INTEL PERFORMANCE LIBRARIES FOR MACHINE LEARNING AND DEEP LEARNING

Feb, 2017

Gennady.Fedorov@intel.com

Legal Disclaimer & 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

Machine Learning: Your Path to Deeper Insight

Driving increasing innovation and competitive advantage across industries

intel strategy provides the foundation for success using AI

Solutions
for reference across industries

Tools/Platforms
to accelerate deployment

Intel® Deep Learning SDK for Training & Deployment

saffron TECHNOLOGY

nervana

Optimized Frameworks
to simplify development

Spark

Caffe

theano

torch

TensorFlow

neon

Libraries/Languages
featuring optimized building blocks

- Intel® Math Kernel Library (Intel® MKL & MKL-DNN)
- Intel® Data Analytics Acceleration Library (Intel® DAAL)
- Intel® Integrated Performance Primitives (Intel® IPP)
- Intel® Distribution for Python*

Hardware Technology
portfolio that is broad and cross-compatible

XEON PHI inside · XEON inside · Arria 10 · CORE i3 inside · CORE i5 inside · CORE i7 inside · ATOM inside

Datacenter ⟷ *Endpoint*

+Network
+Memory
+Storage

Optimization Notice

Agenda

- Intel® Math Kernel Library (Intel® MKL)
    - Overview, what's new in Intel MKL v.2017
    - Intel® MKL for machine learning
    - Neural network primitives in Intel MKL 2017 for deep learning
    - Introduction to open source Intel MKL-DNN
- Intel® Data Analytics Acceleration Library (Intel® DAAL)
- Intel® DAAL – Case Studies

Intel MKL v.2017

Linear Algebra	Fast Fourier Transforms	Vector Math	Summary Statistics	And More…	Deep Neural Networks
• BLAS • LAPACK • ScaLAPACK • Sparse BLAS • Sparse Solvers(SMP & for Clusters) • Iterative	• Multidimensional • FFTW interfaces • Cluster FFT	• Trigonometric • Hyperbolic • Exponential • Log • Power • Root • Vector RNGs	• Kurtosis • Variation coefficient • Order statistics • Min/max • Variance-covariance	• Splines • Interpolation • Trust Region • Fast Poisson Solver	• Convolution • Pooling • Normalization • ReLU • Softmax

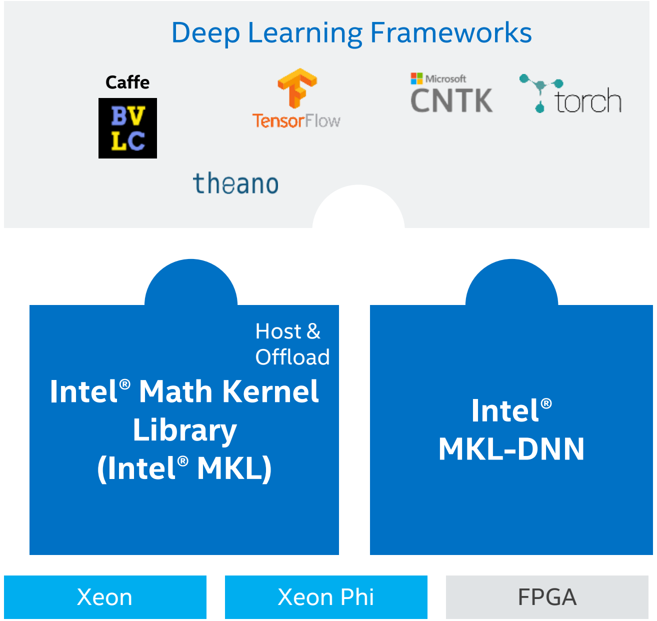
Intel MKL v.2017 - what's New

- Introduced optimizations for the Intel® Xeon Phi™ processor x200
- Introduced Deep Neural Networks (DNN) primitives
- Introduced new packed matrix multiplication interfaces
- Added fully distributed reordering step to Parallel Direct Sparse Solver for Clusters
- Included the latest LAPACK v3.6 enhancements
- Improved LU factorization, solve, and inverse (?GETR?) performance for very small sizes (<16).
- SparseQR & *Largest or smallest EV problem – Prototype Packages*
- *.... Many others*

Intel® MKL for machine learning

- **BLAS and Lapack**
  - Packed gemm, batch gemm
  - Small matrix multiply enhancements
  - gemmt for calculating U/L triangular part of C = alpha*A*B + beta*C
  - QR, SVD, Cholesky
  - Equation solvers
- **Vector Statistics**
  - Random Number Generation, Summary Statistics, Convolution
- **Vector Mathematical Functions**
- **Sparse matrix operations**
  - Sparse BLAS inspector-executor API
- **Deep Neural Network primitives**

Intel® MKL – Intel®MKL-DNN



Deep Learning frameworks

Intel MKL-DNN is an open source IA optimized DNN APIs, combined with Intel® MKL and build tools designed for scalable, high-velocity integration with ML/DL frameworks.

- Includes open source implementations of new DNN functionality
- Delivers new algorithms ahead of MKL releases
- Open for community contributions

Intel MKL is commercial SW Performance Library to extract max Intel HW performance and provide a common interface to all Intel processors and accelerators.

Intel libraries as path to bring optimized ML/DL frameworks to Intel hardware

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

Intel MKL 2017- Deep Neural Network (DNN) Primitives

	Functionality
Convolution	Direct (3-D); Fully connected
Pooling	Max; Min; Average
Activation	ReLU + variants
Normalization	LRN; Batched
Auxiliary	Conversion; Sum; Scale; Concat; Split
Topologies	AlexNet; ResNet; GoogleNet v1 & v2; VGG

Intel MKL **v.2017:** Deep Learning Framework Optimization

	MKL	**MKL-DNN**
Functionality	BLAS, LAPACK, FFT, Sparse Solvers, Vector Math & Statistics, **DNN**, & more …	**DNN**
API	C	C, C++
Product type	Proprietary commercial grade software Binary/Free/paid support	Open source + (GEMM binary*)
License	EULA	Apache 2.0
Distribution	Intel Registration Center	Github
Source code	Not available	Available (less GEMM*)
Maturity	Production	Tech preview
Release schedule	quarterly updates	incremental weekly updates

* GEMM matrix multiply building blocks are binary

Deep learning with Intel® MKL-DNN

Intel® MKL-DNN Programming Model

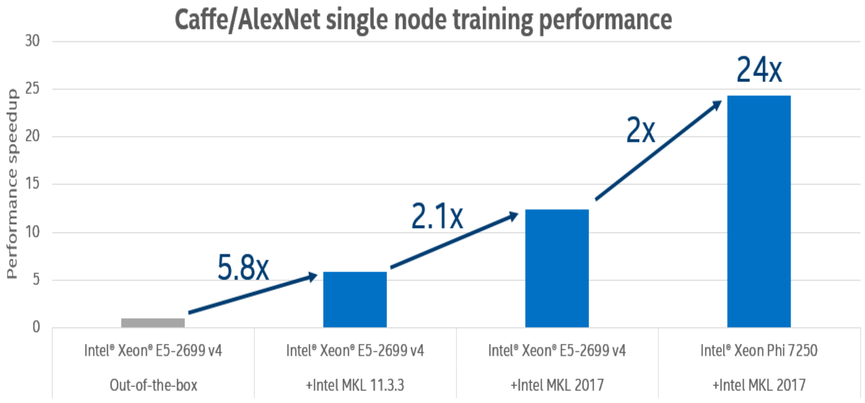
- **Primitive –** any operation (convolution, data format re-order, memory)
  - **Operation/memory descriptor** - convolution parameters, memory dimensions
  - **Descriptor** - complete description of a primitive
  - **Primitive –** a specific instance of a primitive relying on descriptor
- **Engine –** execution device (e.g., CPU)
- **Stream –** execution context

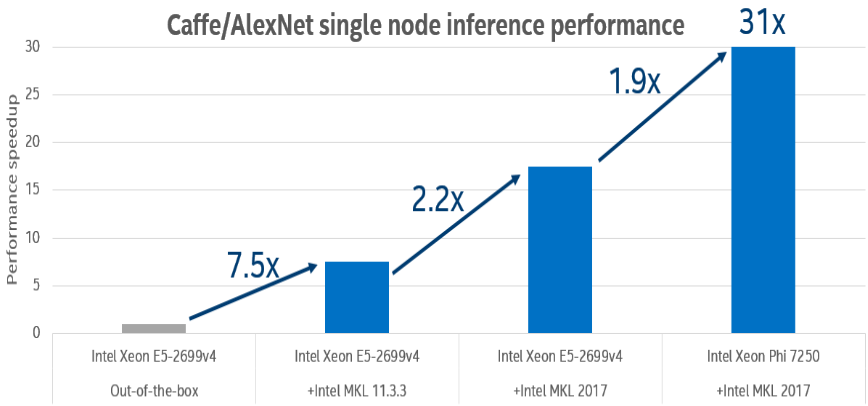
```
/* Initialize CPU engine */
auto cpu_engine = mkldnn::engine(mkldnn::engine::cpu, 0);
/* Create a vector of primitives */
std::vector<mkldnn::primitive> net;
/* Allocate input data and create a tensor structure that describes it */
std::vector<float> src(2 * 3 * 227 * 227);
mkldnn::tensor::dims conv_src_dims = {2, 3, 227, 227};
/* Create memory descriptors, one for data and another for convolution input */
auto user_src_md = mkldnn::memory::desc({conv_src_dims},
mkldnn::memory::precision::f32, mkldnn::memory::format::nchw);
auto conv_src_md = mkldnn::memory::desc({conv_src_dims},
mkldnn::memory::precision::f32, mkldnn::memory::format::any);
/* Create convolution descriptor */
auto conv_desc = mkldnn::convolution::desc(
mkldnn::prop_kind::forward, mkldnn::convolution::direct,
conv_src_md, conv_weights_md, conv_bias_md, conv_dst_md,
{1, 1}, {0, 0}, mkldnn::padding_kind::zero);

/* Create a convolution primitive descriptor */
auto conv_pd = mkldnn::convolution::primitive_desc(conv_desc, cpu_engine);

/* Create a memory descriptor and primitive */
auto user_src_memory_descriptor
= mkldnn::memory::primitive_desc(user_src_md, engine);
auto user_src_memory = mkldnn::memory(user_src_memory_descriptor, src);

/* Create a convolution primitive and add it to the net */
auto conv = mkldnn::convolution(conv_pd, conv_input, conv_weights_memory,
conv_user_bias_memory, conv_dst_memory);
net.push_back(conv);

/* Create a stream, submit all primitives and wait for completion */

mkldnn::stream().submit(net).wait();
```

Intel MKL v.2017- DNN Primitives, Performance

Brings IA-optimized performance to popular image recognition topologies:

- AlexNet, Visual Geometry Group (VGG), GoogleNet, and ResNet





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 information go to http://www.intel.com/performance . *Other names and brands may be property of others

Configurations:

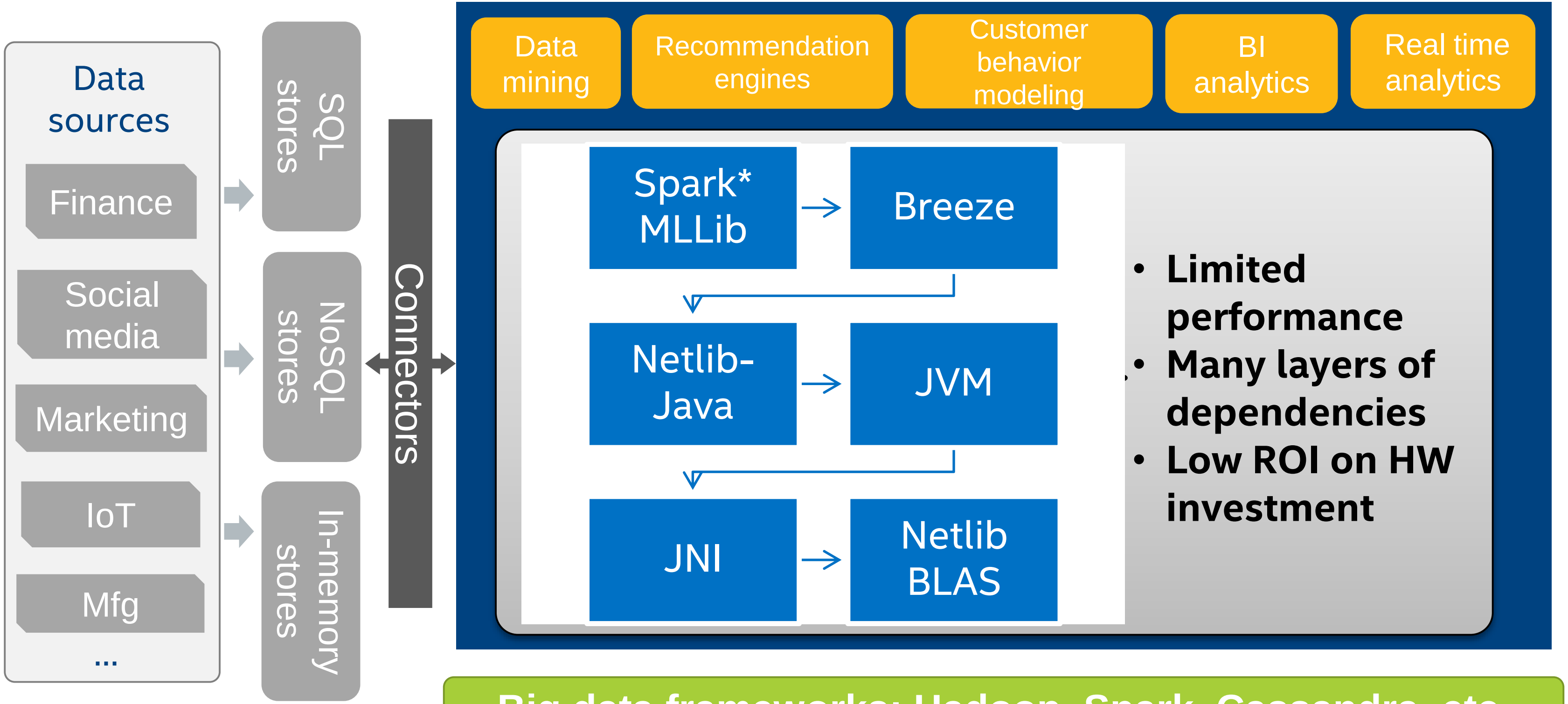
- 2 socket system with Intel® Xeon® Processor E5-2699 v4 (22 Cores, 2.2 GHz,), 128 GB memory, Red Hat* Enterprise Linux 6.7, BVLC Caffe, Intel Optimized Caffe framework, Intel® MKL 11.3.3, Intel® MKL 2017
- Intel® Xeon Phi™ Processor 7250 (68 Cores, 1.4 GHz, 16GB MCDRAM), 128 GB memory, Red Hat* Enterprise Linux 6.7, Intel® Optimized Caffe framework, Intel® MKL 2017

All numbers measured without taking data manipulation into account.

Agenda

- Intel® Math Kernel Library (Intel® MKL)
  - Overview, what's new in Intel MKL v.2017
  - Intel® MKL for machine learning
  - Neural network primitives in Intel MKL 2017 for deep learning
  - Introduction to open source Intel MKL-DNN
- Intel® Data Analytics Acceleration Library (Intel® DAAL)
- Intel® DAAL – Case Studies

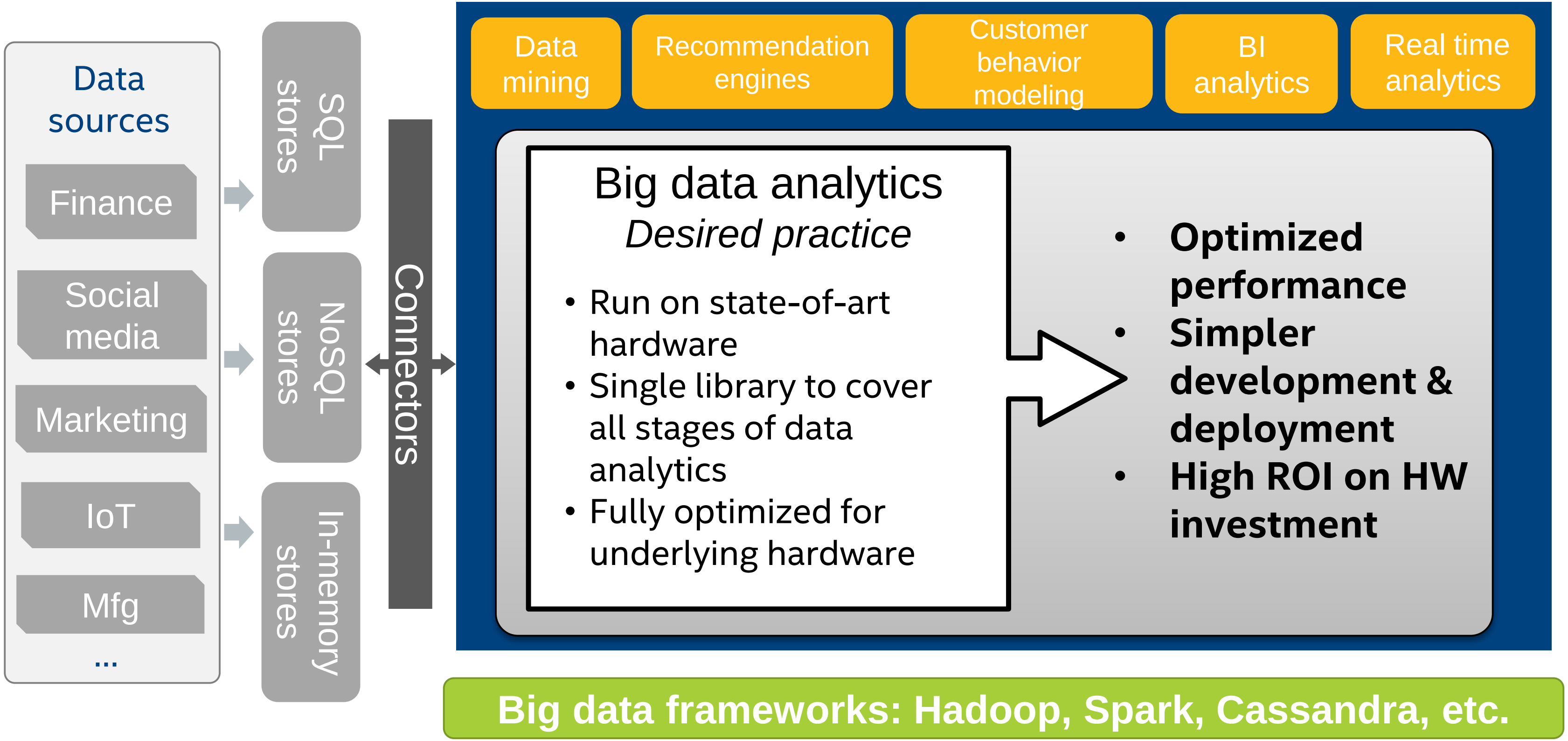
Problem Statement



- **Limited performance**
- **Many layers of dependencies**
- **Low ROI on HW investment**

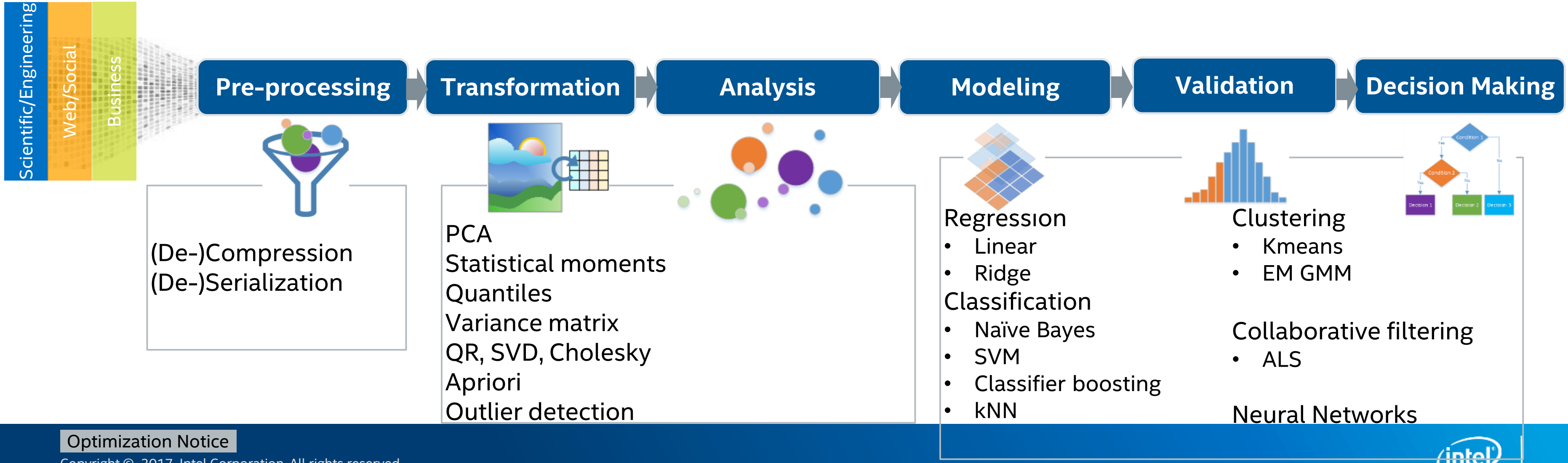
Big data frameworks: Hadoop, Spark, Cassandra, etc.

Desired Solution

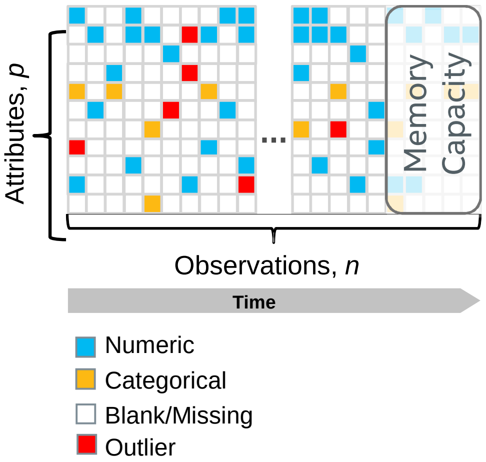


Intel® Data Analytic Acceleration Library

- Targets both data centers (Intel® Xeon® and Intel® Xeon Phi™) and edge-devices (Intel® Atom)
- Perform analysis close to data source (sensor/client/server) to optimize response latency, decrease network bandwidth utilization, and maximize security
- Offload data to server/cluster for complex and large-scale analytics

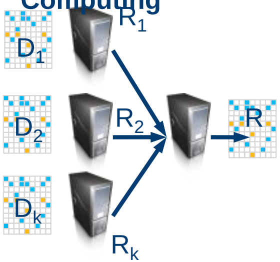


Intel® Data Analytics Acceleration Library



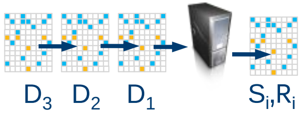
Big Data Attributes	Solution
Volume: • Huge data not fitting into node/device memory/distributed across nodes	• Distributed computing (e.g. communication-avoiding algorithms), streaming algorithms
Velocity: • Data arriving in time	• Data buffering, streaming algorithms
Variety: • Non-homogeneous/sparse/missing/noisy data	• Categorical→Numeric (counters, histograms, etc) • Homogeneous numeric data kernels • Conversions, Indexing, Repacking • Sparse data algorithms • Recovery methods (bootstrapping, outlier correction)

Distributed Computing



$R = F(R_1,\ldots,R_k)$

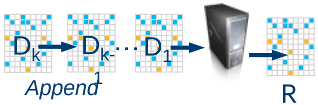
Streaming Computing



$S_{i+1} = T(S_i,D_i)$

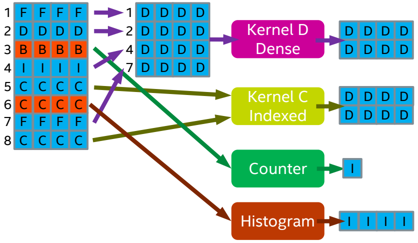
$R_{i+1} = F(S_{i+1})$

Batch Computing

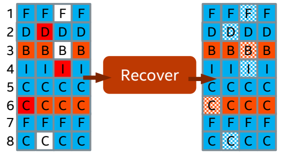


$R = F(D_1,\ldots,D_k)$

Converts, Indexing, Repacking



Data Recovery



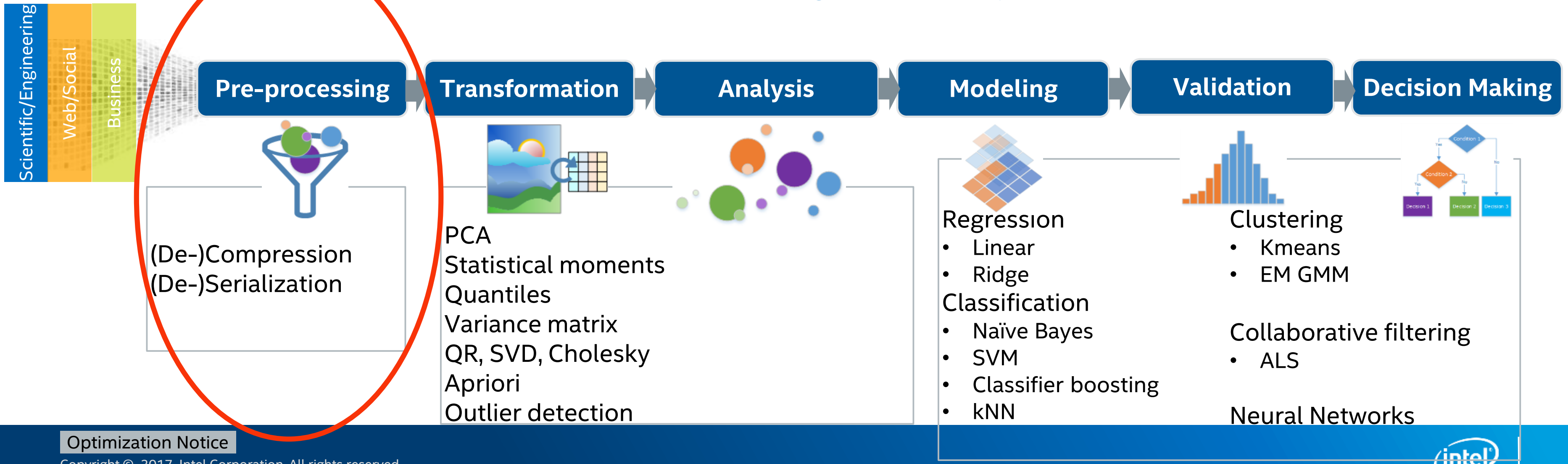
Intel® Data Analytics Acceleration Library

	Algorithms	Batch	Distributed	Online
Descriptive statistics	Low order moments	√	√	√
	Quantiles/sorting	√		
Statistical relationships	Correlation / Variance-Covariance	√	√	√
	(Cosine, Correlation) distance matrices	√		
Matrix decomposition	SVD	√	√	√
	Cholesky	√		
	QR	√	√	√
Regression	Linear/ridge regression	√	√	√
Classification	Multinomial Naïve Bayes	√	√	√
	SVM (two-class and multi-class)	√		
	Boosting (Ada, Brown, Logit)	√		
Unsupervised learning	Association rules mining (Apriori)	√		
	Anomaly detection (uni-/multi-variate)	√		
	PCA	√	√	√
	KMeans	√	√	
	EM for GMM	√		
Recommender systems	ALS	√	√	
Deep learning	Fully connected, convolution, normalization, activation layers, model, NN, optimization solvers,	√		

Optimization Notice

Intel® Data Analytic Acceleration Library

- Targets both data centers (Intel® Xeon® and Intel® Xeon Phi™) and edge-devices (Intel® Atom)
- Perform analysis close to data source (sensor/client/server) to optimize response latency, decrease network bandwidth utilization, and maximize security
- Offload data to server/cluster for complex and large-scale analytics



Intel® Integrated Performance Primitives, v.2017

Image Processing

- Geometry transformations
- Linear and non-linear filtering
- Linear transforms
- Statistics and analysis
- Color models

Computer Vision

- Feature detection
- Objects tracking
- Pyramids functions
- Segmentation, enhancement
- Camera functions
- And more

Signal Processing

- Transforms
- Convolution, Cross-Correlation
- Signal generation
- Digital filtering
- Statistical

Data Compression

- LZSS
- LZ77(ZLIB)
- LZO
- Bzip2

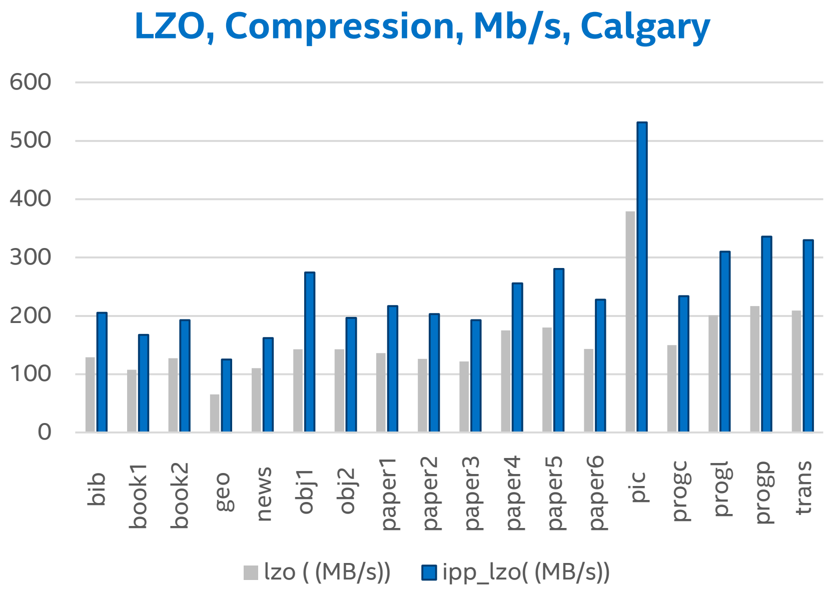
Cryptography

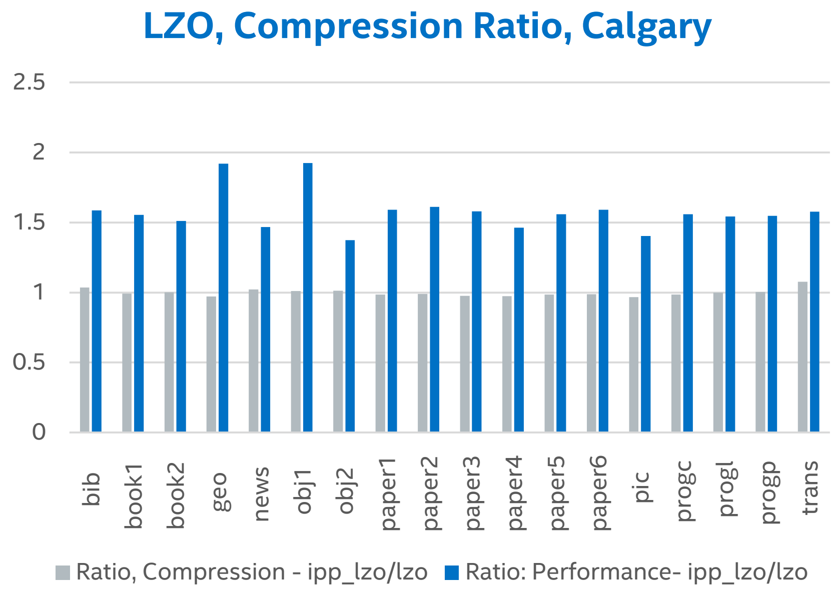
- Symmetric cryptography
- Hash functions
- Data authentication
- Public key

String Processing

- String Functions: Find, Insert, Remove, Compare, etc.
- Regular expression

Intel® IPP v.2017 update 2, LZO optimization



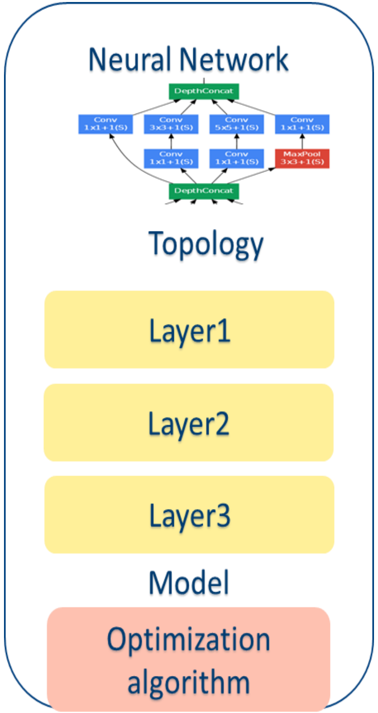


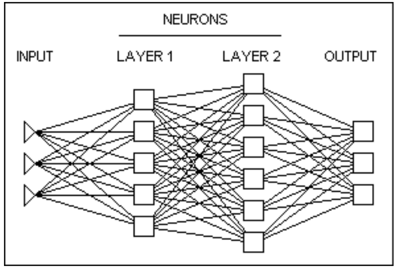
Configuration Info - Versions: Intel® Integrated Performance Primitives Library 2017 update 2;
Hardware: Intel® Xeon® Processor E5-2699 v3, 2 Eighteen-core CPUs (45MB LLC, 2.3GHz), 128GB of RAM per node; Operating System: CentOS 6.6 x86_64.
Software and workloads used in performance tests may have been 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. * Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

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® Data Analytics Acceleration Library 2017

- **Major new functionality Python APIs**
  - Python APIs (in addition to existing C++ and Java APIs)
- Support building neural networks for deep learning applications
    - Layers: convolution, pooling, fully connected, locally connected, dropout, etc.
    - Activation functions: logistic regression, hyperbolic tangent, ReLU, pReLU, soft ReLU, etc.
    - Function optimizations: SGD, L-BFGS, minibatch, Adagrad, etc.
  - Support more types of data sources (Attribute-Relation File Format, PostgreSQL, asynchronous)
- Open source version under Apache 2.0 license





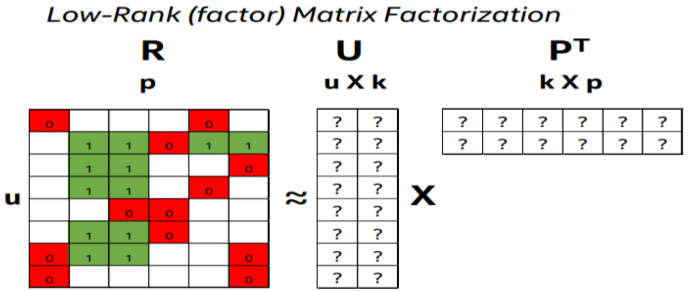
Agenda

- Intel® Math Kernel Library (Intel® MKL)
  - Overview, what's new in Intel MKL v.2017
  - Intel® MKL for machine learning
  - Neural network primitives in Intel MKL 2017 for deep learning
  - Introduction to open source Intel MKL-DNN
- Intel® Data Analytics Acceleration Library (Intel® DAAL)
- Intel® DAAL – Case Studies

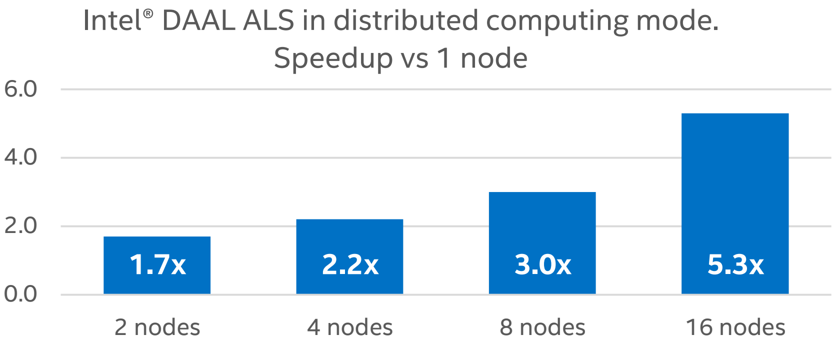
Intel®DAAL::Alternating Least Square (ALS) Algorithm 1/2

Alternating Least Square algorithm

- Moves to low-dimensional latent feature space and factorize: R = U x P^T
  - U - matrix of size u x k, association (user, feature)
  - P – matrix of size p x k, association (product, feature)
- Iterative algorithm minimizing least square error
  - Initializes U with random data, calculates P
  - Fixes P and calculates U
  - Matrix decompositions and linear solvers
- ***Computation: Matrix Factorization, Linear Solvers, Dot-Products and ….***

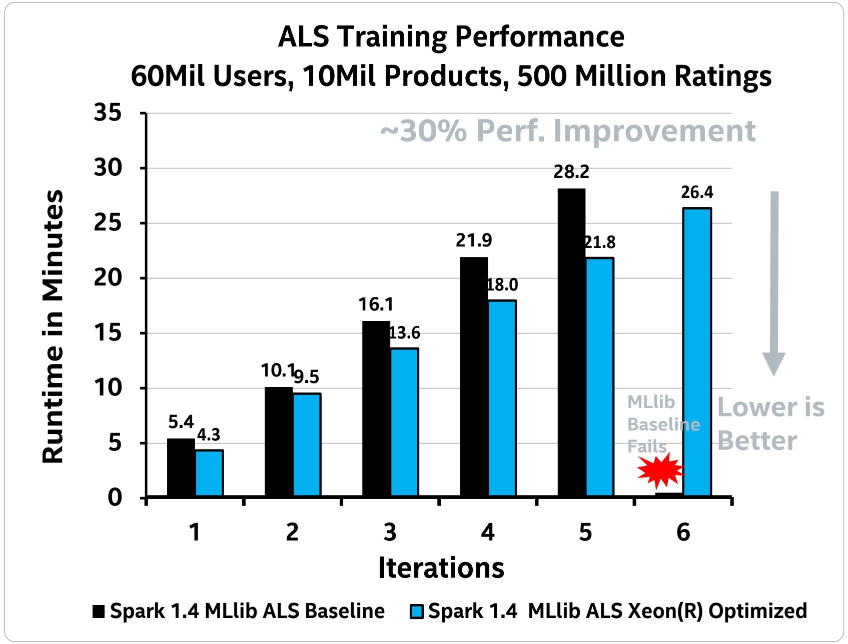


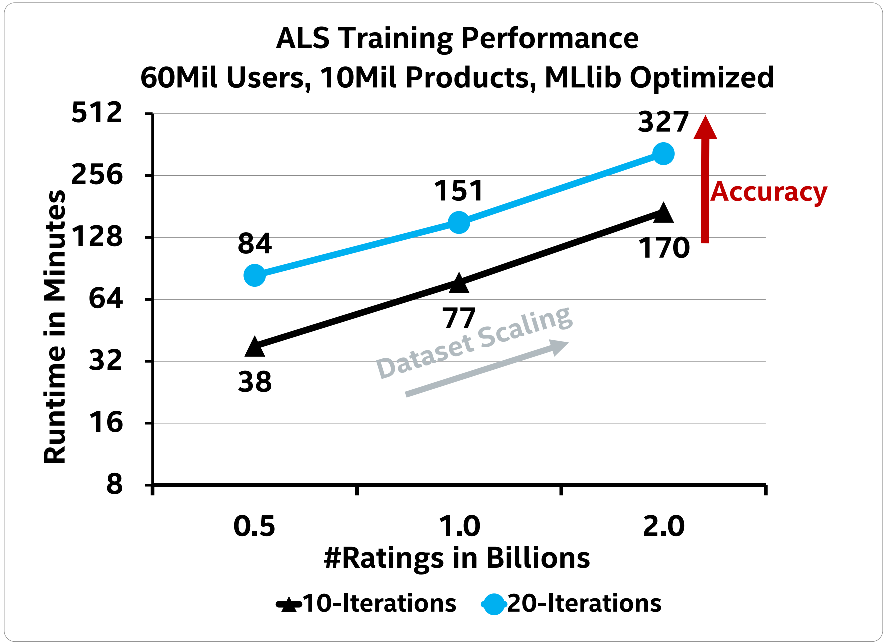
$$\min \sum_{ij} (r_{ij} - u_i p_j^T)^2 + \lambda \left(\sum_i \| r_i \|^2 + \sum_j \| p_j \|^2 \right)$$



Configuration Info: HW (each node): Intel(R) Xeon(R) CPU E5-2697 v4 @ 2.30GHz, 2x18 cores, HT is ON, RAM 128GB; Versions: Oracle Linux Server 6.6, Intel® DAAL 2017 Gold, Intel® MPI 5.1.3; Interconnect: 1 GB Ethernet. 10M users, 10M items, 100M ratings, 10 factors 15 iterations

Intel® DAAL::Alternating Least Square (ALS) Algorithm 2/2

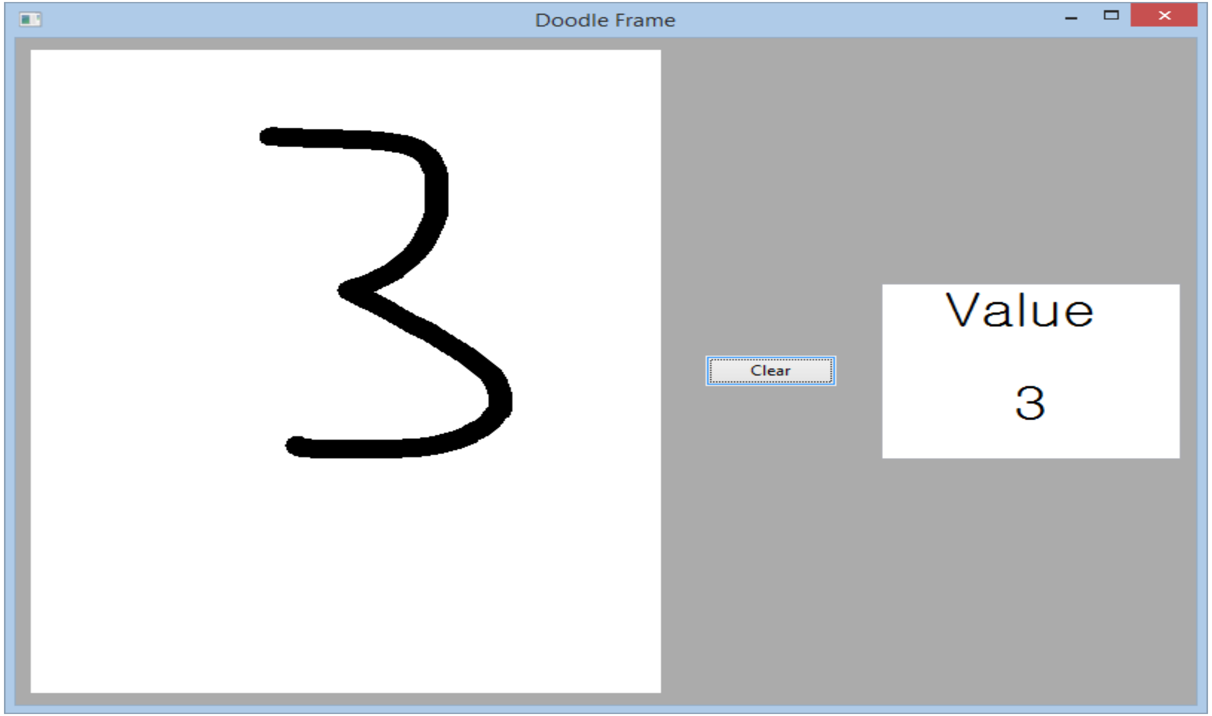




- Significant performance & scalability with Cluster HW, Application & Spark optimization
- Linear dataset scalability and accuracy

Optimizations achieve good performance improvement and linear dataset scalability

Intel® DAAL - Handwritten Digit Recognition



Intel® DAAL - Handwritten Digit Recognition

Training multi-class SVM for 10 digits recognition.

3,823 pre-processed training data.

- available at http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

99.6% accuracy with 1,797 test data from the same data provider.

Confusion matrix:

177.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000
0.000	181.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
0.000	2.000	173.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000
0.000	0.000	0.000	176.000	0.000	1.000	0.000	0.000	3.000	3.000
0.000	1.000	0.000	0.000	179.000	0.000	0.000	0.000	1.000	0.000
0.000	0.000	0.000	0.000	0.000	180.000	0.000	0.000	0.000	2.000
0.000	0.000	0.000	0.000	0.000	0.000	180.000	0.000	1.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	170.000	1.000	8.000
0.000	3.000	0.000	0.000	0.000	0.000	0.000	0.000	166.000	5.000
0.000	0.000	0.000	2.000	0.000	1.000	0.000	0.000	2.000	175.000

Average accuracy: 0.996
Error rate: 0.004
Micro precision: 0.978
Micro recall: 0.978
Micro F-score: 0.978
Macro precision: 0.978
Macro recall: 0.978
Macro F-score: 0.978

Training Handwritten Digits

```
void trainModel()
{
        /* Initialize FileDataSource<CSVFeatureManager> to retrieve input data from .csv file */
        FileDataSource<CSVFeatureManager> trainDataSource(trainDatasetFileName,
                DataSource::doAllocateNumericTable, DataSource::doDictionaryFromContext);

        /* Load data from the data files */
        trainDataSource.loadDataBlock(nTrainObservations);

        /* Create algorithm object for multi-class SVM training */
        multi_class_classifier::training::Batch<> algorithm;

        algorithm.parameter.nClasses = nClasses;
        algorithm.parameter.training = training;

        /* Pass training dataset and dependent values to the algorithm */
        algorithm.input.set(classifier::training::data,trainDataSource.getNumericTable());

        /* Build multi-class SVM model */
        algorithm.compute();

        /* Retrieve algorithm results */
        trainingResult = algorithm.getResult();

        /* Serialize the learned model into a disk file */
        ModelFileWriter writer("./model");
        writer.serializeToFile(trainingResult->get(classifier::training::model));
}
```

Create a numeric table

Create an alg. Obj.

Set input and parameters

Compute

Serialize the learned model

Handwritten Digit Prediction

```
void testDigit()
{
        /* Initialize FileDataSource<CSVFeatureManager> to retrieve the test data
from .csv file */
        FileDataSource<CSVFeatureManager> testDataSource(testDatasetFileName,
                DataSource::doAllocateNumericTable, DataSource::doDictionaryFromContext);
        testDataSource.loadDataBlock(1);

        /* Create algorithm object for prediction of multi-class SVM values */
        multi_class_classifier::prediction::Batch<> algorithm;

        algorithm.parameter.prediction = prediction;

        /* Deserialize a model from a disk file */
        ModelFileReader reader("./model");
        services::SharedPtr<multi_class_classifier::Model> pModel(new
multi_class_classifier::Model());
        reader.deserializeFromFile(pModel);

        /* Pass testing dataset and trained model to the algorithm */
        algorithm.input.set(classifier::prediction::data,
testDataSource.getNumericTable());
        algorithm.input.set(classifier::prediction::model, pModel);

        /* Predict multi-class SVM values */
        algorithm.compute();

        /* Retrieve algorithm results */
        predictionResult = algorithm.getResult();

        /* Retrieve predicted labels */
        predictedLabels = predictionResult->get(classifier::prediction::prediction);
}
```

Deserialize learned model

Summary

- Intel® DAAL+Intel® MKL+Intel® IPP = Complementary Big Data Libraries Solution
- Intel MKL, Intel DAAL and Intel IPP provide high performance and optimized building blocks for data analytics and machine learning algorithms on Intel platforms.
- Deep learning applications can benefit from DNN primitives in Intel MKL 2017 and Neural Networks API in Intel DAAL 2017

Intel® MKL Resources

Intel® MKL website, forum, benchmarks

- https://software.intel.com/en-us/intel-mkl
- https://software.intel.com/en-us/forums/intel-math-kernel-library
- https://software.intel.com/en-us/intel-mkl/benchmarks#

Intel® MKL link line advisor

- http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Intel® MKL-DNN

- https://01.org/mkl-dnn

Intel® IA optimized frameworks

- https://github.com/intel/caffe
- https://github.com/intel/theano

Intel® DAAL Resources

Intel® Machine Learning

- http://www.intel.com/content/www/us/en/analytics/machine-learning/overview.html

Intel® DAAL website

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

Intel® DAAL forum

- https://software.intel.com/en-us/forums/intel-data-analytics-acceleration-library

Intel® DAAL blogs

- https://software.intel.com/en-us/blogs/daal
- https://01.org/daal/blogs/kmoffat/2016/intel%C2%AE-daal-and-intel%C2%AE-mkl-%E2%80%93-complementary-high-performance-machine-learning

Legal Disclaimer and Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

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

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details.

The Intel Core and Itanium processor families may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

The code names Arrandale, Bloomfield, Boazman, Boulder Creek, Calpella, Chief River, Clarkdale, Cliffside, Cougar Point, Gulftown, Huron River, Ivy Bridge, Kilmer Peak, King's Creek, Lewisville, Lynnfield, Maho Bay, Montevina, Montevina Plus, Nehalem, Penryn, Puma Peak, Rainbow Peak, Sandy Bridge, Sugar Bay, Tylersburg, and Westmere presented in this document are only for use by Intel to identify a product, technology, or service in development, that has not been made commercially available to the public, i.e., announced, launched or shipped. It is not a "commercial" name for products or services and is not intended to function as a trademark.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel, Intel Core, Core Inside, Itanium, and the Intel Logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Optimization Notice

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