# Volumetric Medical Image Segmentation with Deep Convolutional Neural Networks

© Manvel Avetisian
Lomonosov Moscow State University
avetisian@gmail.com

**Abstract.** This paper presents a neural network architecture for segmentation of medical images. The network trains from manually labeled images and can be used to segment various organs and anatomical structures of interest. We propose an efficient reformulation of a 3D convolution into a series of 2D convolutions in different dimensions. A loss function that directly optimizes intersection-over-union metric popular in medical image segmentation field is proposed.

**Keywords:** medical image segmentation, convolutional neural networks, deep learning, convolution, loss function.

## 1 Introduction

Medical image is a visual representation of the interior of a body; it reveals internal anatomical structures and thus can be used for clinical analysis, intervention planning etc.

Volumetric medical images are obtained from various medical image acquisition technologies, such as computed tomography (CT), magnetic resonance tomography (MRT), etc. These images are represented by a stack of 2D image slices thus forming a 3D representation of a body [2].

Medical image segmentation is an automatic or semi-automatic process of splitting a medical image into regions, which may correspond to an organ, a tissue, a tumor, or any other anatomical structure of interest.

Some of the applications of medical image segmentation are surgical planning, virtual simulation of surgeries, tumor detection and segmentation, brain development study, functional mapping, automated classification of blood cells, mass detection in mammograms, image registration, heart segmentation and analysis of cardiac images, border detection in angiograms of coronary, etc.

Earliest medical image segmentation techniques were based on low-level processing of image data (comparing gray level values of voxels to one or multiple thresholds, edge detector filters, unsupervised clustering algorithms etc.).

Later, supervised techniques, where training data (manually labeled examples) is used to train a model, became increasingly popular. Examples of such methods are maximum likelihood and expectation maximization methods, maximum a posteriori and Markov random field methods, deformable models (active contour models, level set models), atlas-based models, conditional random field, graph cut algorithms.

**Proceedings of the XIX International Conference “Data Analytics and Management in Data Intensive Domains” (DAMDID/RCDL’2017), Moscow, Russia, October 10-13, 2017**

Convolutional neural networks had their applications in image segmentation, but did not gather momentum until various new techniques and computing architectures were developed. In December 2012 CNNs won ImageNet challenge for the first time. AlexNet [5] architecture proposed by Krizhevsky et al. won the competition by large margin. In subsequent years, further progress has been made [6][7]. Convolutional neural networks have became technique of choice, showing state of the art results in computer vision.

A supervised learning algorithms experience a dataset, consisting of examples, each of which contains features $x_i$ and a target $y_i$. For example, popular Iris dataset contains measurements of various species of iris plants. A supervised learning algorithm can study the Iris dataset and learn to classify iris plants into three different species based on their measurements. In our task, $x_i$ can be a computed tomography medical image, while $y_i$ can be a segmentation of that image done by an experienced radiologist.

An artificial neural network consists of many simple units called neurons. Neurons receive and send information via weighted connections. Each neuron calculates weighted sum of inputs and applies nonlinear activation function $f$ to them:

$$h(x; w, b) = f\left(\sum_i x_i * w_i + b\right).$$

Historically popular choices for activation functions were $sigmoid$ and $tanh$, where

$$sigmoid(x) = \frac{1}{1 + e^{-x}}.$$

Recently, one of the most popular activation functions used in computer vision are rectified linear units (ReLU) defined as:

$$relu(x) = \begin{cases} x, if\ x \geq 0 \\ 0, otherwise \end{cases}.$$

In a simple feed forward architecture, neurons are organized into groups called layers. Neurons in the first layer (called input layer) process information from the environment, while neurons in subsequent layers process information from previous layers. Neurons in the last layer (called output layer) produce information of interest. Because of this multi-layered structure, neural networks show very complex behavior:

$$y = h(\dots h(h(x; w_1, b_1); w_2, b_2) \dots ; w_n, b_n).$$

The universal approximation theorem states that a feed-forward network with a single hidden layer containing a finite number of neurons, can approximate continuous functions. Thus, the theorem states that simple neural networks can represent a wide variety of interesting functions when given appropriate number of parameters [1].

Convolutional neural networks (CNNs) are type of artificial neural networks specialized for processing data that has grid-like topology. Examples of such data domains include 1D time-series data or 2D or 3D images. Given two-dimensional image $I$ and kernel $K$, convolution operation can be defined as [1]:

$$(I * K)(i,j) = \sum_{m,n} I(i-m, j-n) * K(i,j).$$

Combining outputs of convolutions from earlier layers with new convolutions on later layers, a neural network can learn very complex features. Usually, first layers of convolutional neural network detect edges and angles, while later layers detect more complex features like eyes, hair, wheels, and even deeper layers detect human faces, cars etc. depending on the task at hand.

If we need to transform data in the direction opposite to convolution, i.e., from something that has the shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution, we can use so called transposed convolutions or deconvolutions [14].

In classification task, the final layer of a neural network computes $logit$ scores applying convolutions to the output of previous layer. The $logit$ scores are not bounded, but we would like to model probability distribution from them. In order to convert them to probabilities, a *softmax* function is used:

$$softmax(z) = \frac{\mathrm{e}^{-z_i}}{\sum \mathrm{e}^{-z_i}}.$$

In order to train a neural network, we minimize a loss function $L(x, y, \hat{y};\ \theta)$ with respect to $\theta$, where $\theta = \{w_1, w_2 \dots, b_1, b_2 \dots\}$, $x$ and $y$ are elements of training set, and $\hat{y}$ is a prediction of the network. The loss function used in this paper will be presented in section 2.

The minimization of the loss function is achieved by calculating partial derivatives of the loss function with respect to parameters of the neural network, and then applying small changes to the parameters. The most basic optimization algorithm is stochastic gradient descent algorithm (SGD), which uses following update rule:

$$\theta_i = \theta_i + \frac{\partial L(x, y, \hat{y};\ \theta_i)}{\partial \theta_i} * \alpha,$$

where $\alpha$ is a small constant called learning rate. $\alpha$ is a hyper parameter of learning algorithm, usually good values for $\alpha$ are between $0.01 - 10^{-6}$.

As neural network may have many layers, gradients of the loss function are computed using backpropagation algorithm: first we computed gradients for last layer, and then we compute gradients of preceding layer using chain rule:

$$\frac{\partial L}{\partial \theta_{i-1}} = \frac{\partial \theta_i}{\partial \theta_{i-1}} \frac{\partial L}{\partial \theta_i}.$$

If neural network has many layers gradient information may be lost during this process (this problem is called vanishing gradients problem). One solution to this problem is skip connections: we sum outputs of deeper layers with outputs of more shallow layers, e.g.:

$$h_1 = h(x; w_1, b_1),$$
$$h_2 = h(h_1; w_2, b_2),$$
$$h_3 = h(h_2; w_3, b_3) + h_1.$$

One of the most useful (and most popular) metrics in medical image segmentation is intersection-over-union metric (IoU). For volumes A and B, IoU is defined as:

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

CNNs can directly classify each pixel of an image individually, given that we present to it a patch of image around pixel of interest. However, a drawback of this naïve sliding window approach is that input patches of neighboring pixels have a huge overlap, and thus some convolutions would be computed many times [2].

A significant speedup can be achieved if we present many pixels to a CNN simultaneously. One of the first implementations of this idea, that were successful in medical image segmentation, were Fully Convolutional Neural Networks (fCNN) [3]. fCNNs added upsampling layers to popular classification neural network architectures, such as AlexNet [6], VGG16 [7], and GoogLeNet [8]. This solution allowed CNN to produce a likelihood map for an entire image rather than a single pixel. The resulting neural network can be applied to an entire input volume in an efficient fashion [3].

The next iteration of fCNN idea is U-Net architecture, where a typical convolutional network architecture (contracting path) is followed by an upsampling layers (expanding path) where the size of an image is increased with upconvolutions. The resulting network forms a U-shape giving the name of the architecture. Another major improvement are skip-connections which directly connect contracting and expanding layers. The architecture showed very good performance on a different biomedical segmentation applications. Thanks to use of data augmentation with elastic deformations, it only needs a very few annotated images and has a very reasonable training time [4].

The 3D U-Net architecture developed ideas of U-Net further to construct a network for volumetric image segmentation that learns from sparsely annotated volumetric images. The implementation replaced all 2D convolutions of U-Net by 3D convolutions. The authors showed a successful application of the proposed method on difficult data set of the Xenopus kidney [5].

## 2 Method

### 2.1 Architecture of the neural network

We used convolution-deconvolution network based on U-Net architecture. The input is processed by blocks of convolutional operations. The data is downscaled with maxpool operations and fed to a next convolutional block in which we would increase number of channels twice to alleviate for loss of resolution. We upscale images with upconvolutions and concatenate data with signals from inner blocks before processing with another convolutional block. Thus, the neural networks forms U-shape with skip connections. Figure 1 summarizes the overall architecture of the neural network.

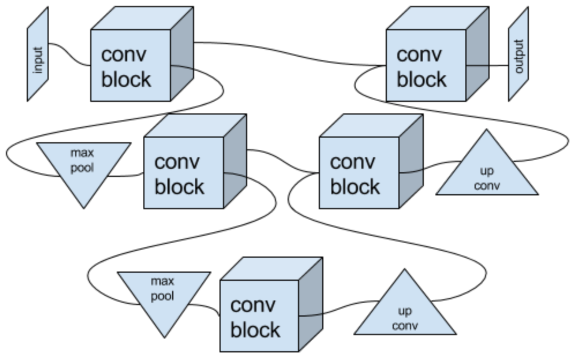

**Figure 1** Architecture of proposed neural network

The skip connections were introduced as it is known that they reduce gradient vanishing problem.

A convolutional block (see Figure 2) consists of four 2D convolutions along different axis. This was done to optimize processing time, as even single 3x3x3 convolution has 27 parameters, while 4 3x3 convolutions have only 36 parameters. Each convolution is followed by ReLU nonlinearity and a dropout.

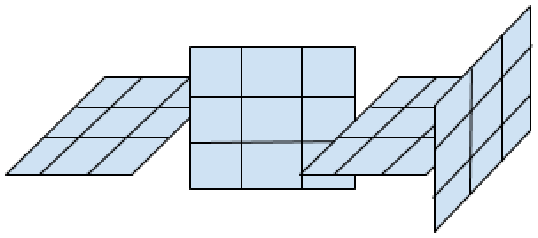

**Figure 2** A convolutional block.

An upconvolution layer has 1x1 kernel which upscales the data, after which we concatenate upscaled data with output of convoltuion block with same size.

### 2.1 Loss function

Our experiments showed that more popular *softmax cross-entropy* function is harder to tune, as it optimizes a metric (accuracy) that we're not interested in and needs tuning of weights of examples. In our setting, IoU metric is much more informative. We extend loss function presented in [13] to multiclass setting. The loss function optimizes IoU metric directly:

$$L(x, y, \hat{y};\ \theta) = \frac{\sum y_{1:} * \widehat{y_{1:}}}{\sum y_{1:} + \sum \widehat{y_{1:}} - \sum y_{1:} * \widehat{y_{1:}}},$$

where $y$ is one-hot encoding of voxel's label, $\hat{y}$ is label probabilities outputted by the network (with *softmax* function). $y_{1:}$ denotes $y$ without the first element.

### 2.2 Implementation details

The proposed method was implemented using TensorFlow library in Python 3 language [4].

A machine with Intel Core i7 6700K CPU, 32 Gb RAM, and NVidia GeForce GTX 1070 GPU was used to train a neural network and perform all experiments.

One of the problems we faced was limited memory of the video card. During training, we were not able to process a full image thus we had to split an image into blocks. This could potentially decrease accuracy for voxels close to the edges of the split because they would have less information about their neighbors. Our experiments showed that this is not a significant problem. To segment an image with trained model, we used TensorFlow's ability to apply convolutions to inputs of variable size to speed up segmentation. However, the architecture of our network forced us to use images with dimensions $2^3 * n$, as otherwise dimensions of upscaled images would not match original images.

The neural network showed strong signs of overfitting. We tested various regularization methods and obtained best results by using dropout right before output layer, as well as l2 regularization of convolutional filters' weights.

### 2.1 Hyperparameters

We performed an extensive search for optimal hyperparameters. Our program would select previous best hyperparameters, randomly generate new ones in interval [$0.1*p_{best}$, $10*p_{best}$], perform 5000 training steps, and select the network which showed higher IoU score on validation set. We summarized final hyperparameters that were used in Table 1.

**Table 1** Best hyperparameters

| **Hyperparameter** | **Value** |
|---|---|
| dropout keep probability | 0.85 |
| l2 regularization weight | $3.0 * 10^{-4}$ |
| learning rate | $7.6 * 10^{-5}$ |
| beta1 | 0.9 |
| beta2 | 0.999 |
| gradient clip | 1.0 |
| channels in first conv layer | 30 |

We used Adam stochastic optimization method. Our experiments showed that using batch normalization is not beneficial in our case and leads to numerical instabilities [5].

In order to minimize numerical problems gradients were clipped to be less than or equal to 1.0.

## 3 Experiments

In order to confirm ability of the neural network to produce segmentation, we conducted an experiment to segment heart's left ventricle using the Cardiac Atlas Project dataset [9]. It consists of 83 volumetric MR images of heart and a mask which highlights region of interest. Figures 4 and 5 show an example of a slice of an image from such dataset, as well a mask for that slice which highlights region of interest. Each image consists of 10-15 slices of various sizes, with 192x192 and 256x256 being the most frequent ones.

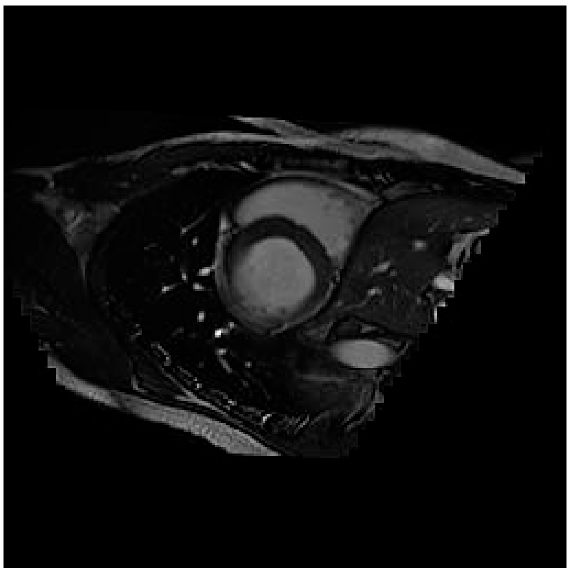
**Figure 4** An example of image from the dataset.

Images in dataset were split to training set, validation set and test set. Validation set was used to tune the parameters of the neural network.

Figure 6 shows segmentation that was obtained using our convolutional neural network. Our model showed quality segmentation with IoU = 0.63.

## 4 Conclusions

Our experiment showed that convolutional neural network is capable of segmenting visually distinguishable anatomical structures on medical images. We plan to extend presented model to more medical image segmentation datasets.

**Support.** This research was supported by the Russian Foundation for Basic Research (grant 16-07-01028).

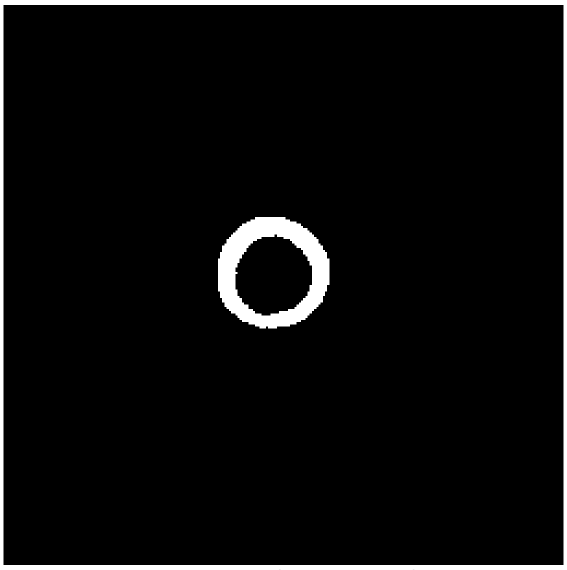
**Figure 5** Segmentation of a ventricle from Fig. 3 by experienced radiologist.

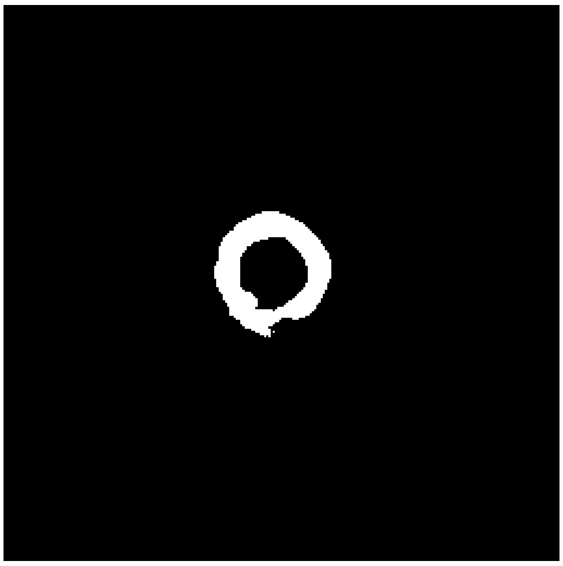
**Figure 6** Segmentation of a ventricle from Fig. 3 by our convolutional neural network.

## References

[1] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. http://www.deeplearningbook.org

[2] Litjens, G., Kooi, T., Bejnordi, B. E.: A Survey on Deep Learning in Medical Image Analysis. https://arxiv.org/abs/1702.05747

[3] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014), arXiv:1411.4038 [cs.CV]

[4] Ronnenberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Medical Image Computing and

Computer-Assisted Intervention -- MICCAI 2015, Part 3, pp. 234-241.

[5] Cicek, O., Abdulkadir, A., Lienkamp, S., Brox, T., Ronneberger, O.: 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015, Part 2, pp. 424-432.

[6] Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[7] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[8] Szegedy, C., Liu, W., Jia, Y. et al.: Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

[9] Fonseca, C.G., Backhaus, M., Bluemke, D.A. et al.: The Cardiac Atlas Project. An imaging database for computational modeling and statistical atlases of the heart. Bioinformatics, 27(16):2288 2295, Aug 2011.

[10] Abadi, M., Agarwal, A., Barham, P.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Kingma, D. P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[12] Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs.LG]

[13] Rahman, A., Wang, Y.: Optimizing Intersection-Over-Union in DeepNeural Networks for Image Segmentation. In: Advances in Visual Computing – ISVC 2016 Proceedings, Part I, pp.234-244.

[14] Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. arXiv:1603.07285 [stat.ML].