Title:
Towards Integration of Algebraic Topological Methods in Deep Learning from Medical Images

Authors:
Francesco Furiani, f.furio@gmail.com, Roma Tre University
Enrico Marino, marino@dia.uniroma3.it, Roma Tre University
Federico Spini, spini@dia.uniroma3.it, Roma Tre University
Alberto Paoluzzi, alberto.paoluzzi@uniroma3.it, Roma Tre University

Keywords:
Deep Learning, Medical Imaging, Segmentation, Model Extraction, Linear Algebraic Representation, LAR

DOI: 10.14733/cadconfP.2017.478-481

Introduction:

We discuss here the introduction of basic algebraic topological methods, specifically linear spaces of chains and cochains, and linear operators of boundary/coboundary and their combinations, in 3D medical image segmentation, using state-of-the-art software and hardware. In this abstract we describe how we started using a *deep learning* approach for performing 3D medical image segmentation, by using unsupervised system identification of analytical models and algebraic topology methods with LAR [5]. In our research we use *"chains" of image cells* as representations in neural network (NN), and employ sparse matrices of boundary and coboundary operators to discriminate between statistical neighborhoods of various portions of body organs.

The research is within the framework of *MedTrain3dModSim*, an EU project in the Erasmus+ program, where an international group of researchers [7] from Europe and Korea started preparing CAD tools for training of students of medical schools within an international curriculum, using medical 3D modelling for education and treatment. This endeavor will make use of 3D printed models of body organs and virtual and augmented reality tools, in the perspective of going towards environments of modeling and simulation for computer-aided surgery. This project strongly relies on the IEEE project for standardization of model extraction from 3D medical images [9].

In the framework described above, we started using deep learning for 3D medical image segmentation. The concept of machine learning has been around for decades, but can now be applied to huge quantities of data. Cheaper data storage, distributed processing, and more powerful computers available have dramatically increased the interest in machine learning using deep unsupervised architectures, and giving rise to the highly successful methods of "deep learning". [6,8]

Method background:

Deep learning with CNN

Artificial Intelligence is blooming in most recent years, due to a resurgence of *neural networks*, a computational technique known from 1970s. This resumption is not causal, since early networks were plagued by issues of inadequate computing power. With the increase in the ratio of computing power over cost, caused by the mass marketing of graphical processing units, application-specific integrated circuit chips, and field-programmable gate arrays, it was possible a comeback and a boost of previously inefficient, but solidly founded techniques.

In machine learning, a convolutional neural network (CNN) is a kind of feed-forward network of simple computational units, connected like the visual cortex in humans, and (mainly) using the discrete convolution operator to elaborate the stimuli. Current neural networks with many variables (layers and nodes) can solve problems that were not thinkable of ten years ago. E.g. winning GO games against human champions, detecting cancerous skin lesion better than a trained professional, detecting the language grammar from a large set of similar sentences, finding winning strategies in games with partial information, and so on.

A neural network is a *dataflow graph* for numeric computations. A network of computational nodes (called *neurons*) provides a stratified computing infrastructure, where the various layers of nodes apply in parallel their SIMD code to the input (*features*) of layer, so producing the layer output, which is input for the next layer of nodes, and so on, until to obtain a valuable output. If we represent the input to a node as a vector $\mathbf{x}$ and the output as a vector $\mathbf{y}$, the typical computation is $\mathbf{y} = \mathbf{W}\,\mathbf{x} + \mathbf{b}$, using matrices of *weights* and vectors of *biases*. Weights and biases are computed when training the network, by minimizing a cost function associated to the known output from data, via *back-propagation* using fast derivatives and stochastic gradient descent.

Such a dataflow graph performs essentially a composition of linear or non-linear functions, that generally form a directed acyclic graph of simple computational nodes. Some computational graphs have loops, as e.g. in recurrent neural networks. Convolutional Neural Networks use discrete convolutions to extract meaningful information about the implicit structure and ordering of the input data. Discrete convolutions can be represented as matrix multiplication by a Toeplitz matrix, thus preserving the easiness of calculation of the forward and backward phases in the computation workflow.

Solid models from 3D images with LAR:

Consistently with the growing availability of quantitative data, the interest in physically-based simulations, customary in engineering CAD, is now strongly growing in medicine field, with the clinical aim of getting a better understanding of physiology and pathologies on a single-patient basis, using personalized models extracted from patient's body scans [11]. To this purpose, we cope with geometric-topological data using the LAR scheme, characterized by a very large domain, encompassing both 2D and 3D meshes, manifold and non-manifold geometric and solid models, and low- and high-resolution 2D and 3D images.

LAR (Linear Algebraic Representation) is a general-purpose representation scheme for geometric and solid modeling introduced recently [5]. The domain of the scheme is provided by dimension-independent *cellular complexes*, while its codomain is that of *sparse matrices*, stored using either the CSR (Compressed Sparse Row) or the CSC (Compressed Sparse Column) 's memory format. The LAR polyhedral domain coincides with complexes of connected d-cells, even non-convex and/or including any number of holes. Also, the algebraic foundation of LAR allows not only for fast queries about incidence and adjacency of cells, but also to resolve -- via *SpMV* accelerated kernels -- the boundary extraction of any 3D subset of the model.

It is worth noting that LAR provides a direct management of all subsets of cells and their physical properties trough the linear spaces of chains induced by the model partitioning, and their dual spaces of cochains. The linear operators of boundary and coboundary between such linear spaces, suitably implemented by sparse matrices, directly provide, depending on the dimension of the mapped spaces, the discrete differential operators of gradient, curl and divergence, while their product gives the Laplacian [4].

Computational tools:

Julia language and larlib.jl

Julia is a young dynamic programming language for scientific and technical computing [2]. The syntax of the language is comparable to Python and MatLab, including optional (static) type checking for documentation, optimization, and dynamic dispatch of methods, so providing the ability to define function behavior across many combinations of argument types. While the syntax is that of a scripting language, the computational performance is close to C, which can be called directly from Julia code. Linear algebra functions are largely implemented in Julia by calling functions from LAPACK and BLAS. Including libraries from other languages (e.g. from Python) is also possible.

Julia's multidimensional arrays are 1-based, as in Fortran and in MatLab, with storage organized by columns. Julia provides a native implementation of functions in the CSparse and CXSparse libraries developed by [3] and an efficient interface to the the NVIDIA CUSPARSE library, that is a high-performance sparse matrix linear algebra package [1]. In the past few months we were porting the bulk of Larlib, the Python implementation of the LAR scheme, to a Julia package named larlib.jl, hosted on GitHub as most of Julia libraries. The package provides a fast implementation of basic topological computations, that we are going to use in this project.

Nvidia DGX-1 for deep learning

In this plan we are using the powerful DGX-1 machine, marketed in the second part of 2016 by NVIDIA as the “AI supercomputer for deep learning”. The DGX-1 system is a two-socket server with two 16-core Xeon processors, and PCI-Express switches to link a complex of *height* Pascal GP100 GPUs to the Xeon processors. The CPU has 512 GB of DDR4 memory, and four 1.92 TB flash SSDs for high bandwidth storage, which is necessary to keep the CPUs and GPUs fed. Using half-precision FP16 data storage in the GPU memory, the eight Pascal GPUs can deliver *170 teraflops* of aggregate performance, each one addressing a total 128 GB of graphics HBM2 (High Bandwidth Memory 2). This huge amount of computational power is specifically addressed by NVIDIA to deep learning, by providing in the same package both the hardware and the most successful software for neural networks.

Segmentation experiments:

Quantitative analysis of medical 3D images requires segmentation of anatomical structures. To improve and generalize the extraction of solid models from 3D medical images is the main goal our group, in this first stage of the MedTrain3dModSim project.

In order to get a quick start, we are choosing to adopt (quite) integrally the CNN architecture and training data (from the MICCAI 2012 multi-atlas challenge) developed by (Pai et al. 2017) for brain segmentation, where the architecture can automatically learn hierarchies of relevant features from raw data, since it reduces the instances of data needed for training, via a better approach to estimate relative errors.

Since in our project we do not have storage limitations, due to using our new DGX-1 machine, we aim to use real 3D features, and not tri-planar image patches (sagittal, coronal, and transverse). In this environment, we will first substitute the current use of mathematical morphology filters to estimate the growing boundaries of features, with actual computation of boundary and coboundary operators over 3D arrays. Then, when having reproduced the (Pai et al. 2017) classification results starting from their (modified) architecture, we will attempt to get incremental improvements, by carefully introducing adjacency constraints between labeled sub-regions, by using incidence operators that are algebraic combinations of boundary and coboundary operators.

In conclusion, it may be useful to quote here that “The choice of where to include the geometric information in the network is a research topic in itself” (Pai et al. 2017). In our project we are therefore attempting this challenging undertaking. To get some initial, but hopefully positive answers, will probably take some more months.

References:

[1] Bell, N.; Garland, M.: Implementing Sparse Matrix-Vector Multiplication on Throughput-Oriented Processors, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 18:1–18:11. SC '09. New York, NY, USA, 2009, ACM. https://doi.org/10.1145/1654059.1654078

[2] Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V. B.: Julia: A Fresh Approach to Numerical Computing, 2014, http://arxiv.org/abs/1411.1607.

[3] Davis, T. A.: Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2). Philadelphia, PA, USA: Society for Industrial; Applied Mathematics, 2006.

[4] DiCarlo, A.; Milicchio, F.; Paoluzzi, A.; Shapiro, V.: Chain-Based Representations for Solid and Physical Modeling, Automation Science and Engineering, IEEE Transactions on, 6(3), 2009, 454–67. https://doi.org/10.1109/TASE.2009.2021342

[5] DiCarlo, A., Paoluzzi, A.; Shapiro, V.: Linear Algebraic Representation for Topological Structures. Comput. Aided Des. 46(1), 2014, 269–74. https://doi.org/10.1016/j.cad.2013.08.044

[6] Goodfellow, I.; Bengio, Y.; Courville, A.: Deep Learning. MIT Press, 2017.

[7] Huri, E.; Tatar,I.; Yıldırım, Y.; Moon,Y. M.; DiCarlo, A.; Paoluzzi, A.; Liska, V.; Hosek, P.; Skolarikos, A.: MedTrain3dModSim Kick-Off Meeting. Istambul, Turkey, February 10–12, 2017, http://www.medtrain3dmodsim.eu.

[8] Masci, J.; Rodolà, E.; Boscaini, D.; Bronstein, M. M.; Li,H.: Geometric Deep Learning, in SIGGRAPH Asia 2016 Courses, 1:1–1:50, New York, NY, USA: ACM. 2016. https://doi.org/10.1145/2988458.2988485

[9] Moon, Y. L.; Sugamoto, K.; Paoluzzi, A.; DiCarlo, A.; Kwak, j.; Shin, D. S.; Kim, D.O.; Lee, D. H.; Kim, J. Y.:. Standardizing 3D Medical Imaging. Computer 47(4), 2014, 76–79. http://doi.ieeecomputersociety.org/10.1109/MC.2014.103.

[10] Pai, A.; Teng, Y,-C.; Blair, J.; Kallenberg, M.; Dam, E.B.; Sommer, S.; Igel, C.; Nielsen, M.: Characterization of Errors in Deep-Learning Based Brain Mri Segmentation, Deep Learning for Medical Image Analysis. Academic Press, 2017.

[11] Paoluzzi, A.; DiCarlo, A.; Furiani, F.; and JiÌrik, M.: CAD Models from Medical Images Using Lar, Computer-Aided Design and Applications, 13(6), 2016, 747-759. https://doi.org/10.1080/16864360.2016.1168216