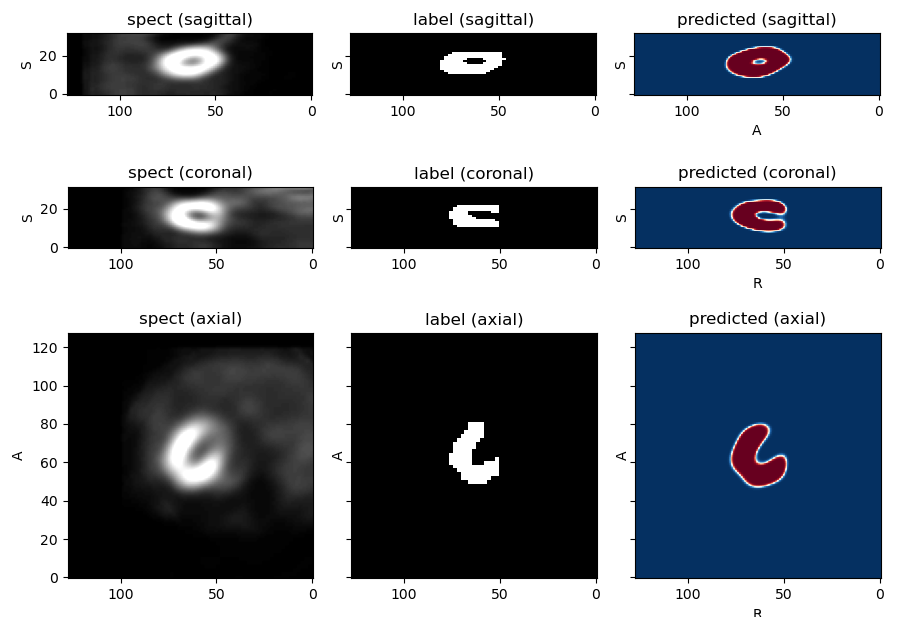


Figure 42: Left ventricle segmentation of the model of “exp-13” illustrated from the 3 aspects of the anatomical space.

# **8. Conclusion**

The extended encoders of 3D U-Net architectures were trained on the relative patch location pretext task could not achieve comparable results with those that were trained on the jigsaw puzzle pretext task. The jigsaw puzzle pretext task proved to be highly efficient to extract general feature representations from SPECT volumes during self-supervised learning. Many of the pre-trained models were fine-tuned in a supervised manner to provide accurate 3D segmentations for the left ventricle of the human heart. The quantitative comparison of each model’s performance with the ground truth labels was limited by the quality of the labels. Although, many of the models were able to reach sufficient quantitative and qualitative results, which were illustrated in anatomical coordinate systems. The hypothesis of the thesis was proved, as many general feature detectors were trained in a self-supervised manner, which then required only 6 patients to reach excellent results for left ventricle segmentation.

# **9. Future work**

The goal of the thesis was not to train highly optimized models, but to prove the underlying thesis hypothesis. For industrial purposes, it would be necessary to obtain high quality annotations from a human specialist. The segmentation pipeline could be extended with a model that could detect the slices of the SPECT volumes, where the left ventricle is present, to provide only the relevant information to the model that performs the segmentation. For 3D segmentations, other cutting-edge technologies could be integrated, like the U-Net Transformer of Hatamizadeh, Ali, et al. [38] and the Autoencoder of Myronenko, Andriy [39], as most of the best results in segmentation are obtained through a model of ensembles. For self-supervised pre-training, the pretext task of playing the Rubik’s cube could be explored, based on the work of Zhuang, Xinrui, et al. [40].

# **10. Acknowledgement**

I would like to express my gratitude to my supervisor, Ádám István Szűcs, who guided me throughout this project and offered me the opportunity to learn about self-supervised learning methods. I would like to thank to Johanna Barbara Orbán, who supported me all the way with all her honest efforts. I would like to thank to my parents, who encouraged me to earn a master’s degree.

# **11. References**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [1] | Noroozi, Mehdi, and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles." *European conference on computer vision*. Springer, Cham, 2016. | | | |
| [2] | Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. "Unsupervised visual representation learning by context prediction." *Proceedings of the IEEE international conference on computer vision*. 2015. | | | |
| [3] | Richardson, John TE, and Tomaso Vecchi. "A jigsaw-puzzle imagery task for assessing active visuospatial processes in old and young people." *Behavior Research Methods, Instruments, & Computers* 34.1 (2002): 69-82. | | | |
| [4] | Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012). | | | |
| [5] | Yosinski, Jason, et al. "How transferable are features in deep neural networks?." *Advances in neural information processing systems* 27 (2014). | | | |
| [6] | Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems* 26 (2013). | | | |
| [7] | Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and [8] computer-assisted intervention*. Springer, Cham, 2015. | | | |
| [8] | Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. | | | |
| [9] | He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. | | | |
| [10] | Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. | | | |
| [11] | Çiçek, Özgün, et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation." *International conference on medical image computing and computer-assisted intervention*. Springer, Cham, 2016. | | | |
| [12] | Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning*. PMLR, 2015. | | | |
| [13] | Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958. | | |
| [14] | Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012). | | |
| [15] | Jadon, Shruti. "A survey of loss functions for semantic segmentation." *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020. | | | |
| [16] | | | Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016. | |
| [17] | | | Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017. | |
| [18] | | | Grandini, Margherita, Enrico Bagli, and Giorgio Visani. "Metrics for multi-class classification: an overview." *arXiv preprint arXiv:2008.05756* (2020). | |
| [19] | | | Wu, Yuxin, and Kaiming He. "Group normalization." *Proceedings of the European conference on computer vision (ECCV)*. 2018. | |
| [20] | | | Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization." *arXiv preprint arXiv:1607.08022* (2016). | |
| [21] | | | Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems* 27 (2014). | |
| [22] | | Ulyanov, Dmitry, et al. "Texture networks: Feed-forward synthesis of textures and stylized images." *ICML*. Vol. 1. No. 2. 2016. | |
| [23] | | LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324. | | |
| [24] | | Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." *arXiv preprint arXiv:1607.06450* (2016). | | |
| [25] | | Li, Hao, et al. "Visualizing the loss landscape of neural nets." *Advances in neural information processing systems* 31 (2018). | | |
| [26] | | Araujo, et al., "Computing Receptive Fields of Convolutional Neural Networks", Distill, 2019. | | |
| [27] | | Luo, Wenjie, et al. "Understanding the effective receptive field in deep convolutional neural networks." *Advances in neural information processing systems* 29 (2016). | | |
| [28] | Yamashita, Rikiya, et al. "Convolutional neural networks: an overview and application in radiology." *Insights into imaging* 9.4 (2018): 611-629. | | |
| [29] | He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." *Proceedings of the IEEE international conference on computer vision*. 2015. | | | |
| [30] | Wang, Tonghe, et al. "A learning-based automatic segmentation and quantification method on left ventricle in gated myocardial perfusion SPECT imaging: A feasibility study." *Journal of Nuclear Cardiology* 27.3 (2020): 976-987. | | | |
| [31] | Adaloglou, N. (2020). Understanding coordinate systems and DICOM for deep learning medical image analysis*. https://theaisummer.com/*. | | | |
| [32] | Pieper, Steve, Michael Halle, and Ron Kikinis. "3D Slicer." 2004 2nd IEEE international symposium on biomedical imaging: nano to macro (IEEE Cat No. 04EX821). IEEE, 2004. | | | |
| [33] | Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019). | | | |
| [34] | Pérez-García, Fernando, Rachel Sparks, and Sebastien Ourselin. "TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning." Computer Methods and Programs in Biomedicine 208 (2021): 106236. | | | |
| [35] | Taleb, Aiham, et al. "3d self-supervised methods for medical imaging." *Advances in Neural Information Processing Systems* 33 (2020): 18158-18172. | | | |
| [36] | Loshchilov, Ilya, and Frank Hutter. "Decoupled weight decay regularization." *arXiv preprint arXiv:1711.05101* (2017). | | | |
| [37] | Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014). | | | |
| [38] | Hatamizadeh, Ali, et al. "Unetr: Transformers for 3d medical image segmentation." *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022. | | | |
| [39] | Myronenko, Andriy. "3D MRI brain tumor segmentation using autoencoder regularization." *International MICCAI Brainlesion Workshop*. Springer, Cham, 2018. | | | |
| [40] | Zhuang, Xinrui, et al. "Self-supervised feature learning for 3d medical images by playing a rubik’s cube." *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Cham, 2019. | | | |

1. <http://oftankonyv.reak.bme.hu/tiki-index.php?page=Myocardial+perfusion&structure=Book+for+Physicists> [↑](#footnote-ref-1)
2. <https://cs231n.github.io/convolutional-networks/> [↑](#footnote-ref-2)
3. <https://cs231n.github.io/convolutional-networks/> [↑](#footnote-ref-3)
4. <https://www.cs.toronto.edu/~kriz/cifar.html> [↑](#footnote-ref-4)
5. <https://medisolab.inf.elte.hu/web/> [↑](#footnote-ref-5)
6. <https://mediso.com/global/en> [↑](#footnote-ref-6)