Implementation and Evaluation of a Nested, Variable-Depth UNet++ Model Architecture for Medical Imaging Segmentation

Feilong Meng
feilongm@umich.edu

Peijing Li
peijli@umich.edu

Yongxiang Zhao
zyxleo@umich.edu

Abstract

For the final project of the EECS 442 course, we implemented and evaluated the performance of a variable-depth UNet++ architecture in the context of medical imaging segmentation. UNet++ essentially is an encoder-decoder network where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways, aiming at reducing the semantic gap between the feature maps of the two sub-networks. We implemented a variation of this architecture where the code base enables the user to modify the depth of the network. We also trained and tested different instances of this network with varying levels of depth on a dataset of retinal imaging scans, comparing their time and space complexities and training and validation accuracies, not only with one another but also with a fixed-depth baseline model. It can be observed that as the depth in layers of the U-Net++ network increases, the time complexity of training the network increases linearly and the space complexity of the network in terms of model parameters increases exponentially; however, only marginal increases in training and validation accuracies can be gained after the model is more than four layers deep.

1. Introduction

Image segmentation plays a crucial role in many medical imaging applications, by automating or facilitating the delineation of anatomical structures and other regions of interest [22]; such segmentation tasks often demand a higher level of accuracy than what is desired in natural images [31]. The state-of-the-art models for medical image segmentation are variants of U-Net and fully convolutional networks (FCN) [30]. In our final project for EECS 442, we will attempt to implement and evaluate a more adaptive and efficient architecture, UNet++, for for medical imaging segmentation [31], which itself contains enhancements to the vanilla U-Net that is already covered in the EECS 442 course [23]. Our primary contributions will be that our implementation of UNet++ is able to adapt its network depth according to user inputs, thus enabling more versatile usage of UNet++ in the segmentation of images with varying scales of features of interest. We shall also utilize our varying depth UNet++ implementation to evaluate the relationships between the depths of UNet++ networks and their training and validation performances.

2. Related Work

2.1. Medical Imaging Segmentation

Classical methods for medical imaging segmentation can be divided into the following eight categories, as described by Pham *et al*. [22] in their review article. (a) **Thresholding** of the image intensities at each pixel to create binary partitions [24]. (b) **Region growing** extracts image regions based on predefined criteria of intensity information and/or edges of adjacent pixels [9]. (c) Supervised **classifiers**, including k-nearest neighbor, Parzen window, and Bayes classifiers [22], using labeled training data as supervision. (d) Unsupervised **clustering** algorithms [22], including K-means [4], C-means [2], and the expectation-maximization algorithms [14, 16]. (e) **Markov random field models** for spatial interactions between neighboring pixels [15]. (f) **Artificial Neural Networks** for classifying and/or clustering, including Convolutional Neural Networks [28], Fully Convolutional Networks [19], UNets [23], Convolutional Residual Networks [10], and Recurrent Neural Networks [11]. (g) **Deformabale parametric models** to fit boundary curves or surfaces [17]. (h) **Atlas-guided approaches** using a reference frame or a template for classifying pixels [5, 1]

Artificial Neural Networks (ANN), among all categories of segmentation algorithms, have garnered the most interest in recent years due to their generalization abilities for complex features over large amounts of data [29, 7]. We shall focus on U-Nets within the category of ANNs for the remainder of our study.

2.2. U-Net and UNet++

Ronneberger *et al*. originally proposed U-Nets as a further development of deep convolutional networks, the latter of which had already "outperformed the state of the art

in many visual recognition tasks" [8, 13]. The idea of U-Nets relies on "a contracting path to capture context and a symmetric expanding path that enables precise localization" [23]; features of each layer of the contracting path are fused with that in the expanding path, which allows the network to propagate contextual information indirectly to higher-resolution layers [7].

Given the effectiveness of such "skip connections" in recovering the fine-grained details of target objects and generating segmentation masks with fine details even with complex background imagery [31], it seems the original U-Net, with its restriction of only allowing the fusion of feature maps between layers at equivalent depths of the network may not prove optimal for pooling and recognizing features of various scales and levels of detail across different images. It was Zhou *et al*. that proposed the UNet++ framework to rectify this issue, which will be discussed below in the "Method" section [31]. The UNet++ framework is claimed to continuously outperform baseline U-Net model in a variety of imaging segmentation tasks [30].

Despite the current advancements, there is still one major flaw in existing implementations of U-Nets and their derivatives: almost all such implementations contain a hard-coded number of convolution layers and skip connections. However, it is also known that the optimal depth of an encoder-decoder network can vary from one application to another, depending on the task difficulty and the amount of labeled data available for training [30]. A simple approach would be to implement and train models of varying depths separately and then ensemble the resulting models during the inference time [25, 6, 3]. However, this is computationally expensive and inefficient in terms of the amount of source code. An encoder-decoder network that can dynamically adapt its structure to a specific depth value can be the first step to improving the efficiency of training such networks. That is what we shall implement.

3. Method

3.1. UNet++ Principles

The following Figure 1 illustrates a UNet++ architecture, with input size $n \times n \times 3$ and output size $n \times n \times \text{out}$. [1] The defining characteristic of such architectures is that UNet++ provides extra convolutional blocks between the encoder and decoder backbones and additional skip connections between each convolution block to facilitate the flow of image feature semantics.

3.1.1 Convolutional Blocks

The convolutional blocks, represented as large circles in Figure 1, are defined as a sequence of convolution, batch normalization, ReLU activation, convolution, batch normalization, and ReLU activation layers, in that order. The number of input and output channels for each convolution block is shown on the top and bottom of each circle in Figure 1, respectively.

3.1.2 Backbone Layers and Definition of "Depth"

In Figure 1, the "encoder backbone" and "decoder backbone" are defined as the leftmost and rightmost convolution blocks, respectively. Note that these backbone blocks would be the only ones present in a conventional U-Net model. In the meantime, the "depth" of a U-Net, as referred to in this report, is defined as the number of convolutional blocks in each of the encoder and decoder backbones. In the case of 1, that number would be five.

3.1.3 Additional Convolutional Blocks and Skip Pathways

We can now turn our attention to the intermediate convolutional blocks and the dense skip pathways in the UNet++ model. With dense connectivity, each block is presented with not only the final aggregated feature maps but also a concatenated version (represented by plus signs in Figure 1) of the intermediate aggregated feature maps and the original same-scale feature maps from the encoder. As such, the aggregation layer in the decoder node may learn to use only the same-scale encoder feature maps or use all collected feature maps available at the gate [31].

3.2. Baseline/Reference Model

The baseline implementation comes from a medium.com article by Jing [12] using the PyTorch library [20]. This baseline model is an exact copy of Figure 1, with a five-layer-deep encoder backbone and hard-coded intermediate convolution blocks and skip connections.

3.3. Variable-depth Implementaion

Our implementation functionally extends the baseline implementation, in the Jupyter notebook environment and also uses PyTorch [20, 21]. We discarded the hardcoded convolutional layers and skip the connection, in favor of code that automatically generates appropriate convolutional blocks and skip connections based on a user-provided number of desired network depth. We trained and tested our model in instances with depths of 2, 3, 4, 5, and 6 levels in the encoder backbone.

4. Experiments

4.1. Dataset and Dataloader

The dataset that we use for this project is named "Drishti-GS" [27]. This is a small dataset focusing on assessing the

[1] Please refer to all figures and graphs in the Appendix section.

damage of glaucoma by calculating the structural changes in the Optic Nerve Head (ONH) by comparing the relative sizes of an "optical disc (OD)" and an optical "cup." This dataset contains 101 retina images, with the annotated ground-truth masks of the optical disc and optical cup by a panel of human experts. Each pixel in each mask represents the *probability* that this pixel belongs to the corresponding segmentation mask, as a value ranging from 0 to 1.

The dataset is originally divided into 50 training and 51 testing images. For our project, we decide to reconfigure the dataset to contain 81 training samples and 20 testing samples since more training samples may reflect better on the performance of our model, given the limited total samples. The dataset is available on Dropbox at https://www.dropbox.com/s/d0hhrsraycouk9p/Drishti-GS1-EECS442-FA22.zip?dl=0.

To implement the data loader, we referred to the data loader that studies for Pset 6. The original size of the image in dataset 2047×1760 as claimed by Sivaswamy *et al*. [26]; however since the image might be cropped due to unknown reasons, the actual image size we use 2045 × 1752, which is different from the claimed size. We decide to resize the image to 224×224 from the aforementioned, higher resolution using PyTorch crop transformation functions.

We also applied the additional transformation of placing the center of the retina in the center of the transformed, square image, using the provided coordinates of the retinal center points of each image sample given by the dataset. Figure 2 contains a visualization of one data sample after the preprocessing by our data loader.

Finally, we arbitrated that the training and testing batch sizes that the data loader generates shall be 4 and 2, respectively, due to the number of training and test samples.

4.2. Model evaluation

Note that our model takes in a full-color, $224 \times 224 \times 3$ representation of a retina image, and outputs two black-and-white masks for both the optic disc (OD) and cup segmentation tasks, in the format of a $224 \times 224 \times 2$ tensor.

For the evaluation of the model, we decide to use the metric from the baseline implementation [12] and use a sum of the Binary Cross Entropy (BCE) with logits and Dice coefficient loss with logits over each output channel. Note that the "with logits" expression for both our loss function means that we first apply a sigmoid activation function to the inputs to the loss function to keep them within the range of [0,1], to prevent negative loss values. The reason that we combine the two models since the two methods allow for some diversity in the loss while keeping the stability by BCE.

4.2.1 Binary cross entropy loss

Here is the formula of the BCE :

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1-y_i\right) \cdot \log\left(1-p\left(y_i\right)\right)$$

From the formula, we can see that y is the label, and p(y) is the predicted probability of the point being correct for all samples.

4.2.2 Dice loss

The Dice coefficient, is a common metric for pixel segmentation that can also be modified to act as a loss function [18]:

$$DSC = \frac{2|X \cap Y| + s}{|X| + |Y| + s}$$

This metric measure of overlap area between the predicted and the ground truth.We take $2 \cdot$ the area of overlap that both the predicted and ground truth area have and divided by the total areas. This metric ranges between 0 and 1 where a 1 denotes a perfect overlap. Note that we set an additional smoothing parameter s to 1 to prevent zero division.

4.2.3 Training configuration

The hyperparameters are adopted from Jing's baseline implementation [12]. Each instance of our models is trained for 40 epochs. To optimize this model, we pick 40 epochs, with an Adam optimizer with a learning rate of 10^{-4}, and a StepLR optimizer that decays the learning rate by a factor of 0.1 for every 10 epochs. We also monitor the BCE, dice, and total loss values, as well as the training time, for each epoch. After that, we save the model state dictionaries for the epoch with the best validation loss, and we return histories of training and validation losses for visualization.

5. Results

5.1. Training Losses For Each Model

Graphs for the change in training losses for each model can be found in the Appendix section.

BCE loss, Dice loss, and total loss for all models share the same downward trend. 2-depth model has the greatest total loss among all models, which is approximately 1.25. Losses for 3-depth and 4-depth model are around 1.20. 5-depth model has the lowest total loss of 1.15, which beats the loss for 6-depth model by a little bit. The loss of our self-implemented 5-depth model, as we expected, achieves a loss that is almost the same as that of the baseline model.

5.2. Analysis of Validation Accuracies

Graphs for the change in validation losses for each model, as well as the prediction results for each trained model, can be found in the Appendix section. The following table presents the minimum validation loss for all instances of our trained models.

Models	Minimal Validation Loss
2-depth model	1.272
3-depth model	1.196
4-depth model	1.200
5-depth model	1.156
6-depth model	1.180
baseline model	1.196

Table 1: Minimal Validation Loss Table

By examining 1, we can see that the 2-layer-deep variable-depth model that we implemented clearly underperforms all other models in terms of validation accuracies. It might be a result of only having *three* convolution blocks in the entire network and subsequent failure to capture finer-grained details within the input image. However, it appears that the 3-deep, 4-deep, 5-deep, and 6-deep, along with the baseline 5-deep model, all achieve similar validation accuracies. Another thing that is worth mentioning is that our implementation for a model with a depth of 5 achieves almost the same result as the baseline model. It is a good indicator of the effectiveness and correctness of our implementation.

We can conclude that it is certainly an example of "the law of diminishing returns" as we introduce more complicated UNet++ structures for segmenting the relatively simple image in the Drishti-GS dataset.

If we only examine images for the the predicted segmentation masks for OD and cup (available in the appendix), as compared to their ground truth counterparts, we can witness a trend of the outline of the object we are examining getting more and more defined for all layers of models. However, the changes of distribution and evenness of the yellow patches in the predicted images failed to have a very clear trend.

5.3. Training Time

For models with a depth of 2, 3, 4, 5, and 6, the average training time per epoch is approximately 15.2, 15.8, 18.25, 25.5, and 32.2 seconds, respectively. For the baseline model, the average training time per epoch is around 25.4 seconds. Here, we provide the following table to directly compare and contrast the differences in overall model training time. Notice that each model was trained for 40 epochs.

In general, average model training time increases as we deepen the models. We can also approximate that the model training time increases at a log-normal rate relative to the depth of the UNet++ model.

Models	Average Model Training Time (s)
2-depth model	608
3-depth model	632
4-depth model	730
5-depth model	1020
6-depth model	1288
baseline model	1016

Table 2: Training Time Table

5.4. Space Complexity

To demonstrate how space complexity changes as we change depth of our model, a table filled in with values of number of parameters and size of parameter storage (in megabytes) for models with different depths is provided down below.

Models	Total Params	Params Size (MB)
2-depth model	408,898	1.56
3-depth model	2,070,466	7.90
4-depth model	8,932,930	34.08
5-depth model	36,629,698	139.73
6-depth model	147,688,258	563.39
baseline model	36,629,698	139.73

Table 3: Space Complexity Table

We can note from Table 3 that the space complexity of the model grows exponentially as the number of depth levels of the UNet++ model increases.

5.5. Discussions

We can conclude that at a certain point, the variable-depth UNet++ model would become too spatially complex to justify the marginal increases in performances in practical applications. As a matter of fact, it has already become impossible to store the entirety of a 7-layer-deep UNet++ model on a single, commercial-grade discrete GPU (Nvidia GTX 1070 in our case). Even a 6-layer-deep model, with more than 500 MB required to store its parameters alone, might prove to be too large for some low-end GPUs which also have to handle other graphical calculations while running the UNet++ model to segment images, and its 20% increase in training time might not justify the performance *decrease* compared to our 5-layer-deep model. In conclusion, we can find that a 4-5-layer deep model might be the most suited for segmenting the images in the Drishti-GS dataset, providing a good balance between validation performance and training efficiency.

6. Conclusions

We successfully implemented a varied depth UNet++ model in less than 1000 lines of code, which turned out to be the most simple one among all such models we can find on the Internet. We verified the effectiveness and correctness of our model by comparing its training loss, training time, and space complexity with the baseline model. After several times of experiments, we are confident to conclude that as we increment the depth of our model, training time grows linearly, accuracy plateaus after a 4-5-deep model, and space complexity increases exponentially. Another important takeaway message that we want to share is that one should beware of the trade-offs between efficiency and accuracy when using this model. As we showed early in the paper, the 4-5 deep model ends up being the most optimal model in a way that the marginal cost of gaining accuracy by adding another layer weighs way over its marginal benefit when the model is 4 to 5 layers deep.

References

[1] Georges B. Aboutanos and Benoit M. Dawant. Automatic brain segmentation and validation: image-based versus atlas-based deformable models. 3034:299–310, Apr. 1997. Conference Name: Medical Imaging 1997: Image Processing ADS Bibcode: 1997SPIE.3034..299A.

[2] J. C. Bezdek, L. O. Hall, and L. P. Clarke. Review of MR image segmentation techniques using pattern recognition. *Medical Physics*, 20(4):1033–1048, 1993.

[3] Francesco Ciompi, Bartjan de Hoop, Sarah J. van Riel, Kaman Chung, Ernst Th Scholten, Matthijs Oudkerk, Pim A. de Jong, Mathias Prokop, and Bram van Ginneken. Automatic classification of pulmonary peri-fissural nodules in computed tomography using an ensemble of 2D views and a convolutional neural network out-of-the-box. *Medical Image Analysis*, 26(1):195–202, Dec. 2015.

[4] G.B. Coleman and H.C. Andrews. Image segmentation by clustering. *Proceedings of the IEEE*, 67(5):773–785, May 1979. Conference Name: Proceedings of the IEEE.

[5] D. Louis Collins, C. J. Holmes, T. M. Peters, and A. C. Evans. Automatic 3-D model-based neuroanatomical segmentation. *Human Brain Mapping*, 3(3):190–208, 1995. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbm.460030304.

[6] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, Lecture Notes in Computer Science, pages 1–15, Berlin, Heidelberg, 2000. Springer.

[7] Getao Du, Xu Cao, Jimin Liang, Xueli Chen, and Yonghua Zhan. Medical Image Segmentation based on U-Net: A Review. *Journal of Imaging Science and Technology*, 64, Mar. 2020.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, Oct. 2014. arXiv:1311.2524 [cs].

[9] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Pearson Education Limited, Harlow, United Kingdom, fourth edition, global edition edition, 2018. OCLC: 1065078188.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, Dec. 2015. arXiv:1512.03385 [cs].

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[12] Hong Jing. Biomedical Image Segmentation: UNet++, Nov. 2020.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[14] T. Lei and W. Sewchand. Statistical approach to X-ray CT imaging and its applications in image analysis. II. A new stochastic model-based image segmentation technique for X-ray CT image. *IEEE Transactions on Medical Imaging*, 11(1):62–69, Mar. 1992.

[15] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer Science & Business Media, Apr. 2009. Google-Books-ID: rDsObhDkCIAC.

[16] Z. Liang, J. R. Macfall, and D. P. Harrington. Parameter estimation and tissue segmentation from multispectral MR images. *IEEE transactions on medical imaging*, 13(3):441–449, 1994.

[17] Tim McInerney and Demetri Terzopoulos. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, 1(2):91–108, June 1996.

[18] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, Oct. 2016.

[19] Dong Nie, Li Wang, Yaozong Gao, and Dinggang Sken. Fully convolutional networks for multi-modality isointense infant brain image segmentation. volume 2016, pages 1342–1345, Apr. 2016.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, Dec. 2019. arXiv:1912.01703 [cs, stat].

[21] Fernando Perez and Brian E. Granger. IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[22] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. Current Methods in Medical Image Segmentation. *Annual Review of Biomedical Engineering*, 2(1):315–337, 2000. _eprint: https://doi.org/10.1146/annurev.bioeng.2.1.315.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*,

Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing.

[24] P. K Sahoo, S Soltani, and A. K. C Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41(2):233–260, Feb. 1988.

[25] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, May 2016.

[26] Jayanthi Sivaswamy, Subbaiah Krishnadas, Arunava Chakravarty, and Gopal Joshi. A comprehensive retinal image dataset for the assessment of glaucoma from the optic nerve head analysis. *JSM Biomed Imaging Data Pap*, 2, Jan. 2015.

[27] Jayanthi Sivaswamy, S. R. Krishnadas, Gopal Datt Joshi, Madhulika Jain, and A. Ujjwaft Syed Tabish. Drishti-GS: Retinal image dataset for optic nerve head(ONH) segmentation. In *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pages 53–56, Apr. 2014. ISSN: 1945-8452.

[28] Wenlu Zhang, Rongjian Li, Houtao Deng, Li Wang, Weili Lin, Shuiwang Ji, and Dinggang Shen. Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, 108:214–224, Mar. 2015.

[29] Tongxue Zhou, Su Ruan, and Stéphane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3-4:100004, Sept. 2019.

[30] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, June 2020.

[31] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support : 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, held in conjunction with MICCAI 2018, Granada, Spain, S*, 11045:3–11, Sept. 2018.

A. Figures and Illustration

The following contains the graphs for the UNet++ architecture, as well as the various losses and illustrations of validation results of the baseline model and our variable-depth UNet++ model of different depths.

The training and validation losses for the dice loss, binary-cross-entropy loss, and the total loss at each epoch are available. The input image, ground truths for the optic disc (OD) and cup masks, as well as predictions for the two masks are also available below.

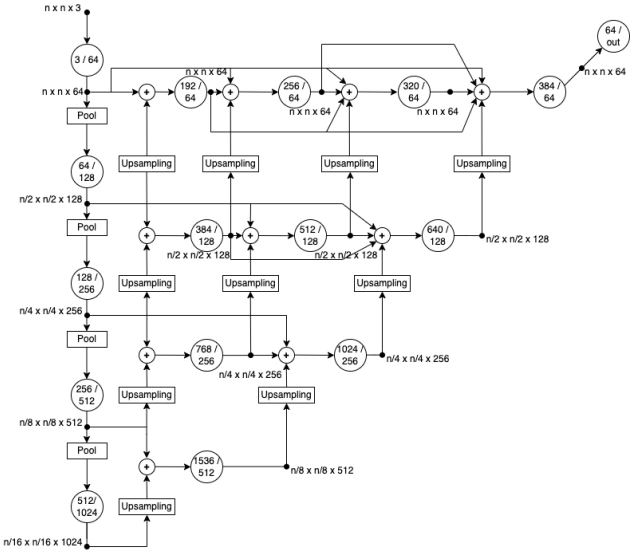

Figure 1: Graph of the UNet++ network with a layer depth of five

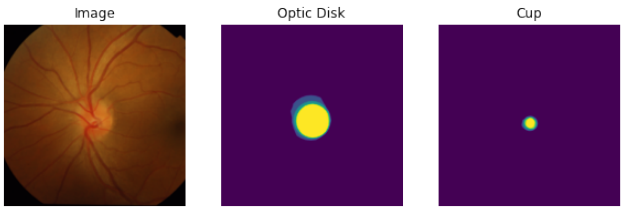

Figure 2: Sample input image and ground truth OD and cup masks from the Drishti-GS dataset

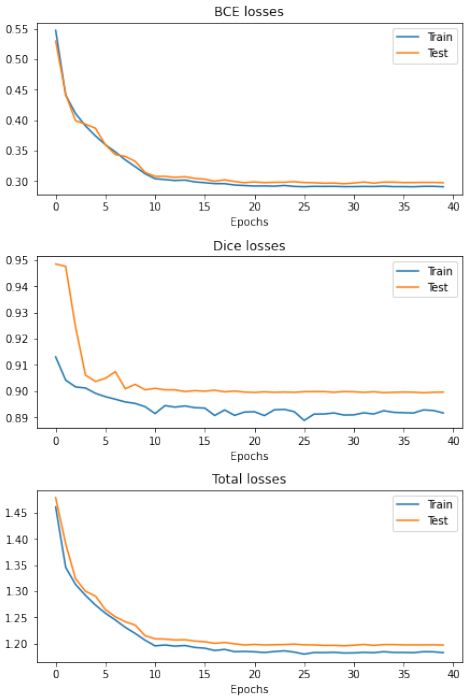

Figure 3: Loss functions for baseline implementation

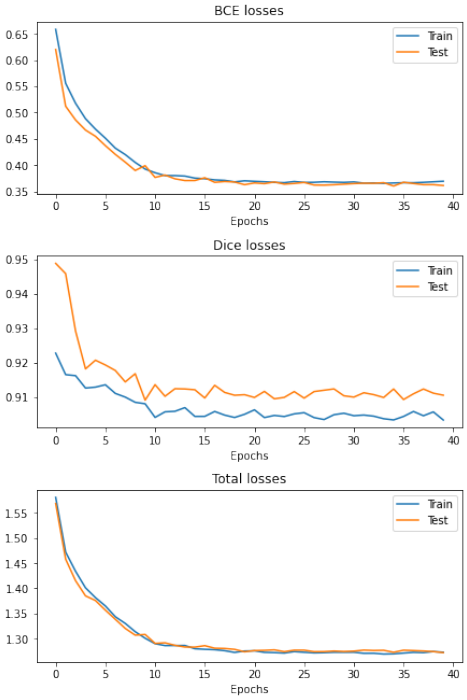

Figure 4: Loss functions for 2-layer-deep variable-depth model

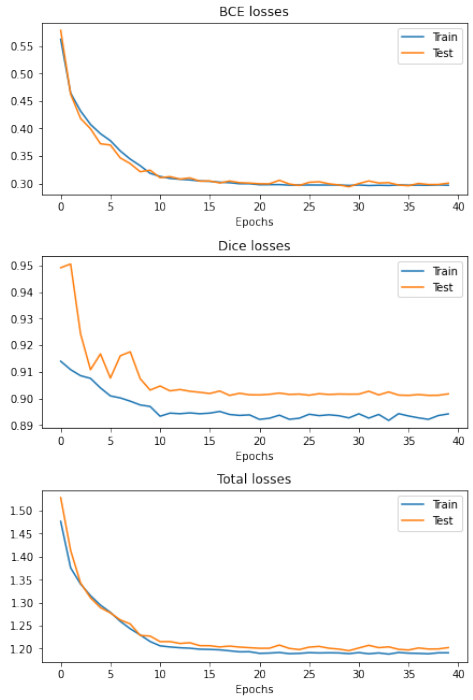

Figure 5: Loss functions for 3-layer-deep variable-depth model

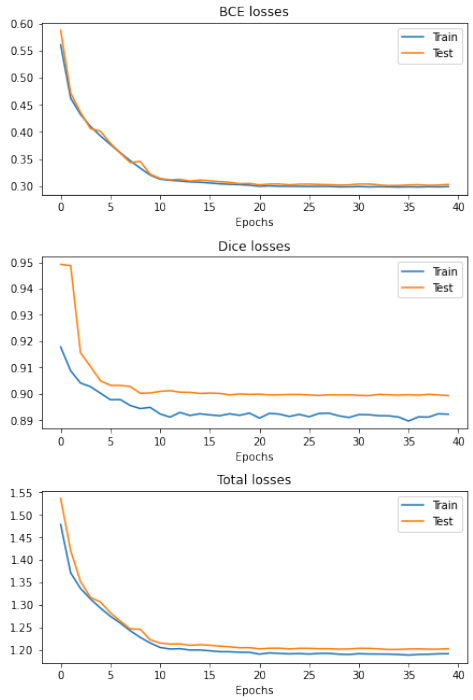

Figure 6: Loss functions for 4-layer-deep variable-depth model

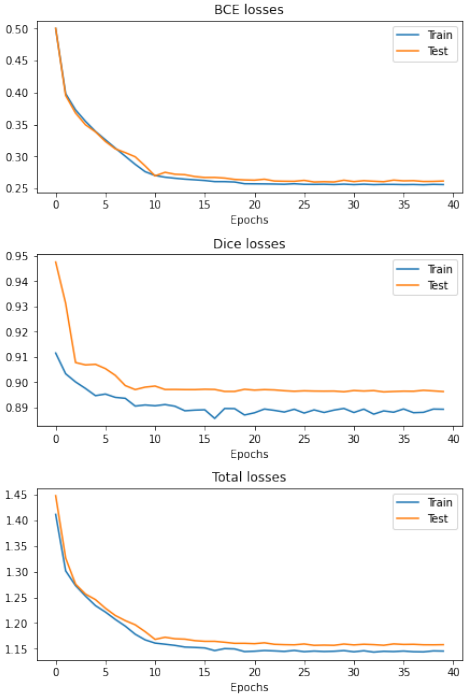

Figure 7: Loss functions for 5-layer-deep variable-depth model

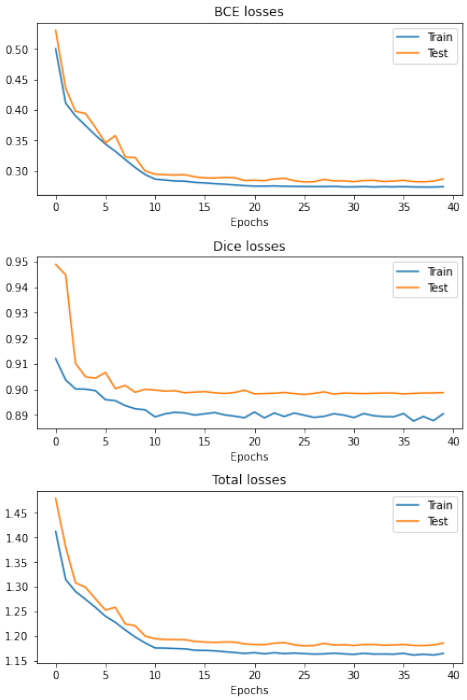

Figure 8: Loss functions for 6-layer-deep variable-depth model

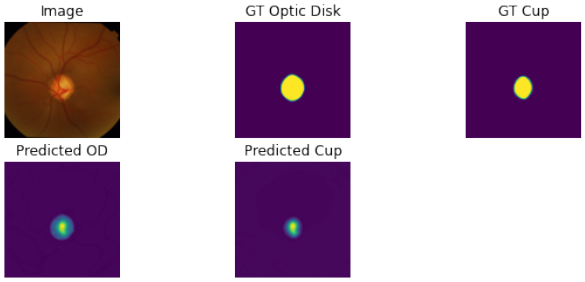

Figure 9: Evaluation results for baseline implementation

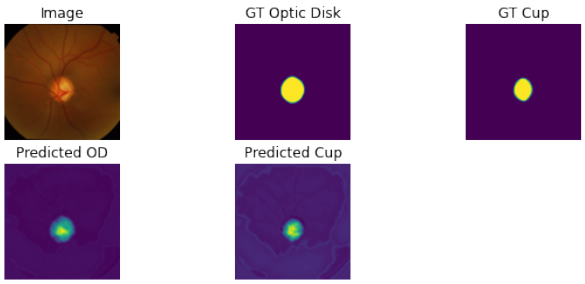

Figure 10: Evaluation results for 2-layer-deep variable-depth model

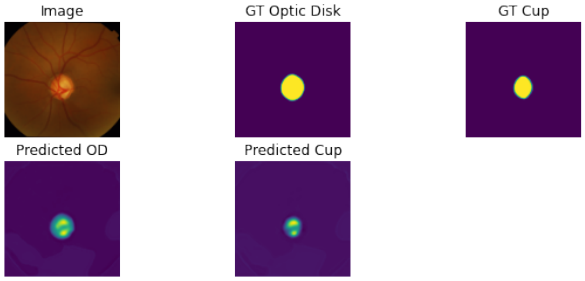

Figure 11: Evaluation results for 3-layer-deep variable-depth model

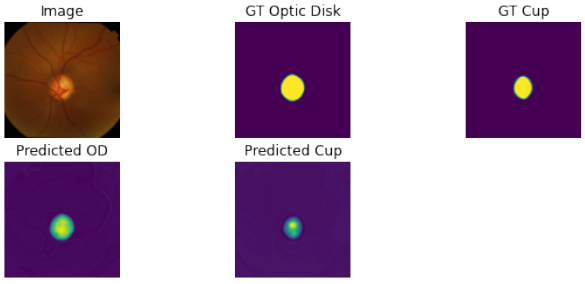

Figure 12: Evaluation results for 4-layer-deep variable-depth model

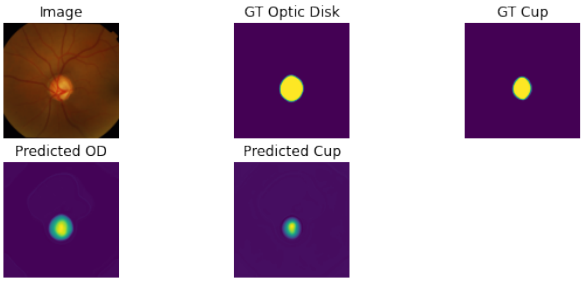

Figure 13: Evaluation results for 5-layer-deep variable-depth model

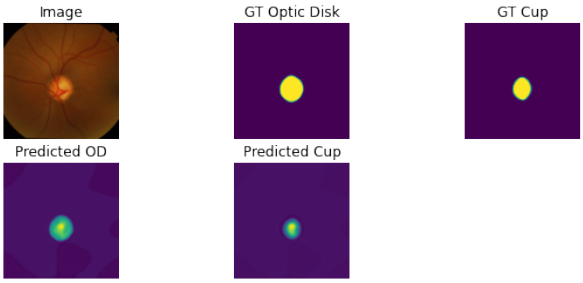

Figure 14: Evaluation results for 6-layer-deep variable-depth model