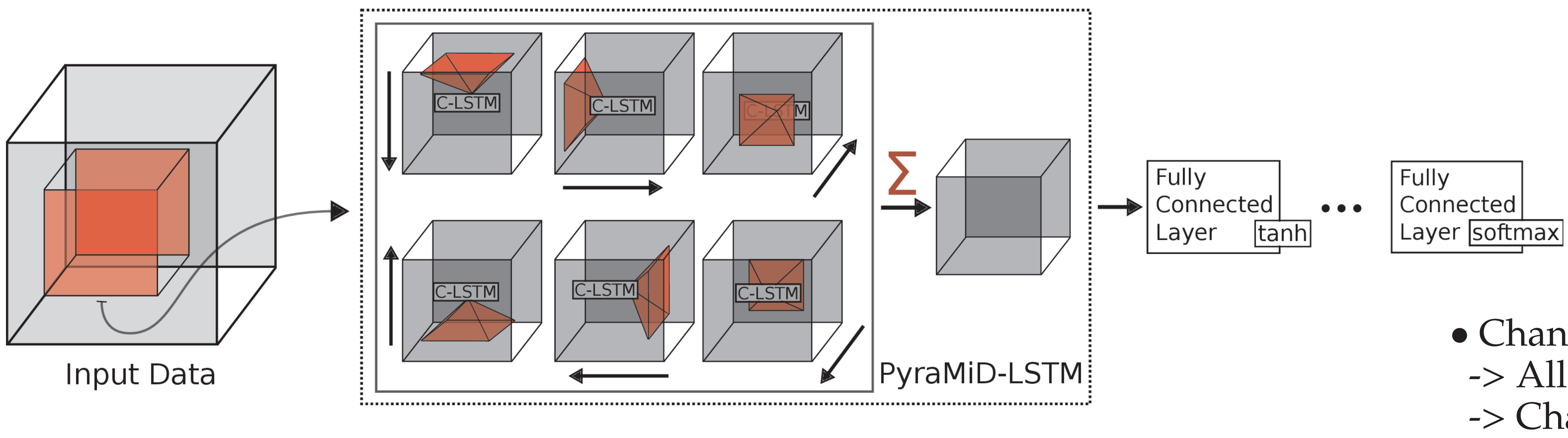
Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation

Marijn Stollenga*, Wonmin Byeon*, Marcus Liwicki, Juergen Schmidhuber

Abstract

Unlike CNNs, 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).

Architecture



Layer Equations

$$i_t^d = \sigma(x_t^d * \theta_{xi}^d + h_{t\text{-}1}^d * \theta_{hi}^d + \theta_{i_{\text{bias}}}^d)$$
$$f_t^d = \sigma(x_t^d * \theta_{xf}^d + h_{t\text{-}1}^d * \theta_{hf}^d + \theta_{f_{\text{bias}}}^d)$$
$$\tilde{c}_t^d = \tanh(x_t^d * \theta_{x\tilde{c}}^d + h_{t\text{-}1}^d * \theta_{h\tilde{c}}^d + \theta_{\tilde{c}_{\text{bias}}}^d)$$
$$c_t^d = \tilde{c}_t^d \odot i_t^d + c_{t\text{-}1}^d \odot f_t^d$$
$$o_t^d = \sigma(x_t^d * \theta_{xo}^d + h_{t\text{-}1}^d * \theta_{ho}^d + \theta_{o_{\text{bias}}}^d)$$
$$h_t^d = o_t^d \odot \tanh(c_t^d)$$
$$h = \sum_{d \in \mathcal{D}} h^d$$

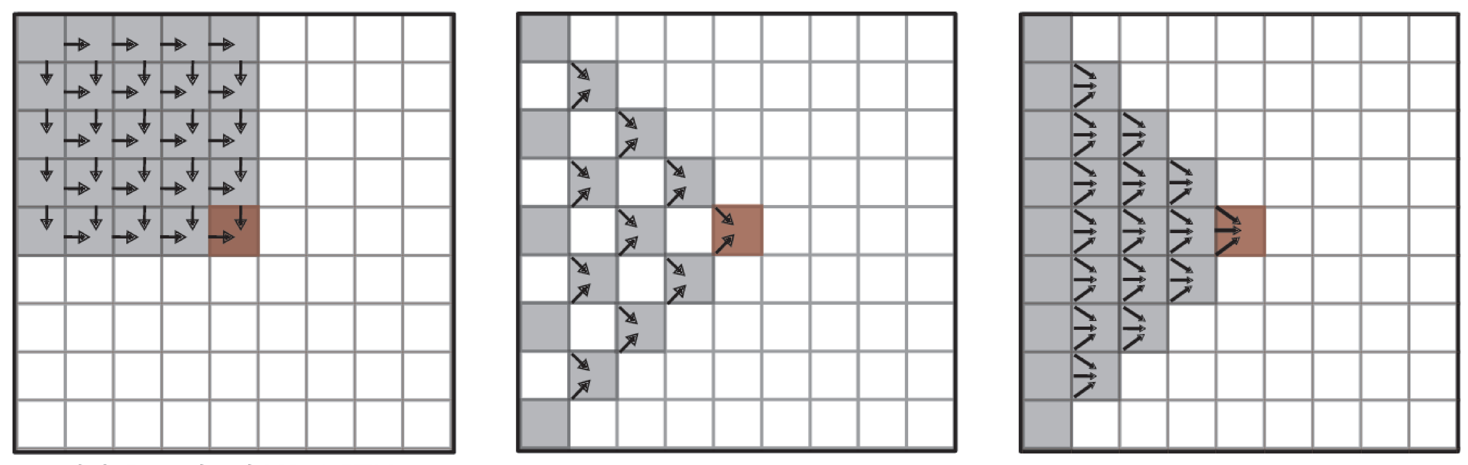
Training

$$E = (y^* - y)^2$$
$$\text{MSE} \xleftarrow{\rho_{\text{MSE}}} \nabla_\theta^2 E$$
$$G = \frac{\nabla_\theta E}{\sqrt{\text{MSE} + \epsilon}}$$
$$M \xleftarrow{\rho_M} G$$
$$\theta = \theta - \lambda_{\text{lr}} M$$

Highlight

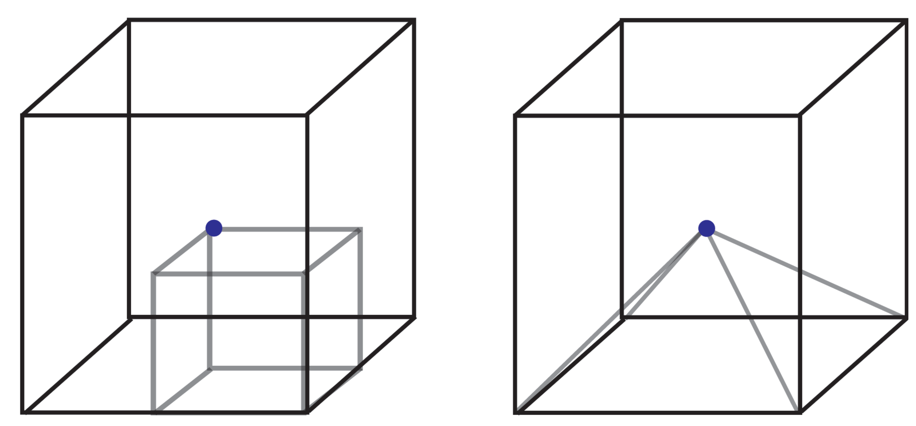
- Easy to parallelise on GPUs
- Retain all advantages of LSTMs
  -> integrate information over long distance where context plays a big role.
- Result in great performance on volumetric segmentation
  -> Best known pixel-level brain image segmentation results

Idea



(a) Standard MD-LSTM (b) 'Turned' MD-LSTM (c) PyraMiD LSTM

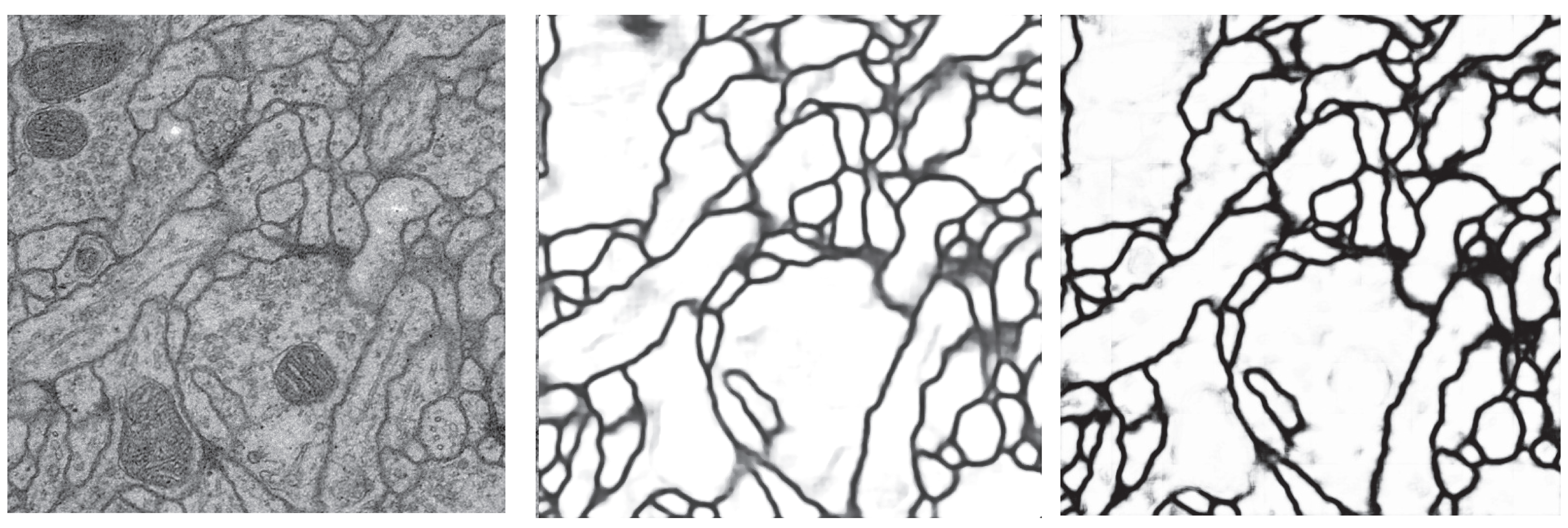
- Change the traditional MD-LSTM structure
  -> All operations in PyraMiD-LSTM can be implemented using convolutions.
  -> Changes the context from block-form to a pyramidal shape
  -> More efficiently covers volume
- The convolutional nature changes several properties
  -> Number of computational steps between pixels becomes smaller
  -> Filters align with the way the data is recorded (parallel to the planes, not diagonal)



Membrane Segmentation

ISBI 2012 Challenges

- 512 x 512 pixels, 30 slices per volume
- 1 volume for training, 1 volume for testing



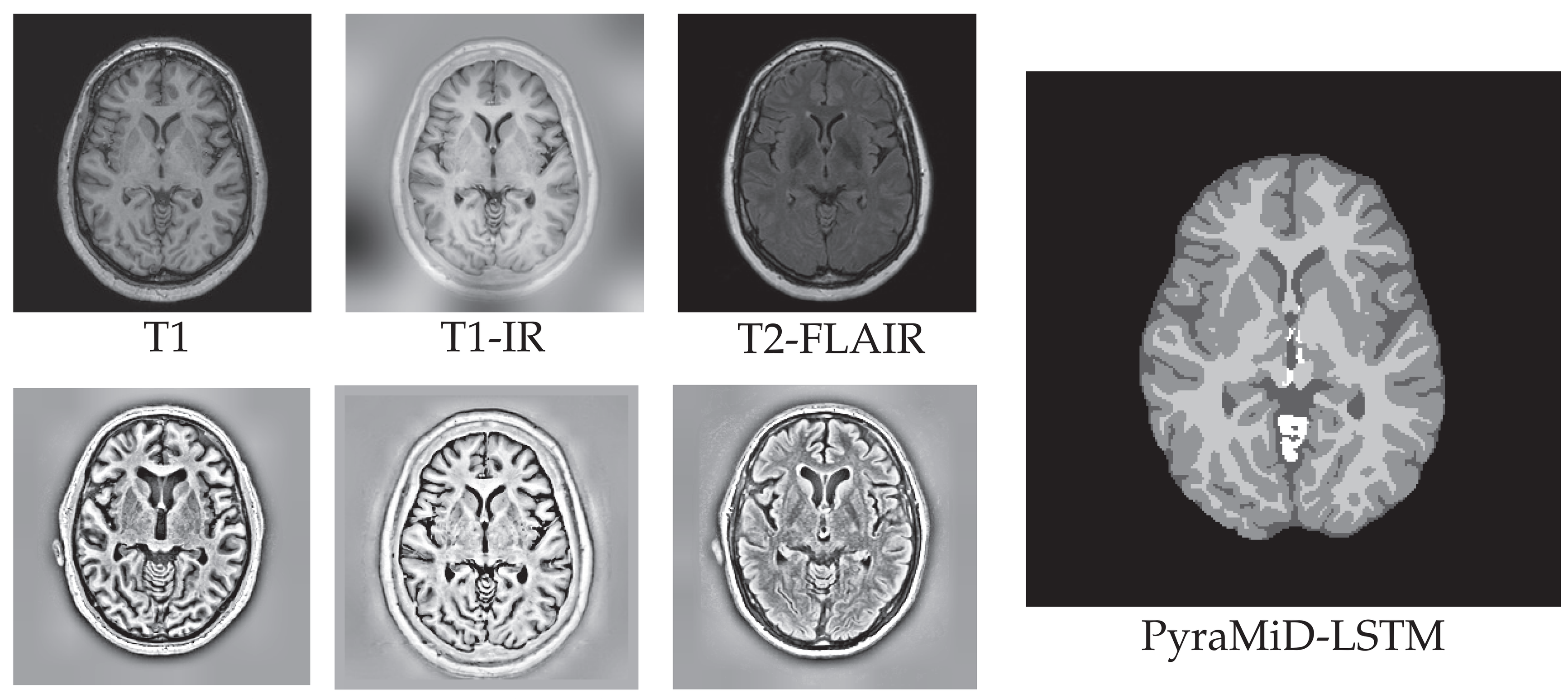
Input | IDSIA (CNNs with post-processing) | PyraMiD-LSTM

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 [2]	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

Brain CT Scan Segmentation

ISBI 2015 Challenges

- 240 x 240 pixels, 48 slices per volume
- three high-field multi-sequences: T1-weighted scan (T1), T1-weighted inversion recovery scan (IR), and fluid-attenuated inversion recovery scan (FLAIR)
- 5 volumes for training, 15 volumes for testing



T1 | T1-IR | T2-FLAIR | T1 (Post) | T1-IR (Post) | T2-FLAIR (Post) | PyraMiD-LSTM

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**

References

[1] http://brainiac2.mit.edu/isbi_challenge/leaders-board}

[2] D. C. Ciresan, et. al. (Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images(. In: NIPS. 2012.