Agenda

- What We Do
  - History
  - Going forward
- How We Scale
  - CNTK
  - FPGA
  - Open Mind
- Q&A
What We Do
ML @ Microsoft: History
Answering questions with experience

1991
Microsoft Research formed

1997
Hotmail launches
Which email is junk?

2008
Bing maps launches
What’s the best way home?

2009
Bing search launches
Which URLs are most relevant?

2010
Kinect launches
What does that motion “mean”?

2014
Skype Translator launches
What is that person saying?

2015
Azure Machine Learning GA
Office 365 Substrate
HoloLens
What will happen next?

Machine learning is pervasive throughout Microsoft products
ML @ Microsoft: Going Forward

- Data => Model => Intelligence => Fuels of Innovation
- Applications & Services
  - Office 365, Dynamic 365 (Biz SaaS), Skype, Bing, Cortana
  - Digital Work & Digital Life
  - Models for: World, Organizations, Users, Languages, Context, ...
- Computing Devices
  - PC, Tablet, Phone, Wearable, Xbox, Hololens (AR/VR), ....
  - Models for: Natural User Interactions, Reality, ...
- Cloud
  - Azure Infrastructure and Platform
  - Azure ML Tools & Services
  - Intelligence Services
# Machine Learning Building Blocks

<table>
<thead>
<tr>
<th>Azure ML (Cloud)</th>
<th>Microsoft R Server (On-Prem &amp; Cloud)</th>
<th>Computational Network Toolkit</th>
<th>Cognitive APIs (Cloud Services)</th>
<th>HDInsight/Spark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of use through Visual Workflows</td>
<td>Enterprise Scale &amp; Performance</td>
<td>Designed for peak performance</td>
<td>See, hear, interpret, and interact</td>
<td>Open source Hadoop with Spark</td>
</tr>
<tr>
<td>Single click operationalization</td>
<td>Write Once, Deploy Anywhere</td>
<td>Works on CPU and GPU (single/multi)</td>
<td>Prebuilt APIs with CNTK and experts</td>
<td>Use Spark ML or MLLib using Java, Python, Scala or R</td>
</tr>
<tr>
<td>Integration with Jupyter Notebook</td>
<td>Secure/Scalable Operationalization</td>
<td>Highly Flexible – description language</td>
<td>Interact with your users on SMS, text, email, Slack, Skype</td>
<td>Includes MRS over Hadoop or over Spark</td>
</tr>
<tr>
<td>Integration with R/Python</td>
<td>Works with open source R</td>
<td>Used to build cognitive APIs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Run large massively parallel compute and data jobs</td>
</tr>
</tbody>
</table>
Azure Machine Learning Services

- Ease of use tools with **drag/drop paradigm**, single click **operationalization**
- Built-in support for **statistical functions**, data **ingest, transform, feature generate/select, train, score, evaluate** for tabular data and text across **classification, clustering, recommendation, anomaly**
- Seamless **R/Python** integration along with support for **SQL lite** to filter, transform
- **Jupyter** Notebooks for data exploration and **Gallery** extensions for quick starts
- Modules for **text preprocessing**, key phrase extraction, language detection, n-gram generation, LDA, compressed feature hash, stats based anomaly
- **Spark/HDInsight/MRS** Integration
- **GPU** support
- New geographies
- Compute **reservation**
Intelligence Suite

Data Sources
- Data Factory
- Data Catalog
- Event Hubs

Information Management
- Data Factory
- Data Catalog
- Event Hubs

Big Data Stores
- Data Lake Store
- SQL Data Warehouse

Machine Learning and Analytics
- Machine Learning
- Data Lake Analytics
- HDInsight (Hadoop and Spark)
- Stream Analytics

Intelligence
- Cognitive Services
- Bot Framework
- Cortana

Dashboards & Visualizations
- Power BI

Data Sources → Information Management → Big Data Stores → Machine Learning and Analytics → Intelligence → Dashboards & Visualizations → Action

Data
- Apps
- Sensors and devices

Intelligence
- Data Sources
- Information Management
- Big Data Stores
- Machine Learning and Analytics
- Intelligence
- Dashboards & Visualizations

Action
- Web
- Mobile
- Bots

Automated Systems
## Cognitive Services

<table>
<thead>
<tr>
<th>Vision</th>
<th>Speech</th>
<th>Language</th>
<th>Knowledge</th>
<th>Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer vision</td>
<td>Speaker recognition</td>
<td>Text analytics</td>
<td>Academic knowledge</td>
<td>Bing search API</td>
</tr>
<tr>
<td>Face</td>
<td>Speech</td>
<td>Bing spell check</td>
<td>Entity linking service</td>
<td>Bing image search API</td>
</tr>
<tr>
<td>Emotion</td>
<td>Custom recognition</td>
<td>Web language model</td>
<td>Knowledge exploration service</td>
<td>Bing video search API</td>
</tr>
<tr>
<td>Video</td>
<td></td>
<td>Linguistic analysis</td>
<td>Recommendations</td>
<td>Bing news search API</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Language understanding</td>
<td></td>
<td>Bing auto suggest API</td>
</tr>
</tbody>
</table>
How We Scale
Key Dimensions of Scaling

- Data volume / dimension
- Model / algorithm complexity
- Training / evaluation time
- Deployment / update velocity
- Developer productivity / innovation agility
- Infrastructure / platform
- Software framework / tool
- Data set / algorithm
How We Scale Example: CNTK
CNTK: Computational Network Toolkit

- CNTK is Microsoft’s open-source, cross-platform toolkit for learning and evaluating models especially deep neural networks
- CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting common network types and applications
- CNTK is production-deployed: accuracy, efficiency, and scales to multi-GPU/multi-server
CNTK Development

- Open-source development model inside and outside the company
  - Created by Microsoft Speech researchers 4 years ago; open-sourced in early 2015
  - On GitHub since Jan 2016 under permissive license
  - Nearly all development is out in the open

- Driving applications: Speech, Bing, Hololens, MSR research
  - Each team have full-time employees actively contribute to CNTK
  - CNTK trained models are tested and deployed in production environment

- External contributions
  - e.g., from MIT and Stanford

- Platforms and runtimes
  - Linux, Windows, .Net, docker, cudnn5
  - Python, C++, and C# APIs coming soon
CNTL Design Goals & Approach

- A deep learning framework that balances
  - **Efficiency**: can train production systems as fast as possible
  - **Performance**: can achieve best-in-class performance on benchmark tasks for production systems
  - **Flexibility**: can support a growing and wide variety of tasks such as speech, vision, and text; can try out new ideas very quickly

- Lego-like composability
  - Support a wide range of networks
  - E.g. Feed-forward DNN, RNN, CNN, LSTM, DSSM, sequence-to-sequence

- Evolve and adapt
  - Design for emerging prevailing patterns
Key Functionalities & Capabilities

- **Supports**
  - CPU and GPU with a focus on GPU Cluster
  - Automatic numerical differentiation
  - Efficient static and recurrent network training through batching
  - Data parallelization within and across machines, e.g., 1-bit quantized SGD
  - Memory sharing during execution planning

- **Modularization with separation of**
  - Computational networks
  - Execution engine
  - Learning algorithms
  - Model description
  - Data readers

- **Model descriptions via**
  - Network definition language (NDL) and model editing language (MEL)
  - Brain Script (beta) with Easy-to-Understand Syntax
Architecture

- **CN Description**
  - Use

- **ICNBuilder**
  - Build

- **CPU/GPU**
  - IExecutionEngine

- **CN**
  - Evaluate
  - Compute Gradient

- **Features & Labels**
  - Load
  -IDataReader
  - Get data

- **ILearner**

- **Task-specific reader**

- **SGD, AdaGrad, etc.**
Roadmap

- CNTK as a library
  - More language support: Python/C++/C#/.Net
- More expressiveness
  - Nested loops, sparse support
- Finer control of learner
  - SGD with non-standard loops, e.g., RL
- Larger model
  - Model parallelism, memory swapping, 16-bit floats
- More powerful CNTK service on Azure
  - GPUs soon; longer term with cluster, container, new HW (e.g., FPGA)
How We Scale Example: FPGA
Catapult v2 Architecture

- Gives substantial acceleration flexibility
  - Can act as a local compute accelerator
  - Can act as a network/storage accelerator
  - Can act as a remote compute accelerator
Cloud becomes network + FPGAs attached to servers

Can continuously upgrade/change datacenter HW protocols (network, storage, security)

Can also use as an application acceleration plane (Hardware Acceleration as a Service (HaaS))

Services communicate with no SW intervention (LTL)

Single workloads (including deep learning) can grab 10s, 100s, or 1000s of FPGAs

Can create service pools as well for high throughput
Scalable Deep Learning on FPGAs

- **Scale ML Engine**: a flexible DNN accelerator on FPGA
  - Fully programmable via software and customizable ISA
  - Over 10X improvement in energy efficiency, cost, and latency versus CPU

- Deployable as large-scale DNN service pools via HaaS
  - Low latency communication in few microseconds / hop
  - Large scale models at ultra low latencies
How We Scale Example: Open Mind
Open Mind Studio: the “Visual Studio” for Machine Learning
Data, Model, Algorithm, Pipeline, Experiment, and Life Cycle Management

Federated Infrastructure
Data Storage, Compliance, Resource Management, Scheduling, and Deployment

Programming Abstractions for Machine Learning / Deep Learning

- CNTK
- Other Deep Learning Frameworks (e.g., Caffe, MxNet, TensorFlow, Theano, Torch)
- Open Source Computation Frameworks (e.g., Hadoop, Spark)
- Specialized, Optimized Computation Frameworks (e.g., SCOPE, ChaNa)
- The Next New Framework ...

Heterogeneous Computing Platform
(CPU, GPU, FPGA, RDMA; Cloud, Client/Device)
ChaNa: RDMA-Optimized Computation Framework

- Focus on faster network
  - Compact memory representation
  - Balanced parallelism
  - Highly optimized RDMA-aware communication primitives
  - Overlapping communication and computation
- An order of magnitude improvement in early results
  - Over existing computation frameworks (with TCP)
  - Against several large-scale workloads in production
Programming Abstraction for Machine Learning

- Graph Engines for Distributed Machine Learning
  - Automatic system-level optimizations
  - Parallelization and distribution
  - Layout for efficient data access
  - Partitioning for balanced parallelism

- Promising early results
  - Simplification of distributed ML programs via high level abstractions
  - About 70-80% reduction in code
    - Relative to ML systems such as Petuum, Parameter Server
    - Matrix Factorization for recommendation system
    - Latent Dirichlet Allocation for topic modeling
Q&A
Thank You!