HPC FOR AI TRAINING & INFERENCE

October 9, 2020

G Anthony Reina, M.D.
Chief AI Architect for Health & Life Sciences, Intel

Ravi Panchumarthy, Ph.D
Machine Learning Engineer, Intel.

NOTICES AND DISCLAIMERS

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 product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

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 http://www.intel.com/benchmarks .

Intel® Advanced Vector Extensions (Intel® AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

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.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Workshop on Deep Learning with Intel Optimized Software

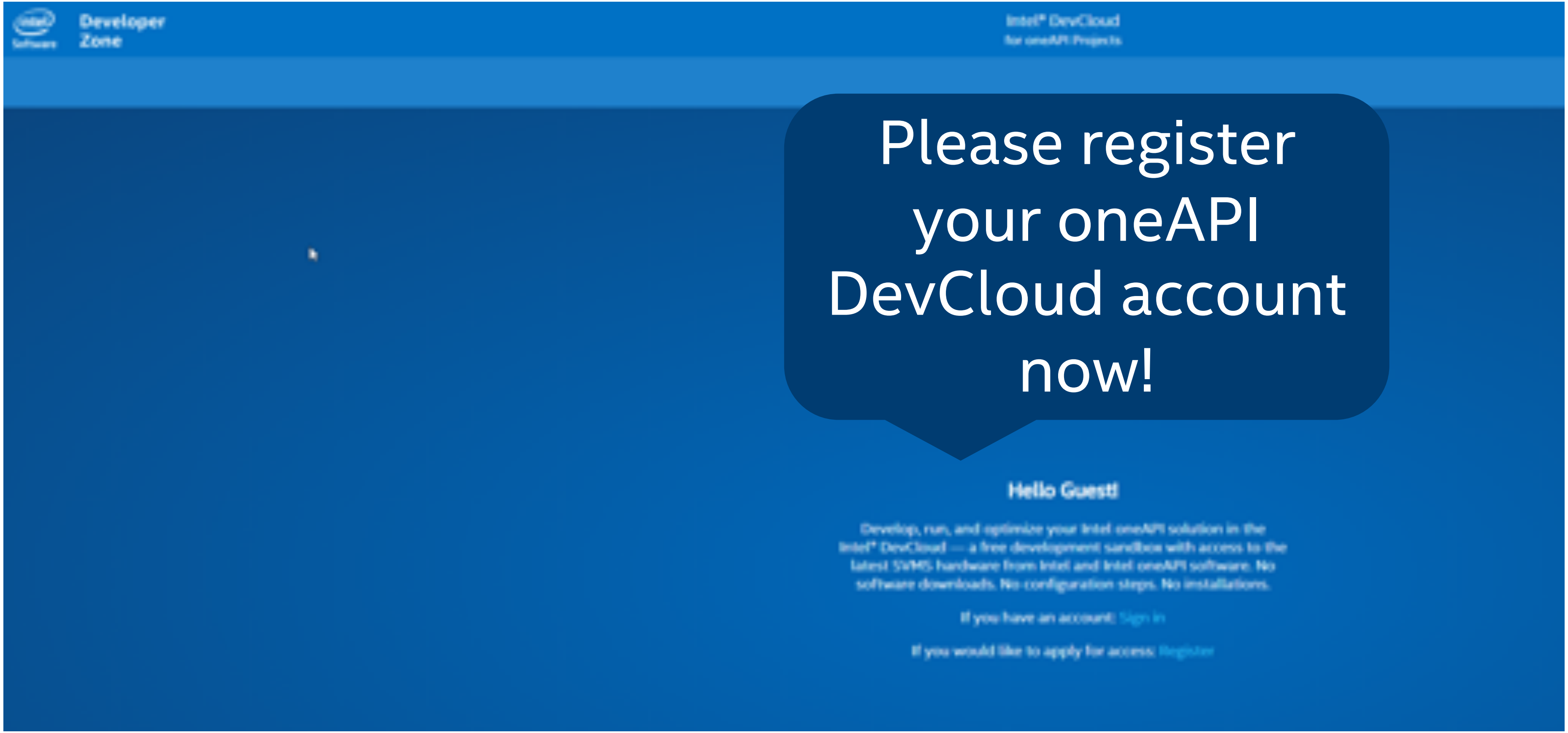
Intel® OneAPI DevCloud

Workshop 2: Deep Learning Module

13:30 - 14:45 Deep Learning – Optimized training instances

- Performance Optimized Deep Learning Frameworks solutions from Intel®
    - TensorFlow and PyTorch optimizations for CPU via Intel® DNNL
- Distributed (data parallel) deep learning training with Horovod on a CPU cluster
- Large memory (100 GB to 1.5 TB) training with TensorFlow
- Federated Learning

14:45 – 15:15 Hands On Session



Please register your oneAPI DevCloud account now!

https://devcloud.intel.com/oneapi/get-started/

INTEL-OPTIMIZED DEEP LEARNING TRAINING

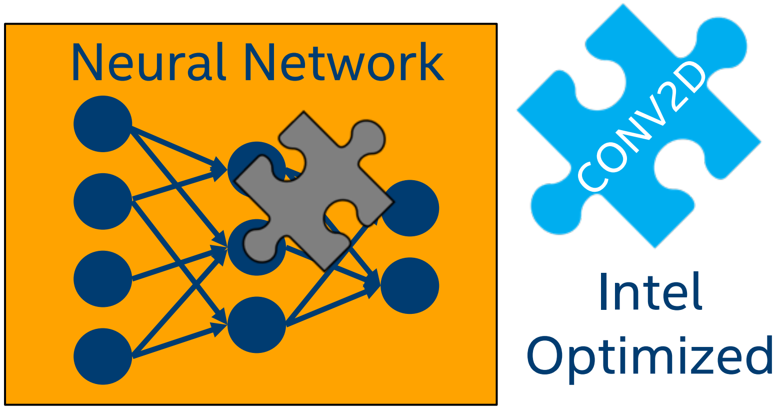
INTEL® DNNL

Intel®'s Open-Source Deep Neural Networks Library

For developers of deep learning frameworks featuring optimized performance on Intel hardware

Distribution Details

- Open Source
- Apache 2.0 License
- Common DNN APIs across all Intel hardware.
- Rapid release cycles, iterated with the DL community, to best support industry framework integration.
- Highly vectorized & threaded for maximal performance, based on the popular Intel® MKL library.



Examples: Direct 1D/2D/3D Convolution | LSTM / GRU | Rectified linear unit activation (ReLU) | Maximum pooling | Inner product

All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

DEEP LEARNING FRAMEWORKS

Popular DL Frameworks are now optimized for CPU

FRAMEWORKS OPTIMIZED BY INTEL

- TensorFlow*
- mxnet*
- PYTORCH
- BigDL FOR Apache Spark*

See installation guides at ai.intel.com/framework-optimizations/

TensorFlow: conda install –c anaconda tensorflow

PyTorch: conda install pytorch-cpu torchvision-cpu -c pytorch

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MlLib on Spark, Mahout)

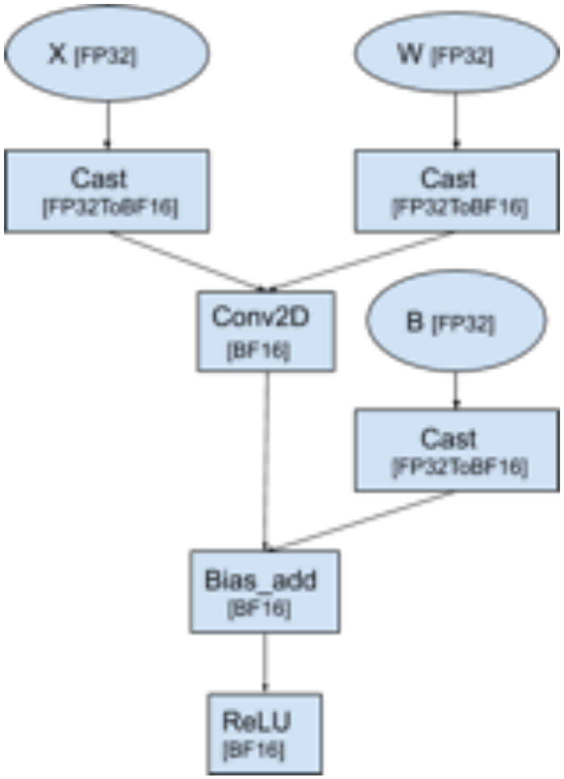
NO CHANGES TO TENSORFLOW / PYTORCH

```
from tensorflow import keras as K

inputs = K.layers.Input((32, 32, 3), name="Image")

cnn_layer1 = K.layers.Conv2D(filters=16,
                             kernel_size=(3,3),
                             activation="relu")(inputs)

cnn_layer2 = K.layers.Conv2D(filters=16,
                             kernel_size=(3,3),
                             activation="relu")(cnn_layer1)

flatten = K.layers.Flatten()(cnn_layer2)

dense1 = K.layers.Dense(units=128, activation="relu")(flatten)

prediction = K.layers.Dense(units=10, activation="softmax")(dense1)

model = K.models.Model(inputs=[inputs], outputs=[prediction])

model.compile(optimizer="adam", loss="binary_crossentropy")
```

BFloat16 – 3rd Generation Intel® Xeon

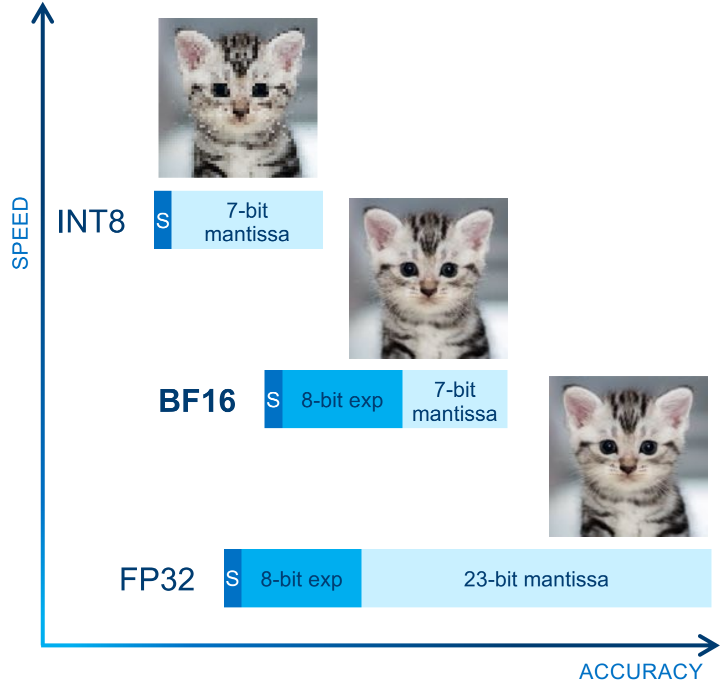
BFloat16 – 3rd Generation Intel® Xeon

```python
import tensorflow as tf
from tensorflow.core.protobuf import rewriter_config_pb2

tf.compat.v1.disable_eager_execution()

def conv2d(x, w, b, strides=1):
    # Conv2D wrapper, with bias and relu activation
    x = tf.nn.conv2d(x, w, strides=[1, strides, strides, 1], padding='SAME')
    x = tf.nn.bias_add(x, b)
    return tf.nn.relu(x)

X = tf.Variable(tf.compat.v1.random_normal([784]))
W = tf.Variable(tf.compat.v1.random_normal([5, 5, 1, 32]))
B = tf.Variable(tf.compat.v1.random_normal([32]))
x = tf.reshape(X, shape=[-1, 28, 28, 1])

graph_options=tf.compat.v1.GraphOptions(
        rewrite_options=rewriter_config_pb2.RewriterConfig(
            auto_mixed_precision_mkl=rewriter_config_pb2.RewriterConfig.ON))

with tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(
        graph_options=graph_options)) as sess:
    sess.run(tf.compat.v1.global_variables_initializer())
    sess.run([conv2d(x, W, B)])
```



Intel Deep Learning Boost, enhanced with bfloat16

The cutting edge of AI innovation



Up to

1.93X

HIGHER AI PERFORMANCE WITH INTEL DL BOOST[1]

For image classification vs. Intel Xeon Platinum 8280 processors

Similar accuracy
BF16 vs. FP32

Improved memory utilization
16 bits vs. 32 bits

Increased performance
2 BF16 processes / cycle vs. 1 FP32

Optimized libraries & frameworks

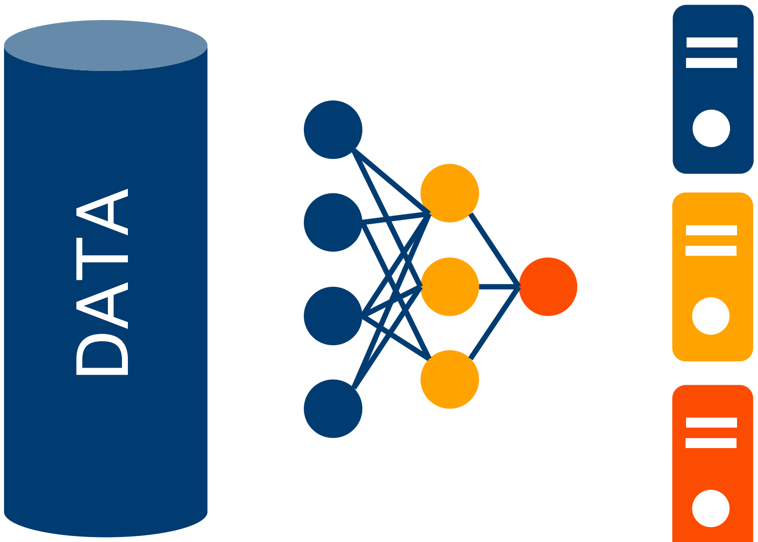
oneAPI, OpenVINO, ONNX RUNTIME, PYTORCH, TensorFlow

1 For more complete information about performance and benchmark results, visit www.intel.com/3rd-gen-xeon-configs. See configuration details on slides 63-67.
For more information regarding performance and optimization choices in Intel software products., please visit https://software.intel.com/en-us/articles/optimization-notice.

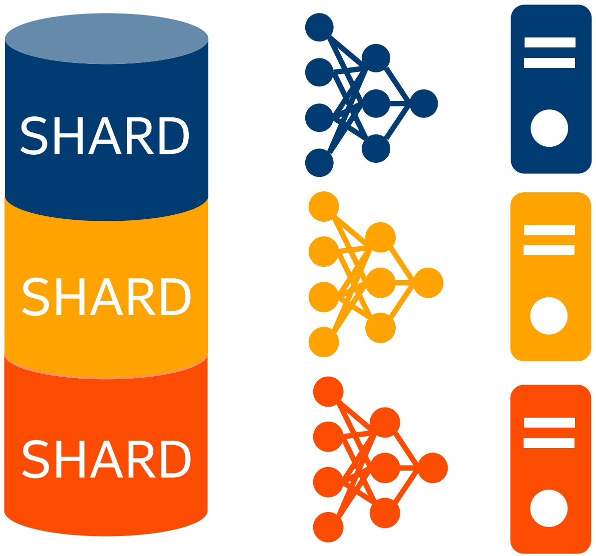
DISTRIBUTED DEEP LEARNING TRAINING

DISTRIBUTED TRAINING

MODEL PARALLELISM

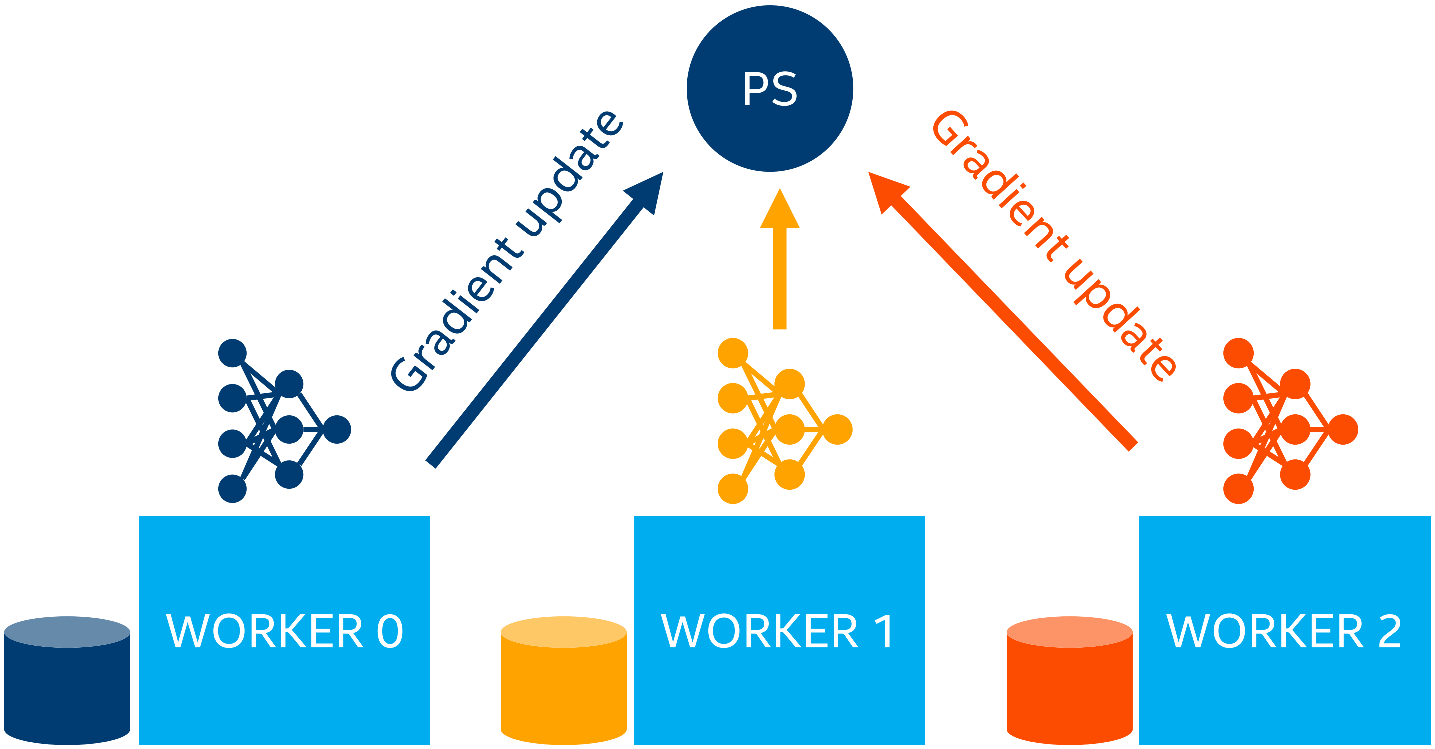


DATA PARALLELISM



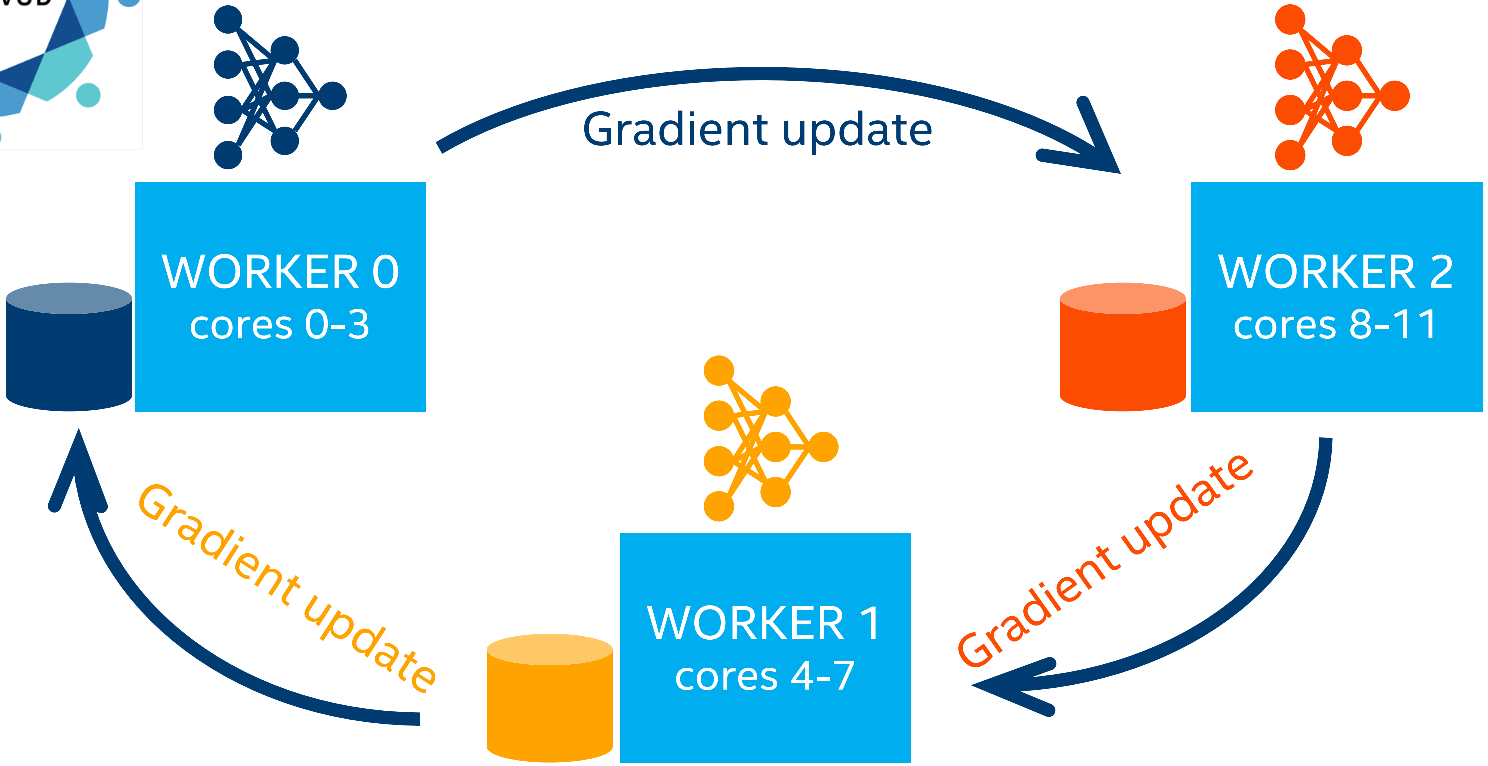
A worker could be a CPU, a GPU, a socket, or multiple cores.

PARAMETER SERVER



Tree using gRPC calls

HOROVOD



https://arxiv.org/abs/1802.05799v3

Ring All-Reduce using MPI

MESSAGE PASSING INTERFACE (MPI)

```
$ mpirun –H 192.168.1.100,192.168.1.105    hostname
aipg-infra-07.intel.com
aipg-infra-09.intel.com
```

```
$ mpirun –H host1,host2,host3    python hello.py
Hello World!
Hello World!
Hello World!
```

CHANGES TO TENSORFLOW

1

```
import tensorflow as tf
import horovod.tensorflow as hvd
```

2

```
hvd.init()
```

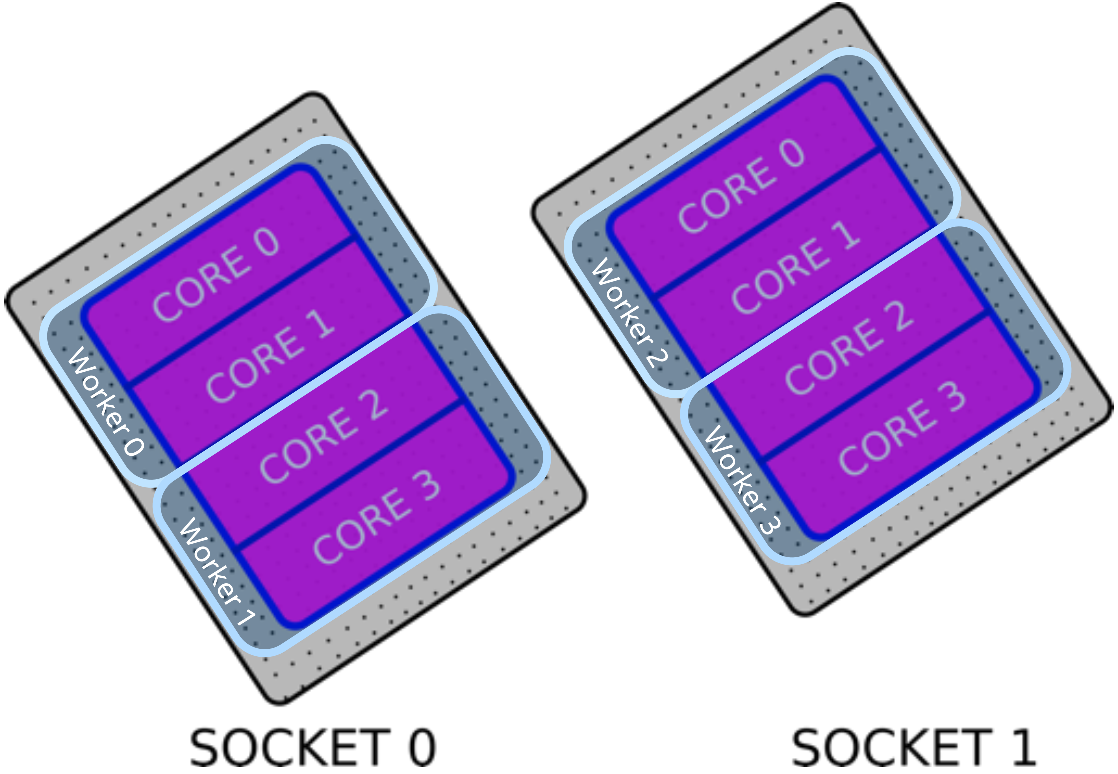
3

```
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())
opt = hvd.DistributedOptimizer(opt)
```

4

```
hooks = [hvd.BroadcastGlobalVariablesHook(0)]
```

SOCKETS & CORES



SOCKET

Receptacle on the motherboard for one physically packaged processor.

CORE

A complete private set of registers, execution units, and queues to execute a program.

`lscpu`

MULTIPLE WORKERS PER CPU

```
$ mpirun
-H hostA,hostB,hostC
-np 6
--map-by ppr:1:socket:pe=2
--oversubscribe
--report-bindings
python train_model.py
```

OpenMPI

MULTIPLE WORKERS PER CPU

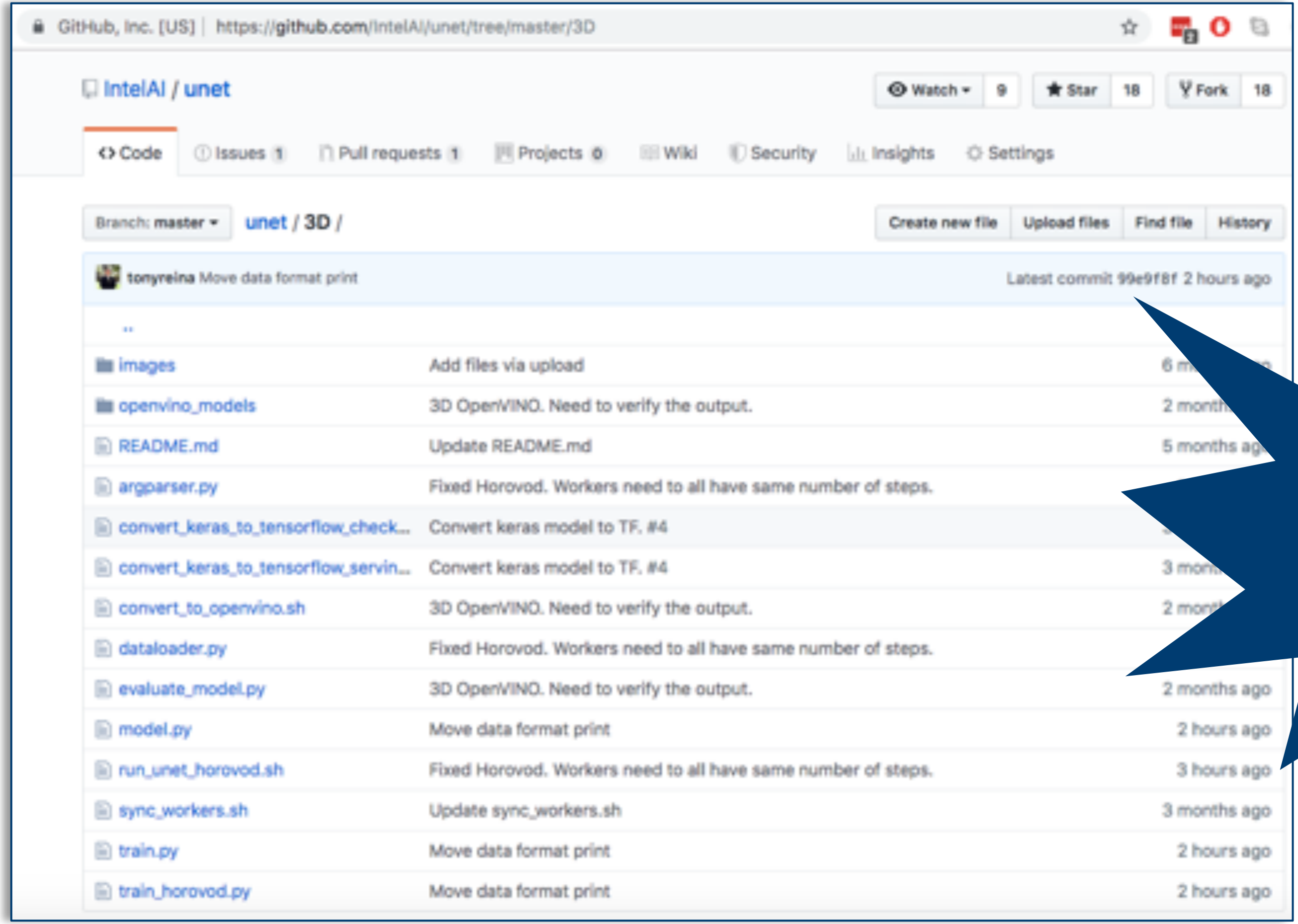
```
$ mpirun
-H hostA, hostB, hostC
-n 6
-ppn 2
-print-rank-map
-genv I_MPI_PIN_DOMAIN=socket
-genv OMP_NUM_THREADS=24
-genv OMP_PROC_BIND=true
-genv KMP_BLOCKTIME=1
python train_model.py
```

Intel MPI

MULTIPLE WORKERS PER CPU

		SOCKET 0	SOCKET 1
R0	hostA	[BB/BB/../..]	[../../../..]
R1	hostA	[../../../..]	[BB/BB/../..]
R2	hostB	[BB/BB/../..]	[../../../..]
R3	hostB	[../../../..]	[BB/BB/../..]
R4	hostC	[BB/BB/../..]	[../../../..]
R5	hostC	[../../../..]	[BB/BB/../..]

```
mpirun -H hostA,hostB,hostC   -np 6  --map-by ppr:1:socket:pe=2 ...
```



Try it on GitHub

github.com/IntelAI/unet

BKC/BKM FOR HPC AI

WHITE PAPER

intel

Best Practices for Scaling Deep Learning Training and Inference with TensorFlow* On Intel® Xeon® Processor-Based HPC Infrastructures

Version:	1.1
Date of Issue:	January 2019
Prepared By:	Aishwarya Bhandare¶, Deepthi Karkada¶, Kushal Datta¶, Anupama Kurpad§, Vamsi Sripathi¶, Sun Choi¶, Vikram Saletore¶
	§Connectivity Group & ¶AI Products Group, Data Center Group
	Customer Solutions Technical Enabling, Intel Corporation

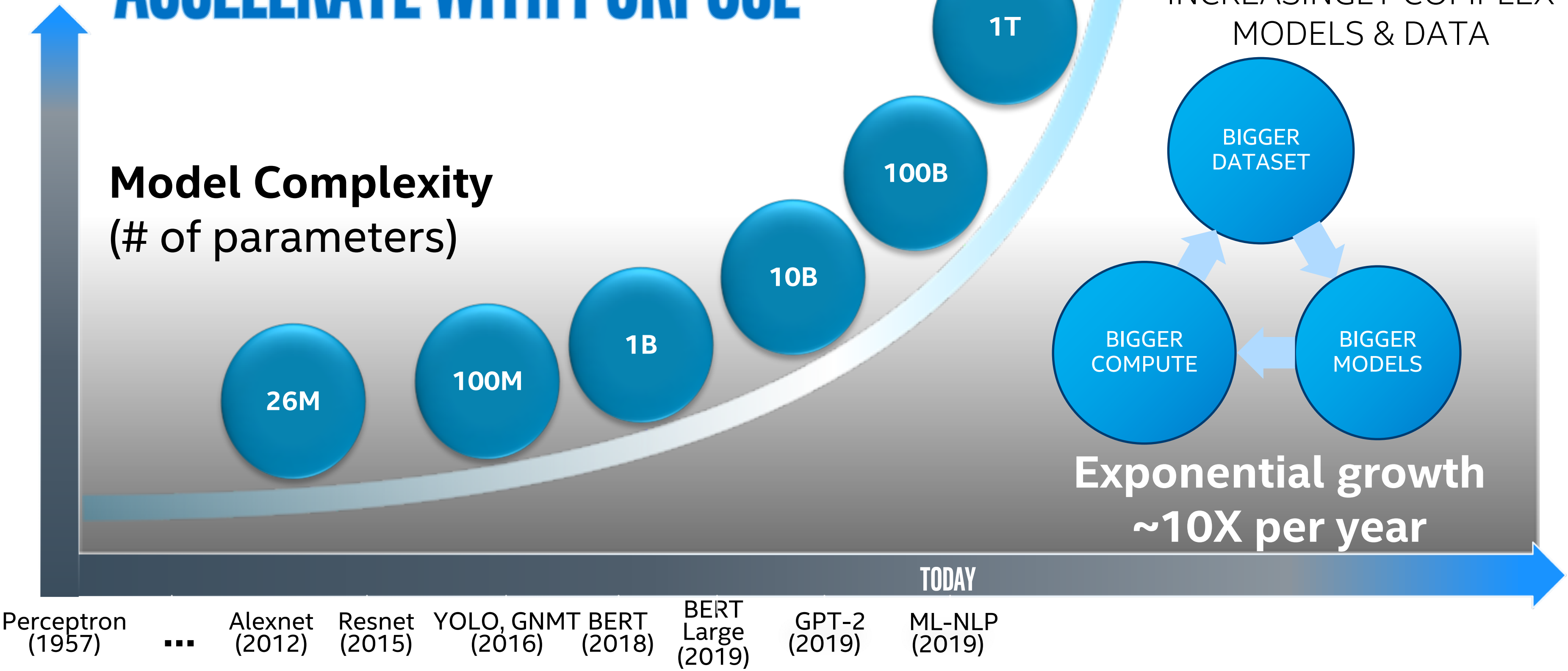
- Docker
- SLURM
- Singularity
- NFS
- Lustre

LARGE MEMORY ADVANTAGES OF CPUS FOR DEEP LEARNING

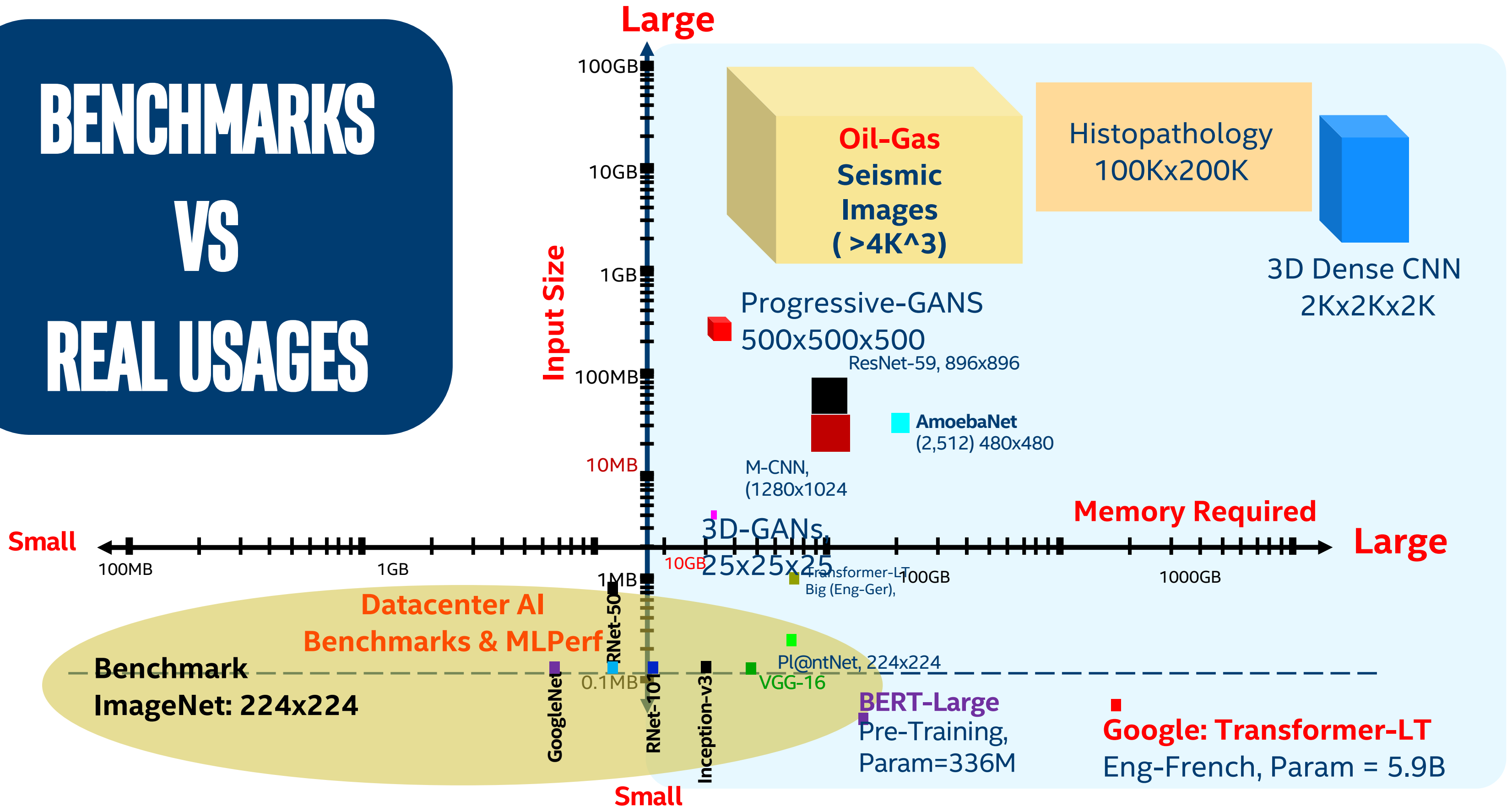
AS MODEL COMPLEXITY GROWS, ACCELERATE WITH PURPOSE

NEXT-GEN DL

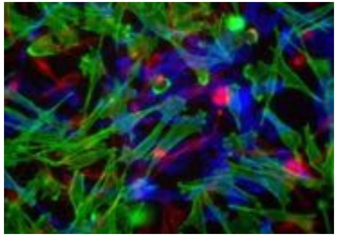
INCREASINGLY COMPLEX MODELS & DATA



BENCHMARKS VS REAL USAGES



DRUG DISCOVERY



224x224x3

1024x1280x3

Real AI workloads require large memory to train

HPC POC: IMAGE CLASSIFICATION

DELL EMC

JOINT COLLABORATION WITH INTEL AND SURFSARA

RESULT

DELL EMC SURF SARA intel

Training time reduced to 11 mins while increasing the accuracy across 10 categories relative to the existing DenseNet-121 model

intel XEON GOLD inside TensorFlow *

Customer: Dell EMC, a multi-national systems and solutions company located in Round Rock, TX

Challenge: Train a chest X-ray model that delivers highly-efficient scaling performance on Intel® Xeon® processor nodes, while also delivering higher accuracy than the existing ChexNet model

Solution: 256-node cluster consisting of Dell EMC* PowerEdge C6420 with dual Intel® Xeon® Gold 6148 processor, Intel® Omni-Path fabric, and ResNet-50 topology. ResNet50 tests performed with TensorFlow* and Horovod*.

*Other names and brands may be claimed as the property of others.

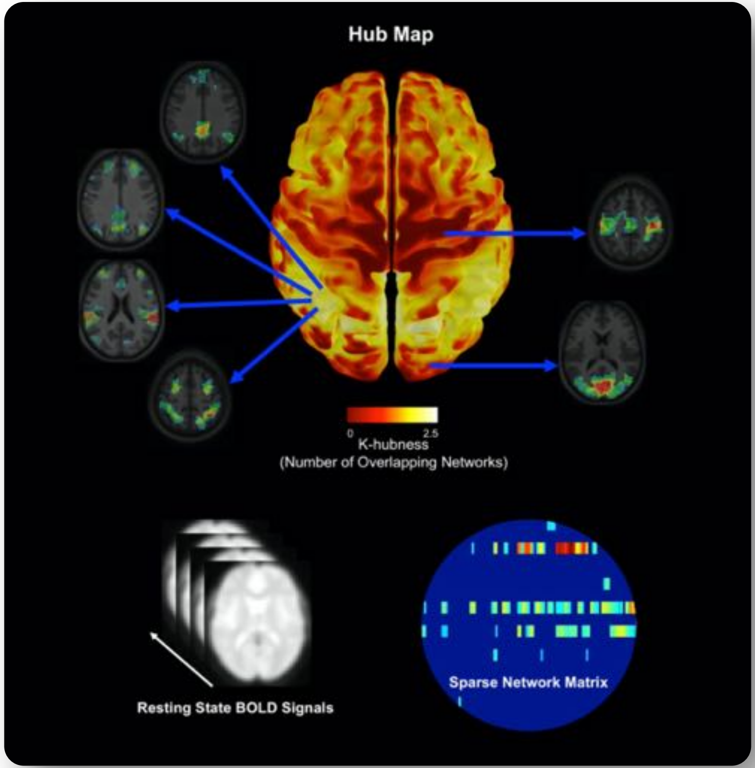
[1] Compute Nodes: 2 sockets Intel® Xeon® Gold 6148F processor with 20 cores each @ 2.40GHz for a total of 40 cores per node, 2 Threads per core, L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K, 96 GB of DDR4, Intel® Omni-Path Host Fabric Interface, dualrail. Software: Intel® MPI Library 2017 Update 4Intel® MPI Library 2019 Technical Preview OFI 1.5.0PSM2 w/ Multi-EP, 10 Gbit Ethernet, 200 GB local SSD, Red Hat* Enterprise Linux 6.7. TensorFlow* 1.6: Built & Installed from source: https://www.tensorflow.org/install/install_sources ResNet-50 Model: Topology specs from https://github.com/tensorflow/tpu/tree/master/models/official/resnet. DenseNet-121Model: Topology specs from https://github.com/liuzhuang13/DenseNet. Convergence & Performance Model: https://surfdrive.surf.nl/files/index.php/s/xrEFLPvo7IDRARs. Dataset: ImageNet2012-1K: http://www.image-net.org/challenges/LSVRC/2012 /. ChexNet*: https://stanfordmlgroup.github.io/projects/chexnet/. Performance measured with: OMP_NUM_THREADS=24 HOROVOD_FUSION_THRESHOLD=134217728 export I_MPI_FABRICS=tmi, export I_MPI_TMI_PROVIDER=psm2 \ mpirun -np 512 -ppn 2 python resnet_main.py --train_batch_size 8192 --train_steps 14075 --num_intra_threads 24 --num_inter_threads 2 -- mkl=True --data_dir=/scratch/04611/valeriuc/tf-1.6/tpu_rec/train --model_dir model_batch_8k_90ep --use_tpu=False --kmp_blocktime 1.

CHALLENGE

How to "decode" the brain using neural networks on fMRI images. In other words, take a series of fMRI scans while patients are performing prescribed action then decode those tasks using just the fMRI images.

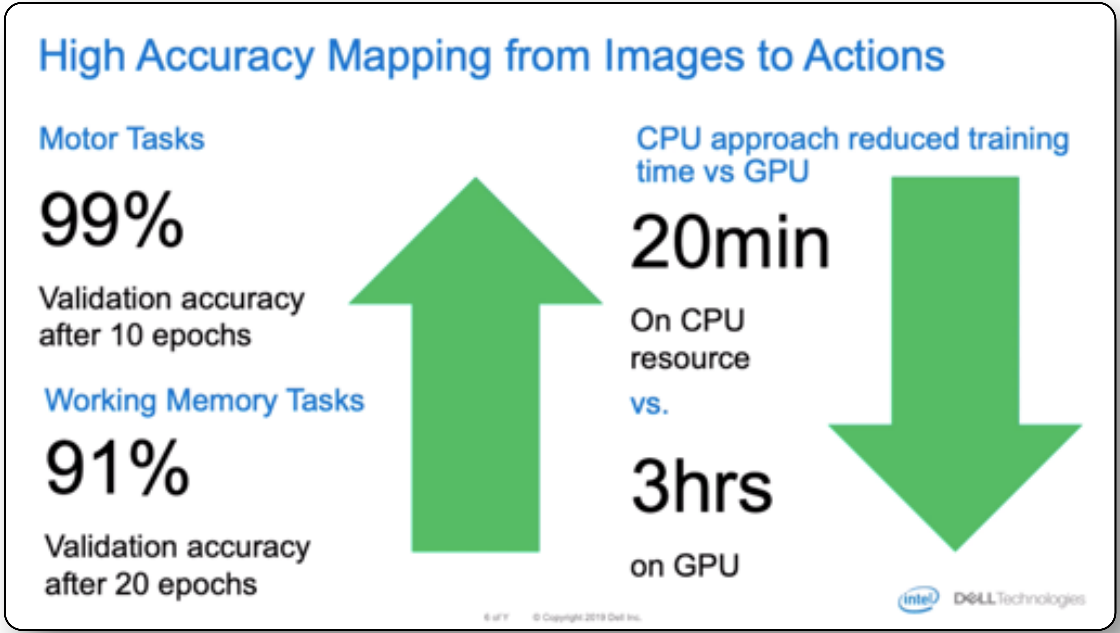
SOLUTION

Training the neural networks on a multinode CPU system rather than using GPUs for training. The compute was done using Intel® Xeon® Gold 6248 packaged in the 2U Dell PowerEdge C6420 dense compute platform, using the Intel® optimized TensorFlow version 1.11 with Intel® deep neural network library.



“If you want to build a better neural network, there is no better model than the human brain. In this project, McGill University was running into bottlenecks using neural networks to reverse-map fMRI images. The team from the Dell EMC HPC and AI Innovation Lab was able to tune the code to run solely on Intel Xeon-Scalable processors, rather than porting to the university’s scarce GPU accelerators.”

– Luke Wilson, AI research Lead, Dell HPC and AI Innovation Lab at Dell EMC



https://insidehpc.com/2019/11/slidecast-dell-emc-using-neural-networks-to-read-minds/

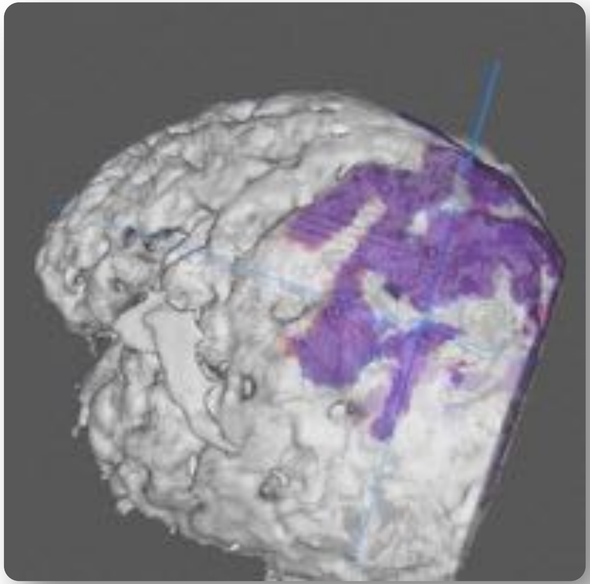
CHALLENGE

Medical imaging workloads require more memory usage than other AI workloads because they often use higher-resolution 3D images. A scalable, large memory system is needed for training of deep learning models.

SOLUTION

Training the neural networks was effected on multinode 4-socket Dell R840 servers, each with 1.5 TB of RAM and equipped with the Intel® Xeon® Gold 6248 processor and using the Intel® optimized TensorFlow version 1.11 with Intel® deep neural network library.

Using the above system configuration, within 25 training iterations (epochs), close to state-of-the-art performance: 0.997 accuracy, 0.125 loss, and 0.82 dice coefficient was achieved.



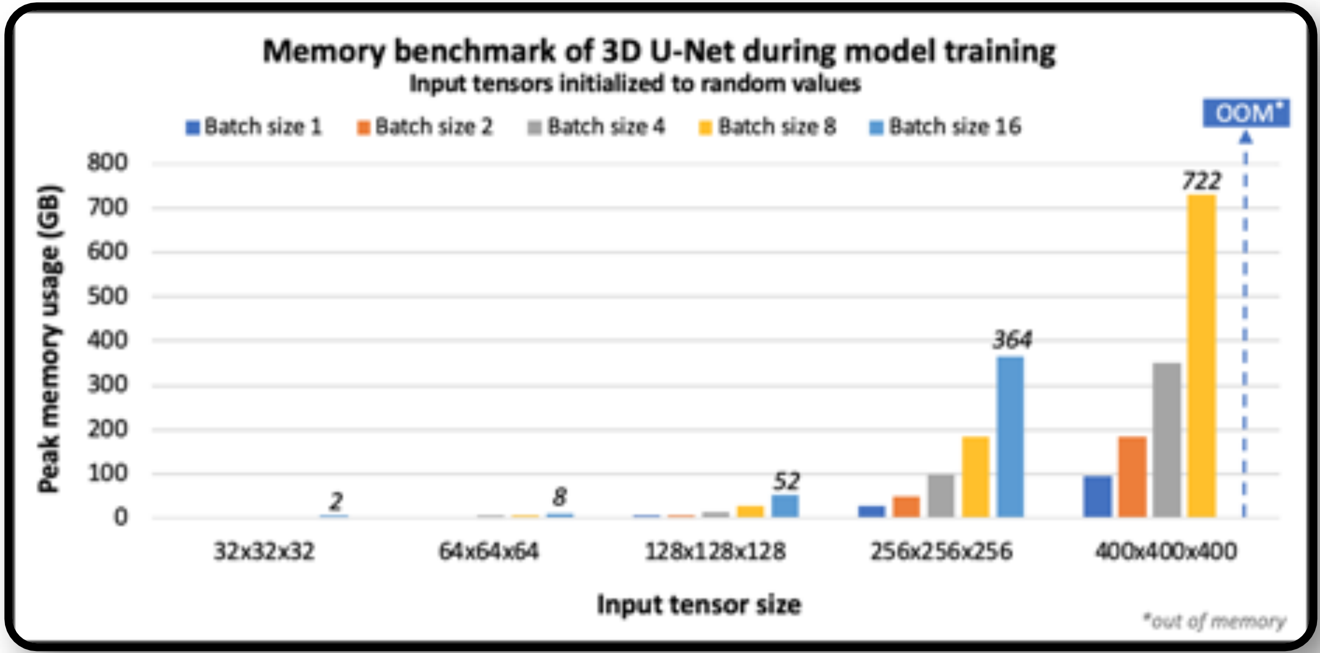
AI-based Gliomas segmentation

https://downloads.dell.com/manuals/common/dellemc_overcoming_memory_bottleneck_ai_healthcare.pdf

"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/Université de Montréal

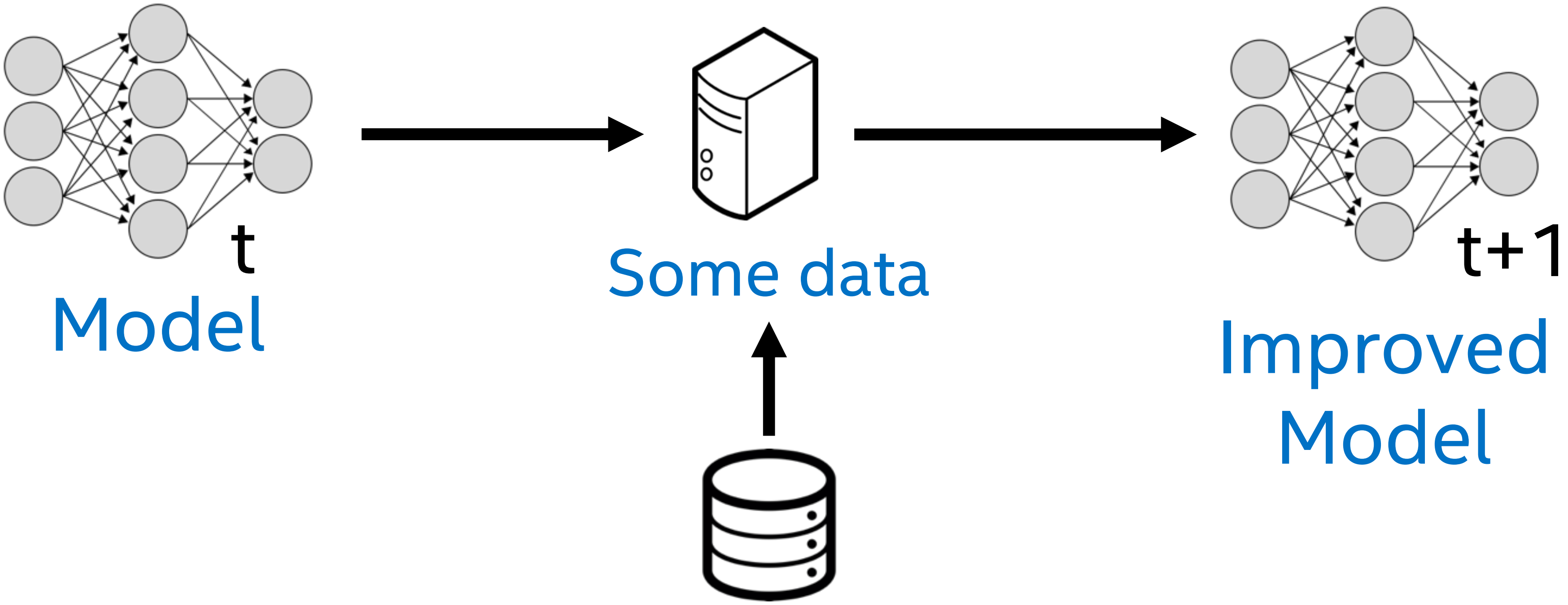


Benchmarking the memory usage of 3D U-Net model-training over various input tensors sizes on an Intel® Xeon® Scalable processor-based server with 1.5 TB system memory

https://downloads.dell.com/manuals/common/dellemc_overcoming_memory_bottleneck_ai_healthcare.pdf

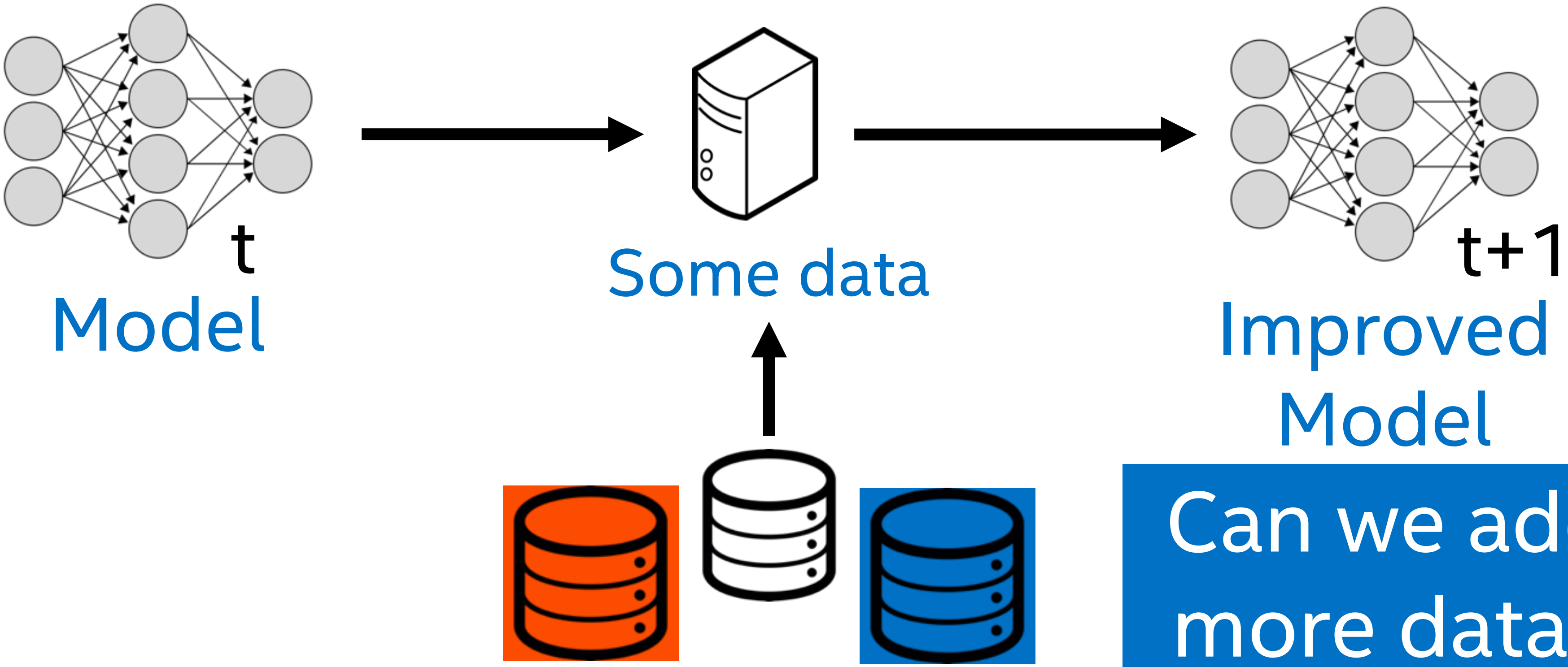
FEDERATED LEARNING

THE DATA SILO PROBLEM



Eventually, we hit the limit of our dataset.

THE DATA SILO PROBLEM

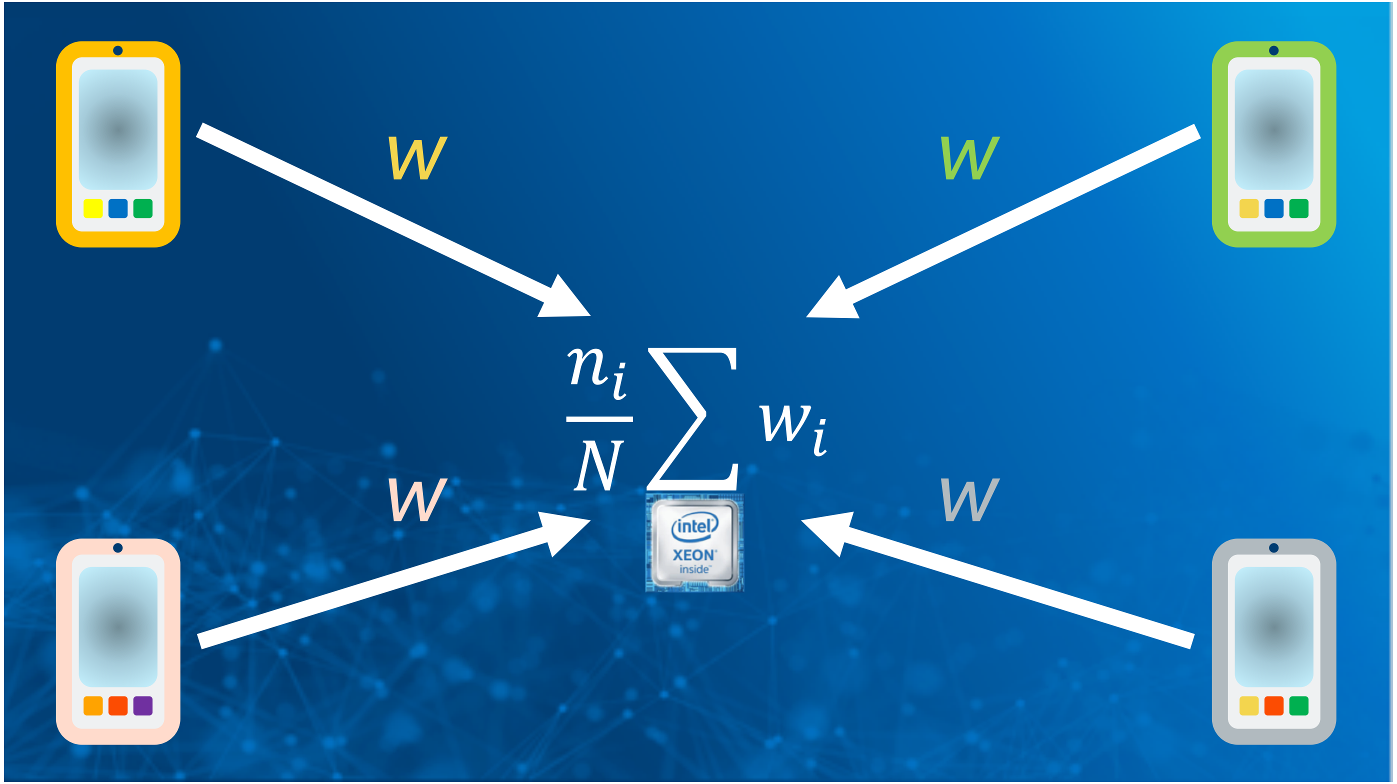


Can we add more data?

THE DATA SILO PROBLEM

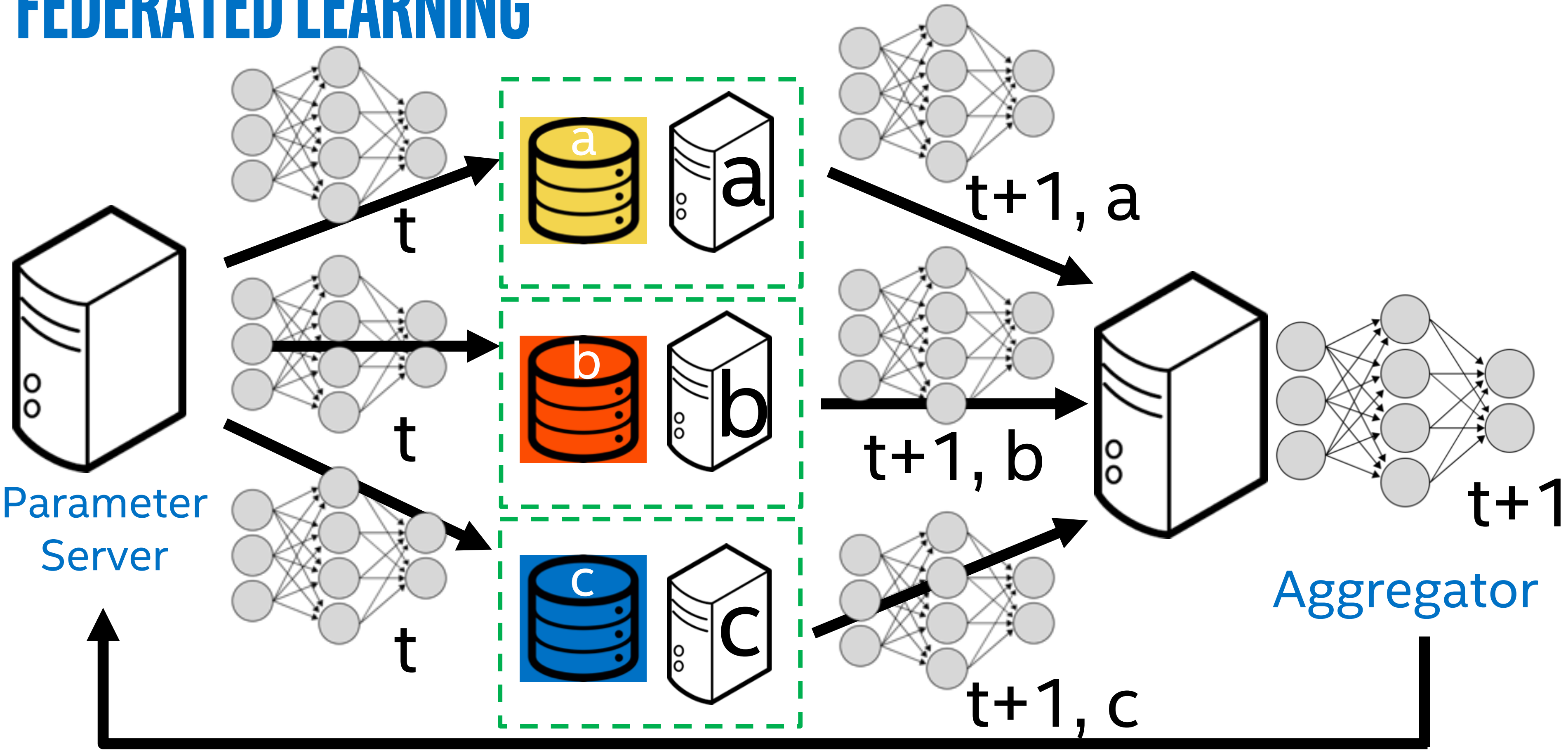
- Privacy / Legality (HIPAA / GDPR)
- Data too valuable (or value unknown)
- Data too large to transmit

FEDERATED LEARNING



https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

FEDERATED LEARNING



Perelman School of Medicine
University of Pennsylvania

BraTS
Multimodal Brain Tumor Segmentation Challenge

scientific reports

Explore our content ˅ Journal information ˅

nature > scientific reports > articles > article

Article | Open Access | Published: 28 July 2020

Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data

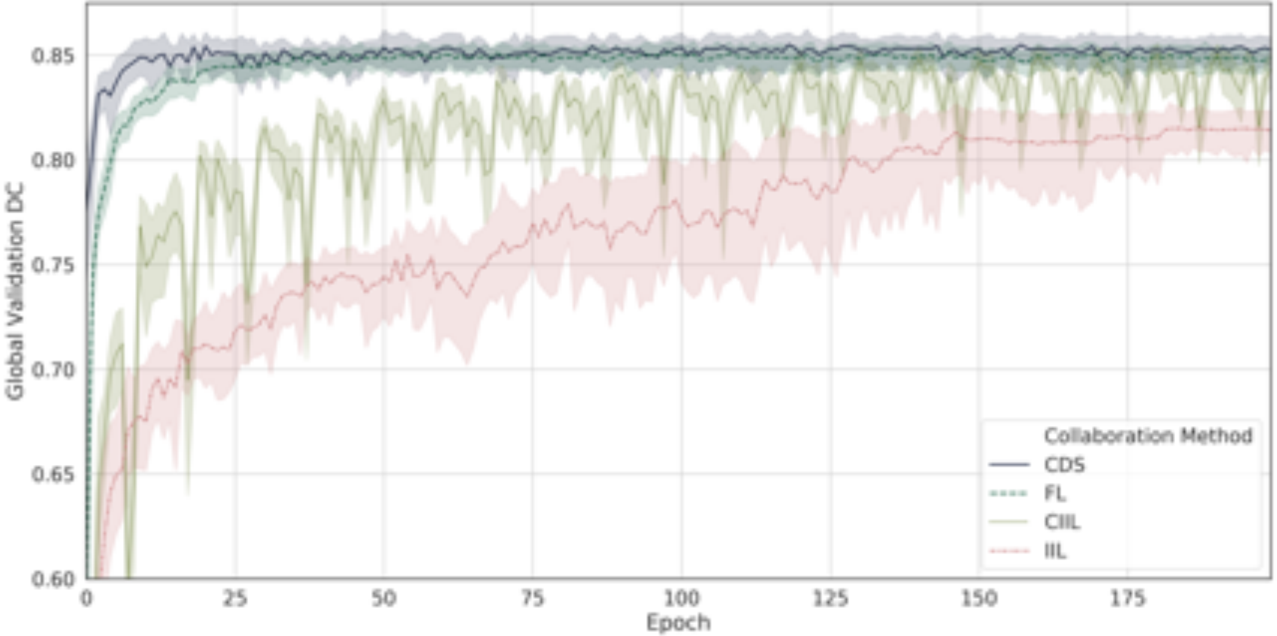
Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen & Spyridon Bakas

Scientific Reports **10**, Article number: 12598 (2020) | Cite this article

Abstract

Several studies underscore the potential of deep learning in identifying complex patterns, leading to diagnostic and prognostic biomarkers. Identifying sufficiently large and diverse datasets, required for training, is a significant challenge in medicine and can rarely be found in individual institutions. Multi-institutional collaborations based on centrally-shared patient data

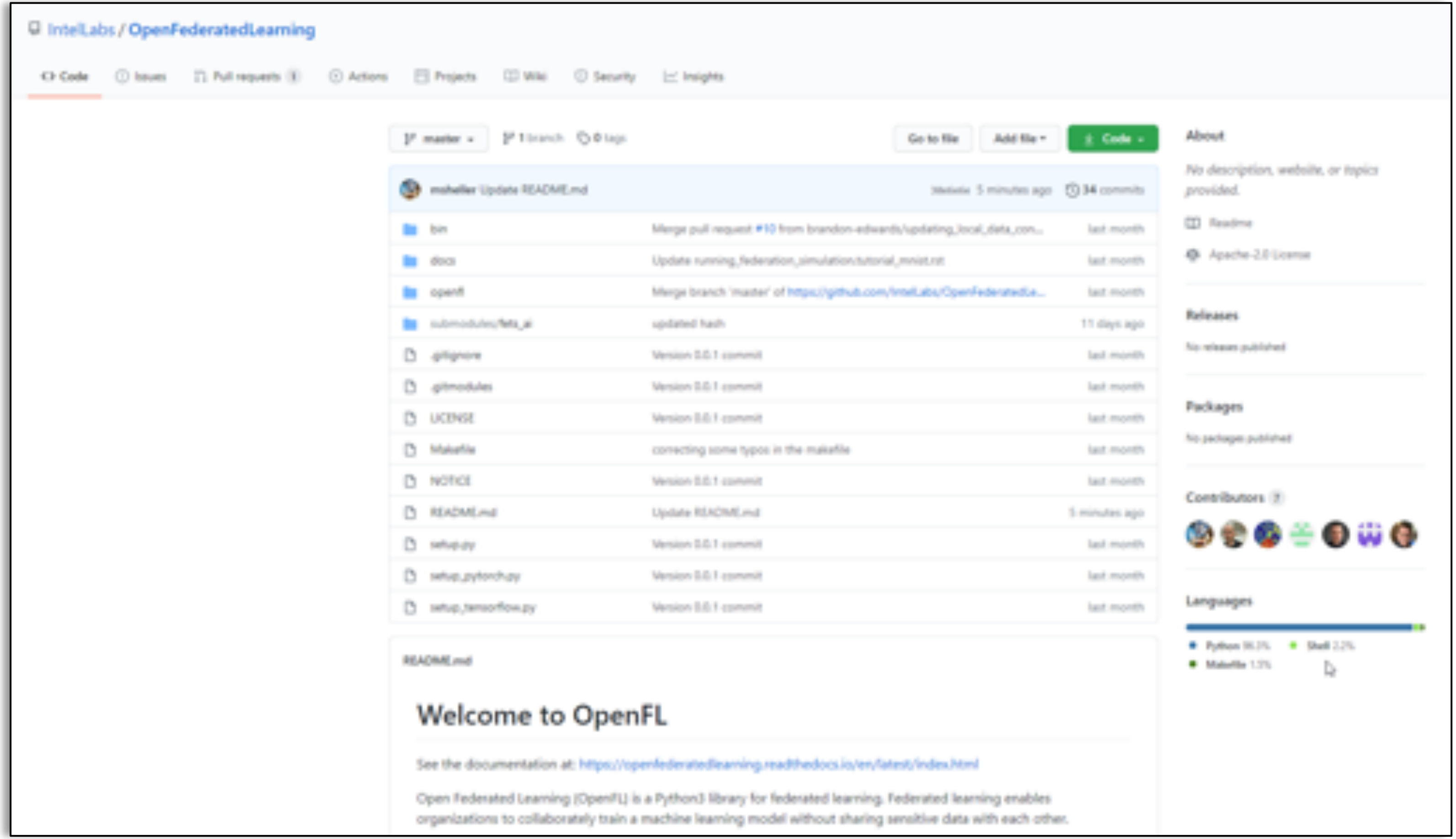
FEDERATING THE U-NET TRAINING [ORIGINAL INSTITUTIONS]*



How much better does each institution do when training on the full data vs. just their own data?

- **~ 17%** better on the hold-out BraTS data
- **~ 2.6%** better on their own validation data

https://www.nature.com/articles/s41598-020-69250-1



https://github.com/IntelLabs/OpenFederatedLearning

14:45 – 15:15 Hands On Session

- Intel®-Optimized Tensorflow

15:15-15:30: Coffee Break

15:30-16:30 Deep Learning – Optimized inference instances

- Performance Optimized Deep Learning Inference using the Intel® distribution of the OpenVINO toolkit
  - What is OpenVINO?
  - Case studies from industry
  - Model Serving
  - Creating an inference pipeline for OpenVINO

16:30 - 17:00 Hands On Session

- AI Inference with the Intel Distribution of OpenVINO

Workshop on Deep Learning Optimized Training Instances

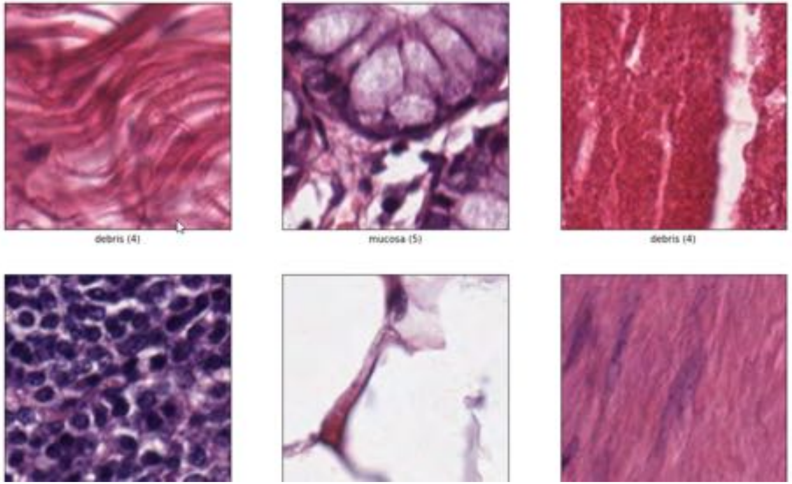
Intel® OneAPI DevCloud

INTEL® OPTIMIZED TENSORFLOW DEMO

https://github.com/IntelAI/unet/tree/master/single-node

Display a few examples from the dataset

In [6]:
```
x_key, y_key = ds_info.supervised_keys
ds_temp = ds.map(lambda x, y: {x_key: x, y_key: y})
tfds.show_examples(ds_temp, ds_info, plot_scale=5);
```



Signup for Access to the Intel® DevCloud for Edge

Sign Up Here:	https://devcloud.intel.com/edge/
Intel's Registration Passcode:	LRZ100951N10E
Code Valid From:	Oct 7, 2020, 00:01 PST
Code Valid To:	Oct 14, 2020, 23:59 PST
Account Activation:	Now
Account Deactivation:	30 days

14:45 – 15:15 Hands On Session
- Intel®-Optimized Tensorflow

15:15-15:30: Coffee Break

15:30-16:30 Deep Learning – Optimized inference instances
- Performance Optimized Deep Learning Inference using the Intel® distribution of the OpenVINO toolkit
  - o What is OpenVINO?
  - o Case studies from industry
  - o Model Serving
  - o Creating an inference pipeline for OpenVINO

16:30 - 17:00 Hands On Session
- o AI Inference with the Intel Distribution of OpenVINO

AI INFERENCE

OpenVINO™

WRITE ONCE, DEPLOY & SCALE DIVERSELY

- TensorFlow
- ONNX
- mxnet
- KALDI
- Caffe

Model Optimizer

OpenVINO™

Inference Engine

- CPU: intel® XEON® inside™
- FPGA: intel® STRATIX® inside™
- Edge: intel® MOVIDIUS™ inside™
- GPU: intel® Iris™ Graphics

WRITE ONCE, DEPLOY & SCALE DIVERSELY

Model Optimizer

OpenVINO™

Inference Engine

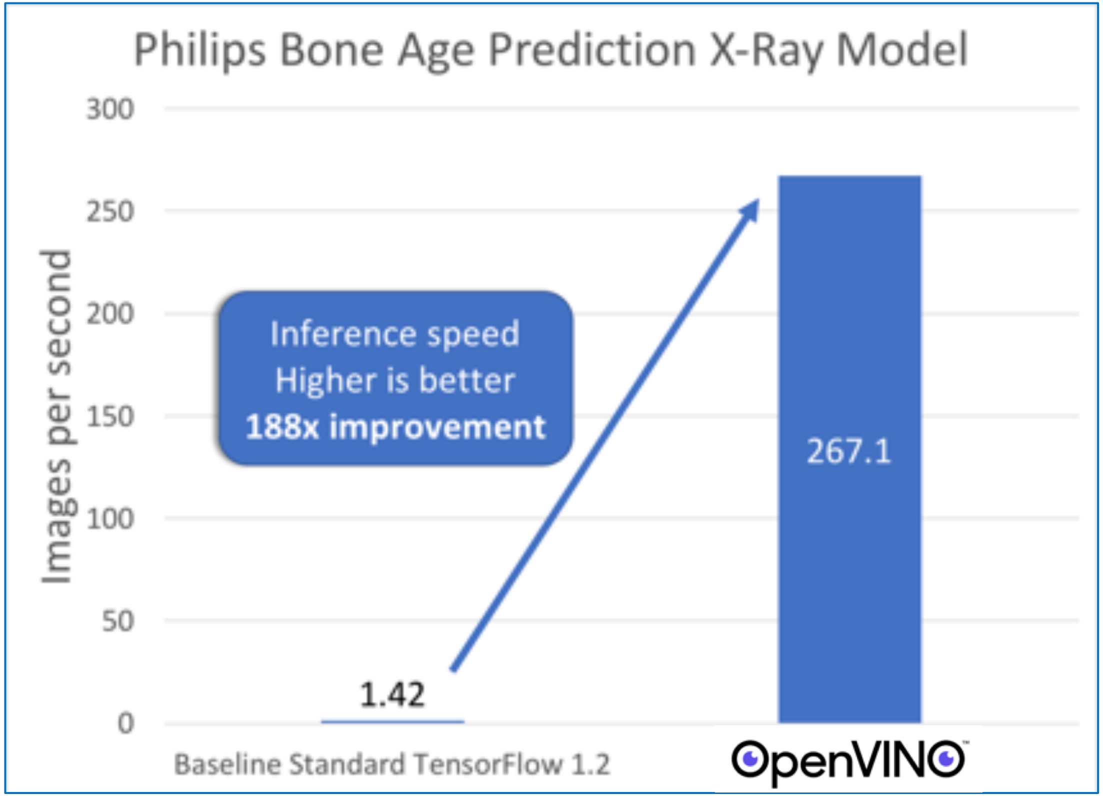
$$\frac{x^5}{x^3}$$

Memory Blocking (Reordering)

USE CASE

"*Intel® Xeon® Scalable processors appear to be the right solution for this type of AI workload. Our customers can use their existing hardware to its maximum potential, while still aiming to achieve quality output resolution at exceptional speeds.*"

–Vijayananda J., Chief Architect and Fellow, Data Science and AI at Philips HealthSuite Insights



https://newsroom.intel.com/news/intel-philips-accelerate-deep-learning-inference-cpus-key-medical-imaging-uses

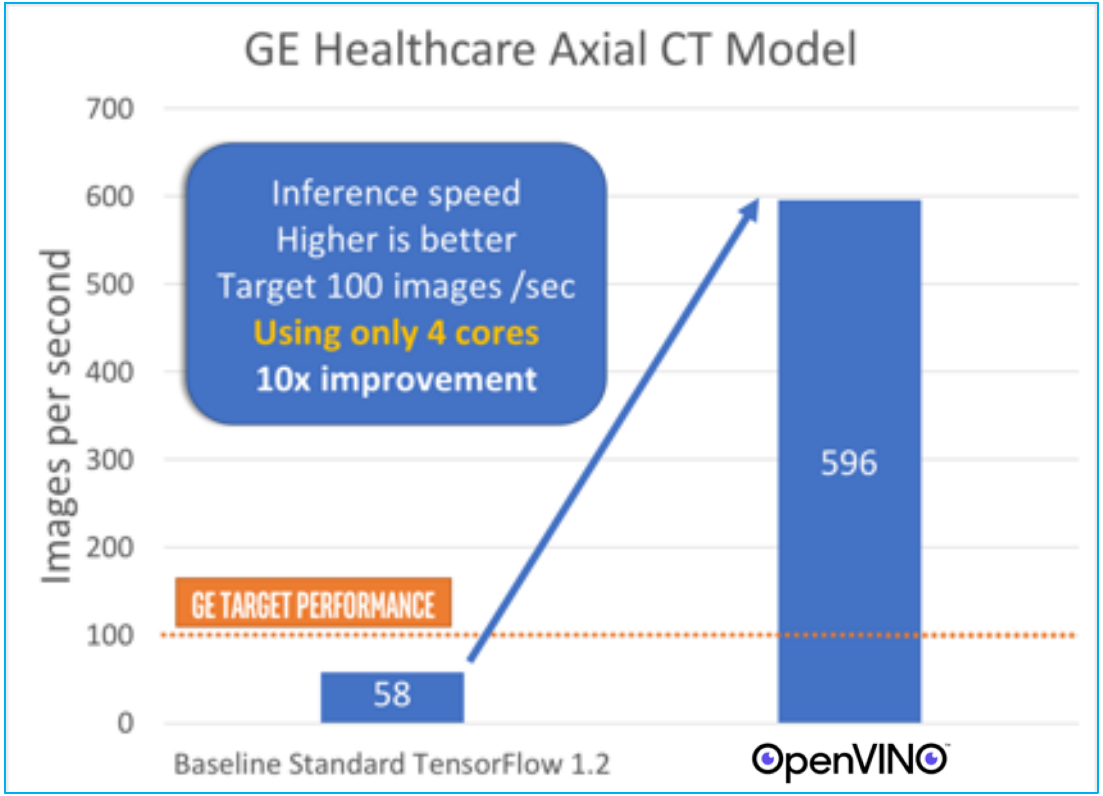
Intel® Xeon® Platinum 8168 processor at 2.70 GHz, Intel® Hyper-Threading Technology (Intel® HT Technology) disabled
Baseline: TensorFlow 1.2 (standard, no Intel optimizations)

USE CASE

GE Healthcare

"We think using general-purpose processors, tools, and frameworks from Intel® can offer a cost-effective way to leverage AI in medical imaging in new and meaningful ways."

David Chevalier, Principal Engineer, GE Healthcare



https://www.intel.ai/ai-enhanced-medical-imaging-to-improve-radiology-workflows

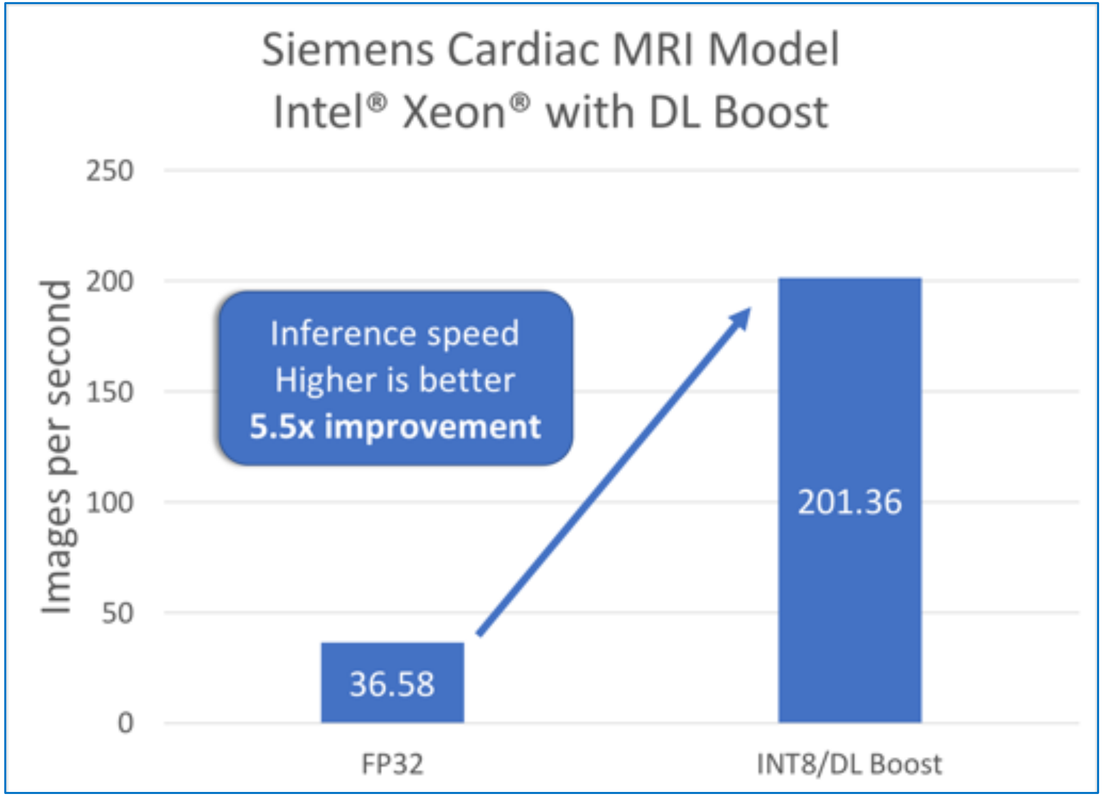
Intel® Xeon® E5-2650 v4 using only 4 cores
Baseline: TensorFlow 1.2 (standard, no Intel optimizations)

USE CASE

SIEMENS Healthineers

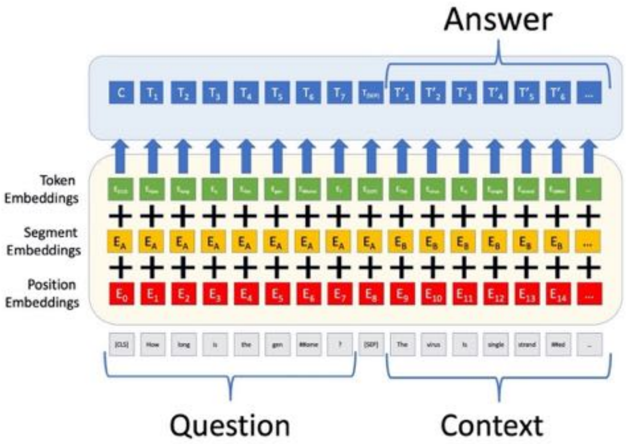
"Siemens Healthineers and Intel® have a shared goal to improve healthcare by applying AI where the data is generated — right at the edge using 2nd-generation Intel® Xeon® Scalable processors with Intel® Deep Learning (DL) Boost and the Intel® Distribution for OpenVINO™. This enables real-time applications of cardiac MRI, making data interpretation available right after it's collected."

David Ryan, General Manager, Health and Life Sciences Sector, Internet of Things Group, Intel

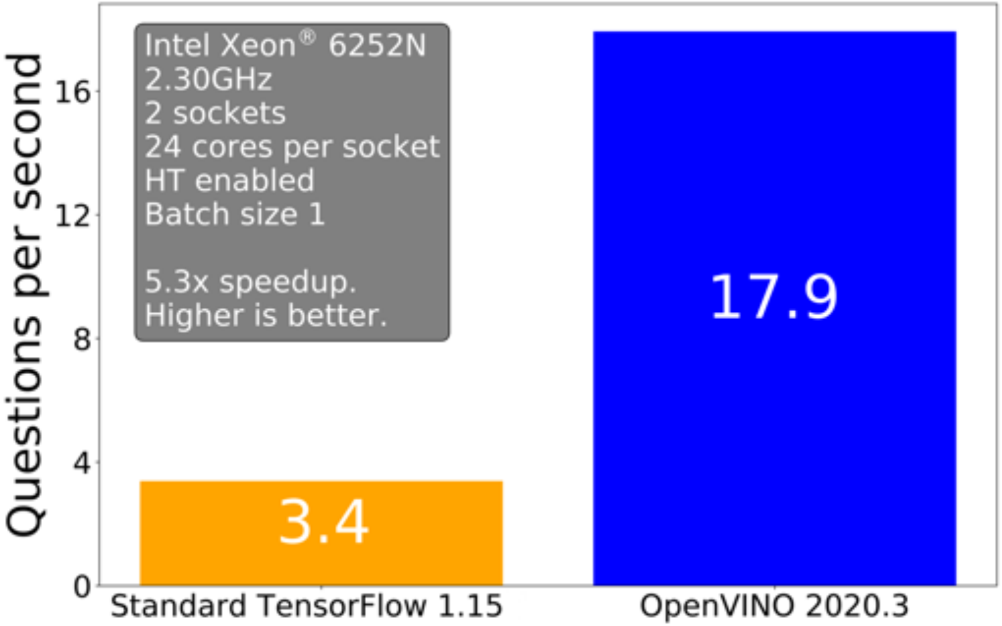


https://newsroom.intel.com/news/siemens-healthineers-intel-demonstrate-potential-of-ai-real-time-cardiac-mri-diagnosis

NLP USE CASE



Context	Symptom severity scores were quantified using the following five measures: (i) individual symptom score for 20 symptoms, (ii) the upper respiratory symptom score, calculated as the sum of severity scores for earache, runny nose, sore throat, and sneezing, (iii) the lower respiratory symptom score, calculated as the sum of severity scores for cough, difficulty breathing, hoarseness, and chest discomfort, (iv) the gastrointestinal symptom score, calculated as the sum of severity scores for diarrhea, vomiting, anorexia, nausea, and (Table 1) . There was season-to-season variability in the leading causes of ... The findings of our study, conducted over a 5-year period at five geographically dispersed sites in the USA, demonstrate that human coronavirus (HCoV) is an important cause of influenza-like illness (ILI) ranged from 4% to 22%. [8] [9] [10] [11] 14 Additionally, we found HCoV-OC43 to be the most common species among adults, as has been reported elsewhere. 8, 9, 11, 12, 14 HCoV-OC43 and HCoV-229E were the most common strains in alternate seasons, reflecting a season-to-season variability of HCoV strain circulation that has been reported in other multiyear studies.
Question	What is the most common species of Human Coronavirus among adults?
Answer	HCoV-OC43



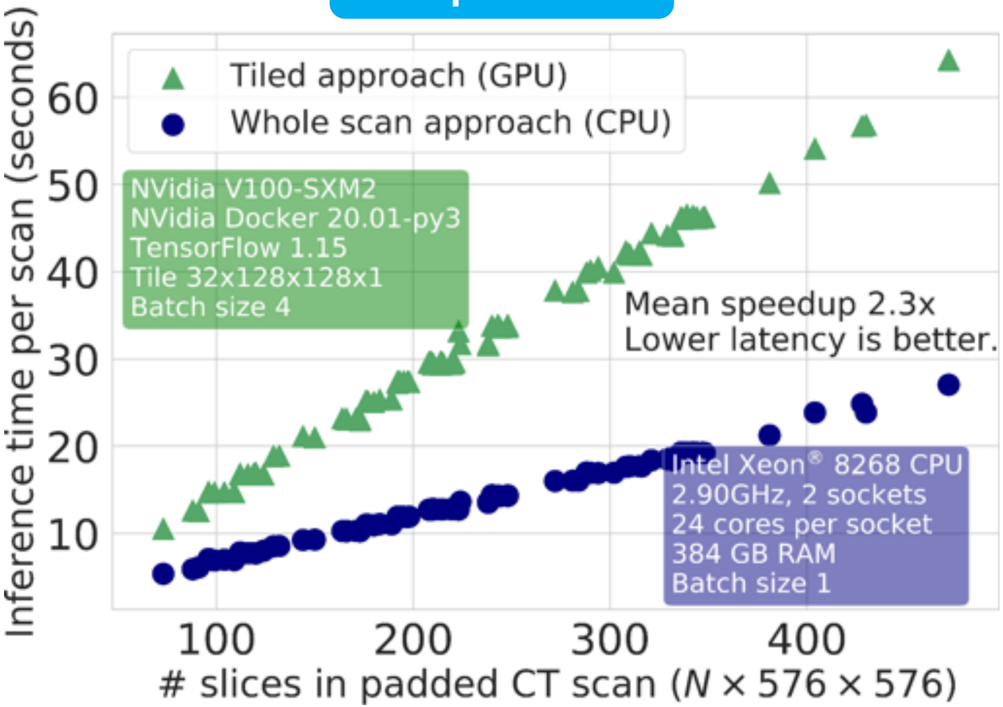
"The 3D data volume is at least a 1,000 times larger than the previous 2D data volume, making the analysis and evaluation of individual layers by human experts impossible. By contrast, with the OpenVINO™ toolkit processing times of one 3D image are now under an hour."

—Andreas Marek, Senior HPC expert and Lead of the Data Analytics Group, Max Planck Computing and Data Facility (MPCDF)

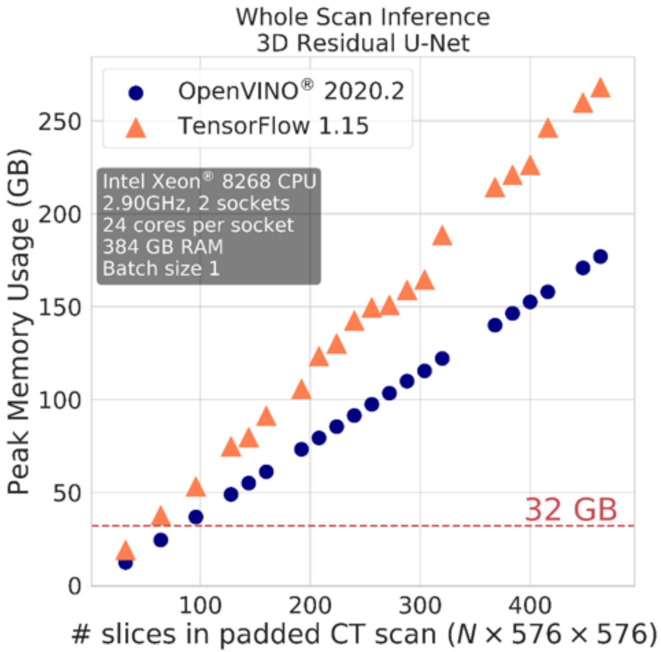
https://www.intel.com/content/www/us/en/customer-spotlight/stories/max-planck-institute-customer-story.html

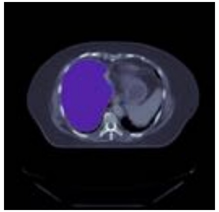
GE Healthcare

Speed



Memory





QUICKLY DEPLOY WITH PRE-BUILT PROJECTS

OPEN-SOURCED REFERENCE IMPLEMENTATIONS

Parking Lot Tracker
Receive or post information on available parking spaces by tracking how many vehicles enter and exit a parking lot.

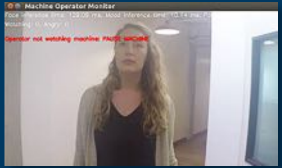
Use Cases
- Track and analyze vehicle activity
- Report on parking space availability

Shopper Gaze Monitor
Build a solution to analyze customer expressions and reactions to product advertising collateral that is positioned on retail shelves.

Use Cases
- Measure active versus inactive user product engagement
- Capture analytics on shopper reactions to visual ads

Shopper Mood Monitor
Detect the mood of shoppers when looking at a retail or kiosk display.

Use Cases
Mall shoppers using interactive or map kiosk
Grocery store shoppers viewing digital signage ads
Hospitals using a kiosk to assist patients or visitors



Machine Operator Monitor
Send notifications when an employee appears to be distracted when operating machinery.

Use Cases
- Industrial or manufacturing facilities
- Construction sites
- Warehouses

Intruder Detector
Build an application that alerts you when someone enters a restricted area. Learn how to use models for multiclass object detection.

Use Cases
- Record and send alerts on activity in controlled spaces
- Track parking lots, entrances, and property

Store Traffic Monitor
Monitor three different streams of video that count people inside and outside of a facility. This application also counts product inventory.

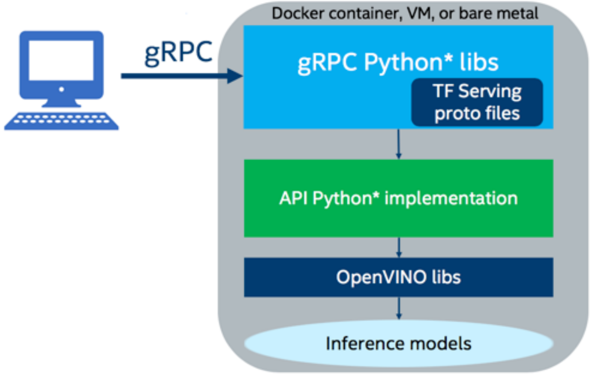
Use Cases
- Movement of people
- Foot activity in retail or warehouse spaces
- Inventory availability of products on shelves

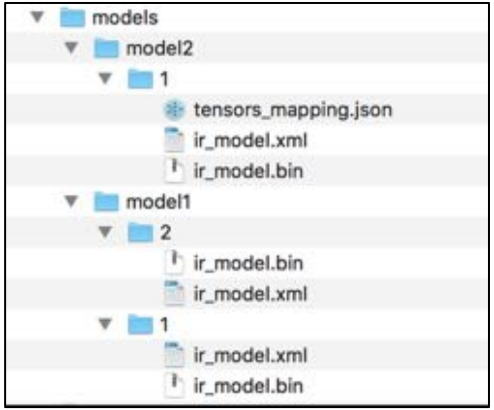
Restricted Zone Notifier
Secure work areas and send alerts if someone enters the restricted space.

Use Cases
- Track worker activity in proximity to heavy machinery
- Develop safety solutions using computer vision technologies

View All ▸ https://software.intel.com/en-us/iot/reference-implementations

OPENVINO MODEL SERVER

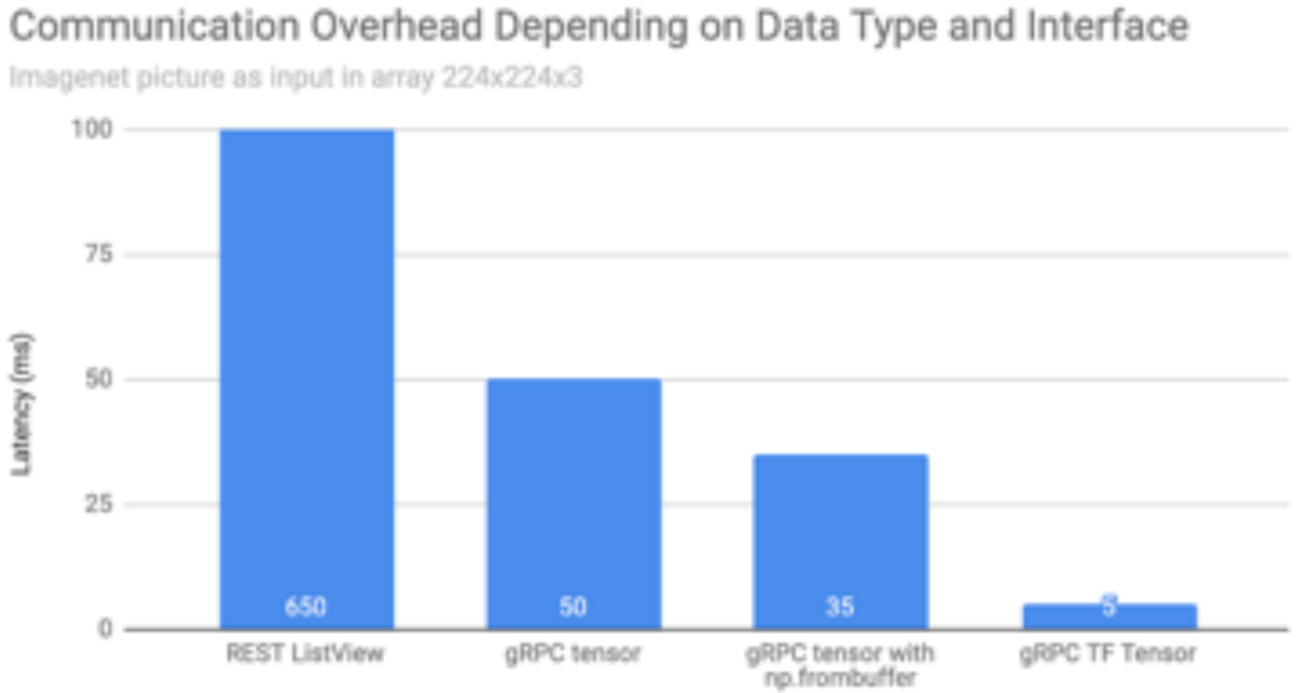




- Same gRPC API as TensorFlow Serving
- Implemented as a Python* service
- Fully compatible with same clients
- Optimized for Intel® CPU, FPGA, VPA
- Suited for Docker containers

https://github.com/IntelAI/OpenVINO-model-server

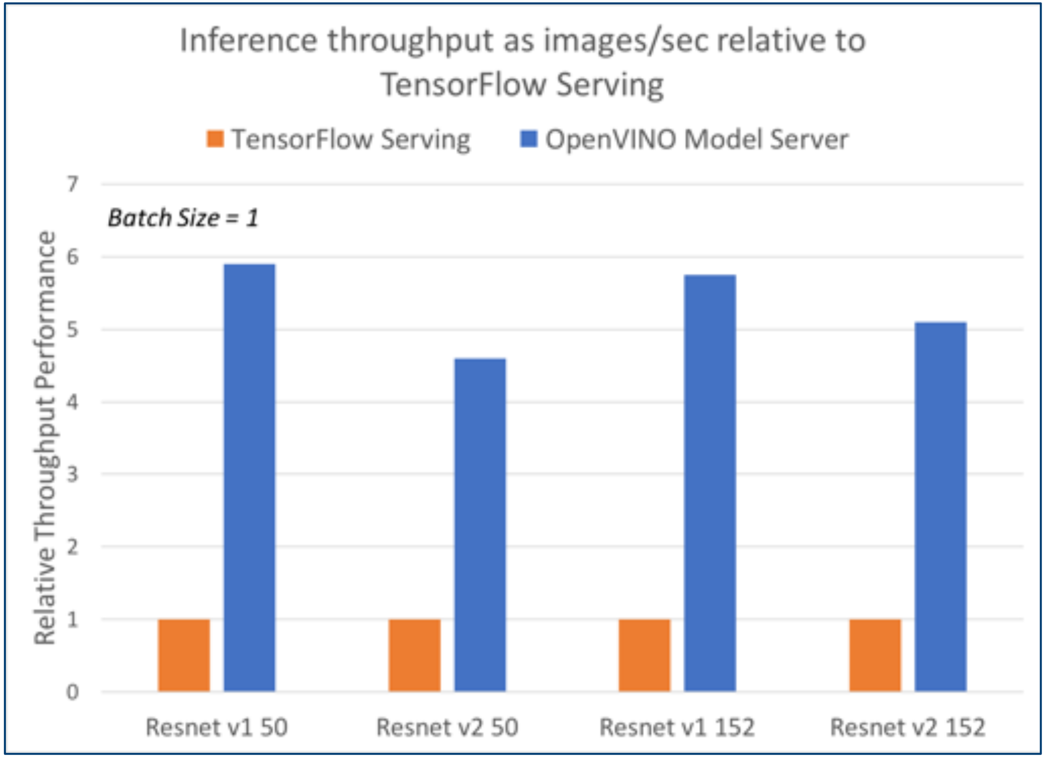
OPENVINO MODEL SERVER



Serialization method has big impact on latency

OPENVINO MODEL SERVER

OpenVINO™



Up to 5x improvement over TensorFlow Serving

Performance results are based on internal testing done on 27th September 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. Test configuration: Dual Intel® Xeon® Platinum 8180 processor @ 2.50GHz, 376.28GB total system memory, Ubuntu-16.04-xenial operating system.

https://www.intel.ai/openvino-model-server-boosts-ai-inference-operations

Enable AWS Greengrass* and OpenVINO™ toolkit

This guide explains how to enable AWS Greengrass* and OpenVINO™ toolkit. Specifically, the guide demonstrates how to:

- Set up the Intel® edge device with Clear Linux* OS
- Install the OpenVINO™ toolkit and Amazon Web Services* (AWS*) Greengrass* software stacks
- Use AWS Greengrass* and AWS Lambda* to deploy the FaaS samples from the cloud

Overview

Hardware accelerated Function-as-a-Service (FaaS) enables cloud developers to deploy inference functionalities [1] on Intel® IoT edge devices with accelerators (CPU, Integrated GPU, Intel® FPGA, and Intel® Movidius™ technology). These functions provide a great developer experience and seamless migration of visual analytics from cloud to edge in a secure manner using a containerized environment. Hardware-accelerated FaaS provides the best-in-class performance by accessing optimized deep learning libraries on Intel® IoT edge devices with accelerators.

Supported platforms

- Operating System: Clear Linux OS latest release
- Hardware: Intel® core platforms (that support inference on CPU only)

ADLINK Teams with Intel and AWS to Offer AI at the Edge for Machine Vision Applications

Solution combines Intel® Distribution of OpenVINO™ toolkit, AWS Greengrass, Amazon Sagemaker and ADLINK Edge™ to simplify Edge AI deployments

2019/12/02 San Jose

ADLINK Technology, a global leader in edge computing, has joined forces with Intel and Amazon Web Services (AWS) to simplify artificial intelligence (AI) at the edge for machine vision. The integrated solution offers an Amazon Sagemaker-built machine learning model optimized by and deployed with the Intel® Distribution of OpenVINO™ toolkit, the ADLINK Edge™ software suite, and certification on AWS Greengrass.

The ADLINK AI at the Edge solution closes the loop on the full cycle of machine learning model building—from design to deployment to improvement—by automating edge computing processes so that customers can focus on developing applications without needing advanced knowledge of data science and machine learning models. The ADLINK AI at the Edge solution features:

- Intel Distribution of OpenVINO toolkit, optimizes deep learning workloads across Intel® architecture, including accelerators, and streamline deployments from the edge to the cloud.
- Amazon Sagemaker, a fully-managed service that covers the entire machine learning workflow.
- AWS Greengrass, which extends AWS to edge devices so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage.
- The ADLINK Data River™, offering translation between devices and applications to enable a vendor-neutral ecosystem to work seamlessly together.

https://www.adlinktech.com/en/News_19120300290236500

Signup for Access to the Intel® DevCloud for Edge

Sign Up Here:	https://devcloud.intel.com/edge/
Intel's Registration Passcode:	LRZ100951N10E
Code Valid From:	Oct 7, 2020, 00:01 PST
Code Valid To:	Oct 14, 2020, 23:59 PST
Account Activation:	Now
Account Deactivation:	30 days

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

OpenVINO™

DEEP LEARNING

Caffe · TensorFlow · ONNX · mxnet · KALDI

Model Optimizer

Inference Engine

Supports 100+ public models, incl. 30+ pretrained models

COMPUTER VISION

OpenCV · OpenCL™ · OpenVX

Computer vision library (kernel & graphic APIs)

Optimized media encode/decode functions

SUPPORTS MAJOR AI FRAMEWORKS

Rapid adoption by developers

CROSS-PLATFORM FLEXIBILITY

Multiple products launched based on this toolkit

HIGH PERFORMANCE, HIGH EFFICIENCY

Breadth of product portfolio

Strong Adoption + Rapidly Expanding Capability

software.intel.com/openvino-toolkit

Obtain open source version at 01.org/openvinotoolkit

WRITE ONCE, DEPLOY & SCALE DIVERSELY

TensorFlow
ONNX
mxnet
KALDI
Caffe

Model Optimizer

OpenVINO™

Inference Engine

CPU
intel® XEON® inside™

FPGA
intel® ARRIA® 10 inside™

Edge
intel® MOVIDIUS™ inside™

GPU
intel® Iris™ Graphics

INTEL NEURAL COMPUTE STICK 2

HEALTHCARE USE CASES

Machine Learning and Mammography
Detecting invasive ductal carcinoma with convolutional neural networks showing how existing deep learning technologies can be utilized to train artificial intelligence (AI) to be able to detect invasive ductal carcinoma (IDC)[1] (breast cancer) in unlabeled histology images.

AI Assists with Skin Cancer Screening
Doctor Hazel, a skin cancer screening service powered by AI that operates in real time, relies on an extensive library of images to distinguish between skin cancer and benign lesions, making it easier for people to seek professional medical advice.

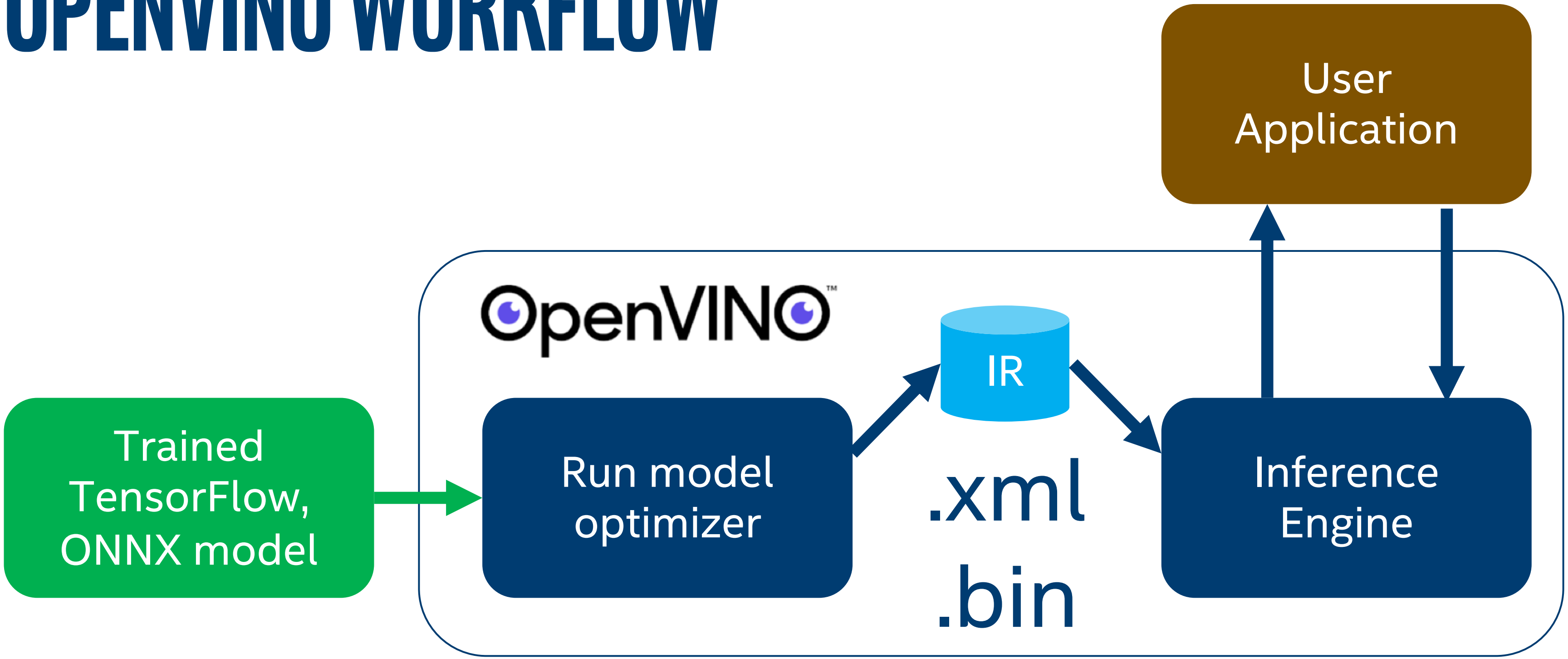
AI Helps Detect Bacteria in Water
Offline analysis is accomplished with a digital microscope connected to a laptop running Ubuntu* and the Intel® Movidius™ Neural Compute Stick. After analysis, contamination sites are marked on a map in real time

AI has the power to make a difference and change lives. What will you make?

Get Started Today ▶

intel.com/ncs

OPENVINO WORKFLOW

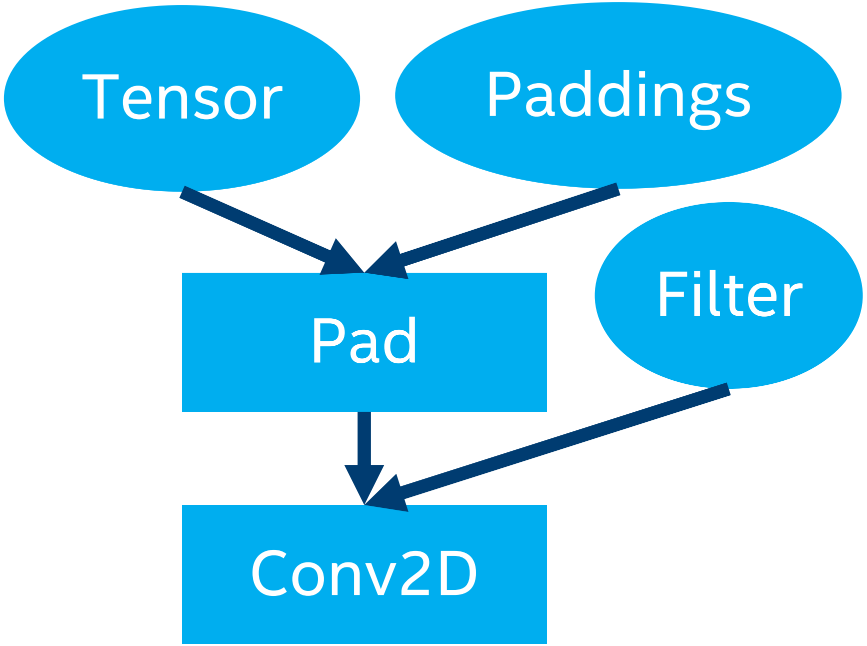


INTERMEDIATE REPRESENTATION (IR) FILE

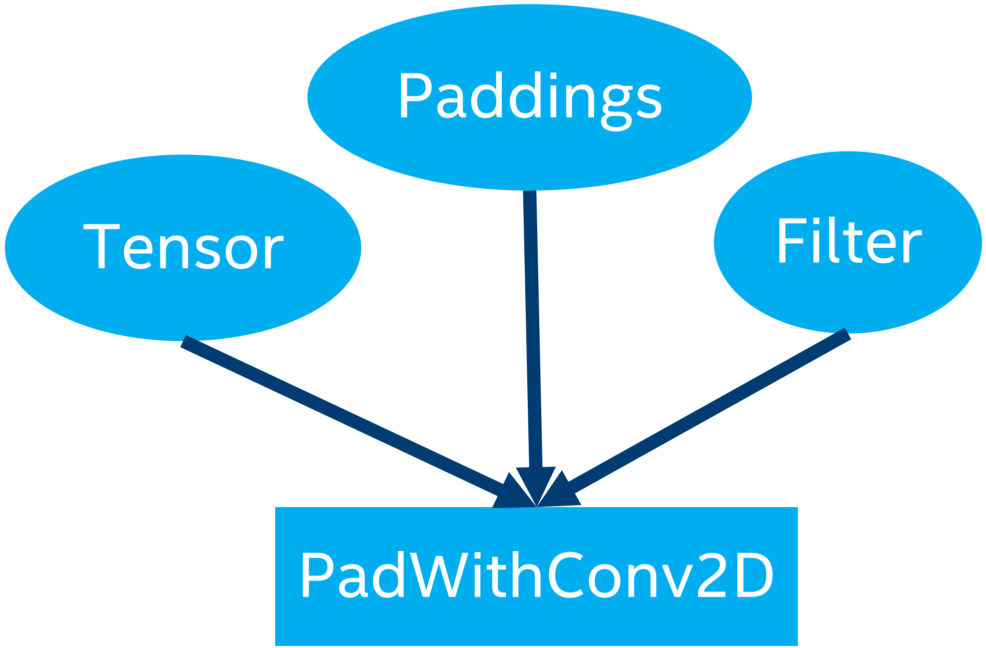
```
[bduser@merlin-param01 FP32]$ head -40 3d_unet_decathlon.xml
<?xml version="1.0" ?>
<net batch="1" name="3d_unet_decathlon" version="5">
        <layers>
                <layer id="0" name="MRImages" precision="FP32" type="Input">
                        <output>
                                <port id="0">
                                        <dim>1</dim>
                                        <dim>1</dim>
                                        <dim>144</dim>
                                        <dim>144</dim>
                                        <dim>144</dim>
                                </port>
                        </output>
                </layer>
                <layer id="1" name="encodeA_conv0/convolution" precision="FP32" type="Convolution">
                        <data auto_pad="same_upper" dilations="1,1,1" group="1" kernel="3,3,3" output="16" pads_begin="1,1,1" pads_end="1,1,1" strides="1,1,1"/>
                        <input>
                                <port id="0">
                                        <dim>1</dim>
                                        <dim>1</dim>
                                        <dim>144</dim>
                                        <dim>144</dim>
                                        <dim>144</dim>
                                </port>
```

The .bin file just has the weights.

GRAPH-LEVEL OPTIMIZATIONS



Before Fusion



After Fusion

SETUP

```
source /opt/intel/openvino/bin/setupvars.sh
```

MODEL OPTIMIZER

```
# Create FP16 IR files
python3 $INTEL_OPENVINO_DIR/deployment_tools/model_optimizer/mo.py \
    --input_model /data/Healthcare_app/data/saved_model_frozen.pb \
    --input_shape=[1,144,144,4] \
    --data_type FP16  \
    --output_dir models/FP16  \
    --model_name saved_model

# Create FP32 IR files
python3 $INTEL_OPENVINO_DIR/deployment_tools/model_optimizer/mo.py \
    --input_model /data/Healthcare_app/data/saved_model_frozen.pb \
    --input_shape=[1,144,144,4] \
    --data_type FP32  \
    --output_dir models/FP32  \
    --model_name saved_model
```

infer_simple.py

```python
#!/usr/bin/env python

from openvino.inference_engine import IECore
import numpy as np

ie = IECore()

# Read the model file
net = ie.read_network(model="model123.xml", weights="model123.bin")

# Load model to hardware
exec_net = ie.load_network(network=net, device_name="CPU")

input_data = np.ones((1, 4, 144, 144))  # Create some data to pass to model

# Do inference
res = exec_net.infer(inputs={"input_name_123": input_data})

# Get prediction
prediction = res["output_name_123"]
```

ONNX too!

Analogous to TensorFlow feed_dict

net.inputs.keys()
net.outputs.keys()

```
print("The network inputs are:")
for idx, input_layer in enumerate(net.inputs.keys()):
    print("{}: {}, shape = {} [N,C,H,W,D]".format(idx,input_layer,net.inputs[input_layer].shape))

print("The network outputs are:")
for idx, output_layer in enumerate(net.outputs.keys()):
    print("{}: {}, shape = {} [N,C,H,W,D]".format(idx,output_layer,net.outputs[output_layer].shape))
```

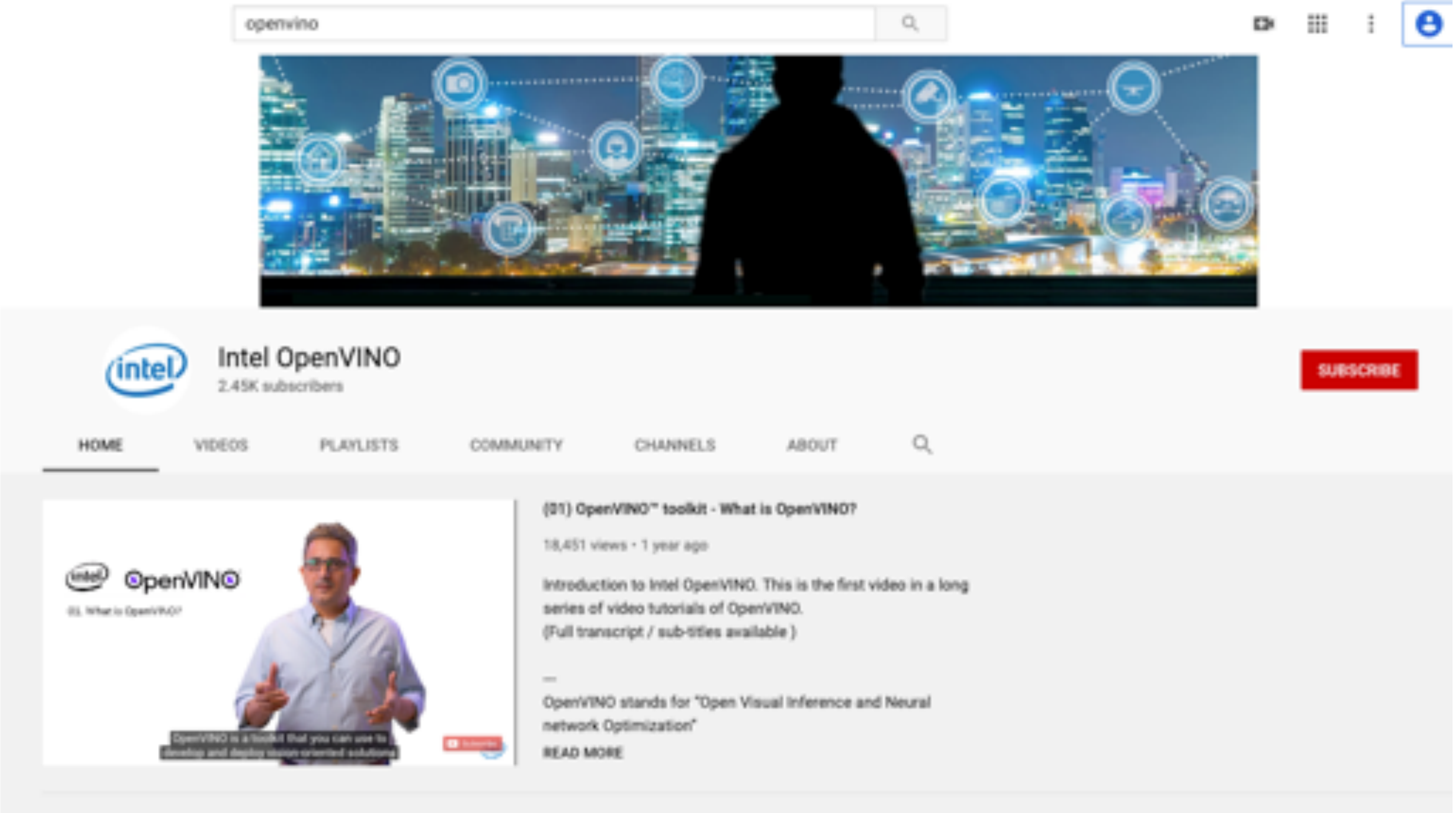
CHANNELS FIRST

Resize the input (e.g. fully convolutional models)

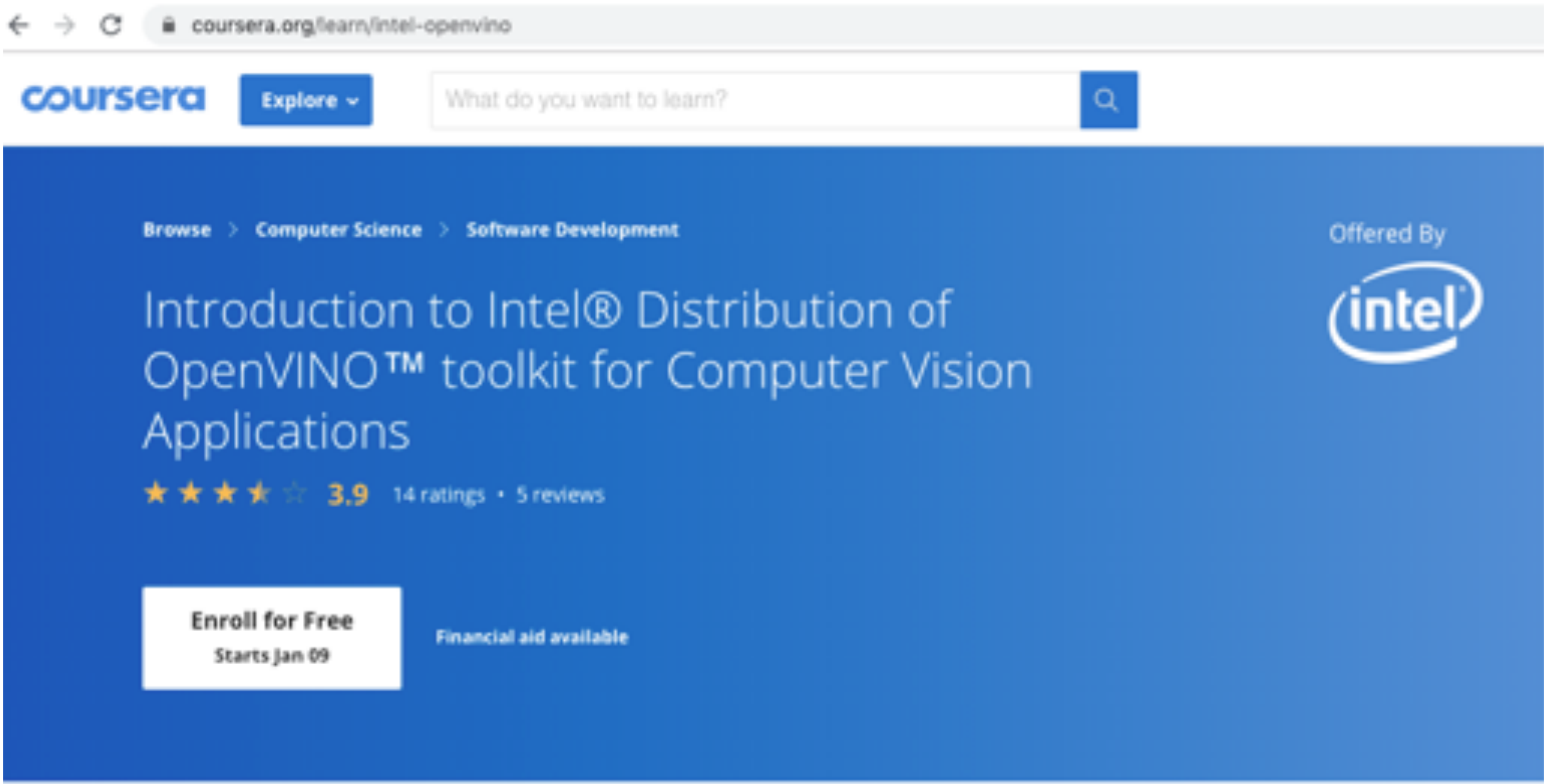
```
net.reshape({"input_name_123":(batch_size,n_channels,height,width,depth)})
```

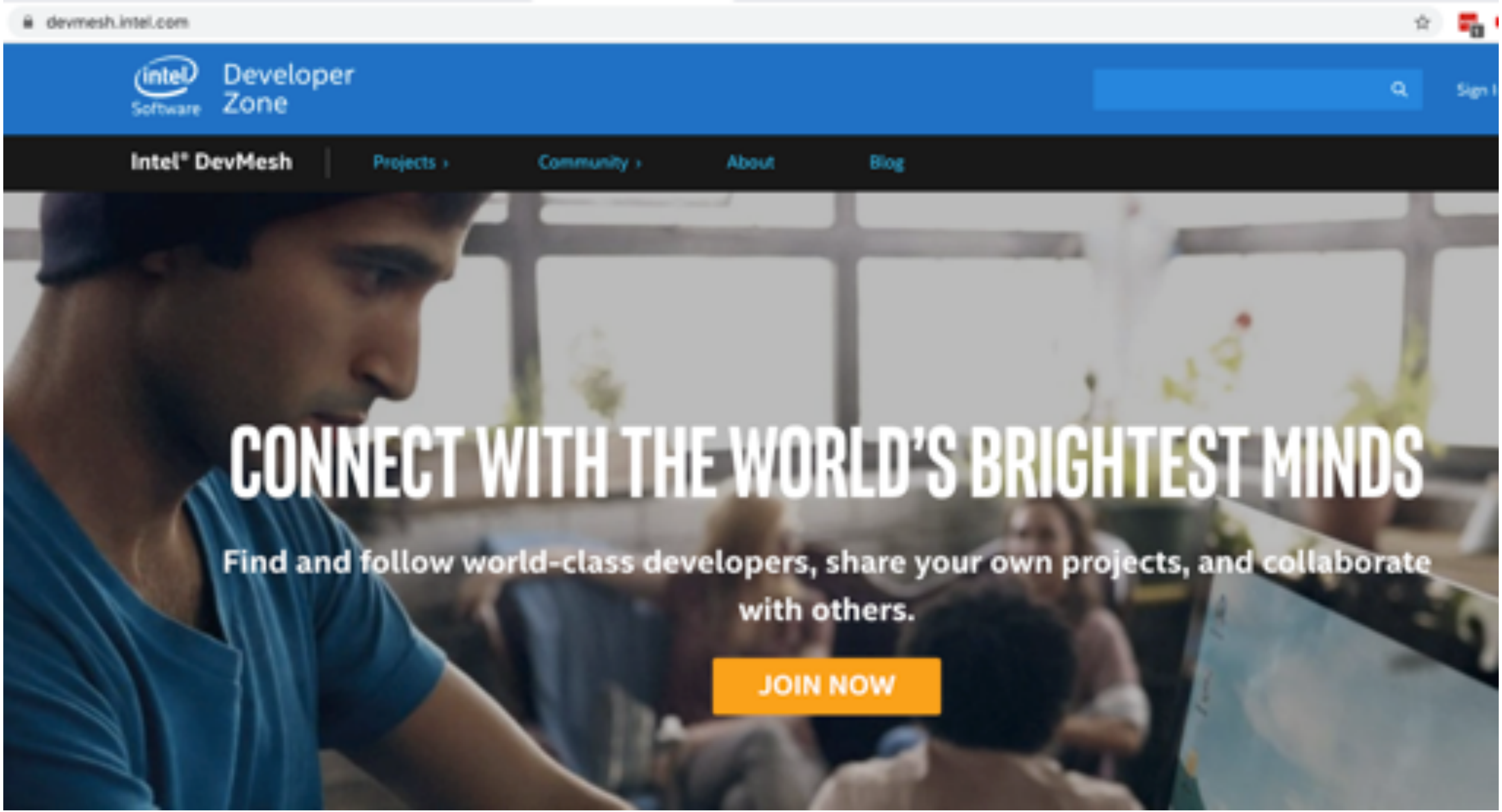
What devices do I have?

```
ie = IECore()
print("Available devices")

for device in ie.available_devices:
    print("\tDevice: {}".format(device))
    print("\Metrics:")
    for metric in ie.get_metric_device(device, "SUPPORTED_METRICS"):
        try:
            metric_val = ie.get_metric(device, metric)
            print("\t\t{}: {}".format(metric, param_to_string(metric_val)))
        except TypeError:
            print("\t\t{}: UNSUPPORTED TYPE".format(metric))
```



Search YouTube for “Intel OpenVINO”


coursera.org/learn/intel-openvino
coursera
Explore
What do you want to learn?
Browse > Computer Science > Software Development
Offered By
intel
Introduction to Intel® Distribution of OpenVINO™ toolkit for Computer Vision Applications
3.9 14 ratings • 5 reviews
Enroll for Free
Starts Jan 09
Financial aid available



devmesh.intel.com

A DEVELOPMENT SANDBOX
FOR DATA CENTER TO EDGE WORKLOADS

Develop, test, and run your workloads on a cluster of the latest Intel® hardware and software. With integrated Intel® optimized frameworks, tools, and libraries, you'll have everything you need for your projects.

Try Out a Diverse Collection of Intel® Hardware

Expand your skills and experiment with this state-of-the-art cluster that offers capabilities such as natural language processing and time-series analysis, as well as edge acceleration hardware.

Develop with Intel Software Tools

Jump-start your projects without having to download, configure, or install the latest compilers, performance libraries, and tools from Intel.

oneAPI

Compilers
Profilers/Analyzers
Libraries
Debuggers

OpenVINO

Use Popular AI Frameworks

TensorFlow
PyTorch
PaddlePaddle

Accelerate your algorithms and applications with Intel®

https://software.intel.com/en-us/devcloud

Signup for Access to the Intel® DevCloud for Edge

Sign Up Here: https://devcloud.intel.com/edge/

Intel's Registration Passcode:	LRZ100951N10E
Code Valid From:	Oct 7, 2020, 00:01 PST
Code Valid To:	Oct 14, 2020, 23:59 PST
Account Activation:	Now
Account Deactivation:	30 days