Automated Segmentation of Multiple Sclerosis Lesions Using Multi-dimensional Gated Recurrent Units

Simon Andermatt(✉), Simon Pezold, and Philippe C. Cattin

Department of Biomedical Engineering, University of Basel, Allschwil, Switzerland
simon.andermatt@unibas.ch

Abstract. We analyze the performance of multi-dimensional gated recurrent units on automated lesion segmentation in multiple sclerosis. The segmentation of these pathologic structures is not trivial, since location, shape and size can be arbitrary. Furthermore, the inherent class imbalance of about 1 lesion voxel to 10 000 healthy voxels further exacerbates the correct segmentation. We introduce a new MD-GRU setup, using established techniques from the deep learning community as well as our own adaptations. We evaluate these modifications by comparing them to a standard MD-GRU network. We demonstrate that using data augmentation, selective sampling, residual learning and/or DropConnect on the RNN state can produce better segmentation results. Reaching rank #1 in the ISBI 2015 longitudinal multiple sclerosis lesion segmentation challenge, we show that a setup which combines these techniques can outperform the state of the art in automated lesion segmentation.

Keywords: MD-GRU · MDGRU · Automatic MS lesion segmentation

1 Introduction

Multiple sclerosis (MS) is a frequent disease of the central nervous system, which prevalently occurs in young adults, especially in women. The evaluation of lesions in the brain is part of the clinical diagnostic procedure and is important when evaluating medical trials for new treatments. The manual segmentation of lesions, especially on high-resolution 3d scans, is very time consuming as well as prone to errors due to inter- and intra-rater variability [5]. Recently, recurrent neural networks (RNN) have shown the capability to match the state of the art in brain segmentation. In the brain segmentation benchmark used in [1,10], three of the top six methods are based on RNN. Not only their performance, but also the elegant way of describing data with tied weights do speak for them, since fewer parameters have to be used for the model. We take a closer look at the multi-dimensional gated recurrent unit (MD-GRU) [1] due to its high ranking on the MRBrains challenge. Lesions, as any pathology, are hard to model. We hence treat lesion segmentation independently from anatomy segmentation and consider to reevaluate some findings in [1,10] in the context of lesion segmentation. In the following, we explore different extensions to the MD-GRU with the focus on improvements on lesion segmentation. We investigate some design choices made in the original publication of the MD-GRU [1] and apply emerging deep learning techniques which proved to be effective. We investigate our adaptations on the training data of a publicly available challenge dataset. We then use the best performing combination of our modifications and apply it on the full dataset. Our implementation can be found at https://github.com/zubata88/mdgru.

2 Materials and Methods

2.1 Longitudinal MS Lesion Segmentation Challenge (ISBI 2015)

The longitudinal MS lesion segmentation challenge [2] was held in conjunction with ISBI 2015, but the data and challenge is still available online for further use. The data consists of 5 training patients and 15 test patients, with 4 to 6 screenings each, consisting of an MPRAGE, a T2, a PD and a FLAIR sequence. In all of our experiments, we only incorporate the provided preprocessed MR data and their high-pass filtered counterparts (see Sect. 2.3), as shown in Fig. 1. The remaining screenings for the first patient in the training data are shown in Fig. 2. For the training data, each screening of each patient holds two segmentation masks. Segmentation masks for the test data are not available, but binary predictions can be evaluated automatically on the challenge website.

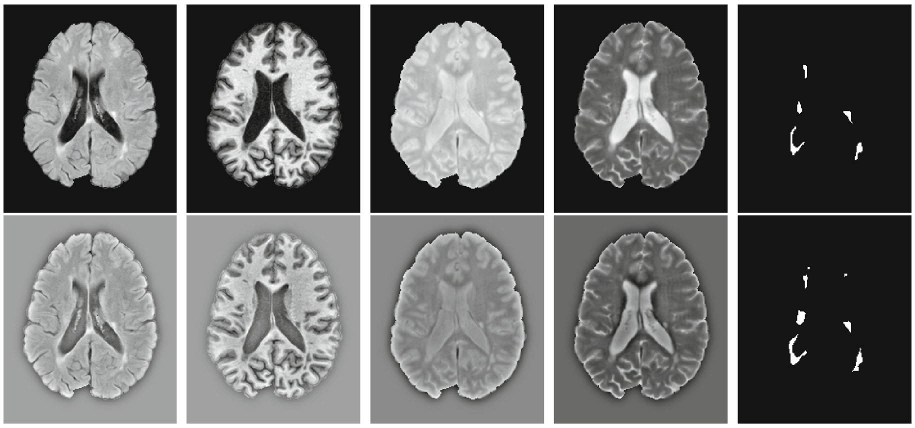

Fig. 1. Slice 90 of the baseline scan of the first training sample. *Top row (left to right):* FLAIR, MPRAGE, PD, T2 scan and label mask of rater 1. *Bottom row:* respective high-pass filtered versions and label mask of rater 2.

2.2 Original MD-GRU Setup

In the following, we define the peculiarities of MD-GRU [1] that are relevant for the evaluation of our modifications to the model. Equation (1) denotes channel j of output H, which consists of the sum of all outputs of the $N \cdot D$ individual convolutional gated recurrent units (C-GRUs) it is made of. Each C-GRU computes the data along either the forward or backward direction n of dimension d:

$$H^j(x) = \sum_{n \in \{1,-1\}} \sum_{d}^{D} h^{j,n,d}. \quad (1)$$

The following are the original C-GRU equations [1]. Index t denotes the timestep and iterates over the slices along d in direction n. Since the computations of each C-GRU are independent, we omit the indices for n, d for better readability in the following:

$$r^j = \sigma\left(\sum_{i}^{I}(x^i * w_r^{i,j}) + \sum_{k}^{J}(h_{t-1}^k * u_r^{k,j}) + b_r^j\right), \quad (2)$$

$$z^j = \sigma\left(\sum_{i}^{I}(x^i * w_z^{i,j}) + \sum_{k}^{J}(h_{t-1}^k * u_z^{k,j}) + b_z^j\right), \quad (3)$$

$$\tilde{h}_t^j = \phi\left(\sum_{i}^{I}(x^i * w^{i,j}) + r^j \odot \sum_{k}^{J}(h_{t-1}^k * u^{k,j}) + b^j\right), \quad (4)$$

$$h_t^j = z^j \odot h_{t-1}^j + (1 - z^j) \odot \tilde{h}_t^j. \quad (5)$$

The indices i and j, k denote the respective input and output channels. Variables u, w and b are trainable weights. We refer to Eqs. (2) and (3) as *reset* and *update gate*, Eq. (4) as *proposal* and Eq. (5) as *output* or *state*.

To analyze the influence of different adjustments, we will use a standard network, similar to the one published in [1]. It consists of 3 layers of MD-GRUs of 16, 32 and 64 channels, which are connected with voxelwise fully connected layers with biases consisting of 25 and 45 channels followed by a tanh activation function. The last MD-GRU is connected to a voxelwise fully connected layer of c channels, one for each class. Finally, a softmax layer is applied and the network is trained minimizing the negative log likelihood. Equation (6) summarizes the setup, where superscript numbers denote the number of channels at each layer and subscripts enumerate the independent layers of the same type:

$$h = \text{softMax}(\underset{3}{\overset{c}{\text{conv}}}111(\overset{64}{H_3}(\tanh(\underset{2}{\overset{45}{\text{conv}}}111(\overset{32}{H_2}(\tanh(\underset{1}{\overset{25}{\text{conv}}}111(\overset{16}{H_1}(x))))))))). \quad (6)$$

2.3 Evaluated Design Choices

The MD-GRU showed promising results with a relatively simple architecture. In the following we motivate and evaluate modifications to the original architecture.

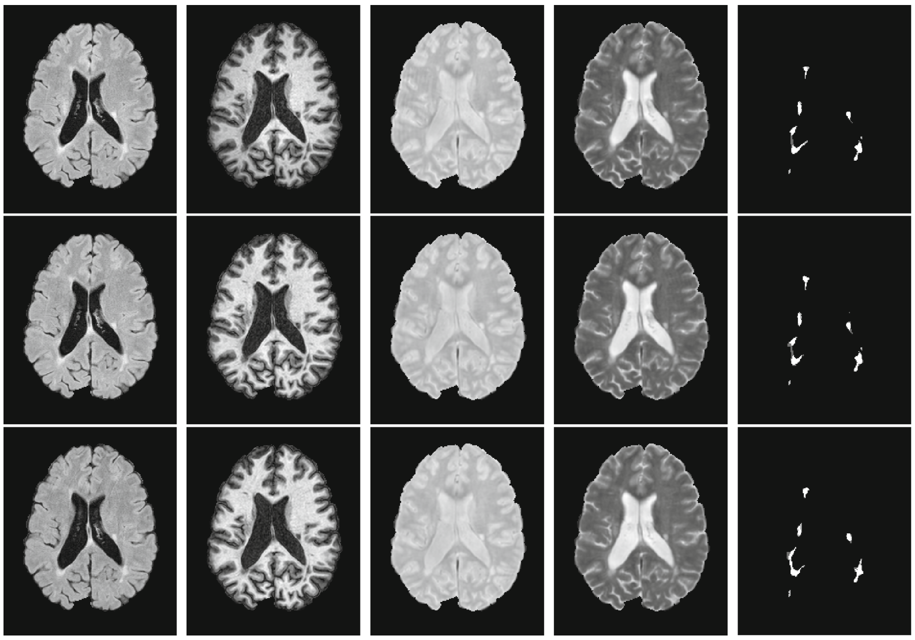

Fig. 2. Slice 90 of each followup scan of first training sample. *From left to right:* FLAIR, MPRAGE, PD, T2 scan and combined rater mask.

High-Pass Filtering. A high-pass filter was applied to the images by subtracting a Gaussian filtered version of the image volumes from the original volumes. Especially in situations with almost piecewise constant functions, such as MR images of the brain, this preprocessing step can help "announcing" a change of tissue before it actually happens, as can be seen for instance around the masked brains in Fig. 1. In Fig. 3, we inspect the voxel values along one anteroposterior line through the volume. In our experiments, we investigate, how much high-pass filtered data can help detract the influence of low frequency intensity changes in the data.

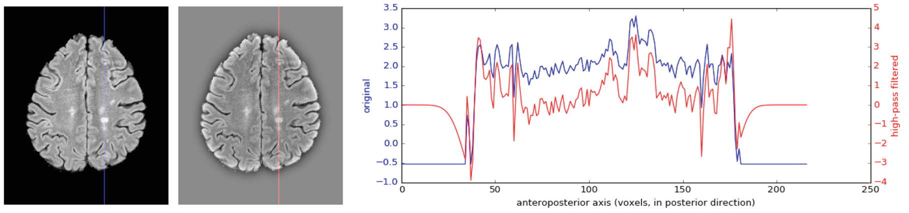

Fig. 3. Impact of high-pass filtering on the fourth screening of the sixth training sample. *Left:* Slice 110 of original and high-pass filtered FLAIR scan. *Right:* Plot of marked red and blue lines on the left for both images after normalization. (Color figure online)

Reset Gate Location. Compared to the original formulation of the GRU [3], the C-GRU applies the reset gate at a slightly different position, as depicted in Fig. 4a. In the GRU, the reset gate r is directly multiplied to the previous output h_{t-1}:

$$\tilde{h}_t^j = \phi([Wx]^j + [U(r \odot h_{t-1})]^j) \tag{7}$$

In the C-GRU however, it is multiplied to the result of the convolution of the previous output h_{t-1} with u, as shown in Eq. (4).

The provided motive for this decision is, that r is the result of convolutions and already contains information of its neighbors. This effectively means that the reset gate of channel j only directly affects the proposal of channel j instead of all proposals. We evaluate this decision by comparing to a modified C-GRU, which more closely follows the original formulation:

$$\hat{\tilde{h}}_t^j = \phi\left(\sum_i^I (x^i * w^{i,j}) + \sum_k^J ((r^{k,j} \odot h_{t-1}^k) * u^{k,j}) + b^j\right). \tag{8}$$

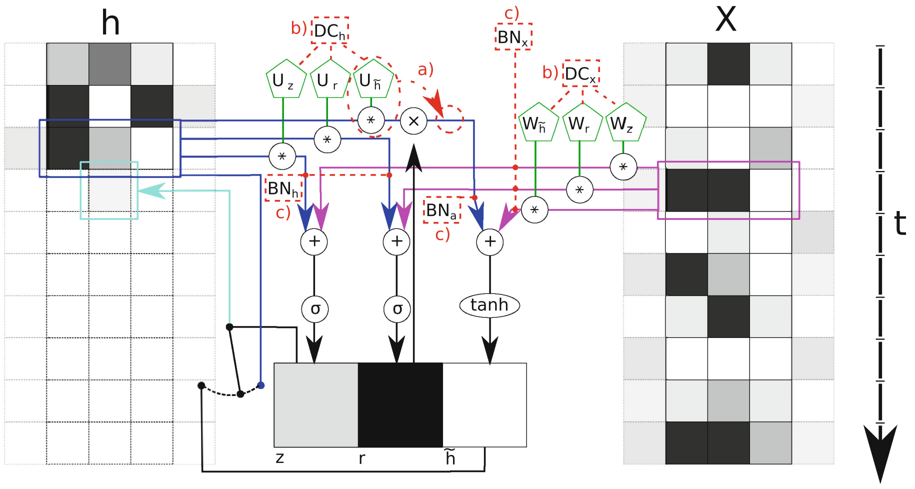

Fig. 4. Schematic and computational graph of a C-GRU with one-dimensional filters. *The proposed changes are marked with dashed red lines:* (a) the order of the reset gate application, (b) DropConnect at state and input weights and (c) batch normalization at input, gate states and proposal activation states. (Color figure online)

Contribution Weights for Individual C-GRU Outputs. In the original MD-GRU formulation, the individual C-GRU outputs are simply summed to gather the result H. As already implemented in the first bidirectional RNN [9], the states for each direction could be weighted independently, resulting in a

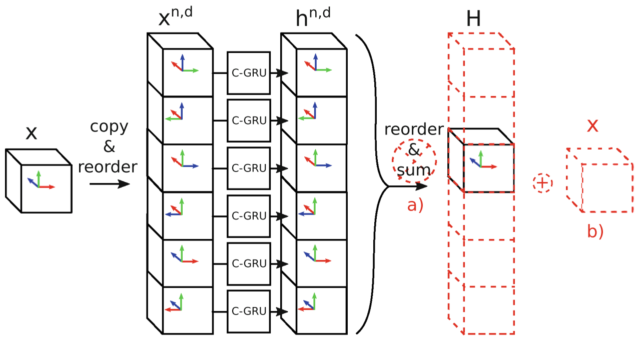

Fig. 5. Composition of an MD-GRU. *The proposed changes are marked with dashed red lines:* (a) leaving out the sum of the individual directional states or (b) adding residual learning at the MD-GRU level. (Color figure online)

more complex model. We investigate the potential benefit of concatenating the C-GRU outputs, thereby in our case of 3d volumes increasing the number of output channels sixfold. Figure 5a shows this on the example of MD-GRUs that handle volumetric data.

DropConnect. Instead of dropout, a similar method called DropConnect (DC) [11] is used at a constant rate of 0.5, which drops weights instead of outputs. In the original formulation [1], we decided to implement DC on the input weights in MD-GRU and to use a fixed drop rate of 0.5. Dropout has been reported to not work well on MD-LSTM [10] and applying it on the state in RNN has been advised against [12]. We analyze the effect of applying DC on input, state or both using different drop rates (Fig. 4b).

2.4 Techniques to Improve Accuracy and Shorten Training Time

Batch/Instance Normalization. The first technique we investigate is batch normalization (BN) [7]. BN allows for higher learning rates and faster convergence, thereby drastically reducing the training time. By normalizing the input to activations, the so-called covariate shift is reduced. This enables a layer further down the network to learn more independently from the layers before it. We build on the results on BN in one-dimensional RNN in [4] and define BN as

$$BN(x, \gamma) = \gamma \frac{x - \hat{\mu}}{\sqrt{\hat{\sigma} + \epsilon}} + \beta, \tag{9}$$

where we set β to 0 at any place, due to the biases that are already in place [4]. Due to our inherent mini-batch of one (we can only process one subvolume at a time per training iteration), we calculate the statistics for mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ on the whole data per channel. Since the data in the subvolumes is heavily correlated, we calculate our training statistics for each training iteration k from the m most recent training samples with $\hat{\mu}^{(k)} = \frac{1}{m}\sum_{q=k-m+1}^{k}\mu^{(q)}$ and $\hat{\sigma}^{(k)} = \frac{1}{m}\sum_{q=k-m+1}^{k}\sigma^{(q)}$. We keep a separate exponential moving average for both mean and standard deviation over all training samples to be used for testing. We apply the following BN:

$$r^j = \sigma(BN_x(\sum_i^I(x^i * w_r^{i,j}), \gamma_x) + BN_h(\sum_k^J(h_{t-1}^k * u_r^{k,j}), \gamma_h) + b_r^j), \quad (10)$$

$$z^j = \sigma(BN_x(\sum_i^I(x^i * w_z^{i,j}), \gamma_x) + BN_h(\sum_k^J(h_{t-1}^k * u_z^{k,j}), \gamma_h) + b_z^j), \quad (11)$$

$$\tilde{h}_t^j = \phi(BN_x(\sum_i^I(x^i * w^{i,j}), \gamma_x) + BN_a(r^j \odot \sum_k^J(h_{t-1}^k * u^{k,j}), \gamma_a) + b^j), \quad (12)$$

where we keep individual statistics for the input convolutions, the gate state convolutions and the state convolution of the proposal. Figure 4c shows the different locations we apply BN at.

Residual Learning. Using skip connections allowed ResNet [6] to ascend on top of a number of ILSVRC & COCO 2015 competitions, as it reportedly allows for deeper networks and faster convergence. We introduce skip connections linking input and output of each MD-GRU (Fig. 5b), allowing the network to choose between learning a residual or ignoring the previous input. We evaluate the following adjustment in between individual MD-GRU layers:

$$H_{res}(x) = \overset{c}{\text{conv111}}(x) + \overset{c}{H}(x), \quad (13)$$

where c denotes the number of output channels of H and the additional convolution increases the input channels to c. We refrain from applying additional

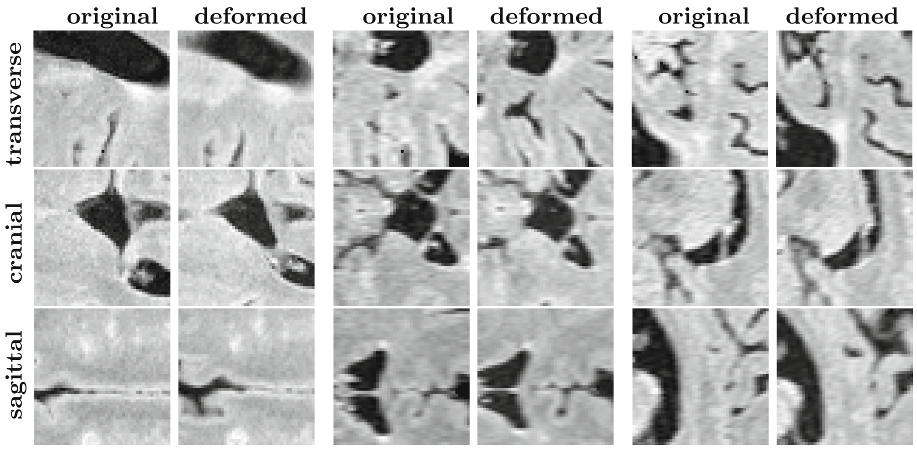

Fig. 6. Examples of random deformations performed on 48^3 subvolumes. *Rows:* transverse, coronal and sagittal planes of the original *(left)* and deformed *(right)* sample. *Columns:* individual random samples.

residual learning inside the MD-GRU, as it has been shown that residual networks with shared weights can be reformulated as plain RNN [8].

Data Augmentation. Data augmentation can have beneficial effects on networks which are trained with little data [1]. On a low resolution grid of spacing f voxels, we draw three values from a normal distribution $\mathcal{N}(0, 5)$. Using cubic interpolation, we create a smooth deformation field, which we apply on the voxelwise sampling of the subvolumes used for training (Fig. 6).

3 Experiments and Results

We train each model for 3000 iterations, which will not produce competitive results, but should give an indicator of how much an adjustment affects the method. We only included the first 4 baseline scans during training and evaluate all combinations on the baseline scan of the 5th training patient. Using only 4 samples, we remove any redundancy in the data and decrease the search space, allowing for faster convergence. For all experiments, we train on 48^3 random samples and create a full volume during the test phase by stitching patches of 32^3 with a padding of 8 together .

Data Sampling Techniques and MD-GRU Design Evaluation. We considered the following different data sampling techniques. To quantify the impact of the high-pass filtered data, we trained networks with only the original data, only the high-pass filtered data and both. We further analyzed the impact of data augmentation through random deformation, varying the size f of the deformations. We evaluate the impact of forcing every second random training sample to contain lesions (selective sampling). We also evaluate potentially not summing the individual C-GRU results and the misplacement of the reset gate as defined in Eq. (12). The respective results are listed in Table 1A.

Table 1. Summary of the different combinations which were trained on the first four and evaluated on the fifth training sample of the ISBI challenge data. Dice coefficients are provided in percent and bold face denotes results that were better than those of the baseline architecture (A, top row). The two provided masks we compare ourselves to are denoted as M1 and M2.

A	Dice	
	M1	M2
Baseline [1]	38.18	37.70
Only original	0.00	0.00
Only filtered	28.30	25.65
Def(f = 24)	**43.74**	36.94
Def(f = 48)	**48.43**	**49.12**
Selective samples	**44.95**	**40.70**
No MD-GRU sum	17.79	15.39
Misplaced r	24.75	21.20

B	Dice	
	M1	M2
DC(0.5,h,x)	29.39	26.35
DC(0.5,h)	**44.06**	**41.55**
No DC	27.85	23.54
DC(0.75,x)	21.27	17.24
DC(0.75,x,h)	15.79	13.63
DC(0.75,h)	23.27	20.13
DC(0.875,x)	21.82	19.33
DC(0.875,h)	19.87	17.29
RL	**41.51**	35.71

C	Dice			
	M1		M2	
	m=16	m=1	m=16	m=1
BNx	31.77	30.58	26.60	24.94
BNx+DC(x)	37.53	20.92	32.19	18.45
BNh	31.78	29.42	30.03	31.98
BNh+DC(x)	28.95	30.13	26.03	27.74
BNa	3.05	3.04	2.50	2.49
BNa+DC(x)	9.70	10.20	8.59	9.02

Table 2. Mean and standard deviation of the crossvalidation with the Dice coefficient in percent, the Hausdorff distance (HD) as well as the average volume distance (AVD) with best scores and lowest standard deviations in bold face.

	Dice		HD		AVD	
	M1	M2	M1	M2	M1	M2
Baseline [1]	20.03 ± 14.13	19.86 ± 13.17	39.82 ± 7.62	39.15 ± 6.69	7.34 ± 7.89	6.79 ± 6.52
DC(h)	33.47 ± 8.57	32.93 ± 8.82	35.84 ± 5.87	$36.64 \pm \mathbf{4.40}$	5.78 ± 6.16	5.64 ± 5.58
Residual learning	35.95 ± 13.78	35.10 ± 10.19	40.61 ± 9.45	36.29 ± 6.65	7.27 ± 6.90	6.61 ± 5.48
Selective sampling	$44.10 \pm \mathbf{4.55}$	$40.54 \pm \mathbf{4.64}$	$41.32 \pm \mathbf{3.15}$	40.31 ± 4.57	5.07 ± 5.42	4.87 ± 4.75
Def ($f = 48$)	38.29 ± 21.51	34.70 ± 19.82	37.27 ± 8.89	38.57 ± 8.09	2.91 ± 3.35	2.64 ± 2.43
All of the above	$\mathbf{62.85} \pm 15.31$	$\mathbf{55.24} \pm 13.66$	$\mathbf{32.60} \pm 8.58$	$\mathbf{29.82} \pm 4.72$	$\mathbf{1.83} \pm \mathbf{1.22}$	$\mathbf{2.18} \pm \mathbf{1.73}$

DropConnect, Batch Normalization and Residual Learning. Since both DC and BN act as regularization, we evaluated them both jointly and individually. In Table 1B, we list Dice coefficients obtained using different DC settings on input x and/or state h with the designated drop rate. At the bottom, we list the result obtained by applying residual learning (RL) in between MD-GRU layers. In Table 1C, the Dice coefficients resulting from different BNs both with and without simultaneous DC with a drop rate of 0.5 on the input are shown.

Putting it All Together. In a last experiment, we performed leave-one-out crossvalidation with the modifications which performed better than the standard network (Table 1) and the sum of those modifications. Using the Dice as performance measure, the selected techniques were random deformation with a grid spacing of 48, selective sampling, residual learning and DC on the state instead of the input. The crossvalidation results can be found in Table 2.

3.1 Improved Network

So far, we restricted ourselves to a subset of the data and only 3 000 training iterations. For the final evaluation on the challenge website, we considered the complete training data, instead of just using the first scan of each patient. We decided to use a combination of data augmentation through random deformation ($f = 75$) and subvolumes of 80^3 together with DC on the state h, residual learning and selective sampling. We merged rater masks by creating 4 classes, one for each label combination during training and assumed a lesion voxel during inference, when its probability for background was below 0.5. We trained the network for 10 000 iterations and managed to achieve first place in the ISBI challenge. Figure 7 shows slice 80 of the best and worst segmentation, judging from the challenge score computed on both raters. The first five entries of the challenge are listed in Table 3 with the mean of each metric that contributes to the challenge score. The fifth entry was created using the MD-GRU as described in its original publication [1] and 40 000 training iterations. Unfortunately, none of the other competing entries have been published yet, which makes a comparison of the methods impossible.

Table 3. The five best scoring methods of the longitudinal MS lesion segmentation challenge with challenge score, volume correlation (VC), Dice coefficient, positive predictive value (PPV), lesion false positive rate (LFPR), lesion true positive rate (LTPR). Dice, PPV, LFPR and LTPR are denoted in percent and best values out of five are printed in bold. In brackets we denote the relative weight of each metric on the final score.

	Score	VC (1/4)	Dice (1/8)	PPV (1/8)	LFPR (1/4)	LTPR (1/4)
asmsl (proposed)	**92.076**	0.862	62.98	**84.46**	20.13	48.71
nic_vicorob_test	91.440	0.840	64.29	79.25	15.46	38.72
VIC_TF_FULL	91.331	0.866	63.05	78.67	**15.29**	36.40
MIPLAB_v3	91.267	0.823	62.74	79.97	23.17	45.40
miac_results [1]	91.011	**0.867**	**66.78**	74.05	40.73	**58.29**

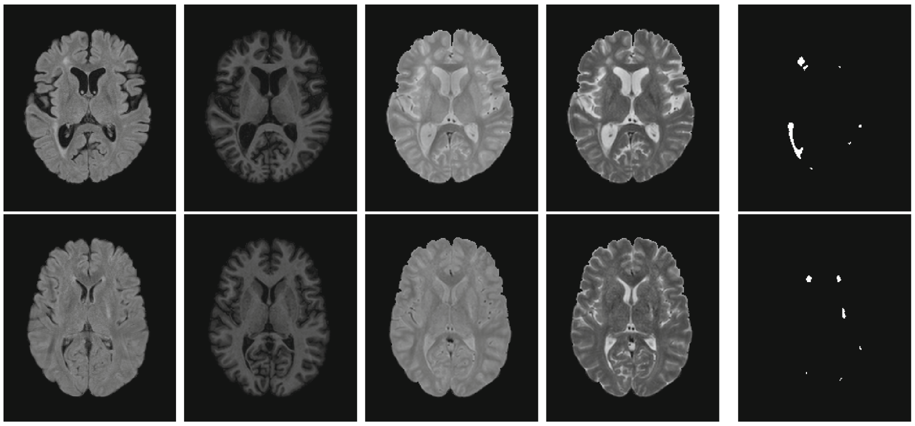

Fig. 7. Main challenge results. *Columns, left to right:* FLAIR, MPRAGE, PD and T2 scan together with the computed segmentation at Slice 80. *Rows:* best *(top)* and worst *(bottom)* segmentation (baseline scan of patient 4 and 7 respectively), with respect to the challenge score computed on both raters' segmentations.

4 Discussion

We encountered expected as well as odd behavior in our exploratory study. Contrary to what has been advised for in the literature [12], dropping information from the state weights results in better regularization, as the Dice coefficients in Table 1B indicate. This behavior could be due to the fact that we only ignore part of the previous state per iteration and channel. Interestingly though, a combination of DC on both input and state produces worse results, even with a reduced drop rate. As dropout tends to prolong training, further experiments with longer training times might shed light on this effect. Another surprising result is the inability of BN to surpass baseline Dice scores in our preliminary tests in Table 1, in all variations we tested. Due to the correlation in our mini-batch of one and the varying weights in the case of the running average, the assumption does not hold that the statistics of our mini-batch are similar to the global statistics. Residual learning between MD-GRU layers seems to contribute to the overall improvement. Surprisingly, neither concatenating the C-GRU nor placing the reset gate as in the original GRU did result in an improved Dice.

The high pass filtering as preprocessing step proved to be fruitful, especially in the setting where we only trained for 3000 iterations, where leaving it out resulted in no segmentation at all. Using only original data, a visible tendency towards lesions could be found, but with probabilities well below 0.5. The main reason why this step is so important can be seen in Fig. 3, where values of the filtered image lie mostly around zero and in the original scan around two. All the weights of our network are initialized to handle data from a standard normal distribution. Inside the brain, filtering the original image would result in sums far away from zero. Using a hyperbolic tangent or sigmoid function on such a result will return a value close to 1 and hence a very flat gradient, which will not be able to help adjust the weights to correct for this in a fast manner.

Selective sampling and random deformation succeeded to be the most important improvements, which is easily explainable with the huge class imbalance present in our data and the low amount of training data. As the crossvalidation shows, all of the selected techniques resulted in overall better scores except for the HD in selective sampling, which is likely due to a higher probability of producing outliers when oversampling the lesion class.

By achieving rank 1 in the actual challenge, we show that our proposed method is at the state of the art. Unfortunately, none of the listed results in Table 3 have been published yet, as already mentioned. The highest Dice score in the top 5 was achieved using exactly the same MD-GRU network as in its original publication [1] and training it for 40 000 iterations. Since we only trained our network for 10 000 iterations and showed superior performance over the original setup in our evaluation, we believe that an even higher score is possible by training for a longer time.

MD-GRUs allow for any number of dimensions in the data, hence it would also be possible to use the actual 4d data from the challenge, with the new dimension being the screening number. Using 4d data could pose a number of problems though, for instance the reduced spatial resolution that can be fed to the network per training iteration due to the memory constraints and the low number of screenings that are available. Further research might be necessary to determine, if a suitable trade-off between spatial resolution and temporal information exists.

In conclusion, the following four modifications can drastically improve the accuracy of lesion segmentation in terms of Dice, HD and AVD with the

MD-GRU: Selective sampling speeds up training drastically, since most of the data can be labeled safely as background. DC on the state does a better job in regularization than on the input. Random deformation prevents the model from overfitting. Finally, residual learning in between MD-GRUs might shorten training time by simplifying the estimation task.

References

1. Andermatt, S., Pezold, S., Cattin, P.: Multi-dimensional gated recurrent units for the segmentation of biomedical 3D-data. In: Carneiro, G., et al. (eds.) LABELS/DLMIA-2016. LNCS, vol. 10008, pp. 142–151. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46976-8_15
2. Carass, A., Roy, S., Jog, A., Cuzzocreo, J.L., Magrath, E., Gherman, A., Button, J., Nguyen, J., Prados, F., Sudre, C.H., Jorge Cardoso, M., Cawley, N., Ciccarelli, O., Wheeler-Kingshott, C.A.M., Ourselin, S., Catanese, L., Deshpande, H., Maurel, P., Commowick, O., Barillot, C., Tomas-Fernandez, X., Warfield, S.K., Vaidya, S., Chunduru, A., Muthuganapathy, R., Krishnamurthi, G., Jesson, A., Arbel, T., Maier, O., Handels, H., Iheme, L.O., Unay, D., Jain, S., Sima, D.M., Smeets, D., Ghafoorian, M., Platel, B., Birenbaum, A., Greenspan, H., Bazin, P.L., Calabresi, P.A., Crainiceanu, C.M., Ellingsen, L.M., Reich, D.S., Prince, J.L., Pham, D.L.: Longitudinal multiple sclerosis lesion segmentation: resource and challenge. NeuroImage **148**, 77–102 (2017)
3. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 [cs, stat], June 2014
4. Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., Courville, A.: Recurrent batch normalization. arXiv:1603.09025 [cs], March 2016
5. Filippi, M., Horsfield, M.A., Bressi, S., Martinelli, V., Baratti, C., Reganati, P., Campi, A., Miller, D.H., Comi, G.: Intra- and inter-observer agreement of brain MRI lesion volume measurements in multiple sclerosis. Brain **118**(6), 1593–1600 (1995)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv:1512.03385 [cs], December 2015
7. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 [cs], February 2015
8. Liao, Q., Poggio, T.: Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv:1604.03640 [cs], April 2016
9. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
10. Stollenga, M.F., Byeon, W., Liwicki, M., Schmidhuber, J.: Parallel multi-dimensional LSTM, with application to fast biomedical volumetric image segmentation. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 2998–3006. Curran Associates, Inc. (2015)
11. Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML-2013), pp. 1058–1066 (2013)
12. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv:1409.2329 [cs], September 2014