Welcome 2017 Faculty Summit Attendees

Faculty Summit 2017
microsoftfacultysummit.com

Microsoft Research
Microsoft.com/research

Facebook
@microsoftresearch

Twitter
@MSFTResearch
#FacSumm
#EdgeofAI
Fast and Scalable Deep Learning with Microsoft Cognitive Toolkit (CNTK)

Sayan Pathak, Principal ML Scientist
Cha Zhang, Principal Researcher/Dev Manager
Microsoft AI & Research

With 130+ contributors.
Outline

• Overview
• CNTK introduction
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Educational resources
• Conclusions
Outline

• Overview
• CNTK introduction
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Educational resources
• Conclusions
Deep Learning at Microsoft

• Microsoft Cognitive Services
• Skype Translator
• Cortana
• Bing
• HoloLens
• Microsoft Research
Historic Achievement: Microsoft researchers reach human parity in conversational speech recognition

Microsoft has made a major breakthrough in speech recognition, creating a technology that recognizes the words in a conversation as well as a person does.
Microsoft had all 5 entries being the 1-st places in 2015: ImageNet classification, ImageNet localization, ImageNet detection, COCO detection, and COCO segmentation.
Microsoft Cognitive Services

Check out the other Cognitive Services APIs

**Vision**
State-of-the-art image processing algorithms help you moderate content automatically and build more personalized apps by returning smart insights about faces, images, and emotions.

- Computer Vision API
- Emotion API
- Video API
- Video Indexer

**Speech**
Processing spoken language in your applications

- Custom Speech Service
- Translator Speech API
- Bing Spelling Check API

**Language**
Allow your apps to process natural language, evaluate sentiment and topics, and learn how to recognize what users want.

- Language Understanding Intelligent Service
- Bing Spell Check API
- Text Analytics API
- Web Language Model API

**Knowledge**
Map complex information and data in order to solve tasks such as intelligent recommendations and semantic search.

- Recommendations API
- Knowledge Exploration Service
- QnA Maker API
- Entity Linking Intelligence Service API
- Custom Decision Service

**Search**
Make your apps, webpages, and other experiences smarter and more engaging with the Bing Search APIs.

- Bing Autosuggest API
- Bing Video Search API
- Bing Entity Search API
- Bing Image Search API
- Bing Web Search API
- Bing News Search API
- Bing Custom Search
Shanghai Changzheng Hospital
Goal: given query image, find similar images.

- Customer: Anonymous ISV (Azure Partner)
- Task: given a retail image, find same product on competitor websites (to compare price).
- Existing solution: solely based on mining text information from the websites of Target, Macy, etc.
- Customer asked for individual similarity measure (e.g. texture, neck style, etc.).
Bing / Bing Ads

Most sold fruit in US

Bananas

The most popular fresh fruits in the United States are (in order): Bananas, apples, oranges, grapes and strawberries. In 2012, U.S. production of the leading noncitrus fruit crops totaled 17.4 million tons, down 4 percent from the previous year.

Fruits | Agricultural Marketing Resource Center
www.agnrc.org/commodities__products/fruits/

shops for bugs bunny books

Your Complete Bugs Bunny: $29.95
Buzz Bunny Calling A Thriftbooks.99
Barnes & Noble: $3.79 Special Offer
Bugs Bunny And The Thriftbooks.99
Bug Bunnies: $3.59 Special Offer
Bugs Bunny & Little Thriftbooks.99
Little Big Book: $7.99 eBay

Microsoft Translator

Use the power of Artificial Intelligence for better subtitles

Microsoft Translator learns from the content on your slides to give you better subtitles for your subject matter

Power point-plug in for translating speech to subtitles

http://translate.it

Microsoft Translator for Business
Microsoft Customer Support Agent
I’m Microsoft’s new virtual support agent. Describe your problem and I’ll look for the best solution. You can also ask to talk to a person at any time.

Hi.

Glad to help. Could you describe your problem in detail?

I’m having trouble setting up a new projector for my laptop. It’s an Epson VS240.
Suggestion

Connect to a projector

To connect to a projector

1. Make sure the projector is turned on, and then plug the projector cable into a video port on your computer. Note: Projectors use VGA or DVI cables. You must plug the cable into a matching video port on your computer. Although some computers have both types of video ports, most laptops just have one type. Some projectors can be connected to a USB port on your computer with a USB cable. VGA and DVI ports.

2. Open Control Panel by clicking the Start button, and then clicking Control Panel.

3. In the search box, type projector, and then click Connect to a projector. (To use a keyboard shortcut instead of Control Panel, press the Windows logo key + P.)

4. Select how you want your desktop to be displayed: Computer only, Second monitor, or Duplicate.
4. Select how you want your desktop to be displayed: Computer only (this shows your desktop only on your computer screen.) Duplicate (this shows your desktop on both your computer screen and a projector.) Extend (this extends your desktop from your computer screen to a projector.) Projector only (this shows your desktop only on a projector)

Hope that helped. If not please rephrase your problem. You can also ask to talk to a live agent anytime.
Hello!

This didn’t fix it. I got an error that my screen resolution is too high for the projector.

Suggestion

For this problem, I found a web page for you. Please check the preview below.

*Restore Screen resolution to default (Projector second screen settings)*

Hi; I was using my laptop with my projector and I happened to change the screen resolution when... the screen was projected to the second screen only (now the projector won’t accept those settings and...)*
Hello!

If this wasn’t helpful, please let me know more details about your problem, or ask to talk to a person.

Talk to a person

Talk to a person

Archie B has joined the chat
Microsoft answer Tech

Thanks for contacting Microsoft support, my name is Archie B.
Microsoft Cognitive Toolkit (CNTK)

- Microsoft’s open-source deep-learning toolkit
  - [https://github.com/Microsoft/CNTK](https://github.com/Microsoft/CNTK)
  - Created by Microsoft Speech researchers (Dong Yu et al.) in 2012, “Computational Network Toolkit”
  - On GitHub since Jan 2016 under MIT license
  - Python support since Oct 2016 (beta), rebranded as “Cognitive Toolkit”
  - External contributions e.g. from MIT, Stanford and Nvidia

- Over 80% Microsoft internal DL workload runs CNTK
- 1st-class on Linux and Windows, docker support
- Python, C++, C#, Java, Keras
- Internal == External
## CTNK Speed

http://dlbench.comp.hkbu.edu.hk/
Benchmarking by HKBU, Version 8
Single Tesla K80 GPU, CUDA: 8.0 CUDNN: v5.1

<table>
<thead>
<tr>
<th>Model</th>
<th>Caffe</th>
<th>CNTK</th>
<th>MxNet</th>
<th>TensorFlow</th>
<th>Torch</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN5 (1024)</td>
<td>55.329ms</td>
<td><strong>51.038ms</strong></td>
<td>60.448ms</td>
<td>62.044ms</td>
<td>52.154ms</td>
</tr>
<tr>
<td>AlexNet (256)</td>
<td>36.815ms</td>
<td><strong>27.215ms</strong></td>
<td>28.994ms</td>
<td>103.960ms</td>
<td>37.462ms</td>
</tr>
<tr>
<td>ResNet (32)</td>
<td>143.987ms</td>
<td><strong>81.470ms</strong></td>
<td>84.545ms</td>
<td>181.404ms</td>
<td>90.935ms</td>
</tr>
<tr>
<td>LSTM (256)</td>
<td>-</td>
<td><strong>43.581ms</strong> (44.917ms)</td>
<td>288.142ms (284.898ms)</td>
<td>- (223.547ms)</td>
<td>1130.606ms (906.958ms)</td>
</tr>
</tbody>
</table>

Caffe: 1.0rc5(39f28e4)
CNTK: 2.0 Beta10(1ae666d)
MXNet: 0.93(32dc3a2)
TensorFlow: 1.0(4ac9c09)
Torch: 7(748f5e3)
“CNTK is production-ready: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.”

speed comparison (samples/second), higher = better

[note: December 2015]
**MICROSOFT COGNITIVE TOOLKIT**

First Deep Learning Framework Fully Optimized for Pascal

---

**Toolkit Delivering Near-Linear Multi-GPU Scaling**

AlexNet Performance

- **Dual Socket CPU Server**: 78 images/sec
- **1x P100**: 2,400 images/sec
- **2x P100**: 3,500 images/sec
- **4x P100**: 7,600 images/sec
- **DGX-1 (8x P100)**: 13,000 images/sec

AlexNet training batch size 128, Grad Bit = 32, Dual socket E5-2699v4 CPUs (total 44 cores)

CNTK 2.0b3 (to be released) includes cuDNN 5.1.8, NCCL 1.6.1, NVLink enabled

170x Faster v. CPU Server
Superior performance

Multi-Node Training with NCCL 2.0 (ResNet-50)

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>8x P100</td>
<td>8</td>
</tr>
<tr>
<td>8x V100</td>
<td>16</td>
</tr>
<tr>
<td>16x V100</td>
<td>24</td>
</tr>
<tr>
<td>32x V100</td>
<td></td>
</tr>
<tr>
<td>64x V100</td>
<td></td>
</tr>
</tbody>
</table>

GPU-Accelerated Microsoft Cognitive Toolkit Now Available in the Cloud on Microsoft Azure and On-Premises with NVIDIA DGX-1

SC16 — To help companies join the AI revolution, NVIDIA today announced a collaboration with Microsoft to accelerate AI in the enterprise.

Using the first purpose-built enterprise AI framework optimized to run on NVIDIA DGX-1 GPUs in Microsoft Azure or on-premises, enterprises now have an AI platform that spans from their data center to Microsoft’s cloud.

“Every industry has awoken to the potential of AI,” said Jen-Hsun Huang, founder and chief executive officer, NVIDIA. “We’ve worked with Microsoft to create a lightning-fast AI platform that is available from on-premises with our DGX-1 supercomputer to the Microsoft Azure cloud. With Microsoft’s global reach, every company around the world can now tap the power of AI to transform their business.”

“We’re working hard to empower every organization with AI, so that they can make smarter products and solve some of the world’s most pressing problems,” said Harry Shum, executive vice president of the Artificial Intelligence and Research Group at Microsoft. “By working closely with NVIDIA and harnessing the power of GPU-accelerated systems, we’ve made Cognitive Toolkit and Microsoft Azure the fastest, most versatile AI platform. AI is now within reach of any business.”

This jointly optimized platform runs the new Microsoft Cognitive Toolkit (formerly CNTK) on NVIDIA GPUs, including the NVIDIA DGX-1 supercomputer, which uses Pascal GPU architecture with NVLink interconnect technology, and on Azure H-Series virtual machines, currently in preview. This combination delivers unprecedented performance and ease of use when using data for deep learning.

As a result, companies can harness AI to make better decisions, offer new products and services faster and provide better customer experiences. This is causing every industry to implement AI. In just two years, the number of companies NVIDIA collaborates with on deep learning has jumped 19x to over 19,000. Industries such as healthcare, life sciences, energy, financial services, automotive and manufacturing are benefiting from deeper insight on extreme amounts of data.
A team of researchers from Microsoft, Cray, and the Swiss National Supercomputing Centre (CSCS) have been working on a project to speed up the use of deep learning algorithms on supercomputers.

The team have scaled the Microsoft Cognitive Toolkit -- an open-source suite that trains deep learning algorithms -- to more than 1,000 Nvidia Tesla P100 GPU accelerators on the Swiss centre's Cray XC50 supercomputer, which is nicknamed Piz Daint.
CNTK Other Advantages

• Python and C++ API
  • Mostly implemented in C++
  • Low level + high level Python API
• Extensibility
  • User functions and learners in pure Python
• Readers
  • Distributed, highly efficient built-in data readers
• Internal == External
CNTK Other Advantages

• Keras interoperability
  • With CNTK backend
  • Try models trained with TF/Theano with CNTK (vice versa)
  • Demo RL with DQN

• Binary evaluation
  • 10x speed-up in model execution
Reinforcement Learning with Flappy Bird

Steps:
- Receive the Game screen input (as image array)
- Pre-process the image
- Process the image using a CNN
- Predict action: Flap vs. no Flap
- Use Q-learning to maximize the future reward

Code
- https://github.com/yanpanlau/Keras-FlappyBird
Binary evaluation

Conv-Net

Binary Conv-Net
Outline

• Overview
  • CNTK introduction
    • Symbolic loop
    • Batch scheduling
    • Data parallel training
  • Educational resources
  • Conclusions
What is CNTK?

• CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications.
What is CNTK?

Example: 2-hidden layer feed-forward NN

\[ h_1 = \sigma(W_1 x + b_1) \]
\[ h_2 = \sigma(W_2 h_1 + b_2) \]
\[ P = \text{softmax}(W_{out} h_2 + b_{out}) \]

with input \( x \in \mathbb{R}^M \)
What is CNTK?

Example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$
$$h_2 = \sigma(W_2 h_1 + b_2)$$
$$P = \text{softmax}(W_{out} h_2 + b_{out})$$

with input $$x \in \mathbb{R}^M$$ and one-hot label $$y \in \mathbb{R}^J$$

and cross-entropy training criterion

$$ce = y^T \log P$$
$$\sum_{\text{corpus}} ce = \max$$
What is CNTK?

example: 2-hidden layer feed-forward NN

\[ h_1 = \sigma(W_1 x + b_1) \]
\[ h_2 = \sigma(W_2 h_1 + b_2) \]
\[ P = \text{softmax}(W_{out} h_2 + b_{out}) \]

with input \( x \in \mathbb{R}^M \) and one-hot label \( y \in \mathbb{R}^J \)

and cross-entropy training criterion

\[ ce = y^T \log P \]
\[ \sum_{\text{corpus}} ce = \max \]

\[ h_1 = \text{sigmoid} (x \ @ \ W_1 + b_1) \]
\[ h_2 = \text{sigmoid} (h_1 @ W_2 + b_2) \]
\[ P = \text{softmax} (h_2 @ W_{out} + b_{out}) \]

\[ ce = \text{cross}_{-}\text{entropy} (P, y) \]
What is CNTK?

\[
\begin{align*}
    h1 &= \text{sigmoid} \left( x \ @ \ W1 + b1 \right) \\
    h2 &= \text{sigmoid} \left( h1 \ @ \ W2 + b2 \right) \\
    P &= \text{softmax} \left( h2 \ @ \ Wout + bout \right) \\
    \text{ce} &= \text{cross_entropy} \left( P, y \right)
\end{align*}
\]
What is CNTK?

h1 = sigmoid (x @ W1 + b1)
h2 = sigmoid (h1 @ W2 + b2)
P = softmax (h2 @ Wout + bout)
ce = cross_entropy (P, y)
What is CNTK?

• Nodes: functions (primitives)
  • Can be composed into reusable composites

• Edges: values
  • Incl. tensors, sparse

• Automatic differentiation
  • $\frac{\partial F}{\partial \text{in}} = \frac{\partial F}{\partial \text{out}} \cdot \frac{\partial \text{out}}{\partial \text{in}}$

• Deferred computation $\rightarrow$ execution engine

• Editable, clonable

LEGÖ-like composability allows CNTK to support wide range of networks & applications
CNTK Unique Features

• Symbolic loops over sequences with dynamic scheduling
• Turn graph into parallel program through minibatching
• Unique parallel training algorithms (1-bit SGD, Block Momentum)
Symbolic Loops over Sequential Data

Extend our example to a recurrent network (RNN)

\[
h_1 = \sigma(W_1 x + b_1)
\]

\[
h_2 = \sigma(W_2 h_1 + b_2)
\]

\[
P = \text{softmax}(W_{\text{out}} h_2 + b_{\text{out}})
\]

\[ce = \sum_{\text{corpus}} \text{cross}\_\text{entropy}(P, L)\]

\[
\sum_{\text{corpus}} ce = \max
\]
Symbolic Loops over Sequential Data

Extend our example to a recurrent network (RNN)

\[
\begin{align*}
    h_1(t) &= \sigma(W_1 x(t) + R_1 h_1(t-1) + b_1) \\
    h_2(t) &= \sigma(W_2 h_1(t) + R_2 h_2(t-1) + b_2) \\
    P(t) &= \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}}) \\
    \sum_{\text{corpus}} ce(t) &= \text{max}
\end{align*}
\]

\[
\begin{align*}
    h_1 &= \text{sigmoid}(x @ W_1 + \text{past_value}(h_1) @ R_1 + b_1) \\
    h_2 &= \text{sigmoid}(h_1 @ W_2 + \text{past_value}(h_2) @ R_2 + b_2) \\
    P &= \text{softmax}(h_2 @ W_{\text{out}} + b_{\text{out}}) \\
    \text{ce} &= \text{cross_entropy}(P, L)
\end{align*}
\]
Symbolic Loops over Sequential Data

Extend our example to a recurrent network (RNN)

\[
\begin{align*}
    h_1(t) &= \sigma(W_1 x(t) + R_1 h_1(t-1) + b_1) \\
    h_2(t) &= \sigma(W_2 h_1(t) + R_2 h_2(t-1) + b_2) \\
    P(t) &= \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}}) \\
    \text{ce}(t) &= \text{cross_entropy}(P(t), L(t)) \sum_{t} \text{ce}(t) = \text{max}
\end{align*}
\]

compare to id (Irvine Dataflow):

[Arvind et al., TR114a, Dept ISC, UC Irvine, Dec 1978; “Executing a Program on the MIT Tagged-Token Dataflow Architecture”, 1988]

```python
@Function
def ip(a: Sequence[tensor], b: Sequence[tensor]):
    s0 = 0
    s_ = ForwardDeclaration()
    s = past_value(s_, initial_value=s0) + a * b
    s_.resolve_to(s)
    s = last(s)
    return s
```

```python
Def ip A B = {  s = 0
               In
                 {For j From 1 To n Do
                     Next s = s + A[j] * B[j]
                 }
               Finally s }
```
Symbolic Loops over Sequential Data

\[
h_1 = \text{sigmoid}(x @ W_1 + \text{past_value}(h_1) @ R_1 + b_1)
\]
\[
h_2 = \text{sigmoid}(h_1 @ W_2 + \text{past_value}(h_2) @ R_2 + b_2)
\]
\[
P = \text{softmax}(h_2 @ W_{out} + b_{out})
\]
\[
\text{ce} = \text{cross_entropy}(P, L)
\]

- CNTK automatically unrolls **cycles at execution time**
  - cycles are detected with Tarjan’s algorithm
  - only nodes in cycles
- efficient and composable

```
lstm = rnn_cell.BasicLSTMCell(lstm_size)
state = tf.zeros([batch_size, lstm.state_size])
probabilities = []
loss = 0.0
for current_batch_of_words in words_in_dataset:
    output, state = lstm(current_batch_of_words, state)
    logits = tf.matmul(output, softmax_w) + softmax_b
    probabilities.append(tf.nn.softmax(logits))
    loss += loss_function(probabilities, target_words)
```
Batch-Scheduling of Variable-Length Sequences

- Minibatches containing sequences of different lengths are automatically packed *and padded*.

Diagram:
- Parallel sequences: sequence 1, sequence 2, sequence 3, sequence 4, sequence 5, sequence 6, sequence 7.
- Time steps computed in parallel.
Batch-Scheduling of Variable-Length Sequences

- Minibatches containing sequences of different lengths are automatically packed and padded

![Diagram showing batch-scheduling of variable-length sequences](image)

- CNTK handles the special cases:
  - past_value operation correctly resets state and gradient at sequence boundaries
  - non-recurrent operations just pretend there is no padding ("garbage-in/garbage-out")
  - sequence reductions

- Batch-scheduling of variable-length sequences:
  - Parallel sequences
  - Time steps computed in parallel
  - Padding sequence 1 and 2 into the same slot, it may come for free!
Batch-Scheduling of Variable-Length Sequences

• Minibatches containing sequences of different lengths are automatically packed and padded
  
  ![](image)

  time steps computed in parallel

• Fully transparent batching
  • Recurrent → CNTK unrolls, handles sequence boundaries
  • Non-recurrent operations → parallel
  • Sequence reductions → mask
Data-Parallel Training

• Degrees of parallelism:
  • Within-vector parallelization: “vectorized”
  • Across independent samples: “batching”
  • Across GPUs: async PCIe device-to-device transfers
  • Across servers: MPI etc., NVidia NCCL

• Parallelization options:
  • Data-parallel
  • Model-parallel
  • Layer-parallel
Data-Parallel Training

• Data-parallelism: distribute minibatch over workers, all-reduce partial gradients

minibatch (sample frames)

all-reduce

each node computes sub-mini-batch sub-gradient of one layer
Data-Parallel Training

• Data-parallelism: distribute minibatch over workers, all-reduce partial gradients

Each node computes sub-mini-batch sub-gradient of one layer

(K-1) concurrent transfers of 1/K-th data

Step 1: each node owns aggregating a stripe
Data-Parallel Training

• Data-parallelism: distribute minibatch over workers, all-reduce partial gradients

<table>
<thead>
<tr>
<th>node 1</th>
<th>node 2</th>
<th>node 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ring algorithm $O(2 \frac{(K-1)}{K M}) \Rightarrow O(1)$ w.r.t. $K$

minibatch (sample frames)

each node computes sub-mini-batch sub-gradient of one layer

step 1: each node owns aggregating a stripe

step 2: aggregated stripes are redistributed
Data-Parallel Training

• Data-parallelism: distribute minibatch over workers, all-reduce partial gradients

• O(1) — enough?

• Example: DNN, MB size 1024, 160M model parameters
  • compute per MB: \(\rightarrow\) 1/7 second
  • communication per MB: \(\rightarrow\) 1/9 second (640M over 6 GB/s)
  • can’t even parallelize to 2 GPUs: communication cost already dominates!

• How about doing it asynchronously?
  • HogWild! [-], DistBelief ASGD [Dean et al., 2012]
  • Helps with latency and jitter, could hide some communication cost with pipeline
  • Does not change the problem fundamentally
Data-Parallel Training

How to reduce communication cost:

**communicate less each time**

- **1-bit SGD:**
  - Quantize gradients to 1 bit per value
  - Trick: carry over quantization error to next minibatch
    - 1-bit quantized with residual

![Diagram showing data parallel training across multiple nodes](image)
Data-Parallel Training

How to reduce communication cost:

**communicate less each time**

- **1-bit SGD:** [F. Seide, H. Fu, J. Droppo, G. Li, D. Yu: “1-Bit Stochastic Gradient Descent...Distributed Training of Speech DNNs”, Interspeech 2014]
  - quantize gradients to 1 bit per value
  - trick: carry over quantization error to next minibatch

**communicate less often**

  - Very recent, very effective parallelization method
  - Combines model averaging with error-residual idea

- **Block momentum** [K. Chen, Q. Huo: “Scalable training of deep learning machines by incremental block training...,” ICASSP 2016]
  - Very recent, very effective parallelization method
  - Combines model averaging with error-residual idea
Outline

• Overview
• CNTK introduction
  • Symbolic loop
  • Batch scheduling
  • Data parallel training

• Educational resources
• Conclusions
Where to begin?

Tutorials:

1. Classify cancer using simulated data (Logistic Regression)
   CNTK 101: Logistic Regression with NumPy

2. Classify cancer using simulated data (Feed Forward, FFN)
   CNTK 102: Feed Forward network with NumPy

3. Recognize hand written digits (OCR) with MNIST data
   CNTK 103 Part A: MNIST data preparation, Part B: Multi-class logistic regression classifier
   Part C: Multi-layer perceptron classifier Part D: Convolutional neural network classifier

4. Learn how to predict the stock market
   CNTK 104: Time Series basics with finance data

5. Compress (using autoencoder) hand written digits from MNIST data with no human input
   unsupervised learning, FFN
   CNTK 105 Part A: MNIST data preparation, Part B: Feed Forward autoencoder

6. Forecasting using data from an IoT device
   CNTK 106: LSTM based forecasting - Part A: with simulated data, Part B: with real IoT data

7. Recognize objects in images from CIFAR-10 data (Convolutional Network, CNN)
   CNTK 201 Part A: CIFAR data preparation, Part B: VGG and ResNet classifiers

8. Infer meaning from text snippets using LSTMs and word embeddings
   CNTK 202: Language understanding

9. Train a computer to perform tasks optionally, (e.g, win games) in a simulated environment
   CNTK 203: Reinforcement learning basics with OpenAI Gym data

10. Translate text from one domain (phraseme) to other (phraseme)
    CNTK 204: Sequence to sequence basics with CMU pronouncing dictionary

11. Teach a computer to paint like Picasso or van Gogh
    CNTK 205: Artistic Style Transfer

12. Produce realistic data (MNIST Images) with no human input (unsupervised learning)
    CNTK 206 Part A: MNIST data preparation, Part B: Basic Generative Adversarial Networks (GAN), Part C: Deep Convolutional GAN

13. Training with Sampled Softmax
    CNTK 207: Training with Sampled Softmax

14. Recognize flowers and animals in natural scene images using deep transfer learning
    CNTK 201: Deep transfer learning with pre-trained ResNet model

Tutorials
Microsoft Cognitive Toolkit (CNTK), an open source deep-learning toolkit [https://docs.microsoft.com/cognitive-](https://docs.microsoft.com/cognitive-)

<table>
<thead>
<tr>
<th>Examples</th>
<th>Manual</th>
<th>Tutorial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Where to begin?

**Azure Notebooks**: Try for free pre-hosted

[https://notebooks.azure.com/cntk/libraries/tutorials]
The Microsoft Cognitive Toolkit

2017-6-1 • 1 min to read • Contributors 🐻 🐾 🐻 🐻 all

The Microsoft Cognitive Toolkit - CNTK - is a unified deep-learning toolkit by Microsoft. This video provides a high-level overview of the toolkit.

The latest release of the Microsoft Cognitive Toolkit is 2.0.

CNTK can be included as a library in your Python or C++ programs, or used as a standalone machine learning tool through its own model description language (BrainScript). In addition you can use the CNTK model evaluation functionality from your C# or Java program.

CNTK supports 64-bit Linux or 64-bit Windows operating systems. To install you can either choose pre-compiled binary packages, or compile the toolkit from the source provided in GitHub.

Here are a few pages to get started:

- Reasons to switch from TensorFlow to CNTK
- Setting up CNTK on your machine
- Tutorials, Examples, Tutorials on Azure
- The CNTK Library APIs
  - Using CNTK from Python
  - Using CNTK from C++
- CNTK using BrainScript
- CNTK Model Evaluation
- How to contribute to CNTK
- Give us feedback through these channels
Python API for CNTK (2.0rc2)

CNTK, the Microsoft Cognitive Toolkit, is a system for describing, training, and executing computational networks. It is also a framework for describing arbitrary learning machines such as deep neural networks (DNNs). CNTK is an implementation of computational networks that supports both CPU and GPU.

This page describes the Python API for CNTK version 2.0rc2. This is an ongoing effort to expose such an API to the CNTK system, thus enabling the use of higher-level tools such as IDEs to facilitate the definition of computational networks, to execute them on sample data in real time. Please give feedback through these channels.

We have a new type system in the layers module to make the input type more readable. This new type system is subject to change, please give us feedback on github or stackoverflow

- Setup
- Getting Started
  - Overview and first run
- Working with Sequences
  - CNTK Concepts
  - Sequence classification
  - Feeding Sequences with NumPy
- Tutorials
- Examples
- Layers Library Reference
  - General patterns
  - Example models
  - Dense()
  - Convolution()
  - MaxPooling(), AveragePooling()
  - GlobalMaxPooling(), GlobalAveragePooling()
Join the experts for an exploration of deep learning, a key enabler—inspired by how our brains work—of the AI-powered technologies which are being developed around the globe. Use the Microsoft Cognitive Toolkit (formerly ONTK) to harness the intelligence within massive datasets in deep learning, with uncompromised scaling, speed, and accuracy. Use Python Jupyter notebooks running on your Windows or Linux machine, and gel hands-on experience with working code, as you walk through this game-changing technology.

What you’ll learn:
- The components of a deep neural network and how they work together
- The basic types of deep neural networks (MLP, CNN, RNN, LSTM) and the type of data each is designed for
- A working knowledge of vocabulary, concepts, and algorithms used in deep learning

What you’ll build:
- An end-to-end model for recognizing hand-written digit images, using a multi-class Logistic Regression and MLP (Multi-Layered Perceptron)
- A CNN (Convolution Neural Network) model for improved digit recognition
- An RNN (Recurrent Neural Network) model to forecast time-series data
- An LSTM (Long Short Term Memory) model to process sequential text data
Outline

• Overview
• CNTK introduction
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Educational resources

• Conclusions
Conclusions

• CNTK: fast, scalable, and extensible
• CNTK Unique features:
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Educational resources
  • Azure Notebooks with CNTK (at no cost)
  • EdX – Deep Learning Explained course

https://github.com/Microsoft/CNTK
Thanks!

https://github.com/Microsoft/CNTK
Appendix
Uber to require selfie security check from drivers

Using Microsoft Cognitive Services, Uber hopes to make riders feel safer by verifying the ID of drivers before rides are given.

By Jake Smith for iGeneration | September 23, 2016 -- 19:59 GMT (03:59 GMT+08:00) | Topic: Innovation

Uber announced on Friday a new security feature called Real-Time ID Check that will require drivers to periodically take a selfie before starting their driving shift. The feature, which begins rolling out to US cities on Friday, uses Microsoft Cognitive Services to reduce fraud and give riders an extra sense of security.

Uber says Microsoft’s feature instantly compares the selfie to the one corresponding with the account on file. If the two images do not match, the driver will be flagged and rejected from driving.
Use the power of Artificial Intelligence for better subtitles
Microsoft Translator learns from the content on your slides to give you better subtitles for your subject matter
Data Partition

- Partition randomly training dataset $\mathcal{D}$ into $S$ mini-batches
  \[ \mathcal{D} = \{ \mathcal{B}_i | i = 1,2, \ldots, S \} \]
- Group every $\tau$ mini-batches to form a split
- Group every $N$ splits to form a data block
- Training dataset $\mathcal{D}$ consists of $M$ data blocks
  \[ S = M \times N \times \tau \]

$\rightarrow$ Training dataset is processed block-by-block
$\rightarrow$ **Incremental Block Training (IBT)**
Intra-Block Parallel Optimization (IBPO)

- Select randomly an unprocessed data block denoted as $\mathcal{D}_t$
- Distribute $N$ splits of $\mathcal{D}_t$ to $N$ parallel workers
- Starting from an initial model denoted as $\mathbf{W}_{init}(t)$, each worker optimizes its local model independently by 1-sweep mini-batch SGD with momentum trick
- Average $N$ optimized local models to get $\overline{\mathbf{W}}(t)$
Blockwise Model-Update Filtering (BMUF)

• Generate model-update resulting from data block $\mathcal{D}_t$:
  \[ \mathbf{G}(t) = \mathbf{W}(t) - \mathbf{W}_{init}(t) \]

• Calculate global model-update:
  \[ \Delta(t) = \eta_t \cdot \Delta(t - 1) + \varsigma_t \cdot \mathbf{G}(t) \]
  - $\varsigma_t$: Block Learning Rate (BLR)
  - $\eta_t$: Block Momentum (BM)
  - When $\varsigma_t = 1$ and $\eta_t = 0 \rightarrow$ MA

• Update global model
  \[ \mathbf{W}(t) = \mathbf{W}(t - 1) + \Delta(t) \]

• Generate initial model for next data block
  - Classical Block Momentum (CBM)
    \[ \mathbf{W}_{init}(t + 1) = \mathbf{W}(t) \]
  - Nesterov Block Momentum (NBM)
    \[ \mathbf{W}_{init}(t + 1) = \mathbf{W}(t) + \eta_{t+1} \cdot \Delta(t) \]
Iteration

• Repeat IBPO and BMUF until all data blocks are processed
  • So-called “one sweep”

• Re-partition training set for a new sweep, repeat the above step

• Repeat the above step until a stopping criterion is satisfied
  • Obtain the final global model $W_{final}$
Benchmark Result of Parallel Training on CNTK

- Training data: 2,670-hour speech from real traffics of VS, SMD, and Cortana
- About 16 and 20 days to train DNN and LSTM on 1-GPU, respectively

Credit: Yongqiang Wang, Kai Chen, Qiang Huo
Results

• Achievement
  • Almost linear speedup without degradation of model quality
  • Verified for training DNN, CNN, LSTM up to 64 GPUs for speech recognition, image classification, OCR, and click prediction tasks

• Released in CNTK as a critical differentiator

• Used for enterprise scale production data loads

• Production tools in other companies such as iFLYTEK and Alibaba
<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld_LogisticRegression</td>
<td>Update README.md</td>
<td>5 months</td>
</tr>
<tr>
<td>ImageHandsOn</td>
<td>Revision based on CR</td>
<td>3 months</td>
</tr>
<tr>
<td>NumpyInterop</td>
<td>Adding tests to layers library, removing previous non-layers construct...</td>
<td>17 days</td>
</tr>
<tr>
<td>SLUHandsOn</td>
<td>Restructuring examples and tutorials</td>
<td>5 months</td>
</tr>
<tr>
<td>CNTK_101_LogisticRegression.ipynb</td>
<td>Bump version numbers</td>
<td>5 days</td>
</tr>
<tr>
<td>CNTK_102_FeedForward.ipynb</td>
<td>Bump version numbers</td>
<td>5 days</td>
</tr>
<tr>
<td>CNTK_103A_MNIST_DataLoader.ipynb</td>
<td>added CR feedback and import cleanup</td>
<td>a day ago</td>
</tr>
<tr>
<td>CNTK_103C_MNIST_MultiLayerPerceptron.ipynb</td>
<td>Notebooks update and deleting stale previous version</td>
<td>a day ago</td>
</tr>
<tr>
<td>CNTK_104_Finance_Timeseries فلاش</td>
<td>Fix Notebook...</td>
<td>29 days</td>
</tr>
<tr>
<td>CNTK_105_Basic_Autoencoder_for_</td>
<td>Merge with master.</td>
<td>28 days</td>
</tr>
<tr>
<td>CNTK_106A_LSTM_Timeseries_with_</td>
<td>Add local worker’s state checkpointing</td>
<td>22 days</td>
</tr>
<tr>
<td>CNTK_106B_LSTM_Timeseries_with_</td>
<td>Py 3.6 build &amp; test</td>
<td>14 days</td>
</tr>
<tr>
<td>CNTK_201A_CIFAR-10_Dataloader</td>
<td>Fix Notebook...</td>
<td>29 days</td>
</tr>
<tr>
<td>CNTK_202A_Language_Understanding</td>
<td>Bump version numbers</td>
<td>5 days</td>
</tr>
<tr>
<td>CNTK_203_Reinforcement_Learning</td>
<td>CNTK v2 library: Migrate past_value and future_value to sequence</td>
<td>27 days</td>
</tr>
<tr>
<td>CNTK_204_Sequence_To_Sequenc...</td>
<td>Bump version numbers</td>
<td>5 days</td>
</tr>
<tr>
<td>CNTK_205_Artistic_Style_Transfer.ipynb</td>
<td>CNTK v2 library: Remove remnants of the use of deprecated input_varia...</td>
<td>27 days</td>
</tr>
<tr>
<td>CNTK_206A_Basic_GAN.ipynb</td>
<td>Merge with master.</td>
<td>28 days</td>
</tr>
<tr>
<td>CNTK_206B_DCGAN.ipynb</td>
<td>Merge with master.</td>
<td>28 days</td>
</tr>
<tr>
<td>CNTK_207_Training_with_Sampled...</td>
<td>CNTK v2 library: Migrate past_value and future_value to sequence</td>
<td>27 days</td>
</tr>
<tr>
<td>CNTK_208_Language_Understanding</td>
<td>More fixes and examples for Parameter and Constant</td>
<td>28 days</td>
</tr>
</tbody>
</table>
Thank you