Optimizing Machine Learning with Tensorflow, ActivePython and Intel

> Ashraf Bhuiyan, Intel Corporation
> Pete Garcin, ActiveState

> October 26, 2017 @ ActiveState/Intel® Webinar

ActiveState

- Pete Garcin
- Developer Advocate at ActiveState
- 15+ years in software in various roles
- Twitter/GitHub: rawktron

ActiveState®

THE OPEN SOURCE LANGUAGES COMPANY

ActiveState®

THE OPEN SOURCE LANGUAGES COMPANY

ActivePerl®
ActiveGo™
ActiveRuby™

ActivePython®
ActiveTcl®
Komodo® IDE

Machine Learning

- Transforming almost every business
- Exploding ecosystem of tools, making it more accessible to even non-experts
- TensorFlow, by Google has become the most popular package in this ecosystem

TensorFlow

- Google's library for ML
- Expresses calculations as a computation graph
- Many language bindings
- Supports/provides pre-trained models
- 72K stars on GitHub!

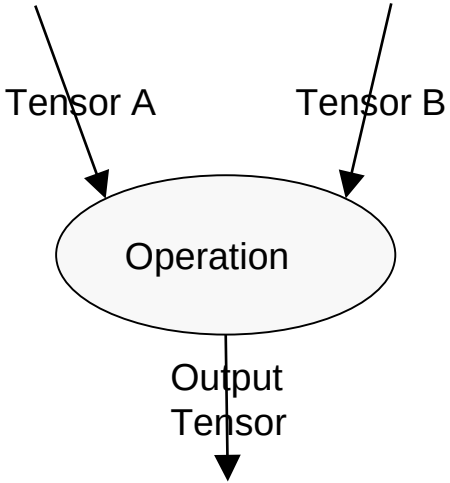
Tensorflow

- Official bindings for Python, C, Java, Go
- Library is written in C++
- Used as a ‘back end’ in wrapper libraries

Keras

TensorFlow

- Computation Graph is a graph where the nodes are operators (add, sub, multiply, etc.)
- Edges are tensors
- Tensors are effectively N-dimensional arrays



Tensors

- N-dimensional arrays
- Types of operations:
  - Matrix operations
  - Linear algebra
  - Vector math

ActiveState

Optimization Cases

- Training neural networks
- Large data sets
- Complex deep learning networks
- Real-time Inference



Optimizing TensorFlow

- Data storage
  - Allocations, Conversions, Layout, etc.
- Parallelization
  - Taking advantage of cores, etc.
- Instruction optimization
  - MKL style operation optimization



Intel Optimizations

- Intel provides optimizations to take maximum advantage of their hardware

- For example, Intel MKL (Math Kernel Library) provides impressive results on fundamental math operations

Intel Optimizations

- ActivePython includes MKL, and work to include additional optimizations as they become available

- TensorFlow specific optimizations offer dramatic speed increases for commercial applications

ActiveState

Simple MKL Performance Example

```
for nSize in range(0, 10):
    a = np.random.rand(nSize,nSize)
    result = np.linalg.eig(a);
```

A simple test that computes the eigenvalues and normalized eigenvectors of a random square matrix of increasing size.

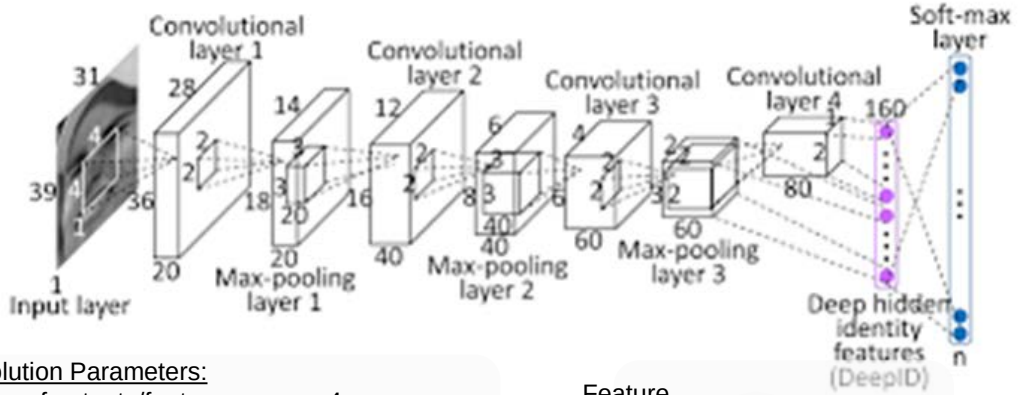


Linear Algebra Test - NumPy w/ Intel® MKL

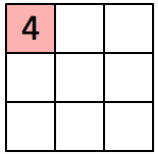
Optimizing TensorFlow

- *Mohammad Ashraf Bhuiyan - Intel Artificial Intelligence Group, Senior Software Engineer*
  - 10+ years in software in various roles
  - GitHub: mbhuiya

Deep Learning: Example



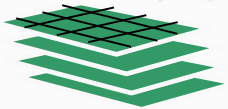
Image

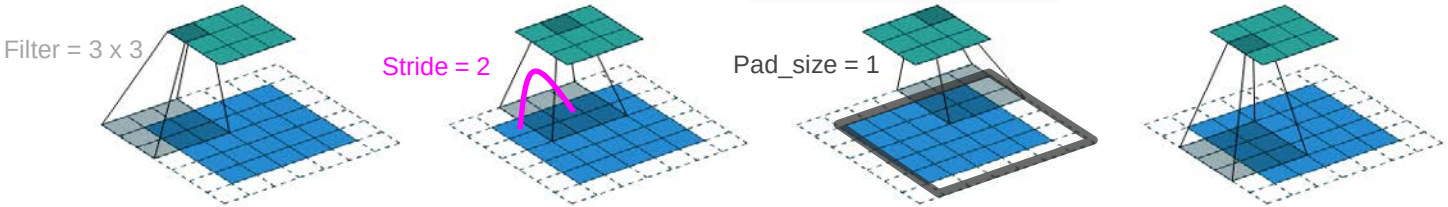


Convolved Feature

Convolution Parameters:
Number of outputs/feature-maps: < 4 >
Filter size: < 3 x 3 >
Stride: < 2 >
Pad_size (for corner case): <1>

Feature maps



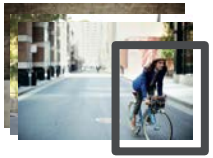


Deep Learning: Train Once Use Many Times

Step 1: Training

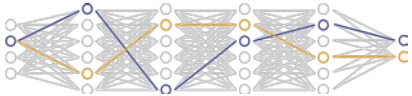
(Over Hours/Days/Weeks)

Input data



Person

Create deep network



Output classification

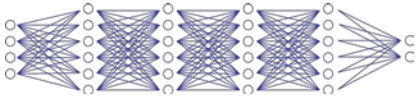
90% person
8% traffic light

Trained Model

Step 2: Inference

(Real Time)

New input from camera and sensors

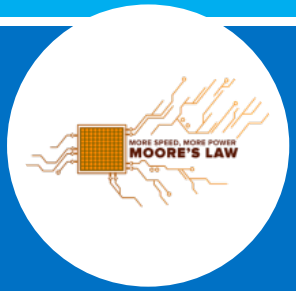


Trained neural network model

97% person

Output classification

Deep Learning: Why Now?

Bigger Data	Better Hardware	Smarter Algorithms
		
Image: 1000 KB / picture Audio: 5000 KB / song Video: 5,000,000 KB / movie	Transistor density doubles every 18 months Cost / GB in 1995: $1000.00 Cost / GB in 2015: $0.03	Advances in algorithm innovation, including neural networks, leading to better accuracy in training models

TensorFlow

- **2nd generation open source machine learning framework from Google***
- **Widely used across Google in many key apps – search, Gmail, photos, translate, etc.**
- **General computing mathematical framework used on:**
  - Deep neural network
  - Other machine learning algorithm

- **Core system provides set of key computational extendable kernel**
- **Core in C++, front end wrapper is in python**
- **Multi-node support using proprietary GRPC, VERBS, MPI protocols**

Tensorflow Optimizations at Intel

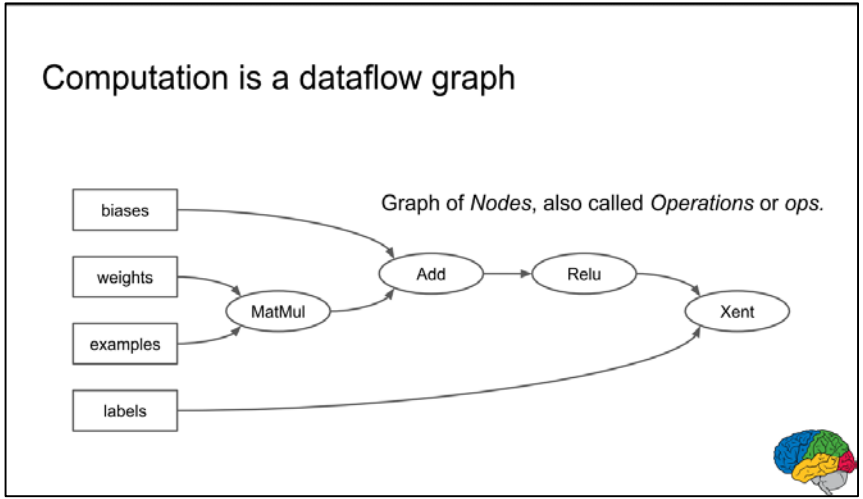
1. Operator-level optimizations in TensorFlow* for Intel® Architectures
   - Intel® MKL integration

2. Graph-level optimizations in TensorFlow* for Intel® Architectures
   - Data layout conversion optimization
   - Node merging optimization
   - Memory allocation
   - Load balancing

Operator-level optimization

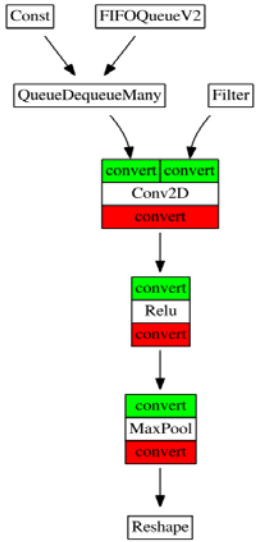
- **Intel® MKL has optimized common set of primitives**
- **Call Intel® MKL API for executing Tensorflow operation**
- **Require Data layout conversion:**

> TF code
> TF layout to MKL layout
> Call MKL API
> MKL layout to TF layout
> TF code



Operator-level optimizations: Example

```
class MklConv2DOp : public OpKernel {
  void Compute (OpKernelContext* context) override {
   const Tensor& tf_input = context->input(0);
   const Tensor& tf_filter = context->input(1);
   Tensor* output = context->allocate_output(..);
   mkl_input = convert_to_mkldnnlayout(tf_input);
   mkl_filter = convert_to_mkldnnlayout(tf_filter);
   mkl_output = mkldnn_conv2d_fwd(mkl_input, mkl_filter,…);
   *output = convert_to_tflayout(mkl_output);
  }
 };
```



Graph optimizations address the overhead of data layout conversion

Tensorflow* Operations optimized for Intel® Architectures

Forward	Backward
• Conv2D • Relu • MaxPooling • AvgPooling • LRN • FusedBatchNorm • MatMul • MklToTF (convert)	• Conv2DGrad • ReluGrad • MaxPoolingGrad • AvgPolingGrad • LRNGrad • FusedBatchNormGrad • TransposeCpu • Reshape

Graph optimizations

Graph optimizations in TensorFlow* for Intel® Architectures

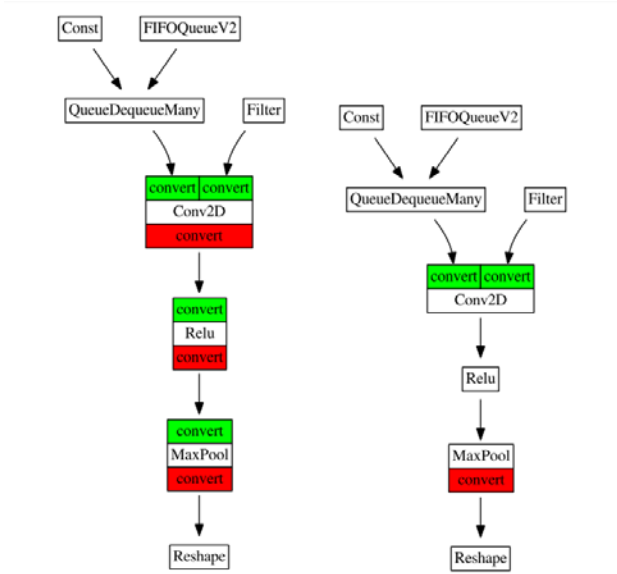
- **Graph has complete view of the operations and their context.**
- **Enable cross-operation optimizations**

- **Graph optimizations**
  1. Data layout conversion optimizations
  2. Node merging (also called Fusion)
  3. Memory allocation
  4. Load balancing

Data Layout Conversion Optimization

Data layout conversion optimization - Example

- **Layout conversions are expensive data shuffling operations.**
- **The challenge is how to avoid unnecessary conversions**
- **Optimizations:**
  - Find out sub-graphs that contain all operators supported by Intel® MKL.
  - Then introduce layout conversions on the boundary of the subgraphs.



(a) Graph with input and output conversions

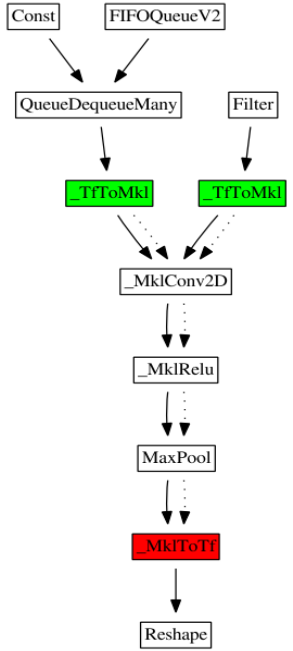
(b) Graph with optimized conversions

Layout conversion optimization

- **Based on Google's suggestions, our current implementation emits Intel® MKL layout as an extra output tensor.**

- **Example: if X = Conv2D(A, B) was earlier operator, then X_mkl = _MklConv2D(A, B, A_m, B_m) is a new operator.**

✓ A_m, B_M are MKL layout of A and B



Need Graph Rewrite Pass : Rewrite TF op to MKL op

- **Example:**
  - Conv2D takes 2 inputs and produces 1 output.
  - We want Conv2D to accept 4 inputs and produce 2 output.
  - That is why we need new Conv2D operator (_MklConv2D).
- A graph pass rewrite ***TF operators*** into ***MKL operators***.

- **File:** core/graph/mkl_layout_pass.cc

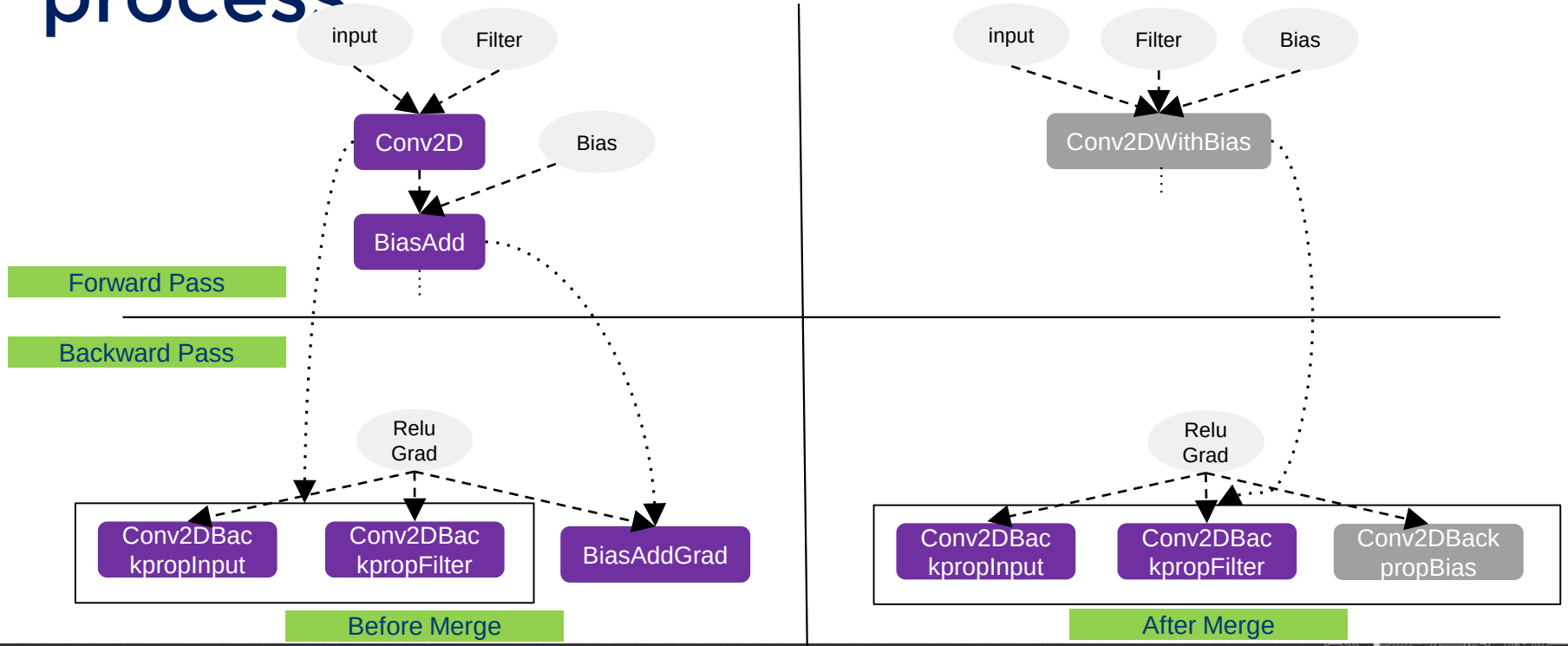
Data: G: Input operation graph
Result: G'': output operation graph with optimized layout conversions
$G_t \leftarrow topological\_sort(G)$;
$G' \leftarrow []$
/* Loop below implements first task of graph rewrite pass. */
for $\forall$ *operation* O *in* G_t **do**
  if $is\_mkldnn\_op(O)$ **then**
    $O'_{inputs} \leftarrow [] ;$
    for *every input* I *of* O **do**
      $O'_{inputs} \leftarrow O'_{inputs} \cup I ;$
      /* I_{mkl} is extra input that carries MKL-DNN layout. */
      $O'_{inputs} \leftarrow O'_{inputs} \cup I\_mkl$
    end
    $O' \leftarrow O'_{inputs} ;$
    $G' \leftarrow G' \cup O' ;$
    delete O;
  else
    $G' \leftarrow G' \cup O ;$
  end
end

Node fusion optimization

Fusion optimization

- **Identify common pattern of operators that arise in most deep learning models**
- **Merge matching subgraph for the pattern to produce smaller graph nodes**
- **Currently, we merge Conv2D+Bias to new node _MklConv2DWithBias.**
- **Implementation**
  - Perform in the same graph rewrite pass that rewrites nodes for data layout conversion optimization

Conv2D and BiasAdd: Merge process



Memory Allocation


Optimization: Memory Allocation

- **Most NN operators allocate huge chunk of memory (Conv2D ~ hundred of MBs)**
- **Default CPU allocator in TensorFlow -> frequent allocs/deallocs of huge chunk of memory -> frequent mmap/unmap -> unnecessary page clears**

- **We developed Custom Pool Allocator using existing Pool allocator.**
  - Allocator holds on to released memory rather than releasing to OS directly.
  - Code: tensorflow/core/common_runtime/mkl_cpu_allocator.h

Load Balancing

Thread Pool and Parallelism

- **Tensorflow is a data-flow graph.**
- **It offers excellent opportunity for exploiting parallelism**

✓ Between operators.

✓ Within operators.

- **Thread pool parameters:**
  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. **MKL Threads =** operators controlled using OMP_NUM_THREADS. OMP_NUM_THREADS is conceptually same as intra_op_parallelism_threads.

Current Threading Issues & Solution

> Problem:

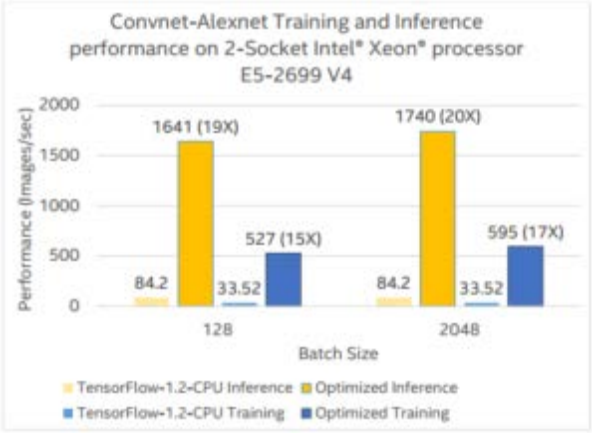
- Incorrect setting of `inter_op_threads` and `intra_op_threads` can lead to over- or under-subscription, leading to poor performance.

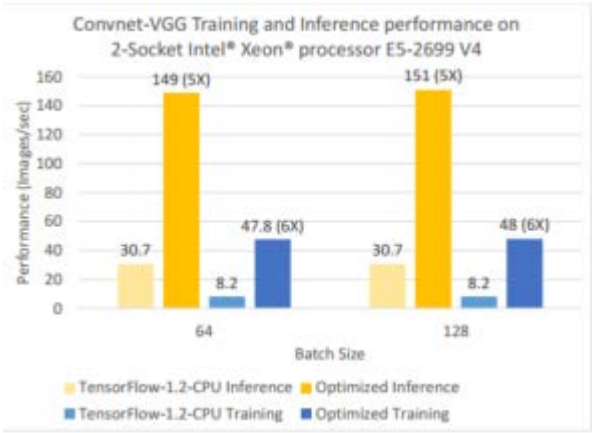
> Solution:

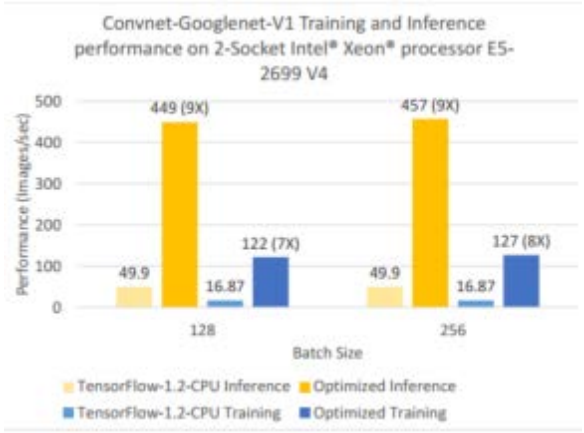
- Settings for `inter_op`, `intra_op` and `OMP_NUM_THREADS` were explored to get the best performance . Typically:
  - Intra_op = OMP_NUM_THREADS = # of physical cores in CPU
  - inter_op = # of sockets in a system
  - Google performance guide: https://www.tensorflow.org/performance/performance_guide
- No changes to Tensorflow code; changes to the run command.

Performance Improvement

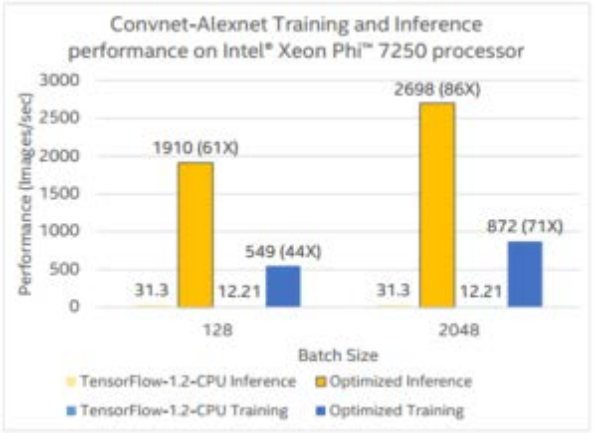
Optimized Tensorflow Performance on Intel® Xeon® processor

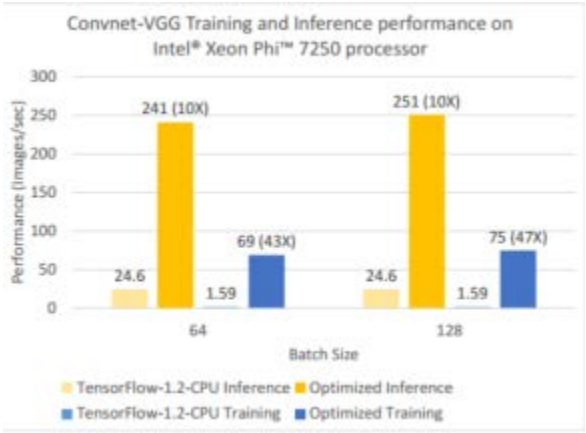


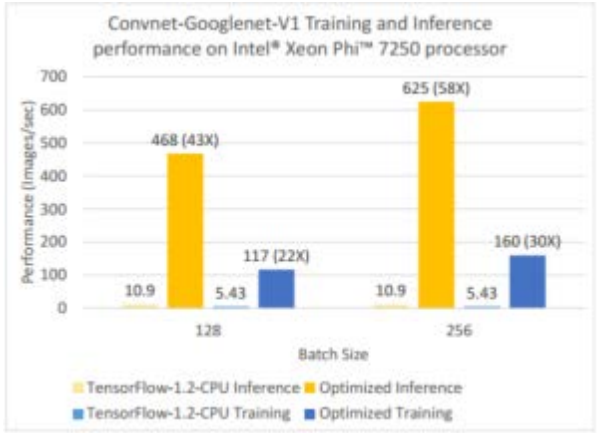




Optimized Tensorflow Performance on Intel® Xeon Phi® processor







How Do I Get Order of Magnitude CPU Speedup?

- Optimized TensorFlow on Intel architectures available from the public git.
  - git clone https://github.com/tensorflow/tensorflow.git
- Configure for best performance on CPU:
  - Run “./configure” from the TensorFlow source directory
- Building for best performance on CPU
  - Use following command to create a pip package that can be used to install the optimized TensorFlow wheel
  - bazel build --config=mkl --s --c opt //tensorflow/tools/pip_package:build_pip_package
  - Automatically downloads latest MKL-ML
- Install the optimized TensorFlow wheel
  - bazel-bin/tensorflow/tools/pip_package/build_pip_package ~/path_to_save_wheel
  - pip install --upgrade --user ~/path_to_save_wheel/wheel_name.whl

Summary

- **TensorFlow* is widely used DL and AI framework**
  - It has been slow on CPU until recently
- **Unique performance challenges addressed: MKL, data layout, inter/intra layer parallelization, etc.**
- **Significant performance gains from Intel optimization on Intel® Xeon and Xeon Phi processors**

- **Call to action:**
  - Use the right configuration for Tensorflow building
  - Find the best set of parameter for running models with Tensorflow
  - Get the orders of magnitude higher performance

Legal Disclaimers

- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families: Go to: Learn About Intel® Processor Numbers http://www.intel.com/products/processor_number
- Some results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.
- 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.
- Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.
- Relative performance is calculated by assigning a baseline value of 1.0 to one benchmark result, and then dividing the actual benchmark result for the baseline platform into each of the specific benchmark results of each of the other platforms, and assigning them a relative performance number that correlates with the performance improvements reported.
- SPEC, SPECint, SPECfp, SPECrate, SPECpower, SPECjbb, SPECompG, SPEC MPI, and SPECjEnterprise* are trademarks of the Standard Performance Evaluation Corporation. See http://www.spec.org for more information.
- TPC Benchmark, TPC-C, TPC-H, and TPC-E are trademarks of the Transaction Processing Council. See http://www.tpc.org for more information.
- No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Xeon® processor E7-8800/4800/2800 v2 product families or Intel® Itanium® 9500 series-based system (or follow-on generations of either.) Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient System Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel processor and enabled technology(ies). Built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.
  For systems also featuring Resilient Memory Technologies: No computer system can provide absolute reliability, availability or serviceability. Requires an Intel® Run Sure Technology-enabled system, including an enabled Intel® processor and enabled technology(ies). built-in reliability features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804