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.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, the Intel logo, Pentium, Xeon, Core, VTune, OpenVINO, Cilk, are trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries.

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.
What’s Inside Intel® Parallel Studio XE 2019
Comprehensive Software Development Tool Suite

**Composer Edition**

**Build**
- Compilers & Libraries
- Intel® Math Kernel Library
- Intel® Data Analytics Acceleration Library
- Intel Threading Building Blocks
- Intel® Integrated Performance Primitives
- Intel® Distribution for 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*

Intel® Architecture Platforms

*Available only in the Composer Edition.

Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
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

---

**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. 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."

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

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

The time to begin AI adoption is now
AI SOLUTIONS IN EVERY MARKET

AGRICULTURE
Achieve higher yields & increase efficiency

ENERGY
Maximize production and uptime

EDUCATION
Transform the learning experience

GOVERNMENT
Enhance safety, research, and more

FINANCE
Turn data into valuable intelligence

HEALTH
Revolutionize patient outcomes

INDUSTRIAL
Empower truly intelligent Industry 4.0

MEDIA
Create thrilling experiences

RETAIL
Transform stores and inventory

SMART HOME
Enable homes that see, hear, and respond

TELECOM
Drive network and operational efficiency

TRANSPORT
Automated driving

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

SIMPLIFY AI
using community solutions

DEPLOY AI ANYWHERE
with unprecedented HW choice

SCALE WITH CONFIDENCE
on the platform for IT & cloud

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

### Platforms
- Finance
- Healthcare
- Energy
- Industrial
- Transport
- Retail
- Home
- More...

### Deep Learning Deployment

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

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

### Deep Learning Frameworks

- TensorFlow*
- MXNet*
- Caffe*
- BigDL/Spark*

### Deep Learning Accelerators

- Intel®
- Xeon
- Atom
- Core
- ARM

**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)

### Deep Learning Graph Compiler

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

### Optimization Notice

Copyright © 2019, Intel Corporation. All rights reserved.

*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.
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

“Most businesses” claim is based on survey of Intel direct engagements and internal market segment analysis.
Deep Learning in Practice


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

Large cloud users employ CPU extensively for deep learning
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”

2018 Developer Skills Report

- Python, Java, R are top 3 languages in job postings for data science and machine learning jobs
The most popular ML/DL packages for Python

- python
- SciPy
- TensorFlow
- Theano
- scikit-learn
- dmlc
- mxnet
- NumPy
- pandas
- Keras
- matplotlib

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
Chapter 19: Performance Optimization of Black—Scholes Pricing

\[ V_{\text{call}} = S_0 \cdot \text{CDF}(d_1) - e^{-rT} \cdot X \cdot \text{CDF}(d_2) \]

\[ V_{\text{put}} = e^{-rT} \cdot X \cdot \text{CDF}(-d_2) - S_0 \cdot \text{CDF}(-d_1) \]

\[ d_1 = \frac{\ln(S_0/X) + (r + \sigma^2/2)T}{\sigma \sqrt{T}} \]

\[ d_2 = \frac{\ln(S_0/X) + (r - \sigma^2/2)T}{\sigma \sqrt{T}} \]
## What’s Inside Intel® Distribution for Python 2019

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

<table>
<thead>
<tr>
<th>FASTER PERFORMANCE</th>
<th>GREATER PRODUCTIVITY</th>
<th>ECOSYSTEM COMPATIBILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Libraries, Parallelism, Multithreading, Language Extensions</td>
<td>Prebuilt &amp; Accelerated Packages</td>
<td>Supports Python 2.7 &amp; 3.x, conda, pip</td>
</tr>
<tr>
<td><strong>Accelerated NumPy/SciPy/scikit-learn</strong> with Intel® MKL(^1) &amp; Intel® DAAL(^2)</td>
<td>Prebuilt &amp; optimized packages for numerical computing, machine/deep learning, HPC, &amp; data analytics</td>
<td><strong>Compatible &amp; powered by Anaconda(^*), supports conda &amp; pip</strong></td>
</tr>
<tr>
<td>Data analytics, machine learning &amp; deep learning with scikit-learn, pyDAAL</td>
<td><strong>Drop in replacement for existing Python</strong> - No code changes required</td>
<td>Distribution &amp; individual optimized packages also available via conda, pip YUM/APT, Docker image on DockerHub</td>
</tr>
<tr>
<td>Scale with Numba(^<em>) &amp; Cython(^</em>)</td>
<td>Jupyter(^*) notebooks, Matplotlib included</td>
<td>Optimizations upstreamed to main Python trunk</td>
</tr>
<tr>
<td>Includes optimized mpi4py, works with Dask(^<em>) &amp; PySpark(^</em>)</td>
<td>Conda build recipes included in packages</td>
<td>Commercial support through Intel® Parallel Studio XE</td>
</tr>
<tr>
<td>Optimized for latest Intel® architecture</td>
<td>Free download &amp; free for all uses including commercial deployment</td>
<td></td>
</tr>
</tbody>
</table>

**ECOSYSTEM COMPATIBILITY**

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

---

\(^1\) Intel® Math Kernel Library

\(^2\) Intel® Data Analytics Acceleration Library

---

1 Available only in Intel® Parallel Studio Composer Edition.

---

*Other names and brands may be claimed as the property of others.*
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
- YUM/APT repositories
- Standalone Python Distribution
- Intel optimized packages via conda
- Docker Hub

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
### 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

<table>
<thead>
<tr>
<th>Pre-processing</th>
<th>Transformation</th>
<th>Analysis</th>
<th>Modeling</th>
<th>Validation</th>
<th>Decision Making</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decompression, Filtering, Normalization</td>
<td>Aggregation, Dimension Reduction</td>
<td>Summary Statistics Clustering, etc.</td>
<td>Machine Learning (Training) Parameter Estimation Simulation</td>
<td>Hypothesis Testing Model Errors</td>
<td>Forecasting Decision Trees, etc.</td>
</tr>
</tbody>
</table>
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
  - Association Rule Mining (Apriori)
  - Optimization Solvers (SGD, AdaGrad, lBFGS)

- Outlier Detection
  - Univariate
  - Multivariate
  - Math Functions (exp, log,...)

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
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
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 ~]$ 

........Some IP address
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:
(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:
But...
m...achines can help!

Automating the process:

• helps gain of time for the radiologist
• gives time back to the patient and surgeon
• improves segmentation quality
• Nearly 153 exabytes of healthcare-related data were generated in 2013
• amount to increase by 48% annually
• expected to reach 2,314 exabytes in 2020

One exabyte is one billion gigabytes or 250 000 000 DVDs worth of information.
The dataset for the deep learning algorithm

- Brain Tumor Segmentation (BraTS) Challenge 2018 dataset
- **Goal:** classify every voxel in the image as either:
  i. healthy tissue
  ii. necrosis or non-enhancing (red)
  iii. edema (green) or
  iv. enhancing tumor (yellow)
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:
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

K Keras
TensorFlow
OpenVINO
How did we boost the performance of the algorithm?

Thanks to Intel® Technologies:

- Intel® Distribution for Python
- Keras
- TensorFlow
- OpenVINO®

Frameworks and Software Stack:

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

The base hardware

Python-based Deep Learning Demo application
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

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMP_BLOCKTIME</td>
<td>Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping.</td>
</tr>
<tr>
<td>KMP_AFFINITY</td>
<td>Enables run-time library to bind threads to physical processing units.</td>
</tr>
<tr>
<td>OMP_NUM_THREADS</td>
<td>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.</td>
</tr>
</tbody>
</table>

[https://www.tensorflow.org/guide/performance/overview](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

1. Trained DL model
2. Model Optimizer
3. IR .xml .bin
4. Your Program

Caffe
mxnet
ONNX
TensorFlow

FP32/FP16

Inference Engine

CNNNetwork

MKLDNN Plugin
clDNN Plugin
FPGA Plugin
GNA Plugin
MyriadX Plugin

CPU
GPU
Arria
GNA
Movidius
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

Example: inference flow

Original code
Convolution → ReLU → Batch Norm

Naïve integration
Reorder → Convolution → Reorder → ReLU → Batch Norm

Layout propagation
Reorder → Convolution → ReLU → Batch Norm → Reorder

Layer fusion
Reorder → Conv+ReLU → Reorder

Transform weights to integrate BN (offline)

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.
Graph optimizations: fusion

Before Merge

After Merge
Graph optimizations: layout Conversion

- All MKL-DNN operators use highly-optimized layouts for TensorFlow tensors.
INTEL® TENSORFLOW OPTIMIZATIONS
INTEL-TENSORFLOW OPTIMIZATIONS

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

<table>
<thead>
<tr>
<th>Forward</th>
<th>Backward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv2D</td>
<td>Conv2DGrad</td>
</tr>
<tr>
<td>Relu, TanH, ELU</td>
<td>ReLUGrad, TanHGrad, ELUGrad</td>
</tr>
<tr>
<td>MaxPooling</td>
<td>MaxPoolingGrad</td>
</tr>
<tr>
<td>AvgPooling</td>
<td>AvgPoolingGrad</td>
</tr>
<tr>
<td>BatchNorm</td>
<td>BatchNormGrad</td>
</tr>
<tr>
<td>LRN</td>
<td>LRNGrad</td>
</tr>
<tr>
<td>MatMul, Concat</td>
<td></td>
</tr>
</tbody>
</table>
OPERATOR OPTIMIZATIONS IN RESNET50

Intel-optimized TensorFlow timeline

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.

\[
\begin{array}{cccc}
21 & 18 & 32 & 6 \\
1 & 8 & 92 & 37 \\
40 & 11 & 9 & 22 \\
23 & 3 & 47 & 29 \\
5 & 15 & 16 & 22 \\
\end{array}
\]

\[
\begin{array}{cccc}
21 & 18 & \ldots & 1 \\
1 & \ldots & 8 & 92 \\
\end{array}
\]

Better optimized for some operations vs.

\[
\begin{array}{cccc}
21 & 8 & 18 & 92 \\
32 & 37 & 6 & \ldots \\
\end{array}
\]

\{N:2, R:5, C:5\}
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
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`
tf.ConfigProto is used to set the inter_op_parallelism_threads and intra_op_parallelism_threads configurations of the Session object.

```python
>>> 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.
Setting the threading model correctly


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

```python
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.

Optimizing for CPU

CPU's, which include Intel® Xeon® and 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.

```python
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™ through 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. G and I7 Intel processors. The Intel published paper TensorFlow* Optimizations on Modern Intel® Architectures contains additional details on the implementation.

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

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

INFEERENCE THROUGHPUT

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

Unoptimized TensorFlow may not exploit the best performance from Intel CPUs.
INTEL-OPTIMIZED TENSORFLOW TRAINING PERFORMANCE

Training Improvement with Intel-optimized TensorFlow over Default (Eigen) CPU Backend

<table>
<thead>
<tr>
<th>Model</th>
<th>Data_format</th>
<th>Intra_op</th>
<th>Inter_op</th>
<th>OMP_NUM_THREADS</th>
<th>KMP_BLCKTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>NCHW</td>
<td>56</td>
<td>1</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>NCHW</td>
<td>56</td>
<td>2</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>ResNet50</td>
<td>NCHW</td>
<td>56</td>
<td>2</td>
<td>56</td>
<td>1</td>
</tr>
</tbody>
</table>

System configuration:
- CPU Thread(s) per core: 2
- 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
INTEL-OPTIMIZED TENSORFLOW INFERENCE PERFORMANCE

Inference Improvement with Intel-optimized TensorFlow over Default (Eigen) CPU Backend

<table>
<thead>
<tr>
<th>Model</th>
<th>Data_format</th>
<th>Intra_op</th>
<th>Inter_op</th>
<th>OMP_NUM_THREADS</th>
<th>KMP_CLOCKTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16</td>
<td>NCHW</td>
<td>56</td>
<td>1</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>NCHW</td>
<td>56</td>
<td>2</td>
<td>56</td>
<td>1</td>
</tr>
<tr>
<td>ResNet50</td>
<td>NCHW</td>
<td>56</td>
<td>2</td>
<td>56</td>
<td>1</td>
</tr>
</tbody>
</table>

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 Commit ID: 926fc13f7378d14fa7980963c4fe774e5922e336.

TensorFlow benchmarks: https://github.com/tensorflow/benchmarks
Distributed TensorFlow™ Compare

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

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/
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.:
