Training Deep Convolutional Encoder Networks for Multiple Sclerosis Lesion Segmentation

Tom Brosch and Roger Tam

***Abstract*—We propose a novel segmentation approach based on deep convolutional encoder networks and apply it to the segmentation of multiple sclerosis (MS) lesions in magnetic resonance images. Our model is a neural network that has both convolutional and deconvolutional layers, and combines feature extraction and segmentation prediction in a single model. The joint training of the feature extraction and prediction layers allows the model to automatically learn features that are optimized for accuracy for any given combination of image types. In contrast to existing automatic feature learning approaches, which are typically patch-based, our model learns features from entire images, which eliminates patch selection and redundant calculations at the overlap of neighboring patches and thereby speeds up the training. Our network also uses a novel objective function that works well for segmenting underrepresented classes, such as MS lesions. We have evaluated our method on the publicly available labeled cases from the MS lesion segmentation challenge 2008 data set, showing that our method performs comparably to the state-of-the-art. In addition, we have evaluated our method on the images of 500 subjects from an MS clinical trial and varied the number of training samples from 5 to 250 to show that the segmentation performance can be greatly improved by having a representative data set.**

***Index Terms*—Segmentation, multiple sclerosis lesions, MRI, machine learning, unbalanced classification, deep learning, convolutional neural nets**

I. Introduction

MULTIPLE sclerosis (MS) is an inflammatory and demyelinating disease of the central nervous system, and is characterized by the formation of lesions, primarily visible in the white matter on conventional magnetic resonance images (MRIs). Imaging biomarkers based on the delineation of lesions, such as lesion load and lesion count, have established their importance for assessing disease progression and treatment effect. However, lesions vary greatly in size, shape, intensity and location, which makes their automatic and accurate segmentation challenging. Many automatic methods have been proposed for the segmentation of MS lesions over the last two decades, which can be classified into unsupervised and supervised methods. Unsupervised methods do not require a labeled data set for training. Instead, lesions are identified as an outlier class using, e.g., clustering methods [7] or dictionary learning and sparse coding to model healthy tissue [8]. Current supervised approaches typically start with a large set of features, either predefined by the user [3] or gathered in a feature extraction step, which is followed by a separate training step with labeled data to determine which set of features are the most important for segmentation in the particular domain. For example, Yoo et al. [9] proposed performing unsupervised learning of domain-specific features from image patches from unlabelled data using deep learning. A major breakthrough for the fully automatic segmentation of medical images using deep learning comes from the domain of cell membrane segmentation, in which Ciresan et al. [2] proposed to classify the centers of image patches directly using a convolutional neural network (CNN) [5] without a dedicated feature extraction step. Instead, features are learned indirectly within the lower layers of the neural network during training, while the higher layers can be regarded as performing the classification. In contrast to unsupervised feature learning, this approach allows the learning of features that are specifically tuned to the segmentation task. Although deep network-based feature learning methods have shown great potential for image segmentation, the time required to train complex patch-based methods can make the approach infeasible when the size and number of patches are large.

More recently, CNN architectures have been proposed that are able to learn to segment entire images instead of patches [1], [4], [6], which removes the need to select representative patches, eliminates redundant calculations where patches overlap, and therefore scales up more efficiently with image resolution. Kang et al. introduced the fully convolutional neural network to segment crowds in surveillance videos [4]. Fully convolutional neural networks predict segmented images of lower resolution than the input images due to the successive use of convolutional and pooling layers, which both reduce the dimensionality. In our previous work [1], we proposed to use a 3-layer convolutional encoder network (CEN) for multiple sclerosis lesion segmentation, which combines convolutional and deconvolutional layers in order to predict lesion segmentations that have the same resolution as the input images and are therefore potentially more accurate compared to segmentations predicted by fully convolutional neural networks. Ronneberger et al. [6] found that increasing the depth of the CNN, and thereby increasing the size of the receptive field of a particular voxel, increases the abstraction from the input data, which makes the segmentation method more robust to anatomical and intensity variations. They proposed an 11-layer u-shaped network architecture called u-net, which leverages high-level and low-level features in order to predict segmentations that are both robust and accurate. However, the successive application of convolutional, pooling and unpooling layers reduces the size of the predicted segmentation by the size of the receptive field.

T. Brosch is with the MS/MRI Research Group and the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada.

R. Tam is with the MS/MRI Research Group and the Department of Radiology, The University of British Columbia, Vancouver, BC, Canada.

This requires special handling of the border regions and limits the maximum size of the receptive field without compromising training performance (and I mean speed thereby and memory).

The contributions of our paper are two-fold. First, we propose a new convolutional network architecture that combines the advantages of a 3-layer CEN and an 11-layer u-net. Our network is divided into two pathways, a convolutional pathway which consists of alternating convolutional and pooling layers and learns increasingly more abstract and robust features, and a deconvolutional pathway which consists of alternating deconvolutional and unpooling layers and predicts the final segmentation. In order for the segmentation to be driven by both low-level and high-level features, we introduce shortcut connections between layers of the two pathways, similar to the u-net architecture. In contrast to the u-net architecture, our network predicts segmentations that have the same resolution as the input images and therefore does not require special handling of the border of the image. Because the size of the predicted segmentation is independent of the receptive field size, our network architecture can be extended to learn global features, whose receptive fields span the entire image.[1] We have evaluated the segmentation performance of our method on a large data set and compared our results with lesionTOADS, a widely used publicly available method for fully automatic lesion segmentation, showing that our newly proposed architecture further improves segmentation accuracy over the state-of-the-art.

There has been a lot of interest in recent years in the machine learning community to develop better training methods for CNNs. However, training CNNs for medical image segmentation is particularly challenging due to the relatively small size of available training sets and the large size of 3D medical images, which only allows a few iterations to be used for training and hyperparameter tuning. Our second contribution is a comprehensive comparison of different first order and second order training method and parameter initialization strategies that can serve as a guideline for other researchers for training convolutional models for medical image segmentation.

II. Methods

In this paper, the task of segmenting MS lesions is defined as finding a function s that maps multi-modal images I, e.g., $I = (I_{\text{FLAIR}}, I_{\text{T1}})$, to corresponding lesion masks S. Given a set of training images I_n, $n \in \mathbb{N}$, and corresponding segmentations S_n, we model finding an appropriate function for segmenting MS lesions as an optimization problem of the following form

$$\hat{s} = \arg\min_{s \in \mathcal{S}} \sum_n E(S_n, s(I_n)) \tag{1}$$

where $\mathcal{S}$ is the set of possible segmentation functions, and E is an error measure that calculates the dissimilarity between ground truth segmentations and predicted segmentations.

[1]might move that to the discussion or future work, because we haven't explored this feature yet.

A. Model Architecture

The set of possible segmentation functions is modeled by the convolutional encoder network illustrated in Fig. 1.

There are gaps in the description and a general understanding of CNNs is required to fill these gaps in order to be able to implement the CEN.

Our network is divided into two pathways, the convolutional pathway and the deconvolutional pathway. The convolutional pathway consists of alternating convolutional and pooling layers. The input layer of the convolutional pathway is composed of the image voxels $x_i^{(0)}(\vec{p})$, $i \in [1, C]$, where i indexes the modality or input channel, C is the number of modalities or channels, and $\vec{p} \in \mathbb{N}^3$ are the coordinates of a particular voxel. The convolutional layers automatically learn a feature hierarchy from the input images. A convolutional layer is a deterministic function of the following form

$$x_j^{(l)} = \max\left(0, \sum_{i=1} \tilde{w}_{\text{c},ij}^{(l)} * x_i^{(l-1)} + b_j^{(l)}\right) \tag{2}$$

where l is the index of a convolutional layer, $x_j^{(l)}$, $j \in [1, F]$, denotes the feature map corresponding to the trainable convolution filter $w_{\text{c},ij}^{(l)}$, F is the number of filters of the current layer, $b_j^{(l)}$ are trainable bias terms, $*$ denotes valid convolution, and $\tilde{w}$ denotes a flipped version of w. A convolutional layer is followed by an average pooling layer that halves the number of units in each dimension.

The deconvolutional pathway consists of alternating unpooling layers and deconvolutional layers with optional shortcut connections to the corresponding convolutional layers. The first deconvolutional layer uses the extracted features of the convolutional pathway to calculate abstract segmentation features as follows

$$y_i^{(L-1)} = \max\left(0, \sum_{j=1} w_{\text{d},ij}^{(L)} \circledast y_j^{(L)} + c_i^{(L-1)}\right) \tag{3}$$

where $y^{(L)} = x^{(L)}$, L denotes the number of layers of the convolutional pathway, $w_{\text{d},ij}^{(L)}$ and $c_i^{(L-1)}$ are trainable parameters of the deconvolutional layer, and $\circledast$ denotes full convolution. Subsequent deconvolutional layers use the activations of the previous layer and corresponding convolutional layer to calculate less abstract segmentation features as follows

$$\begin{aligned} y_i^{(l)} = \max\Bigg(0, &\sum_{j=1} w_{\text{d},ij}^{(l+1)} \circledast y_j^{(l+1)} \\ &+ \sum_{j=1} w_{\text{s},ij}^{(l+1)} \circledast x_j^{(l+1)} + c_i^{(l)}\Bigg) \end{aligned} \tag{4}$$

where l is the index of a deconvolutional layer with shortcuts, and $w_{\text{s},ij}^{(l+1)}$ are the filter kernels connecting the activations of the convolutional pathway with the activations of the deconvolutional pathway. The last deconvolutional layer integrates the low-level features extracted by the first convolutional layer

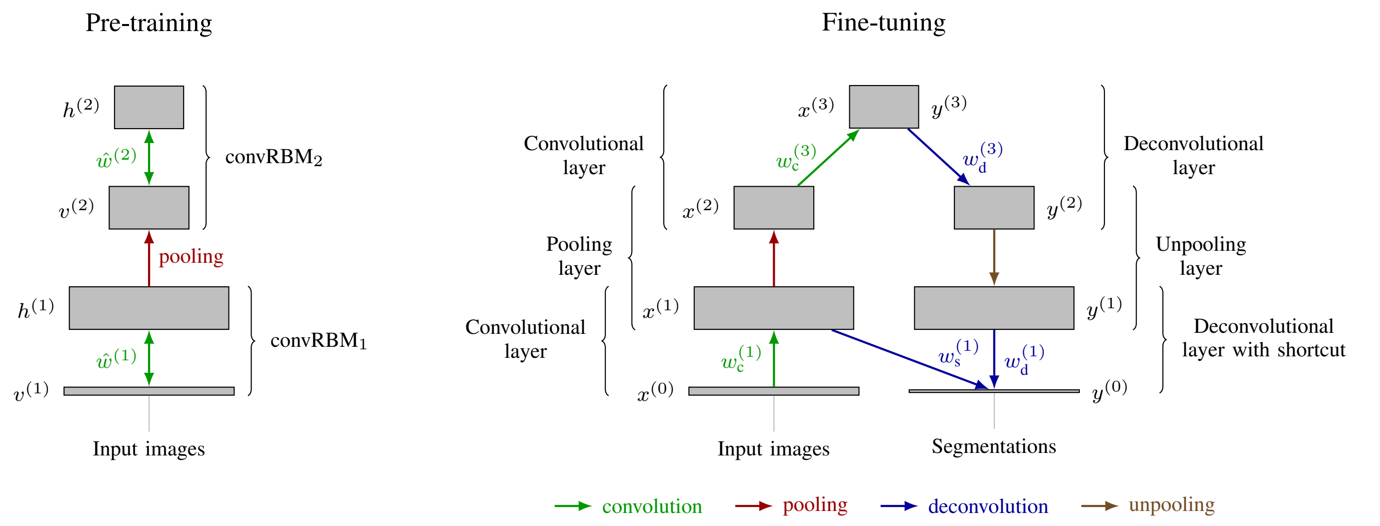

Fig. 1. Pre-training and fine-tuning of a 7-layer convolutional encoder network with shortcuts. Pre-training is performed on the input images using a stack of convolutional RBMs. The pre-trained weights and bias terms are used to initialize a convolutional encoder network, which is fine-tuned on pairs of input images, $x^{(0)}$, and segmentations, $y^{(0)}$.

with the high-level features from the previous layer to calculate a probabilistic lesion mask as follows

$$y_1^{(0)} = \text{sigm}\left(\sum_{j=1}\left(w_{\text{d},1j}^{(1)} \circledast y_j^{(1)} + w_{\text{s},1j}^{(1)} \circledast x_j^{(1)}\right) + c_1^{(0)}\right) \quad (5)$$

where $\text{sigm}(z)$ denotes the sigmoid function defined as $\text{sigm}(z) = (1+\exp(-z))^{-1}, z \in \mathbb{R}$. To obtain a binary lesion mask from the probabilistic output of our model, we chose a fixed threshold such that the mean Dice similarity coefficient is maximized on the training set.

B. Gradient Calculation

The parameters of the model can be efficiently learned by minimizing the error E on the training set, which requires the calculation of the gradient of E with respect to the model parameters. Typically, neural networks are trained by minimizing the sum of squared differences (SSD)

$$E = \frac{1}{2}\sum_{\vec{p}}\left(S(\vec{p}) - y^{(2)}(\vec{p})\right)^2. \quad (6)$$

The partial derivatives of the error with respect to the model parameters can be calculated using the delta rule and are given by

$$\frac{\partial E}{\partial w_{\text{d},ij}^{(l)}} = \delta_{\text{d},i}^{(l-1)} * \tilde{y}_j^{(l)}, \qquad \frac{\partial E}{\partial c_i^{(l)}} = \sum_{\vec{p}} \delta_{\text{d},i}^{(l)}(\vec{p}) \quad (7)$$

$$\frac{\partial E}{\partial w_{\text{s},ij}^{(l)}} = \delta_{\text{d},i}^{(l-1)} * \tilde{x}_j^{(l)} \quad (8)$$

$$\frac{\partial E}{\partial w_{\text{c},ij}^{(l)}} = x_i^{(l-1)} * \tilde{\delta}_{\text{c},j}^{(l)}, \qquad \frac{\partial E}{\partial b_i^{(l)}} = \sum_{\vec{p}} \delta_{\text{c},i}^{(l)}(\vec{p}) \quad (9)$$

The deltas of the first layer can be calculated with

$$\delta_{\text{d},1}^{(0)} = \left(y_1^{(0)} - S\right)y_1^{(0)}\left(1 - y_1^{(0)}\right) \quad (10)$$

The derivatives of the error with respect to the other layers can be calculated by applying the chain rule of partial derivatives and are given by

$$\delta_{\text{d},j}^{(l)} = \left(\tilde{w}_{\text{d},ij}^{(l)} * \delta_{\text{d},i}^{(l-1)}\right)\mathbb{I}\left(y_j^{(l)} > 0\right) \quad (11)$$

$$\delta_{\text{c},i}^{(l)} = \left(w_{\text{c},ij}^{(l+1)} \circledast \delta_{\text{c},j}^{(l+1)}\right)\mathbb{I}\left(x_i^{(l)} > 0\right) \quad (12)$$

where l is the index of a deconvolutional layer, and convolutional layer, respectively, $\delta_{\text{c},i}^{(L)} = \delta_{\text{d},j}^{(L)}$, and $\mathbb{I}(z)$ denotes the indicator function defined as 1 if the predicate z is true and 0 otherwise. If a layer participates in a shortcut connection, $\delta_{\text{c},j}^{(l)}$ needs to be adjusted by propagating the error back through the shortcut connection. In this case, $\delta_{\text{c},j}^{(l)}$ is calculated by

$$\delta_{\text{c},j}^{(l)} = \left(\delta_{\text{c},j}^{(l)\prime} + \tilde{w}_{\text{s},ij}^{(l)} * \delta_{\text{d},i}^{(l-1)}\right)\mathbb{I}\left(x_j^{(l)} > 0\right) \quad (13)$$

where $\delta_{\text{c},j}^{(l)\prime}$ is the activation before taking the shortcut into account.

The sum of squared differences is a good measure of classification accuracy, if the two classes are fairly balanced. However, if one class contains vastly more samples, as is the case for lesion segmentation, the error measure is dominated by the majority class and consequently, the neural network would learn to completely ignore the minority class. To overcome this problem, we use a combination of sensitivity and specificity, which can be used together to measure classification performance even for vastly unbalanced problems. More precisely, the final error measure is a weighted sum of the mean squared difference of the lesion voxels (sensitivity) and non-lesion voxels (specificity), reformulated to be error terms:

$$E = r\frac{\sum_{\vec{p}}\left(S(\vec{p}) - y^{(2)}(\vec{p})\right)^2 S(\vec{p})}{\sum_{\vec{p}} S(\vec{p})} + (1-r)\frac{\sum_{\vec{p}}\left(S(\vec{p}) - y^{(2)}(\vec{p})\right)^2\left(1 - S(\vec{p})\right)}{\sum_{\vec{p}}\left(1 - S(\vec{p})\right)} \quad (14)$$

We formulate the sensitivity and specificity errors as squared errors in order to yield smooth gradients, which makes the optimization more robust. The sensitivity ratio r can be used to assign different weights to the two terms. Due to the large number of non-lesion voxels, weighting the specificity error higher is important, but the algorithm is stable with respect to changes in r, which largely affects the threshold used to binarize the probabilistic output. In all our experiments, a sensitivity ratio between 0.10 and 0.01 yields very similar results.

To train our model, we must compute the derivatives of the modified objective function with respect to the model parameters. Equations 7–9 and 11–13 are a consequence of the chain rule and independent of the chosen similarity measure. Hence, we only need to derive the update rule for $\delta^{(0)}_{\text{d},1}$. With $\alpha = 2r(\sum_{\vec{p}} S(\vec{p}))^{-1}$ and $\beta = 2(1-r)(\sum_{\vec{p}}(1 - S(\vec{p})))^{-1}$, we can rewrite E as

$$E = \frac{1}{2}\sum_{\vec{p}} \left(\alpha S(\vec{p}) + \beta(1 - S(\vec{p}))\right)\left(S(\vec{p}) - y^{(0)}_1(\vec{p})\right)^2 \quad (15)$$

Our objective function is similar to the SSD, with an additional multiplicative term applied to the squared differences. The additional factor is constant with respect to the model parameters. Consequently, $\delta^{(0)}_{\text{d},1}$ can be derived analogously to the SSD case, and the new factor is simply carried over:

$$\delta^{(0)}_{\text{d},1} = \left(\alpha S + \beta(1 - S)\right)\left(y^{(0)}_1 - S\right)y^{(0)}_1\left(1 - y^{(0)}_1\right) \quad (16)$$

C. Training

At the beginning of the training procedure, we have to set the model parameters to some initial values, which are then fine-tuning during training. The choice of the initial parameters can have a big impact on the learned model. There are two ways to initialize a new model: a) using random values drawn from some random distribution, or b) using pre-training. Pre-training can be performed layer by layer using a stack of convolutional restricted Boltzmann machines (convRBMs) (see Figure 1), thereby avoiding the potential problem and diminishing or exploding gradients [??]. The first convRBM is trained on the input images without the need for labeled data. Subsequent convRBMs are trained on the hidden activations of the previous convRBM. After all convRBMs have been trained, the model parameters of, e.g., the first convolutional and last deconvolutional layer of the CEN can be initialized as follows

$$w^{(1)}_\text{c} = \hat{w}^{(1)}, \quad w^{(1)}_\text{d} = 0.5\hat{w}^{(1)}, \quad w^{(1)}_\text{s} = 0.5\hat{w}^{(1)} \quad (17)$$

$$b^{(1)} = \hat{b}^{(1)}, \quad c^{(0)} = \hat{c}^{(1)}, \quad (18)$$

where $\hat{w}^{(1)}$ are the filter weights, $\hat{b}^{(1)}$ are the hidden bias terms, and $\hat{c}^{(1)}$ are the visible bias terms of the first convRBM, respectively. Alternatively, the bias terms can be initialized with zero and the filter weights are initialized using random values that are drawn from a distribution such that the variance of the activations of the layer units remains roughly the same throughout all layers. This can be achieved by drawing samples from the normal distribution $\mathcal{N}(0, \sigma)$, where $\sigma = \sqrt{2/N}$, and N denotes the number of incoming connections for one unit. The influence of the initialization strategy on the learned model is analyzed in Section ??.

A major challenge for gradient-based optimization methods is the choice of an appropriate learning rate. Classic stochastic gradient descent [??] uses a fixed or decaying learning, which is the same for all parameters of the model. However, the partial derivatives of parameters of different layers can vary substantially in magnitude, which might require different learning rates for the parameters of different layers. Many methods have been proposed to automatically chose an individual learning rate for each parameter of the model. Most methods (e.g., AdaGrad [??], AdaDelta [??], RMSprop [??], and Adam [??]) collect information about the variance of the partial derivatives over multiple iterations and use this information to set an appropriate learning rate. Second-order methods, like Hessian-free optimization [??], do not require a learning rate. Instead, a step towards the minimum of a quadratic approximation of the error function (or a semi positive definite approximation thereof) is performed in each iteration. Calculation of one update is much more costly for second-order methods, but it also performs much more progress and navigates valleys of "pathological curvature" more efficiently, which reduces the number of epochs required to train the model. For more details about the aforementioned methods, the reader is referred to the relevant papers.

III. Experiments and Results

A. Data Sets and Pre-processing

The method was evaluation on a large data set from a multi-center MS clinical trial of 195 subjects. The data was acquired from 67 different scanning sites. The data set consists of 377 pairs of FLAIR and T1-weighted MRIs with a resolution of $256 \times 256 \times 60$ voxels and a voxel size of $0.936\,\text{mm} \times 0.936\,\text{mm} \times 3.000\,\text{mm}$. The main preprocessing steps included rigid intra-subject registration, brain extraction, intensity normalization, and background cropping to a $164 \times 206 \times 52$ voxels region of interest.

B. Comparison of Network Architectures

- We used 3 different architectures: 3-layer, 7-layer without shortcuts and 7-layer with shortcuts.
- 7-layer CEN parameters are summarized in Table II.
- Comparison of segmentation performance on the test set with 3 different architectures and lesionTOADS is shown in Table III
- Even the 3-layer model performs much better than lesionTOADS on average.
- Better was confirmed using a one-sided paired t-test. Give the p-value.
- Better than lesionTOADS (p-value = 4.732×10^{-9})
- Better than 3-layer (p-value = 0.002)
- Better with shortcut connections (p-value = 1.566×10^{-11})
- Adding more layers also improves performance. T-test.
- Adding shortcut connections improves performance. t-test.

TABLE I
PARAMETERS OF THE 3-LAYER CEN USED TO EVALUATING DIFFERENT TRAINING METHODS.

Layer type	Kernel Size	#Filters	Image Size
Input	—	—	$164 \times 206 \times 52 \times 2$
Convolutional	$9 \times 9 \times 5 \times 2$	32	$156 \times 198 \times 48 \times 32$
Deconvolutional	$9 \times 9 \times 5 \times 32$	1	$164 \times 206 \times 52 \times 1$

TABLE II
PARAMETERS OF THE 7-LAYER CEN USED TO EVALUATING DIFFERENT TRAINING METHODS.

Layer type	Kernel Size	#Filters	Image Size
Input	—	—	$164 \times 206 \times 52 \times 2$
Convolutional	$9 \times 9 \times 5 \times 2$	32	$156 \times 198 \times 48 \times 32$
Average Pooling	$2 \times 2 \times 2$	—	$78 \times 99 \times 24 \times 32$
Convolutional	$9 \times 10 \times 5 \times 32$	32	$70 \times 90 \times 20 \times 32$
Deconvolutional	$9 \times 10 \times 5 \times 32$	32	$78 \times 99 \times 24 \times 32$
Unpooling	$2 \times 2 \times 2$	—	$156 \times 198 \times 48 \times 32$
Deconvolutional	$9 \times 9 \times 5 \times 32$	1	$164 \times 206 \times 52 \times 1$

C. *Segmentation performance vs. lesion load*

- Analyse the relationship of segmentation performance and lesion load
- Divided the test set into 5 classes with roughly the same number of samples. Classes are (0,1250), (1250,2500), (2500, 3800), (3800,10000), above 10000.
- CEN outperforms lesionTOADS for all lesion load categories, except for very high lesion load.
- For very high lesion load, no difference to lesion TOADS found using t-test.
- TPR, PPV, DSC for different lesion loads in a table.

D. *Discussion*

- Have some sample images and discuss what we can see here.

IV. CONCLUSIONS

We have introduced a new method for the automatic segmentation of MS lesions based on convolutional encoder networks. The joint training of the feature extraction and prediction layers with a novel objective function allows for the automatic learning of features that are tuned for a given combination of image types and a segmentation task with very unbalanced classes. We have evaluated our method on two data sets showing that approximately 100 images are required to train the model without overfitting but even when only a relatively small training set is available, the method still performs comparably to the state-of-the-art algorithms. For future work, we plan to increase the depth of the network, which would allow the learning of a set of hierarchical features. This could further improve segmentation accuracy, but may require larger training sets. We would also like to investigate the use of different objective functions for training based on other measures of segmentation performance.

TABLE III
COMPARISON OF THE SEGMENTATION ACCURACY OF DIFFERENT CEN MODELS WITH LESIONTOADS

Method	TPR [%]	PPV [%]	DSC [%]
3-layer CEN	49.62 ± 20.32	59.87 ± 20.95	49.10 ± 15.78
7-layer CEN*	49.94 ± 19.96	63.5 ± 20.0	51.04 ± 14.71
7-layer CEN**	52.75 ± 20.31	66.58 ± 20.71	54.02 ± 15.24
lesionTOADS	49.83 ± 14.79	39.86 ± 20.95	40.04 ± 14.86

Note: The table shows the mean and standard deviation of the true positive rate (TPR), positive predictive value (PPV), and Dice similarity coefficient (DSC). (* without shortcut connections, ** with shortcut connections)

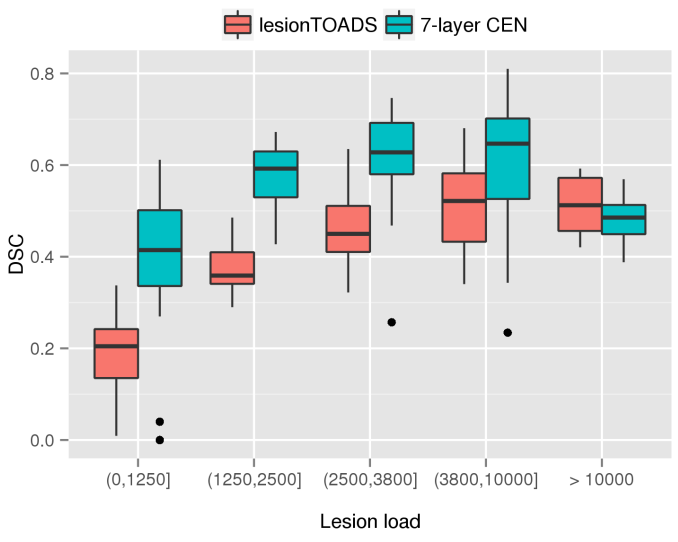

Fig. 2. Comparison of DSC scores of lesionTOADS and a 7-layer CEN for different lesion loads. CEN approach performs consistently well through a wide range of lesion loads.

ACKNOWLEDGEMENTS

This work was supported by Natural Sciences and Engineering Research Council of Canada and the Milan and Maureen Ilich Foundation.

REFERENCES

[1] Tom Brosch, Youngjin Yoo, Lisa Y.W. Tang, David K.B. Li, Anthony Traboulsee, and Roger Tam. Deep convolutional encoder networks for multiple sclerosis lesion segmentation. In *A. Frangi et al. (Eds.): MICCAI 2015, Part III, LNCS, vol. 9351*, pages 3–11. Springer, 2015.
[2] D Ciresan, Alessandro Giusti, and J Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in Neural Information Processing Systems*, pages 1–9, 2012.
[3] Ezequiel Geremia, Bjoern H Menze, Olivier Clatz, Ender Konukoglu, Antonio Criminisi, and Nicholas Ayache. Spatial decision forests for MS lesion segmentation in multi-channel MR images. In *Jian, T., Navab, N., Pluim, J., Viergever, M. (eds.) MICCAI 2010, Part I. LNCS, vol. 6362*, pages 111–118. Springer, Heidelberg, 2010.
[4] Kai Kang and Xiaogang Wang. Fully convolutional neural networks for crowd segmentation. *arXiv preprint arXiv:1411.4464*, 2014.
[5] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the 18th International Conferene on Medical Image Computing and Computer Assisted Interventions (MICCAI 2015)*, page 8, 2015.
[7] Jean-Christophe Souplet, Christine Lebrun, Nicholas Ayache, and Grégoire Malandain. An automatic segmentation of T2-FLAIR multiple sclerosis lesions. In *MIDAS Journal - MICCAI 2008 Workshop*, 2008.

[8] Nick Weiss, Daniel Rueckert, and Anil Rao. Multiple sclerosis lesion segmentation using dictionary learning and sparse coding. In *Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds.) MICCAI 2013, Part I. LNCS 8149*, pages 735–742. Springer, Heidelberg, 2013.

[9] Youngjin Yoo, Tom Brosch, Anthony Traboulsee, David KB Li, and Roger Tam. Deep learning of image features from unlabeled data for multiple sclerosis lesion segmentation. In *Wu, G., Zhang D., Zhou L. (eds.) MLMI 2014, LNCS, vol. 8679*, pages 117–124. Springer, Heidelberg, 2014.