ACAT 2017
22 August 2017
University of Washington, Seattle

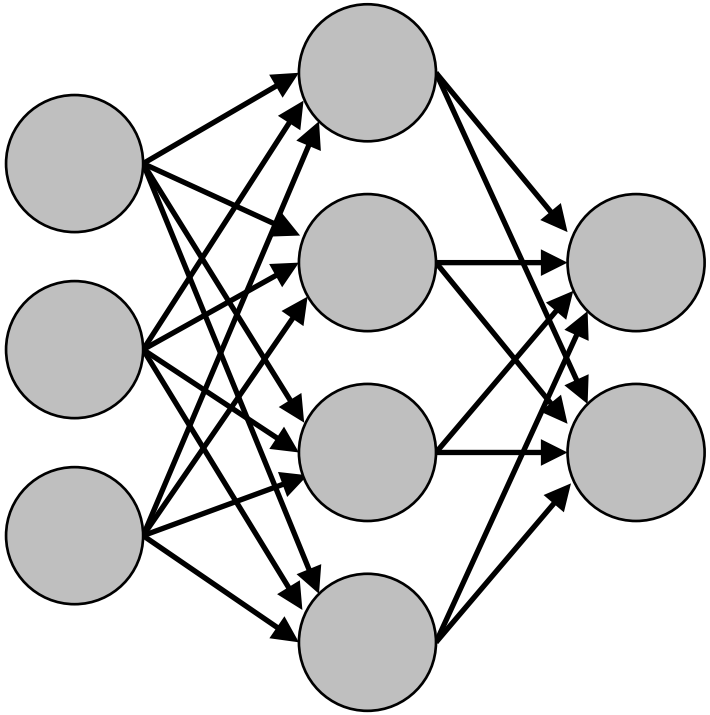
DEEP LEARNING ON THE INTEL® NERVANA™ PLATFORM

Ravi Panchumarthy, PhD

Machine Learning Engineer, Intel®

Outline

- **Intel® Nervana™ Overview**
- Intel's Deep Learning Hardware
- Intel's Deep Learning Software
  - MKL-DNN
  - Neon and Nervana Cloud
  - Nervana Graph



Intel® Nervana™ Deep Learning Platform

Compressing the "innovation cycle" to deliver custom, high-accuracy, enterprise-grade DL solutions in record time


Images

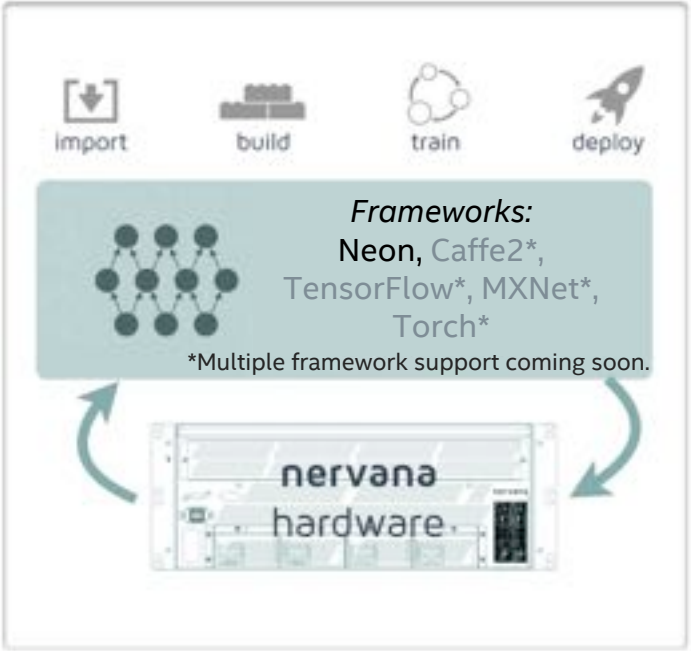

Video


Text


Speech


Tabular


Time series

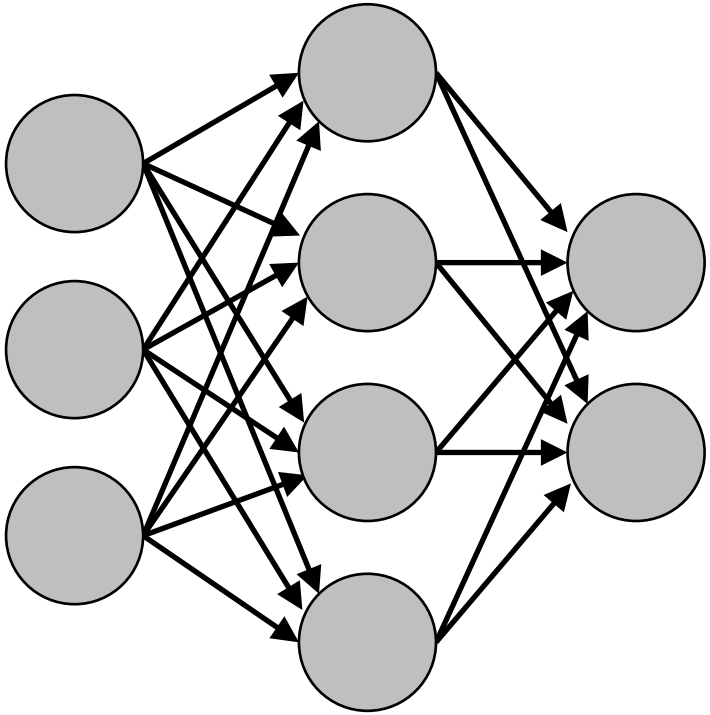



Solutions

Hosted Nervana Cloud *-or-*
On-premise Nervana DL Appliance *-or-*
Single-user version

Outline

- Intel® NervanaTM Overview
- **Intel's Deep Learning Hardware**
- Intel's Deep Learning Software
  - MKL-DNN
  - Neon and Nervana Cloud
  - Nervana Graph



End-to-End AI Compute

DATACENTER

Many-to-many hyperscale for stream and massive batch data processing

Ethernet & Wireless

GATEWAY

1-to-many with majority streaming data from devices

Wireless and non-IP wired protocols

- ✓ *Secure*
- ✓ *High throughput*
- ✓ *Real-time*

EDGE

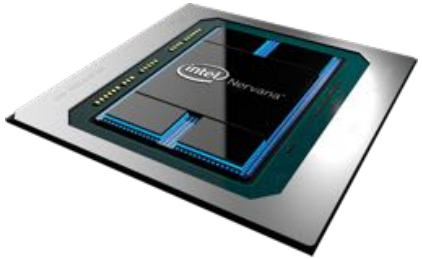
1-to-1 devices with lower power and often UX requirements

- Intel® Xeon® Processors
- Intel® Xeon Phi™ Processors*
- Intel® Core™ & Atom™ Processors
- CPU+ Intel® Processor Graphics
- Intel® FPGA
- Crest Family (Nervana ASIC)*
- Movidius Myriad (VPU)
- Intel® GNA (IP)*

CREST FAMILY

1ST SILICON 2018

DEEP LEARNING BY DESIGN



Scalable acceleration with best performance for intensive deep learning training & inference, period

CUSTOM HARDWARE

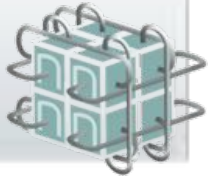
- Unprecedented compute density

BLAZING DATA ACCESS

- 32 GB of in package memory via HBM2 technology
- 8 Tera-bits/s of memory access speed

HIGH-SPEED SCALABILITY

- 12 bi-directional high-bandwidth links
- Seamless data transfer via interconnects



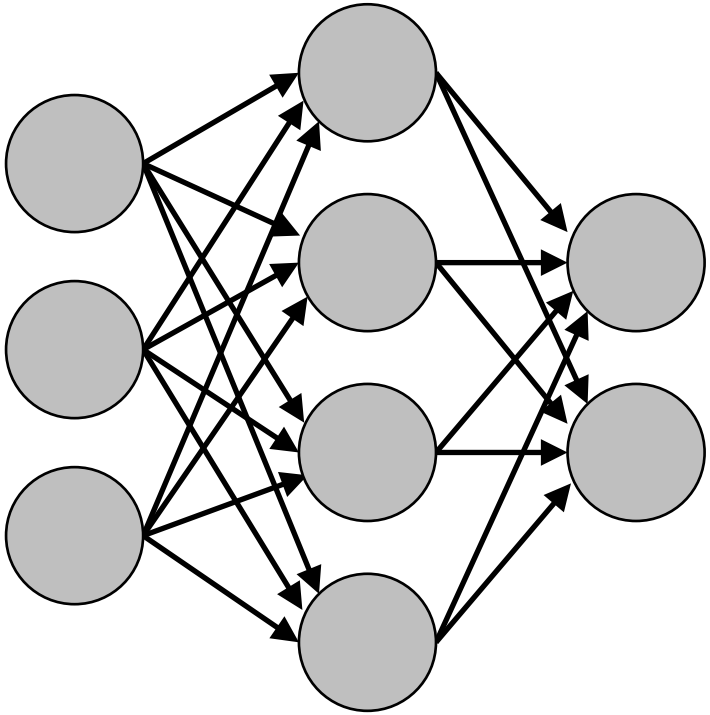
[1]Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance Software and workloads used in performance 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. For more complete information visit: http://www.intel.com/performance Source: Intel measured as of November 2016

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.

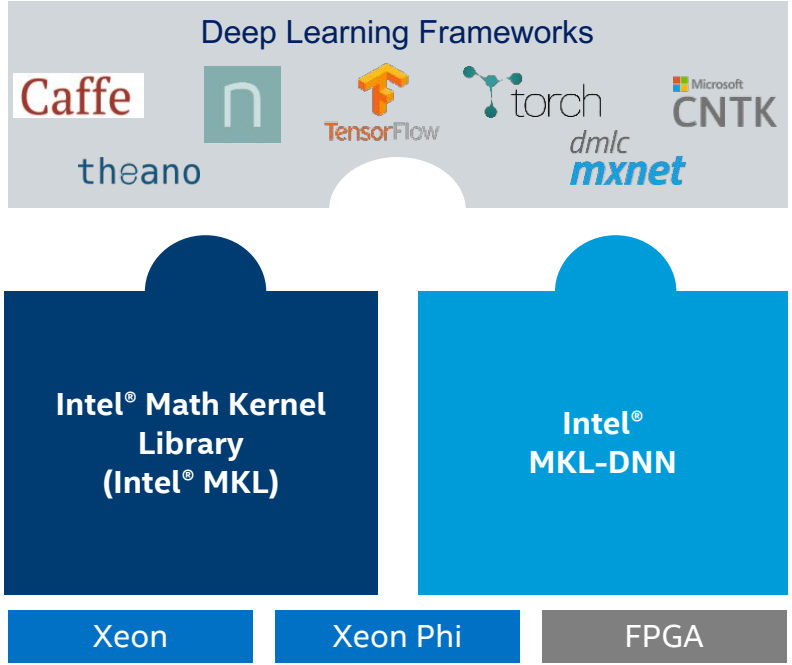
Notice Revision #20110804

Outline

- Intel® NervanaTM Overview
- Intel's Deep Learning Hardware
- **Intel's Deep Learning Software**
  - MKL-DNN
  - Neon and Nervana Cloud
  - Nervana Graph



Intel® MKL and Intel® MKL-DNN for Deep Learning



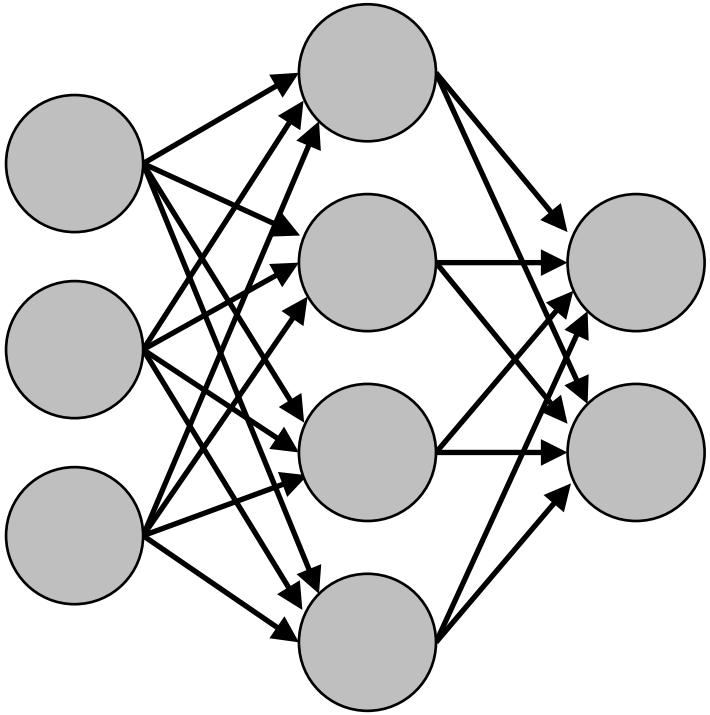
https://github.com/01org/mkl-dnn

Intel® MKL	Intel® MKL-DNN
DNN primitives + wide variety of other math functions	DNN primitives
C DNN APIs (C++ future)	C/C++ DNN APIs
Binary distribution	Open source DNN code*
Free community license. Premium support available as part of Parallel Studio XE	Apache 2.0 license
Broad usage DNN primitives; not specific to individual frameworks	Multiple variants of DNN primitives as required for framework integrations
Quarterly update releases	Rapid development ahead of Intel MKL releases

* GEMM matrix multiply building blocks are binary

Outline

- Intel® NervanaTM Overview
- Intel's Deep Learning Hardware
- **Intel's Deep Learning Software**
  - MKL-DNN
  - **Neon and Nervana Cloud**
  - Nervana Graph



Neon Framework

NervanaSystems / neon

Watch 339 | Star 3,158 | Fork 696

Code | Issues 61 | Pull requests 13 | Projects 0 | Wiki | Insights

Intel® Nervana™ reference deep learning framework committed to best performance on all hardware
http://neon.nervanasys.com/docs/latest

1,022 commits | 4 branches | 29 releases | 63 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jennifermyers committed on GitHub Update for v2.1.0 (#917) — Latest commit 5584328 12 days ago

bin	MKL backend (#783)	2 months ago
doc	Update for v2.1.0 (#917)	12 days ago
examples	Adding `numactl --interleave=all` Performance Best Practice Notes (#919)	12 days ago
loader	Makefiles now check for virtualenv and pkg-config. (#872)	24 days ago
neon	add mkl backend for neon tests (#881)	13 days ago
tests	add mkl backend for neon tests (#881)	13 days ago

Neon: Enterprise-class Deep Learning Library

OxfordNet [Model-A] - Input 64x3x224x224

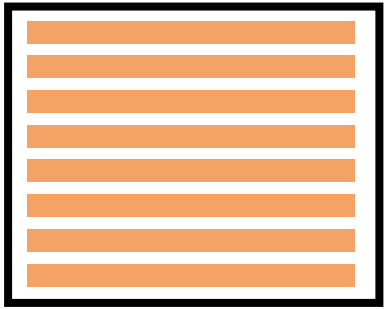
Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-neon-fp16	ConvLayer	254	82	171
Nervana-neon-fp32	ConvLayer	320	103	217
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	471	140	331
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	529	162	366
TensorFlow	conv2d	540	158	382
Chainer	Convolution2D	885	251	632

Third-party (Facebook) benchmarking

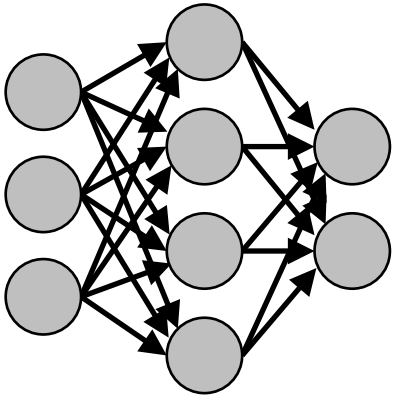
Why Python ?

- Popular, well established, developer familiarity
- Fast to prototype
- Rich ecosystem of existing packages.
- Data Science: pandas, pycuda, ipython, matplotlib, h5py, …
- Good “glue” language: scriptable plus functional and OO support, plays well with other languages

Neon Workflow



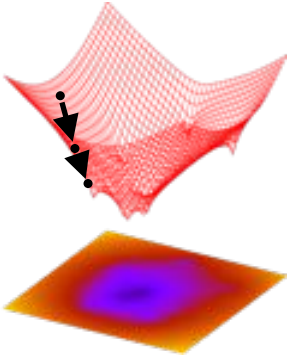
Dataset



Model/Layers

$$C(y,t)$$

Cost



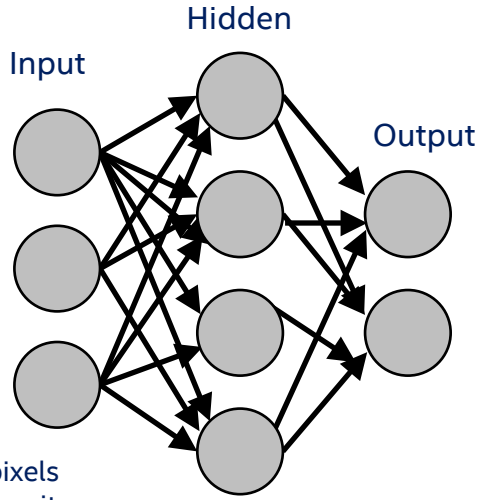
Optimizer

Neon Overview

Backend	NervanaGPU, NervanaCPU
Datasets	MNIST, CIFAR-10, Imagenet 1K, PASCAL VOC, Mini-Places2, IMDB, Penn Treebank, Shakespeare Text, bAbI, Hutter-prize, UCF101, flickr8k, flickr30k, COCO
Initializers	Constant, Uniform, Gaussian, Glorot Uniform, Xavier, Kaiming, IdentityInit, Orthonormal
Optimizers	Gradient Descent with Momentum, RMSProp, AdaDelta, Adam, Adagrad, MultiOptimizer
Activations	Rectified Linear, Softmax, Tanh, Logistic, Identity, ExpLin
Layers	Linear, Convolution, Pooling, Deconvolution, Dropout, Recurrent, Long Short-Term Memory, Gated Recurrent Unit, BatchNorm, LookupTable, Local Response Normalization, Bidirectional-RNN, Bidirectional-LSTM
Costs	Binary Cross Entropy, Multiclass Cross Entropy, Sum of Squares Error
Metrics	Misclassification (Top1, TopK), LogLoss, Accuracy, PrecisionRecall, ObjectDetection

Example: recognition of handwritten digits

MNIST dataset
70,000 images (28x28 pixels)
Goal: classify images into a digit 0-9



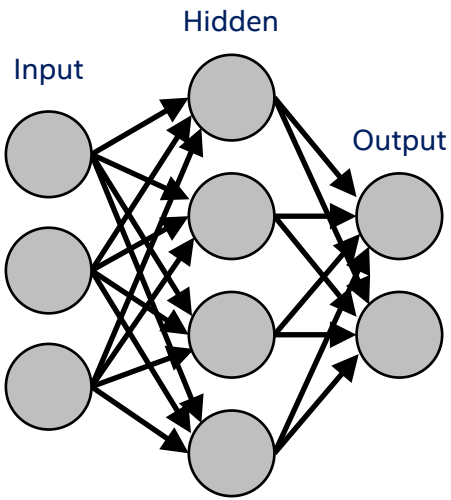
N = 10 **output** units
(one for each digit)

Each unit *i* encodes the probability of the input image of being of the digit *i*

N = 28 x 28 pixels
= 784 **input** units

N = 100 **hidden** units
(user-defined parameter)

MNIST




```
be = gen_backend(backend='cpu')
```

Generate backend

```
(X_train, y_train), (X_test, y_test), nclass = load_mnist()

train_set = ArrayIterator(X_train, y_train, nclass=nclass)
test_set = ArrayIterator(X_test, y_test, nclass=nclass)
```

Load data

```
init_norm = Gaussian(loc=0.0, scale=0.01)


layers = []
layers.append(Affine(nout=100, init=init_norm, activation=Rectlin()))
layers.append(Affine(nout=10, init=init_norm, activation=Softmax()))

mlp = Model(layers=layers)
```

Define model

```
cost = GeneralizedCost(costfunc=CrossEntropyMulti())
```

Cost

```
optimizer = GradientDescentMomentum(0.1, momentum_coef=0.9)
```

Optimizer

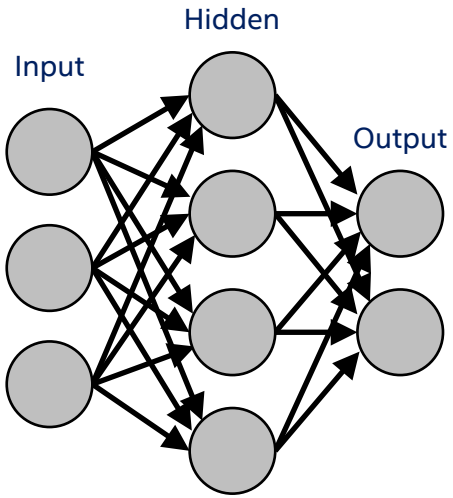
```
callbacks = Callbacks(mlp, eval_set=valid_set)
```

Callbacks

```
mlp.fit(train_set, optimizer=optimizer, num_epochs=10, cost=cost,
callbacks=callbacks)
```

Train model

MNIST



mlp.fit(train_set, optimizer=optimizer, num_epochs=10, cost=cost, callbacks=callbacks)

```
(.venv)Katherines-MacBook-Air:neon KatherineFritsch$ ./examples/mnist_mlp.py
Epoch 0   [Train |____________________|  469/469  batches, 0.45 cost, 1.44s]
Epoch 1   [Train |____________________|  469/469  batches, 0.16 cost, 1.51s]
Epoch 2   [Train |____________________|  469/469  batches, 0.16 cost, 1.58s]
Epoch 3   [Train |____________________|  468/468  batches, 0.23 cost, 1.58s]
Epoch 4   [Train |____________________|  469/469  batches, 0.26 cost, 1.58s]
Epoch 5   [Train |____________________|  469/469  batches, 0.08 cost, 1.58s]
Epoch 6   [Train |____________________|  469/469  batches, 0.13 cost, 1.58s]
Epoch 7   [Train |____________________|  468/468  batches, 0.24 cost, 1.58s]
Epoch 8   [Train |____________________|  469/469  batches, 0.24 cost, 1.57s]
Epoch 9   [Train |____________________|  469/469  batches, 0.06 cost, 1.57s]
Misclassification error = 2.5%
```

customizable callbacks

Neon: Example operations - command line arguments

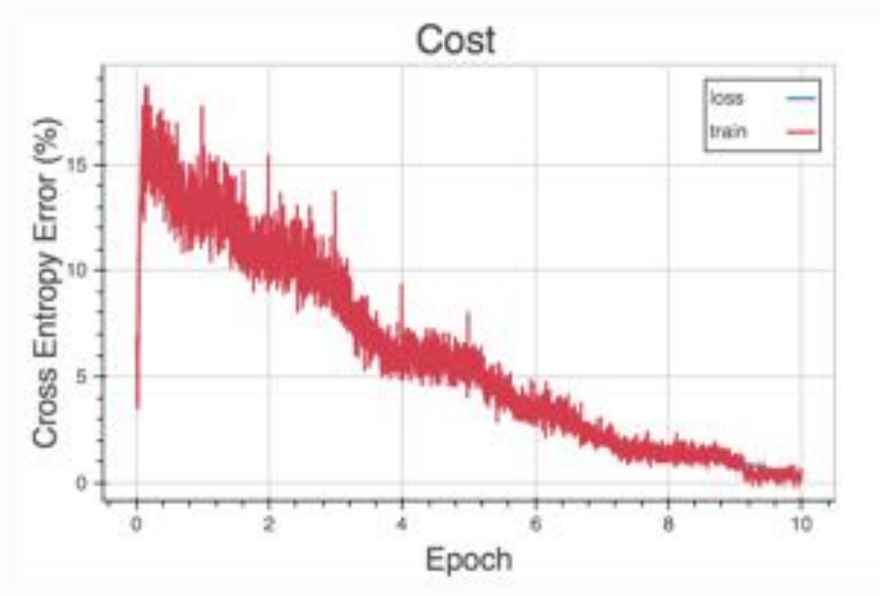
```
./examples/mnist_mlp.yaml -e 10 –s “mymodel.pkl” –r 0 –i 4
```

Flag	Description
-w, --data_dir	Path to data directory (default: nervana/data)
-e, --epochs	Number of epochs to run during training (default: 10)
-s, --save_path	Path to save the model snapshots (default: None)
-o, --output_file	Path to save the metrics and callback data generated during training. Can be used by nvis for visualization (default: None)
-b, --backend {cpu,gpu}	Which backend to use (default: cpu)
-z, --batch_size	Batch size for training (default: 128)

http://neon.nervanasys.com/index.html/

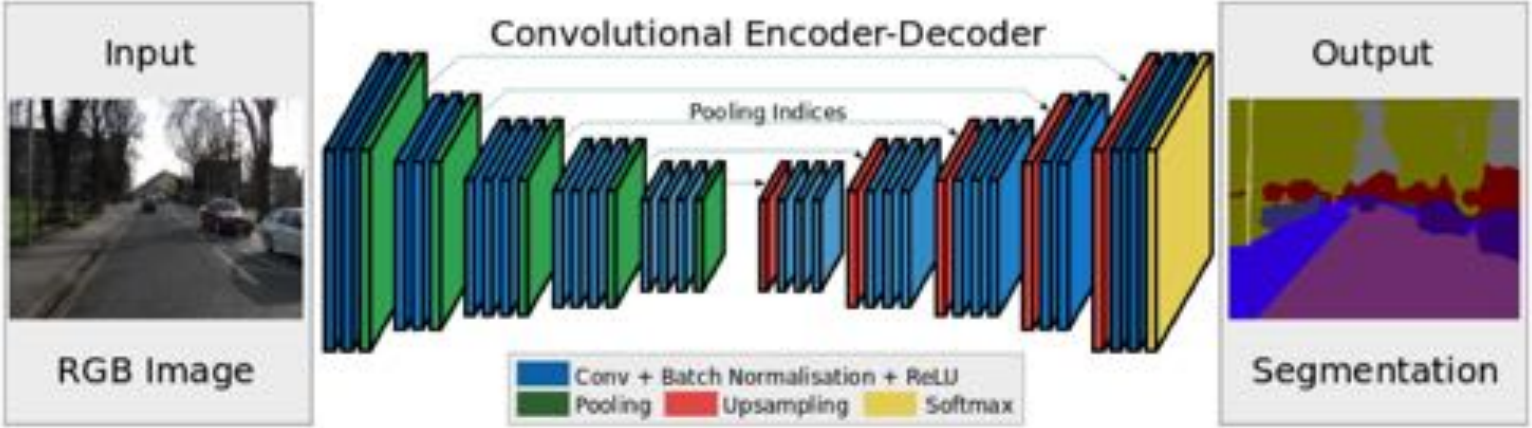
Neon: Example operations – Visualizing Results

```
./examples/mnist_mlp.yaml -o data.h5 nvis -i data.h5 -o viz/
```



http://neon.nervanasys.com/index.html/tools.html#visualization

Curated Models: Segnet



Badrinarayanan et al., 2015

	Neon (ms)	Caffe (ms)	Speed-up
Forward	101	719	7.1x
Backward	164	746	4.5x
Total	265	1455	5.5x

Number of GPUs	Iteration time per image	Speed-up
1	66 ms	-
2	35 ms	1.9 x
4	18 ms	3.7 x

Model Zoo

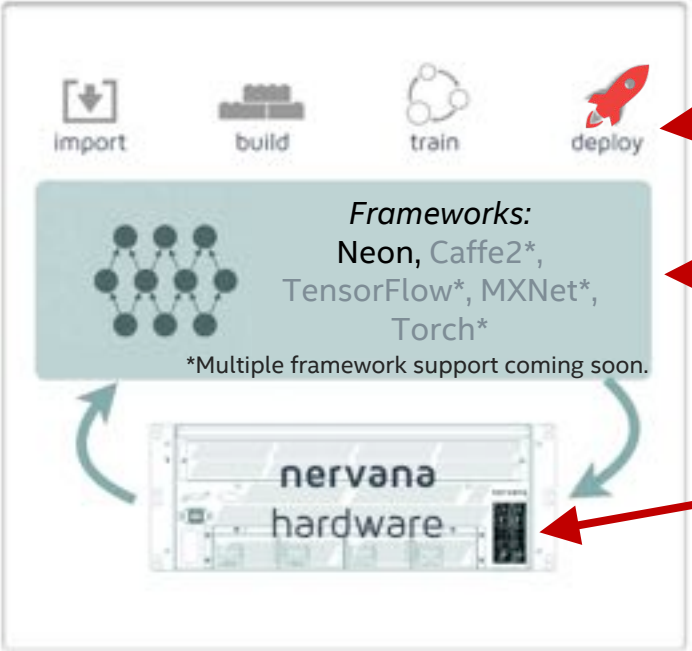
- https://github.com/NervanaSystems/ModelZoo
- Pre-trained weights and models

Alexnet GoogLeNet
AllCNN VGG
bAbI Q&A
Deep Residual Net
SegNet Video Activity Detection
Skip-thought imdb Sentiment Analysis
Deep Speech 2 LSTM Image Captioning
Autoencoders Deep Reinforcement Learning
Deep Dream Fast-RCNN Object Localization

Neon…

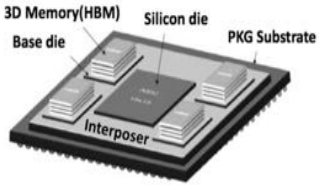
- Easy-to-use deep learning framework
- Populated model zoo with pre-trained models
- Integration within nervana cloud for easy deployment and experiment-tracking
- 10-20x speed-ups with forthcoming custom processor

Nervana Cloud



n cloud

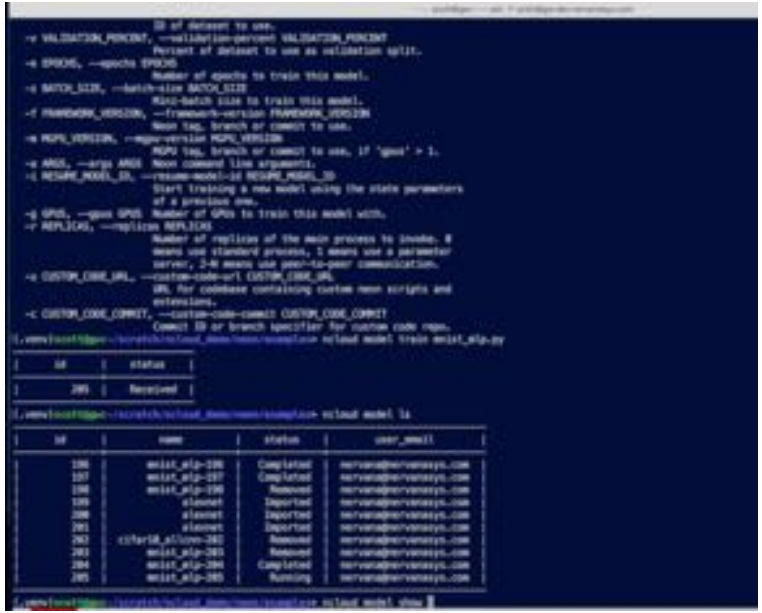
2-3x speedup on
Titan X GPUs



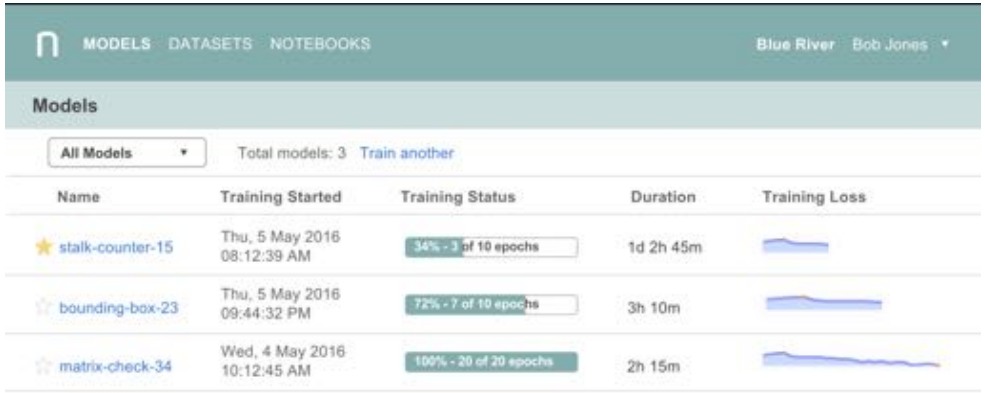
http://docs.cloud.nervanasys.com/index.html/

Nervana Cloud: Two Powerful Interfaces

Nervana ncloud CLI



Nervana Cloud Web UX



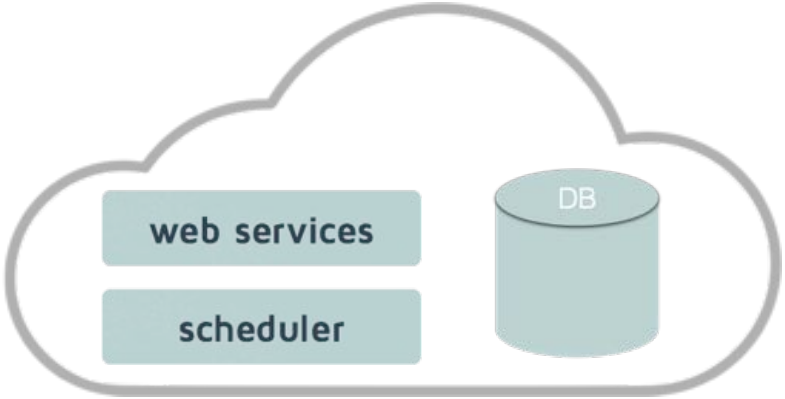
http://docs.cloud.nervanasys.com/index.html/

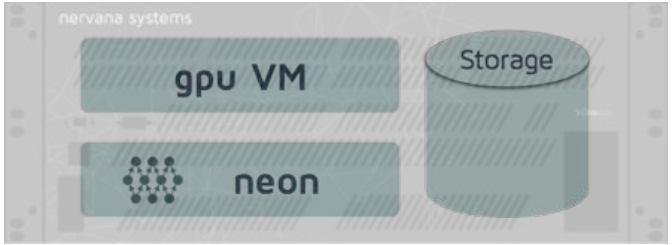
Nervana Cloud

CLI

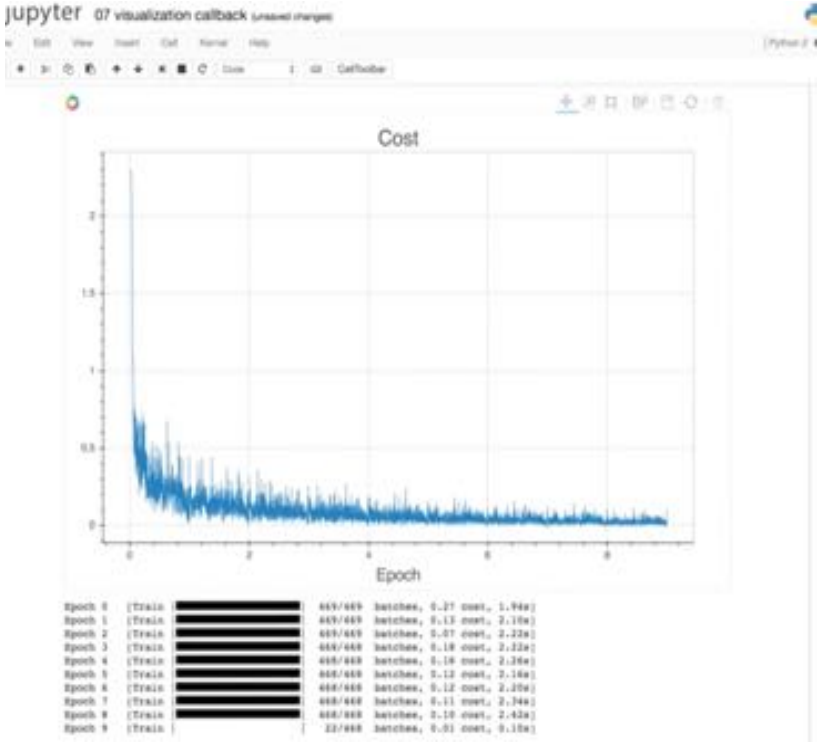
Web UI

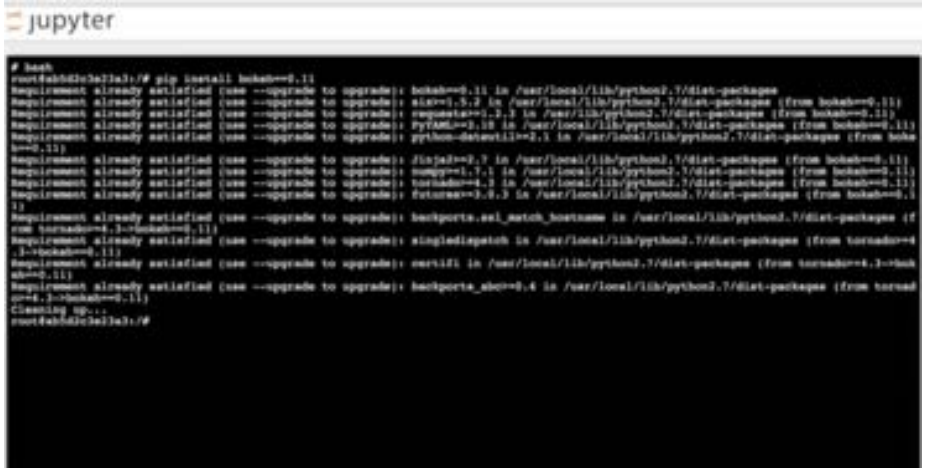
https





Nervana Cloud: Interactive Mode

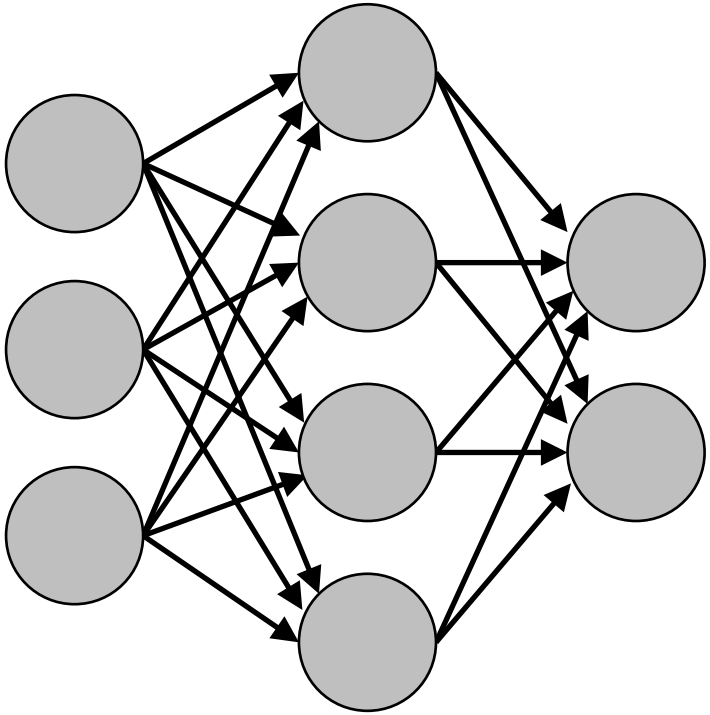




- Interactive Python environment within container
- Access Nervana Cloud GPUs & software
- Attach data sets and code repository
- Extend neon with custom callbacks (e.g., BokehJS)
- Get root access to virtual environment
- Install packages, change system settings

Outline

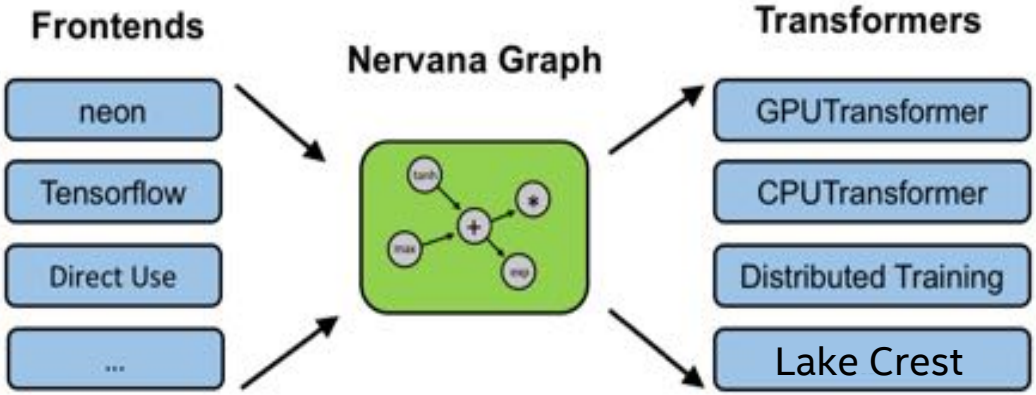
- Intel® NervanaTM Overview
- Intel's Deep Learning Hardware
- **Intel's Deep Learning Software**
  - MKL-DNN
  - Neon and Nervana Cloud
  - **Nervana Graph**



The Nervana Graph Project

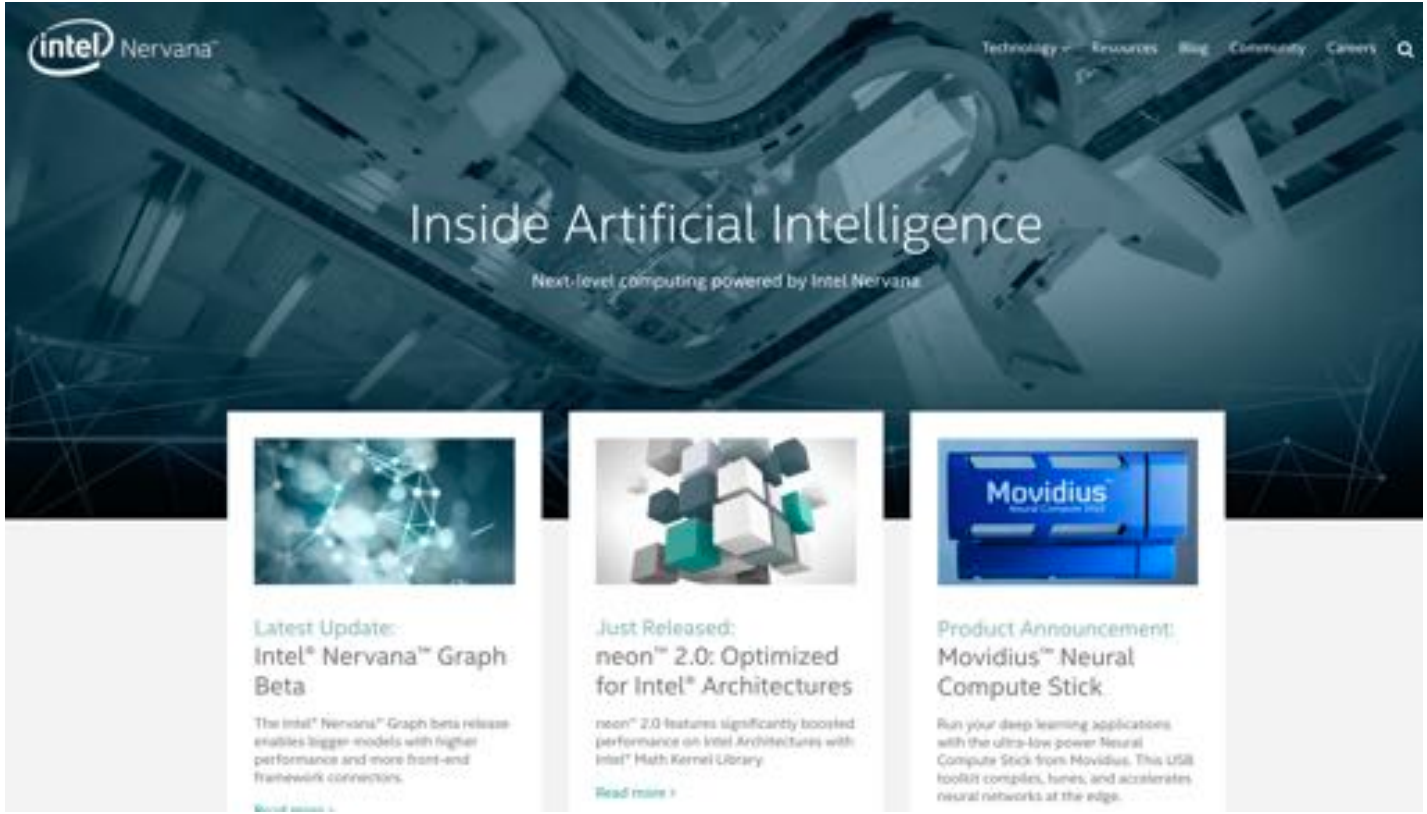
Tech preview available

- An intermediate representation (IR) for deep learning
- A compiler for Nervana graph IR
- Frontends for other frameworks to Nervana graph
- Many-to-many problem
- A reference frontend (neon)



https://github.com/NervanaSystems/ngraph

Learn More… intelnervana.com



intel®

experience
what's inside™