Addressing the Memory Bottleneck in AI Model-Training for Healthcare

Key Takeaways

- Near-terabyte memory footprint in 3D model training
- 3.4x speedup with Deep Neural Network Library optimizations

Executive Summary

Intel, Dell, and researchers at the University of Florida have collaborated to help data scientists optimize the analysis of healthcare data sets using artificial intelligence (AI). Healthcare workloads, particularly in medical imaging, require more memory usage than other AI workloads because they often use higher resolution 3D images.

In this white paper, we demonstrate how Intel-optimized TensorFlow* on a Dell EMC PowerEdge server equipped with 2nd Generation Intel Xeon Scalable Processors with large system memory allows for the training of memory-intensive AI/deep-learning models in a scale-up server configuration. We believe our work represents the first training of a deep neural network having large memory footprint (~ 1 TB) on a single-node server. We recommend this configuration to users who wish to develop large, state-of-the-art AI models but are currently limited by memory.

Revisions

Date	Description
1/9/2020	Initial release

Acknowledgements

This paper was produced by the following:

Name	
Bhavesh Patel	Dell EMC
David Ojika	PhD, University of Florida
G Anthony Reina	MD, Intel
Trent Boyer	Intel
Chad Martin	Intel
Prashant Shah	Intel

Table of Contents

Motivation

Healthcare data sets often consist of large, multi-dimensional modalities. Deep learning (DL) models developed from these data sets require both high accuracy and high confidence levels to be useful in clinical practice. Researchers employ advanced hardware and software to speed up this both data- and computation-intensive process.

"These models were only moderate size, and we require more GPU or CPU memory to be able to train larger models..."

"Our estimations are based on our current GPU hardware specifications. We hope that switching to a CPU based model (and using Intel-optimized TensorFlow) will make training large model more feasible."

- NEUROMOD / University de Montreal.

Medical image analytics, such as semantic segmentation, are particularly challenging because the model is trained to automatically classify individual voxels from large volumetric images [1]. The 3D (and sometimes 4D) nature of this data type demands increased memory capacity and processing power when training the model. Consequently, researchers resort to tricks, such as downsizing and tiling images, to cope with available system memory or adopting shallower neural network topologies to address the high processing requirement. Ultimately, most researchers choose a model based on the memory limitations of the hardware rather than based on the best possible model design.

A high-memory CPU-based server solution, such as the 2nd Generation Intel Xeon Scalable Processor, presents an attractive architecture for addressing the compute and memory requirement of 3D semantic segmentation algorithms, such as 3D U-Net model. With more than 1 TB of system memory available, the 2nd Generation Intel Xeon Scalable Processor allows researchers to develop large DL models that can be several orders of magnitude larger than those available on DL accelerators.

Multimodal Brain Tumor Analysis

Multimodal brain tumor analysis is an important diagnosis process in the healthcare industry. A brain tumor occurs when abnormal cells form within the brain. Gliomas are the most frequent primary brain tumors in adults, presumably originating from glial cells and infiltrating the surrounding tissues [2]. Current imaging techniques used in clinical studies are limited to basic assessments, indicating for example, the presence of gliomas, or limited to non-wholistic coverage of the scan as a result of the reliance on rudimentary measurement techniques [3]. By

replacing current assessments with highly accurate and reproducible measurements, AI and DL techniques can automatically analyze brain tumor scans, providing an enormous potential for improved diagnosis, treatment planning and patient follow-ups.

A typical MRI scans of the brain may contain 4D volumes with multimodal, multisite MRI data (FLAIR, T1w, T1gd, T2w). With appropriate training data sets, an AI-based brain tumor analysis solution should perform segmentation on the images, annotating regions of interest as necrotic/active tumor, oedema or benign.

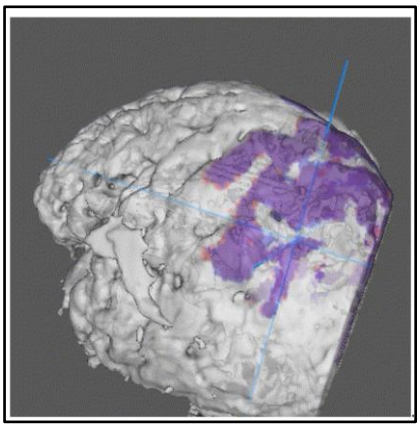

Figure 1. AI-based Gliomas segmentation.

Computing Challenges

While the high *processing* requirement of medical data analysis may be addressed with hardware accelerators, such as GPUs, addressing the *memory* requirement is not straightforward. As an example, a GPU accelerator has between 8 GB to 32 GB of memory. Although convolutional neural networks may only have several million trainable parameters, the actual memory footprint of these models is not due to solely those parameters. Instead, most of the memory footprint of these models comes from the activation (feature) maps in the model (Figure 2, green boxes). These activation maps—essentially copies of the original images—are a function of the size of the input to the network. Therefore, models that use large batch, high resolution, high dimensional image inputs often require more memory than the accelerator card can accommodate. As a simple example, a ResNet-50 topology that can train successfully on a 224x224x3 RGB input image may report an out of memory (OOM) error when training on 4096x2160x3 input images common to 4k video streams.

To compensate for the memory constraints of accelerator cards, researchers use the following "tricks":

- Image size: Images are often down sampled to a lower resolution
- Batch size: Batch sizes are often reduced to one or two images
- Tiling/Patching: Images are often subsampled into overlapping tiles/patches
- Model Complexity: Reductions in the number of feature maps and/or layers are often necessary
- Model Parallelism: Models may be distributed across several compute nodes in a parallel fashion

Although these tricks have been used to produce clinically-relevant models, we believe that researchers would not choose to use them if it were not for the memory limitations in hardware. In other words, these tricks were not created to obtain better models—they are instead necessary workarounds for hardware limitations. We believe that researchers would prefer to use the full resolution image without having to account for hyperparameters such as batch size, model complexity, or subsampling (tiling/patching). The large memory capacity of the 2nd Generation Intel Xeon Scalable Processor allows researchers this ability.

Experimental Data

The medical decathlon dataset [4] is a 3D semantic segmentation challenge with a broad range of medical imaging tasks including tumor and cancer diagnoses for various parts of the human body, including the liver, brain, lung, colon, and prostate. The images were generated either through a CT or an MRI scan at various universities and research centers from across the globe. Given this variety of data, the images present the opportunity for data scientists and machine learning practitioners to optimize AI algorithms for generalizability in medical imaging tasks with a primary focus on semantic segmentation. Thus, the most commonly used metric in segmentation tasks, Dice Similarity Coefficient (DSC) [5], along with Normalized Surface Distance (NSD) (distance between reconstructed surfaces) are used to assess different aspects of the performance of each task and region of interest. In this paper, we focus on the DSC (or simply, “dice coefficient”) of the Brain Tumor task from the BraTS dataset, which contains 750 4D MRI volumes: 484 for training and 266 for testing.

3D U-Net Model

Convolutional neural networks (CNNs) such as U-Net have been widely successfully in 2D segmentation in computer vision problems [6]. However, most medical data used in clinical practice consists of 3D volumes. Since only 2D slices can be displayed on a computer screen, annotating these large volumes with segmentation labels in a slice-by-slice manner is cumbersome and inefficient. 3D U-Net [7], based on U-Net architecture, performs volumetric segmentation by taking 3D volumes as input and processing them with corresponding 3D operations: 3D convolutions, 3D max-pooling, 3D up-sampling, etc. The resulting output is a trained model that reasonably generalizes well since the image slices contain mostly repetitive

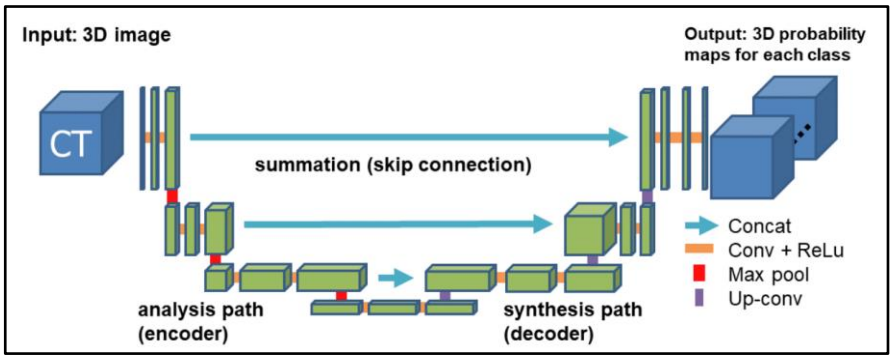

Figure 2. 3D U-Net architecture. Each box corresponds to a multi-channel feature map; the arrows denote different operations. [8]

structures with corresponding variation. In general, the 3D U-Net model is both computation- and memory-intensive.

Memory Profiling

Memory footprint is as important to deep-learning training as is raw processing throughput or Floating-Point Operations per Second (FLOPs), especially when dealing with volumetric data and large models such as 3D U-Net. Table 1. shows the breakdown of the memory requirement of the 3D U-Net model at the largest available image size (240x240x144 in the case of the BraTS dataset) using a kernel size of 3x3x3. As indicated, the estimated system memory requirement is a little less than 1 TB for a batch size of 16 MRI scans. On our development server equipped with only 192 GB of system memory (Table 2), it took only a couple of minutes after starting model training before the system ran out of memory and the whole experiment came to a stall.

Layer (type)	Batch size	Height	Width	Depth	# Feature Maps	# Trainable Weights	Feature Map Size	# Params	Memory usage (MB)
Input_Image (InputLayer)	16	240	240	144	4	0	530,841,600	530,841,600	2,123.37
conv1a (Conv3D)	16	240	240	144	32	3,488	4,246,732,800	4,246,736,288	16,986.95
conv1b (Conv3D)	16	240	240	144	64	55,360	8,493,465,600	8,493,520,960	33,974.08
pool1 (MaxPooling3D)	16	120	120	72	64	0	1,061,683,200	1,061,683,200	4,246.73
conv2a (Conv3D)	16	120	120	72	64	110,656	1,061,683,200	1,061,793,856	4,247.18
conv2b (Conv3D)	16	120	120	72	128	221,312	2,123,366,400	2,123,587,712	8,494.35
pool2 (MaxPooling3D)	16	60	60	36	128	0	265,420,800	265,420,800	1,061.68
conv3a (Conv3D)	16	60	60	36	128	442,496	265,420,800	265,863,296	1,063.45
dropout_1 (Dropout)	16	60	60	36	128	0	265,420,800	265,420,800	1,061.68
conv3b (Conv3D)	16	60	60	36	256	884,992	530,841,600	531,726,592	2,126.91
pool3 (MaxPooling3D)	16	30	30	18	256	0	66,355,200	66,355,200	265.42
conv4a (Conv3D)	16	30	30	18	256	1,769,728	66,355,200	68,124,928	272.50
dropout_2 (Dropout)	16	30	30	18	256	0	66,355,200	66,355,200	265.42
conv4b (Conv3D)	16	30	30	18	512	3,539,456	132,710,400	136,249,856	545.00
transConv4 (Conv3DTranspose)	16	60	60	36	512	2,097,664	1,061,683,200	1,063,780,864	4,255.12
concatenate_1 (Concatenate)	16	60	60	36	768	0	1,592,524,800	1,592,524,800	6,370.10
conv5a (Conv3D)	16	60	60	36	256	5,308,672	530,841,600	536,150,272	2,144.60
conv5b (Conv3D)	16	60	60	36	256	1,769,728	530,841,600	532,611,328	2,130.45
transConv5 (Conv3DTranspose)	16	120	120	72	256	524,544	4,246,732,800	4,247,257,344	16,989.03
concatenate_2 (Concatenate)	16	120	120	72	384	0	6,370,099,200	6,370,099,200	25,480.40
conv6a (Conv3D)	16	120	120	72	128	1,327,232	2,123,366,400	2,124,693,632	8,498.77
conv6b (Conv3D)	16	120	120	72	128	442,496	2,123,366,400	2,123,808,896	8,495.24
transConv6 (Conv3DTranspose)	16	240	240	144	128	131,200	16,986,931,200	16,987,062,400	67,948.25
concatenate_3 (Concatenate)	16	240	240	144	192	0	25,480,396,800	25,480,396,800	101,921.59
conv7a (Conv3D)	16	240	240	144	128	663,680	16,986,931,200	16,987,594,880	67,950.38
conv7b (Conv3D)	16	240	240	144	128	442,496	16,986,931,200	16,987,373,696	67,949.49
PredictionMask (Conv3D)	16	240	240	144	1	129	132,710,400	132,710,529	530.84

	# Trainable weights	Feature map size	Total float32 parameters	Total size (MB)	**Double for training**
Total	19,735,329	114,330,009,600	114,349,744,929	457,398.98	**914,797.96 MB**

Table 1. Memory requirement for training 3D U-Net.

Image size	Batch size	Training outcome	Server system memory	Server CPU family	Server tag
128x128x128	16	Fail	192 GB	1st Generation Intel Xeon Scalable Processor	*dev server*
144x144x144	8	Success	384 GB	1st Generation Intel Xeon Scalable Processor	*standard server*
240x240x144	16	-	1.5 TB	2nd Generation Intel Xeon Scalable Processor	*memory-rich server*

Table 2. Provisioning training infrastructure for 3D U-Net. We used random pixel values as input tensors. Our development server failed when executing just the 3D convolution-kernel part of the full 3D U-Net architecture.

We overcame this memory bottleneck on our development server by reducing the training batch size from 16 down to 2, while reducing the image size to reasonably smaller sized dimensions instead of the full-scale image feature map (240x240x144). Of course, this has an impact on the model accuracy and convergence time. Next, we upgraded our server's system memory to its maximum supported memory capacity (384 GB), increased the image size of the dataset to about one-half, but reduced the batch size by half. In this scenario, the training job completed successfully. In the next section, we will go over the details of the training infrastructure — with a "memory-rich" server—using the full-scale BraTS images.

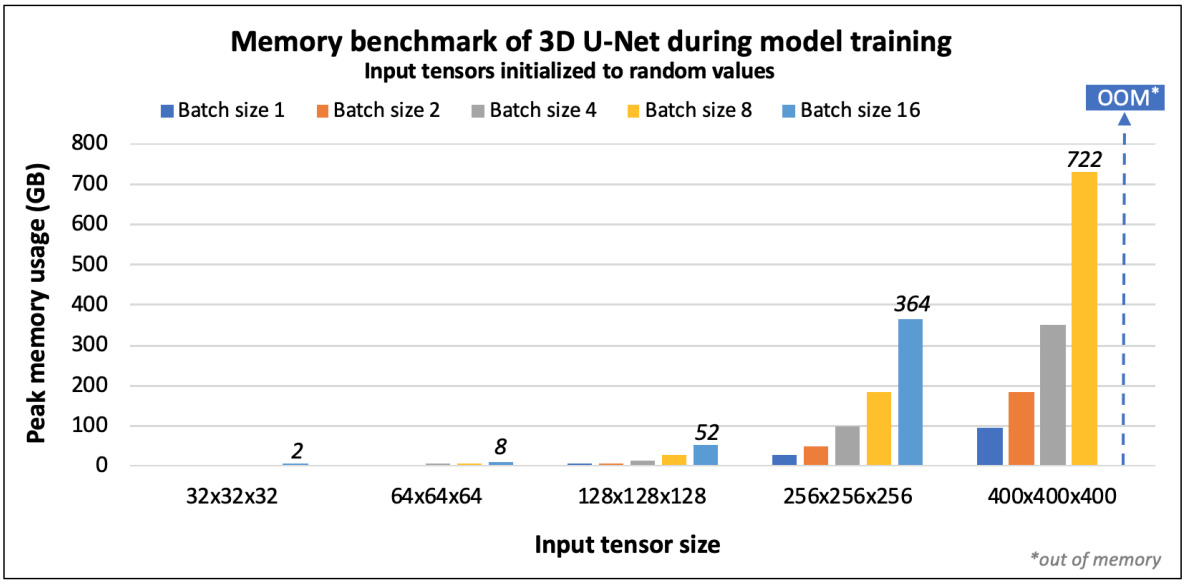

*Figure 3. Benchmarking the memory usage of 3D U-Net model-training over various input tensors sizes on an Intel Xeon Scalable Processor-based server with **1.5 TB system***

Training 3D U-Net on a Large-Memory System

A single-node server with large memory has the potential to reduce organization's total cost of ownership (TCO), while addressing the memory bottleneck involved with training large models with complex datasets. Using a 4-socket 2nd Generation Intel Xeon Scalable Processor system on a Dell EMC PowerEdge server equipped with 1.5 TB of system memory (Figure 4), we trained the 3D U-Net model with the BraTS dataset (using only the "FLAIR" channel) without the need for

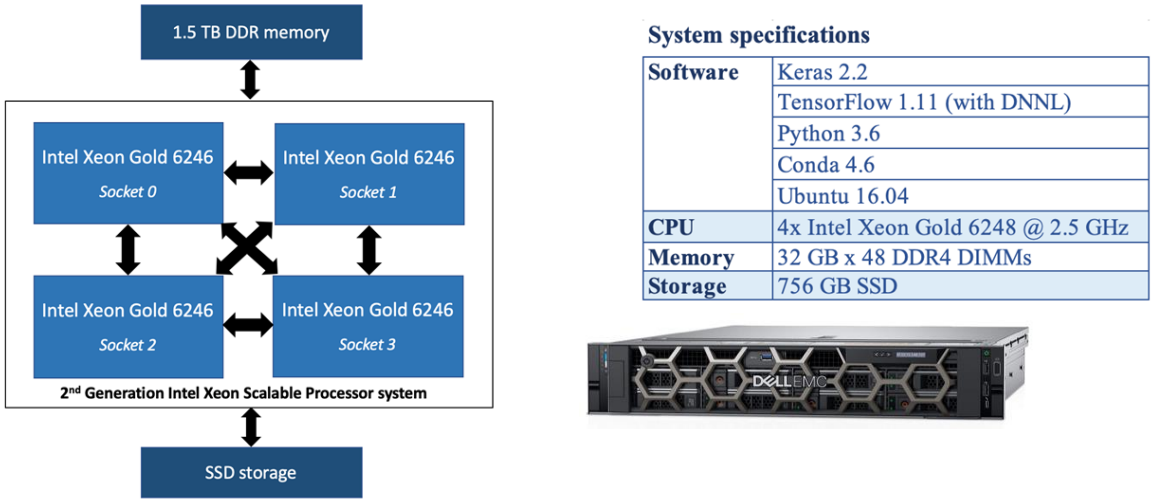

Figure 4. Training infrastructure for 3D U-Net model with a 4-socket 2nd Generation Intel Xeon Scalable Processor system on a 2U Dell EMC PowerEdge R840 server.

scaling down the data nor tiling images to fit in memory. We used Intel-optimized TensorFlow - available as an Anaconda library [9]

- and Conda as the Python virtual execution environment. The Intel-optimized TensorFlow distribution incorporates Deep Neural Network Library (DNNL) [10] (formerly MKL-DNN), allowing us to leverage the processors' underlying hardware features, including high CPU core count (80 cores), AVX-512 for floating-point operations, and integrated memory controllers supporting 1TB-per-socket system memory, to speed up the training process.

Using this system configuration, we achieved, within 25 training iterations (epochs), close to state-of-the-art performance: 0.997 accuracy, 0.125 loss and 0.82 dice coefficient. We also profiled the memory footprint of the training task, comparing the results (Figure 5) with our theoretical calculations from Table 1 and found our estimations to be accurate for our chosen hyperparameters (batch, feature-map, and image sizes). Meanwhile, the training speed (TS) for a single step (involving forward pass and backward pass of a single 3D scan) per training epoch

was 30 seconds per image, a 3.4x speedup (Figure 6) compared to stock TensorFlow (without DNNL) at the same training batch size of 16.

Figure 7 depicts the prediction performance of the trained model. As observed, the segmentation mask from the model predictions closely match the ground truth mask. Using Table 1 as a reference, along with the TS and epoch count, machine learning practitioners can "plug in" their specific training data and hyperparameters to estimate both the required system memory and task completion time when training their own deep learning models on Intel architecture.

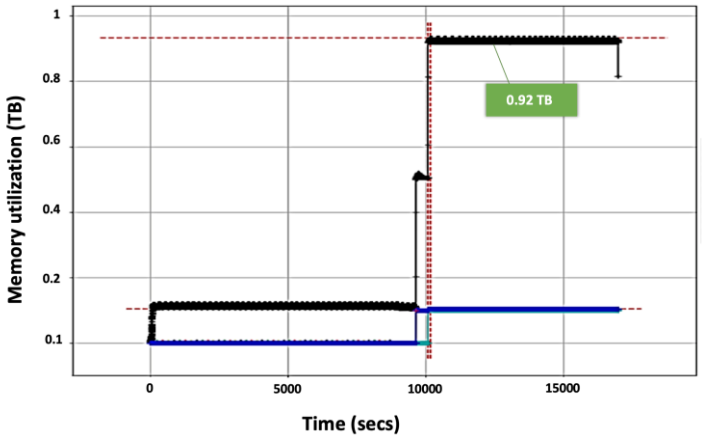

Figure 5. 3D U-Net memory footprint shows correlation with our theoretical calculations from Table 1.

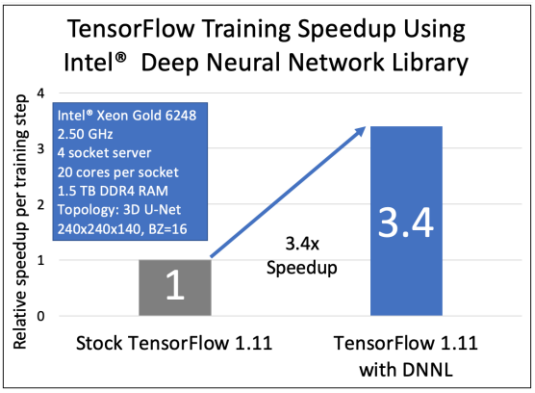

Figure 6. TensorFlow with Deep Neural Network Library (DNNL) enabled achieves increased performance versus stock TensorFlow (without DNNL).

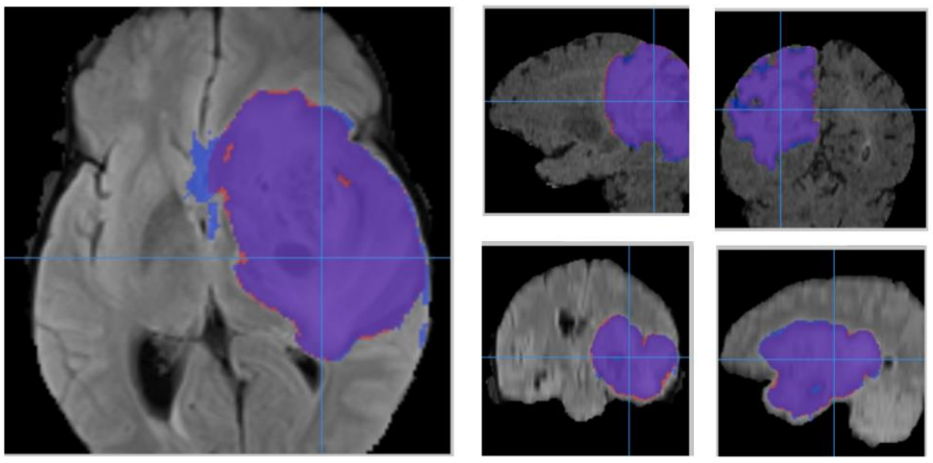

Figure 7. Prediction performance of the trained model, showing a slice of the brain from different views. The red overlay is the prediction from the model and the blue overlay is the ground truth mask. Any purple voxels are true positives.

Conclusions

In this white paper, we presented the multimodal brain tumor analysis for medical diagnosis, highlighted the computing challenges, and presented the 3D U-Net model for the task of volumetric image segmentation. We pre-calculated the memory requirement of the model and analyzed 3 different server configurations with varying memory capacity: from a “dev server” with 192 GB of memory to a “memory-rich” server with over 1 TB of memory. With the memory-rich sever, we trained the 3D U-Net model using the BraTS dataset (a medical segmentation benchmark) and achieved close to state-of-the-art accuracy of 0.997 and dice coefficient of 0.83. The maximum memory utilization of the model during training also corresponds to our pre-calculated memory requirement, suggesting the generalizability of our approach to other memory-bound deep learning algorithms.

To the best of our knowledge, the results presented in this paper represent the first milestone in training a deep neural network having large memory footprint (close to 1 TB) on a single-node server without hardware accelerators like GPUs. Further, by enabling Deep Neural Network Library (DNNL) optimizations, we achieved a speedup of 3.4x per training step compared to stock TensorFlow. By replicating the single-node, memory-rich configuration described in this paper into a multi-node CPU cluster setup, we can expect to see greatly enhanced training performance of the 3D U-Net model as well as that of other complex 3D models and data sets, potentially reducing organizations TCO [13].

Acknowledgments

Center for Space High-Performance and Resilient Computing (SHREC), University of Florida
University de Montreal
NEUROMOD
Dell EMC
Intel

References

[1] Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. 2019. Understanding Deep Learning Techniques for Image Segmentation. ACM Comput. Surv. 52, 4, Article 73 (August 2019), 35 pages.

[2] Holland, E n.d., 'Progenitor cells and glioma formation', Current Opinion in Neurology, vol. 14, no. 6, pp. 683–688.

[3] B. H. Menze *et al.*, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," in *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993-2024, Oct. 2015.

[4] BRATS dataset https://www.med.upenn.edu/sbia/brats2018.html

[5] Maier-Hein, Lena, Eisenmann, Matthias, Reinke, Annika, Onogur, Sinan, Stankovic, Marko, Scholz, Patrick, & Full, Peter M. (2018). Is the winner really the best? A critical analysis of common research practice in biomedical image analysis competitions (Version 1.0.0) Zenodo.

[6] Olaf Ronneberger, Philipp Fischer & Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol.9351, 234--241, 2015

[7] Ciçek Ö., Abdulkadir A., Lienkamp S.S., Brox T., Ronneberger O. (2016) 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In: Ourselin S., Joskowicz L., Sabuncu M., Unal G., Wells W. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016. MICCAI 2016. Lecture Notes in Computer Science, vol 9901. Springer, Cham

[8] H. R. Roth, C. Shen, H. Oda, M. Oda, Y. Hayashi, K. Misawa, and K. Mori, "Deep learning and its application to medical image segmentation," Medical Imaging Technology, vol. 36, no. 2, pp. 63– 71, 2018.

[9] Intel Optimization for TensorFlow. https://software.intel.com/en-us/articles/intel-optimization-for-tensorflow-installation-guide

[10] Deep Neural Network Library. https://intel.github.io/mkl-dnn

[11] Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J.S., Freymann, J.B., Farahani, K., & Davatzikos, C. (2017). Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Scientific data*.

[12] Spyridon Bakas, Hamed Akbari, Aristeidis Sotiras, Michel Bilello, Martin Rozycki, Justin Kirby, John Freymann, Keyvan Farahani, and Christos Davatzikos. (2017) Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection. The Cancer Imaging Archive.

[13] Using Intel® Xeon® for Multi-node Scaling of TensorFlow with Horovod. https://www.intel.ai/using-intel-xeon-for-multi-node-scaling-of-tensorflow-with-horovod/#gs.mqqqpc

Appendix: Reproducibility

Software	Data	Model
Keras 2.2 TensorFlow 1.11 DNNL Python 3.6 Anaconda 3 Conda 4.6 Ubuntu 16.04	Dataset name: BRATS Tensor image size: 4D Train, val, test images: 406, 32, 46 Dataset license: CC-BY-SA 4.0 Release: 2.0 04/05/2018 Dataset source: https://www.med.upenn.edu/sbia/brats2017.html	Architecture: 3D U-Net Input format: Channels last Params: 5,650,801 Trainable params: 5,647,857 Non-trainable params: 2,944 Code repository: https://github.com/IntelAI/unet

Hardware

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                80
On-line CPU(s) list:   0-79
Thread(s) per core:    1
Core(s) per socket:    20
Socket(s):             4
NUMA node(s):          4
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
Stepping:              6
CPU MHz:               2494.155
BogoMIPS:              4989.86
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              1024K
L3 cache:              28160K
NUMA node0 CPU(s):     0,4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68,72,76
NUMA node1 CPU(s):     1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61,65,69,73,77
NUMA node2 CPU(s):     2,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,66,70,74,78
NUMA node3 CPU(s):     3,7,11,15,19,23,27,31,35,39,43,47,51,55,59,63,67,71,75,79
Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology
nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16
xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm
abm 3dnowprefetch epb intel_pt tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep
bmi2 erms invpcid rtm cqm mpx avx512f rdseed adx smap clflushopt clwb avx512cd xsaveopt xsavec xgetbv1
cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts
```

FTC Disclaimer: For Performance Claims

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. *Configurations:* Tested by Dell EMC as of 12/17/2019. 4 socket Intel® Xeon® Gold 6248 Processor, 20 cores per socket HT OFF Turbo OFF Total Memory 1.5TB GB (DDR4, 48 slots/ 32GB), NUMA Not Enabled, KMP_AFFINITY="granularity=thread,compact", OMP_NUM_THREADS=80, KMP_BLOCKTIME=1, Number of intraop threads=80, Number of interop threads=1, Ubuntu 16.04, Deep Learning Framework: TensorFlow 1.11 with Intel® Deep Neural Network Library (DNNL/MKL-DNN), 3D U-Net:https://github.com/IntelAI/unet, BS=16, Medical Decathlon (BraTS, http://medicaldecathlon.com/) + synthetic data, Datatype: FP32

FTC Optimization Notice:

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Security Disclaimer

Performance results are based on testing by Dell EMC as of 12/17/2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product or component can be absolutely secure.

Technology Disclaimer

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].