doi:10.1017/S1431927619002745
Microsc. Microanal. 25 (Suppl 2), 2019

Simplifying and Streamlining Large-Scale Materials Image Processing with Wizard-Driven and Scalable Deep Learning

Benjamin Provencher[1], Nicolas Piché[1] and Mike Marsh[2*]

[1.] Object Research Systems. Montreal, Canada.
[2.] Object Research Systems. Denver, USA.
*Corresponding author: mmarsh@theobjects.com

Image methods and imaging throughput--especially in 3D and 4D--continue to evolve and elucidate rich details about materials samples at an ever increasing pace. The full potential for quantitative analysis of those materials, however, is often limited by the speed with which imaging scientists can process the data. For different studies, those bottlenecks can occur both for image analyst time and compute time. To address these rate limitations of image processing, we present an improved Deep Learning engine with a user-guided wizard that simplifies and accelerates user interaction and is parallelized for high-throughput computation on both typical laboratory computer hardware and high-performance compute (HPC) systems. Deep Learning tasks here include, but are not limited to, the fully automatic and parameter-free operations of image denoising and image segmentation.

Deep Learning is the machine learning approach of supervised learning implemented on deep neural networks (convolutional neural networks for image applications). We previously described our Deep Learning solution [1] integrated into the Dragonfly software platform, a feature-rich image processing and image visualization platform that is free for non-commercial use.

We enhance Deep Learning user productivity by presenting the user with a wizard which initially prompts the user to manually paint features of interest; minimal initial training data is required so the user is advised to spend no more than 5 minutes on this task, painting on the order of 50,000 labeled pixels. Dragonfly then trains four or more models (with no user parameters), and presents the user with multiple automatic segmentations. The user then chooses which segmentation performed the best, and is prompted to manually paint the areas of that prediction that are accurate, and to correct select areas that are inaccurate. At the end of this stage, much more training data is now available, e.g. for a full slice of a 1000 x 1000 pixel image, we now have 1 million pixels. At each stage, the training data are further inflated by established data augmentation techniques. This boostrapping approach allows the user to go from no training data to extensive training data with minimal user interaction, thereby greatly accelerating the otherwise labor-intensive practice of preparing training data.

The two stages of the Deep Learning life cycle are training, where an untrained neural network model is provided examples in order to learn the required image transformation, and inference, where a trained model is used to transform previously unseen images. Both stages benefit from graphical processing unit (GPU) acceleration, but training is the more compute intensive operation. We now make it possible for both stages to benefit from distributed execution on otherwise idle GPUs to achieve the same results in shorter time.

For a user who wishes to distribute her training or inference jobs, she may connect to a remote QueueManager, and add her tasks to the growing queue. The QueueManager maintains a first-in first-

out list of processing jobs. Other Dragonfly-enabled computers can, at any time, register with the QueueManager as available to take jobs, thereby taking on the role of ComputeNodes. Each of those nodes can abort a job at any time, and no progress is lost. The QueueManager will re-assign the remainder of the task to another available ComputeNode. All three roles (primary Dragonfly client, QueueMaster, and ComputeNode) run on both Windows and Linux. Communication between QueueManager and ComputeNodes is implemented with the huey python framework [2].

The role of ComputeNode can be filled by any Dragonfly-enabled computer with a CUDA-capable GPU. There is no restriction that the ComputeNodes must have the same hardware; this permits the QueueMaster to manage a heterogeneous compute network. In a typical research laboratory, any computer that is not being used can be enabled as a ComputeNode. This model is akin to the SETI@Home [3] solution of distributed computing. Alternatively, the user can configure his own HPC cluster to perform as a network of ComputeNodes. If the user requires greater computer performance, but doesn't have local resources, Dragonfly can be spawned elastically on Amazon cloud instances to serve as ComputeNodes at a price of less than $1.50 per node per hour, and those prices continue to fall.

When used for distributed training, a user is empowered to select multiple neural networks of different topology and different training parameters to train in parallel, which can more quickly lead to an optimal trained model. When ComputeNodes terminate a job before completion, no work is lost; the remainder of training epochs are captured as a new task on the QueueManager. When a user is performing inference, the entire operation is parallelized for a near linear speedup, sublinearity arising only due to model and data transfer between QueueMaster and ComputeNodes.

These enhancements take the arcane operation of Deep Learning and make it accessible in a user-friendly solution that is easy to adopt and deploy at large scale for large data pipelines. The enhanced productivity realized with these improvements help fully quantitative image analysis keep pace with the volume of images generated by emerging high-throughput imaging techniques.

References:

[1] R Makovetsky et al., Microscopy and Microanalysis **24 (S1)** (2018), p. 532.
[2] C Liefer, GitHub, https://github.com/coleifer/huey.
[3] E Korpela et al., Computing in Science & Engineering **3** (2001), p. 78.

https://doi.org/10.1017/S1431927619002745 Published online by Cambridge University Press