Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation

Marijn F. Stollenga[*123], Wonmin Byeon[*1245], Marcus Liwicki[4], and Juergen Schmidhuber[123]

[*]Shared first authors, both Authors contribruted equally to this work. Corresponding authors:
marijn@idsia.ch, wonmin.byeon@dfki.de
[1]Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (The Swiss AI Lab IDSIA)
[2]Scuola universitaria professionale della Svizzera italiana (SUPSI), Switzerland
[3]Universitá della Svizzera italiana (USI), Switzerland
[4]University of Kaiserslautern, Germany
[5]German Research Center for Artificial Intelligence (DFKI), Germany

Abstract

Convolutional Neural Networks (CNNs) can be shifted across 2D images or 3D videos to segment them. They have a fixed input size and typically perceive only small local contexts of the pixels to be classified as foreground or background. In contrast, Multi-Dimensional Recurrent NNs (MD-RNNs) can perceive the entire spatio-temporal context of each pixel in a few sweeps through all pixels, especially when the RNN is a Long Short-Term Memory (LSTM). Despite these theoretical advantages, however, unlike CNNs, previous MD-LSTM variants were hard to parallelise on GPUs. Here we re-arrange the traditional cuboid order of computations in MD-LSTM in pyramidal fashion. The resulting PyraMiD-LSTM is easy to parallelise, especially for 3D data such as stacks of brain slice images. PyraMiD-LSTM achieved best known pixel-wise brain image segmentation results on MRBrainS13 (and competitive results on EM-ISBI12).

1 Introduction

Long Short-Term Memory (LSTM) networks [1, 2] are recurrent neural networks (RNNs) initially designed for sequence processing. They achieved state-of-the-art results on challenging tasks such as handwriting recognition [3], large vocabulary speech recognition [4] and machine translation [5]. Their architecture contains gates to store and read out information from linear units called error carousels that retain information over long time intervals, which is hard for traditional RNNs.

Multi-Dimensional LSTM networks (MD-LSTM [6]) connect hidden LSTM units in grid-like fashion[1]. Two dimensional MD-LSTM is applicable to image segmentation [6, 7, 8] where each pixel is assigned to a class such as background or foreground. Each LSTM unit sees a pixel and receives input from predecessing LSTM units, thus recursively gathering information about all other pixels in the image.

There are many biomedical 3D volumetric data sources, such as computed tomography (CT), magnetic resonance (MR), and electron microscopy (EM). Most previous approaches process each 2D slice separately, using image segmentation algorithms such as snakes [9], random forests [10] and Convolutional Neural Networks [11]. 3D-LSTM, however, can process the full context of each pixel in such a volume through 8 sweeps over all pixels by 8 different LSTMs, each sweep in the general direction of one of the 8 directed volume diagonals.

[1]For example, in two dimensions this yields 4 directions; up, down, left and right.

Due to the sequential nature of RNNs, however, MD-LSTM parallelisation was difficult, especially for volumetric data. The novel Pyramidal Multi-Dimensional LSTM (PyraMiD-LSTM) networks introduced in this paper use a rather different topology and update strategy. They are easier to parallelise, need fewer computations overall, and scale well on GPU architectures.

PyraMiD-LSTM is applied to two challenging tasks involving segmentation of biological volumetric images. Competitive results are achieved on EM-ISBI12 [12]; best known results are achieved on MRBrainS13 [13].

2 Method

We will first describe standard one-dimensional LSTM [2]. Then we introduce the MD-LSTM and topology changes to construct the PyraMiD-LSTM, which is formally described and discussed.

A one-dimensional LSTM unit consists of an input gate (i), forget gate[2] (f), output gate (o), and memory cell (c) which control what should be remembered or forgotten over potentially long periods of time. The input x and all gates and activations are real-valued vectors: $x, i, f, \tilde{c}, c, o, h \in \mathbb{R}^T$, where T is the length of the input. The gates and activations at discrete time t (t=1,2,...) are computed as follows:

$$i_t = \sigma(x_t \cdot \theta_{xi} + h_{t\text{-}1} \cdot \theta_{hi} + \theta_{i_{\text{bias}}}), \tag{1}$$

$$f_t = \sigma(x_t \cdot \theta_{xf} + h_{t\text{-}1} \cdot \theta_{hf} + \theta_{f_{\text{bias}}}), \tag{2}$$

$$\tilde{c}_t = \tanh(x_t \cdot \theta_{x\tilde{c}} + h_{t\text{-}1} \cdot \theta_{h\tilde{c}} + \theta_{\tilde{c}_{\text{bias}}}), \tag{3}$$

$$c_t = \tilde{c}_t \odot i_t + c_{t\text{-}1} \odot f_t, \tag{4}$$

$$o_t = \sigma(x_t \cdot \theta_{xo} + h_{t\text{-}1} \cdot \theta_{ho} + \theta_{o_{\text{bias}}}), \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where $(\cdot)$ is a (matrix) multiplication, $(\odot)$ an element-wise multiplication, and θ denotes the weights. $\tilde{c}$ is the input to the 'cell' c, which is gated by the input gate, and h is the output. The non-linear functions σ and $\tanh$ are applied element-wise, where $\sigma(x) = \frac{1}{1+e^{-x}}$. Equations (1, 2) determine gate activations, Equation (3) cell inputs, Equation (4) the new cell states (here 'memories' are stored or forgotten), Equation (5) output gate activations which appear in Equation (6), the final output.

2.1 Pyramidal Connection Topology

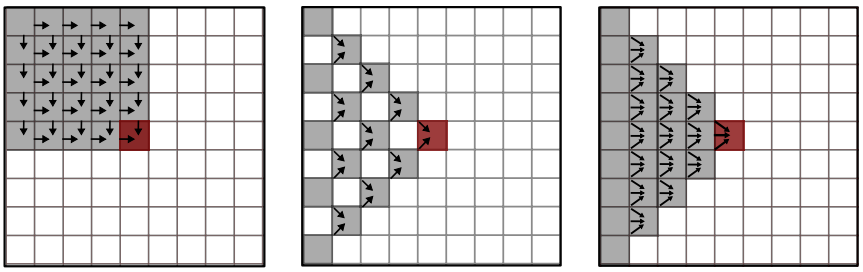


Figure 1: The standard MD-LSTM topology (a) evaluates the context of each pixel recursively from neighbouring pixel contexts along the axes, that is, pixels on a simplex can be processed in parallel. Turning this order by 45° (b) causes the simplex to become a plane (a column vector in the 2D case here). The resulting gaps are filled by adding extra connections, to process more than 2 elements of the context (c).

The multi-dimensional LSTM (MD-LSTM; [6]) aligns LSTM-units in a grid and connects them over the axis. Multiple grids are needed to process information from all directions. A 2D-LSTM adds the pixel-wise outputs of 4 LSTMs: one scanning the image pixel by pixel from north-west to south-east, one from north-east to south-west, one from south-west to north-east, and one from south-east to north-west. Figure 1–a shows one of these directions.

[2] Although the forget gate output is inverted and actually 'remembers' when it is on, and forgets when it is off, the traditional nomenclature is kept.

However, a small change in connections can greatly facilitate parallelisations: If the connections are rotated by $45°$, all inputs to all units come from either left, right, up, or down (left in case of Figure 1–b), and all elements of a row in the grid row can be computed independently. However, this introduces context gaps as in Figure 1–b. By adding an extra input, these gaps are filled as in Figure 1–c. Extending this approach in 3 dimensions results in a Pyramidal Connection Topology, meaning the context of a pixel is formed by a pyramid in each direction.

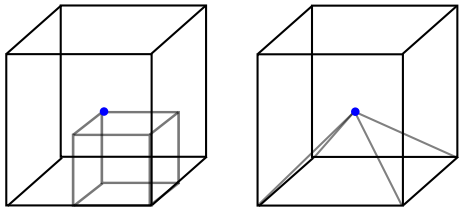


Figure 2: On the **left** we see the context scanned so far by one of the 8 LSTMs of a 3D-LSTM: a cube. In general, given d dimensions, 2^d LSTMs are needed. On the **right** we see the context scanned so far by one of the 6 LSTMs of a 3D-PyraMiD-LSTM: a pyramid. In general, $2 \times d$ LSTMs are needed.

One of the striking differences between PyraMiD-LSTM and MD-LSTM is the shape of the scanned contexts. Each LSTM of an MD-LSTM scans rectangle-like contexts in 2D or cuboids in 3D. Each LSTM of a PyraMiD-LSTM scans triangles in 2D and pyramids in 3D (see Figure 2). An MD-LSTM needs 8 LSTMs to scan a volume, while a PyraMiD-LSTM needs only 6, since it takes 8 cubes or 6 pyramids to fill a volume. Given dimension d, the number of LSTMs grows as 2^d for an MD-LSTM (*exponentially*) and $2 \times d$ for a PyraMiD-LSTM (*linearly*).

A similar connection strategy has been previously used to speed up non-Euclidian distance computations on surfaces [14]. There are however important differences:

- We can exploit efficient GPU-based CUDA convolution operations, but in a way unlike what is done in CNNs, as will be explained below.
- As a result of these operations, input filters that are bigger than the necessary 3×3 filters arise naturally, creating overlapping contexts. Such redundancy turns out to be beneficial and is used in our experiments.
- We apply several layers of complex processing with multi-channelled outputs and several state-variables for each pixel, instead of having a single value per pixel as in distance computations.
- Our application is focused on volumetric data.

2.2 PyraMiD-LSTM

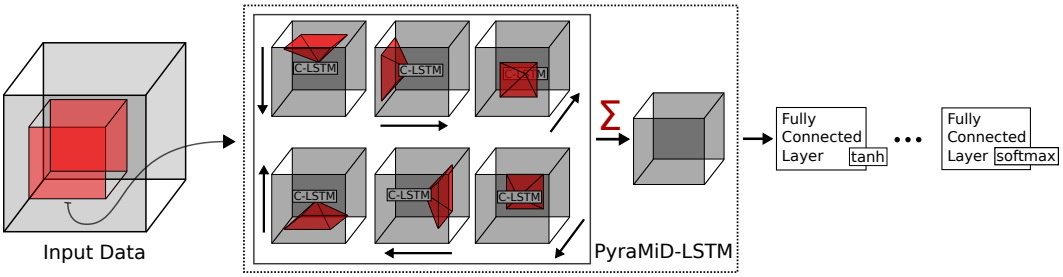


Figure 3: PyraMiD-LSTM network architecture. Randomly rotated and flipped inputs are sampled from random locations, then fed to six C-LSTMs over three axes. The outputs from all C-LSTMs are combined and sent to the fully-connected layer. $tanh$ is used as a squashing function in the hidden layer. Several PyraMiD-LSTM layers can be applied. The last layer is fully-connected and uses a softmax function to compute probabilities for each class for each pixel.

Here we explain the PyraMiD-LSTM network architecture for 3D volumes (see Figure 3). The working horses are six convolutional LSTMs (C-LSTM) layers, one for each direction to create the full context of each pixel. Note that each of these C-LSTMs is a entire LSTM RNN, processing the

entire volume in one direction. The directions $\mathcal{D}$ are formally defined over the three axes (x, y, z): $\mathcal{D} = \{(\cdot, \cdot, 1), (\cdot, \cdot, -1), (\cdot, 1, \cdot), (\cdot, -1, \cdot), (1, \cdot, \cdot), (-1, \cdot, \cdot)\}$. They essentially choose which axis is the *time* direction; i.e. with $(\cdot, \cdot, 1)$ the positive direction of the z-axis represents the time.

Each C-LSTM performs computations in a plane moving in the defined direction. The input is $x \in \mathbb{R}^{W \times H \times D \times C}$, where W is the width, H the height, D the depth, and C the number of channels of the input, or hidden units in the case of second- and higher layers. Similarly, we define the volumes $f^d, i^d, o^d, \tilde{c}^d, c^d, h^d, h \in \mathbb{R}^{W \times H \times D \times O}$, where $d \in \mathcal{D}$ is a direction and O is the number of hidden units per pixel. Since each direction needs a separate volume, we denote volumes with $(\cdot)^d$.

The time index t selects a slice in direction d. For instance, for direction $d = (\cdot, \cdot, 1)$, v_t^d refers to the plane x, y, z, c for $x = 1..X, y = 1..Y, c = 1..C$, and $z = t$. For a negative direction $d = (\cdot, \cdot, -1)$, the plane is the same but moves in the opposite direction: $z = Z - t$. A special case is the first plane in each direction, which does not have a previous plane, hence we omit the corresponding computation.

C-LSTM equations:

$$i_t^d = \sigma(x_t^d * \theta_{xi}^d + h_{t\text{-}1}^d * \theta_{hi}^d + \theta_{i_{\text{bias}}}^d), \tag{7}$$

$$f_t^d = \sigma(x_t^d * \theta_{xf}^d + h_{t\text{-}1}^d * \theta_{hf}^d + \theta_{f_{\text{bias}}}^d), \tag{8}$$

$$\tilde{c}_t^d = \tanh(x_t^d * \theta_{x\tilde{c}}^d + h_{t\text{-}1}^d * \theta_{h\tilde{c}}^d + \theta_{\tilde{c}_{\text{bias}}}^d), \tag{9}$$

$$c_t^d = \tilde{c}_t^d \odot i_t^d + c_{t\text{-}1}^d \odot f_t^d, \tag{10}$$

$$o_t^d = \sigma(x_t^d * \theta_{xo}^d + h_{t\text{-}1}^d * \theta_{ho}^d + \theta_{o_{\text{bias}}}^d), \tag{11}$$

$$h_t^d = o_t^d \odot \tanh(c_t^d), \tag{12}$$

$$h = \sum_{d \in \mathcal{D}} h^d, \tag{13}$$

where $(*)$ is a convolution[3], and h is the output of the layer. All biases are the same for all LSTM units (i.e., no positional biases are used). The outputs h^d for all directions are summed together.

Fully-Connected Layer: The output of our PyraMiD-LSTM layer is connected to a pixel-wise fully-connected layer, which output is squashed by the hyperbolic tangent ($\tanh$) function. This step is used to increase the number of channels for the next layer. The final classification is done using a pixel-wise softmax function: $y(x, y, z, c) = \frac{e^{-h(x,y,z,c)}}{\sum_c e^{-h(x,y,z,c)}}$ giving pixel-wise probabilities for each class.

3 Experiments

We evaluate our approach on two 3D biomedical image segmentation datasets: electron microscopy (EM) and MR Brain images.

EM dataset The EM dataset [12] is provided by the ISBI 2012 workshop on Segmentation of Neuronal Structures in EM Stacks [15]. Two stacks consist of 30 slices of 512×512 pixels obtained from a $2 \times 2 \times\ 1.5\ \mu m^3$ microcube with a resolution of $4 \times 4 \times 50\ nm^3$/pixel and binary labels. One stack is used for training, the other for testing. Target data consists of binary labels (membrane and non-membrane).

MR Brain dataset The MR Brain images are provided by the ISBI 2015 workshop on Neonatal and Adult MR Brain Image Segmentation (ISBI NEATBrainS15) [13]. The dataset consists of twenty fully annotated high-field (3T) multi-sequences: 3D T1-weighted scan (T1), T1-weighted inversion recovery scan (IR), and fluid-attenuated inversion recovery scan (FLAIR). The dataset is divided into a training set with five volumes and a test set with fifteen volumes. All scans are bias-corrected and aligned. Each volume includes 48 slices with 240×240 pixels ($3mm$ slice thickness). The slices

[3] In 3D volumes, convolutions are performed in 2D; in general an n-D volume requires n-1-D convolutions. All convolutions have stride 1, and their filter sizes should at least be 3×3 in each dimension to create the full context.

are manually segmented through nine labels: cortical gray matter, basal ganglia, white matter, white matter lesions, cerebrospinal fluid in the extracerebral space, ventricles, cerebellum, brainstem, and background. Following the ISBI NEATBrainS15 workshop procedure, all labels are grouped into four classes and background: 1) cortical gray matter and basal ganglia (GM), 2) white matter and white matter lesions (WM), 3) cerebrospinal fluid and ventricles (CSF), and 4) cerebellum and brainstem. Class 4) is ignored for the final evaluation as required.

Sub-volumes and Augmentation The full dataset requires more than the 12 GB of memory provided by our GPU, hence we train and test on sub-volumes. We randomly pick a position in the full data and extract a smaller cube (see the details in *Bootstrapping*). This cube is possibly rotated at a random angle over some axis and can be flipped over any axis. For EM images, we rotate over the z-axis and flipped sub-volumes with 50% chance along x, y, and z axes. For MR brain images, rotation is disabled; only flipping along the x direction is considered, since brains are (mostly) symmetric in this direction.

During test-time, rotations and flipping are disabled and the results of all sub-volumes are stitched together using a Gaussian kernel, providing the final result.

Pre-processing We normalise each input slice towards a mean of zero and variance of one, since the imaging methods sometimes yield large variability in contrast and brightness. We do not apply the complex pre-processing common in biomedical image segmentation [10].

We apply simple pre-processing on the three datatypes of the MR Brain dataset, since they contain large brightness changes under the same label (even within one slice; see Figure 5). From all slices we subtract the Gaussian smoothed images (filter size: 31×31, $\sigma = 5.0$), then a Contrast-Limited Adaptive Histogram Equalisation (CLAHE) [16] is applied to enhance the local contrast (tile size: 16×16, contrast limit: 2.0). An example of the images after pre-processing is shown in Figure 5. The original and pre-processed images are all used, except the original IR images (Figure 5b), which have high variability.

Training We apply RMS-prop [17] with momentum. We define $a \overset{\rho}{\leftarrow} b$ to be $a_n = \rho a_n + (1-\rho) b_n$, where $a, b \in \mathbb{R}^N$. The following equations hold for every epoch:

$$E = (y^* - y)^2, \tag{14}$$

$$\text{MSE} \overset{\rho_{\text{MSE}}}{\leftarrow} \nabla_\theta^2 E, \tag{15}$$

$$G = \frac{\nabla_\theta E}{\sqrt{\text{MSE}} + \epsilon}, \tag{16}$$

$$M \overset{\rho_M}{\leftarrow} G, \tag{17}$$

$$\theta = \theta - \lambda_{\text{lr}} M, \tag{18}$$

where y^* is the target, y is the output from the networks, E is the squared loss, MSE a running average of the variance of the gradient, ∇^2 is the element-wise squared gradient, G the normalised gradient, M the smoothed gradient, and θ the weights. The squared loss was chosen as it produced better results than using the log-likelihood as an error function. This algorithm normalises the gradient of each weight, such that even weights with small gradients get updated. This also helps to deal with vanishing gradients [18].

We use a decaying learning rate: $\lambda_{\text{lr}} = 10^{-6} + 10^{-2} \cdot \frac{1}{2}^{\frac{\text{epoch}}{100}}$, which starts at $\lambda_{\text{lr}} \approx 10^{-2}$ and halves every 100 epochs asymptotically towards $\lambda_{\text{lr}} = 10^{-6}$. Other hyper-parameters used are $\epsilon = 10^{-5}$, $\rho_{\text{MSE}} = 0.9$, and $\rho_M = 0.9$.

Bootstrapping To speed up training, we run three learning procedures with increasing sub-volume sizes: first, 3000 epochs with size $64 \times 64 \times 8$, then 2000 epochs with size $128 \times 128 \times 15$. Finally, for the EM-dataset, we train 1000 epochs with size $256 \times 256 \times 20$, and for the MR Brain dataset 1000 epochs with size $240 \times 240 \times 25$. After each epoch, the learning rate λ_{lr} is reset.

Table 1: Performance comparison on EM images. Some of the competing methods reported in the ISBI 2012 website are not yet published. Comparison details can be found under `http://brainiac2.mit.edu/isbi_challenge/leaders-board`.

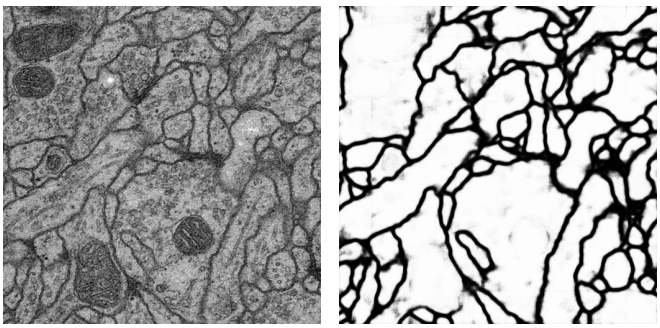
Group	Rand Err.	Warping Err.($\times 10^{-3}$)	Pixel Err.
Human	0.002	0.0053	0.001
Simple Thresholding	0.450	17.14	0.225
IDSIA [11]	0.050	0.420	0.061
DIVE	0.048	**0.374**	**0.058**
PyraMiD-LSTM	**0.047**	0.462	0.062
IDSIA-SCI	0.0189	0.617	0.103
DIVE-SCI	0.0178	0.307	0.058

Experimental Setup All experiments are performed on a desktop computer with an NVIDIA GTX TITAN X 12GB GPU. Due to the pyramidal topology all major computations can be done using convolutions with NVIDIA's cuDNN library [19], which has reported $20\times$ speedup over an optimised implementation on a modern 16 core CPU. On the MR brain dataset, training took around three days, and testing per volume took around 2 minutes.

We use exactly the same hyper-parameters and architecture for both datasets. Our networks contain three PyraMiD-LSTM layers. The first PyraMiD-LSTM layer has 16 hidden units followed by a fully-connected layer with 25 hidden units. In the next PyraMiD-LSTM layer, 32 hidden units are connected to a fully-connected layer with 45 hidden units. In the last PyraMiD-LSTM layer, 64 hidden units are connected to the fully-connected output layer whose size equals the number of classes.

The convolutional filter size for all PyraMiD-LSTM layers is set to 7×7. The total number of weights is 10,751,549, and all weights are initialised according to a uniform distribution: $\mathcal{U}(-0.1, 0.1)$.

3.1 Neuronal Membrane Segmentation



(a) Input **(b)** PyraMiD-LSTM

Figure 4: Segmentation results on EM dataset (slice 26)

Membrane segmentation is evaluated through an online system provided by the ISBI 2012 organisers. The measures used are the Rand error, warping error and pixel error [15]. Comparisons to other methods are reported in Table 1. The teams IDSIA and DIVE provide membrane probability maps for each pixel. The IDSIA team uses a state-of-the-art deep convolutional network [11], the method of DIVE was not provided.

These maps are adapted by the post-processing technique of the teams SCI [20], which directly optimises the rand error (DIVE-SCI (top-1) and IDSIA-SCI (top-2)); this is most important in this particular segmentation task.

Table 2: The performance comparison on MR brain images.

Structure	GM			WM			CSF			
Metric	DC (%)	MD (mm)	AVD (%)	DC (%)	MD (mm)	AVD (%)	DC (%)	MD (mm)	AVD (%)	Rank
BIGR2	84.65	1.88	6.14	88.42	2.36	**6.02**	78.31	3.19	22.8	6
KSOM GHMF	84.12	1.92	**5.44**	87.96	2.49	6.59	82.10	2.71	12.8	5
MNAB2	84.50	1.69	7.10	88.04	2.12	7.73	82.30	2.27	8.73	4
ISI-Neonatology	**85.77**	**1.62**	6.62	88.66	**2.06**	6.96	81.08	2.66	9.77	3
UNC-IDEA	84.36	**1.62**	7.04	**88.69**	**2.06**	6.46	82.81	2.35	10.5	2
PyraMiD-LSTM	84.82	1.69	6.77	88.33	2.07	7.05	**83.72**	**2.14**	**7.10**	**1**

Without post-processing, PyraMiD-LSTM networks outperform other methods in rand error, and are competitive in wrapping and pixel errors. Of course, performance could be further improved by applying post-processing techniques. Figure 4 shows an example segmentation result.

3.2 MR Brain Segmentation

The results are compared using the DICE overlap (DC), the modified Hausdorff distance (MD), and the absolute volume difference (AVD) [13]. MR brain image segmentation results are evaluated by the ISBI NEATBrain15 organisers [13] who provided the extensive comparison to other approaches on `http://mrbrains13.isi.uu.nl/results.php`. Table 2 compares our results to those of the top five teams. The organisers compute nine measures in total and rank all teams for each of them separately. These ranks are then summed per team, determining the final ranking (ties are broken using the standard deviation). PyraMiD-LSTM leads the final ranking with a new state-of-the-art result and outperforms other methods for CSF in all metrics.

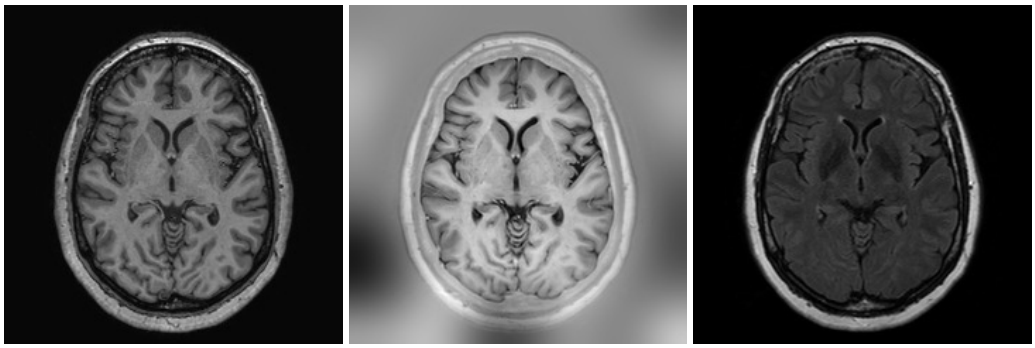
We also tried regularisation through dropout [21]. Following earlier work [22], the dropout operator is applied only to non-recurrent connections (50% dropout on fully connected layers and/or 20% on input layer). However, this did not improve performance.

4 Conclusion

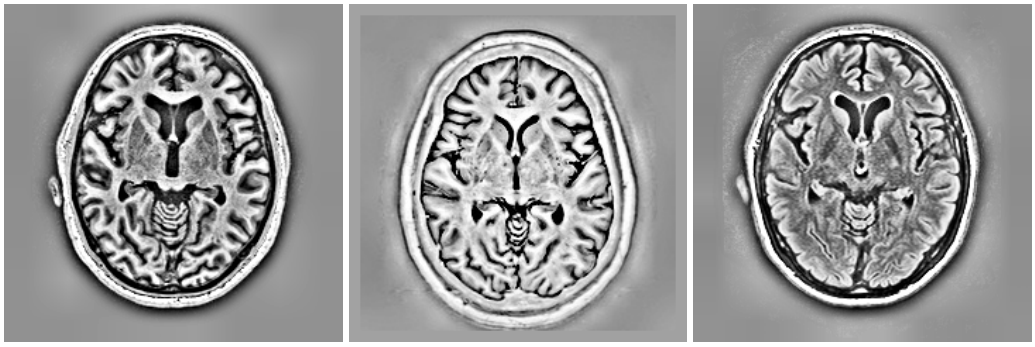
Since 2011, GPU-trained max-pooling CNNs have dominated classification contests [23, 24, 25] and segmentation contests [11]. MD-LSTM, however, may pose a serious challenge to such CNNs, at least for segmentation tasks. Unlike CNNs, MD-LSTM has an elegant recursive way of taking each pixel's entire spatio-temporal context into account, in both images and videos. Previous MD-LSTM implementations, however, could not exploit the parallelism of modern GPU hardware. This has changed through our work presented here. Although our novel highly parallel PyraMiD-LSTM has already achieved state-of-the-art segmentation results in challenging benchmarks, we feel we have only scratched the surface of what will become possible with such PyraMiD-LSTM and other MD-RNNs.

5 Acknowledgements

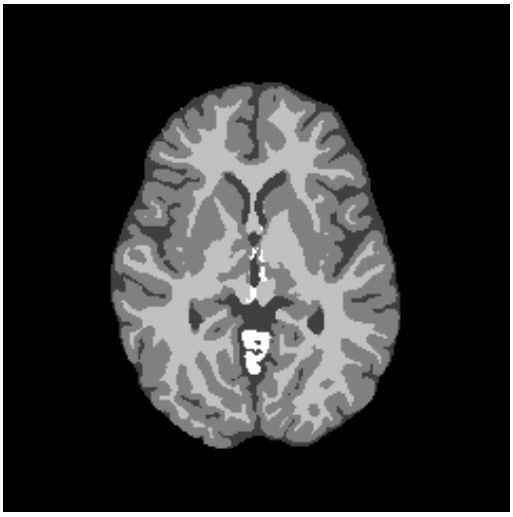
We would like to thank Klaus Greff and Alessandro Giusti for their valuable discussions, and Jan Koutnik and Dan Ciresan for their useful feedback. We also thank the ISBI NEATBrain15 organisers [13] and the ISBI 2012 organisers, in particular Adriënne Mendrik and Ignacio Arganda-Carreras. Lastly we thank NVIDIA for generously providing us with hardware to perform our research. This research was funded by the NASCENCE EU project (EU/FP7-ICT-317662).



(a) T1 **(b)** IR **(c)** FLAIR



(d) T1 (pre-processed) **(e)** IR (pre-processed) **(f)** FLAIR (pre-processed)



(g) segmentation result from PyraMiD-LSTM

Figure 5: Slice 19 of the test image 1. (a)-(c) are examples of three scan methods used in the MR brain dataset, and (d)-(f) show the corresponding images after our pre-processing procedure (see pre-processing in Section). Input (b) is omitted due to strong artefacts in the data — the other datatypes are all used as input to the PyraMiD-LSTM. The segmentation result is shown in (g).

References

[1] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997). Based on TR FKI-207-95, TUM (1995), pp. 1735–1780.

[2] F. A. Gers, J. Schmidhuber, and F. Cummins. “Learning to Forget: Continual Prediction with LSTM”. In: *ICANN*. 1999.

[3] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. “A Novel Connectionist System for Improved Unconstrained Handwriting Recognition”. In: *PAMI* 31.5 (2009).

[4] H. Sak, A. Senior, and F. Beaufays. “Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling”. In: *Proc. Interspeech*. 2014.

[5] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. Tech. rep. NIPS. 2014.

[6] A. Graves, S. Fernández, and J. Schmidhuber. “Multi-dimensional Recurrent Neural Networks”. In: *ICANN*. 2007.

[7] A. Graves and J. Schmidhuber. “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”. In: *NIPS*. 2009.

[8] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. “Scene Labeling With LSTM Recurrent Neural Networks”. In: *CVPR*. 2015.

[9] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active contour models”. In: *International Journal of Computer Vision* (1988).

[10] L. Wang, Y. Gao, F. Shi, G. Li, J. H. Gilmore, W. Lin, and D. Shen. “LINKS: Learning-based multi-source IntegratioN frameworK for Segmentation of infant brain images”. In: *NeuroImage* (2015).

[11] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. “Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images”. In: *NIPS*. 2012.

[12] A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomancak, and V. Hartenstein. “An integrated micro-and macroarchitectural analysis of the Drosophila brain by computer-assisted serial section electron microscopy”. In: *PLoS biology* 8.10 (2010), e1000502.

[13] A. M. Mendrik, K. L. Vincken, H. J. Kuijf, G. J. Biessels, and M. A. Viergever (organizers). *MRBrainS Challenge: Online Evaluation Framework for Brain Image Segmentation in 3T MRI Scans,* `http://mrbrains13.isi.uu.nl`. 2015.

[14] O. Weber, Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel. “Parallel algorithms for approximation of distance maps on parametric surfaces”. In: *ACM Transactions on Graphics* (2008).

[15] Segmentation of Neuronal Structures in EM Stacks Challenge. *IEEE International Symposium on Biomedical Imaging (ISBI), http://tinyurl.com/d2fgh7g*. 2012.

[16] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. T. H. Romeny, and J. B. Zimmerman. “Adaptive Histogram Equalization and Its Variations”. In: *Comput. Vision Graph. Image Process.* (1987).

[17] T. Tieleman and G. Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural Networks for Machine Learning* 4 (2012).

[18] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. Advisor: J. Schmidhuber. 1991.

[19] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. “cuDNN: Efficient Primitives for Deep Learning”. In: *CoRR* abs/1410.0759 (2014).

[20] T. Liu, C. Jones, M. Seyedhosseini, and T. Tasdizen. “A modular hierarchical approach to 3D electron microscopy image segmentation”. In: *Journal of Neuroscience Methods* (2014).

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* (2014).

[22] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. “Dropout improves recurrent neural networks for handwriting recognition”. In: *ICFHR*. 2014.

[23] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. “A Committee of Neural Networks for Traffic Sign Classification”. In: *IJCNN*. 2011.

[24] A. Krizhevsky, I Sutskever, and G. E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NIPS*. 2012.

[25] M. D. Zeiler and R. Fergus. *Visualizing and Understanding Convolutional Networks*. Tech. rep. arXiv:1311.2901 [cs.CV]. NYU, 2013.