INTEL® NERVANA™ GRAPH
A UNIVERSAL DEEP LEARNING COMPILER
MOTIVATION
DEEP LEARNING ECOSYSTEM - A MANY TO MANY PROBLEM

Users

Frameworks

Hardware

O(mn)
engineering effort

Neon™
Caffe²
MXNet
TensorFlow

Intel® Nervana™ Neural Network Processor

Intel® Xeon®

GPU

Intel® Xeon®

Intel and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

neon is a trademark of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
DEEP LEARNING HARDWARE - A MANY TO MANY PROBLEM

Users

Frameworks

O(m+n) engineering effort

Hardware
THE INTEL® NERVANA™ GRAPH PROJECT

- An intermediate representation (IR) for deep learning
- Compiler backends (transformers)
- Frontend “connectors”
INTEL® NERVANA™ GRAPH – JUST ONE MORE LAYER
INTEL® NERVANA™ GRAPH USERS

• Framework developers
  • Create special purpose or new frameworks with high performance across many platforms easily
• Hardware and System Software developers
  • Commit once and plug into the ecosystem
• Optimization experts
  • Implement/leverage this optimization technique once and propagate across other platforms
• Deployment/Operations
  • Take a model and deploy it somewhere; now people can export models from other frameworks
INTEL® NERVANA™ GRAPH – IR AND FRONTENDS

Intermediate representation and deep learning library “connectors”
INTEL® NERVANA™ GRAPH IR

- Dataflow graph
  - With support for control dependencies for side-effecting operations
- Rectangles are sources
  - Trainable (variable)
    - External (placeholder)
  - Constant
- Ellipses are operations
- Arrows show data inputs
INTEL® NERVANA™ GRAPH IR - CURRENTLY

- Try to strike a balance
  - Small enough to avoid ‘op creep’
  - Large enough to maintain performance
- Tensor math
  - Unary/binary element wise, Reductions
- Tensor manipulation (slicing, broadcasting)
- Control flow (parallel, sequential)
- Data mutation (Assignment)
• More control flow
  • Limited looping (while)
  • Iterate over selectable axes
• Reductions and maps
  • With user specified sub computations
• With the goal of:
  • Enable people to avoid writing x86 assembly, CUDA, etc and still get robust performance for new recurrent kernels and layer types
Tensor dimension management (dimshuffles, axis ordering, ...)
- Pain point for end users
- Difficult for device specific layout optimizations

Intel® Nervana™ Graph introduces named Axes
- Give meaningful names to axes for more meaning
- Optional for frontends without support
- Enables compiler to perform more static verification/analysis for the user
def sigmoid(x):
    return ng.reciprocal(ng.exp(-x) + 1)

X = ng.placeholder(axes=[ax.C, ax.N])
Y = ng.placeholder(axes=[ax.N])
W = ng.variable(axes=[ax.C], initial_value=0)
b = ng.variable(axes=ax.C, initial_value=0)
Y_hat = ng.sigmoid(ng.dot(W, X) + b)
L = ng.cross_entropy(Y_hat, Y) / ng.batch_size(Y_hat)
**WORKING WITH THE GRAPH**

- Find all variables that contribute to an Op
  - `Y_hat.variables()`
- Graph traversal and mutation
  - `op_list = ng.ordered_ops(cost)`
- Generate graph for the derivative of one Op with respect to another
  - `grad = ng.deriv(L, W)`
  - Uses reverse-mode autodiff backprop
INTEL® NERVANA™ GRAPH CONNECTORS

• Originally: Converters
  • Start with graph or layer representation of source framework
    • Protobuf output of TensorFlow
    • Prototxt of Caffe
  • Pattern match operations to one or more ngraph Ops
• Now: seamless interop with TensorFlow and MXNet
  • By obtaining graph internally to the framework allows for reuse of host framework's API's, serialization, etc.
ONNX – OPEN NEURAL NETWORK EXCHANGE

• Designed as an interchange format between frameworks
• Initial release from Microsoft and Facebook:
  • Protocol buffer based definition of dataflow graph
  • Initial set of deep learning operators
• Inference only for now
• No optimizers or compiler ecosystem
• Intel® is participating in open design process
INTEL® NERVANA™ GRAPH STATUS – FRONTEND AND IR

• Currently
  • Python API and implementation
  • neon on ngraph running with MLP, Convolutional, RNN, GANs, ...
  • Serialization, visualization (including Tensorboard support), CSS style selectors
  • Tensorflow XLA connector POC

• In Progress
  • C/C++ API and port
  • Full TensorFlow XLA backend and MXNet integration
  • External Op (FFI) support (support custom user kernels analogous to inline assembly in C/C++)
INTEL® NERVANA™ GRAPH - BACKENDS

Graph compilation
Graph Transformation/Compilation

- Design inspired by the LLVM project
- Uses a series of passes to transform the graph from abstract tensor operations into an executable primitive
- Example Passes:
  - Arithmetic simplifications, eg: \( \log(\exp(x)) \rightarrow x \)
  - Dimension reduction and element layout
  - Storage planning
  - Maintain/exploit parallelism opportunities
WHY NOT USE EXISTING COMPILERS? (LLVM, ICC)

- Operations are primarily tensor operations
  - Tensor == Large multidimensional (often aliased) array
  - Fairly regular structure at this level
- Many optimizations at the tensor level
  - Horizontal fusion
  - Memory liveness for large tensors (rather than registers)
- Can still leverage these compilers within codegen of transformers
  - POC for C++ code gen and JIT using LLVM
How does this compare to CUDA, cuDNN, MKL-DNN?

- CUDA, cuDNN, and MKL-DNN offer low levels of abstraction
  - ‘Raw’ matrix multiplies and convolutions
  - Deep learning practitioners usually don’t need this amount of control
- Many ways to hurt performance when working at this level
  - Memory layout/allocation strategies
  - Operation fusion
Why neon vs TensorFlow, Caffe, etc?

- We plan for neon to set the bar for performance in the industry.
- Innovation laboratory for deep learning infrastructure
  - Axes
  - Containers
  - Dynamic networks
• Transformers
  • GPU transformer using Nervana™ GPU Kernels on CUDA
  • CPU transformer using MKL-DNN and Numpy
  • Heterogeneous distributed training
  • Crest

• Common optimization pass API
  • Operator fusion
  • Pattern matching utilities
  • Memory sharing
  • Layout
INTEL® NERVANA™ GRAPH ROADMAP - HIGHLIGHTS

• 2017Q4
  • Seamless interop with TensorFlow through XLA backend
  • Multi-device training support via Heterogeneous Transformer
  • Multi-node training for small (~8) number of nodes
  • Performance milestones

• And Beyond
  • More frontends: MXNet, CNTK, Pytorch
  • More backends
  • Deployment optimizations: quantization and pruning
 MORE INFORMATION 

 Github Repository 

 https://github.com/NervanaSystems/ngraph 

 Documentation 

 https://ngraph.nervanasys.com/docs/latest/
STAY CONNECTED

@intelnervana
@intelAI
#intelAI
intelnervana.com
This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

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. No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

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

Statements in this document that refer to Intel’s plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel’s results and plans is included in Intel’s SEC filings, including the annual report on Form 10-K.

The products described 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.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

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

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon and others are trademarks of Intel Corporation in the U.S. and/or other countries.

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

© 2017 Intel Corporation.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel® Nervana™ Graph</th>
<th>XLA</th>
<th>TVM</th>
<th>Halide</th>
<th>Tensor RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework independence</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Framework connectors</td>
<td>✓</td>
<td>✓ (only 1)</td>
<td>~</td>
<td>~</td>
<td>✓</td>
</tr>
<tr>
<td>Hardware independence</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Leverage existing tensor libraries</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inference and training</td>
<td>✓</td>
<td>✓*</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Production ready</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>