Deep Learning Seminar Final Project DoubleAU-Net: Attention based DoubleU-Net for Medical Image Segmentation

Tel Aviv University, School of Electrical Engineering, Deep Learning Seminar, 0510-7255, Spring 2022

Roe Barlev
Roebarlev@mail.tau.ac.il

Ron Dudkman
Rondudkman@mail.tau.ac.il

***Abstract*—The process of labeling each pixel of an image with its corresponding class is called semantic image segmentation. A popular strategy for solving medical image segmentation tasks is encoder-decoder based approach, like U-Net and its variants. We based our work mainly on the DoubleU-Net architecture. First, we converted the original code from TensorFlow to PyTorch library, and in order to implement DoubleU-Net we had to understand the architecture's components and pipline. DoubleU-Net is a combination of two U-Net architectures stacked on top of each other. The first U-Net uses a pre-trained VGG-19 as the encoder, and between the encoder-decoder an Atrous Spatial Pyramid Pooling (ASPP) block was used to capture contextual information within the network. Another U-Net was added at the bottom to capture more semantic information efficiently. After implementing the DoubleU-Net in a different library we searched for a way to improve the model, when we encountered the attention mechanism, which we then added to the DoubleU-Net hence, our modified architecture is called DoubleAU-Net. Interpolation blocks were also used to overcome resources problem in the work environment we used, Google Colab. We have evaluated DoubleAU-Net using two medical segmentation datasets, including colonoscopy and microscopy data. Experiments on the CVC-ClinicDB and the 2018 Data Science Bowl challenge dataset demonstrate our DoubleAU-Net improvement in some metrics compared to U-Net and DoubleU-Net. Using interpolation blocks also raises interesting results and discussion about when to use them and how they affect model run-time-performance trade-off.**

I. Introduction

Medical image segmentation is the task of labeling each pixel of an object of interest in medical images. Medical image segmentation helps clinicians focus on a particular area of the disease and extract detailed information for a more accurate diagnosis. The key challenges associated with medical image segmentation are the unavailability of a large number of annotated images, lack of high quality labeled images for training, low image quality, lack of a standard segmentation protocol, and a large variations of images among patients. Convolutional Neural Networks (CNNs) have shown state of-the-art performance for automated medical image segmentation. For semantic segmentation tasks, one of the earlier Deep Learning (DL) architecture trained end-to-end for pixel-wise prediction is a Fully Convolutional Network (FCN).

Our work is based on DoubleU-Net [1] which suggests an architecture based on the popular image segmentation network, U-Net [4]. The authors of DoubleU-Net published their implementation using TensorFlow library on GitHub[1]. We implemented their architecture using PyTorch library and received similar results as the DoubleU-Net. In order to improve the network results, we based on "U-Net and its Variants" [5] to modify the DoubleU-Net architecture by adding attention units, regularization techniques and optimize hyperparameters.

The paper is organized into seven sections. Section II provides an overview of the similarities and differences between our work and related works. In Section III, we describe the data we used in our project. Section IV we discuss our approach for implementing and improving DoubleU-Net work. Section V describes the experiments and their results. Conclusions and key results are provided in Section VI. Finally, we include more details about our code in Section VII.

II. Related Work

Our work is based on two main articles. The first one is the DoubleU-Net [1] that suggested an architecture that is based on U-net [4] architecture. U-Net is a popular image segmentation architecture trained end-to-end for pixel-wise prediction. Inspired by the success of U-Net and its variants for medical image segmentation, the paper [1] propose an architecture that uses modified U-Net and VGG-19 in the encoder part of the network. Because they use two U-Net architectures in the network, they term the architecture as "DoubleU-Net". In our work we implemented the whole DoubleU-Net paper [1] (Architecture and Pipeline) using PyTorch in Google Colab environment. Afterwords we added some regularization techniques, used different hyperparameters and added new ideas to the net as we will describe later on. The second article we based our work on is "U-Net and its Variants" [5] which reviews different techniques to improve U-Net. One of the variants that is discussed in the paper and we found relevant is the Attention U-Net, which suggested to add attention units

[1] https://github.com/DebeshJha/2020-CBMS-DoubleU-Net

in the expansive path of the network. We read more about the attention units [2] and used the same idea in our architecture and added them to the DoubleU-Net in both of the expansive paths so basically we have two attention U-Nets, or in other words Double Attention U-Net (DoubleAU-Net).

III. Data

To evaluate the effectiveness of our suggested network, we have used two publicly available datasets from medical domain, the same as paper [1] used so we can make a comparison between the models.

A. 2018 Data Science Bowl challenge dataset

2018 Data Science Bowl challenge dataset[2] contains a large number of segmented nuclei images. The images were acquired under a variety of conditions and vary in the cell type, magnification, and imaging modality (brightfield vs. fluorescence). Each image of this dataset got his GT (Ground Truth) image split into multiple images (each image contains one segment data). Figure 1 shows the process that we have made in order to get one final GT image for each of the original images.

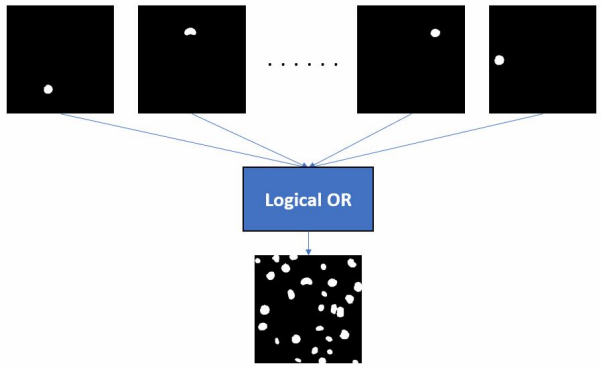


Fig. 1: Preprocessing the ground truth images of the 2018 Data Science Bowl challenge dataset

B. CVC-ClinicDB dataset

CVC-ClinicDB[3] is an open-access dataset of 612 images with a resolution of 384×288 from 31 colonoscopy sequences. It is used for medical image segmentation, in particular polyp detection in colonoscopy videos.

In order to increase our datasets examples, we applied data augmentations which will be described later on. More information about the datasets are presented in Table I. All of the datasets are clinically relevant during diagnosis, and therefore, their segmentation can be crucial for patient outcome.

[2]https://www.kaggle.com/competitions/data-science-bowl-2018/data
[3]https://www.kaggle.com/datasets/balraj98/cvcclinicdb

TABLE I: Summery of biomedical segmentation dataset used in our experiments

Dataset	No. of images	Input size	Application
CVC-ClinicDB	612	384×288	Colonoscopy
2018 Data Science Bowl challenge	670	256×256	Nuclei

IV. Methods

U-Net is a popular image segmentation architecture trained end-to-end for pixel-wise prediction. The U-Net architecture consists of two parts, namely, analysis path and synthesis path. In the analysis path, deep features are learned, and in the synthesis path, segmentation is performed based on the learned features. Additionally, U-Net uses skip connections operation. The skip connection allows propagating dense feature maps from the analysis path to the corresponding layers in the synthesis part. In this way, the spatial information is applied to the deeper layer, which significantly produces a more accurate output segmentation map. Thus, adding more layers to the U-Net will allow the network to learn more representative features leading to better output segmentation masks.

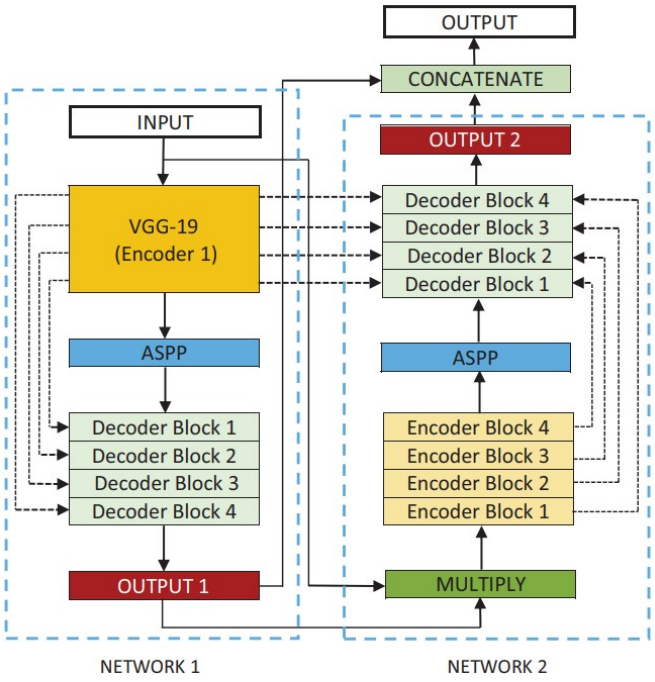


Fig. 2: Block diagram of DoubleU-Net [1] architecture

A. The DoubleU-Net Architecture

Figure 2 shows an overview of the DoubleU-Net architecture. As seen from the figure, DoubleU-Net starts with a pre-trained VGG-19 as encoder sub-network, which is followed by decoder subnetwork. What distinguishes DoubleU-Net from U-Net in the first network (NETWORK 1) is the use of VGG-19 marked in yellow, ASPP marked in blue, and decoder block marked in light green. The squeeze-and-excite block is used in the encoder of NETWORK 2

and decoder blocks of NETWORK 1 and NETWORK 2. An element-wise multiplication is performed between the output of NETWORK 1 with the input of the same network. The difference between DoubleU-Net and U-Net in the second network (NETWORK 2) is only the use of ASPP and squeeze-and-excite block.

1) Encoder: The first encoder in DoubleU-Net (encoder1) uses pre-trained VGG-19. This fact forced us to understand how the VGG-19 network is built in order to know from which layer to extract the encoded feature maps and where to connect them in the DoubleU-Net architecture. Figure 3 demonstrates the relations between the VGG-19 extracted features (from layers: 3, 8, 17, 26, 35) and DoubleU-Net decoder blocks. The second encoder (encoder2), is built from scratch. Each encoder block in the encoder2 performs two 3 × 3 convolution operation, each followed by a batch normalization. The batch normalization reduces the internal co-variant shift and also regularizes the model. A Rectified Linear Unit (ReLU) activation function is applied, which introduces non-linearity into the model. This is followed by a squeeze-and- excitation block, which enhances the quality of the feature maps. After that, max-pooling is performed with a 2 × 2 window and stride 2 to reduce the spatial dimension of the feature maps.

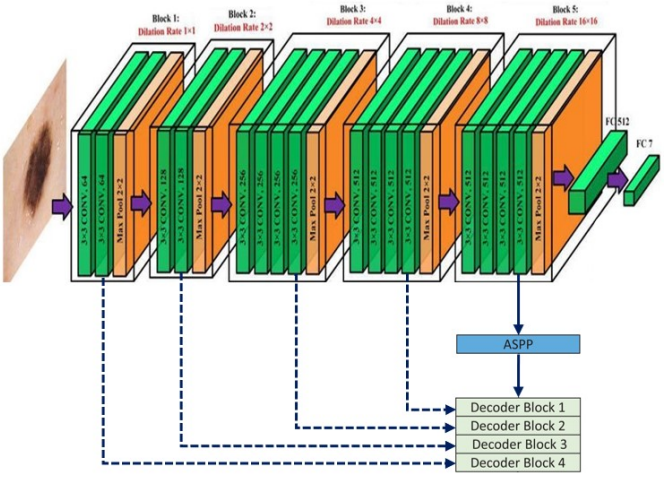


Fig. 3: Block diagram of VGG-19 architecture [3], with relevant connections to DoubleU-Net components

2) ASPP: Atrous Spatial Pyramid Pooling (ASPP) is a semantic segmentation module for resampling a given feature layer at multiple rates prior to convolution (see Figure 4). This amounts to probing the original image with multiple filters that have complementary effective fields of view, thus capturing objects as well as useful image context at multiple scales. Rather than actually resampling features, the mapping is implemented using multiple parallel atrous convolutional layers with different sampling rates.

3) Decoder: As shown in Figure 2, the authors used two decoders in the entire network. The decoders were implemented as in U-Net, each block in the decoder performs

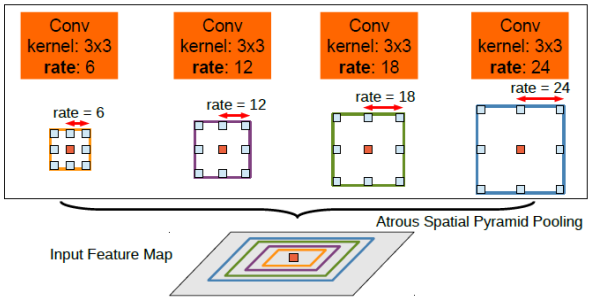


Fig. 4: The ASPP process

a 2 × 2 bi-linear up-sampling on the input feature, and then concatenate the appropriate skip connections feature maps from the encoder to the output feature maps. The first decoder uses skip connections only from the first encoder, but the second decoder, uses skip connections from both of the encoders, which differs and maintains the spatial resolution and enhance the quality of the output feature maps. Here is where another difference from U-Net takes place, instead of using only two 3 × 3 convolution operation followed by ReLu, they added batch normalization after each convolution and at the end of the process they applied squeeze and excitation block. At last, a convolution layer with a sigmoid activation function is applied, which is used to generate the mask for the corresponding modified U-Net.

4) Squeeze And Excite: Squeeze and Excitation (SE) is a block that consists of multiple layers as shown in Figure 5. The process starts with global pooling which down sample the spatial information in the input data from H x W x C to 1 x 1 x C, continuing with fully connected and ReLU layers that down sample the data in the feature maps domain, fully connected layer is used again and sigmoid at the end. We rescale the sigmoid output to the input size and multiplying it with the input. On that way we get the input data with feature maps weighted based on their importance.
We use the SE block as part of the convolutional block that is being used in both encoder and decoder blocks. It improves the representational power of the network by enabling it to perform dynamic channel-wise feature re-calibration. The squeeze-and-excite block also reduces the redundant information and passes the most relevant information.

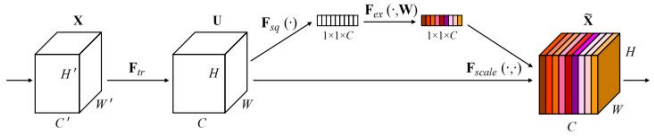


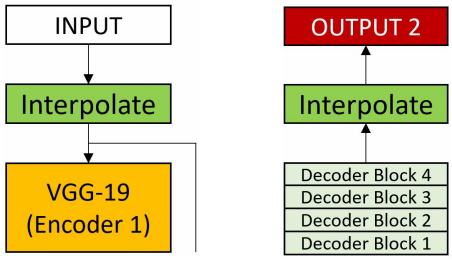
Fig. 5: Squeeze and excite process

B. Our Modified Architecture

We implemented DoubleAU-Net using python in Google Colab environment. We had a few challenges while doing

this work, our main challenge was Google Colab lack of resources and time limits of sessions. The lack of resources has prevented us from using the same batch size as the original paper [1] suggested, and running an insufficient number of epochs. Another challenge was converting all of the authors work from TensorFlow library to PyTorch. We also implemented some improvements such as attention units, regularization techniques and more.

1) Interpolation Block: We thought about different ways to overcome Google Colab lack of resources issue. First, we came with the idea of resizing all of our images by 2 in each dimension (reducing the total number of pixels by 4). The results on small images were good, but when testing it on original size images the results were bad. Then we came up with the idea of using interpolation blocks. The idea is to resize images before processing them to the network (Figure 6a), and resize them back to their original size right before calculating the loss and other metrics (Figure 6b). The motivation behind this idea is to train a model, which produces small mask predictions, to get minimal loss after the interpolation block (at the output). In other words, the model learns to make predictions according to the interpolation operation that will come after the prediction. On that way, we reduce the computing power needed to train the model, reducing the runtime of epochs, and keeping the backpropagation calculations based on original image size.



(a) Input (b) Output

Fig. 6: The interpolation blocks at the input (a) and output (b)

2) Attention Units: Paper [5] suggested different ways to improve U-Net, one of them is the attention U-Net (see Figure 7). An often-desirable trait in an image processing network is the ability to focus on specific objects that are important while ignoring unnecessary areas. The attention U-Net achieves this by making use of the attention gate. An attention gate is a unit that trims features that are not relevant to the ongoing task. The attention unit is useful in encoder-decoder models since it can provide localized classification information as opposed to global classification. We wanted to examine how the attention gate will affect the DoubleU-Net architecture, since the attention units improved U-Net results we believe it will improve DoubleU-Net results as well, since they are both encoder-decoder models. Each layer in both of the expansive paths has an attention gate through which the corresponding features from the contracting paths must pass through before the features are concatenated with the upsampled features in the expansive paths. Note that in the second decoder (decoder2) we used attention gates on both in-going skip connections, one from the first encoder and the second from encoder 2, that so we emphasise the relevant parts in both skip connections. Moreover repeated uses of the attention gate after each layer significantly improves segmentation performance without adding excessive computational complexity to the model. Another reason we chose to use attention in our architecture is because we already implemented feature weighting methods (squeeze and excite), so we wanted to add a mechanism that highlights the relevant activations in the spatial domain (spatial weighting method).

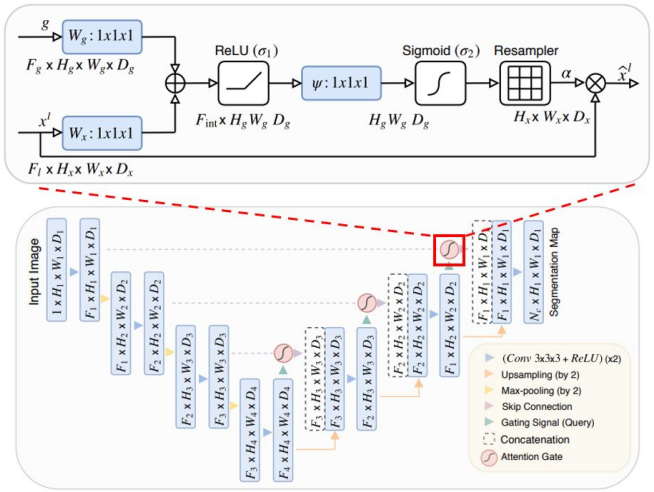


Fig. 7: Block diagram of the attention units

3) Regularization Techniques: Regularization is often used as a solution to the overfitting problem in Machine Learning. Two common causes for overfitting are complex models that starts modeling the noise in the training data and when the training data is relatively small which may lead for model that fails to generalize. As we described earlier, our data we use is pretty small, we had two main solutions for this problem. The first, using data augmentations which will be elaborated in further sections. The second solution we implemented is dropout. During training, some number of layer outputs are randomly ignored or "dropped out." This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different "view" of the configured layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistic take on more or less responsible for the inputs. This conceptualization suggests that perhaps dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn

making the model more robust.

4) Hyperparameters: We tried to keep our hyperparameters as close as we can to the original work [5] hyperparameters. As we mentioned before, Google Colab has limited resources thus we could not train our models as much epochs as the paper suggested. The scheduler and early stop parameters were tuned relatively to the number of epochs that we used. The size of our input images and batch size are the same as the paper used thanks to our interpolation technique that helped us to reduce computing power. We also changed the learning rate for some models which will be specified later on. Note that all of our hyperparameters can be easily adjusted through the configuration code that we wrote.

V. Experiments

In this section, we present the evaluation metrics, discuss about the experiments that we have performed to demonstrate our different approaches and how they improved the results on both datasets that we have worked with. We compare the proposed model DoubleAU-Net with the DoubleU-Net [1] results by using the same data augmentation techniques as they used. We worked with Google Colab using the Tesla T4 GPU. Each training session lasted on average for 4 hours using the GPU, sometimes we managed to have maximum time of 6 hours of session run time.

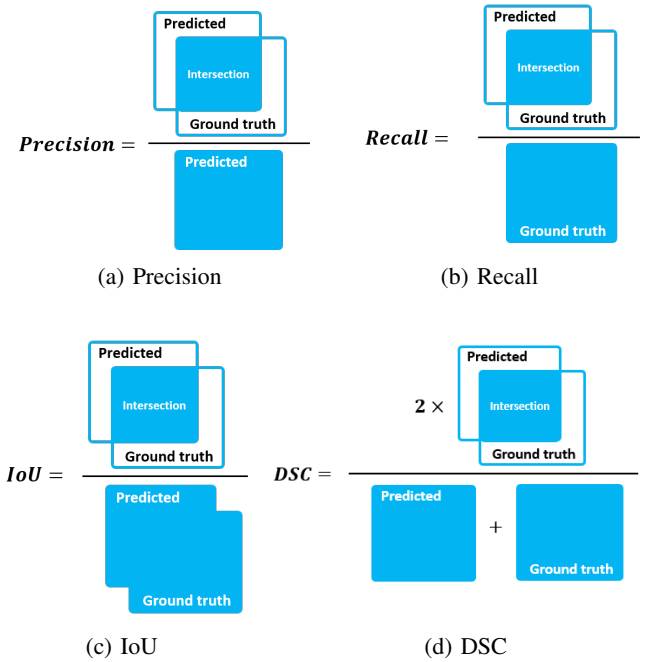
We split each of the datasets into train, validation and test set (80% for train, 10% for validation, 10% for test) just as paper [1] suggested.

A. Augmentations

Working with medical datsets is not an easy task, medical datasets are challenging to obtain and annotate. Moreover most existing datasets have only a few samples, which makes training Deep Learning models on challenging. One potential solution to the challenge of data insufficiency, is to use data augmentation techniques that increase the number of samples during training. We applied different data augmentation methods to our training set, such as center crop, random rotation, transpose, elastic transform, etc. We used the same augmentations as the autours [1] used, more details can be found in their GitHub repository. In total, each image from the training set was augmented (with some probability) 25 times, which gives us 26 variants (including the original one) for a singe image. The same augmentation techniques were applied for both datasets (CVC-ClinicDB and 2018 Data Science Bowl challenge dataset).

B. Metrics

The metrics we used are the same as in DoubleU-Net [1], the reason is simple, we want to compare our implementation to theirs and see if we managed to improve the results. DoubleAU-Net is evaluated on the basis of Sørensen–Dice coefficient (Figure 8d), mean Intersection over Union (Figure 8c), Precision (Figure 8a) and Recall (Figure 8b). The loss function (Equation 1) is a combination of Dice loss



(a) Precision (b) Recall

(c) IoU (d) DSC

Fig. 8: Metrics

(Equation 2) and standard Binary Cross Entropy (BCE) loss (Equation 3) that is generally the default for segmentation models. Combining the two methods allows for some diversity in the loss, while benefiting from the stability of BCE.

$$Loss = BCE + Dice \tag{1}$$

$$Dice\ loss = 1 - DSC \tag{2}$$

$$BCE = -\frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} y_{i,j} \cdot \log \hat{y}_{i,j} + (1 - y_{i,j}) \cdot \log(1 - \hat{y}_{i,j}) \tag{3}$$

Where $H, W, y_{i,j}, \hat{y}_{i,j}$ stands for image Height, Width, pixel value in the ground truth and prediction images respectively.

C. Comparison on CVC-ClinicDB dataset

Table II and Figure 9 presents the results achieved using the CVC-ClinicDB dataset. We have compared our work with U-Net [4] and DoubleU-Net [1]. We used learning rate of 1e-4, batch size of 16, loss function as described in Equation 1 and NAdam optimizer.

At first we implemented the DoubleU-Net in PyTorch, but as we explained before there was resources problem, so we had to train it with batch 4. This first experiment was just to see if we implemented correctly the DoubleU-Net architecture, and as shown in Table II we got pretty solid results compare to the original paper (except Recall which we got higher result). The first experiment was our base of compare to know if we got better each experiment we conducted.

We wanted to use higher batch size than 4 and run more epochs in one session, so then we came up with the idea of interpolate the input and the output, hence we called the results by the name "DoubleU-Net + Interpulation" in the forth line in Table II. The forth method (second experiment) gave us

TABLE II: Result comparison on CVC-ClinicDB

Method	DSC	mIoU	Recall	Precision
U-Net [1]	0.8781	0.7881	0.7865	0.9329
DoubleU-Net [1]	0.9239	0.8611	0.8457	**0.9592**
DoubleU-Net*	0.9372	0.8914	0.9416	0.9426
DoubleU-Net + Interpolation	0.9370	0.8909	0.9283	0.9579
DoubleAU-Net*	0.9368	0.8925	0.9337	0.9528
DoubleAU-Net + Interpolation	**0.9439**	**0.9007**	**0.9464**	0.9504

*Batch size is 4

a bit unexpected results, since we got similar results to the previous experiment but we used resized images at the input. Then, we duplicated those 2 experiments to our DoubleAU-Net architecture.

Table II shows all the experiments we had conduct, and at first look we can see that the attention units did improve the net performance (with and without interpolation), and our suggested DoubleAU-Net got the best results with the interpolation for DSC, mIoU, and Recall metrics. The Precision of the original DoubleU-Net remain unbeatable but we got close result in the Precision metric, so we believe that with some more epochs and learning rate scheduling we can do better. Also, in the original challenge that this dataset is related to, the evaluation metrics were mIoU and Dice loss, which as shown got improved with our DoubleAU-Net.

Figure 9 also compares between DoubleAU-Net with and without interpolation, as we can see the interpolation method achieved better segmentations around the edges of the images and had less false segments. On the other hand the segmentations themselves look a bit rough, pixelated. After experimenting on CVC-CliniDB dataset, we chose to test our suggested architecture (DoubleAU-Net) on a different dataset. Graphs of loss and different metrics results during training can be seen in Figure 11

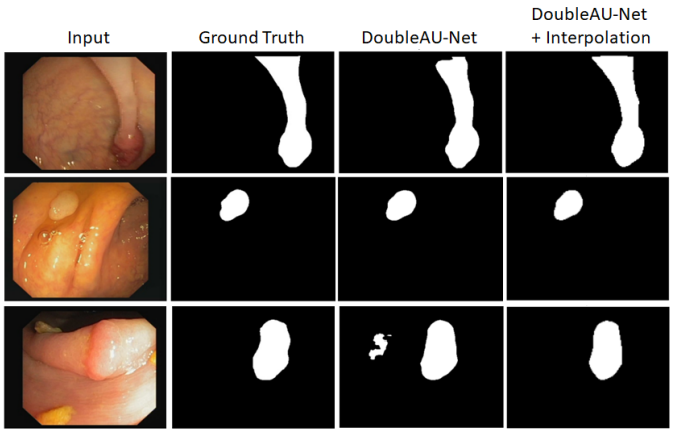


Fig. 9: Results of DoubleAU-Net on CVC-ClinicDB dataset

D. Comparison on 2018 Data Science Bowl challenge dataset

Table III and Figure 10 presents the results achieved using the 2018 Data Science Bowl challenge dataset. We have compared our work with U-Net [4] and DoubleU-Net [1]. We

TABLE III: Result comparison on 2018 Data Science Bowl challenge dataset

Method	DSC	mIoU	Recall	Precision
U-Net [1]	0.7573	**0.9103**	-	-
DoubleU-Net [1]	0.9133	0.8407	0.6407	**0.9496**
DoubleAU-Net	**0.9364**	0.8832	**0.9353**	0.9403
DoubleAU-Net + Interpolation	0.9167	0.8492	0.9194	0.9174

used learning rate of 1e-4 and batch size of 16, loss function as described in Equation 2 and Adam optimizer. At first, we trained DoubleAU-Net with the interpolation technique just as we did in the CVC-ClinicDB dataset, assuming it will get the highest results for this dataset as well. This model achieved improvement in the DSC, mIOU and recall metrics comparing it to the DoubleU-Net results (see Table III), but not the highest results that we could get. We tried to train DoubleAU-Net without the interpolation technique, meaning processing the original sized images through the network. We trained it for 20 epochs and achieved our best results including improvements in all metrics compare to DoubleAU-Net with interpolation. U-Net beat all the methods in a relatively big margin in the mIoU metric, although we managed to improve compare to DoubleU-Net. The disadvantages of our best model are the use of more GPU memory while training and the time it takes to complete an epoch due to more calculations per epoch. Graphs of loss and different metrics results during training can be seen in Figure 12.

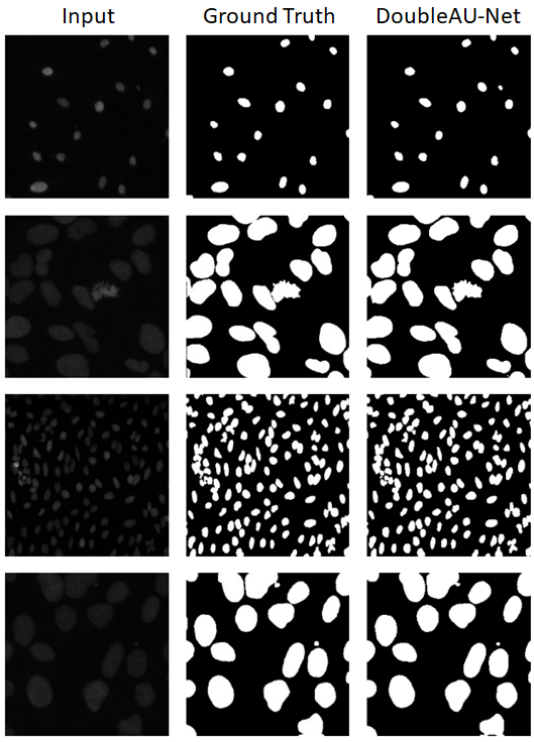


Fig. 10: Results of DoubleAU-Net on 2018 Data Science bowl challenge dataset

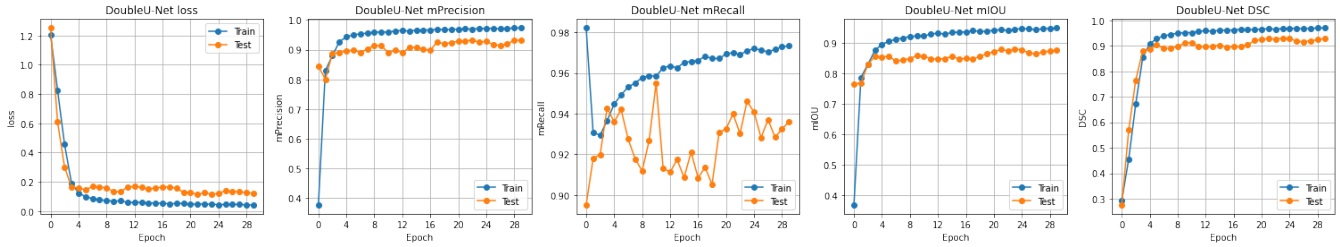


Fig. 11: Graphs of DoubleAU-Net + Interpolation on CVC-ClinicDB dataset

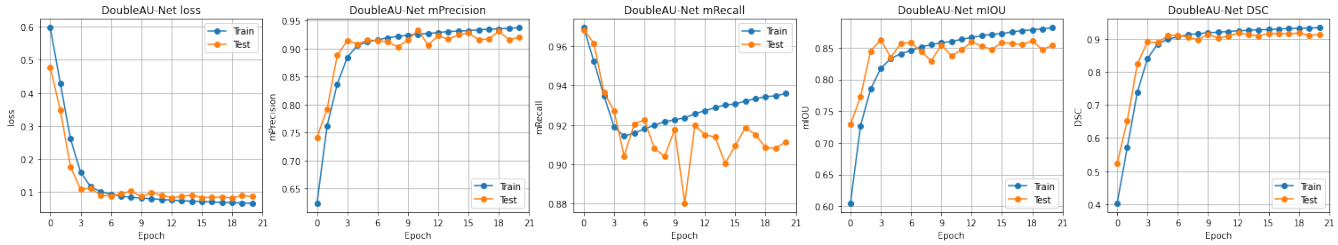


Fig. 12: Graphs of DoubleAU-Net on 2018 Data Science bowl challenge dataset

E. Run-time - Performance tradeoff

There is a tradeoff between the runtime of training session and model performence. On the one hand, applying the interpolation technique reduced the computing power and the epoch runtime. On the other hand, resizing an image before processing it to the network meaning losing information, and as we saw the results in Table III, DoubleAU-Net with interpolation gets lower results than DoubleAU-Net with no interpolation.

In Table IV and Table V we can see the time it takes to run one epoch in different experiments that we conducted.

We succeeded to train DoubleAU-Net with no interpolation using 2018 Data Science bowl challenge dataset for only 20 epochs as we can see in the graphs in Figure 10 and it was enough to get high results.

We couldn't do the same with the CVC-ClinicDB dataset and train it without the interpolation unless we reduce batch size to 4. Table IV shows that CVC-ClinicDB dataset time per epoch are higher than the 2018 Data Science bowl challenge dataset time per epoch (when observing equivalent methods), the reason for that is because CVC-ClinicDB dataset images are bigger. It forced us to choose between lower batch size or using interpolation in order to achieve good results.

TABLE IV: Runtime per epoch using CVC-ClinicDB dataset

Method	Epoch Runtime [Minutes]
DoubleU-Net*	20:51
DoubleU-Net + Interpolation	5:02
DoubleAU-Net*	25:40
DoubleAU-Net + Interpolation	6:13

*Batch size is 4

TABLE V: Runtime per epoch using 2018 Data Science bowl challenge dataset

Method	Epoch Runtime [Minutes]
DoubleAU-Net	14:50
DoubleAU-Net + Interpolation	4:20

VI. Conclusions

In this paper we discussed about our work, implementing the DoubleU-Net [1] and suggesting a new modified architecture named DoubleAU-Net.

The implementation part was not easy and took us some time. We had to learn how to use PyTorch library and also about TensorFlow library since the original work was implemented in TensorFlow and we had to transfer it into our PyTorch pipeline.

We think that the results we achieved are pretty good, considering the lack of resources that we had. The Interpolation technique that we used in order to train the model with bigger batch size did improve the results when we used the CVC-ClinicDB dataset. When we used the 2018 Data Science bowl challenge dataset, the results were different. We achieved better performance without the interpolation technique and we assume that's because of the segmentation shapes of the data (see Figure 10), which are smaller and contain much more segments than the CVC-ClinicDB dataset. As we seen in Figure 9, the interpolation makes the predicted segmentations look a bit pixelated (especially around the edges of the segment), this effect also happens in the 2018 Data Science bowl challenge dataset. We assume that since the 2018 Data Science bowl challenge dataset contains much more smaller segmentations then CVC-ClinicDB, we'll get a lot of false positives and false negatives. The interpolation expands the

image dimensions and fill the gaps with values corresponding to the pixel's neighbours, thus around the edges of a segments we have a higher probability for false predictions. This fact may arise a counter claim that the model has a lower upper bound when using the interpolation technique, means that the model not necessarily takes in account the interpolation operation after the prediction, which can make sense in a way since we reduced the amount of information the model learns from (smaller images). We may suggest that the interpolation method is more suitable for data with small amount of big sized continues segmentions in a single image, and not for data with large amount of small segmentations.

In general, it would be better to use batch 16 without the interpolation block, but we found it useful to use interpolation when you are lacked of resources, and as explained before there is a trade-off between model runtime and performance.

In order to improve our results we had to research about new techniques, and when we added attention units to our model, we achieved better results in both dataset as we can see in Table II and Table III.

Other ways to improve our results was modifying some hyperparameters, and add regularization techniques like dropout as we learnt in class.

During the work we learnt a lot about different methods that has been used in DoubleU-Net implementation like squeeze and excite, ASPP, different metrics, dice loss function and about segmentation networks in general like U-Net.

In the future, we can try to implement some other variants of U-Net [5] in the DoubleU-Net architecture, train for more epochs, schedule better the learning rate, change different hyperparameters and maybe modify the architecture a little and adjust it to different datasets. We also thought about post-processing techniques that can be worth to test like ensemble modeling, applying some test time augmentation (TTA) and more.

VII. Appendix

All of our code can be found in our GitHub[4]. repository. We added "README" text file that will help to understand the code and how to run it properly.

REFERENCES

[1] Debesh Jha et al. "DoubleU-Net: A deep convolutional neural network for medical image segmentation". In: *2020 IEEE 33rd International symposium on computer-based medical systems (CBMS)*. IEEE. 2020, pp. 558–564.

[2] Ozan Oktay et al. "Attention u-net: Learning where to look for the pancreas". In: *arXiv preprint arXiv:1804.03999* (2018).

[3] Md Aminur Rab Ratul et al. "Skin lesions classification using deep learning based on dilated convolution". In: *BioRxiv* (2020), p. 860700.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[5] Nahian Siddique et al. "U-Net and its variants for medical image segmentation: A review of theory and applications". In: *Ieee Access* 9 (2021), pp. 82031–82057.

[4]https://github.com/RDudkman/DoubleAU-Net