Multi-dimensional Gated Recurrent Units for the Segmentation of White Matter Hyperintensites

Simon Andermatt, Simon Pezold, and Philippe Cattin

Department of Biomedical Engineering, University of Basel, Switzerland

Multi-dimensional Gated Recurrent Units Multi-dimensional Gated Recurrent Units (MD-GRU) [2] are a multi-dimensional generalization of gated recurrent units [3]. Simply put, MD-GRU handle multi-dimensional data similar to bi-directional recurrent neural networks (biRNN) for one-dimensional data. A biRNN travels along the time dimension in forward and backward direction through the signal and summarizes its state to gather an output at the respective timesteps of both directions. In MD-GRU, all spatial dimensions are considered as time dimensions. For each dimension, the data are processed in forward and backward fashion along that dimension, and the sum of the resulting intermediate outputs constitutes the final output of the MD-GRU.

Network We model and train our network of MD-GRU as published in [2], if not stated otherwise. To summarize the architecture, we use three consecutive MD-GRU layers of 16, 32, and 64 channels, which are connected with voxel-wise fully connected layers of 25 and 45 channels, both followed by a hyperbolic tangent activation function. Finally, a voxel-wise fully connected layer of three channels is appended and fed into a softmax activation function, resulting in three probabilities for each voxel, one for each class (background, WMH, other pathology).

Data and Preprocessing We use both the preprocessed FLAIR and T1 scans and create two additional high-pass filtered versions. We accomplish this by first filtering each scan with a 3d Gauss filter of $\sigma_{x,y} = 5$ and $\sigma_z = 1.6$ voxels, where x and y refer to in-plane dimensions, and subtract the results from the originals. Each of the 4 volumes is then normalized to zero mean and unit variance. No further preprocessing is applied.

Optimization The network is trained using Tensorflow 1.1 [1] on a machine equipped with an NVIDIA GeForce Titan X. We apply Gaussian Dropout [4] of 0.5 on both state and input filters of the MD-GRU layers. We selectively sample such that each second training sample is guaranteed to contain the WMH class. Throughout training, we sample randomly deformed subvolumes using a smooth deformation field. We acquire this deformation field by sampling random

deformation vectors on a low resolution grid with a spacing of 75, 75, and 15 voxels. We then scale this grid using bicubic interpolation to full resolution and use the resulting deformation vectors to deform our samples. Starting from training iteration 40 000, we additionally apply random rotation and scaling to the volumes, uniformly drawn from $[-10°, +10°]$ and $[0.8, 1.2]$ respectively. We decided to stop training at iteration 112 500, indicated by a small validation set of three volumes (numbers 29, 58, and 110) which we excluded from training .

Evaluation Due to memory constraints, we need to divide the volume of size $S_x \times S_y \times S_z$ into chunks of size $s_x \times s_y \times s_z$. We let patches overlap along dimension i by $p_i = \frac{s_i}{2}$ and use linear blending to combine them. For each dimension, we additionally pad the full volume by p_i. We automatically determine the best patch size with respect to available memory for each sample, given that volumes of roughly $2 \cdot 10^6$ voxels fit into memory: If $S_z + 2 \cdot 8 < 100$, we set $s_z = S_z + 2 \cdot 8$ and $p_z = 8$. Otherwise, we set $s_z = 100$ and $p_z = 50$. The remaining two sizes are approximated using $s_i = \sqrt{\frac{2 \cdot 10^6}{s_z}}$. For each dimension i that needs blending, we use the smallest s_i that still produces the same number of patches per dimension. From the 3 predicted classes, we ignore the third class, since its also ignored in the evaluation and determine the label from the maximum probability of either background or WMH class.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), http://tensorflow.org/, software available from tensorflow.org
2. Andermatt, S., Pezold, S., Cattin, P.: Multi-dimensional Gated Recurrent Units for the Segmentation of Biomedical 3D-Data. In: International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis. pp. 142–151. Springer (2016)
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] (Jun 2014)
4. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)