Scaling Deep Learning without Impacting Batchsize

CUG 2018

Alexander Heye, Cray Inc.

Agenda

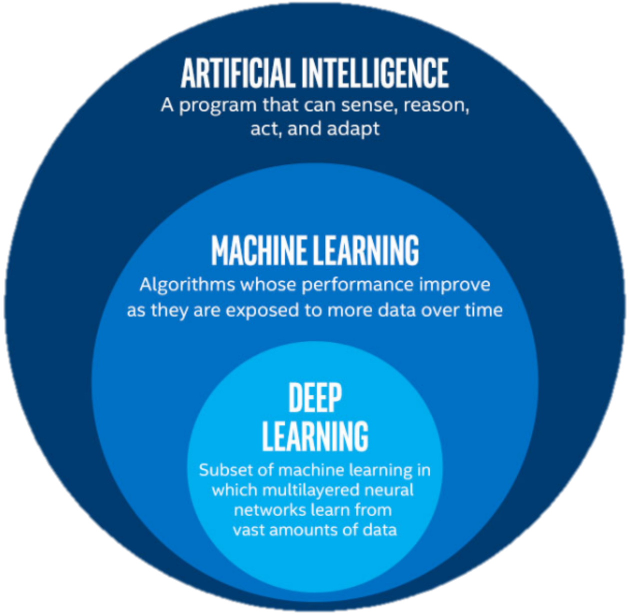
- Purpose
- Introduction
- Distributed training
- Distributed workflow
- Combined scalability
- Evaluation
- Results
- Summary
- Q&A

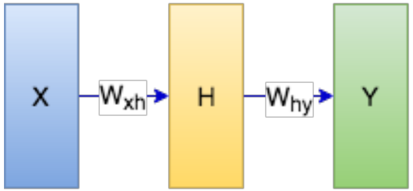
Purpose

- **Survey current scaling techniques for deep learning**
- **Discuss drawbacks to these techniques**
  - Especially when scaled up to very large systems
- **Bring attention to workflow orientated opportunities to develop at scale**
  - View any workflow optimization as an opportunity to distribute the workload
- **Provide insight into how these can be applied**
  - Independently
  - When Combined

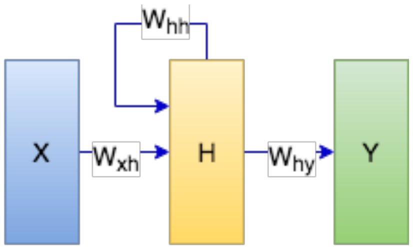
Introduction – Deep Learning

- Deep learning vs neural networks vs machine learning
- Stochastic gradient descent
- Stochastic, mini-batch and batch training
- Convolutional, recurrent and feed-forward neural networks
- Genetic learning algorithms





Feed Forward Neural Network



Recurrent Neural Network

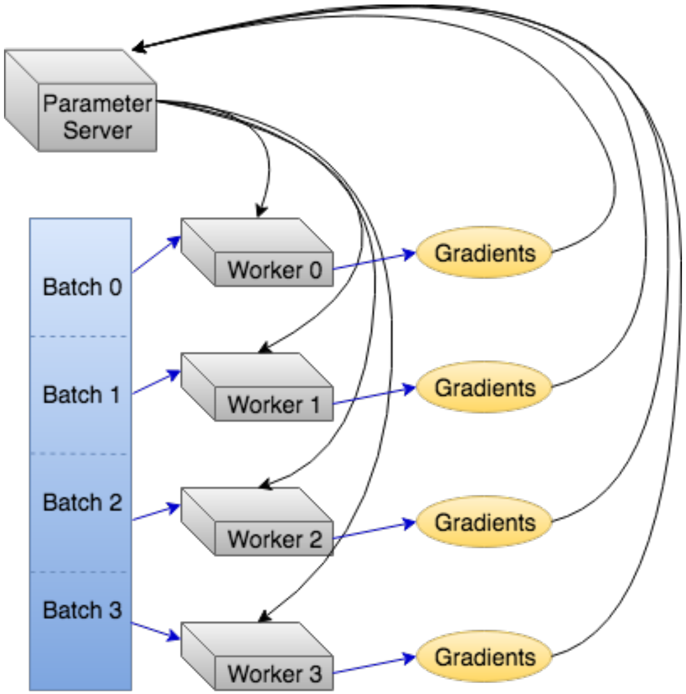
Introduction – Distributed Deep Learning

- **Distributed training**
  - Metrics: Time to accuracy, throughput
  - Data parallelism, model parallelism
- **Distributed workflow**
  - Metrics: time to tuned model
  - Hyperparameter optimization
  - Transfer learning
  - Ensemble networks

Distributed Training

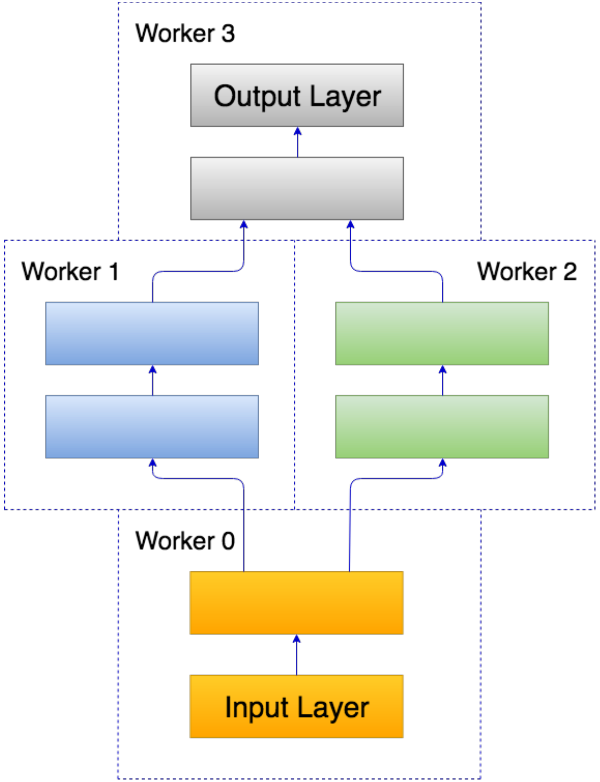
Distributed Training – Data Parallelism

- **Method**
  - Divide training by dividing the data.
- **Parameter layout**
  - Replicated on each worker, master on parameter server.
- **Pros**
  - Efficiently scales throughput
  - No requirements on model design
- **Cons**
  - Leads to very large global batchsize
  - Special attention must be paid to training hyperparameters



Distributed Training – Model Parallelism

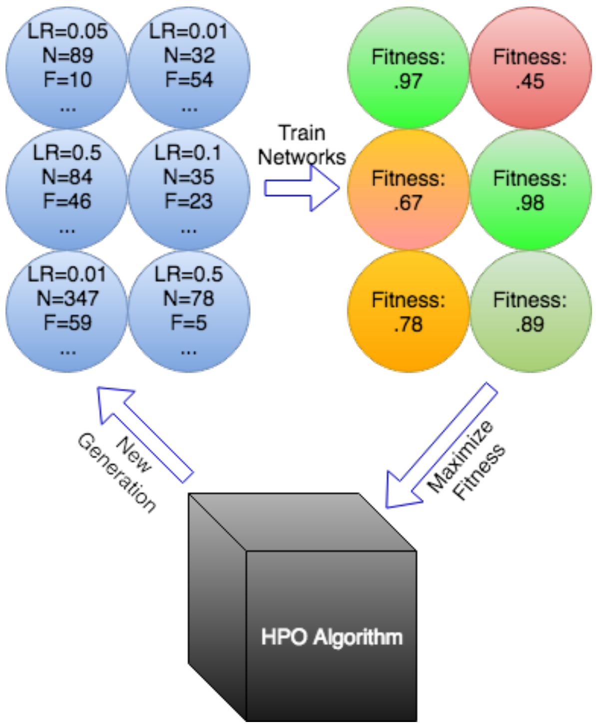
- **Method**
  - Divide model and distribute parameter and execution among workers
- **Parameter layout**
  - Each worker has a fraction of the model parameter
- **Pros**
  - Less parameter replication
  - Spread memory load among workers
- **Cons**
  - Some sequential processing necessary
  - Very model dependent



Distributed Workflow

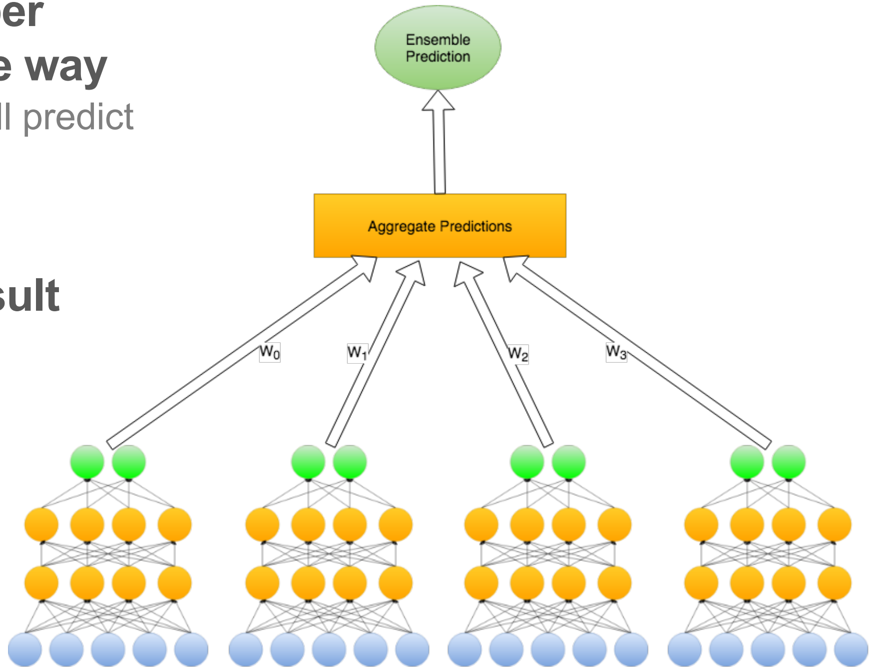
Distributed Workflow – HPO

- **Hyperparameters**
  - Define network design
    - Number of layers, activation functions, etc.
  - Define training process
    - Learning rate(s), dropout rate, etc.
- **Optimization**
  - Sweeps
    - Grid or random
  - Guided
    - Bayesian, genetic, etc.



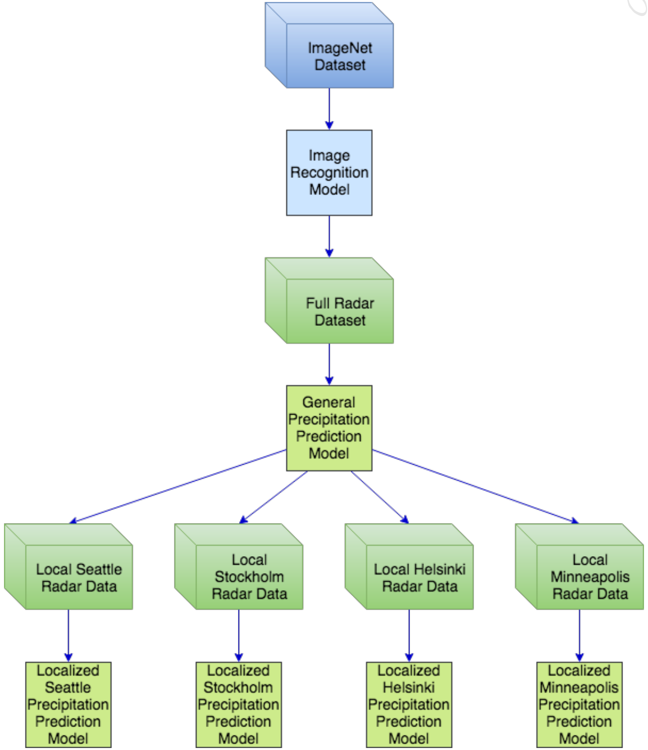
Distributed Workflow – Ensemble Networks

- **Many neural network models can lead to better results than any individual member**
- **Each network is allowed in vary in some way**
  - Input data (temperature, pressure, humidity all predict precipitation rate)
  - Structure (hyperparameters)
  - Initializations
- **Aggregate and weight each member result**
  - Aid in feature selection
  - More interpretable results
- **Individual member can be trained independently in parallel**
  - Can follow directly from HPO



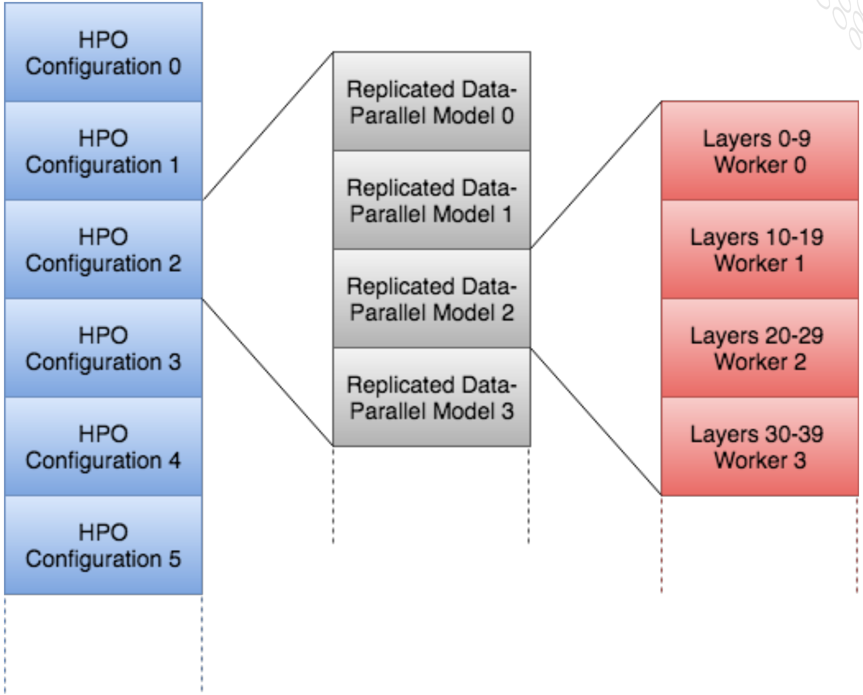
Distributed Workflow – Transfer Learning

- **Definition:**
  - Applying learning from a separate but similar domain to a new problem
- **Standard application:**
  - Ex: Use Imagenet trained network to prime the training of a new set of input/output
- **Distributed application:**
  - Hierarchy of training data
  - Split problem into sub-problems, train sub-models in parallel



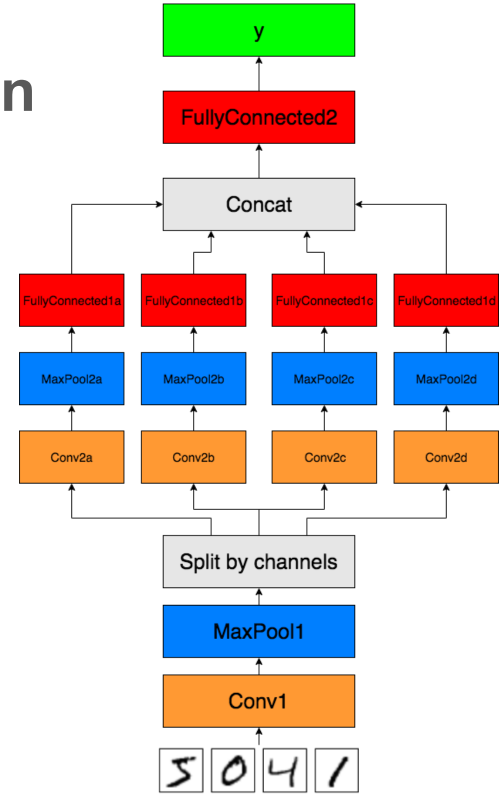
Combined Scalability

- **Limitations to large scale parallelism**
  - Model constraints
  - Training efficiency
- **Combining techniques allows simplified approach to full utilization and sufficient scaling efficiency**
- **Example:**
  - Data parallelism to 16 nodes model parallelism to 8 nodes
  - HPO with generation size of 16
  - Total node count = 2,048
- **Benefits if properly implemented**
  - Global minibatchsize remains manageable
  - Optimized model configuration with little manual input
  - Memory distribution allowing more parameters



Evaluation

- **Simple dataset—MNIST digit recognition**
- **Specialized CNN model**
  - Horizontal model parallelism possible
  - Largest parameter count in split layers
- **Single model training evaluation**
- **HPO evaluation**
- **Combined scalability**
- **System details**
  - Cray XC30
  - Cray Urika-XC analytics platform



Results – Single Model Training

- **Limited in scope**
  - Goal to gain a baseline understanding
- **Model parallel**
  - Partial parallel processing
  - Tensorflow gRPC
- **Data parallel**
  - 4 nodes for consistency
  - MPI communication (no PS)

Method	Time (s)	Nodes	Improve-ment
Baseline	1090	1	1x
Model Parallel	718	4	1.5x
Data Parallel	310	4	3.5x

Results – Hyperparameter Optimization

- **Genetic algorithm**
- **Track runs and generations to threshold**
  - 98.6% on validation set
- **Max accuracy at convergence**
  - 10 generations without improvement
- **Baseline: random search**
  - 5000 random configurations
  - 0.1% reached threshold
  - Max accuracy: 98.63%
- **Speedup**
  - Decrease in total training runs
  - Decrease in seconds/run

Nodes	Gen.	Runs	Time	Speedup	Acc.
1	50	90	13496	6.3x	98.60%
2	34	101	11258	7.5x	98.67%
4	31	207	10312	8.2x	98.69%
8	22	272	7860	10.7x	98.66%
16	16	443	5529	15.3x	98.69%
32	14	761	4961	17.0x	98.90%
64	11	1187	3855	21.9x	99.15%

Results – Combined Scalability

- **For illustration**
  - Improvement calculated by multiplying individual improvements over baseline
  - Actual tests will needed to verify (future work)
- **HPO improvement takes into account**
  - Decrease in runs to threshold (1.8x)
  - Scaling efficiency to 8 nodes (5.8x)
- **At 128 nodes, global batchsize only increases 4x**

Method	Runs	Nodes	BS	Speedup
Random Search	500	1	100	1x
MP and DP	500	8	400	5x
HPO	272	8	100	11x
HPO and MP	272	32	100	16x
HPO and DP	272	32	400	38x
HPO, DP and MP	272	128	400	56x

Summary

- **Distributed training vs. distributed workflow**
  - Model and data parallelism
  - Hyperparameter optimization, ensemble networks, transfer learning
- **Combined scalability**
  - Keep global batchsize within reasonable range
  - Fewer total distributed training runs to optimal configuration
- **Future work**
  - Complete evaluation of combine scaling
  - “Distributed Toolkit”

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

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

Cray hardware and software products 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.

Cray uses codenames internally to identify products that are in development and not yet publicly announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: CHAPEL, CLUSTER CONNECT, CLUSTERSTOR, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE, REVEAL. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used on this website are the property of their respective owners.

Q&A

Alexander Heye
aheye@cray.com