

Deep Neural Networks for Fast Segmentation of 3D Medical Images

Karl Fritscher¹✉, Patrik Raudaschl¹, Paolo Zaffino², Maria Francesca Spadea², Gregory C. Sharp³, and Rainer Schubert¹

¹ University for Health Sciences, Medical Informatics and Technology (UMIT),
Innsbruck, Austria

karl.fritscher@umit.at

² Magna Graecia University, Catanzaro, Italy

³ Massachusetts General Hospital, Boston, MA, USA

Abstract. During the last years Deep Learning and especially Convolutional Neural Networks (CNN) have set new standards for different computer vision tasks like image classification and semantic segmentation. In this paper, a CNN for 3D volume segmentation based on recently introduced deep learning components will be presented. In addition to using image patches as input for a CNN, the usage of orthogonal patches, which combine shape and locality information with intensity information for CNN training will be evaluated. For this purpose a publically available CT dataset of the head-neck region has been used and the results have been compared with other state-of-the art atlas- and model-based segmentation approaches.

The presented approach is fully automated, fast and not restricted to specific anatomical structures. Quantitative evaluation provides good results and shows the great potential of deep learning approaches for the segmentation of medical images.

1 Introduction

During the last years, Deep Learning algorithms have set new standards for several tasks in computer vision. While most core components of Deep Learning have been available for a long time, technical advancements of computational hardware which provide the possibility to work with large training sets and extremely deep network architectures have initiated a rebirth of Neural Networks in computer vision. This is especially true for Convolutional Neural Networks (CNN) [1]. The introduction of the AlexNet by Krizhevski et al. [2] was a milestone for image classification and jumpstarted the development of even deeper and more complex network architectures. In contrast to image classification, where the output for an input image is one class label, segmentation tasks require additional locality output providing one label per pixel. For this purpose, patch based approaches [3] as well as fully convolutional networks [4] have been developed. Despite the large success of CNNs and associated Deep Learning methods in computer vision, only a comparably low number of studies using Deep Learning for medical imaging segmentation have been published. E.g. in [5, 6] CNN have been used for brain (lesion) segmentation, in [7] CNN have been used for knee cartilage

segmentation [7] and in 2015 Roth et al. presented a Deep Learning approach for pancreas segmentation [8]. This under-representation of Deep Learning for medical image segmentation might be due to the limited availability of large training sets and the challenge of processing 3D image information, which further increases the complexity and computational expense of the training process. On the other hand, human anatomy and the way it is represented in volumetric images also provides the possibility to use locality information as well as shape statistics and atlases in combination with Deep Learning methods.

In this paper, we introduce an approach to use CNN for medical image segmentation using recently introduced Deep Learning components. A pseudo 3D patch-based approach that integrates shape and locality information into CNN in order to decrease training time and potentially improve segmentation results will be presented. The method is evaluated using a publically available training set of CT images of the head neck region [9]. Used for the segmentation of parotid gland (PG), submandibular gland (SG) and optic chiasm (OC), the approach proves to be easy to use, fast and at the same time provides state of the art segmentation results.

2 Methods

Like almost all learning based segmentation approaches, the presented framework can be separated into an offline training phase and an online testing/segmentation phase.

Figure 1 is giving an overview on the general workflow for training and testing phase. More details on the single steps during training and testing will be given in the following.

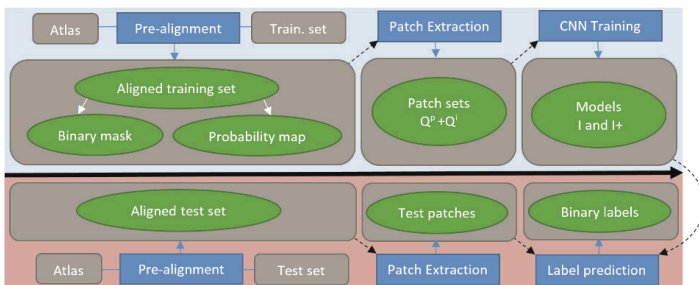


Fig. 1. General workflow of the proposed segmentation approach

2.1 Training Phase

In principle, the training phase is used to train a CNN using a large number of image patches extracted from training images. By this means the network is supposed to learn features which can be used to label each voxel within a region of interest as being either inside or outside a structure of interest.

- a. **Pre-alignment.** However, in order to increase the training speed and at the same time exclude irrelevant information from the training set, patches shall only be

extracted within specific regions around structures of interest. For this purpose, the segmented training images I are pre-aligned onto a publically available reference atlas R of the head-neck area using an affine transformation. The set of preregistered training labels L is used for two purposes: First, the pre-aligned binary labels are summed up and thresholded using

$$t(g) = \begin{cases} 0 & \text{if } g = 0 \\ 1 & \text{if } g \geq 0 \end{cases} \quad (1)$$

where t is a threshold function and g refers to the grey value of a voxel. By this means n binary masks $M_{1..n}$ (n = number of structures) can be created. In order to create a fuzzy boundary and account for shape variations, which are not described by the training set, M is enlarged by including all voxels with a distance $d \leq 5 \text{ mm}$ to the boundary of M in order to obtain the final binary masks $M_{1..n}^L$. During training and testing the binary masks are used to restrict patch extraction to voxels within M^L . Secondly, L is used to create probability maps ($P_{1..n}$), which reflect the likelihood of each voxel in the atlas space to contain either a specific tissue type or background. $P_{1..n}$ are used to integrate shape and locality information into the CNN training process as described in the next section.

- b. **Patch Extraction.** For each structure of interest a separate set of image patches is created. For this purpose, 2D orthogonal patches of size $s \times s$ in the sagittal, coronal and axial plane are extracted using each voxel within M^L as center. As a result 3 sets of 2D intensity training patches $Q_{1..v}^i$ ($v = s * m$, s = number of non-zero voxels within M^L and m = number of training images) are extracted for each structure. Intensity patches $Q_{1..v}^i$ are used as input in order to train a CNN, which will be referred to as *Model I* in the remainder of this paper.

Equivalently, patches with the same dimensionality and orientation as Q^i are extracted from the structures probability map $P_{1..n}$ resulting in a set of probability patches $Q_{1..v}^p$. Patch pairs ($Q_{1..v}^i, Q_{1..v}^p$) are used to train a second type of CNN model referred to as *Model I +*. By integrating $Q_{1..v}^p$ into the training process, it shall be evaluated if CNN can learn locality and shape information for voxels and structures based on the probability maps $P_{1..n}$.

- c. **Neural Network Training.** The architecture of the network used in this project is illustrated in Fig. 2.

Using a feed-forward network, each layer is used to model different representations of the underlying data by using the neurons as feature detectors. The output of each node in the network is computed by using the activation function

$$f\left(\sum w_i * x_i + b\right) \quad (2)$$

where x_i refers to input nodes, w_i and b indicate associated weights and a bias. By concatenating weights and biases of all neurons in a vector θ , the behavior of a

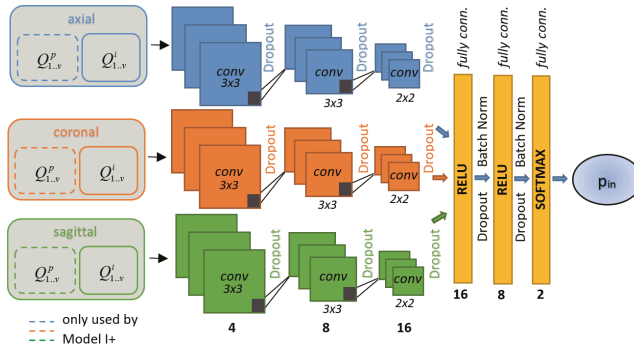


Fig. 2. Architecture of the proposed network

network is completely defined [10]. In a stack of network layers deeper neurons detect new features based on those detected in upper layers resulting in a variety of features which are autonomously learned by the network.

The network presented in this work consists of three different pathways, each made up of 3 convolutional layers. Each pathway corresponds to one of the three patch types (orthogonal, sagittal, coronal). The 3 convolutional layers are used in order to learn lower-level features specific for different patches. After the 3 convolutional layers the pathways are merged followed by 3 fully connected layers. For Model I + each pathway has two input channels, where the first channel corresponds to $Q_{1..v}^i$ and the second channel to $Q_{1..v}^p$, whereas Model I only has one input channel ($Q_{1..v}^i$).

Convolutional Layers. Neurons of convolutional layers are only using a subset of the input voxels commonly also referred to as the receptive field of the neuron. Receptive fields are combined in feature maps, in which all neurons share the same size of weights for the different receptive fields. By this means the number of parameters can be significantly reduced. In order to avoid overfitting, “dropout” as introduced in [11] has been used to train the network. The main idea behind using dropouts is to randomly deactivate a certain percentage of neurons in a layer during each training iteration. By this means, neurons are forced to learn independently potentially leading to a higher number of useful neurons with less redundancy [11].

Fully Connected Layers. After merging the three convolutional pathways three fully connected layers are used in order to learn higher level features. Due to the lack of receptive fields the number of parameters in fully connected layers is typically much higher than in convolutional layers. For this reason a two-step strategy has been used in this project in order to avoid overfitting in fully connected layers: First - equivalently to convolutional layers – dropout has been performed. In addition, an approach called batch normalization [12] has been applied. Batch normalization reduces the internal-covariate-shift problem, which leads to an amplification of input distribution changes in each layer of the network. This phenomenon makes training deeper network significantly harder by requiring the usage of lower learning rates and increasing the risk of getting trapped

in local minima. These problems can be diminished by normalizing the input distribution of a layer to a standard Gaussian distribution [12, 13] in order to circumvent internal-covariate-shifts.

Activation Function. Typically hyperbolic tangent or logistic functions have been used as activation functions (see also Eq. (1)) in Neural Networks. However, especially when using a higher number of layers these activation functions turned out to be susceptible to the vanishing gradients (VG) problem: When back-propagation reaches a neuron with a very small derivative, all nodes behind this neurons only train very slowly or not at all. In order to circumvent this problem Glorot et al. suggested to use the rectifier function $f:x \rightarrow \max(0, x)$ instead [14]. Neurons using rectified functions are also referred to as Rectified Linear Units (RELU) [15]. In the proposed network all neurons except for the neurons in the last (= top) layer are RELUs. In the last layer, a softmax activation function is used. By this means the output of the network can be interpreted as probabilities of a voxel being inside a structure of interest.

Optimization and Weight Initialization. Commonly a stochastic gradient descent (SGD) approach has been used to optimize Deep Neural Nets. However, recently alternatives like Adagrad and Adadelta [16] have been introduced. These methods are easy to use, require no tuning of the learning rate and have shown better and faster convergence than SGD especially in combination with CNNs and sparse inputs. Recently, another stochastic optimization approach has been presented by Kingma [17]. Similar to Adadelta, this Adaptive Moment Estimation method (Adam) computes adaptive learning rates for each parameter. Adam stores an exponentially decaying average of past gradients (m_t) and squared gradients (β_t) in order to estimate the gradients' first and second moments (= mean and variance). Decay rates for m_t and v_t can be tuned using hyperparameters β_1 and β_2 . In this project Adam has been used with an initialization scheme proposed by He et al. which turned out to be a good combination for optimizing RELU networks in other projects [18].

2.2 Testing Phase

In order to segment images using the trained network, the new image has to be transferred in the same reference system as the training images. For this purpose, the atlas that has been used for training is applied. Using a multi-scale approach, masks $M_{1..n}$ are used in the final stage to constrain the metric computation to different regions around the structures of interest. As a result, n affine transforms $A_{1..n}$ are computed and used for pre-alignment. Equivalently to the training phase, patches $Q_{1..v}^i$ and $Q_{1..v}^p$ are extracted. Finally, by using the trained CNN a probability p_{in} of being inside the structure of interest can be generated for each voxel within $M_{1..n}$ of the pre-aligned images and mapped back into the original test image space using the inverse of $A_{1..n}$. Final labels are obtained by thresholding p_{in} .

3 Results

The presented approach has been evaluated using the publically available dataset used for the MICCAI Head Neck Segmentation Challenge 2015 (HNC 2015) [19]. The dataset contains 20 training and 10 test datasets as well as manually segmented labels for 6 different structures. In this project, parotid gland (PG), submandibular gland (SG) as well as optic chiasm (OC) have been used for evaluation. Accuracy of the segmentation is measured using the DICE coefficient ($= 2A \cap B / (A + B)$), where A represents the ground truth and B the results using the proposed approach. In addition, max. Hausdorff Distance is used to quantify the maximum segmentation error.

For patch extraction the images have been resampled to $1.5 \times 1.5 \times 1.5$ mm voxel spacing, patch size was set to 31×31 . The affine pre-registration has been performed using an affine transform in combination with Mutual Information metric to account for the significant noise level in some of the test images. Parameter settings for training the CNN are summed up in Table 1.

Table 1. Summary of parameter setting for generating the CNN

Parameter	Value	Parameter	Value
Initial prob. distribution	he_normal	Loss function	Logloss
β_1, β_2 (Adam optimizer)	0.9, 0.999	dropout rate	0.5
Learning rate (optimizer)	0.05	Epsilon (batch norm)	1e-05
Mini batch size	500		

The CNNs were trained using an Nvidia K6000 GPU. During CNN training, the training set was split into a test (15 datasets) and validation dataset (5 datasets). For each model type (Model I and Model I+), separate CNNs (using equal parameter settings) have been trained for each structure of interest. Between 30000 (Chiasm) and 150000 (Parotid) sample patches have been used for training (training time $\sim 0.5 - 3$ h).

Optimal threshold values for creating binary labels from the CNN output p_{in} have been obtained empirically using the validation set. Table 2 compares the segmentation results obtained with Models I and I+ with the results of HNC 2015 based on avg. Dice and avg. max HD (\pm std. dev).

Table 2. Segmentation results compared to results of HNC 2015

Struct.	DICE			Max HD		
	Model I	Model I+	HNC 2015 (min-max)	Model I	Model I+	HNC 2015 (min-max, mm)
OC	0.52 (± 0.11)	0.49 (± 0.09)	0.35 - 0.57	5.6 (± 1.6)	6.8 (± 1.4)	4.7 - 6.4
PG	0.81 (± 0.04)	0.81 (± 0.04)	0.68 - 0.86	16.9 (± 8.7)	14.2 (± 6.6)	12.4 - 15.1
SG	0.65 (± 0.09)	0.65 (± 0.08)	0.42 - 0.74	12.5 (± 6.7)	9.7 (± 4.8)*	9.0 - 12.4

* Significant difference between avg. HD of Model I and I+ ($p < 0.05$)

Figure 3 is showing a representative result of SG segmentation. When looking at the contour obtained with Model I+ (right) it can be observed that the resulting boundaries

are smoother and more regularized than the result obtained with Model I (left). Higher amount of regularization is also expressed by lower max. HD values for Model I+ (except for OC segmentation, most probably caused by lower pre-alignment accuracy).



Fig. 3. Result of SG segmentation using Model I (left), Model I+ (right) vs manual segmentation (center)

Applying the CNN on a new dataset took between 1 (OC) and ~ 20 s (PG) per dataset. Affine pre-alignment roughly took 1 min per dataset.

4 Conclusion

A fast segmentation approach for volumetric images based on Convolutional Neural Networks has been presented. To the best of our knowledge a deep CNN has not been used and evaluated for segmenting multiple structures in clinical CT images before. In contrast to [8] shape and locality information have been directly integrated into the CNN using atlases and probability maps. In addition, the approach is applying and combining state-of-the-art components for Deep Learning that have been introduced recently and partly have not been used in medical imaging before.

Evaluation was performed using a publically available dataset of the Head & Neck area. Especially compared to highly developed approaches used during HNC15, the results are very promising especially when considering that only 75 % of the HNC datasets have been used for learning (15 vs. 20 dataset for HNC15, 5 datasets have been used as validation set for CNN training). In addition the approach is easy to use and comparably fast. Whereas combining probability patch information with intensity patches does not significantly improve DICE scores, it (significantly) decreases the maximum HD for PG and SG segmentation. This shows the potential of combining image information with meta-information on shape and locality of structures. In future work, the approach will be extended by using more complex CNNs based on residual learning in combination with additional shape descriptors.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advance in Neural Information processing systems*, pp. 1097–1105 (2012)

3. Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: NIPS. pp. 2852–2860 (2012)
4. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014) [arXiv:1411.4038](https://arxiv.org/abs/1411.4038)
5. Brebisson, A., Montana, G.: Deep Neural Networks for Anatomical Brain Segmentation. In: Proceedings of the IEEE CVPR Workshops, pp. 20–28 (2015)
6. Kamnitsas, K., Chen, L., Ledig, C., Rueckert, D., Glocker, B.: Multi-scale 3D convolutional neural networks for lesion segmentation in brain MRI. Ischemic Stroke Lesion Segmentation, p. 13 (2015)
7. Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., Nielsen, M.: Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds.) MICCAI 2013, Part II. LNCS, vol. 8150, pp. 246–253. Springer, Heidelberg (2013)
8. Roth, H.R., Lu, L., Farag, A., Shin, H.-C., Liu, J., Turkbey, E.B., Summers, R.M.: DeepOrgan: multi-level deep convolutional networks for automated pancreas segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015, Part I. LNCS, vol. 9349, pp. 556–564. Springer, Heidelberg (2015)
9. <http://www.imagenglab.com/newsite/pddca>
10. Lai, M.: Deep Learning for Medical Image Segmentation (2015). arXiv preprint [arXiv:1505.02000](https://arxiv.org/abs/1505.02000)
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv preprint [arXiv:1207.0580](https://arxiv.org/abs/1207.0580)
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015). arXiv preprint [arXiv:1502.03167](https://arxiv.org/abs/1502.03167)
13. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of IEEE ICCV, pp. 1520–1528 (2015)
14. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Networks **5**(2), 157–166 (1994)
15. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
16. Zeiler, M.D., ADADELTA: an adaptive learning rate method (2016). arXiv preprint [arXiv:1212.5701](https://arxiv.org/abs/1212.5701)
17. Kingma, D., Ba, J.: Adam: a method for stochastic optimization (2014). arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
18. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)
19. <http://www.midasjournal.org/browse/journal/80>