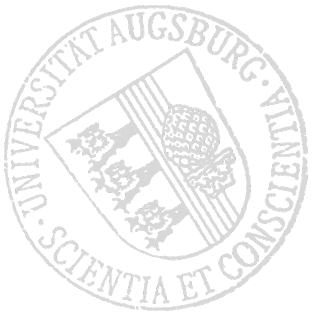
Universität Augsburg
Fakultät für Angewandte Informatik

MISiT IT-Infrastrukturen für die Translationale Medizinische Forschung

IT infrastructures for translational medical research
University of Augsburg

Bachelor Thesis in Informatics

Integration of ConvNeXt U-Net and CBAM U-Net into the medical image segmentation framework MIScnn



Jonas Waibel

Universität Augsburg
Fakultät für Angewandte Informatik

IT-Infrastrukturen für die Translationale Medizinische Forschung

Integration of ConvNeXt U-Net and CBAM U-Net into the medical image segmentation framework MIScnn

Author:	Jonas Waibel
Student ID:	1682618
First reviewer:	Prof. Dr. Frank Kramer
Second reviewer:	Prof. Dr. Bernhard Bauer
Supervisor:	Dennis Hartmann
Submission date:	23.01.2024

Declaration of authorship

I confirm that this bachelor thesis is my own work and that I have documented all sources and materials used.

Abstract

Convolutional neural networks (CNN) are widely used deep learning models used in the field of medical image segmentation. Among them, U-Net is one of the most popular architectures and continues to be expanded with new technologies and methods. We propose two convolutional neural networks, CBAM U-Net, and ConvNeXt U-Net, both being based on the common U-Net model and modifying it by integrating the methods of the convolutional block attention module and the ConvNeXt block into different aspects of its architecture. Both models were integrated into MIScnn, a framework for medical image segmentation, which was also used to prepare the training and prediction pipelines for this thesis. We trained CBAM U-Net, ConvNeXt U-Net, and the default U-Net of MIScnn on two distinct datasets of 2D medical images and compared the generated predictions against the ground truth with the goal of examining the impact of the included methods on the models performances. The results indicate that both models compare favorably in terms of segmentation precision, scoring higher values in the examined metrics. The performance improvements come at the cost of longer training times due to the increase in the total number of layers and complexity.

Zusammenfassung

Convolutional Neural Networks (CNN) sind weit verbreitet im Feld der medizinischen Bildsegmentierung. Unter ihnen ist das U-Net eine der häufigsten Modelle und wird ständig durch neue Technologien und Methoden erweitert. Wir präsentieren zwei Architekturen für neuronale Netzwerke, CBAM U-Net und ConvNeXt U-Net. Beide basieren auf dem U-Net-Modell und erweitern es durch die Integration der Methoden des Convolutional Block Attention Modules (CBAM) und des ConvNeXt-Blocks in jeweils verschiedene Aspekte der Architektur. Beide Modelle wurden in MIScnn integriert, einen Framework für die Segmentierung medizinischer Bilder, welches im Rahmen dieser Arbeit auch zur Erstellung von Trainings- und Vorhersagepipelines verwendet wurde. Wir trainierten CBAM U-Net, ConvNeXt U-Net und das U-Net von MIScnn auf zwei verschiedenen Datensätzen von medizinischen 2D-Bildern und verglichen die generierten Vorhersage-Masken mit der Grundwahrheit um die Auswirkungen der benannten Methoden auf die Leistung der Modelle zu untersuchen. Die Ergebnisse zeigen, dass beide Modelle in Bezug auf die Segmentierungspräzision im Vergleich zu U-Net besser abschneiden und in den untersuchten Metriken höhere Werte erzielen. Die Leistungsverbesserungen gehen jedoch auf Kosten längerer Trainingszeiten, da die Gesamtzahl der Schichten und die Komplexität steigen.

Contents

1 Introduction

Convolutional neural networks (CNN) or ConvNets have long been established as state-of-the-art deep learning methods for medical image segmentation, lending to their "sliding window" approach, their ability to effectively capture spatial context, and their translation equivariance, among other inherent properties [1]. One of the most widely used models is U-Net, boasting a unique architecture consisting of different 2 juxtapositioned 'paths', the first performing downsampling and feature extraction and the second performing several upsampling operations to generate a segmentation mask in the form of a map of pixelwise classifications. These two parts, commonly referred to as the contracting and the expanding paths are only connected by a middle layer and several skip connections [2], which leads to the unique U-shape of the architecture, when visualized. Several modifications and extensions to U-Net have since been proposed, addressing various aspects of the architecture with the goal of improving the scalability and/or the performance of the architecture [3].
In recent years, the field of medical image segmentation has witnessed a paradigm shift with the ascendancy of hierarchical Vision Transformers (ViT) architectures like Swin-Transformers challenging the conventional dominance of CNNs. Derived from the Transformers which were already dominant in the realm of natural language processing (NLP), ViTs superseded CNNs as the dominant architecture in image classification in 2020, offering better scaling behavior and generally outperforming state-of-the-art models like ResNet, but were unable to replace them when it came to general computer vision and semantic segmentation tasks [4]. This changed in 2021 with the advent of hierarchical Vision Transformers (e.g. Swin Transformers) which employ a sliding window approach similar to ConvNets and which were rapidly adopted for general computer vision tasks beyond classification [5].
In "A ConvNet for the 2020s", Liu et al. identify several key design decisions from ViTs and adapt them into the CNN structure, going from a baseline ResNet and discovering ConvNeXt, which was found to outperform Swin Transformers in terms of accuracy on several benchmark datasets [4].
The Convolutional Block Attention Module (CBAM) is a feature enhancement technique designed to improve the performance of convolutional neural networks (CNNs) in image recognition tasks. It aims to capture both channel-wise and spatial attention in an image [6].
Both of these methods could be promising candidates for improving the ability of U-Net when it comes to computer vision tasks such as medical image segmentation, which is why this bachelor thesis focuses on two models, CBAM U-Net and ConvNeXt U-Net, each of which tries to integrate one of these techniques into U-Nets unique architecture, and whether this leads to a positive impact on segmentation performance.

2 Theory

In order to be familiar with the presented concepts, this chapter will give a short introduction to the concepts of image segmentation, neural networks, the standard architecture of a U-Net, and the corresponding differences to the techniques of the Convolutional Block Attention Module (CBAM) and the ConvNeXt Block. We also give a short overview of three evaluation metrics and their definitions.

2.1 Image Segmentation

The term image segmentation describes the task of partitioning an image into segments that represent regions of interest (RoI). More precisely, image segmentation is the process of labeling every pixel as being part of a specific class [7]. Commonly, the RoI are referred to as the foreground or the foreground class, and surrounding tissue as well as the rest of the image not belonging to a RoI are called background [1]. The goal is to provide a representation of objects of interest and their boundaries within the image, typically for the purposes of diagnosis, treatment planning, and quantitative analysis [8]. Segmented objects of interest are often represented by color coding them, as seen in figure 2.1.

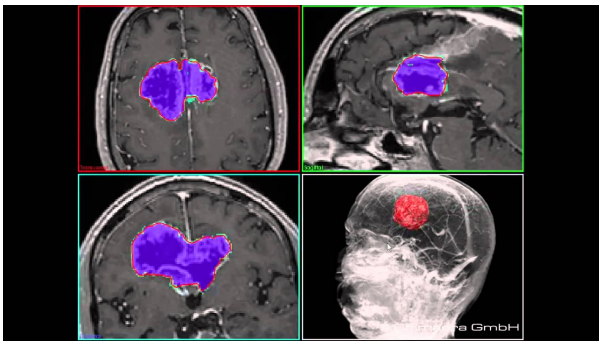


Figure 2.1: MR images of a patients brain. The region marked in blue is the segmentation of a brain tumor. Source: [9].

2.2 Artifical Neural networks

Artificial Neural networks (ANN) are part of the field of machine learning and are inspired by biological neural networks, as they mimic the way neurons interact with each other in the brain.
ANNs are comprised of sets of layers of nodes, with the first layer receiving the model's inputs, multiple hidden layers that the inputs get passed through, and the last layer, which computes the final output, thus also called output layer [10].

[1] from now on, we will use the terms RoI and foreground interchangeably and refer to the rest of the image as background

The node is the most basic computational unit of a neural network. It takes a series of weighted inputs and sums them up. In many cases, the result is then plugged into an activation function, most commonly the sigmoid function, although this is not mandatory. The output is then passed along via connections to other nodes as their input [10].

2.2.1 Feedforward neural networks

Feed-forward neural networks (FFNN) are a type of ANN characterized by the flow of information within the model. Contrary to residual neural networks, outputs are only ever propagated to the nodes of the next layer. FFNNs are typically comprised of a series of fully connected layers, meaning that every single node possesses connections to every node in the successive layer [10].

2.2.2 Convolutional neural networks

For the task of image classification and segmentation, the ability to capture the spatial context of a pixel, like surrounding pixel information, has been shown to improve the performance of machine learning algorithms [11]. For this purpose, CNNs utilize convolutional and pooling layers.
Instead of weighted connections, convolutional layers use a weight vector, also known as a filter, that slides across the input vector in order to compute the output for the next layer.
Pooling layers are used to reduce the size of the output vector by applying a function over a sliding window such as selecting the maximum (max pooling) or the mean (average pooling) [11].
The usage of filters has 2 important advantages.

1 Since weights are shared between neurons in the same layer, the number of trainable parameters is significantly reduced.

2 The inclusion of spatial context allows the model to extract information about local features from the image.

2.3 U-Net

U-Net is a convolutional neural network architecture that was first proposed in 2015 by Ronneberger, Fisher, and Brox, and is a successor to the so-called "fully convolutional network" [12]. The biggest modification of U-Net in comparison to earlier convolutional neural networks is the supplementation of a series of contracting layers by the inclusion of successive layers that upsample the output feature map. Ronneberger describes these layers as the contracting and expanding path [13]. U-Net also includes several connections that aim to combine high-resolution feature maps from the contracting path with the upsampled output of layers from the expanding path. These differences result in

a quite symmetrical architecture that resembles a U, hence the name. These modifications aim to allow the network to achieve good segmentation results using only very few training examples [13]. The general U-Net architecture is depicted in figure 2.2.

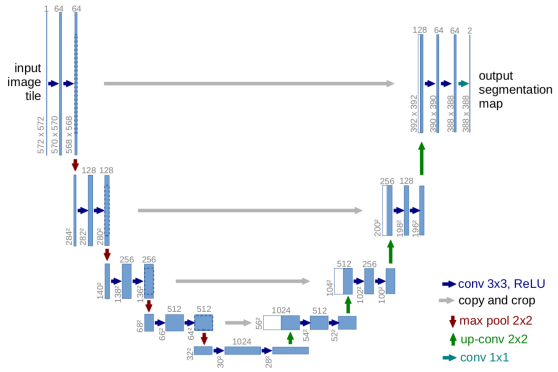


Figure 2.2: Example of the U-Net architecture for a 572x572 input image. In the contracting path, a series of convolution and pooling operations downsample the input to a resolution of 32x32 pixels. In the expanding path, convolution and upsampling operations expand the output image back to a higher resolution, resulting in the final segmentation map. Source: [13]

Due to its performance on image segmentation tasks and its ability to work well on small training sets, U-Net has become widely used to segment images from a number of modern medical modalities, such as CT-scans, MRI and X-rays [14].

2.4 CBAM

The Convolutional Block Attention Module (CBAM) is a module intended for use in CNNs [6]. Its goal is to utilize attention, the ability of the network to focus on important features, in order to improve performance. CBAM consists of 2 sub-modules, the channel and the spatial attention modules, which focus on cross-channel and spatial information respectively, which are depicted in figure 2.3.

2.5 ConvNeXt

In 2021, Zhuang Liu et al. proposed ConvNeXt, a CNN model that integrated a couple of design characteristics from Vision Transformers (ViT), a transformer adapted for computer vision tasks, with the goal of increasing the performance of Convolutional neural networks [4]. Among them were the adaptation of larger kernel sizes, a inverted bottleneck structure, where the dimensions of the hidden layers are up to 4 times larger

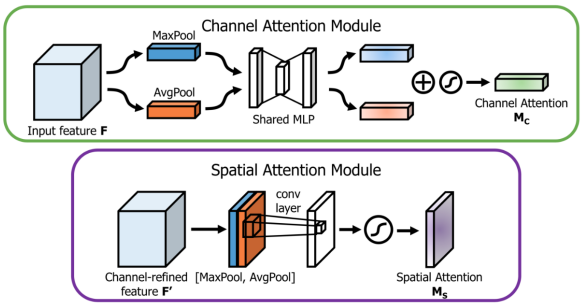


Figure 2.3: The channel and spatial attention modules of CBAM. Source: [6]

than those of the input and the seperation of downsampling layers, which led to the discovery of the ConvNeXt block. Blocks are a series of layers that are often frequently used in feature extraction [4]. Figure 2.4 depicts the structure of a ConvNeXt block side-by-side with a Swin Transformer and a ResNet block.

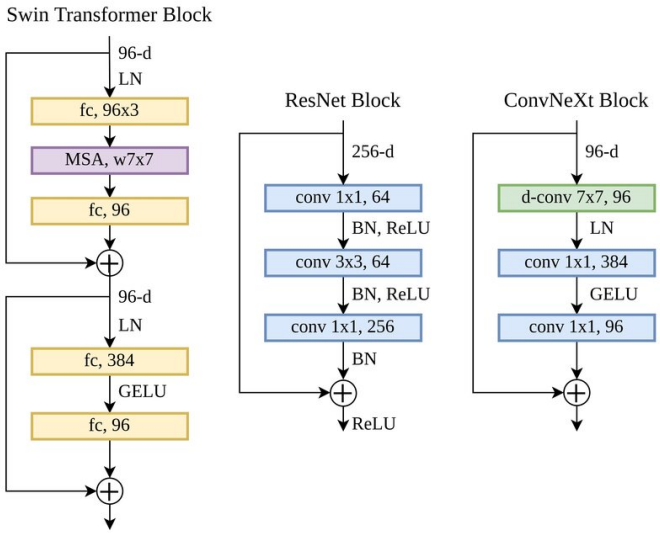


Figure 2.4: Structure of a ConvNeXt block compared to a Swin transformer and a ResNet block. The first layer consists of a convolution with a 7x7 kernel, and the hidden layer realises the inverted bottleneck approach by increasing the number of channels by the factor 4. Source: [4]

2.6 MIScnn

MIScnn is a medical image segmentation framework that aims to provide the user with the ability to set up medical image segmentation pipelines. It includes a number of functionalities such as data I/O, preprocessing, data augmentation, deep learning mod-

els, and their evaluation via its open-source Python library [15]. Available models also include neural network architectures like U-Net.

2.7 Evaluation metrics

Evaluation metrics in the field of medical image segmentation are used to assess the quality of a segmentation masks measuring the similarity or dissimilarity between a predicted segmentation mask and a ground truth segmentation mask when it comes to the correct outlining of RoI [16]. Most often this involves the calculation of a Confusion matrix. It contains the number of pixels correctly classified as being part of the foreground, also called true positives (TP), the number of pixels mistakenly classified as belonging to the foreground, called false positives (FP), the number of pixels correctly classified as belonging to the background, called true negatives (TN), and the number of pixels mistakenly classified as part of the background, called false negatives (FN) [17].

2.7.1 Sensitivity

The sensitivity , also called the true positive rate, measures the ability to identify pixels belonging to a RoI. It is defined as the proportion of correctly classified pixels among all pixels of the foreground [16].

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.1}$$

2.7.2 Specificity

The specificity, refers to the proportion of TN pixels among all background pixels, and measures the ability to correctly identify the background class [16].

$$Specificity = \frac{TN}{TN + FP} \tag{2.2}$$

2.7.3 Dice Similarity Coefficient (DSC)

The DSC, also called dice-score or f1-score, measures the spatial overlap between the predicted segmented area and the ground truth mask and also penalizes false positives [16][2].

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{2.3}$$

[2]We may refer to the DSC as the dice-score

3 Methods

We implemented ConvNeXt U-Net and CBAM U-Net and trained them both along with a common U-Net on two image datasets fit for benchmarking segmentation performance using the MIScnn framework. Predictions were generated on smaller test subsets containing only a fraction of all available images, and compared with respect to three evaluation metrics, namely DSC, sensitivity, and specificity. All scripts were implemented in the Python programming language and are documented in the GitHub repository belonging to this work, found under https://git.rz.uni-augsburg.de/misit-bachelor/ConvNextUnet. The repository contains separate folders for each model and dataset containing the architecture as a Python class, training and prediction pipelines as well as logging files from said procedures if available and .hdf5 files in which the models weights after training are stored and which can be used to load the fitted models in order to reproduce the results. Each folder also contains the predicted segmentation masks generated by the models on each of the 2 datasets. The evaluation was performed in the form of Jupyter notebooks, giving a documented overview of the process of calculating and plotting evaluation metric scores. The notebooks also showcase some of the achieved predictions and visually compare them with the ground truth masks. Additional information for traversing the scripts can be found in the README. Lastly, all images and plots, generated and displayed in the evaluation notebooks are stored in the folder 'Images'.

3.1 Datasets

The datasets used for training and generating predictions on are the dataset of the ISIC 2018 Task 1 Challenge and a set of gastrointestinal polyp images called Kvasir-SEG [18] [19]. Both datasets consist of 2D images in the .png or .jpg format and the associated binary segmentation masks and will be used to evaluate the performance both between the different models and the datasets.

3.1.1 ISIC2018

Hosted by the International Skin Imaging Collaboration (ISIC), the ISIC2018 Task1 dataset consists of 2594 RGB images of dermatoscopic lesions in JPEG format [18]. The image sizes vary, with the smallest being 540x722 and the largest being 4499x6748. each image comes with a binary mask in which the pixel value indicates whether it belongs to the background of the image or the foreground, in this case being part of a lesion (see figure 3.1). All ground truth masks were generated using either a fully automated algorithm, a semi-automated 'flood-fill' algorithm or manually annotated, and all were reviewed by dermatology experts [18]. Task 1 of the ISIC Challenge was to submit an algorithmic approach to generating segmentations for these images and to compare them against the ground truth segmentation masks. As such, the dataset lends itself well to

testing deep-learning models for image segmentation [18]. For this thesis, we had 2000 images (72%) available for the training set, 600 images (21%) for the testing set and 150 images comprised the validation set used for controlling the generalization error during training.

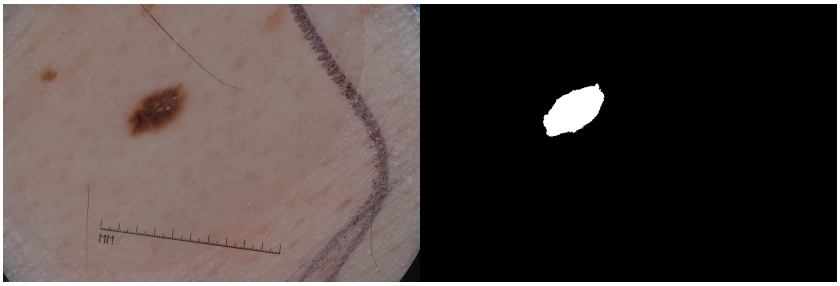


Figure 3.1: Example of an image and its binary mask from the ISIC2018 task 1 dataset. The original RGB image (left) displays a pigmented skin lesion, and the binary mask (right) displays its outline. White indicates, that the pixel is part of the lesion.

3.1.2 Kvasir-SEG

A successor to the multi-class image dataset Kvasir, Kvasir-SEG consists of 1000 .jpg images of polyps in the gastrointestinal tract, which were taken during endoscopic procedures as well as their corresponding binary segmentation masks. Similar to the ISIC dataset, the segmentation masks annotate pixels as either belonging to the foreground, in this case meaning to a polyp, or as belonging to the background. All masks were manually created by a medical doctor and verified by a gastroenterologist [19]. Kvasir-SEG also contains a JSON file of bounding box information, which could also be used to highlight and annotate regions of interest [19]. The shapes of the images are not homogenous and range from 352x568 to 1072x1920. 700 (70%) randomly chosen images were assigned to the training set, 100 (10%) images were used for validation and 200 (20%) images composed the test set. Figure 3.2 depicts an example of an image from Kvasir-SEG along with the ground truth segmentation mask.

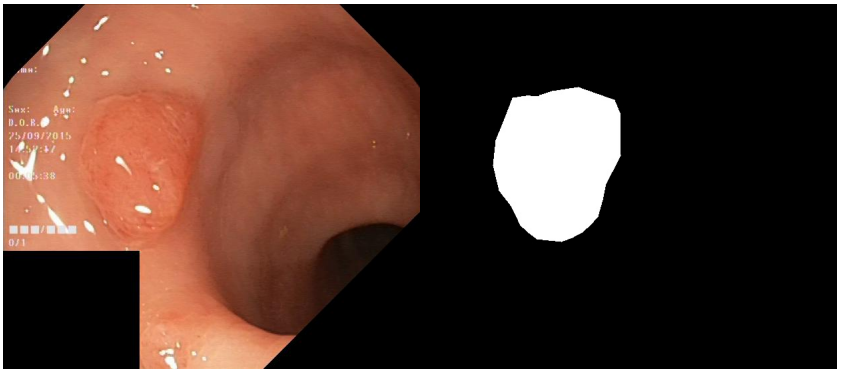


Figure 3.2: 'Image_0487' from the Kvasir-SEG dataset along with its binary segmentation mask. The original image (left) displays a part of the gastrointestinal tract containing a polyp, and the segmentation mask (right) outlines said polyp in white.

3.2 CBAM U-Net

In the U-Net architecture, high-resolution feature maps, which are the result of contracting layers, are stored and later concatenated with the output of upsampling operations. The goal is to allow successive convolutional layers to learn to assemble more precise outputs [13]. In order to profit from the attention mechanism, CBAM was used to compute spatial and cross-channel information on the high-resolution feature map, before combining it with the upsampling output. The resulting architecture was coined "CBAM U-Net".
For implementing CBAM U-Net, methods available in Pythons tensorflow.keras library were used. The channel attention and spatial attention modules were separated into methods. At the start of both modules, the input is pooled with the 'GlobalAveragePooling2D' and 'GlobalMaxPooling2D' operations respectively. In the channel attention module, the two resulting outputs are both fed into a shared multilayer perceptron consisting of 2 fully connected layers with the first layer dividing the feature dimensions of the input by a ratio, which was 8 by default, and the second layer having the same feature dimensions as the input. The results are concatenated and plugged into the sigmoid function. Lastly, the original feature map is re-weighted by mutliplying it with the output of the above described process. In the spatial attention module, the two resulting outputs of the pooling operations are concatenated along the channel dimension, followed by a convolution with a 7x7 kernel and a stride of 1. Lastly, the sigmoid function is applied and the result is once again multiplied with the original input feature. Finally, the *cbam_2d* method implements CBAM by sequentially applying the channel attention method, then the spatial attention method on the input.
MIScnns implementation of U-Net sees the outputs of the contracting layer method being stored in a list which is later used for concatenating them in reverse order of insertion

against the feature maps in the expanding layer method. In CBAM U-Net, the feature maps are first being fed to the *cbam_2d* method as described above before being stored in the stack. The architecture is documented in the file 'cbam_unet.py' of the GitHub repository.

3.3 ConvNeXt U-Net

A contracting layer in U-Net performs a series of convolutions for feature extraction and a max pooling operation to reduce the size of the feature map. Since the ConvNeXt Block does not perform downsampling on its own, a separate downsampling block consisting of a convolutional layer with kernel size 2x2 and a stride of 2 was used in combination. ConvNeXt U-Net replaces the final Max Pooling layer by a ConvNeXt block (see figure 2.4) and the downsampling layer in hopes of boosting the models feature extraction capabilities while keeping the dimensions of the output in order for the successive layer to process it without major alterations to the contracting layers structure. This modification to the base U-Net architecture is largely inspired by the Attention Augmented ConvNext U-Net first proposed by Wu et al in 2022 [20] and pictured in figure 3.3.
Just like CBAM U-net, ConvNeXt U-Net makes use of TensorFlows keras library and the available predefined methods and layers within it. the Convnext block was implemented as a method 'convnext_block_2D' which takes a feature map as input. The method sequentially applies a number of layers to the initial feature map, starting with a depthwise convolution with a 7x7 kernel and a stride of 1, followed by a 'LayerNormalization' layer. The next two Convolutional layers (both with kernel size 1x1 and strides of 1) realize the inverted bottleneck approach of ConvNeXt adapted from the vision transformer architecture and raise the number of filters to 4 times the amount of the input before reducing them again. In between, the 'GeLu' activation function is utilized. Lastly, we apply LayerScale (see 3.3.1) and StochasticDepth (see 3.3.2) to the feature map before returning the output.
Similarily to the ConvNeXt block, the downsampling layer is also realized as a method 'downsample_2D', which applies LayerNormalization and a convolution operation with kernel size 2x2 and a stride of 2 on the input.
Our implementation of ConvNeXt is based on the plain variant of the U-Net architecture of MIScnn, with the main difference being the application of the method 'convnext_block_2D' and 'downsample_2D' instead of the final Max Pooling operation of the contracting layer. The architecture is documented in the the file 'convnext_unet.py' of the github repository.

3.3.1 LayerScale

LayerScale was first defined in "Going deeper with Image Transformers" by Touvron et al. [21], where a simple diagonal matrix with randomly initialized values (close to 0) is added to the output. This approach has been shown to increase accuracy and training of

image tansformers while adding relatively few extra parameters, especially in the context of deep networks [21].
Since there is currently no implementation in tensorflow or its addons for LayerScale, we created our own custom tensorflow layer for this purpose. It consists of a $\_\_init\_\_$ method to initialize the layer object, provide it with an initial value for seeding and create the diagonal matrix, as well as the call method, which forwards the input through the layer by multiplying it with said matrix.

3.3.2 StochasticDepth

StochasticDepth is an approach that aims to reduce the size of a deep neural network during training by randomly skipping layers [22].
Tensorflows addons provide an implementation of StochasticDepth, which we've used for this work. It is important to note that the development of tensorflows addon library has ceased, and it is expected to reach end of life in early 2024, at which point one should look into implementing a custom StochasticDepth layer for themselves.

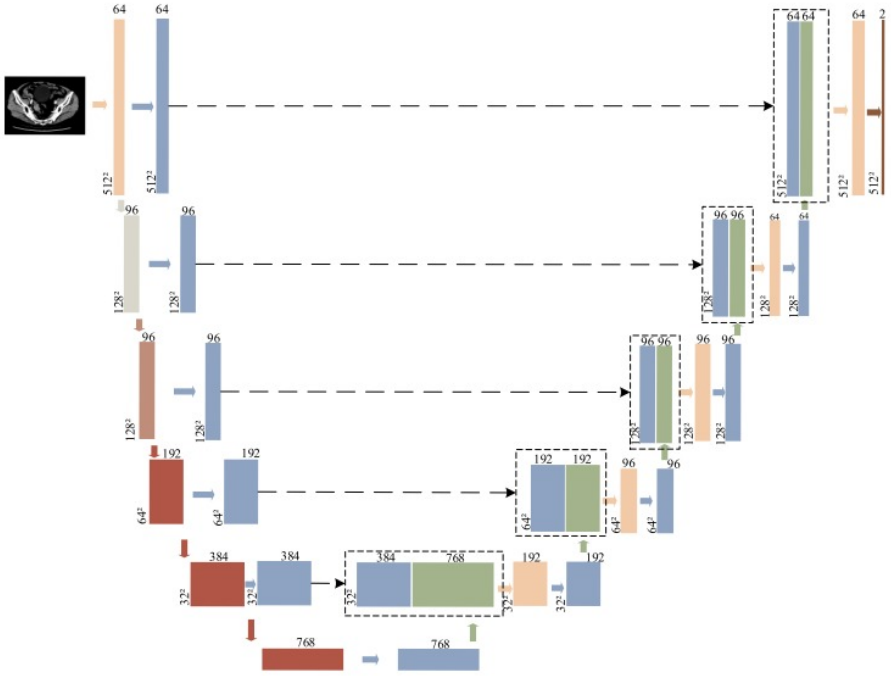


Figure 3.3: Attention Augmented ConvNeXt U-Net. The brown and red arrows represent the application of a ConvNeXt block with and without being followed by a downsampling block. Source: [20].

3.4 Integration into MIScnn

Both CBAM and ConvNeXt U-Net were implemented as classes that inherit from Miscnn's '*Abstract_Architecture*', which provides the abstract methods '*create_model_2D*' (cm2) and '*create_model_3D*' (cm3). Miscnn also provides a couple of concrete subclasses, such as a multi-resolution and a compact version of the U-Net architecture. We took the standard U-Net class as the starting point for implementing our custom architectures and modified cm2 and cm3 to reflect the proposed changes. For CBAM U-Net, the feature maps resulting from contracting layers are first given as the input to the *cbam_2d* method before being stored in a queue for later concatenation. The Max Pooling layer of the contracting block of U-Net was replaced by a ConvNeXt block followed by a custom downsampling layer in ConvNeXt U-Net.

3.5 Training and prediction

The training and prediction procedures for every architecture were done on a server cluster provided by the chair of IT infrastructures for translational medical research. Proper package administration was achieved by creating a Python virtual environment on which we installed all required libraries for training a neural network with MIScnn. The files specifying each custom architecture were copied onto the server. The training and testing sets were identical for each architecture. All training pipelines start by specifying a number of subfunctions, which are applied during the preprocessing. These include the resizing of the images from their heterogeneous sizes to the 512x512 format, as well as a z-score normalization and a custom subfunction, which transforms the foreground pixels of the ground truth masks from the value 255 to 1, making them recognizable to the model. MIScnn provides a class for data augmentation, that allows us to make use of a number of data augmentation methods in order to artificially increase the amount of available data for training. We performed data augmentation by rescaling, rotating, mirroring, elastically deforming, and adding Gaussian noise in 2 cycles, meaning every image is augmented twice. The training was performed in batches of 4, using Tversky loss as the loss metric. We specified the number of epochs as 1000, however, none of the training procedures reached that number due to the included early stopping callback assuring the termination of the training once the generalization error increased, which also triggered the saving of the models tuned weights as a .hdf5 file and a plot of the fitting curve as a .png.

3.6 Evaluation

For each of the two datasets, we evaluated the predictions generated by the 3 architectures with respect to the DSC, the sensitivity, and the specificity. The code for calculating metric scores is documented in the Jupyter notebook 'Evaluation_isic.ipynb' and 'Evaluation_kvasir.ipynb', Analysing the predictions on the Images of the ISIC2018

and the Kvasir-SEG dataset respectively. We made use of the publicly available Python library miseval, which provides a number of methods for calculating metrics commonly used in the performance assessment of medical image segmentation models [23]. The notebook contains 3 chapters, namely a section sampling some of the archived predictions and comparing them to the ground truth masks via an overlay, a section inspecting the occurrence and frequency of artifacts in the predictions of the different models, and lastly the calculation of the metric scores of the dice-score, the sensitivity and the specificity. Finally, each notebook contains a number of plots in order to visually compare the metric scores.

4 Results

In the following chapter, we present the results of our experiment by showcasing some of the generated prediction masks and listing the values that each architecture achieved on the respective datasets. The chapter also includes some of the plots that were created as part of the evaluation notebooks of the GitHub repository as visual aids for the later interpretation of the results.

4.1 Performance on ISIC2018

Figures 4.1 showcases examples of generated predictions along with the ground truths and the original images. The calculated scores are depicted in Table 1, separated by metric and architecture, and rounded to the third decimal place. They are directly extracted from 'Evaluation_isic.ipynb'.

	UNet	CBAM	ConvNeXt
DSC	0.726	0.760	0.796
Sensitivity	0.720	0.721	0.757
Specificity	0.977	0.985	0.986

Table 1: Table of calculated metric scores (rounded to third decimal place) for each architecture on the ISIC2018 dataset.

The figures 4.2, 4.3, and 4.4 depict box plots of each models metric scores side by side. The medians reflect the values listed in table 1. For both the dice-score and the sensitivity, the graphs show broad interquartile ranges, comprising the middle 50% of the achieved scores. The plots also visualize a large amount of outliers.

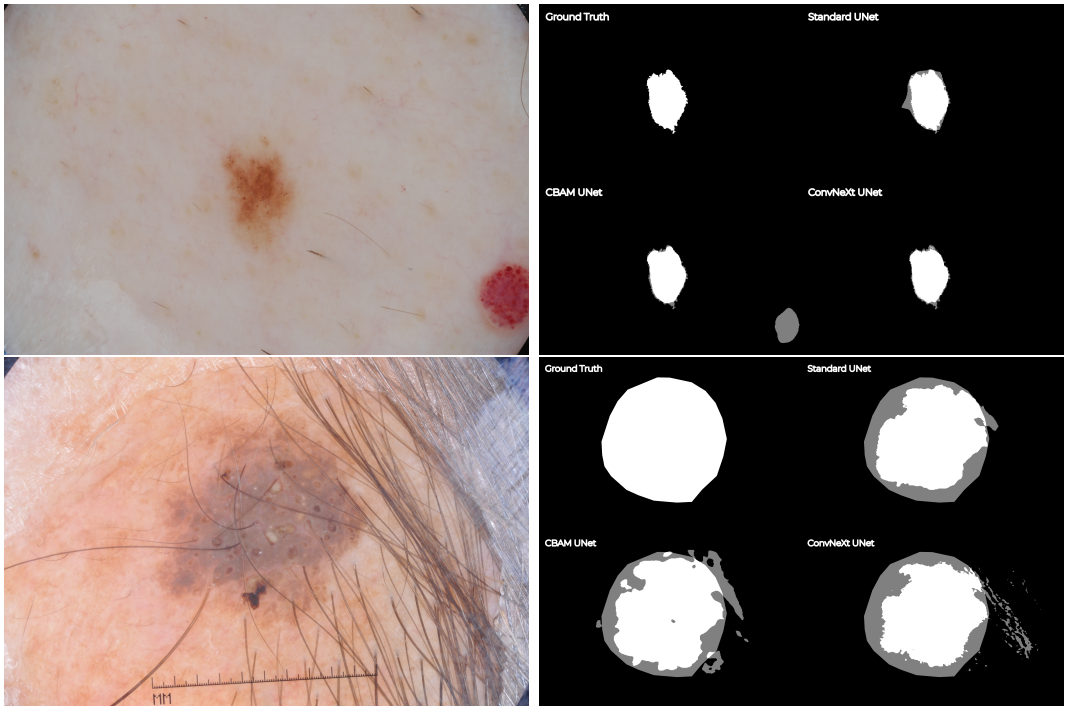


Figure 4.1: Images 'ISIC_0013045' and 'ISIC_0012548' from the ISIC2018 dataset (left column) The right column features 4x4 grids in which we display the ground truth in white and overlay the predictions from all 3 models in gray

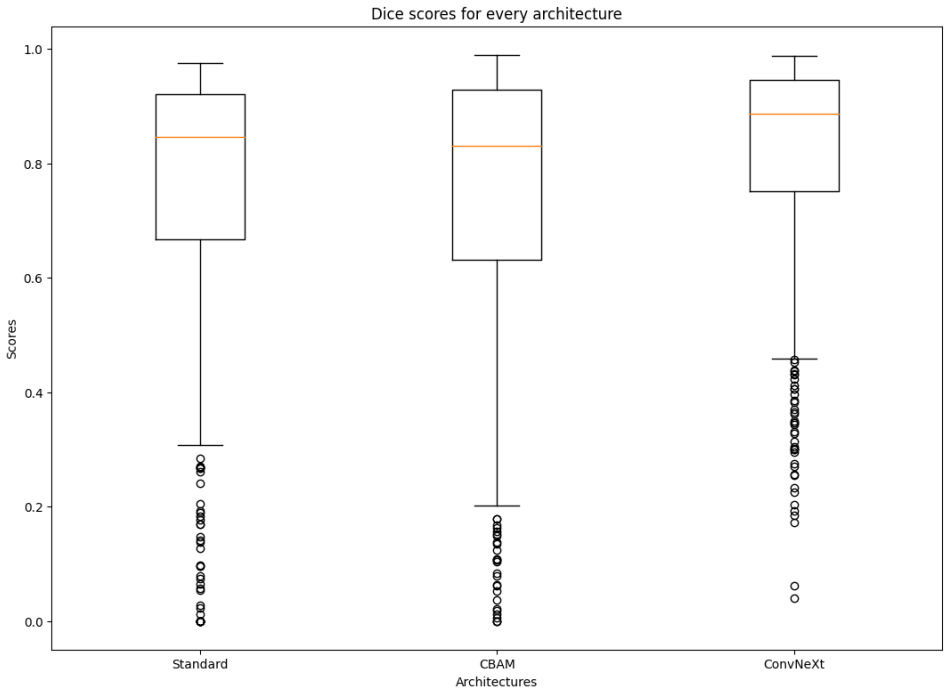


Figure 4.2: Box plots of the average dice-score for every architecture.

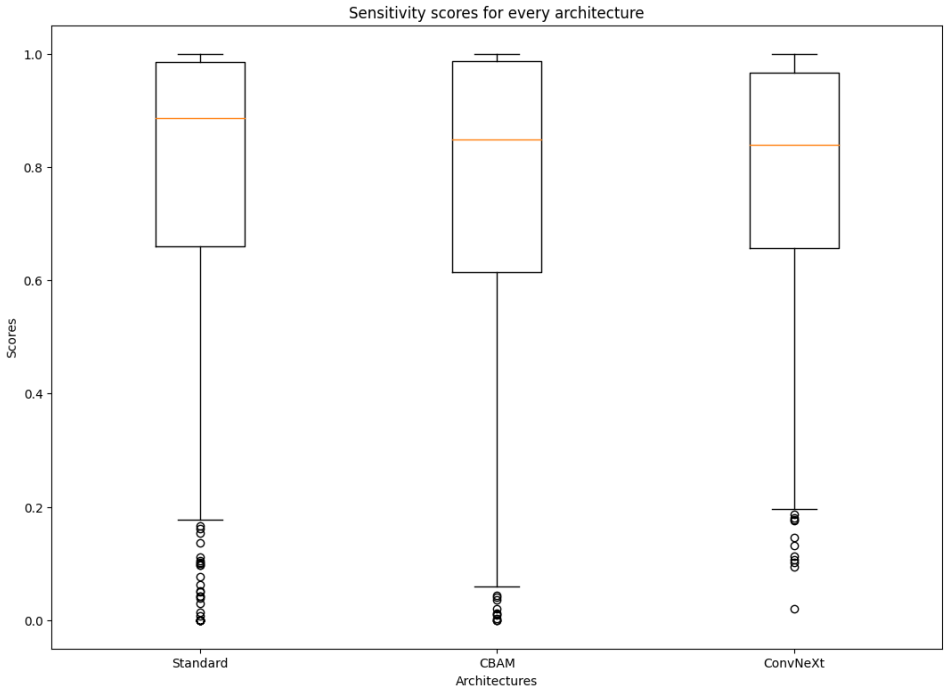


Figure 4.3: Box plots of the average sensitivity for every architecture.

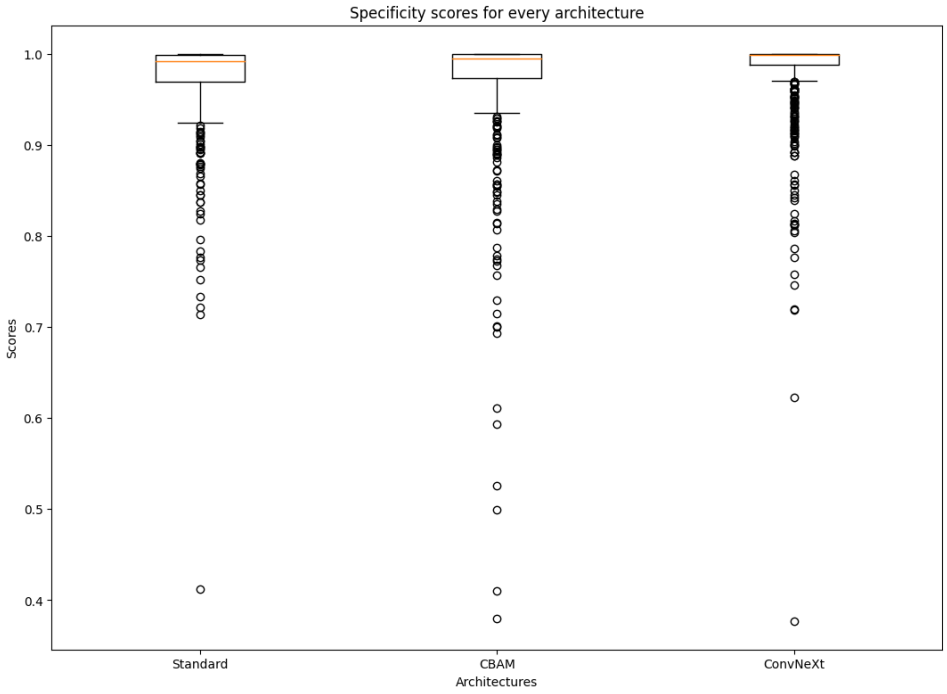


Figure 4.4: Box plots of the average specificity for every architecture.

As seen in figure 4.5, All models training procedures took between 12 and 22 epochs before being terminated by the EarlyStopping callback, which activated after the validation loss increased for 5 Epochs.

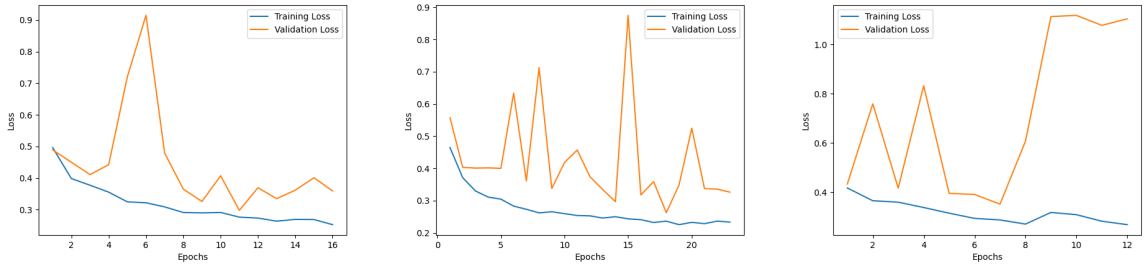


Figure 4.5: Training losses (blue) and validation losses (yellow) during training for each architecture on the ISIC2018 dataset. From left to right: U-Net, CBAM U-Net, ConvNeXt U-Net.

4.2 Performance on Kvasir-SEG

Figure 4.6 showcases examples of generated predictions along with the ground truths and the original images. Table 2 lists the achieved mean scores for every metric and model on the predictions for the Kvasir-SEG dataset. The achieved values are generally lower than the results on the ISIC2018 dataset, with the average dice score of U-Net being ~ 0.39. Both CBAM U-Net and ConvNeX U-Net performed favorably compared to the standard U-Net architecture with regards to the average dice-score [3] and specificity, while the sensitivity scores are more varied.

	UNet	CBAM	ConvNeXt
Dice	0.389	0.483	0.465
Sensitivity	0.760	0.718	0.58
Specificity	0.726	0.862	0.920

Table 2: Table of calculated metric scores (rounded to third decimal place) for each architecture on the Kvasir-SEG dataset.

[3] the term dice-score refers to the DSC metric and may be used interchangeably

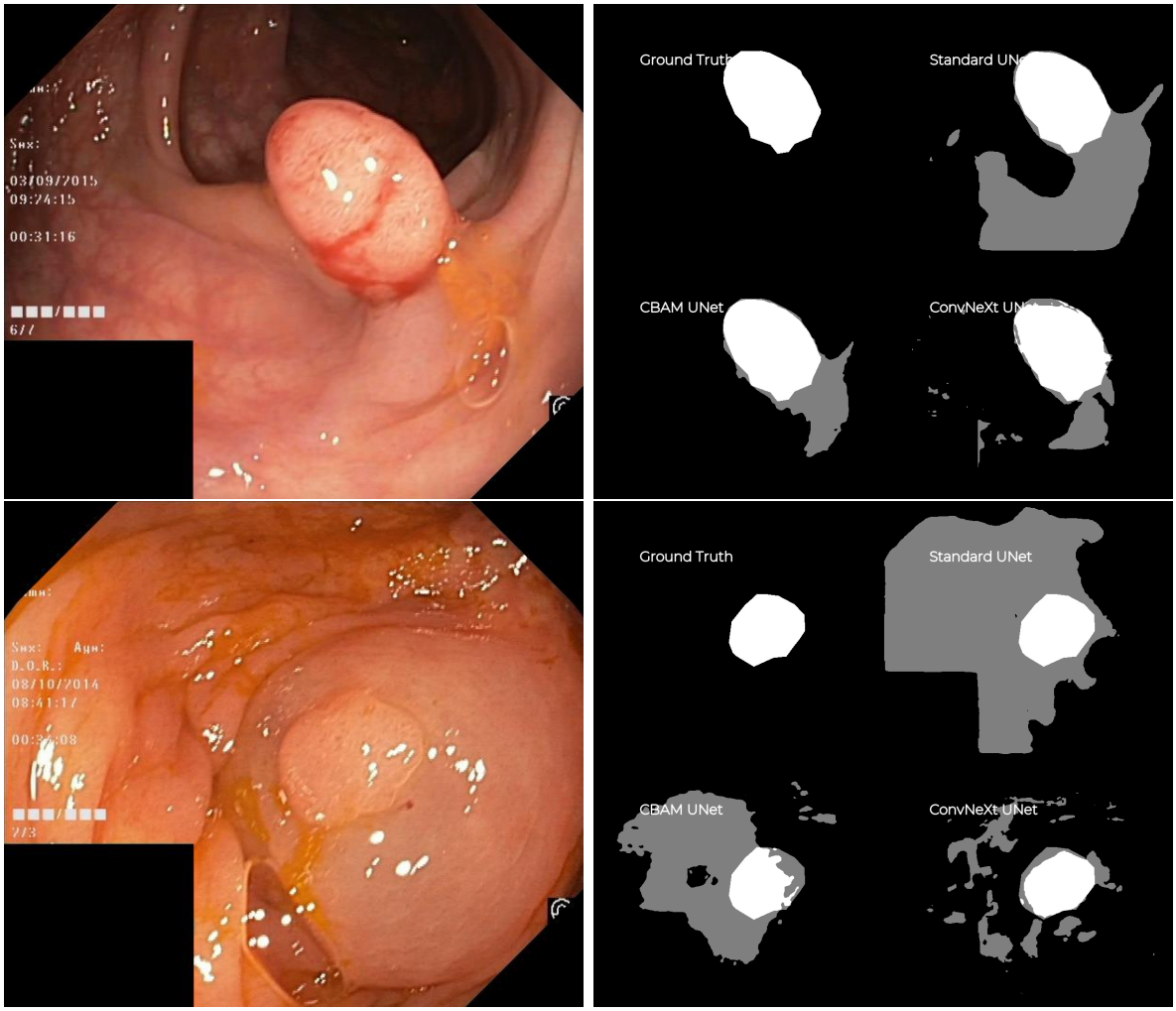


Figure 4.6: Images 'Image_0047' and 'Image_0064' from the Kvasir-SEG dataset (left column) The right column features 4x4 grids of the ground truth masks (upper left corner of the grid) and the predictions generated by the 3 architectures overlayed with the ground truth.

The figures 4.7, 4.8, and 4.9 depict box plots of the metric scores that were achieved by the models, as they were computed in the respective Evaluation notebook. The graphs show broad interquartile ranges, with the whiskers often spanning the whole range, except for the specificity scores depicted in figure 4.9.

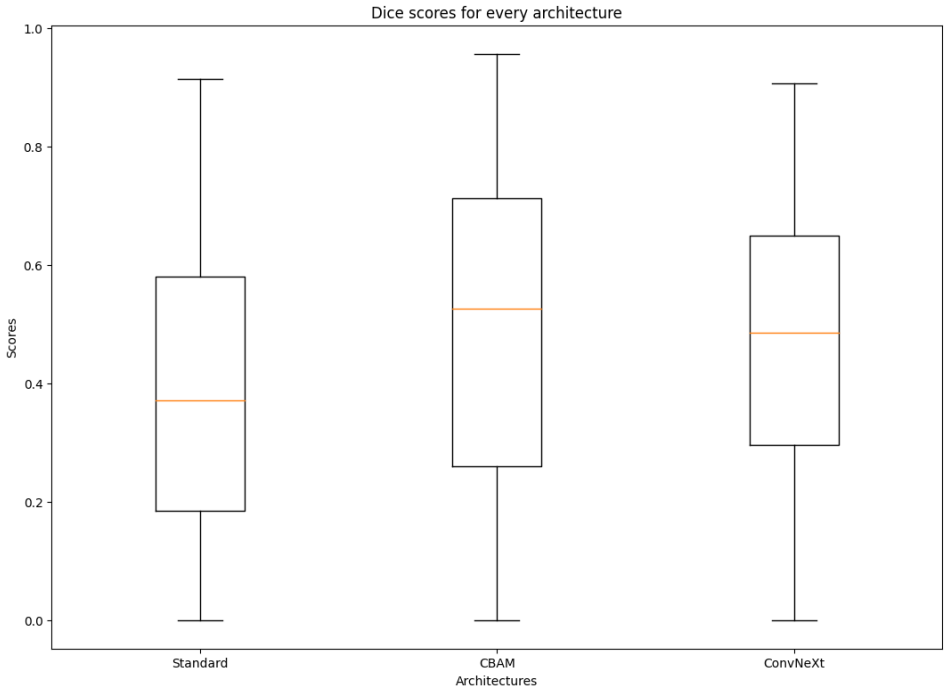


Figure 4.7: Box plots of the average dice-score for every architecture.

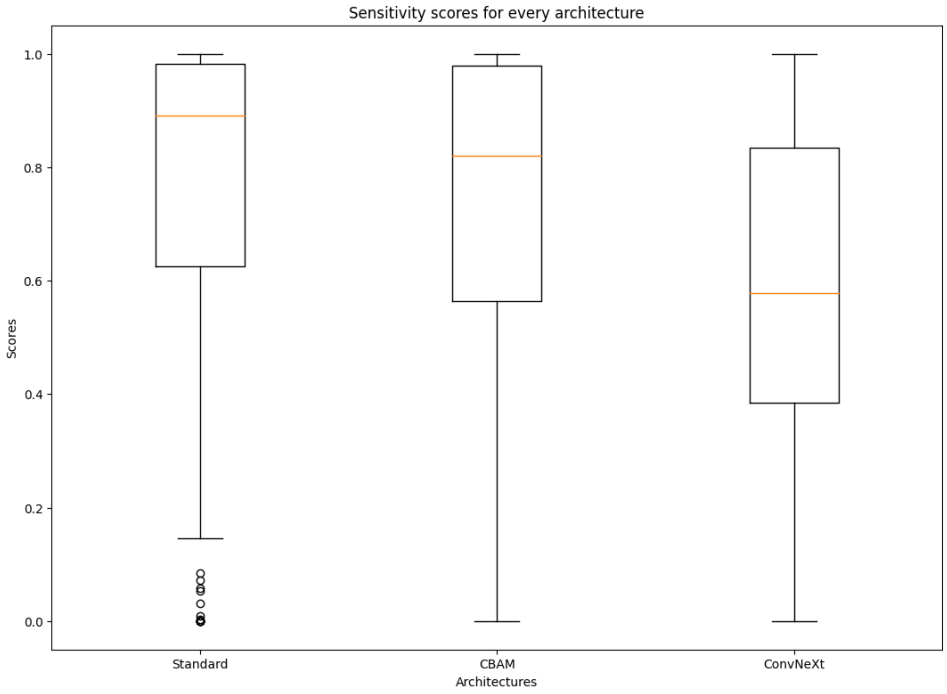


Figure 4.8: Box plots of the average sensitivity for every architecture.

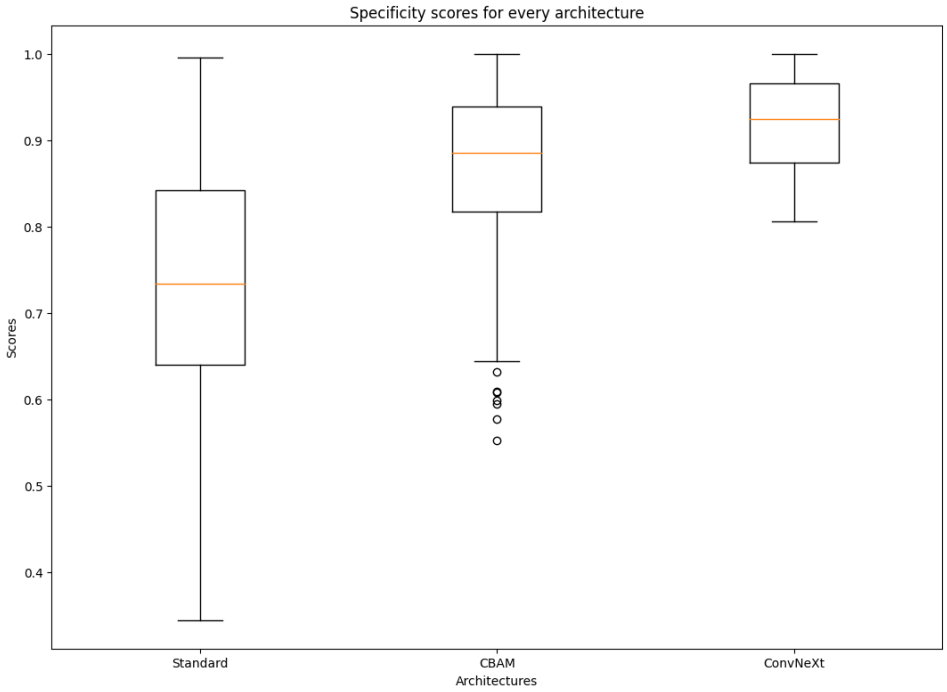


Figure 4.9: Box plots of the average sensitivity for every architecture.

Depicted in figure 4.10 are the training and validation losses during the training procedures, which took between 10 and 20 epochs.

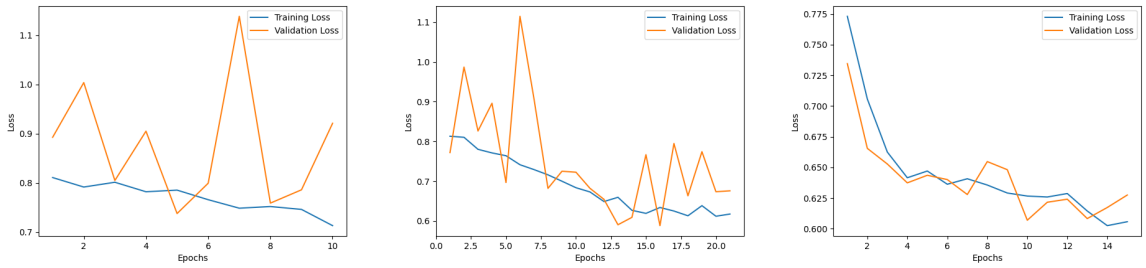


Figure 4.10: Training losses (blue) and validation losses (yellow) during training for each architecture on the Kvasir-SEG dataset. From left to right: U-Net, CBAM U-Net, ConvNeXt U-Net.

4.3 Training times

We logged the duration from start to finish for all 6 training procedures in order to examine potential increases in training time as a consequence of the modifications to the U-Net architecture in CBAM U-Net and ConvNeXt U-Net. The results are listed below in table 3.

	UNet	CBAM	ConvNeXt
ISIC2018	63 min / $\sim$ 6.3 mpe	87 min / $\sim$ 4.35 mpe	139 min / $\sim$ 9.2 mpe
Kvasir-SEG	11 min / $\sim$ 1.1 mpe	17 min / $\sim$ 0.85 mpe	50 min / $\sim$ 3.3 mpe

Table 3: Training times seperated by dataset and model, both as the total duration rounded to the minute and averaged over the number of epochs (mpe: minutes per epoch).

Table 3 shows an increase in total training duration with CBAM U-Net and ConvNeXt U-Net. When averaged over the number of epochs of the training procedures however, CBAM U-Net took less time per epoch than U-Net, while training ConvNeXt U-Net took longer for one iteration over the dataset, up to a factor of 3 on the Kvasir-SEG dataset compared to U-Net.

5 Discussion

We examined the proposed architectures with regards to their performance on 2 medical image datasets and compared them to the benchmark of a conventional U-Net model. In the following chapter, we aim to interpret the results by analyzing the achieved metric scores, taking additional information such as fitting curves and training times into account, and discussing some properties of the datasets and training and prediction procedures. The goal is to determine whether the 2 methods of interest, namely the Convolutional Block Attention Mechanism (CBAM) and the ConvNeXt block, may play a role in modernizing and improving the accuracy of U-Net on medical image segmentation tasks. Additionally, we want to highlight some of the difficulties and challenges we encountered during the process of implementing and training both models and how they could be avoided, as well as present some alternative architectures. Lastly, we give an outlook on possible points of interest for future work.

5.1 Interpreting the results

The calculated evaluation metrics reveal that ConvNeXt U-Net and CBAM U-Net achieved better performances in most cases across both datasets, consistently reaching comparable or higher scores in terms of specificity, measuring the ability of the model to correctly identify pixels as belonging to the background and avoiding false positives and sensitivity, which represents the accurate identification of regions of interests. The only exception to this is the Sensitivity on the Kvasir-SEG dataset, in which both architectures actually performed worse than U-Net, with ConvNeXt U-Net only achieving a mean score of ~ 0.58 compared to the average of ~ 0.76 of U-Net. However, the DSC, a measure of the spatial overlap between the regions of interest in the predicted and the ground truth segmentation masks, shows an altogether increase in the metric scores for both CBAM U-Net and ConvNeXt U-Net compared to the standard model.
The training timetable reveals longer total training times for both proposed models coupled with more epochs of training. Averaged over the mean time per epoch, the results show an increase in training duration with ConvNeXt U-Net, which is unsurprising considering the inclusion of more layers in every block of the contracting path. CBAM U-Net's time per epoch is lower than U-Net's, meaning the usage of CBAM U-Net appears to lead to smaller convergence steps. This correlation should be further investigated.
The overall performance of all models leaves room for improvement. for example, the mean sensitivity over all models of the predictions on the ISIC2018 dataset is ~ 0.73 and the average dice score for the predictions generated on Kvasir-SEG is ~ 0.44. Additionally, the box plots of chapter 4 show a high variance in the achieved scores across the board. We argue, that a number of key aspects both inherent properties of the utilized datasets and the training and prediction pipelines have contributed to low performances, which we listed below.

5.1.1 Inaccurate ground truth masks

Both datasets feature binary segmentation masks representing the ground truth, which were either generated by algorithms or manually annotated by experts. Despite this, some masks appear to be more inaccurate than others, providing a less-than-ideal segmentation of the regions of interest. Most commonly these are too general, only approximately outlining the lesion or the polyp. This may lead to situations in which the generated prediction is more precise than the actual ground truth, leading to an increased number of detected false positives and therefore possibly skewing the resulting evaluation scores. Figure 5.1 showcases two examples of images from the ISIC2018 dataset, where the ground truth segmentation masks cover a much larger area than the lesion on the original image seems to occupy. In figure 5.2, We can see a ground truth mask, which fails to include most of the inner area of the foreground. Due to this discrepancy, the predicted segmentation masks prove to be more precise than the ground truth.

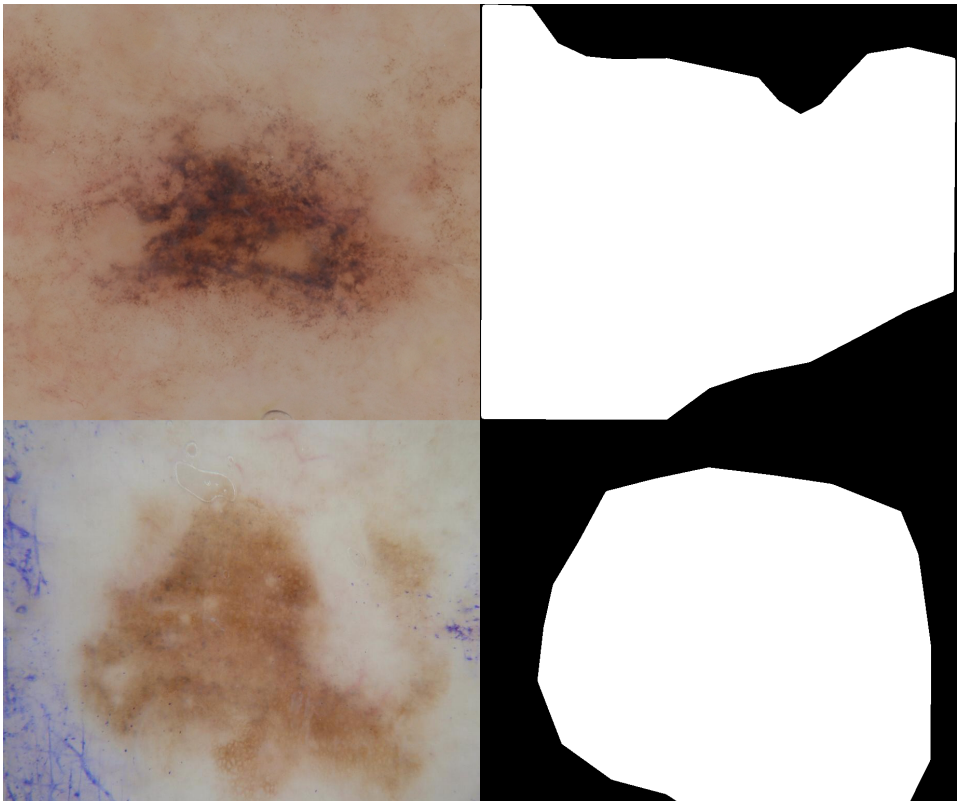


Figure 5.1: Examples of inaccurate ground truth masks (right column) along with their original images for comparison (left column), 'ISIC_0000517' (upper row) and 'ISIC_0010497' (lower row). The foreground pixels encompass a much larger area than the lesion and only roughly outline the region of interest

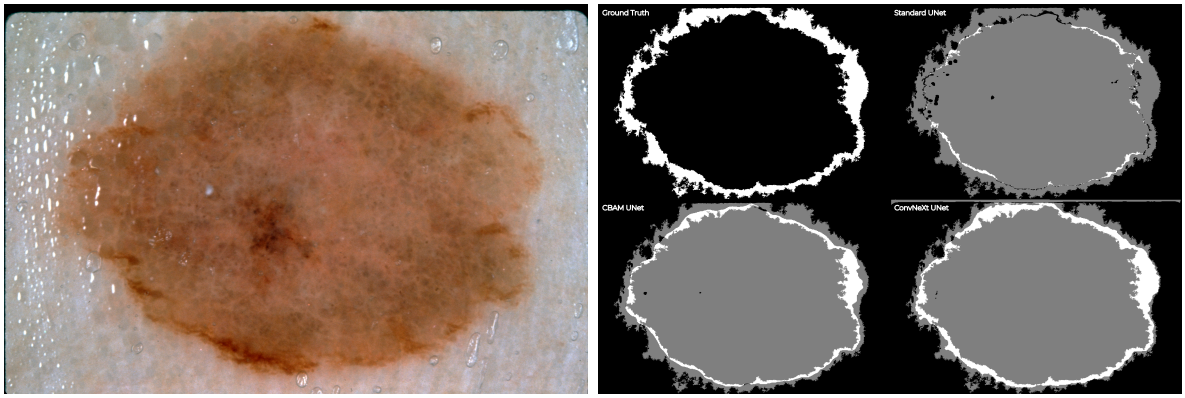


Figure 5.2: 'ISIC_0015353' from the ISIC2018 dataset. The ground truth mask (upper left corner on the right) fails to identify most of the inner are of the lesion as the foreground.

5.1.2 Complex/Difficult regions of interest

Skin lesions come in a number of different and varied shapes and colors. As a result, the ISIC2018 dataset contains several images of regions of interest that are similar in color to the surrounding tissue and/or feature convoluted outlines. Lastly, lesions could be partially obscured by other image features such as body hair or markings. Similarly, polyps come in different forms and vary in size and/or shape depending on their origin tissue, and histology. All the mentioned factors pose unique challenges for computer vision algorithms such as medical image segmentation models, which could have impacted the performance of our architectures. We depicted examples of images with foregrounds that are difficult to distinguish from the background in figures 5.3 and 5.4.

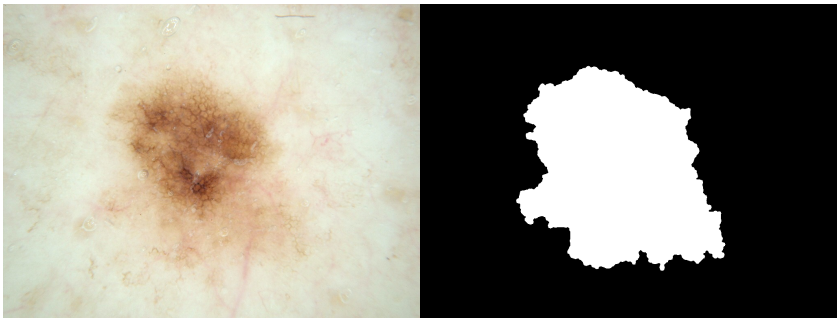


Figure 5.3: Example of a lesion with a boundary that is difficult to distinguish from the surrounding skin tissue. 'ISIC_0011131'.

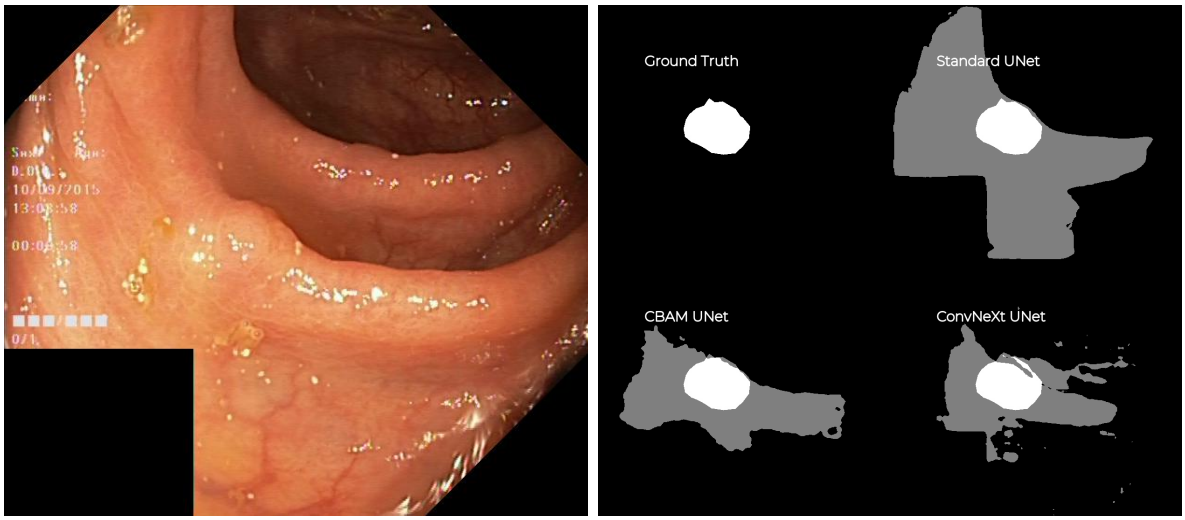


Figure 5.4: 'Image_0024' from the Kvasir-SEG dataset. The polyp in the original image (left) is difficult to distinguish from the intestinal wall, due to similar coloration and brightness.

5.1.3 Strict criteria for EarlyStopping callback

The main mechanism behind regulating the training procedure was the inclusion of the EarlyStopping callback of TensorFlows keras library with a patience of 5, which means the training procedure terminated after the validation loss increased for five consecutive epochs. Examining the fitting curves of figures 4.5 and 4.10 reveal, that most models only trained for a relatively low number of epochs, with most lasting between 10 and 20 epochs. Additionally, the variance of the validation loss seems relatively high. Considering these insights, increasing the patience might be beneficial for the training procedure as it could lead to more iterations over the dataset before it is terminated by EarlyStopping, thus leading to a better-fitted model.

5.2 Challenges

During the process of implementing the proposed architectures as Python classes and integrating them into the MIScnn training and prediction pipelines, we encountered a couple of hurdles that warrant a mention, as they were the motivation behind and the reason for certain design decisions.

5.2.1 Splitting up ConvNeXt U-Net and CBAM U-Net

Since both models modify different aspects of U-Nets' distinct architecture, with the former changing the structure of the downsampling path and the latter adding CBAM to the skip connections, merging both architectures into an attention-augmented ConvNeXt U-Net (AACN) seems an easy task. Such a model has already been proposed in "Attention Augmented ConvNeXt UNet For Rectal Tumour Segmentation" by Wu et al [20]. However, for reasons unknown to us, this paper has since been withdrawn from arXiv, although it is still available on ResearchGate. How this model would look like is documented in 'aacn.py' of the GitHub repository under https://git.rz.uni-augsburg.de/misit-bachelor/ConvNextUnet. In Fact, implementing this architecture was the initial goal of this thesis. Unfortunately, we have been unable to train this architecture, as we encountered numerous out-of-memory (OOM) errors when fitting the model to the training partition of both the ISIC2018 and Kvasir-SEG datasets. It is possible that similar issues were the reasoning for withdrawing "Attention Augmented ConvNeXt UNet For Rectal Tumour Segmentation". Attempts to contact the authors were unfruitful, as we received no answer from the University of central Mongolia, as well as the researchers. We spent a considerable amount of time trying to make AACN work before we decided to split it into separate architectures, both integrating one of the 2 different key aspects respectively, with the intention of gaining a greater understanding of where the problem originated from. Developing a functioning AACN would be an interesting goal for further research.

5.2.2 Custom LayerScale layer

After separating AACN into the architectures that are CBAM U-Net and ConvNeXt U-Net, the culprit of the OOM errors when trying to run the training procedures was identified as the LayerScale class we implemented as a custom tensorflow layer. We prevented these errors by setting the 'layer_scale_init_value' argument of the 'convnext_block_2D' method to 0, which specifies the initial value with which the diagonal matrix that is multiplied with the input is seeded with. This caused the process not to initialize and to skip the LayerScale layer, circumventing any associated issues.

5.3 Alternative architectures

Both the ConvNeXt block and CBAM have been implemented as modular Python methods with the goal of offering them as reusable code blocks and providing the ability to easily integrate them into an existing architecture, given it is also implemented with tensorflow and python. Theoretically, there are numerous different ways to insert these modularities into UNet, a couple of which we want to highlight.

5.3.1 Replacing convolutions in the contracting layer

In ConvNeXt U-Net, we removed the final pooling operation of the contracting layer and replaced it with a ConvNeXt block and a custom downsampling layer. Every step beforehand is identical to the contracting layer of the original model, including the convolutional layers, which use a 3x3 kernel with a stride of 1. They are also responsible for applying the specified number of filters, usually resulting in a change in the number of channels compared to the input. The Convnext block is unable to change the channel dimensions, as tensorflow addons implementation of StochasticDepth relies on the input and the final feature map having the same amount of channels. One possibility would be to keep the first convolution of the contracting layer and replace the second with an additional ConvNeXt block.

5.3.2 Integrate CBAM in different places

In our proposed architecture for CBAM U-Net , CBAM is applied on the feature maps that are concatenated to the outputs of the expanding layers via skip connections. Alternatively, it could be integrated in between the contracting or expanding layers to make further use of the attention mechanism.

5.4 Future Work

As an addendum to the points of interest already mentioned in this chapter, we want to provide an outlook on two major potential avenues for future research and works which aim to iterate upon the results of this paper. This encompasses the usage of both CBAM U-Net and ConvNeXt U-Net on different datasets, possibly along with other models for comparison, and customizing the hyperparameters used in the training and prediction pipelines as well as those of the models themselves.

5.4.1 Using different datasets

We benchmarked the performance of our proposed architectures on the dataset of the 2018 ISIC segmentation challenge and the dataset of polyp images, Kvasir-SEG, by comparing the generated segmentation masks against the ground truth with regards to commonly used metrics in evaluating medical image segmentation models. However, the datasets contain features, such as very general ground truth masks and class imbalance as mentioned in chapter 5.1, which we suspect to have influenced the results in some way. To reproduce the results, verify our findings and reinforce the beneficial impact of CBAM and ConvNeXt on U-Nets performance, further research could be aimed at training ConvNeXt U-Net and CBAM U-Net on other datasets for medical image segmentation, For example a set of 3D medical images.

5.4.2 Experimenting with hyperparameters

The right choice of hyperparameters is an important step in optimizing a machine-learning model for a chosen task. Therefore, fine-tuning these parameters could significantly improve CBAM U-Nets and ConvNext U-Nets performance and ability to generalize well. Examples include the learning rate, the batch size, the values used for weight initialization, or the patience of the EarlyStopping callback, as discussed in 5.1.3. Furthermore, increasing the extent of data augmentation or the inclusion/exclusion of preprocessing functions could prove to be an interesting prospect. Lastly, the fine-tuning of parameters also extends to the ConvNeXt block, as both the LayerScale and StochasticDepth come with additional arguments, those being the initial values of the LayerScale diagonal matrix and the drop chance of layers in StochasticDepth.

6 Conclusion

This bachelor thesis aimed to evaluate and compare the performance of two custom architectures, CBAM U-Net, integrating the Convolutional Block Attention Mechanism (CBAM) into the common U-Net and ConvNeXt U-Net, making use of Vision Transformers (ViT) design decisions, in conjunction with a common U-Net. The study involved training all three models on two distinct medical image datasets and assessing their predictive capabilities with regard to the task of medical image segmentation by comparing the predicted segmentation masks against those that were deemed representative of the ground truth. The experiment made use of the medical image segmentation framework MIScnn by implementing the custom models in a way that made them directly integrable into the interfaces provided by the MIScnn module in order to allow for the setup of training and prediction pipelines.

Throughout the experiments, it became evident that both CBAM U-Net and ConvNeXt U-Net outperformed the baseline U-Net architecture. The enhanced performance observed can be attributed to the unique features and design choices embedded in CBAM U-Net and ConvNeXt U-Net, showcasing their efficacy in capturing complex spatial dependencies and learning intricate patterns within the data.

The comparison of the predictions against the ground truths revealed consistent and significant improvements in segmentation accuracy, highlighting the potential of CBAM U-Net and ConvNeXt U-Net to elevate the performance of convolutional neural networks in image segmentation tasks, potentially elevating them to a competitive state with Vision Transformers. The inclusion of CBAM and the adaptation of the ConvNeXt block operation contributed to a more refined understanding of spatial context, leading to more precise segmentation outcomes in both cases. Additionally, this opens up the prospect of combining the two models into a combined, Attention Augmented ConvNeXt U-Net (AACN).

The application of CBAM U-Net also produced faster training times when averaged over the number of epochs. ConvNeXt U-Net results show a longer training duration, both in terms of total elapsed time and the average time it took for a whole iteration over a dataset

Despite the promising results, it is imperative to acknowledge the need for careful consideration in choosing the appropriate architecture for specific tasks. While CBAM U-Net and ConvNeXt U-Net showcased superior performance in this study, the applicability of these architectures may vary depending on the nature of the data and the objectives of the segmentation task.

In conclusion, the findings of this research suggest that the custom architectures CBAM U-Net and ConvNeXt U-Net, can significantly enhance the performance of U-Net-based models in image segmentation tasks. The success observed in this study prompts further exploration and application of these architectures in other scenarios, with the potential for advancements in the field of medical image segmentation.

References

[1] R. Wang, T. Lei, R. Cui, B. Zhang, H. Meng, and A. K. Nandi, "Medical image segmentation using deep learning: A survey," *IET Image Processing*, vol. 16, no. 5, pp. 1243–1267, Jan. 2022, ISSN: 1751-9667. DOI: 10.1049/ipr2.12419. [Online]. Available: http://dx.doi.org/10.1049/ipr2.12419 (cit. on p. 7).

[2] S. Indolia, A. K. Goswami, S. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network- a deep learning approach," *Procedia Computer Science*, vol. 132, pp. 679–688, 2018, International Conference on Computational Intelligence and Data Science, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.069. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918308019 (cit. on p. 7).

[3] R. Azad, E. K. Aghdam, A. Rauland, *et al.*, *Medical image segmentation review: The success of u-net*, 2022. arXiv: 2211.14830 [eess.IV] (cit. on p. 7).

[4] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A convnet for the 2020s*, 2022. arXiv: 2201.03545 [cs.CV] (cit. on pp. 7, 10, 11).

[5] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: 2103.14030 [cs.CV] (cit. on p. 7).

[6] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, *Cbam: Convolutional block attention module*, 2018. arXiv: 1807.06521 [cs.CV] (cit. on pp. 7, 10, 11).

[7] A. Peláez-Vegas, P. Mesejo, and J. Luengo, *A survey on semi-supervised semantic segmentation*, 2023. arXiv: 2302.09899 [cs.CV] (cit. on p. 8).

[8] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation," *Annual Review of Biomedical Engineering*, vol. 2, no. 1, pp. 315–337, 2000, PMID: 11701515. DOI: 10.1146/annurev.bioeng.2.1.315. eprint: https://doi.org/10.1146/annurev.bioeng.2.1.315. [Online]. Available: https://doi.org/10.1146/annurev.bioeng.2.1.315 (cit. on p. 8).

[9] C. GmbH, *Semi-automatic medical image segmentation*, Online; accessed 09-December-2023, 2011. [Online]. Available: https://www.youtube.com/watch?v=7wCC2NaVLjs (cit. on p. 8).

[10] R. Choi, A. Coyner, J. Kalpathy-Cramer, M. Chiang, and J. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science technology*, vol. 9, p. 14, Feb. 2020. DOI: 10.1167/tvst.9.2.14 (cit. on pp. 8, 9).

[11] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 645–657, 2017. DOI: 10.1109/TGRS.2016.2612821 (cit. on p. 9).

[12] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, 2015. arXiv: 1411.4038 [cs.CV] (cit. on p. 9).

[13] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV] (cit. on pp. 9, 10, 15).

[14] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, vol. 9, pp. 82 031–82 057, 2021. DOI: 10.1109/access.2021.3086020. [Online]. Available: https://doi.org/10.1109%2Faccess.2021.3086020 (cit. on p. 10).

[15] D. Müller and F. Kramer, "MIScnn: A framework for medical image segmentation with convolutional neural networks and deep learning," *BMC Medical Imaging*, vol. 21, no. 1, Jan. 2021. DOI: 10.1186/s12880-020-00543-7. [Online]. Available: https://doi.org/10.1186%2Fs12880-020-00543-7 (cit. on p. 12).

[16] D. Müller, I. Soto-Rey, and F. Kramer, *Towards a guideline for evaluation metrics in medical image segmentation*, 2022. arXiv: 2202.05273 [eess.IV] (cit. on p. 12).

[17] I. Düntsch and G. Gediga, "Confusion matrices and rough set data analysis," *Journal of Physics: Conference Series*, vol. 1229, no. 1, p. 012 055, May 2019, ISSN: 1742-6596. DOI: 10.1088/1742-6596/1229/1/012055. [Online]. Available: http://dx.doi.org/10.1088/1742-6596/1229/1/012055 (cit. on p. 12).

[18] N. Codella, V. Rotemberg, P. Tschandl, *et al.*, *Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)*, 2019. arXiv: 1902.03368 [cs.CV] (cit. on pp. 13, 14).

[19] D. Jha, P. Smedsrud, M. Riegler, *et al.*, "Kvasir-seg: A segmented polyp dataset," Jan. 2020 (cit. on pp. 13, 14).

[20] H. Wu, J. Wang, X. Wang, *et al.*, *Attention augmented convnext unet for rectal tumour segmentation*, 2022. arXiv: 2210.00227 [eess.IV] (cit. on pp. 16, 17, 30).

[21] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, *Going deeper with image transformers*, 2021. arXiv: 2103.17239 [cs.CV] (cit. on pp. 16, 17).

[22] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, *Deep networks with stochastic depth*, 2016. arXiv: 1603.09382 [cs.LG] (cit. on p. 17).

[23] D. Müller, D. Hartmann, P. Meyer, F. Auer, I. Soto-Rey, and F. Kramer, *Miseval: A metric library for medical image segmentation evaluation*, 2022. arXiv: 2201.09395 [cs.CV] (cit. on p. 19).

[24] D. Guo, Y. Pei, K. Zheng, H. Yu, Y. Lu, and S. Wang, "Degraded image semantic segmentation with dense-gram networks," *IEEE Transactions on Image Processing*, vol. 29, pp. 782–795, 2020. DOI: 10.1109/TIP.2019.2936111.

[25] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.