Software

SUPERCHARGE YOUR AI APPLICATIONS ON INTEL ARCHITECTURE

Shailen Sobhee

AI Specialist @ Intel

Developer Products Division

About your trainer today

- Shailen Sobhee
- AI Software Technical Consulting Engineer @ Intel
- Computer Science and Electrical Engineering (Jacobs University Bremen)
- Computational Science and Engineering (Technical University Munich)

Agenda

- Overview of Intel® software and hardware
  - We will go quick through them ☺
- Hands-on activity with a concrete medical example
  - Brain tumour detection using deep learning

Legal Disclaimer & Optimization Notice

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
Performance results are based on testing as of July 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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

INTEL® PARALLEL STUDIO XE 2019

What's Inside Intel® Parallel Studio XE 2019

Comprehensive Software Development Tool Suite

COMPOSER EDITION

BUILD
Compilers & Libraries

C / C++, Fortran Compilers

Intel® Math Kernel Library

Intel® Data Analytics Acceleration Library

Intel Threading Building Blocks
C++ Threading

Intel® Integrated Performance Primitives
Image, Signal & Data Processing

Intel® Distribution for Python*
High Performance Python

PROFESSIONAL EDITION

ANALYZE
Analysis Tools

Intel® VTune™ Amplifier
Performance Profiler

Intel® Inspector
Memory & Thread Debugger

Intel® Advisor
Vectorization Optimization
Thread Prototyping
& Flow Graph Analysis

CLUSTER EDITION

SCALE
Cluster Tools

Intel® MPI Library
Message Passing Interface Library

Intel® Trace Analyzer & Collector
MPI Tuning & Analysis

Intel® Cluster Checker
Cluster Diagnostic Expert System

Operating System: Windows*, Linux*, MacOS[1]*

Intel® Architecture Platforms

[1]Available only in the Composer Edition.

HPC & AI Software Optimization Success Stories

Intel® Parallel Studio XE

SCIENCE & RESEARCH

Up to **35X** faster application performance

NERSC (National Energy Research Scientific Computing Center)

Read case study

ARTIFICIAL INTELLIGENCE

Performance speedup of up to **23X** faster with Intel optimized scikit-learn vs. stock scikit-learn

Google Cloud Platform

Read blog

LIFE SCIENCE

Simulations ran up to **7.6X** faster with **9X** energy efficiency**

LAMMPS code - Sandia National Laboratories

Read technology brief

For more success stories, review Intel® Parallel Studio XE Case Studies

**Intel® Xeon Phi™ Processor Software Ecosystem Momentum Guide
Performance results are based tests from 2016-2017 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
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 & functions. Any change to any of those factors may cause the results to vary. You should consult other information & performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/performance. See configurations in individual case study links.
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. For more complete information about compiler optimizations, see our Optimization Notice.

AI - OVERVIEW

THE AI MANDATE

"AI technologies are evolving fast and growing increasingly **critical** to firms' ability to win, serve, and retain customers."

FORRESTER*

"...strategic technologies for 2019 with the potential to drive significant **disruption** and deliver **opportunity** over the next five years"

Gartner*

"...**70%** of CIOs will aggressively apply data and AI to IT operations, tools, and processes by 2021."

IDC
Analyze the Future

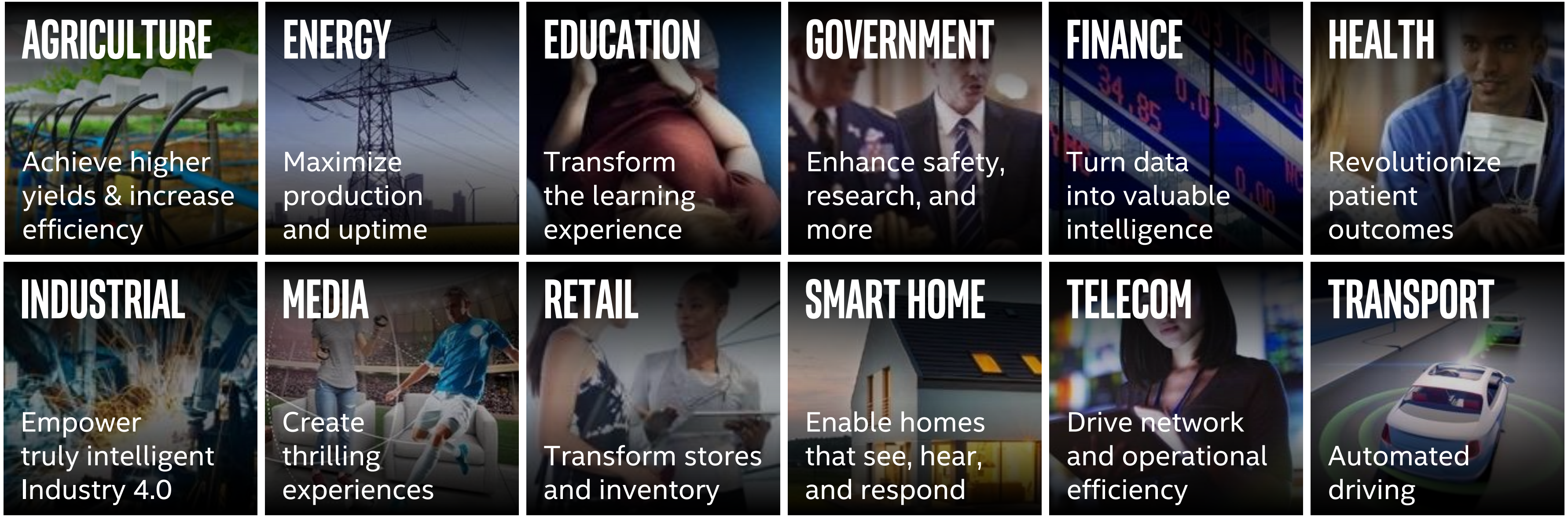
The time to begin AI adoption is now

*Other names and brands may be claimed as the property of others
Source: https://www.forrester.com/report/The+Forrester+Tech+Tide+Artificial+Intelligence+For+Business+Insights+Q3+2018/-/E-RES143252
Source: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019
Source: https://www.idc.com/getdoc.jsp?containerId=prUS44420918

Optimization Notice

intel

AI SOLUTIONS IN EVERY MARKET



Intel and our partners are driving real-world value with AI

BREAKING BARRIERS BETWEEN AI THEORY AND REALITY

Partner with Intel to accelerate your AI journey

CHOOSE ANY APPROACH
from machine to deep learning

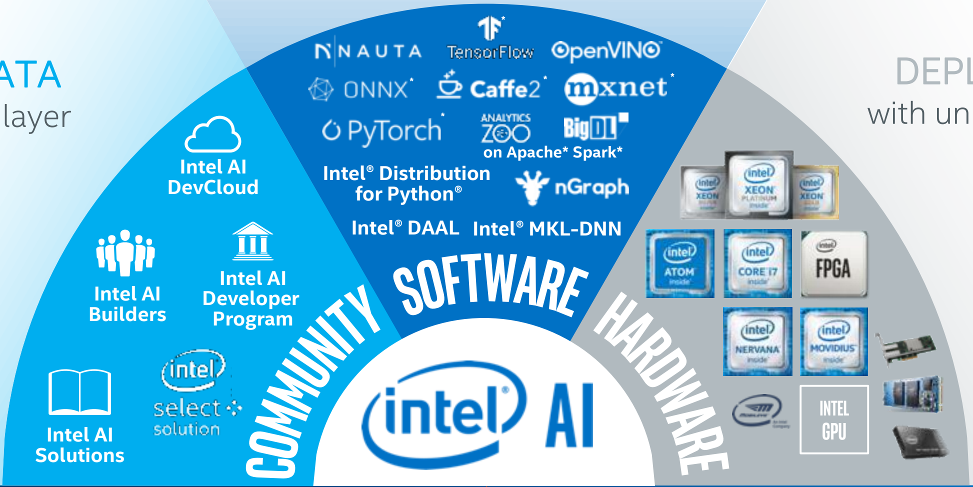
SPEED UP DEVELOPMENT
using open AI software

TAME YOUR DATA
with a robust data layer

DEPLOY AI ANYWHERE
with unprecedented HW choice

SIMPLIFY AI
using community solutions

SCALE WITH CONFIDENCE
on the platform for IT & cloud



All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.
**Other names and brands may be claimed as the property of others*

intel

ARTIFICIAL INTELLIGENCE

SOLUTIONS
Solution Architects

ARTIFICIAL INTELLIGENCE

Platforms · Finance · Healthcare · Energy · Industrial · Transport · Retail · Home · More...

TOOLKITS
App Developers

DEEP LEARNING DEPLOYMENT

OpenVINO™ †
Open Visual Inference & Neural Network Optimization toolkit for inference deployment on CPU, processor graphics, FPGA & VPU using TF, Caffe & MXNet**

Intel® Movidius™ SDK
Optimized inference deployment for all Intel® Movidius™ VPUs using TensorFlow & Caffe**

DEEP LEARNING
COMING SOON!

Intel® Deep Learning Studio‡
Open-source tool to compress deep learning development cycle

LIBRARIES
Data Scientists

MACHINE LEARNING LIBRARIES

Python
- Scikit-learn
- Pandas
- NumPy

R
- Cart
- Random Forest
- e1071

Distributed
- MlLib (on Spark)
- Mahout

DEEP LEARNING FRAMEWORKS

Now optimized for CPU

TensorFlow* MXNet* Caffe* BigDL/Spark*

Optimizations in progress
COMING SOON!

Caffe2* PyTorch* PaddlePaddle*

FOUNDATION
Library Developers

ANALYTICS, MACHINE & DEEP LEARNING PRIMITIVES

Python
Intel distribution optimized for machine learning

DAAL
Intel® Data Analytics Acceleration Library (for machine learning)

MKL-DNN
Open-source deep neural network functions for CPU, processor graphics

DEEP LEARNING GRAPH COMPILER

Intel® nGraph™ Compiler (Alpha)
Open-sourced compiler for deep learning model computations optimized for multiple devices (CPU, GPU, NNP) using multiple frameworks (TF, MXNet, ONNX)

HARDWARE
IT System Architects

AI FOUNDATION

ATOM · CORE 8th Gen · Iris Graphics · XEON PLATINUM

Data Center
Edge
Device

DEEP LEARNING ACCELERATORS

NERVANA (COMING 2019) · STRATIX 10 · ARRIA 10 · MOVIDIUS · Mobileye · GNA

NNP L-1000

Inference

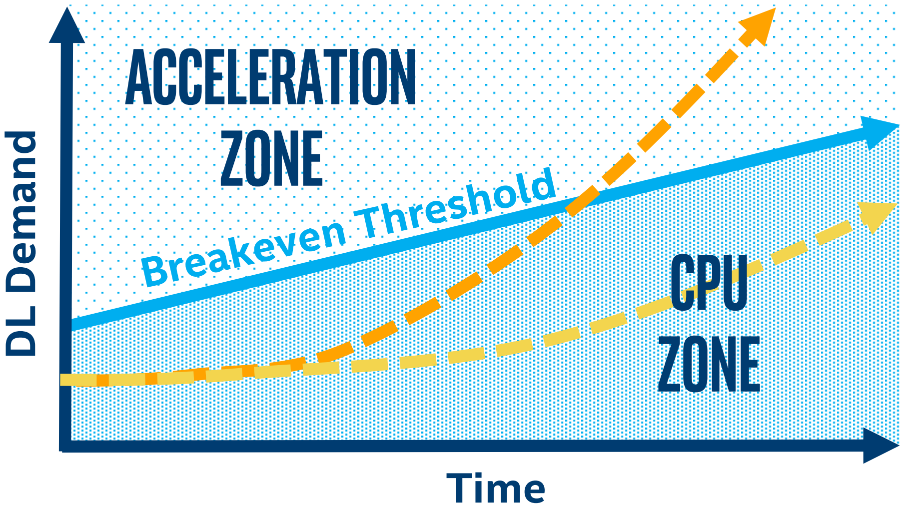
† Formerly the Intel® Computer Vision SDK
*Other names and brands may be claimed as the property of others.
All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

AI.INTEL.COM

intel

THE DEEP LEARNING MYTH

"A GPU is required for deep learning…" **FALSE**



- **Most businesses (---)** use the CPU for machine & deep learning needs
- **Some early adopters (---)** may reach a deep learning tipping point when acceleration is needed[1]

[1]"Most businesses" claim is based on survey of Intel direct engagements and internal market segment analysis

DEEP LEARNING IN PRACTICE

Source Paper:
research.fb.com/wpcontent/uploads/2017/12/hpca-2018-facebook.pdf


Applied Machine Learning at Facebook:
A Datacenter Infrastructure Perspective

Services	Ranking Algorithm	Photo Tagging	Photo Text Generation	Search	Language Translation	Spam Flagging	Speech
Model(s)	MLP	SVM,CNN	CNN	MLP	RNN	GBDT	RNN
Inference Resource	CPU	CPU	CPU	CPU	CPU	CPU	CPU
Training Resource	CPU	GPU & CPU	GPU	Depends	GPU	CPU	GPU
Training Frequency	Daily	Every N photos	Multi-Monthly	Hourly	Weekly	Sub-Daily	Weekly
Training Duration	Many Hours	Few Seconds	Many Hours	Few Hours	Days	Few Hours	Many Hours

Large cloud users employ CPU extensively for deep learning

Software

INTEL® DISTRIBUTION FOR PYTHON 2019

The most popular languages for Data Science

"**Python wins the heart of developers** across all ages, according to our Love-Hate index. Python is also the most popular language that **developers want to learn** overall, and a **significant share already knows it**"

HackerRank

2018 Developer Skills Report

- Python, Java, R are top 3 languages in job postings for data science and machine learning jobs
  - https://www.kdnuggets.com/2017/01/most-popular-language-machine-learning-data-science.html

KDnuggets

The most popular ML/DL packages for Python

python™

SciPy

TensorFlow

scikit learn

theano

dmlc mxnet

jupyter

NumPy

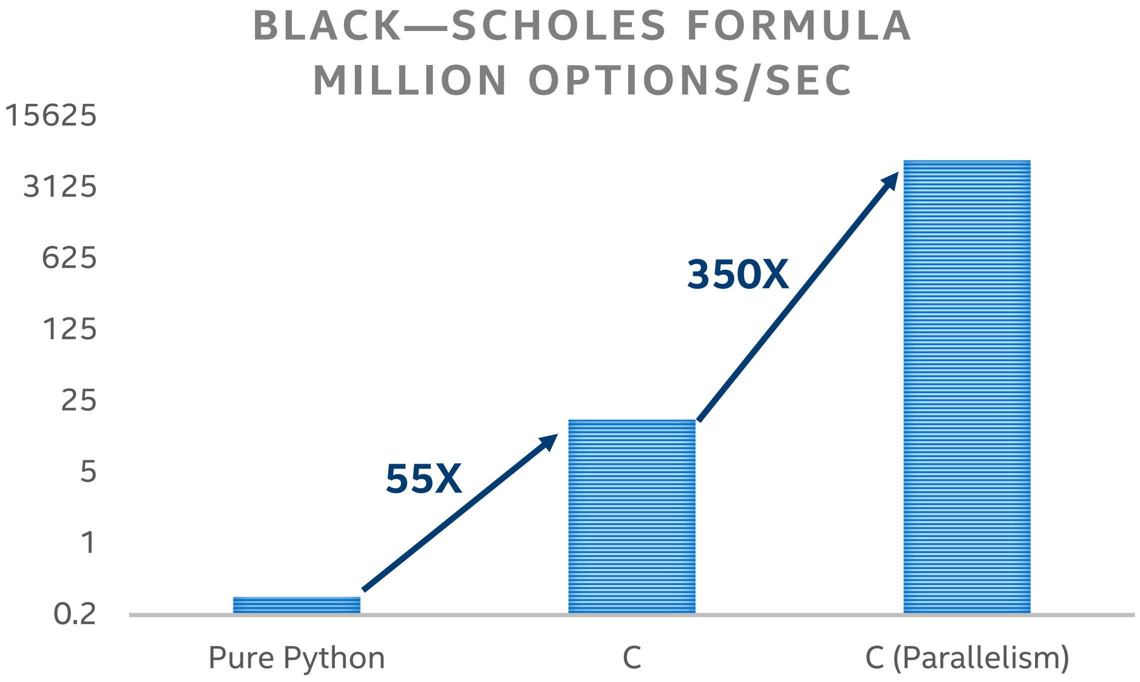
pandas

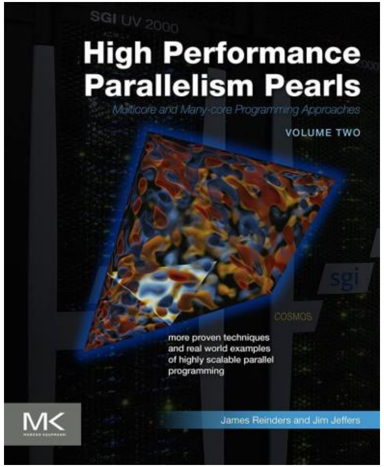
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Keras

matplotlib

Performance gap between C and Python





Chapter 19: Performance Optimization of **Black—Scholes** Pricing

$$V_{call} = S_0 \cdot \mathrm{CDF}(d_1) - e^{-rT} \cdot X \cdot \mathrm{CDF}(d_2)$$
$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$

$$d_1 = \frac{\ln\left(\frac{S_0}{X}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln\left(\frac{S_0}{X}\right) + \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}$$

What's Inside Intel® Distribution for Python 2019

High Performance Python* for Scientific Computing, Data Analytics, Machine Learning

FASTER PERFORMANCE

Performance Libraries, Parallelism, Multithreading, Language Extensions

Accelerated **NumPy/SciPy/scikit-learn** with Intel® MKL[1] & Intel® DAAL[2]

Data analytics, machine learning & deep learning with scikit-learn, pyDAAL

Scale with Numba* & Cython*

Includes optimized **mpi4py**, works with Dask* & PySpark*

Optimized for latest Intel® architecture

GREATER PRODUCTIVITY

Prebuilt & Accelerated Packages

Prebuilt & optimized packages for numerical computing, machine/deep learning, HPC, & data analytics

Drop in replacement for existing Python - No code changes required

Jupyter* notebooks, Matplotlib included

Conda build recipes included in packages

Free download & free for all uses including commercial deployment

ECOSYSTEM COMPATIBILITY

Supports Python 2.7 & 3.x, conda, pip

Compatible & powered by Anaconda*, supports **conda** & pip

Distribution & individual optimized packages also available via conda, pip YUM/APT, Docker image on DockerHub

Optimizations upstreamed to main Python trunk

Commercial support through Intel® Parallel Studio XE

Intel® Architecture Platforms

Operating System: Windows*, Linux*, MacOS[1]*

[1]Intel® Math Kernel Library
[2]Intel® Data Analytics Acceleration Library

[1] Available only in Intel® Parallel Studio Composer Edition.

What’s New for 2019?

Intel® Distribution for Python*

Faster Machine learning with Scikit-learn functions

- Support Vector Machine (SVM) and K-means prediction, accelerated with Intel® DAAL

Built-in access to XGBoost library for Machine Learning

- Access to Distributed Gradient Boosting algorithms

Ease of access installation

- Now integrated into Intel® Parallel Studio XE installer.

Access Intel-optimized Python packages through

PARALLEL STUDIO XE

ACCELERATE PYTHON* PERFORMANCE
POWERED BY ANACONDA*

Standalone Python Distribution

CONDA
Intel optimized packages via conda

python Package Index

Docker Hub
docker

YUM/APT
repositories

Speedup Analytics & Machine Learning with Intel® Data Analytics Acceleration Library (Intel® DAAL)

- Highly tuned functions for classical machine learning & analytics performance from datacenter to edge running on Intel® processor-based devices
- Simultaneously ingests data & computes results for highest throughput performance
- Supports batch, streaming & distributed usage models to meet a range of application needs
- Includes Python*, C++, Java* APIs, & connectors to popular data sources including Spark* & Hadoop

What's New in the 2019 Release

New Algorithms

- **Logistic Regression**, most widely-used classification algorithm
- **Extended Gradient Boosting Functionality** for inexact split calculations & user-defined callback canceling for greater flexibility
- **User-defined Data Modification Procedure** supports a wide range of feature extraction & transformation techniques

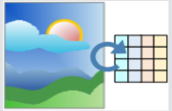
Learn More: software.intel.com/daal

Pre-processing → **Transformation** → **Analysis** → **Modeling** → **Validation** → **Decision Making**

Pre-processing

Decompression, Filtering, Normalization

Transformation



Aggregation, Dimension Reduction

Analysis

Summary Statistics Clustering, etc.

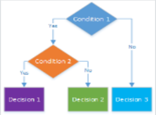
Modeling

Machine Learning (Training) Parameter Estimation Simulation

Validation

Hypothesis Testing Model Errors

Decision Making



Forecasting Decision Trees, etc.

Algorithms, Data Transformation & Analysis

Intel® Data Analytics Acceleration Library

Basic Statistics for Datasets
- Low Order Moments
- Quantiles
- Order Statistics

Correlation & Dependence
- Cosine Distance
- Correlation Distance
- Variance-Covariance Matrix

Matrix Factorizations
- SVD
- QR
- Cholesky

Dimensionality Reduction
- PCA

Outlier Detection
- Univariate
- Multivariate

Association Rule Mining (Apriori)

Optimization Solvers (SGD, AdaGrad, lBFGS)

Math Functions (exp, log,…)

Algorithms supporting batch processing

Algorithms supporting batch, online and/or distributed processing

LIVE DEMO – MEDICAL SEGMENTATION AND BRAIN TUMOUR PREDICTION

Login to the machine

Make sure you have internet connection and open the following link:

https://tinyurl.com/ep19intel

Mac and Linux users:

1. Download the file: `gcp.key` (this is your private key to allow access to your individual virtual machine).
   - Note, make sure the content starts with: -----BEGIN RSA PRIVATE KEY-----
2. Open the `GCP_IP_Addresses` sheet and keep the IP addresses assigned to you. You should have received a number from me.
3. `chmod 600 gcp.key`
4. Connect to the VM: `ssh user01@<IP_address> -i gcp.key`

Windows users:

Use PuTTy and the user01.ppk file.

If you see this…it worked

```
Using username "user01".
Authenticating with public key "user01"
Last login: Tue Jul  9 08:31:21 2019 from ........Some IP address
Intel(R) Parallel Studio XE 2019 Update 4 for Linux*
Copyright (C) 2009-2019 Intel Corporation. All rights reserved.
[setupvars.sh] OpenVINO environment initialized
(base) [user01@medical-isc2019-vm ~]$
```

Start the Jupyter Server

Copy paste the following in the terminal:

```
jupyter notebook --no-browser --ip 0.0.0.0 --port 8888 &
```

After that, open the Jupyter notebook in your browser of choice:

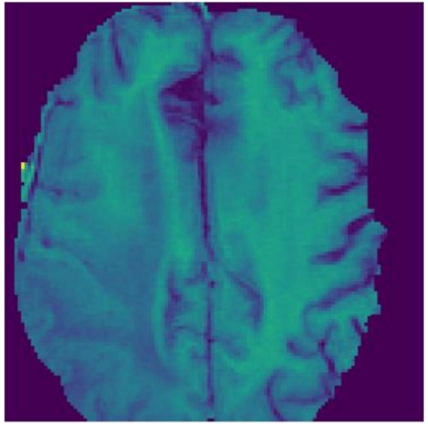
<IP_ADDRESS>:8888

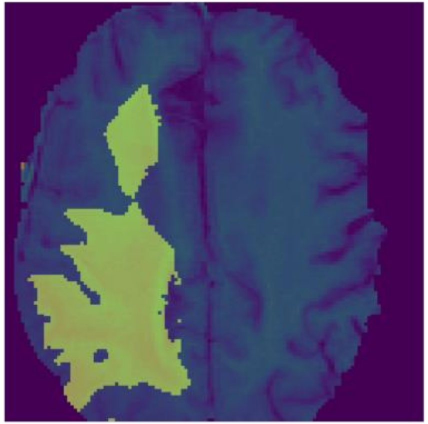
Password(if any): `intel123`

A bit of statistics

As per Globocan 2018 (Global Cancer statistics):

- There were **18.1** million new cancer cases
- and **9.6** million cancer deaths
- in 2018





Statistics source:
https://onlinelibrary.wiley.com/doi/full/10.3322/caac.21492
(36 cancers in 185 countries)

Introduction

- **Gliomas** are the most commonly occurring type of **brain tumors**
  - and are potentially very dangerous
  - with about 90% of Gliomas belonging to a highly aggressive class of cancerous tumors
- Multi-sequence **Magnetic Resonance Imaging (MRI)** is the primary method of screening and diagnosis for Gliomas

Segmenting the brain tumor

- To assess the severity/for treatment of the tumour, segmentation is important for:
  - focusing on the tumour areas during radiotherapy
  - navigation during surgery

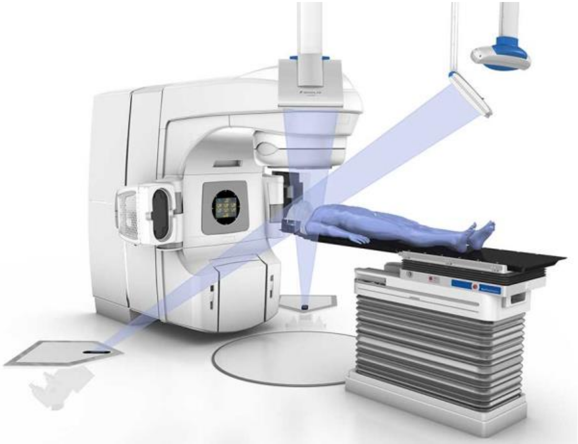


Image sources: Brainlab

The medical challenge

- Not enough expert doctors to analyze all the medical data[1]
- Tumor regions segmentation is time-consuming and expensive

Sources:
[1] https://www.diagnosticimaging.com/residents/physician-shortage-too-many-radiologists
[2]Corbin K. How CIOs Can Prepare for Healthcare “Data Tsunami” [Internet]. CIO. 2014 [cited 8 FEB 2019].
[3] Fenton SH, Low S, Abrams KJ, Butler-Henderson K. Health Information Management: Changing with Time. IMIA Yearbook of Medical Informatics 2017.
[4]Stanford Medicine. 2017 Health Trends Report: Harnessing the Power of Data in Health. Accessed online 8 FEB 2019.

But...

...machines ca

One exabyte is one billion gigabytes
or
250 000 000 DVDs worth of information.

Automating the process:

- helps gain of time f
- gives time back to t
- improves segmentation qua

- Nearly 153 exabytes of healthcare ... data were generated in 2013
  - amount to increase by 48% annually
  - expected to reach 2,314 exabytes in 2020 [1], [2], [3]

The dataset for the deep learning algorithm

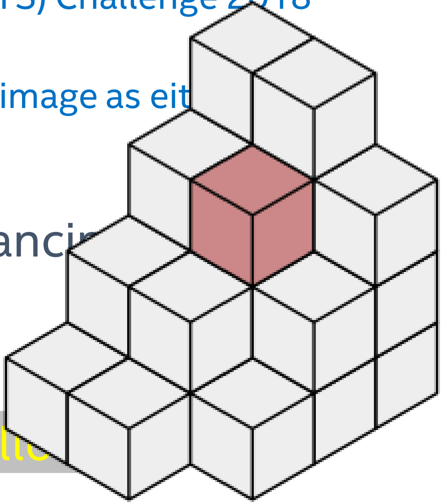
- Brain Tumor Segmentation (BraTS) Challenge 2018 dataset
- **Goal**: classify every **voxel** in the image as eit

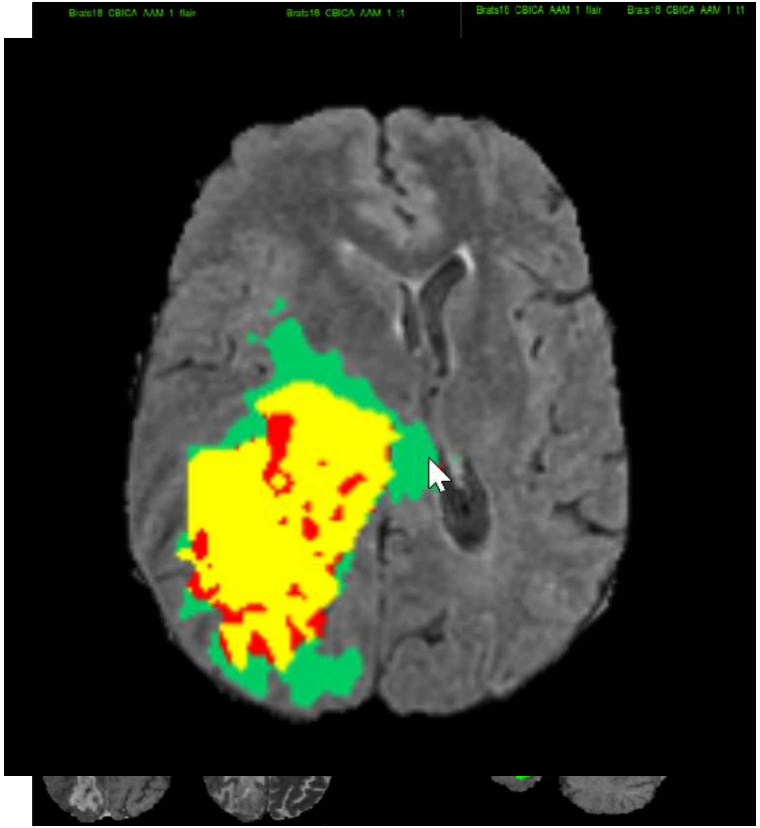
  i. healthy tissue

  ii. necrosis or non-enhanci (red)

  iii. edema (green) or

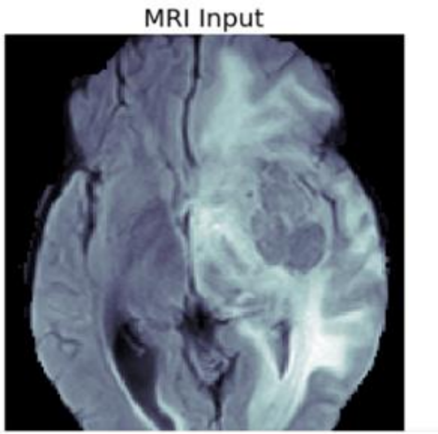
  iv. enhancing tumor (yell





Further dataset details

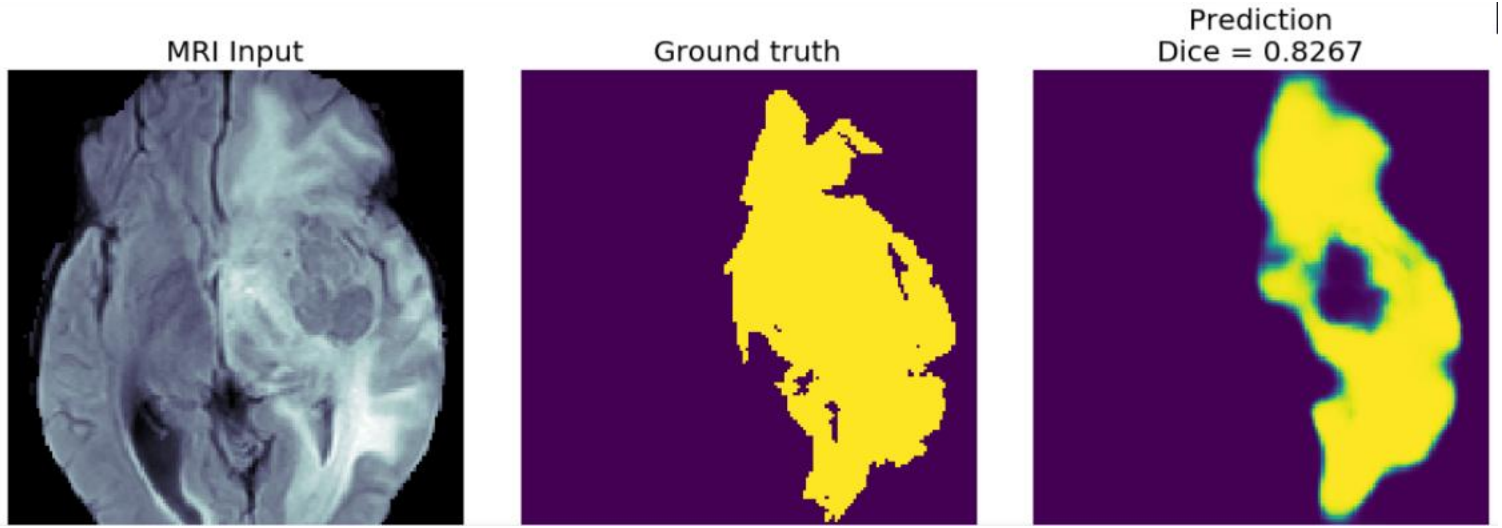
- Training dataset: Images of 220 high-grade glioma (HGG), 54 low-grade glioma (LGG) patients [1]
- Image size: 240X240X155 voxels, contain 4-channels



Sources:
[1] https://arxiv.org/ftp/arxiv/papers/1702/1702.04528.pdf

The result of our deep learning algorithm

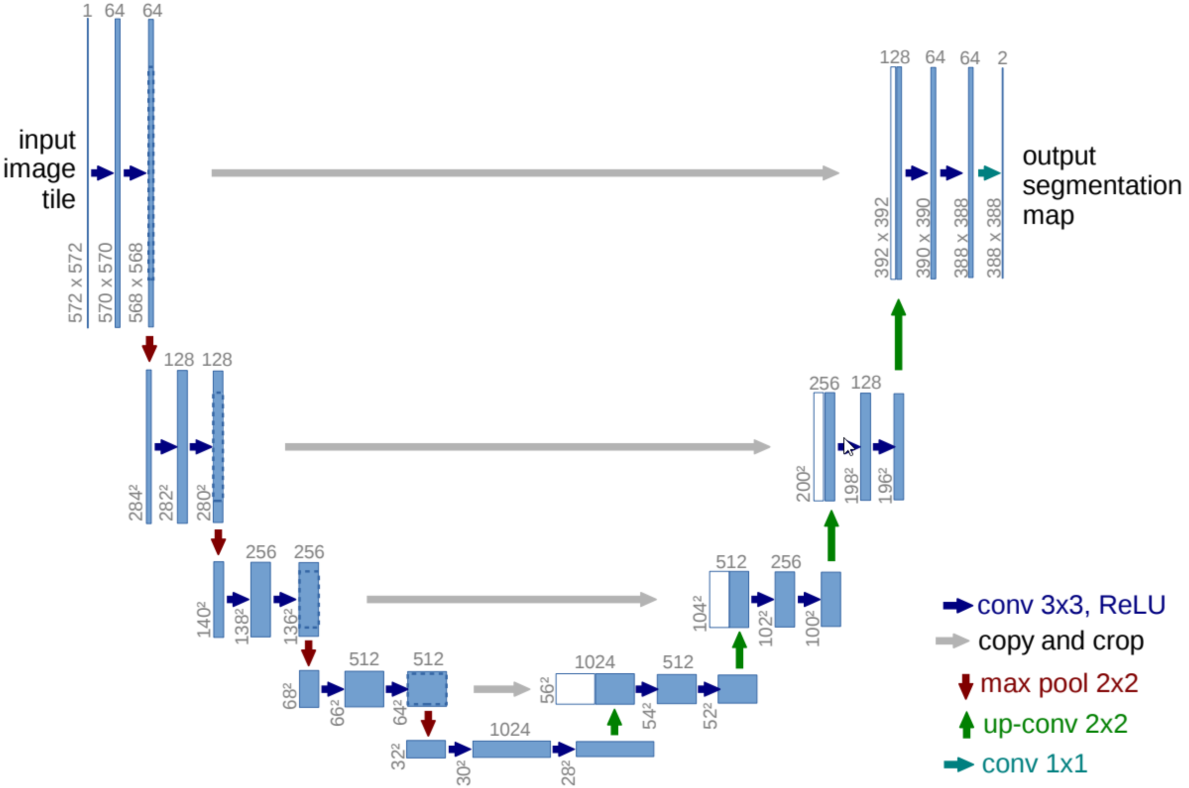
Example segmentation has been prepared for to compare with target (expert's) segmentation.



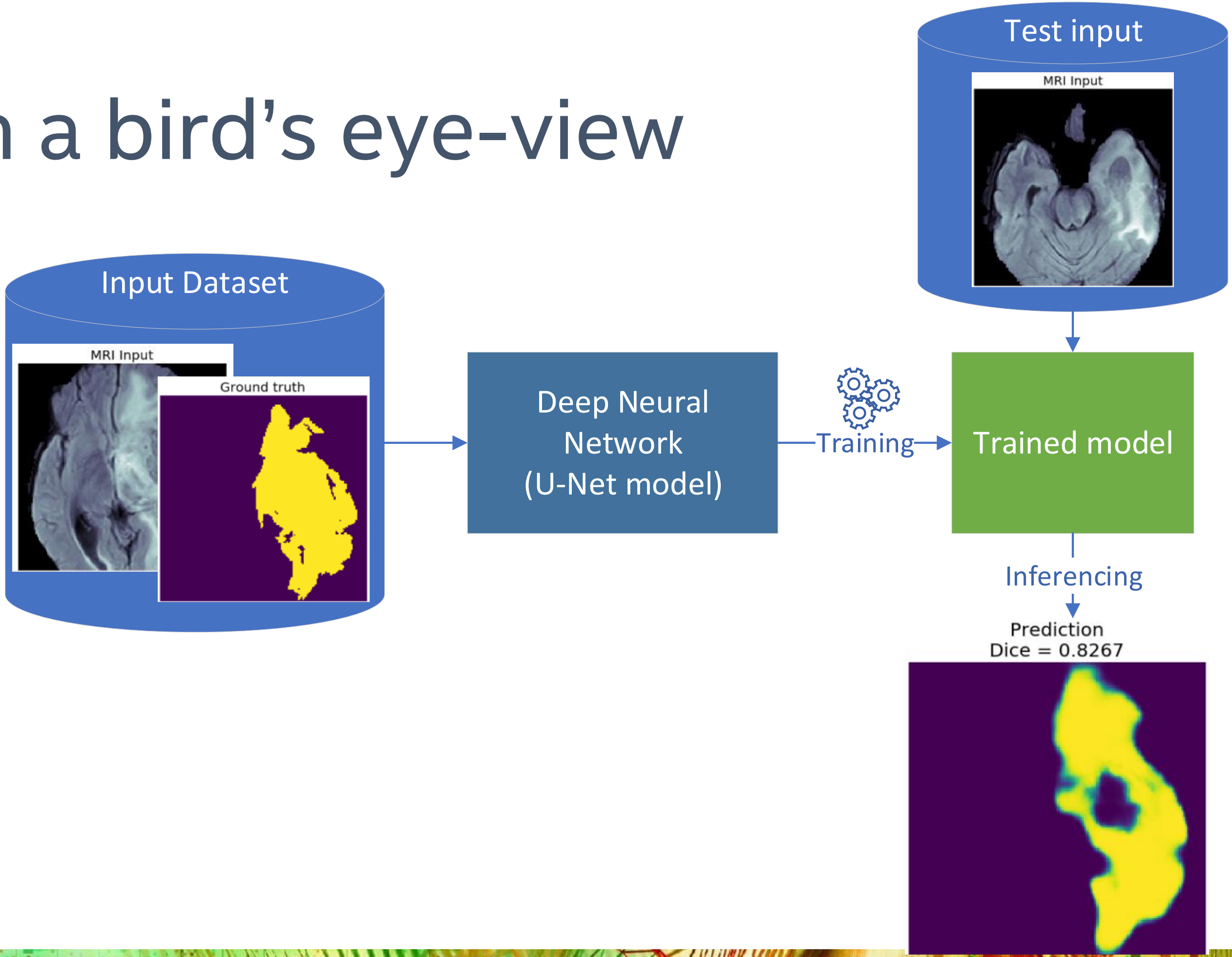
The algorithm used (U-Net model)

- Has an **encoding path** ("contracting") paired with a **decoding path** ("expanding")

- For each pixel in the original image, it asks the question: **"To which class does this voxel belong?"**



From a bird's eye-view



What software tools did we use in this project?

Intel® Distribution for Python

Frameworks and Software Stack

How did we boost the performance of the algorithm?

Thanks to Intel® Technologies:

Python-based Deep Learning Demo application

Intel® Distribution for Python

python™

Keras

HOROVOD

TensorFlow

OpenVINO™

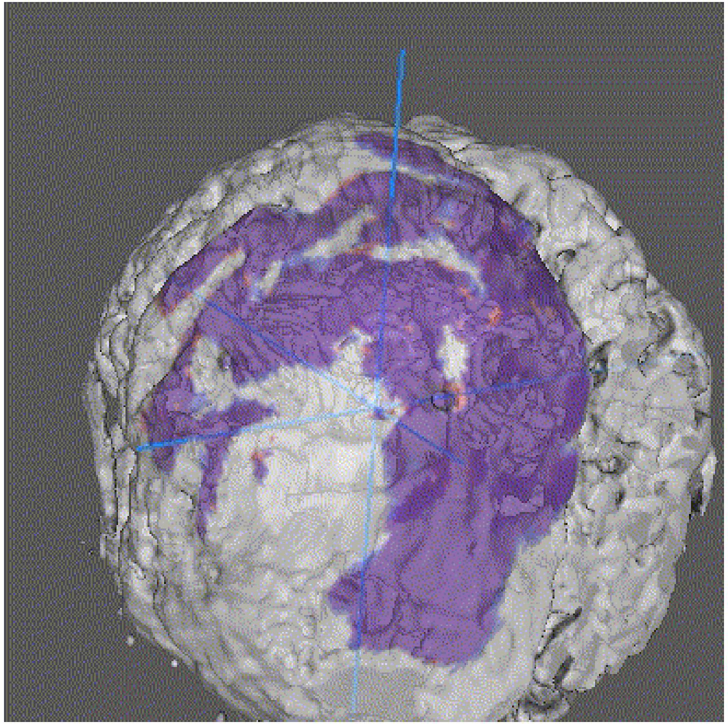
Frameworks and Software Stack

Intel® Math Kernel Library – Deep Neural Network (Intel® MKL-DNN)

The base hardware

intel XEON PLATINUM inside

Visualization of the end result in 3D



Code source

https://github.com/shailensobhee/medical-decathlon

From the GitHub link:

- Code source
- instructions on how to get the medical dataset

HANDS-ON ACTIVITY

Project files description

`Train.ipynb`
`Inference.ipynb`

`model.py` – model description
`settings.py` – project settings
`argsparser.py` – see how `argsparser` picks up the settings.

OPTIMIZATION TECHNIQUES

Parallelism parameters

- `inter_op_parallelism_threads`
  independent ops on how many cores?

- `intra_op_parallelism_threads`
  one op on how many cores?

Parallelism considerations

KMP_BLOCKTIME	Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.
KMP_AFFINITY	Enables run-time library to bind threads to physical processing units.
OMP_NUM_THREADS	Sets the maximum number of threads to use for OpenMP* parallel regions if no other value is specified in the application. Default: Number of processors visible to the operating system.

https://www.tensorflow.org/guide/performance/overview

Tuning MKL for the best performance

The MKL is optimized for the **NCHW** (`channels_first`) data format and Intel is working to get near performance parity when using **NHWC**.
MKL uses the following environment variables to tune performance:

- `KMP_BLOCKTIME` - Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.
- `KMP_AFFINITY` - Enables the run-time library to bind threads to physical processing units.
- `KMP_SETTINGS` - Enables (true) or disables (false) the printing of OpenMP* run-time library environment variables during program execution.
- `OMP_NUM_THREADS` - Specifies the number of threads to use.

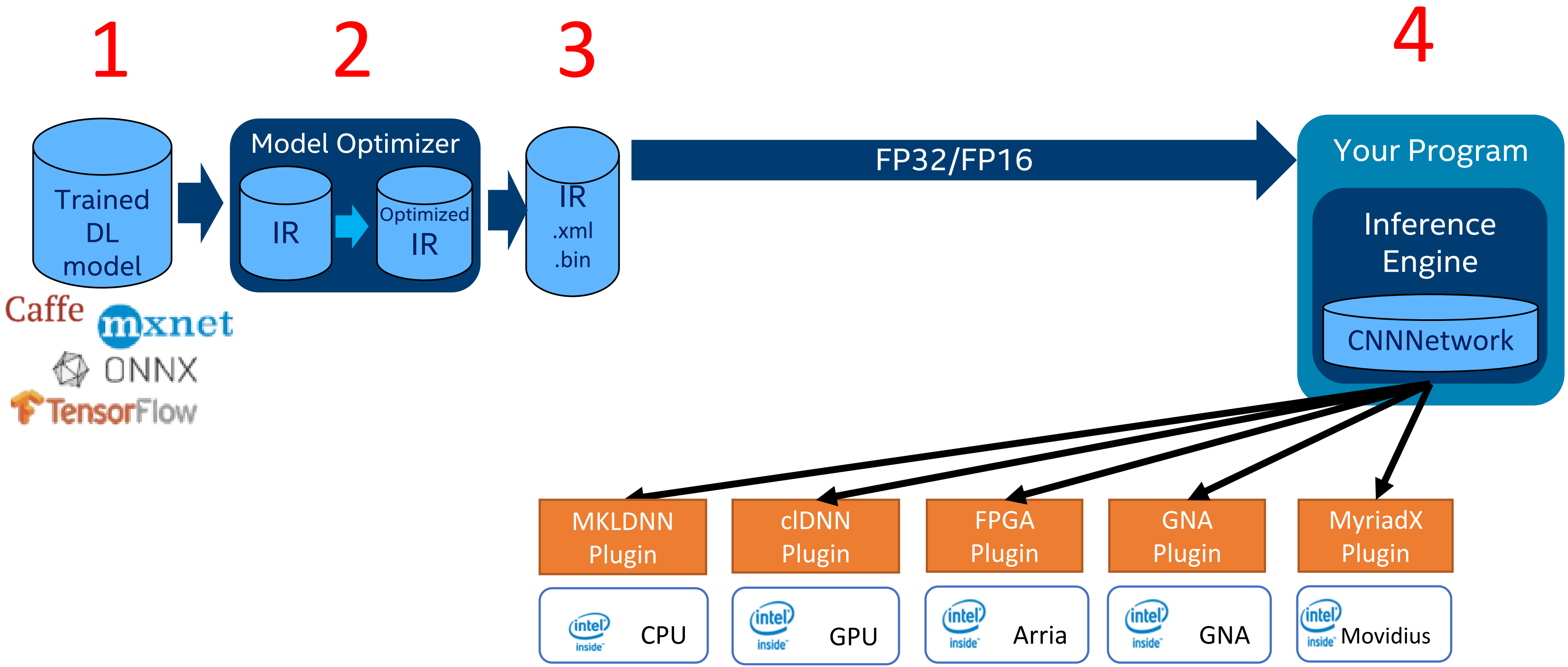
What it is?

A toolkit to accelerate development of **high performance computer vision** & **deep learning into vision applications** from device to cloud. It enables deep learning on hardware accelerators and easy deployment across multiple types of Intel® platforms.

Free Download ▸ software.intel.com/openvino-toolkit

Open Source version ▸ 01.org/openvinotoolkit

Intel® Distribution of OpenVINO™ in a nutshell



BACKUP

KEY PERFORMANCE CONSIDERATIONS ON INTEL PROCESSORS

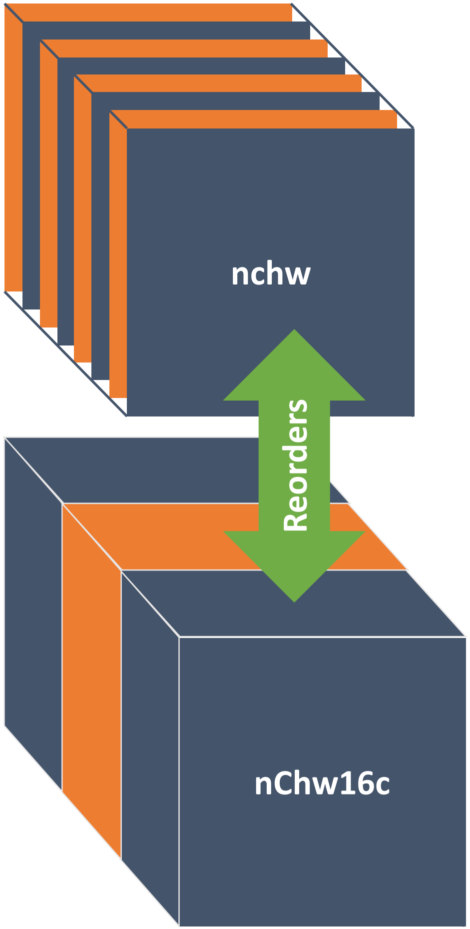
MEMORY LAYOUTS

Most popular memory layouts for image recognition are **nhwc** and **nchw**

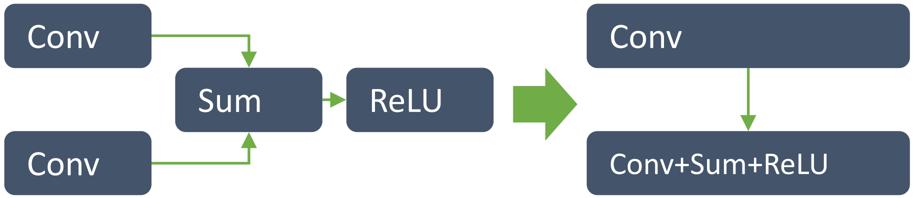
- Challenging for Intel processors either for vectorization or for memory accesses (cache thrashing)

Intel MKL-DNN convolutions use blocked layouts

- Example: **nhwc** with channels blocked by 16 – **nChw16c**
- Convolutions define which layouts are to be used by other primitives
- Optimized frameworks track memory layouts and perform reorders **only** when necessary



Fusing computations



On Intel processors a high % of time is typically spent in BW-limited ops

- ~40% of ResNet-50, even higher for inference

The solution is to fuse BW-limited ops with convolutions or one with another to reduce the # of memory accesses

- Conv+ReLU+Sum, BatchNorm+ReLU, etc
- Done for inference, WIP for training

The FWKs are expected to be able to detect fusion opportunities

- IntelCaffe already supports this

Major impact on implementation

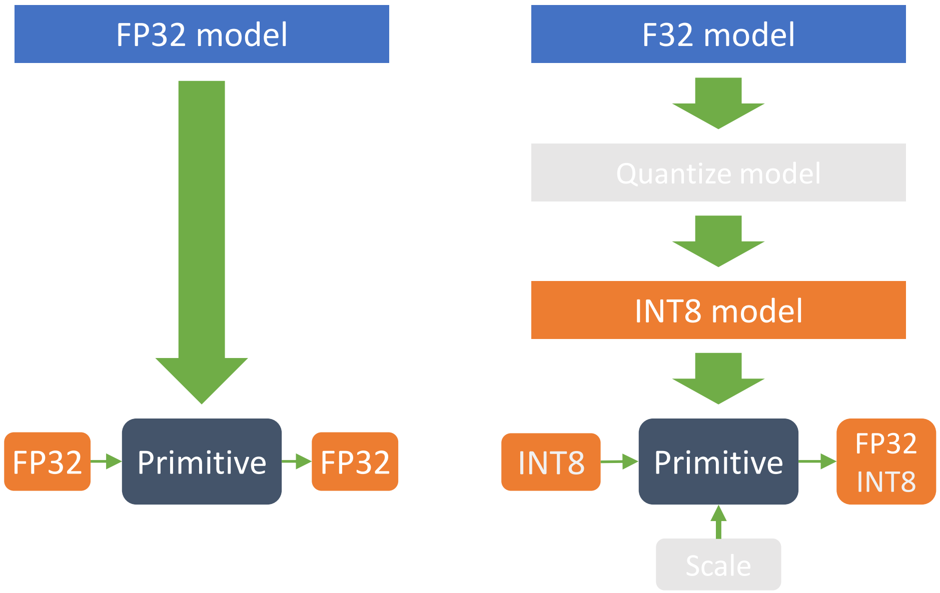
- All the impls. must be made aware of the fusion to get max performance
- Intel MKL-DNN team is looking for scalable solutions to this problem

LOW-PRECISION INFERENCE

Proven only for certain CNNs by IntelCaffe at the moment

A trained float32 model quantized to int8

Some operations still run in float32 to preserve accuracy



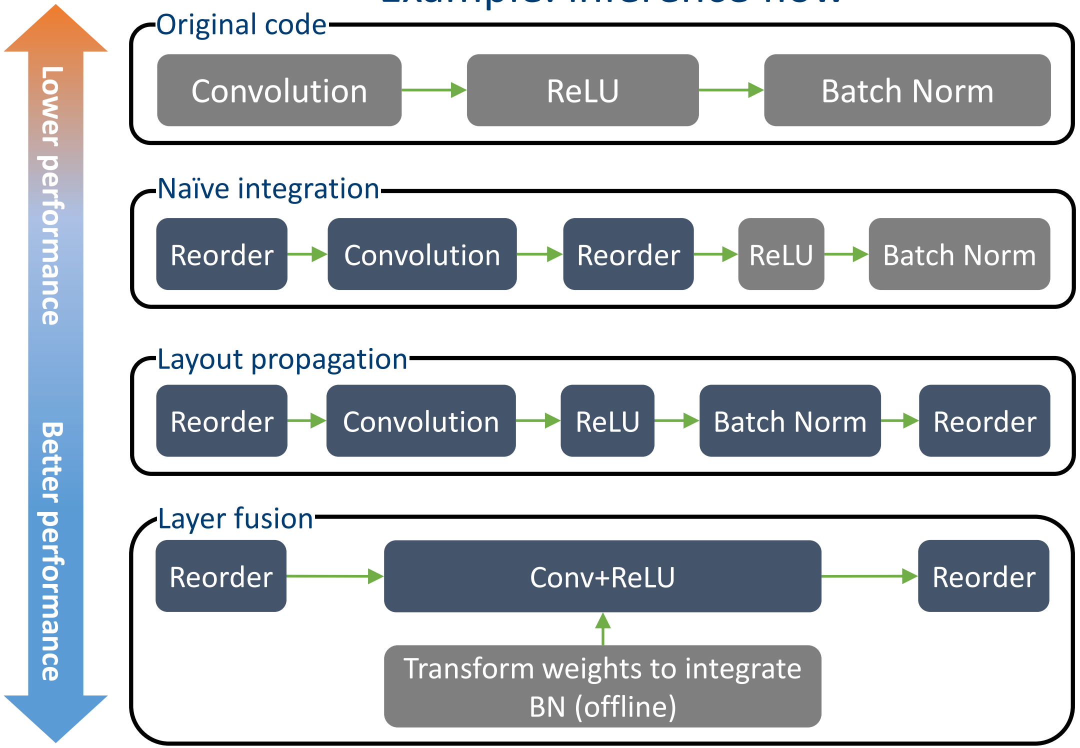
Intel MKL-DNN integration levels

Intel MKL-DNN is designed for best performance.

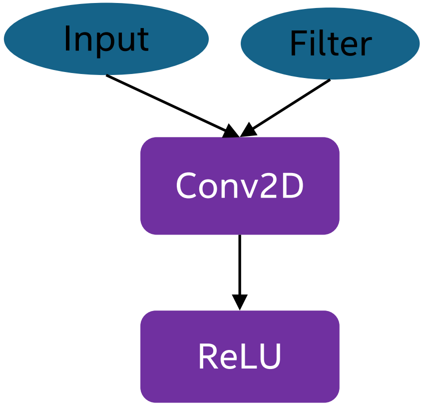
However, topology level performance will depend on Intel MKL-DNN integration.

- Naïve integration will have reorder overheads.
- Better integration will propagate layouts to reduce reorders.
- Best integration will fuse memory bound layers with compute intensive ones or with each other.

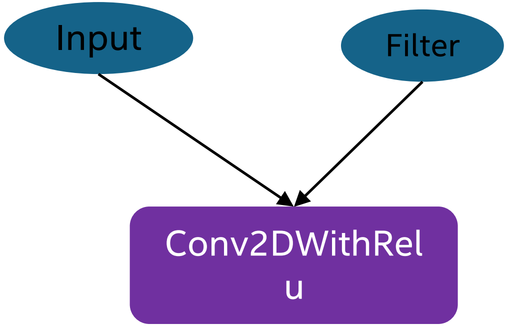
Example: inference flow



Graph optimizations: fusion

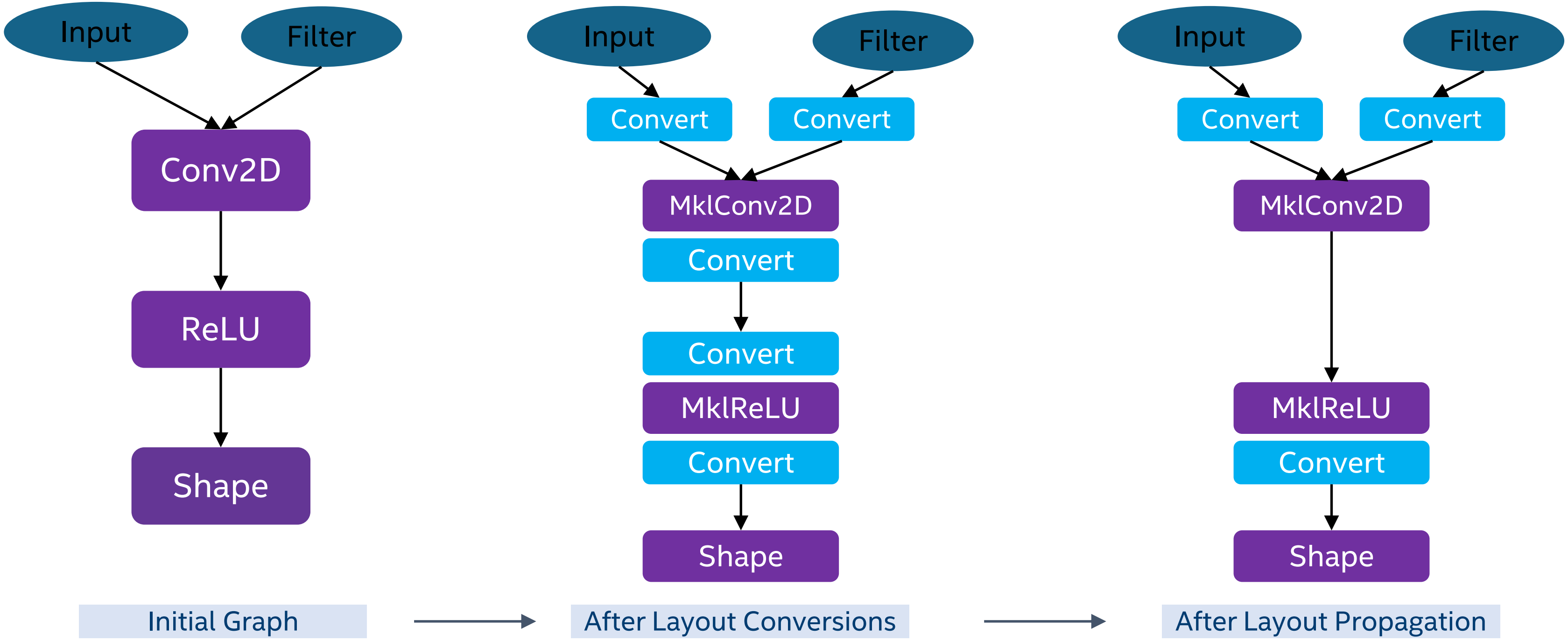


Before Merge



After Merge

Graph optimizations: layout Conversion



- All MKL-DNN operators use highly-optimized layouts for TensorFlow tensors.

BACKUP 2

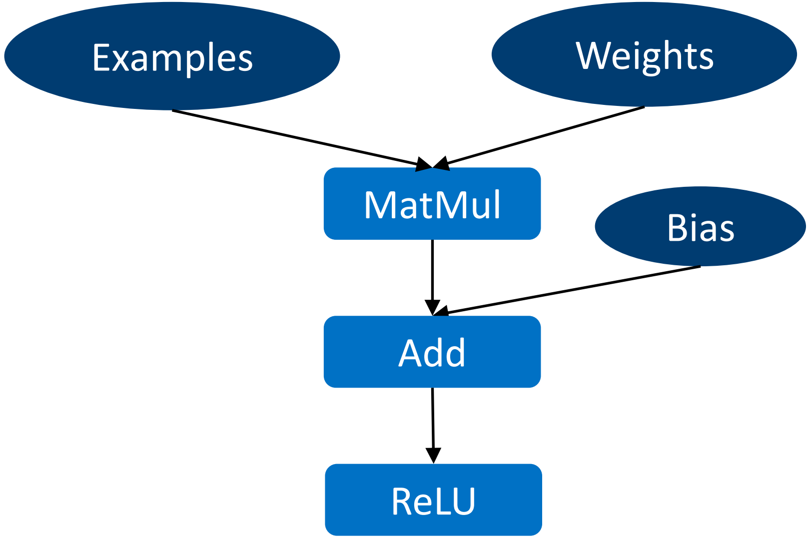
INTEL® TENSORFLOW OPTIMIZATIONS

INTEL-TENSORFLOW OPTIMIZATIONS

1. Operator optimizations
2. Graph optimizations
3. System optimizations

OPERATOR OPTIMIZATIONS

In TensorFlow, computation graph is a data-flow graph.

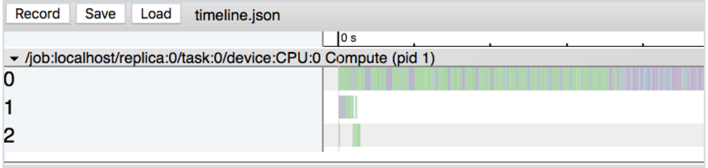


OPERATOR OPTIMIZATIONS

- Replace default (Eigen) kernels by highly-optimized kernels (using Intel® MKL-DNN)
- Intel® MKL-DNN has optimized a set of TensorFlow operations.
- Library is open-source (https://github.com/intel/mkl-dnn) and downloaded automatically when building TensorFlow.

Forward	Backward
`Conv2D`	`Conv2DGrad`
`Relu, TanH, ELU`	`ReLUGrad, TanHGrad, ELUGrad`
`MaxPooling`	`MaxPoolingGrad`
`AvgPooling`	`AvgPoolingGrad`
`BatchNorm`	`BatchNormGrad`
`LRN`	`LRNGrad`
`MatMul, Concat`	

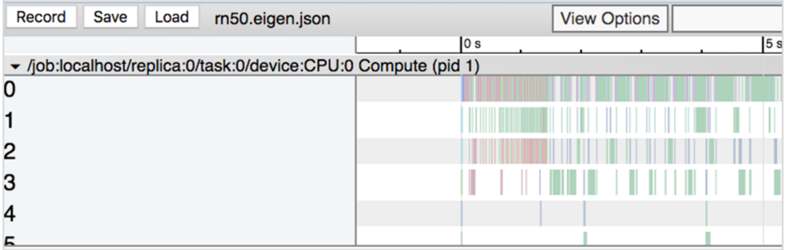
OPERATOR OPTIMIZATIONS IN RESNET50



3005 items selected. Slices (3005)

Name	Wall Duration	Self time
_MklConv2DBackpropFilter	545.502 ms	545.502 ms
_MklConv2DBackpropInput	440.090 ms	440.090 ms
_MklConv2D	391.094 ms	391.094 ms
_MklFusedBatchNormGrad	184.920 ms	184.920 ms
_MklFusedBatchNormWithRelu	158.366 ms	158.366 ms
_MklReluGrad	155.874 ms	155.874 ms
_MklAdd	109.858 ms	109.858 ms
_MklAddN	103.248 ms	103.248 ms
Slice	84.905 ms	84.905 ms
Pad	38.684 ms	38.684 ms
ApplyMomentum	32.977 ms	32.977 ms
L2Loss	28.264 ms	28.264 ms
_MklToTf	22.379 ms	22.379 ms
VariableV2	19.422 ms	19.422 ms

Intel-optimized TensorFlow timeline

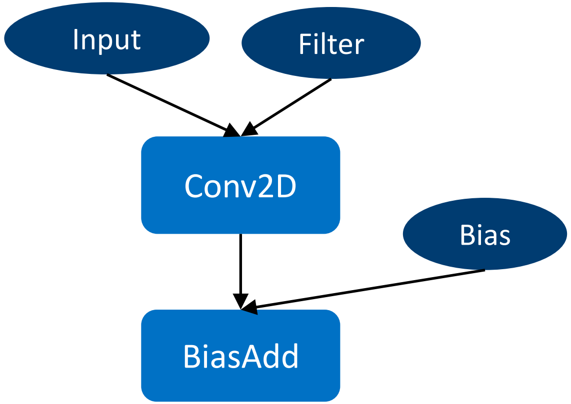


1490 items selected. Slices (1490)

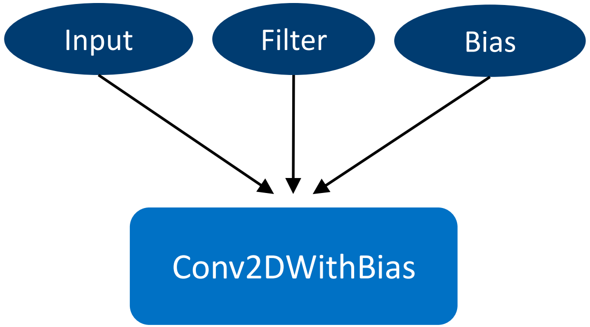
Name	Wall Duration	Self time
FusedBatchNormGrad	7,933.108 ms	7,933.108 ms
Conv2DBackpropInput	3,139.385 ms	3,139.385 ms
Conv2DBackpropFilter	2,539.365 ms	2,539.365 ms
FusedBatchNorm	873.292 ms	873.292 ms
Conv2D	640.633 ms	640.633 ms
ReluGrad	74.733 ms	74.733 ms
AddN	68.955 ms	68.955 ms
Add	38.213 ms	38.213 ms
Relu	38.010 ms	38.010 ms

Default TensorFlow timeline

GRAPH OPTIMIZATIONS: FUSION



Before Merge



After Merge

GRAPH OPTIMIZATIONS: FUSION

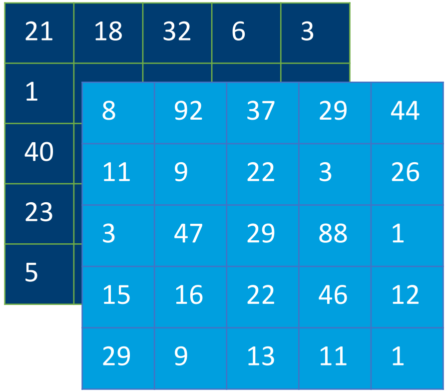
Before Merge

After Merge

GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

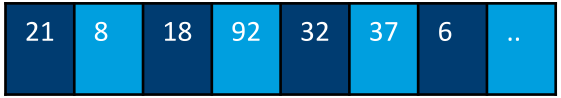
What is layout?

- How do we represent N-D tensor as a 1-D array.



{N:2, R:5, C:5}

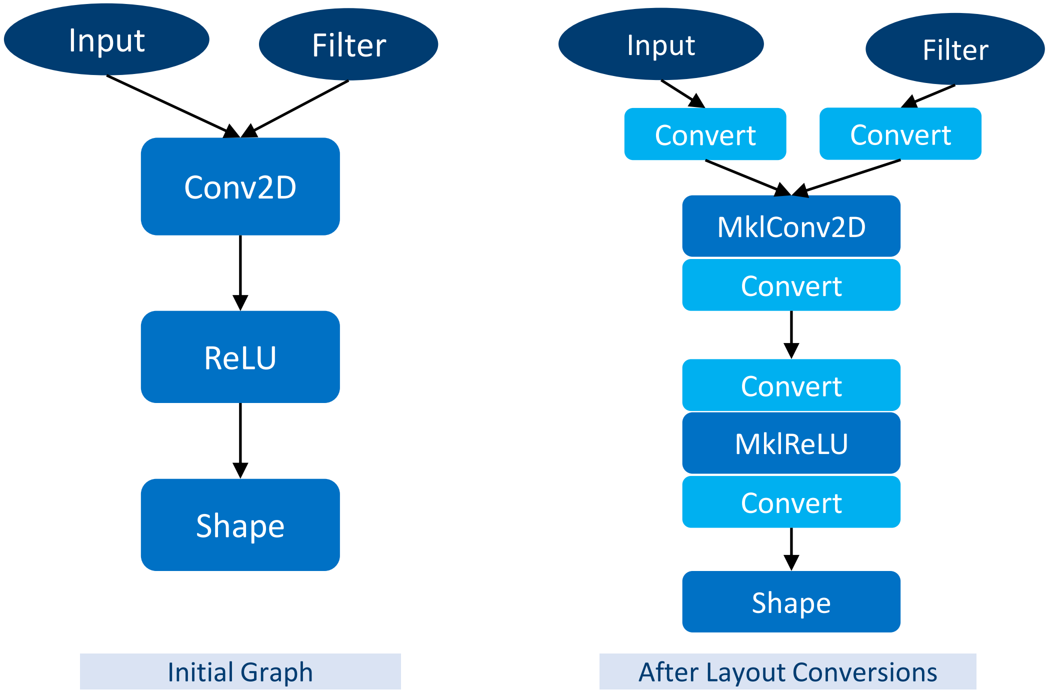
Better optimized for
some operations
vs.



GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

Converting to/from optimized layout can be less expensive than operating on un-optimized layout.

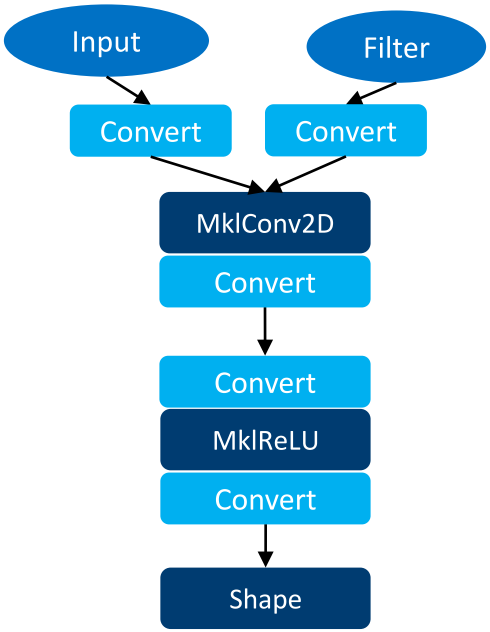
All MKL-DNN operators use highly-optimized layouts for TensorFlow tensors.



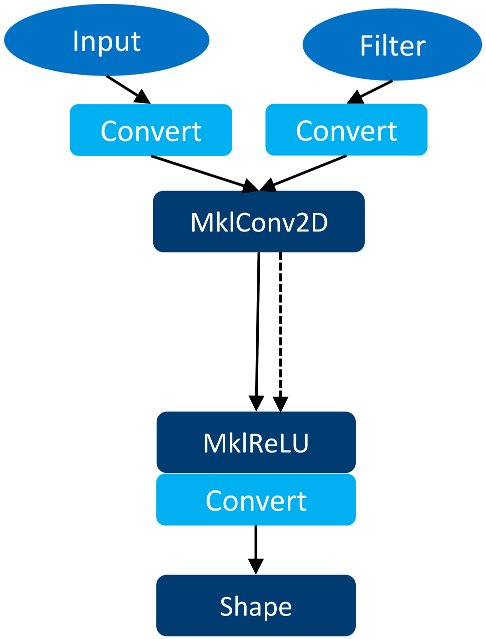
GRAPH OPTIMIZATIONS: LAYOUT PROPAGATION

Did you notice anything wrong with previous graph?

Problem: redundant conversions



After Layout Conversion



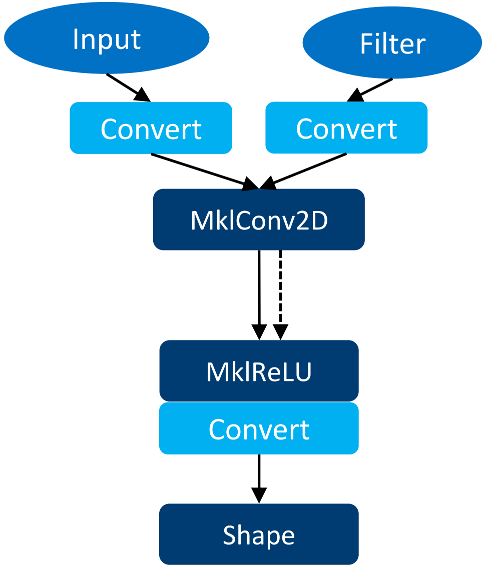
After Layout Propagation

SYSTEM OPTIMIZATIONS: LOAD BALANCING

TensorFlow graphs offer opportunities for parallel execution.

Threading model

1. **`inter_op_parallelism_threads`** = max number of operators that can be executed in parallel
2. **`intra_op_parallelism_threads`** = max number of threads to use for executing an operator
3. **`OMP_NUM_THREADS`** = MKL-DNN equivalent of **`intra_op_parallelism_threads`**



performance GUIDE

`tf.ConfigProto` is used to set the **`inter_op_parallelism_threads`** and **`intra_op_parallelism_threads`** configurations of the **`Session`** object.

```
>>> config = tf.ConfigProto()
>>> config.intra_op_parallelism_threads = 56
>>> config.inter_op_parallelism_threads = 2
>>> tf.Session(config=config)
```

https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn

SYSTEM OPTIMIZATIONS: LOAD BALANCING

Incorrect setting of threading model parameters can lead to over- or under-subscription, leading to poor performance.

Solution:

- Set these parameters for your model manually.
- Guidelines on TensorFlow webpage

```
OMP: Error #34: System unable
to allocate necessary resources
for OMP thread:

OMP: System error #11: Resource
temporarily unavailable

OMP: Hint: Try decreasing the
value of OMP_NUM_THREADS.
```

PERFORMANCE GUIDE

Setting the threading model correctly

- We provide best settings for popular CNN models. (https://ai.intel.com/tensorflow-optimizations-intel-xeon-scalable-processor)

Example setting MKL variables with python **os.environ** :

```
os.environ["KMP_BLOCKTIME"] = "1"
os.environ["KMP_AFFINITY"] = "granularity=fine,compact,1,0"
os.environ["KMP_SETTINGS"] = "0"
os.environ["OMP_NUM_THREADS"] = "56"
```

Tuning MKL for the best performance

This section details the different configurations and environment variables that can be used to tune the MKL to get optimal performance. Before tweaking various environment variables make sure the model is using the `NCHW` (`channels_first`) data format. The MKL is optimized for `NCHW` and Intel is working to get near performance parity when using `NHWC`.

MKL uses the following environment variables to tune performance:

- KMP_BLOCKTIME - Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.
- KMP_AFFINITY - Enables the run-time library to bind threads to physical processing units.
- KMP_SETTINGS - Enables (true) or disables (false) the printing of OpenMP* run-time library environment variables during program execution.
- OMP_NUM_THREADS - Specifies the number of threads to use.

https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn

PERFORMANCE GUIDE

TensorFlow ™ | Install | Develop | API r1.8 | Deploy | Extend | Community | Versions | Ecosystem | Search | GITHUB

GET STARTED | PROGRAMMER'S GUIDE | TUTORIALS | PERFORMANCE | MOBILE | HUB

Optimizing for CPU

CPUs, which includes Intel® Xeon Phi™, achieve optimal performance when TensorFlow is built from source with all of the instructions supported by the target CPU.

Beyond using the latest instruction sets, Intel® has added support for the Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) to TensorFlow. While the name is not completely accurate, these optimizations are often simply referred to as 'MKL' or 'TensorFlow with MKL'. TensorFlow with Intel® MKL-DNN contains details on the MKL optimizations.

The two configurations listed below are used to optimize CPU performance by adjusting the thread pools.

- `intra_op_parallelism_threads` : Nodes that can use multiple threads to parallelize their execution will schedule the individual pieces into this pool.
- `inter_op_parallelism_threads` : All ready nodes are scheduled in this pool.

These configurations are set via the `tf.ConfigProto` and passed to `tf.Session` in the `config` attribute as shown in the snippet below. For both configuration options, if they are unset or set to 0, will default to the number of logical CPU cores. Testing has shown that the default is effective for systems ranging from one CPU with 4 cores to multiple CPUs with 70+ combined logical cores. A common alternative optimization is to set the number of threads in both pools equal to the number of physical cores rather than logical cores.

```
config = tf.ConfigProto()
config.intra_op_parallelism_threads = 44
config.inter_op_parallelism_threads = 44
tf.session(config=config)
```

The Comparing compiler optimizations section contains the results of tests that used different compiler optimizations.

TensorFlow with Intel® MKL DNN

Intel® has added optimizations to TensorFlow for Intel® Xeon® and Intel® Xeon Phi™ though the use of Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) optimized primitives. The optimizations also provide speedups for the consumer line of processors, e.g. i5 and i7 Intel processors. The Intel published paper TensorFlow* Optimizations on Modern Intel® Architecture contains additional details on the implementation.

https://www.tensorflow.org/performance/performance_guide#tensorflow_with_intel_mkl_dnn

INTEL-OPTIMIZED TENSORFLOW PERFORMANCE AT A GLANCE

TRAINING THROUGHPUT

14X

Intel-optimized TensorFlow ResNet50 training performance compared to
default TensorFlow for CPU

INFERENCE THROUGHPUT

3.2X

Intel-optimized TensorFlow InceptionV3 inference throughput compared to
Default TensorFlow for CPU

Inference and training throughput uses FP32 instructions

System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

TensorFlow Source: https://github.com/tensorflow/tensorflow
TensorFlow Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

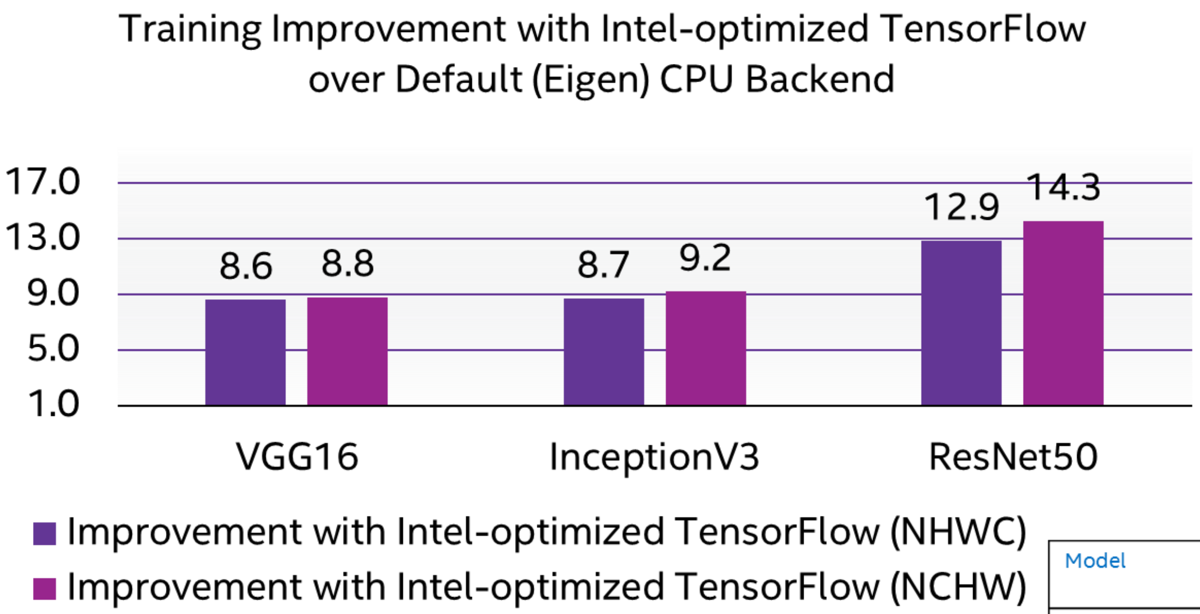
TensorFlow benchmarks: https://github.com/tensorflow/benchmarks

Unoptimized TensorFlow may not exploit the best performance from Intel CPUs.

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

THIS IS HPC ON INTEL

INTEL-OPTIMIZED TENSORFLOW TRAINING PERFORMANCE



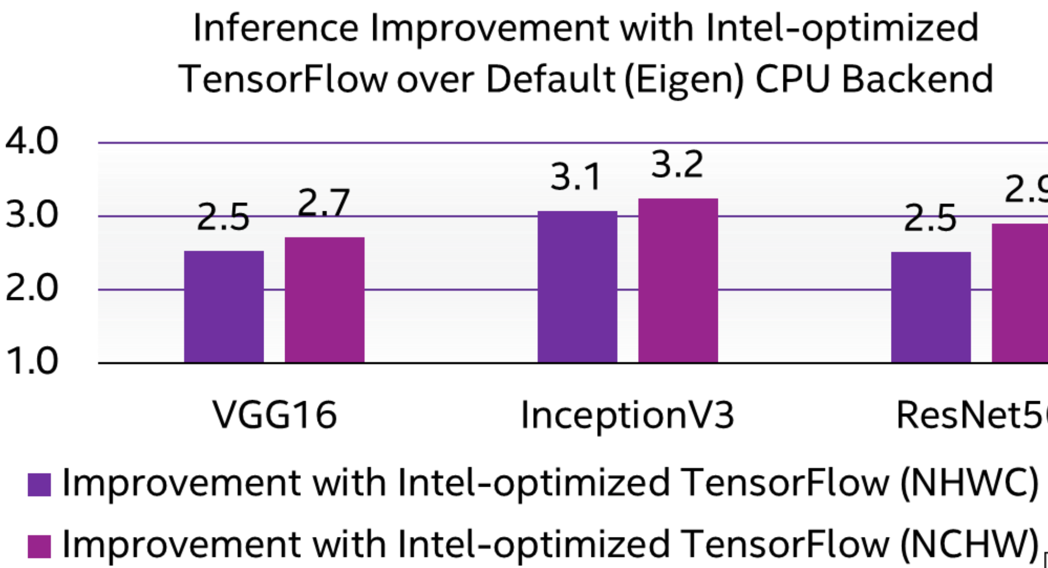
System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

TensorFlowSource: https://github.com/tensorflow/tensorflow
TensorFlow Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

TensorFlow benchmarks: https://github.com/tensorflow/benchmarks

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

INTEL-OPTIMIZED TENSORFLOW INFERENCE PERFORMANCE



System configuration:
CPU Thread(s) per core: 2 **Core(s) per socket**: 28 **Socket(s)**: 2 **NUMA node(s)**: 2 **CPU family**: 6 **Model**: 85 **Model name**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz Stepping: 4 **HyperThreading**: ON **Turbo**: ON **Memory** 376GB (12 x 32GB) 24 slots, 12 occupied 2666 MHz Disks Intel RS3WC080 x 3 (800GB, 1.6TB, 6TB) **BIOS** SE5C620.86B.00.01.0004.071220170215 **OS** Centos Linux 7.4.1708 (Core) Kernel 3.10.0-693.11.6.el7.x86_64

TensorFlowSource: https://github.com/tensorflow/tensorflow
TensorFlow Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

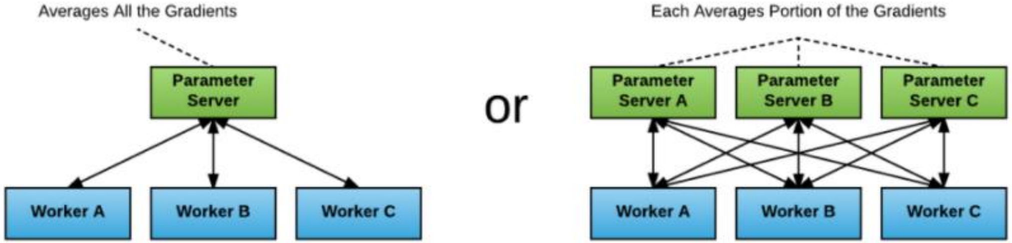
TensorFlow benchmarks: https://github.com/tensorflow/benchmarks

Model	Data_format	Intra_op	Inter_op	OMP_NUM_THREADS	KMP_BLOCKTIME
VGG16	NCHW	56	1	56	1
InceptionV3	NCHW	56	2	56	1
ResNet50	NCHW	56	2	56	1

Distributed TensorFlow™ Compare

Distributed Tensorflow with Parameter Server

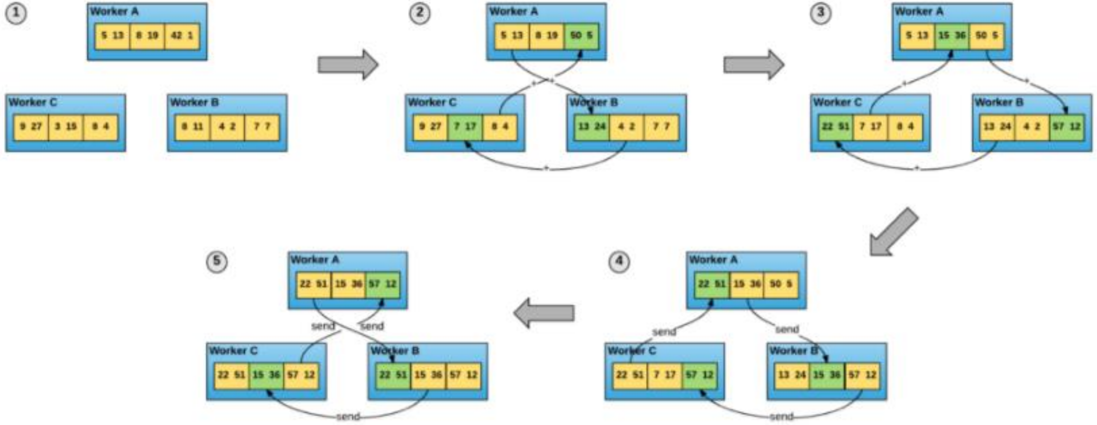
With Parameter Server



The parameter server model for distributed training jobs can be configured with different ratios of parameter servers to workers, each with different performance profiles.

Uber's open source Distributed training framework for TensorFlow

No Parameter Server



The ring all-reduce algorithm allows worker nodes to average gradients and disperse them to all nodes without the need for a parameter server.

Source: https://eng.uber.com/horovod/

DISTRIBUTED TRAINING : MULTI-NODE MULTI-SOCKET WITH HOROVOD MPI LIB

Run as Distributed Training Across Multiple Nodes & Multiple Sockets

- No Parameter Server required
- Each **socket** on each worker node running 2 or more Framework Streams
- Internode communication with horovod MPI library

HOROVOD for multinode:

from Parameter server (PS):

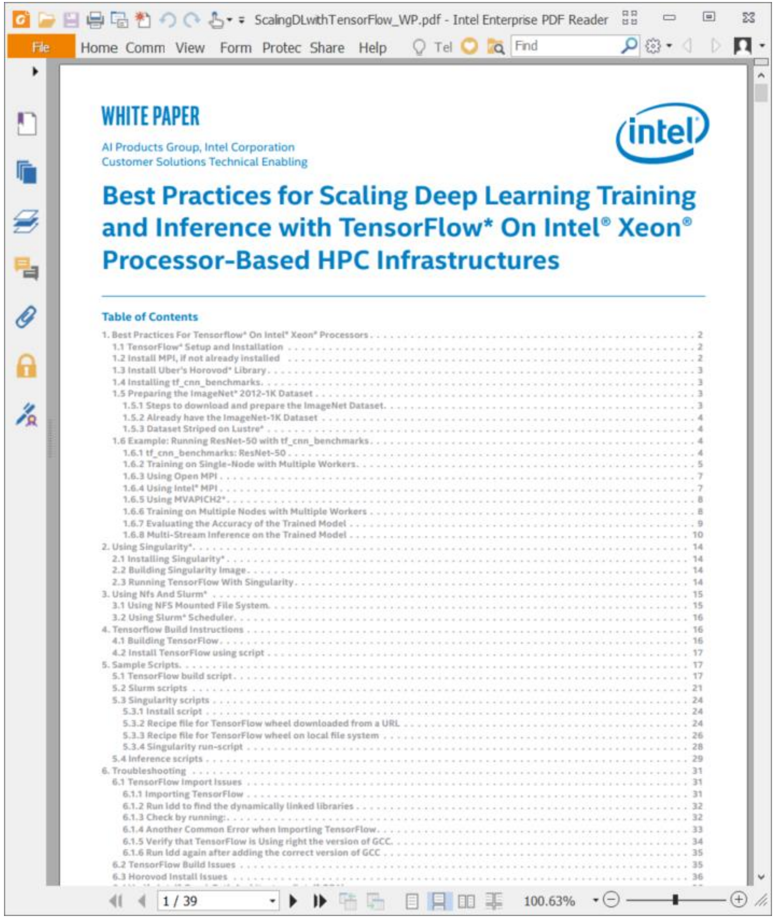
```
NP=4
PER_PROC=10
HOSTLIST=192.168.10.110
MODEL=inception3
BS=64
BATCHES=100
INTRA=10
INTER=2
```

```
/usr/lib64/openmpi/bin/mpirun --allow-run-as-root -np $NP -cpus-per-proc $PER_PROC –
map-by socket -H $HOSTLIST --report-bindings --oversubscribe -x LD_LIBRARY_PATH python
./tf_cnn_benchmarks.py --model $MODEL --batch_size $BS --data_format NCHW –
num_batches $BATCHES --distortions=True --mkl=True --local_parameter_device cpu –
num_warmup_batches 10 --optimizer rmsprop --display_every 10 --kmp_blocktime 1 –
variable_update horovod --horovod_device cpu --num_intra_threads $INTRA –
num_inter_threads $INTER  --data_dir /home/tf_imagenet --data_name imagenet
```

Scaling TensorFlow

There is way more to consider when striking for peak performance on distributed deep learning training.:

https://ai.intel.com/white-papers/best-known-methods-for-scaling-deep-learning-with-tensorflow-on-intel-xeon-processor-based-clusters/



intel®

intel®

experience
what's inside™