Dental X-ray Image Segmenation using a U-shaped Deep Convolutional Network

Olaf Ronneberger[1,2], Philipp Fischer[1], and Thomas Brox[1,2]

[1] Computer Science Department, University of Freiburg, Germany
[2] BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
http://lmb.informatik.uni-freiburg.de/people/ronneber/

Abstract. Segmentation of dental X-ray images is a very challenging task. To investigate possible automated methods a challenge for dental X-ray image analysis was organized in conjunction with the ISBI 2015. We present a pure machine learning approach using a u-shaped deep convolutional neural network ("u-net") for the fully automated segmentation of dental x-ray images into the classes *caries*, *enamel*, *dentin*, *pulp*, *crown*, *restoration* and *root canal treatment*. The architecture of the u-net consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images. Moreover, the network is fast. Segmentation of one x-ray image (scaled to 670x400 pixels) takes only approx. 1.5 seconds. The network reaches a dice similarity of over 70% for the frequent classes *enamel*, *dentin* and *pulp*. The average dice similarity for all classes is 56,4%.

1 Introduction

In the last two years, deep convolutional networks have outperformed the state-of-the-art in many visual recognition tasks, e.g. [8, 4, 13]. While convolutional networks have already existed for a long time [3, 9, 10], their success was limited due to the size of the available training sets and the size of the considered networks. The breakthrough by Krizhevsky et al. [8] was due to supervised training of a large network with 8 layers and millions of parameters on the ImageNet dataset with 1 million training images. Since then, even larger and deeper networks have been trained [12].

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks. Hence, Ciresan et al. [1] trained a network to predict the class label of each pixel by providing a local region (patch) around that pixel as input. First, this network can localize. Secondly, the training data in terms of patches is much

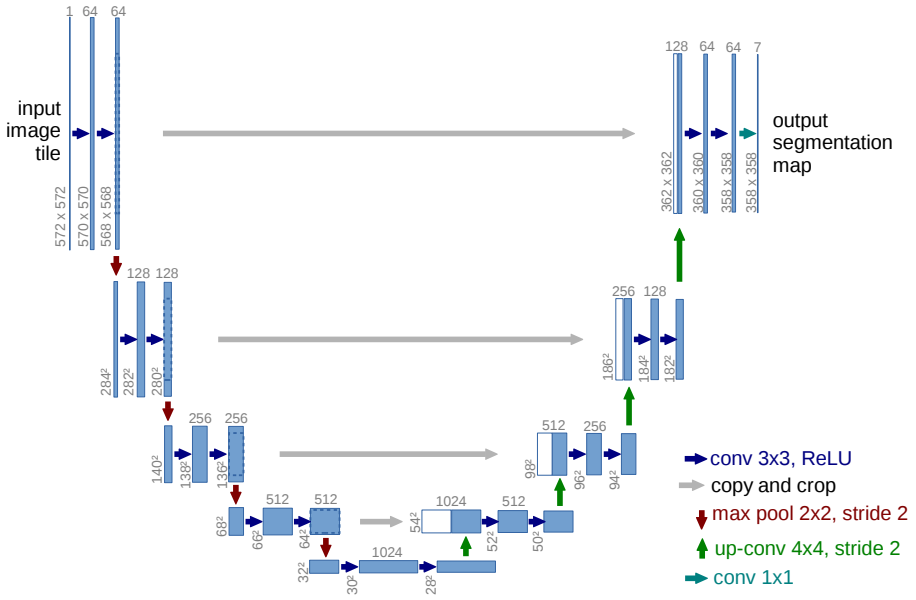


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a stack of feature maps. The number of features is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

larger than the number of training images. The resulting network won the EM segmentation challenge at ISBI 2012 by a large margin.

Obviously, the strategy in Ciresan et al. [1] has two drawbacks. First, it is quite slow because the network must be run separately for each patch, and there is a lot of redundancy due to overlapping patches. Secondly, there is a trade-off between localization accuracy and the use of context. Larger patches reduce localization accuracy, while small patches allow the network to see only little context. Hariharan et al. [5] have proposed a classifier output that takes into account the features from multiple layers. Good localization and the use of context are possible at the same time.

In this paper, we build upon a more elegant architecture, the so-called "fully convolutional network" [11]. We modify and extend this architecture such that it works with very few training images and yields more precise segmentations; see Fig. 1. The main idea in [11] is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled output. A successive convolution layer can then learn to assemble a more precise output based on this information.

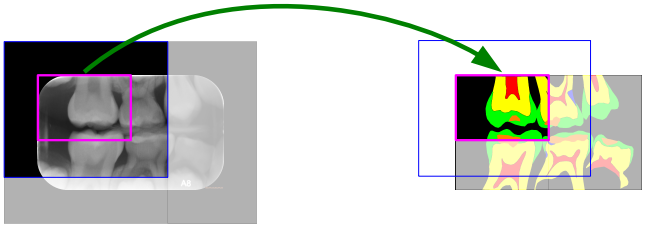


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrarily large images. Prediction of the segmentation in the magenta area, requires image data within the blue area as input. Missing input data is extrapolated by zero padding

One important modification in our architecture is that in the upsampling part we have also a large number of feature maps, which allows the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image. This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy (see Fig. 2). To predict the pixels in the border region of the image, the missing context is extrapolated by zero padding. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

As there is very little training data available, we use excessive data augmentation by applying elastic deformations to the available training images. This allows the network to learn invariance to such deformations, without the need to see these transformations in the annotated image corpus. This is particularly important in biomedical segmentation, since deformation used to be the most common variation in tissue and realistic deformations can be simulated efficiently. The value of data augmentation for learning invariance has been shown in Dosovitskiy et al. [2] in the scope of unsupervised feature learning.

The resulting network is applicable to various biomedical segmentation problems. In the segmentation of neuronal structures in EM stacks (an ongoing competition started at ISBI 2012) we outperformed the network of Ciresan et al. [1] (publication in review). Here we show its application to the segmentation of dental X-ray images.

2 Network Architecture

The network architecture is illustrated in Fig. 1. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (only using the valid part), each followed

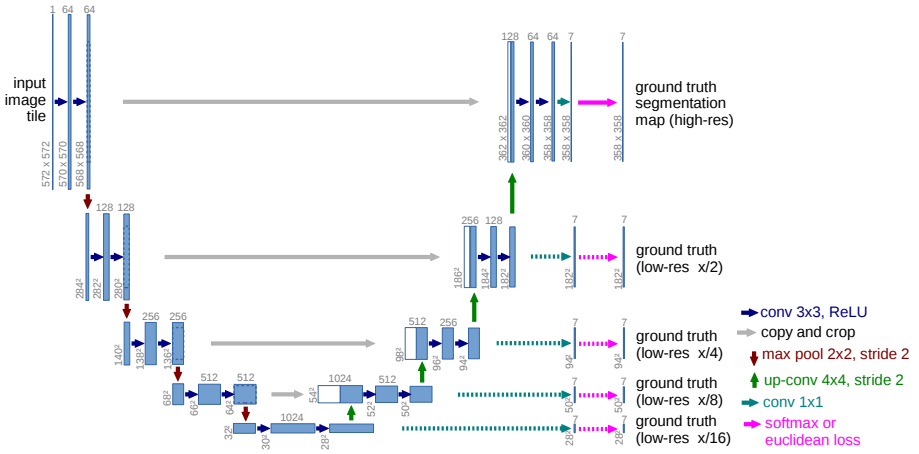


Fig. 3. Training of the u-net. For training of the net u1 only the high-resolution ground truth map was used. Network u2 and u3 were trained with additional loss layers at lower resolution (dashed arrows). u2 uses softmax loss and u3 uses euclidean loss in the low-resolution loss layers.

by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature maps. Every step in the expansive path consists of a spatial upsampling of the feature maps with a factor of 2 followed by a 4x4 convolution that halves the number of feature maps, a concatenation with the correspondingly cropped feature maps from the contracting path, and one or two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-dim feature vector to the desired number of classes (here 7). In total the network has 23 convolutional layers.

To allow a seamless tiling of the output segmentation map (see Fig. 2), it is important to select the input tile size such that all 2x2 max-polling operations are applied to a layer with an even x- and y-size.

3 Training

The input images and their corresponding segmentation maps are used to train the network (see Fig. 3). The optimization is done with the stochastic gradient descent implementation of Caffe [7]. Due to the loss of border pixels in every convolution, the input image is larger than the output by a constant border width. To minimize the overhead and make maximum use of the GPU memory, we favor large input tiles over a large batch size and hence reduce the batch to a single image. To compensate for instable gradients we accordingly set a high momentum (0.99) such that a large number of the previously seen training samples determine the current optimization step.

We have tested three different training schemes. The first scheme (denoted as "u1") performs a straight forward end-to-end training, where only one high resolution loss layer at the end of the network is applied. In the second scheme (denoted as "u2") we applied additional loss layers to the low-resolution feature maps, in order to guide the deep layers to directly learn the segmentation classes. In a third scheme ("u3") we replaced the softmax loss in the low-resolution layers with a Euclidean loss, to allow multiple classes per low-resolution pixel.

In deep networks with many convolutional layers and different paths through the network, a good initialization of the weights is extremely important. Otherwise, parts of the network might give excessive activations, while other parts never contribute. Ideally the initial weights should be adapted such that each feature map in the network has approximately unit variance. For a network with our architecture (alternating convolution and ReLU layers) this can be achieved by drawing the initial weights from a Gaussian distribution with a standard deviation of $\sqrt{2/N}$, where N denotes the number of incoming nodes of one neuron [6]. E.g. for a 3x3 convolution and 64 feature maps in the previous layer $N = 9 \cdot 64 = 576$.

3.1 Data Augmentation

Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available. In case of x-ray images we primarily need invariance to translations and robustness to rotations, deformations and gray value variations. Especially random elastic deformations of the training samples seem to be the key concept to train a segmentation network with a very low number of annotated images. We generate smooth deformations using random displacement vectors on a coarse 3 by 3 grid. The displacements are sampled from a Gaussian distribution with 10 pixels standard deviation. Per-pixel displacements are then computed using bicubic interpolation. Drop-out layers at the end of the contracting path perform further implicit data augmentation.

4 Experiments

Table 1. Network Architectures

Name	Expansion Path	Initialization	Additional Loss Layers
u1	4x4 upconv + one 3x3 conv	manual	only high-res segmentation map
u2	4x4 upconv + two 3x3 conv	$\sqrt{2/N}$	+ low-res, softmax loss
u3	4x4 upconv + two 3x3 conv	$\sqrt{2/N}$	+ low-res, Euclidean loss

We trained three slightly different networks (see Table. 1) with the data provided by the challenge organizers (39 annotated bitewing images from 39

patients – See Appendix 1). The images were scanned in different resolutions, so we rescaled all of them to a standard size of 670x400 pixels and normalized to a gray value range of $[0, 1]$. Using the upper listed augmentations we created 20,000 training image tiles with 524x524 pixels (largest possible size for the GPU memory of 6GB) with the correspondingly transformed segmentation maps. All networks were trained by stochastic gradient descent, starting with a learning rate of 0.001, which was decreased 2 times by a factor of 10 after 20,000 iterations. The whole training process of one network took about 10 hours on a NVidia Titan GPU. For classification we averaged the result of the network for the original image and its 3 mirrored versions. The segmentation of a new image (using multiple tiles, and including the 4 mirrored versions) takes approx. 1.5 seconds. The outcome of our u-net on the test image 46 (that is also displayed on the challenge web page) is shown in Fig. 4. Complete results of network "u2" are shown in Appendix 2. The quantitative scores provided by the organizers on the test set are shown in Fig. 5 and Table 2.

Table 2. Average results on the test data set.

Name	Precision	True Positives	True Negatives	Dice Similarity
u1	0.450	**0.613**	0.980	0.558
u2	**0.453**	0.576	**0.983**	**0.564**
u3	0.426	0.583	**0.983**	0.529

5 Conclusion

The u-net architecture achieves a high performance on the dental segmentation task. The most common classes (*enamel*, *dentin* and *pulp*) are segmented with a dice similarity of over 70%. Thanks to data augmentation with elastic deformations it can be trained with a number of training images (only 39) that is orders of magnitudes lower than in other successful deep convolutional network applications. The additional loss layers in architecture "u2" lead to a visible improvement of the results. The guiding with multiple labels in the low-resolution layers (architecture "u3") did not further improve the performance. Most probably the use of an Euclidean loss for such type of multi-label training data is suboptimal, as it also violates correct activations, that are greater than 1. We will develop more sophisticated loss layers in the future.

We assume that the overall performance in this application can be increased by providing more rigorously annotated training data. In the current data sets (see Appendix 1) teeth are sometimes labelled as background (e.g. in image 6), which makes the learning task difficult. Furthermore the data augmentation by deformation seems to create reasonable additional training instances for *enamel*, *dentin* and *pulp* but the other classes *caries*, *crown*, *restoration* and *root canal treatment*, appear quite different according to their relative location, so the augmentation is less successful here. Presumably a higher number of training images

a

No.	Important Properties Parts
1	caries (blue color)
2	enamel (green color)
3	dentin (yellow color)
4	pulp (red color)
5	crown (skin color)
6	restoration (orange color)
7	root canal treatment (cyan color)

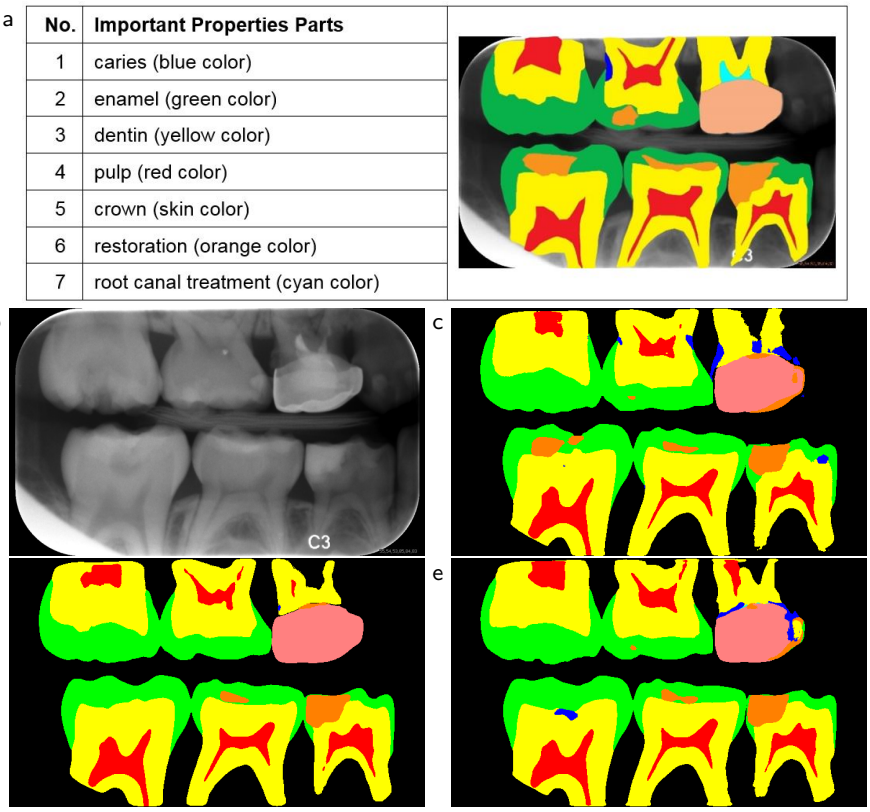


Fig. 4. Example results for sample 46. (**a**) legend and manual ground truth from the challenge web page. (**b**) input x-ray image. (**c**) result of u-net architecture "u1" (**d**) architecture "u2" (**e**) architecture "u3"

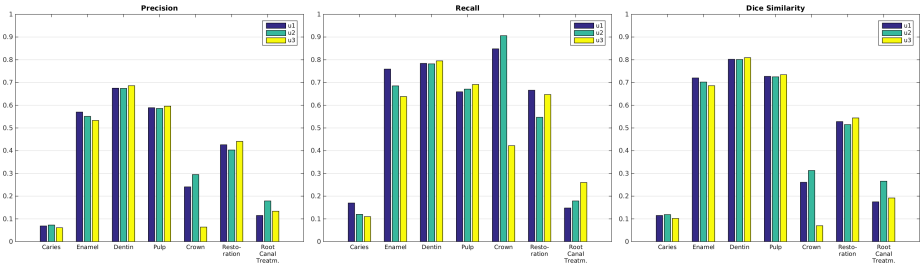


Fig. 5. Results on the test data set

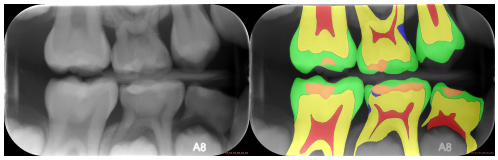
is needed to learn these classes. We will try to improve the performance by using even deeper networks (such that each output pixel "sees" a bigger context).

References

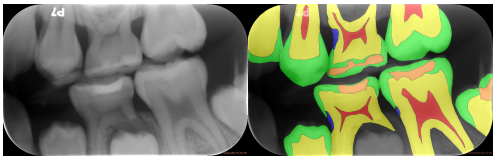
1. Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: In NIPS. pp. 2852–2860 (2012)
2. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS (2014)
3. Fukushima, K.: Neocognitron: A Self-Organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193–202 (1980)
4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
5. Hariharan, B., Arbelez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. arXiv [cs.CV] p. arXiv:1411.5752 (2014)
6. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv [cs.CV] p. arXiv:1502.01852 (2015)
7. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1106–1114 (2012)
9. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541–551 (1989)
10. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
11. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. arXiv [cs.CV] p. arXiv:1411.4038 (2014)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: arxiv:cs/arXiv:1409.1556 (2014)
13. Zhang, N., Donahue, J., Girshick, R., Darrell, T.: Part-based R-CNNs for fine-grained category detection. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014)

6 Appendix 1: Training data

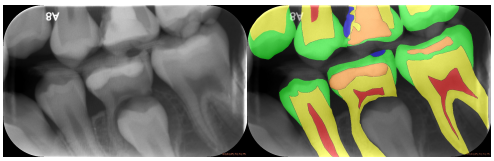
1



2

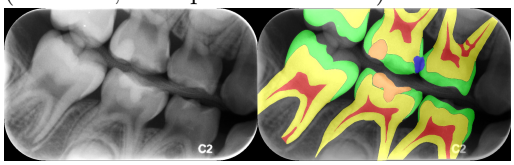


4

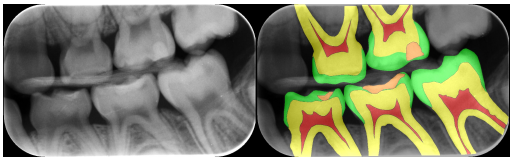


3 (not used, corrupted annotation)

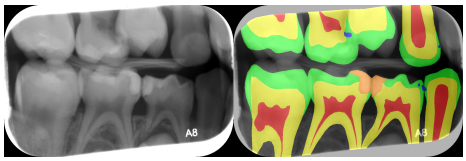
5



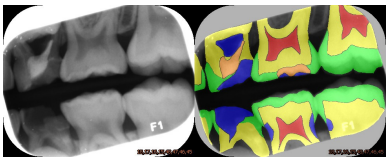
6



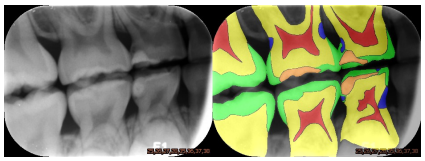
7



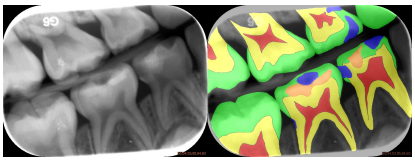
8



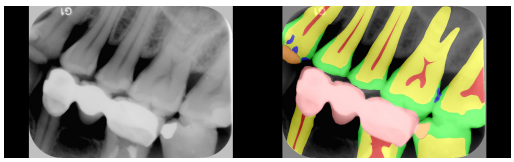
9



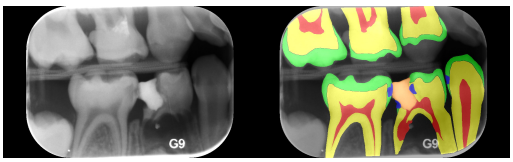
10



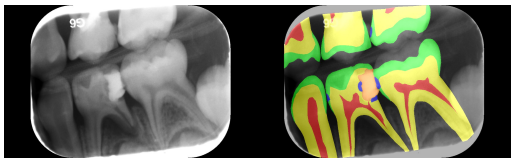
11



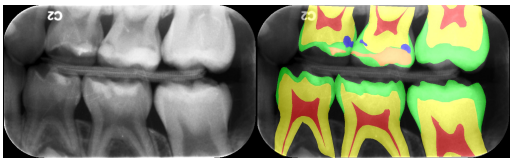
12



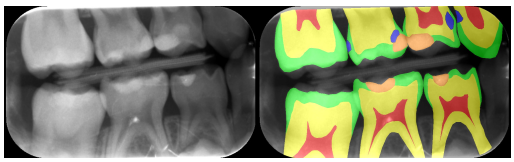
13



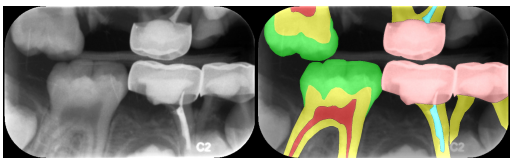
14

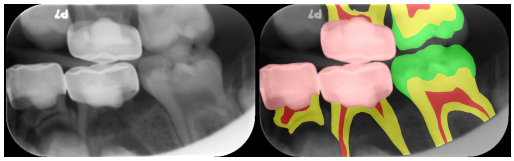


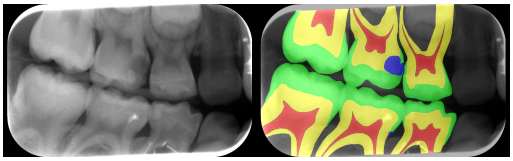
15

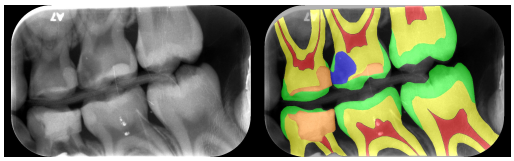


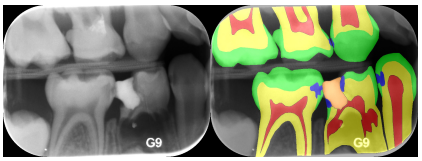
16

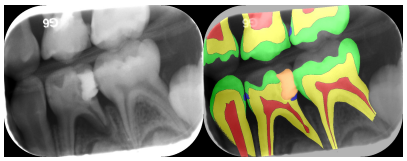


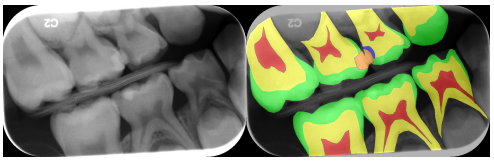
17 

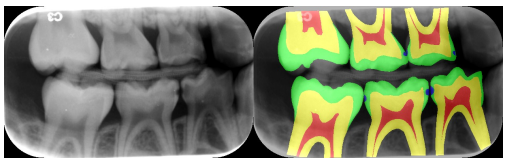
18 

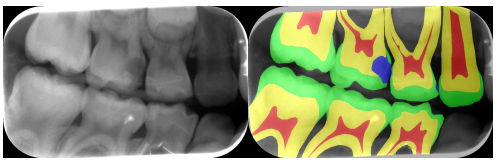
19 

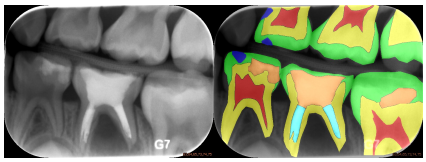
20 

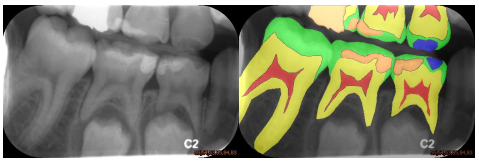
21 

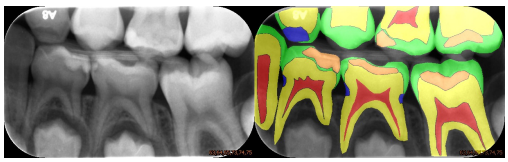
22 

23 

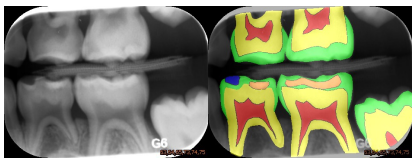
24 

25 

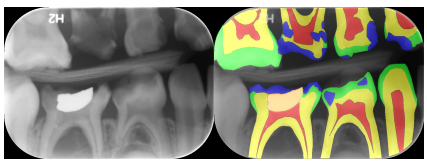
26 

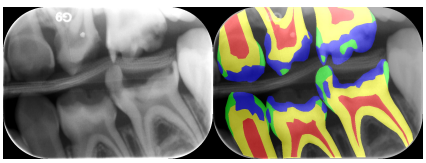
27 

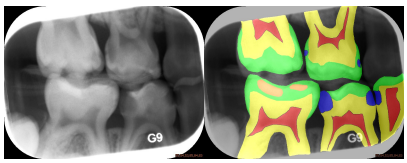
28 

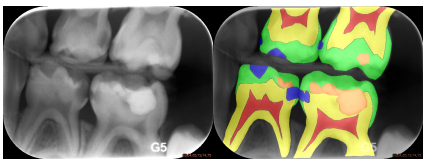
29 

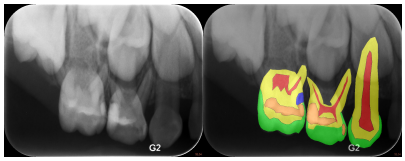
30 

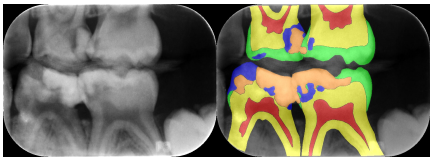
31 

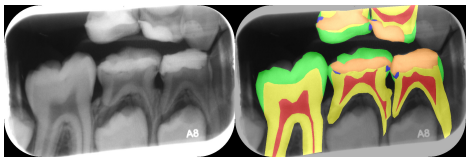
32 

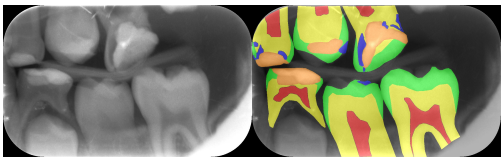
33 

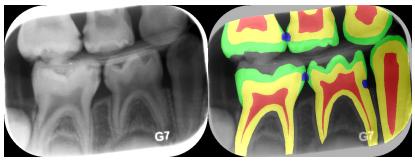
34 

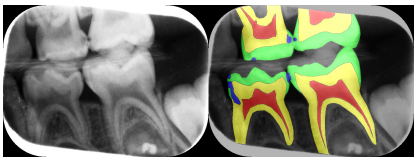
35 

36 

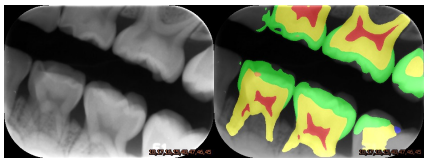
37 

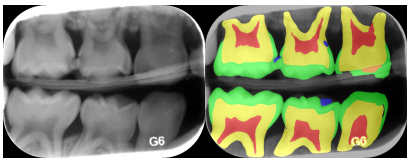
38 

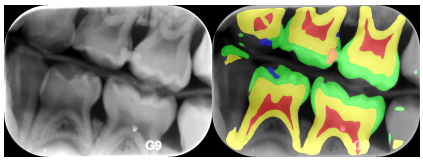
39 

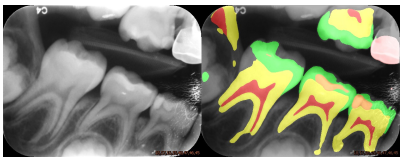
40 

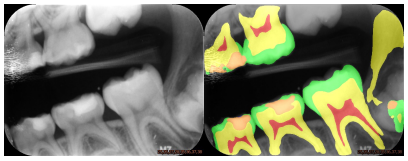
7 Appendix 1: Results of U-Net "u2" on Test Data

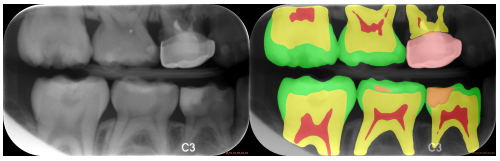
41 

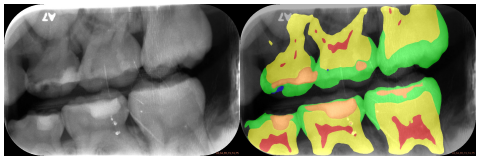
42 

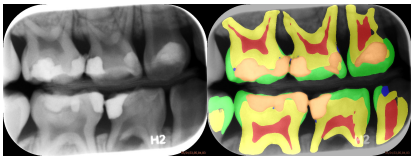
43 

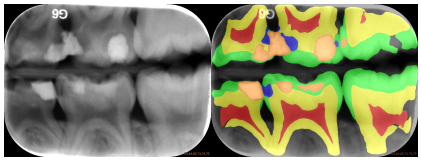
44 

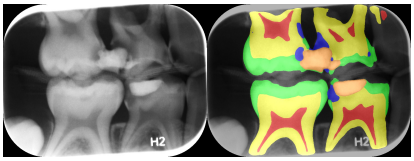
45 

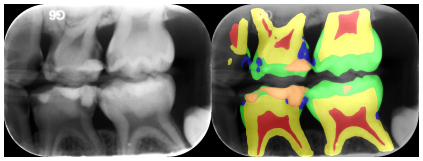
46 

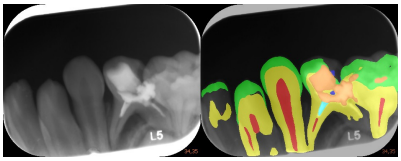
47 

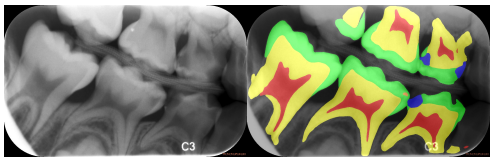
48 

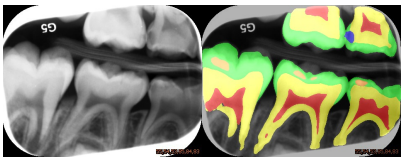
49 

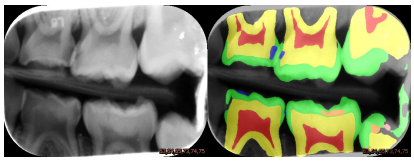
50 

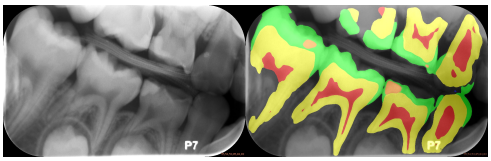
51 

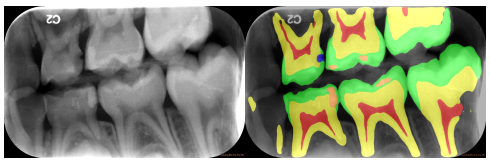
52 

53 

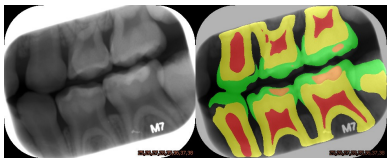
54 

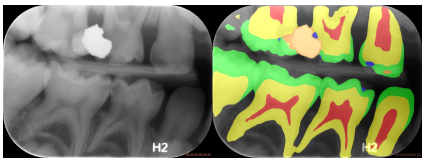
55 

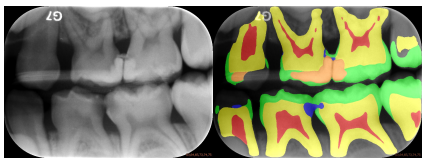
56 

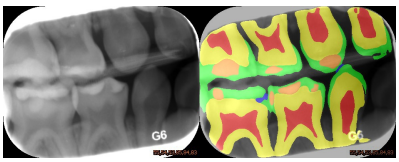
57 

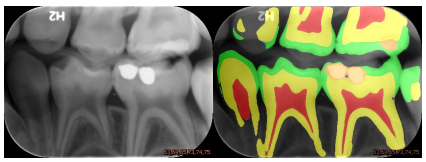
58 

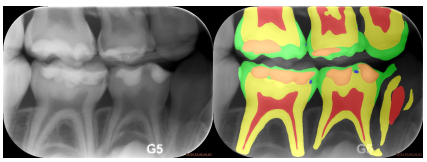
59 

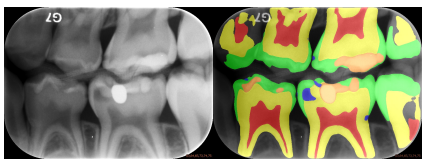
60 

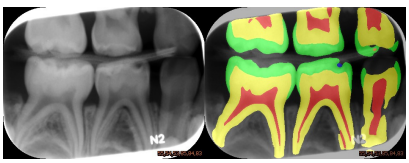
61 

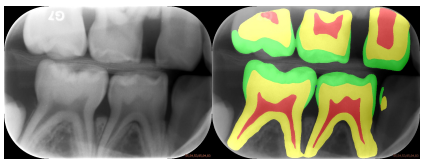
62 

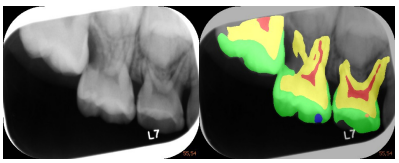
63 

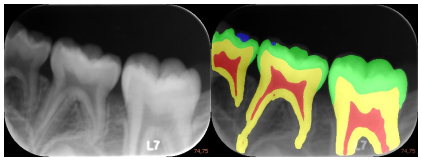
64 

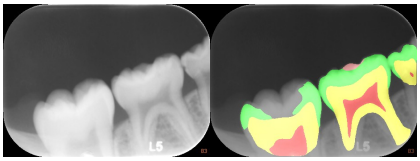
65 

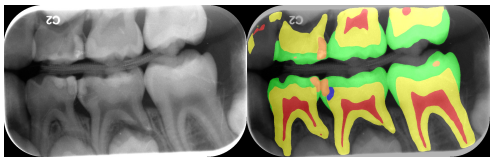
66 

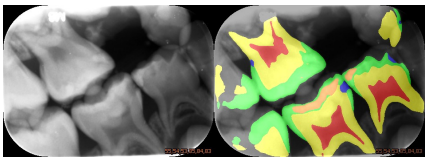
67 

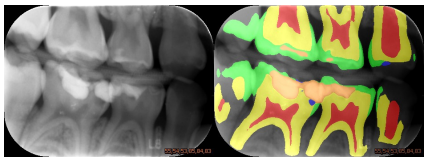
68 

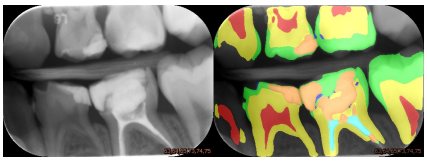
69 

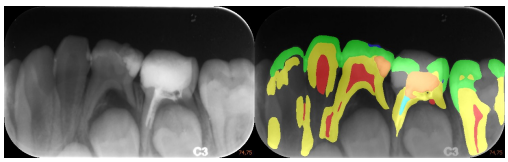
70 

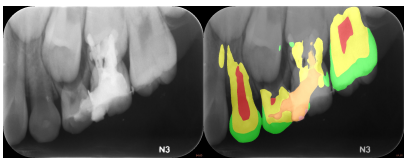
71 

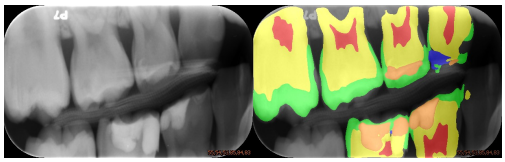
72 

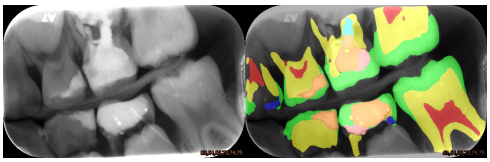
73 

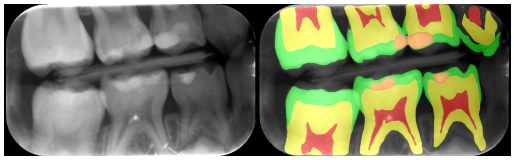
74 

75 

76 

77 

78 

79 

80