Learning to segment from object sizes

Denis Baručić[1], Jan Kybic[1]

[1]*Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague*

Abstract

Deep learning has proved particularly useful for semantic segmentation, a fundamental image analysis task. However, the standard deep learning methods need many training images with ground-truth pixel-wise annotations, which are usually laborious to obtain and, in some cases (e.g., medical images), require domain expertise. Therefore, instead of pixel-wise annotations, we focus on image annotations that are significantly easier to acquire but still informative, namely the size of foreground objects. We define the object size as the maximum Chebyshev distance between a foreground and the nearest background pixel. We propose an algorithm for training a deep segmentation network from a dataset of a few pixel-wise annotated images and many images with known object sizes. The algorithm minimizes a discrete (non-differentiable) loss function defined over the object sizes by sampling the gradient and then using the standard back-propagation algorithm. Experiments show that the new approach improves the segmentation performance.

Keywords

semantic segmentation, weakly-supervised learning, deep learning, distance transform

1. Introduction

Semantic segmentation is the process of associating a class label to each pixel of an image. With the advent of deep learning, deep networks have achieved incredible performance on many image processing tasks, including semantic segmentation. Deep learning for semantic segmentation has many benefits; for example, it is flexible w.r.t. the model architecture and scales particularly well [1, 2]. On the contrary, the standard deep learning demands many *ground-truth* (GT) pixel-wise annotations to prevent overfitting. Since a human expert annotator must usually provide the GT annotations, acquiring a good-quality training dataset can be difficult. To combat this issue, we focus on learning from GT image annotations that are easier to produce but still informative enough, namely the sizes of foreground objects. In practice, our approach assumes a training dataset that consists of relatively few pixel-wise annotated images and many images with known object sizes. We present a work-in-progress solution.

1.1. Proposed approach

Suppose a standard convolutional network for image segmentation (e.g., a U-Net [3]). Given an input image, we feed it to the network and collect the output prediction. The prediction is then thresholded to obtain a binary mask, which is processed by a distance transform, assigning to each foreground pixel the shortest distance to the background. Finally, the object size is defined as double the maximum of the computed distances.

Due to the thresholding, the cost function is not differentiable and it is therefore not possible to use the standard gradient descent for learning. We overcome this obstacle by adding random noise to the output of our network. The predicted binary masks then become stochastic and the gradient can be sampled. A detailed description of our method is given later in Sec. 2 and 3.

1.2. Related work

Cano-Espinosa et al. [4] considered a similar learning problem. They proposed a network architecture that performs a biomarker (fat contents) regression and image segmentation after being trained directly on images annotated by biomarker values only. Similarly to ours, their method derives the biomarker value from the predicted segmentation deterministically. The difference is that their biomarker, equivalent to the foreground area, can be obtained by a simple summation. Furthermore, the method assumes that the foreground objects can be roughly segmented using thresholding. Pérez-Pelegrí et al. [5] took a similar approach. Although their method does not involve thresholding to produce approximate segmentation, it was tailored explicitly for learning from images annotated by the foreground volume (as their images are 3D).

Karam et al. [6] implemented a differentiable distance transform via a combination of the convolution operations. The method is fast but exhibits numerical instabilities for bigger images. Resolving the numerical instabilities, Pham et al. [7] later proposed a cascaded procedure with locally restricted convolutional distance transforms. Nonetheless, both methods substitute the minimum func-

tion with the *log-sum-exp* operation, which leads to inaccurate results.

The way our method deals with a non-differentiable cost function is borrowed from stochastic binary networks [8]. In a stochastic binary network, one needs to deal with zero gradient after each layer of the network. However, methods such as ARM [9] or PSA [10] are unnecessarily complex. Instead, we employ a single sample estimation, which has been discussed in [11].

2. Model

The proposed model consists of (1) a segmentation network, $f_{\boldsymbol{\theta}}$, parametrized by $\boldsymbol{\theta}$, and (2) a deterministic algorithm to derive the object size based on distance transform, denoted as g.

Given an input image $\boldsymbol{x} = (x_1, \ldots, x_V)$, the network produces a pixel-wise segmentation

$$\boldsymbol{a} = f_{\boldsymbol{\theta}}(\boldsymbol{x}), \qquad (1)$$

such that $a_i \in \mathbb{R}$, $1 \leq i \leq V$, where V is the number of pixels. The method does not make any assumptions about the network's technical details, except that it can be trained using the standard back-propagation algorithm and gradient descent. In our experiments, we always employed a U-Net [3] with a residual network encoder [12] and a mirroring decoder.

To obtain a binary mask $\hat{\boldsymbol{y}} \in \{\pm 1\}^V$, the network response $\boldsymbol{a}$ is thresholded,

$$\hat{y}_i = \operatorname{sign} a_i. \qquad (2)$$

2.1. Object size

We use a distance transform of the binary mask to define the object size (see Fig. 1). Distance transform assigns to each pixel the shortest distance to the background, i.e.,

$$d_i = \min_{j, \hat{y}_j = -1} \delta(i, j), \quad i = 1, \ldots, V, \qquad (3)$$

where $\delta(i, j)$ is the Chebyshev ℓ_∞ distance. After that, we take double the maximum distance to define the object size,

$$\hat{s} = 2 \max_i d_i. \qquad (4)$$

The composition of the distance transform and the maximum aggregation determines the object size, denoted as $g \colon \{\pm 1\}^V \to \mathbb{R}$,

$$g(\hat{\boldsymbol{y}}) = 2 \max_i \min_{j, \hat{y}_j = -1} \delta(i, j). \qquad (5)$$
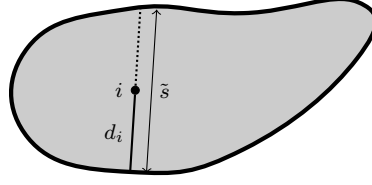

Figure 1: Illustrative example of an object and its derived size. The object is outlined by the thick boundary line. The point i denotes the foreground pixel whose shortest distance to the background, d_i, is the highest among the pixels. The derived object size $\hat{s} = 2d_i$.

2.1.1. Implementation details

There is an efficient, *two-pass* algorithm that computes the distance transform in $\Theta(V)$ time. Furthermore, when evaluating a batch of images, it is possible to compute the distance transform on all images in parallel.

We have implemented a CPU version[1] of this algorithm that works with PyTorch tensors and is faster than, e.g., the SciPy implementation.

3. Learning

Suppose a training dataset $\mathcal{D} = \mathcal{D}_f \cup \mathcal{D}_w$ consists of fully- and weakly-annotated subsets $\mathcal{D}_f$ and $\mathcal{D}_w$. The fully-annotated subset $\mathcal{D}_f$ contains pairs $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is an input image and $\boldsymbol{y}$ the corresponding GT pixel-wise segmentation, while $\mathcal{D}_w$ comprises of pairs $(\boldsymbol{x}, s)$, where s is the size of the object present in the image $\boldsymbol{x}$. We focus on situations when $|\mathcal{D}_f| \ll |\mathcal{D}_w|$.

3.1. Supervised pre-training

Our method starts by optimizing a pixel-wise loss w.r.t. the network parameters $\boldsymbol{\theta}$ on the small subset $\mathcal{D}_f$, as in the standard supervised learning. For a particular training pair $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_f$ and the corresponding prediction $\boldsymbol{a} \in \mathbb{R}^V$, the loss function reads

$$\sum_{i=1}^{V} \left(a_i(1 - y_i) + \log(1 + \exp(-a_i)) \right), \qquad (6)$$

which is known as the binary cross-entropy with logits loss. The optimization continues until convergence.

Using proper data augmentation to extend the training dataset, the network tends to recognize useful features and produces decent predictions after this initial stage (see Sec. 4.2).
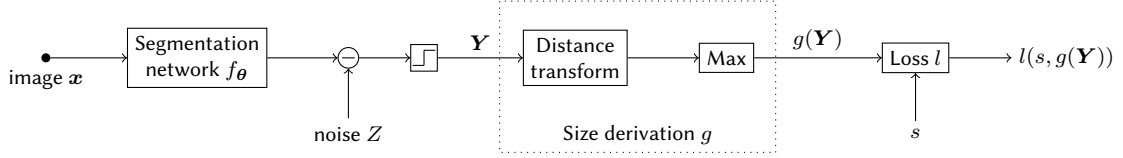
---

[1]https://github.com/barucden/chdt

Figure 2: An overview of the proposed probabilistic model.

3.2. Weakly-supervised training

Consider a training pair $(\boldsymbol{x}, s) \in \mathcal{D}_w$. As described in Sec. 2, one can obtain a prediction of the object size, $\hat{s} = g(\hat{\boldsymbol{y}})$, from the thresholded network response $\hat{\boldsymbol{y}}$. We penalize the prediction error by the square loss

$$l(s, \hat{s}) = (s - \hat{s})^2. \tag{7}$$

We propose to follow an approach similar to those used in binary neural networks [10] and subtract random noise Z from the real predictions a_i before thresholding. Consequently, the binary segmentation becomes a collection $\boldsymbol{Y} = (Y_1, \ldots, Y_V)$ of V independent Bernoulli variables,

$$Y_i = \text{sign}(a_i - Z), \tag{8}$$

with

$$\Pr(Y_i = +1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \Pr(Z \leq a_i) = F_Z(a_i), \tag{9}$$

where F_Z is the cumulative distribution function (CDF) of the noise Z (see Fig. 2).

Then, instead of minimizing the loss l (7), we minimize the expected loss $\mathcal{L} = \mathbb{E}_{\boldsymbol{Y}}[l(s, g(\boldsymbol{Y}))]$,

$$\mathcal{L} = \sum_{\boldsymbol{y} \in \{\pm 1\}^V} \Pr(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) l(s, g(\boldsymbol{y})). \tag{10}$$

Contrary to (7), the expected loss (10) is differentiable, assuming a smooth F_Z.

3.2.1. Noise distribution

Following [10], we sample the noise Z from the logistic distribution with mean $\mu = 0$ and scale $s = 1$. Hence, the CDF of Z is a smooth, sigmoid function,

$$F_Z(a) = \frac{1}{1 + \exp(-a)}. \tag{11}$$

3.2.2. Exact gradient

To compute the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$, we need to evaluate the derivative

$$\frac{\partial \mathbb{E}_{\boldsymbol{Y}}[l(s, g(\boldsymbol{Y}))]}{\partial F_Z(a_i)} \tag{12}$$

for each pixel $i = 1, \ldots, V$. The gradient can be then computed automatically by the back-propagation algorithm. However, an exact computation of (12) leads to

$$\sum_{\boldsymbol{y} \in \{\pm 1\}^V} \frac{\Pr(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})}{\Pr(Y_i = y_i \mid \boldsymbol{x}; \boldsymbol{\theta})} l(s, g(\boldsymbol{y})) y_i, \tag{13}$$

which involves summing 2^V terms and is thus tractable only for very small images. Instead, we resort to a single sample estimator.

3.2.3. Single sample estimator

The single sample estimator is based on Lemma 1, which is, in fact, a specific form of [10, Lemma B.1].

Lemma 1. *Let $\boldsymbol{Y} = (Y_1, \ldots, Y_V)$ be a collection of V independent $\{\pm 1\}$-valued Bernoulli variables with probabilities $\Pr(Y_i = +1) = p_i$. Let h be a function $h \colon \{\pm 1\}^V \to \mathbb{R}$. Let $\boldsymbol{y} = (y_1, \ldots, y_V)$ denote a random sample of $\boldsymbol{Y}$ and $\boldsymbol{y}_{\downarrow i} = (y_1, \ldots, y_{i-1}, -y_i, y_{i+1}, \ldots, y_V)$. Then*

$$y_i (h(\boldsymbol{y}) - h(\boldsymbol{y}_{\downarrow i})) \tag{14}$$

is an unbiased estimate of $\frac{\partial}{\partial p_i} \mathbb{E}_{\boldsymbol{y} \sim \boldsymbol{Y}}[h(\boldsymbol{y})]$.

Proof. We take the derivative of the expectation,

$$\frac{\partial}{\partial p_i} \mathbb{E}_{\boldsymbol{y} \sim \boldsymbol{Y}}[h(\boldsymbol{y})] = \sum_{\boldsymbol{y}} \frac{\Pr(\boldsymbol{y})}{\Pr(y_i)} h(\boldsymbol{y}) y_i, \tag{15}$$

and write out the sum over y_i,

$$\sum_{\boldsymbol{y}_{\neg i}} \sum_{y_i} \Pr(\boldsymbol{y}_{\neg i}) h(\boldsymbol{y}) y_i = \sum_{\boldsymbol{y}_{\neg i}} \Pr(\boldsymbol{y}_{\neg i}) \sum_{y_i} h(\boldsymbol{y}) y_i \tag{16}$$

where $\boldsymbol{y}_{\neg i}$ denotes vector $\boldsymbol{y}$ with the i-th component omitted. Notice that the inner sum simplifies and no longer depends on y_i,

$$\sum_{\boldsymbol{y}_{\neg i}} \Pr(\boldsymbol{y}_{\neg i})(h(\boldsymbol{y}_{i=+1}) - h(\boldsymbol{y}_{i=-1})), \tag{17}$$

where $\boldsymbol{y}_{i=z}$ is the vector $\boldsymbol{y}$ with the i-th component set to z. Then, we multiply the inner subtraction by the constant factor $1 = p_i + (1 - p_i) = \sum_{y_i} \Pr(y_i)$,

$$\sum_{\boldsymbol{y}_{\neg i}} \Pr(\boldsymbol{y}_{\neg i}) \sum_{y_i} \Pr(y_i)(h(\boldsymbol{y}_{i=+1}) - h(\boldsymbol{y}_{i=-1})), \tag{18}$$
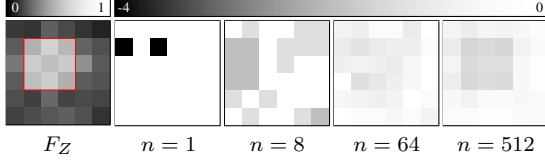
Figure 3: Examples of derivatives (12) computed according to (21) for different number of samples n, given the output of F_Z, for a small, 6×6 image. The red frame outlines the object.

ultimately leading to the following expression for (15):

$$\sum_{\boldsymbol{y}} \Pr(\boldsymbol{y})(h(\boldsymbol{y}_{i=+1}) - h(\boldsymbol{y}_{i=-1})), \qquad (19)$$

which can be written as

$$\sum_{\boldsymbol{y}} \Pr(\boldsymbol{y}) y_i \left[h(\boldsymbol{y}) - h(\boldsymbol{y}_{\downarrow i}) \right]. \qquad (20)$$

Thus, (14) is a single sample unbiased estimate of (15). $\qquad \square$

According to Lemma 1, an unbiased estimate of the derivative (12) is

$$\frac{\partial \, \mathbb{E}_Y [l(s, g(Y))]}{\partial F_Z(a_i)} \approx y_i \left[l(s, g(\boldsymbol{y})) - l(s, g(\boldsymbol{y}_{\downarrow i})) \right], \tag{21}$$

where $\boldsymbol{y}$ is a random sample of Bernoulli variables with probabilities (9) (see a few examples of sampled derivatives in Fig. 3).

4. Experiments

The proposed method was implemented in the PyTorch Lightning framework[2] using a ResNet implementation from the Segmentation Models PyTorch library[3]. The presented experiments were perfomed on a server equipped with Intel Xeon Silver 4214R (2.40GHz) and NVIDIA GeForce RTX 2080 Ti.

The data for our experiments was based on a dataset of 3D MRI images of the hippocampus [13]. The dataset consists of 394 volumes provided with GT segmentation of classes *hippocampus head*, *hippocampus body*, and *background*. We decomposed the volumes into individual 2D slices of size 48×32 pixels and kept only those with at least 1% foreground, obtaining a total of 6093 images. Next, we merged the *hippocampus* classes to get a binary segmentation problem (see Fig. 4). Afterward, we derived the object sizes from the GT pixel-wise annotations to use in training. Finally, we randomly split the data into
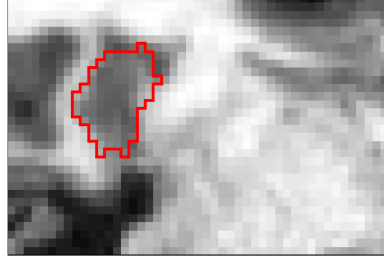

Figure 4: Example of a hippocampus image [13] with the object outlined in red.
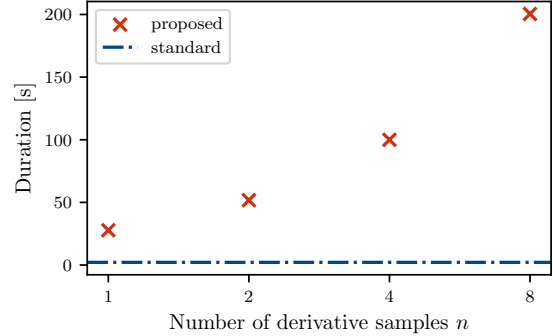

Figure 5: Average epoch duration for the proposed method with different number of gradient samples. The duration of the standard method is given as a reference.

training, validation, and testing subsets containing 70%, 10%, and 20% of the images.

Given a GT segmentation $\boldsymbol{y}$ and a predicted segmentation $\hat{\boldsymbol{y}}$, we evaluate two metrics, the squared size prediction error E and the intersection-over-union IoU,

$$E(\boldsymbol{y}, \hat{\boldsymbol{y}}) = l(g(\boldsymbol{y}), g(\hat{\boldsymbol{y}})), \qquad (22)$$

$$IoU(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{\sum_{i=1}^{V} 1 + y_i + \hat{y}_i + y_i \hat{y}_i}{\sum_{i=1}^{V} 3 + y_i + \hat{y}_i - y_i \hat{y}_i}. \qquad (23)$$

In the case of standard supervised method, vertical and horizontal flipping was randomly applied to augment the training dataset. The proposed method did not apply any augmentation.

4.1. Number of derivative samples

A toy example (see Fig. 3) indicated that taking more samples of the derivatives (21) might lead to better results than taking just one. This experiment investigates how the number of derivative samples n impacts learning speed and prediction quality.

We considered four different numbers of samples n, $n \in \{1, 2, 4, 8\}$. For each n, the other parameters (such as the batch size or the learning rate) were the same, and
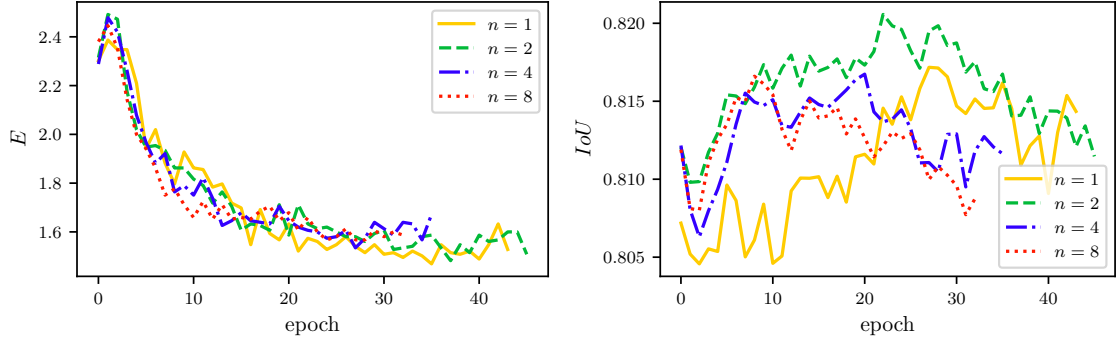
Figure 6: Development of the squared size prediction error E and the intersection-over-union IoU on the validation images over the course of learning for different numbers of derivative samples n.

the learning began with the same segmentation network f_θ that was pre-trained in the standard way on 85 pixel-wise annotated images from the training subset. The proposed method always ran until the squared error E on the validation data stopped improving.

To assess the learning speed, we measured the duration of one learning epoch. For $n = 1$, an epoch took $\approx 10\times$ longer than the standard supervised learning. Generally, the duration grew roughly exponentially with n (see Fig. 5).

Higher values of n did not lead to a lower E or a faster convergence speed (see Fig. 6). In fact, $n = 1$ and $n = 2$ achieved the lowest E, but not by a large margin. Given the speed benefits, we use $n = 1$ always. Interestingly, even though E kept decreasing over the course of learning for all n, IoU improved only slightly and started declining after ≈ 20 epochs. This observation suggests that the squared error of the object size is not a sufficient objective for learning the segmentation.

4.2. Pre-training impact

This experiment tests the essential question: given a segmentation model trained on a few pixel-level annotated images, can we improve its testing performance by further learning from size annotations?

We trained different segmentation networks until convergence on randomly selected training subsets of size m. Then, we fine-tuned these networks on the whole training dataset using the proposed method. We measured the test performance in terms of IoU.

The proposed method led to a $\approx 5\%$ increase of IoU for small $m < 100$ (see Fig. 7), improving the segmentation quality. For higher m, the effect was negligible, which complements the observation from the previous experiment that improving the size estimate does not necessarily improve the segmentation quality.
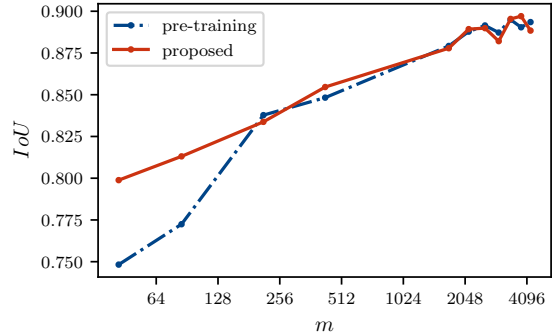

Figure 7: IoU on the test data for different sizes m of the pre-training dataset. The plot shows results achieved by a network after pre-training and after subsequent fine-tuning by the proposed method.

5. Discussion

The method is promising but there is definitely potential for improvement in both speed and prediction performance.

The proposed method samples the derivatives according to (21) for each pixel i. However, flipping the prediction, $y_i \mapsto -y_i$, changes the derived size only for some i; particularly those within and on the border of the predicted object. Therefore, given a sample $\boldsymbol{y}$, $l(s, g(\boldsymbol{y})) = l(s, g(\boldsymbol{y}_{\downarrow i}))$ for many pixels i, and the sampled derivatives (21) are sparse. The method might sample only those derivatives that are potentially non-zero and set the rest to zero directly, which would save much computational time.

We have seen in the experiments that lower size prediction error does not strictly imply better segmentation. We need to closely investigate in what cases the size prediction loss is insufficient and adjust the objective. The adjustment might involve adding an L1 regularization (as in [4]) or drawing inspiration from unsupervised meth-

ods (e.g., demand for the segmentation to respect edges in images, etc.).

The proposed approach entails some principled limitations. For example, it allows only a single object in an image. We also expect the method to be ill-suited for complex object shapes, but we have not performed any experiments in that regard yet.

6. Conclusion

We proposed a weakly-supervised method for training a segmentation network from a few pixel-wise annotated images and many images annotated by the object size. The key ingredients is a method for evaluating the object size from a probabilistic segmentation and a method for optimizing a deep network using a non-differentiable objective.

The achieved results seem promising. We believe the improvements suggested in the discussion will improve performance, rendering the method valuable for training segmentation models for biomedical images.

Acknowledgments

References

[1] X. Liu, et al., A review of deep-learning-based medical image segmentation methods, Sustainability 13 (2021) 1224.

[2] S. Minaee, et al., Image segmentation using deep learning: A survey, IEEE transactions on pattern analysis and machine intelligence (2021).

[3] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.

[4] C. Cano-Espinosa, et al., Biomarker localization from deep learning regression networks, IEEE Transactions on Medical Imaging 39 (2020) 2121–2132.

[5] M. Pérez-Pelegrí, et al., Automatic left ventricle volume calculation with explainability through a deep learning weak-supervision methodology, Computer Methods and Programs in Biomedicine 208 (2021) 106275.

[6] C. Karam, K. Sugimoto, K. Hirakawa, Fast convolutional distance transform, IEEE Signal Processing Letters 26 (2019) 853–857.

[7] D. D. Pham, G. Dovletov, J. Pauli, A differentiable convolutional distance transform layer for improved image segmentation, in: DAGM German Conference on Pattern Recognition, Springer, 2020, pp. 432–444.

[8] T. Raiko, M. Berglund, G. Alain, L. Dinh, Techniques for learning binary stochastic feedforward neural networks, in: 3rd International Conference on Learning Representations, 2015.

[9] M. Yin, M. Zhou, ARM: augment-REINFORCE-merge gradient for stochastic binary networks, in: 7th International Conference on Learning Representations, 2019.

[10] A. Shekhovtsov, V. Yanush, B. Flach, Path sample-analytic gradient estimators for stochastic binary networks, Advances in Neural Information Processing Systems 33 (2020) 12884–12894.

[11] Y. Cong, M. Zhao, K. Bai, L. Carin, GO gradient for expectation-based objectives, in: 7th International Conference on Learning Representations, 2019.

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[13] A. L. Simpson, et al., A large annotated medical image dataset for the development and evaluation of segmentation algorithms, 2019. arXiv:arXiv:1902.09063.