U-Net+: Pushing the plain U-Net to its limits for saliency detection problems

Abstract

Recently developed algorithms for saliency detection are mostly deep learning based methods. Many of the state-of-the art methods utilize pretrained deep networks that were originally trained for classification tasks. In addition, methods exploit both global and local features of the images by training separate networks. In this article we challenge those highly complex methods by proposing a simple model called U-Net+ that is trained intuitively end-to-end from scratch. It is based on a plain U-Net architecture that was originally used for image segmentation. We apply some of the widely used deep learning techniques targeting regularization and stable training. U-Net+ was able to beat the conventional methods by a wide margin and performed comparably to the state-of-the-art deep learning based methods. We also analyze the limitations of our model and propose improvements for future work.

1 Introduction

Deep learning has been successful in various high-level computer vision tasks [1], such as image classification [2, 3], object detection [4, 5], and semantic segmentation [6, 7]. Many successful algorithms utilize convolutional neural networks (CNN) as they are naturally suitable for digital images.

Recently, many methods started to get rid of fully-connected layers and train fully-convolutional networks [7]. Networks containing fully-connected layers tend to have large number of parameters which leads to slow training and overfitting. Moreover, an all-convolutional-network is perfect when we want to output images instead of flat vectors.

One of the successful applications of full-convolutional networks is U-Net [8], that was used for image segmentation tasks. It consists of convolutional, downsampling and umpsampling layers to exploit wide range of information from an image. The unique features of U-Net were the skip connections between encoder and the decoder layers. They provided information that was lost after downsampling and greatly helped during training. Many other methods from various fields adopted U-Net style architecture to solve computer vision problems [9].

For saliency detection, many of the recent state-of-the-art methods use neural networks [10, 11]. Some of them use deep pretrained networks such as VGG-16 [12] as a part of their algorithm. This is because saliency detection is regarded as a high-level computer vision task and to get high performance the algorithms need to be able to extract more abstract information from the images. Other deep learning based methods exploit local and global features of the image by training two separate networks.

In this work we try to find out how those recent state-of-the-art methods perform compared to the plain U-Net model. We apply some of the widely used deep learning techniques to U-Net targeting regularization and stable training. We analyze how those techniques improve or hinder the performance of our model and propose a U-Net+ method for saliency detection that is trained end-to-end from scratch.

Preprint. Work in progress.

2 Method

In this section, we explain how the U-Net was constructed for the saliency detection problem. Each section describes the thought behind the application of particular deep learning methods.

2.1 U-Net modifications

Several changes were introduced to the vanilla U-Net architecture that was first proposed in [8]. The original U-Net was used for image segmentation, and thus had two channels in the output image for mask and border. Since we are dealing with the saliency detection, we only need a single channel output that has the mask of salient objects. We also added padding for the convolutional layers to avoid changing size of the image. Sigmoid activation function is applied at the last layer to generate saliency map.

Next we feed 256×256 input images to our network unlike 512×512 in the original U-Net. The depth (number of downsamplings) was not changed in order to maintain big receptive field. At this point our network had the receptive field of 230, so we added one more convolutional layer at the bottleneck to arrive at the receptive field of 262 which covers the whole input image.

2.2 Batch normalization

Introduced in [13], batch normalization (batchnorm) has become one of the most vital parts of stable training of deep networks. This is achieved by controlling the the first two moments (mean and variance) of the layer inputs. Applying batchnorm layers in the network eliminates the need for normalization of the input data and careful initialization of the weights. Moreover, it also adds some regularization to the network and thus helps to reduce overfitting.

Obviously, batchnorm layers are not used in all neural networks, and it is important to test its effectiveness by comparing the performance of the network before and after applying batchnorm (refer to section 3 for results).

2.3 Data augmentation

The quality and number of training data directly affect the generalization of the network. To make the network robust, several transformations are applied to artificially increase the size of the training data. They include rotation, smooth deformations and scale transformations. This can potentially improve the generalization of the network to unseen validation and test images. For our network, we made simple rotations of multiples of 90° and horizontal and vertical flipping. Using all transformation can give us 8-fold increase of the training dataset. We experimented several scales of data augmentation to see how it affects the performance (section 3).

2.4 Binary cross entropy loss

We tried two different cost functions for network optimization: binary cross entropy, mean squared error (MSE), as shown in (1) and (2) respectively. In general, cross entropy loss is more suitable for classification problems. In a way, we are doing pixel-wise binary classification, by distinguishing between salient and non-salient pixels. After training two separate networks, one with MSE and one with cross-entropy loss, we found that the latter gave better performance (refer to section 3 for results).

$$cross\ entropy = mean\left(\sum_{j}^{M}\sum_{i} g_i^j log(s_i^j) + (1 - g_i^j) log(1 - s_i^j)\right) \tag{1}$$

$$MSE = mean\left(\sum_{j}^{M}\sum_{i} (s_i^j - g_i^j)^2\right) \tag{2}$$

where s_i and g_i are predicted and ground truth saliency values of pixels i. We sum over the minibatch of size M and take the mean.

2.5 Other methods

We applied other frequently used methods in our network. First was the weight decay algorithm that helps to stabilize training and achieve improved performance in many modern networks. However, after trying out it was found out that aggressive weight decay hindered the performance, while small weight decay didn't result in the improvements. This might be because the other methods were already providing good regularization.

We also tried adding dropout layers, which usually helps to reduce overfitting. However, dropout resulted in slow training by limiting capacity of the network. As a result, the network performance worsened compared to when not using dropout layers. Therefore, we decided not to use dropout layers in our model.

3 Results

In this section, we present the simulation results under different conditions. Various networks were trained by applying the methods from the section 2 and their performance were compared. We then select the best performing U-Net, and compare to the state-of-the art methods.

3.1 Experimental setup

Training dataset: We used MSRA10K [14] dataset for training. It is one of the widely used datasets for the saliency detection and it contains 10,000 images. When training a network with data augmentation, we used transformations such as flipping up and down and rotations by the multiples of 90°. Images and their binary saliency maps were resized to 256×256 in order to facilitate stochastic training with mini-batches.

Evaluation datasets: We used 3 popular datasets to evaluate the performance of the methods during training or inference. From MSRA-B [15] dataset we chose 200 images for validation set, to analyze how our network evolved during training. Moreover, the decisions of including regularization techniques were also taken according to the performance on this validation set.

For testing the performance we used two datasets: DUT-OMRON [16] and ECSSD [17]. They contain 5,168 and 1,000 images with complex backgrounds respectively. All images from these datasets were resized to 256×256 and passed through the network. Then we resized the output saliency maps back to the original image sizes and compared to the ground truth saliency maps.

General implementation details: Tensorflow framework [18] was used to train the network using Adam optimizer [19] (default settings). The initial learning rate of 0.001 was decayed by 10 times after the 80% of epochs. The size of the mini-batches was set to 5.

3.2 Evaluation metrics

We use mean absolute error (MAE) and F-measure to evaluate the performance of our methods. The average MAE and F-measure are calculated for test sets and are reported in the following sections.

$$MAE = \frac{1}{W \times H} \sum_{x=1}^{W} \sum_{y=1}^{H} |S(x,y) - G(x,y)| \tag{3}$$

where W and H are the width and height of a saliency map. $S(x,y)$ is the predicted saliency at the pixel (x,y) while $G(x,y)$ is the ground truth saliency value.

F-measure is calculated using average precision and average recall of the test set as shown below. We set β to 0.3 following previous works.

$$F_{\beta} = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{4}$$

Precision and recall are calculated using the binarized version of saliency maps. For this we use an image-dependent threshold:

$$T = \frac{2}{W \times H} \sum_{x=1}^{W} \sum_{y=1}^{H} S(x, y) \tag{5}$$

3.3 Deep learning techniques

Importance of batch normalization: In order to find out if the batchnorm layers affected the training we trained two models: one with the batchnorm layers and one without for 50 epochs. The number of feature channels in the U-Net were reduced by a factor of 2 for faster training and no data augmentation was used. As you can see from the training loss and validation average MAE graphs (figures 1), batchnorm helps to stabilize the training and faster convergence of the network. Thus in the later experiments all the models contain batch normalization.

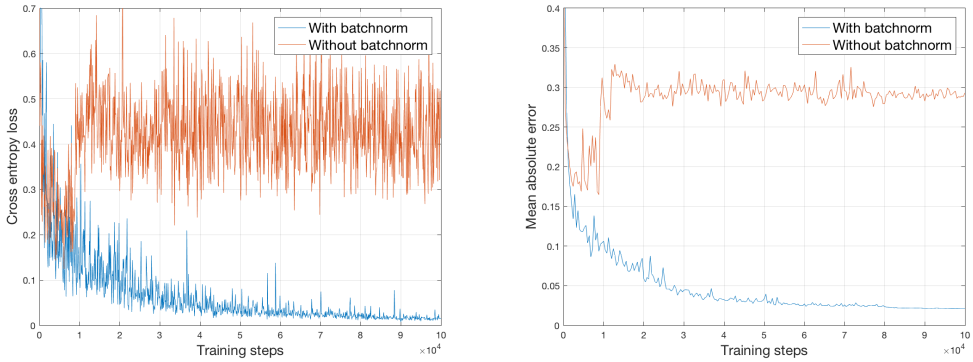

Figure 1: Effect of batchnorm layers on the training process. Cross entropy loss (left) and MAE on validation set (right).

Cross entropy vs. mean squared error: We also train two networks by minimizing cross entropy and MSE losses. Again we use the U-Net with 2 times less feature channel for faster training. No data augmentation was used and training took 50 epochs. The model trained on cross entropy loss resulted on average MAE of 0.0211 on the validation set. Meanwhile, with MSE the performance was worse with average MAE of 0.0252. Thus cross entropy loss was found to be more suitable for our problem and was used in all other models.

The scale of data augmentation: It is beneficial to have high number of training images. However, it doesn't always lead to higher performance. If the network does not have big enough capacity, it will struggle to converge on a big-sized training dataset. Therefore, we trained to several networks with different scales of data augmentation:

- *no_aug* - trained on original training dataset with no data augmentation.
- *aug_2* - applying only 1 transformation (flipping vertically), 2-fold increase of the training dataset
- *aug_4* - applying 4 transformations, 4-fold increase of the training dataset
- *aug_8* - applying all 8 transformations, 8-fold increase of the training dataset

Table 1 demonstrates the performance of the methods on the validation set. We can see that the apart from aug_8, all methods have similar performance. This shows that 8-fold increase in the training dataset does not necessarily lead to the best results. While it is plausible to choose no_aug based on the lowest MAE, it will not necessarily beat others when tested on complex datasets such as DUT-OMRON. Therefore, we report the performance of all 4 methods on both test sets.

3.4 Comparison with other algorithms

We compare our proposed algorithm with other state-of-the-art 8 deep learning based methods and 4 conventional methods (Table 2). We also indicate if the deep learning methods use pretrained

Table 1: Average MAE on validation set of methods with different data augmentation scales

Methods	U-Net+ no_aug	U-Net+ aug_2	U-Net+ aug_4	U-Net+ aug_8
MAE on validation set	**0.0195**	0.2138	0.0226	0.0280

networks such as VGG-16. For qualitative analysis refer to Figure 2, where we show saliency maps outputted by different methods.

As we can see from Table 2, our proposed methods outperform conventional algorithms and show comparable performance to the state-of-the-art deep learning based algorithms. Note that all of them except LEGS [20] use pretrained VGG-16 backbone. Our proposed method doesn't utilize any kind of pretrained networks and is trained from scratch end-to-end. This shows that we can achieve good results by implementing several deep learning techniques on a plain U-NET architecture. PiCANet F-measures were not included, because in their paper they used different thresholds for calculation of precision and recall.

Figure 2 demonstrates the example output saliency maps from different methods. U-Net+ yields relatively good saliency maps retaining important structures.

Table 2: Average F-measure and MAE of methods on the 2 test sets. The best three results are indicated with red, green and blue.

Dataset	DUT-OMRON		ECSSD	
Metric	F_β	MAE	F_β	MAE
Proposed methods				
U-Net+ no_aug	0.634	0.092	0.835	0.086
U-Net+ aug_2	0.636	0.091	0.844	0.081
U-Net+ aug_4	0.611	0.098	0.832	0.082
U-Net+ aug_8	0.616	0.099	0.839	0.077
DL based methods (all except LEGS use VGG-16 backbone)				
DCL [21]	0.684	0.157	0.829	0.150
DS [22]	0.603	0.120	0.826	0.122
ELD [23]	0.611	0.092	0.810	0.080
LEGS [20]	0.592	0.133	0.785	0.118
MDF [24]	0.644	0.092	0.801	0.105
RFCN [25]	0.627	0.111	0.834	0.107
UCF [10]	0.628	0.120	0.852	0.069
PiCANet [11]	-	0.068	-	0.047
Conventional methods				
BL [26]	0.499	0.239	0.684	0.216
BSCA [27]	0.510	0.190	0.705	0.182
DRFI [28]	0.550	0.138	0.733	0.164
DSR [29]	0.524	0.139	0.662	0.178

4 Discussion

U-Net+ method can reliably detect the salient objects in many cases, and their quality is on par with the state-of-the-art algorithms as demonstrated in Figure 2. It performs especially well on high quality images and on images with moderately complex background. From Figure 3 we can see that the saliency maps have clear boundaries and are very close to the ground truth.

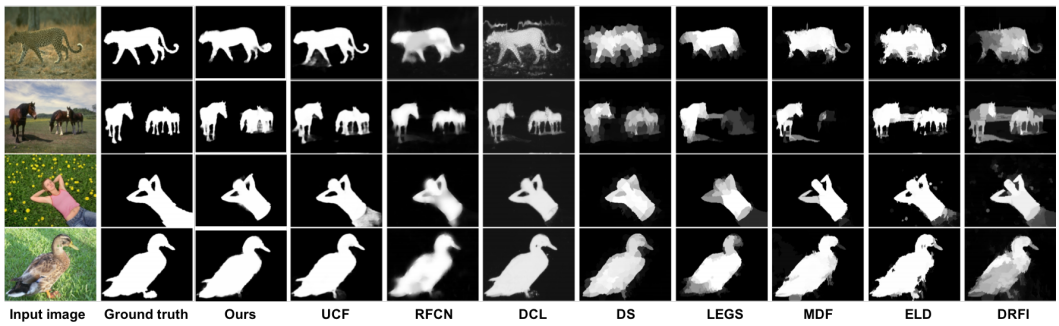

Figure 2: Comparison of methods based on the saliency maps. UNET+ aug_4 is indicated as Ours.

However, there are also cases when U-Net+ outputs poor saliency maps or completely fails to detect the salient objects. In the first and second images on Figure 4 the backgrounds are highly complex. The predicted saliency maps are clearly are not close to the ground truth.

In the third image the saliency map of image of the deer is missing head and antler. Since those parts are slightly blended with the background U-Net+ did not classify them as salient. Perhaps this is where VGG-16 based algorithms can shine, since they can utilize higher level classification. Antler and head are important features for classifying deer, and thus VGG-16 should be able to utilize them more efficiently.

U-Net architecture is versatile and very intuitive and it can potentially achieve higher performance with some tweaks. One of the most important changes could be to train on more complex dataset so that U-Net+ model has better generalization. In addition a regularization term can be added to our cost function to constrain the output to have more detail.

5 Conclusion

In this work a method for saliency detection called U-Net+ was proposed. It utilizes a widely used plain U-Net architecture [8]. Several modifications were implemented including increasing receptive field and sigmoid output layer. We also used deep learning techniques such as batch normalization and data augmentation to stabilize the training process and improve generalization. U-Net+ beat the conventional methods by a wide margin and performed comparably to the state-of-the-art methods that use far more complex networks and techniques. We analyzed the limitations of our model and proposed improvements for future work.

Figure 3: Successful cases of UNET+. Input image (left), ground truth, predicted saliency map (right)

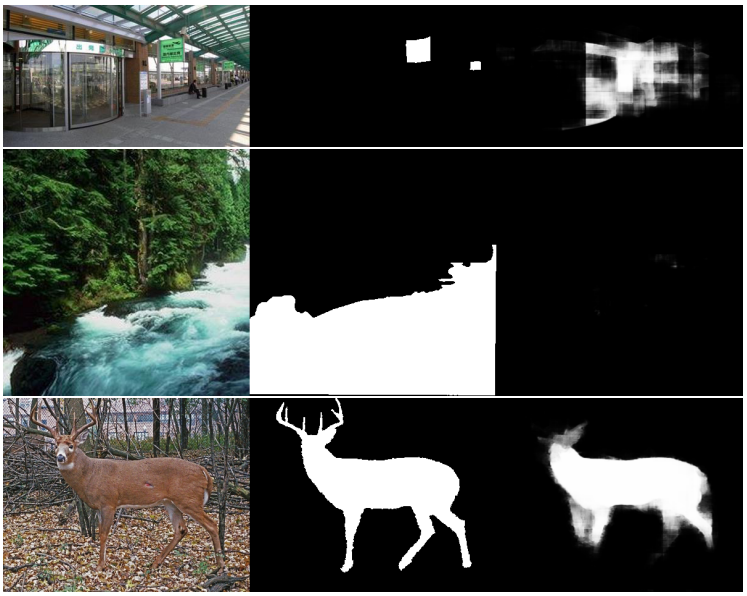

Figure 4: Failure cases of UNET+. Input image (left), ground truth, predicted saliency map (right)

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[2] A Krizhevsky, I Sutskever, and G E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS) 25*, pages 1097–1105, 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] R Girshick, J Donahue, and T Darrell. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS) 28*, pages 91–99, 2015.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representation (ICLR)*, 2015.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[9] Kyong Hwan Jin, Michael T McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26(9):4509–4522, 2017.

[10] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Baocai Yin. Learning uncertain convolutional features for accurate saliency detection. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 212–221. IEEE, 2017.

[11] Nian Liu, Junwei Han, and Ming-Hsuan Yang. Picanet: Learning pixel-wise contextual attention for saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3089–3098, 2018.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[14] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

[15] Jingdong Wang, Huaizu Jiang, Zejian Yuan, Ming-Ming Cheng, Xiaowei Hu, and Nanning Zheng. Salient object detection: A discriminative regional feature integration approach. *International Journal of Computer Vision*, 123(2):251–268, 2017.

[16] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3166–3173, 2013.

[17] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1155–1162, 2013.

[18] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan,

Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pages 265–283, 2016.

[19] Diederik P Kingma and Jimmy Ba. Adam - A Method for Stochastic Optimization. In *International Conference on Learning Representation (ICLR)*, 2015.

[20] Lijun Wang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Deep networks for saliency detection via local estimation and global search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3183–3192, 2015.

[21] Guanbin Li and Yizhou Yu. Deep contrast learning for salient object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–487, 2016.

[22] Xi Li, Liming Zhao, Lina Wei, Ming-Hsuan Yang, Fei Wu, Yueting Zhuang, Haibin Ling, and Jingdong Wang. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE Transactions on Image Processing*, 25(8):3919–3930, 2016.

[23] Gayoung Lee, Yu-Wing Tai, and Junmo Kim. Deep saliency with encoded low level distance map and high level features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 660–668, 2016.

[24] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5455–5463, 2015.

[25] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Saliency detection with recurrent fully convolutional networks. In *European Conference on Computer Vision*, pages 825–841. Springer, 2016.

[26] Na Tong, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Salient object detection via bootstrap learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1884–1892, 2015.

[27] Yao Qin, Huchuan Lu, Yiqun Xu, and He Wang. Saliency detection via cellular automata. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–119, 2015.

[28] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013.

[29] Xiaohui Li, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via dense and sparse reconstruction. In *Proceedings of the IEEE international conference on computer vision*, pages 2976–2983, 2013.