Scalable Deep Learning with Microsoft Cognitive Toolkit

Sayan Pathak
Principal ML Scientist

Chris Basoglu
Partner Dev Manager

With many contributors:
Outline

• CNTK overview
• Key features
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Conclusions
Outline

• CNTK overview
• Key features
  • Symbolic loop
  • Batch scheduling
  • Data parallel training
• Conclusions
Deep learning at Microsoft

- Microsoft Cognitive Services
- Skype Translator
- Cortana
- Bing
- HoloLens
- Microsoft Research
Microsoft Cognitive Toolkit

How-Old.net
How old do I look?

Microsoft

CaptionBot

I am not really confident, but I think it’s a group of young children sitting next to a child and they seem 😊.

How did I do?

★★★★★

Microsoft and Liebherr together to make Refrigerators smart

Smart refrigerators, Cortana, Microsoft and Liebherr

When this joint venture of Microsoft and Liebherr will come into reality, it will be the next level of machine learning. SmartDeviceBox is nothing a communication module which fits into Liebherr refrigerators and freezers, connecting them to the internet. The modular units can be integrated and upgraded at any time in existing SmartDevice-ready appliances to create value and comfort for customers through new digital features and solutions.
Microsoft had all 5 entries being the 1-st places this year: ImageNet classification, ImageNet localization, ImageNet detection, COCO detection, and COCO segmentation.
I'm sending you a new file tonight.

Chris Whiting said:

Ich sende dir heute abend eine neue Datei.
Break the language barrier

Translated conversations across devices, for one-on-one chats and for larger group interactions.
You can follow along to this presentation on your own device, in the language of your choice.

1. Download the Microsoft Translator app for Android, iOS, or Windows
   or
   Visit translate.it/<ENTER CODE HERE>

2. Type in this unique conversation code below to join this conversation

<ENTER CODE HERE>

Microsoft Translator is powered by machine learning. Any voice or text information you provide will be used to improve Microsoft products and services.
Bing / Bing Ads

**Most sold fruit in US**

Bananas

The most popular fresh fruits in the United States are (in order): **Bananas, apples, 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.agmrz.org/commodities_products/fruits/

**Shop for bugs bunny books**

- **Your Complete Bugs Bunny**
  - Bugs: Bugs Bunny Calling (A Bugs Bunny Special Offer)
  - $29.95

- **Bugs Bunny And The Little Prince**
  - $3.79

- **Bugs Bunny & Little Prince**
  - $3.79

- **Little Big Book Bugs bunny**
  - $7.99

eBay
Microsoft’s historic speech breakthrough

- Microsoft 2016 research system for conversational speech recognition
- 5.9% word-error rate
- enabled by CNTK’s multi-server scalability

Microsoft Customer Support Agent
Hello!

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.)
Hello!

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.
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

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 in 2012
  - 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
Microsoft Cognitive Toolkit (CNTK)

• Over 80% Microsoft internal DL workload runs CNTK
• 1st-class on Linux and Windows, docker support
• Python, C++, C#, Java
• Internal == External
# CNTK – The Fastest Toolkit

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></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></td>
<td>288.142ms</td>
<td>-</td>
<td>1130.606ms</td>
</tr>
<tr>
<td>(v7 benchmark)</td>
<td>(44.917ms)</td>
<td></td>
<td>(284.898ms)</td>
<td>(223.547ms)</td>
<td>(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]
Superior performance

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® 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™ architecture GPUs 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 194% 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.
Scalability

Microsoft, Cray claim deep learning breakthrough on supercomputers

Steve Ranger

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.
Microsoft has now released a major upgrade of the software and rebranded it as part of the Microsoft Cognitive Toolkit. This release is a major improvement over the initial release.

There are two major changes from the first release that you will see when you begin to look at the new release. First is that CNTK now has a very nice Python API and, second, the documentation and examples are excellent.

Installing the software from the binary builds is very easy on both Ubuntu Linux and Windows.
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
Defining CNTK networks
The Microsoft Cognitive Toolkit (CNTK)

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

• CNTK is **production-ready**: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.
MNIST Handwritten Digits (OCR)

- Data set of hand written digits with
  - 60,000 training images
  - 10,000 test images
- Each image is: 28 x 28 pixels
Multi-layer perceptron

- Input layer: $D_{i=784}$, output layer: $O=400$
- Activation function: $a = \text{relu}$
- Hidden layers:
  - $D_{i=400}$, output layer: $O=200$
  - 10 nodes, output layer: $O=10$
  - Activation function: $a = \text{None}$

Weights:
- 784 x 400
- 400 x 200
- 200 x 10

Bias:
- 400
- 200
- 10

Deep Model:
- $e^{z_i}$
- $\sigma_j = \frac{e^{z_j}}{\sum_{j=0}^{9} e^{z_j}}$

softmax

https://github.com/Microsoft/CNTK/tree/master/Tutorials
Loss function

\[
\text{ce} = - \sum_{j=0}^{9} y_j \log(p_j)
\]

Cross entropy error

Label One-hot encoded (Y)

0 0 0 1 0 0 0 0 0 0

28 x 28 pix (p)

Predicted Probabilities (p)

0.08 0.08 0.10 0.17 0.11 0.09 0.08 0.08 0.13 0.01

Model (w, b)
CNTK Model

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

\[ ce = y^T \log P \]
CNTK Model

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

\[ ce = y^T \log P \]
h1 = sigmoid (x @ W1 + b1)
h2 = sigmoid (h1 @ W2 + b2)
P = softmax (h2 @ Wout + bout)
ce = cross_entropy (P, y)
CNTK Model

- 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

LEGO-like composability allows CNTK to support wide range of networks & applications
Authoring networks as functions

• “model function”
  • features \(\rightarrow\) predictions
  • defines the model structure & parameter initialization
  • holds parameters that will be learned by training

• “criterion function”
  • (features, labels) \(\rightarrow\) (training loss, additional metrics)
  • defines training and evaluation criteria on top of the model function
  • provides gradients w.r.t. training criteria
Authoring networks as functions

- **CNTK** model: neural networks are functions
  - pure functions
  - with “special powers”:
    - can compute a gradient w.r.t. any of its nodes
    - external deity can update model parameters

- user specifies network as function objects:
  - formula as a Python function (low level, e.g. LSTM)
  - function composition of smaller sub-networks (layering)
  - higher-order functions (equiv. of scan, fold, unfold)
  - model parameters held by function objects

- “compiled” into the static execution graph under the hood
Layers Library Reference

Note: This documentation has not yet been completely updated with respect to the latest update of the Layers library. It should be correct but misses several new options and layer types.

CNTK predefines a number of common “layers,” which makes it very easy to write simple networks that consist of standard layers layered on top of each other. Layers are function objects that can be used like a regular `Function` but hold learnable parameters and have an additional pair of parameters to pass construction parameters or attributes.

For example, this is the network description for a simple 1-hidden layer model using the `Dense()` layer:

```python
h = Dense(1024, activation=relu)(features)
p = Dense(9000, activation=softmax)(h)
```

which can then, e.g., be used for training against a cross-entropy criterion:

```python
ce = cross_entropy(p, labels)
```

If your network is a straight concatenation of operations (many are), you can use the alternative `Sequential()` notation:

```python
from cntk.layers import *
my_model = Sequential([
    Dense(1024, activation=relu),
    Dense(9000, activation=softmax)
])
```
Distinguishing Features
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)

\[
\begin{align*}
    h_1 &= \sigma(W_1 x + b_1) \\
    h_2 &= \sigma(W_2 h_1 + b_2) \\
    P &= \text{softmax}(W_{\text{out}} h_2 + b_{\text{out}}) \\
    ce &= y^T \log P
\end{align*}
\]
Symbolic loops over sequential data

extend our example to a recurrent network (RNN)

\[ h_1(t) = \sigma(W_1 x(t) + b_1) \]
\[ h_2(t) = \sigma(W_2 h_1(t) + b_2) \]
\[ P(t) = \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}}) \]
\[ ce(t) = y^T(t) \log P(t) \]
Symbolic loops over sequential data

extend our example to a recurrent network (RNN)

\[ h_1(t) = \sigma(W_1 x(t) + H_1 h_1(t-1) + b_1) \]
\[ h_2(t) = \sigma(W_2 h_1(t) + H_2 h_2(t-1) + b_2) \]
\[ P(t) = \text{softmax}(W_{out} h_2(t) + b_{out}) \]
\[ ce(t) = y^T(t) \log P(t) \]
Symbolic loops over sequential data

extend our example to a recurrent network (RNN)

\[
\begin{align*}
    h_1(t) &= \sigma(W_1 x(t) + H_1 h_1(t-1) + b_1) \\
    h_2(t) &= \sigma(W_2 h_1(t) + H_2 h_2(t-1) + b_2) \\
    P(t) &= \text{softmax}(W_{out} h_2(t) + b_{out}) \\
    ce(t) &= L^T(t) \log P(t) \\
    \sum_{\text{corpus}} ce(t) &= \text{max}
\end{align*}
\]

\[
\begin{align*}
    h_1 &= \text{sigmoid}(x \ @ W_1 + \text{past_value}(h_1) \ @ H_1 + b_1) \\
    h_2 &= \text{sigmoid}(h_1 \ @ W_2 + \text{past_value}(h_2) \ @ H_2 + b_2) \\
    P &= \text{softmax}(h_2 \ @ W_{out} + b_{out}) \\
    ce &= \text{cross_entropy}(P, L)
\end{align*}
\]
Symbolic loops over sequential data

\[
\begin{align*}
    h_1 &= \text{sigmoid}(x \cdot W_1 + \text{past\_value}(h_1) \cdot H_1 + b_1) \\
    h_2 &= \text{sigmoid}(h_1 \cdot W_2 + \text{past\_value}(h_2) \cdot H_2 + b_2) \\
    P &= \text{softmax}(h_2 \cdot W_{\text{out}} + b_{\text{out}}) \\
    \text{ce} &= \text{cross\_entropy}(P, y)
\end{align*}
\]

- CNTK automatically unrolls cycles $\rightarrow$ deferred computation
- Efficient and composable
Batch-scheduling of variable-length sequences

• minibatches containing sequences of different lengths are automatically packed and padded

  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

sequence 2

sequence 3

sequence 4

sequence 5

sequence 6

sequence 7
Batch-scheduling of variable-length sequences

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

Parallel sequences:

- time steps computed in parallel

Examples of sequences:
- sequence 1
- sequence 2
- sequence 3
- sequence 4
- sequence 5
- sequence 6
- sequence 7
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]
Batch-scheduling of variable-length sequences

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

![Diagram showing parallel sequences and time steps computed in parallel](image-url)
Batch-scheduling of variable-length sequences

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

---

**Diagram:**
- Time steps computed in parallel
- Parallel sequences
  - Sequence 1
  - Sequence 2
  - Sequence 3
  - Sequence 4
Batch-scheduling of variable-length sequences

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

<table>
<thead>
<tr>
<th>Parallel sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence 1</td>
</tr>
<tr>
<td>sequence 2</td>
</tr>
<tr>
<td>sequence 3</td>
</tr>
<tr>
<td>sequence 4</td>
</tr>
<tr>
<td>sequence 5</td>
</tr>
<tr>
<td>sequence 6</td>
</tr>
</tbody>
</table>

Time steps computed in parallel
Batch-scheduling of variable-length sequences

• minibatches containing sequences of different lengths are automatically packed *and padded*

![Diagram of batch-scheduling of variable-length sequences]

- **Sequence Reductions**: parallel sequences across multiple time steps computed in parallel.

- **CNTK Handling**:
  - **past_value** operation correctly resets state and gradient at sequence boundaries.
  - Non-recurrent operations just pretend there is no padding ("garbage in/garbage out").
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)
Batch-scheduling of variable-length sequences

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

- speed-up is automatic:

![Speed comparison on RNNs](image)

- time steps computed in parallel
Data-parallel training

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

minibatch (sample frames)

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

all-reduce
Data-parallel training

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

![Diagram showing data-parallel training]

- Minibatch (sample frames)
- 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

ring algorithm
$O\left(\frac{2(K-1)}{KM}\right)$
$\Rightarrow O(1)$ w.r.t. $K$
Data-parallel training

How to reduce communication cost:

*communicate less each time*

*communicate less often*
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
    1-bit quantized with residual

1-bit quantized with residual
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
    1-bit quantized with residual
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**

- **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
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

### 1bit/BMUF Speedup Factors in LSTM Training

<table>
<thead>
<tr>
<th>GPUs</th>
<th>1bit-average</th>
<th>1bit-peak</th>
<th>BMUF-average</th>
<th>BMUF-peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2.9</td>
<td>3.3</td>
<td>5.4</td>
<td>6.7</td>
</tr>
<tr>
<td>8</td>
<td>4.1</td>
<td>6.9</td>
<td>8.1</td>
<td>10.8</td>
</tr>
<tr>
<td>16</td>
<td>8.0</td>
<td>13.8</td>
<td>11.1</td>
<td>25.5</td>
</tr>
<tr>
<td>32</td>
<td>14.1</td>
<td>17.7</td>
<td>21.9</td>
<td>27.3</td>
</tr>
<tr>
<td>64</td>
<td>25.5</td>
<td>37.4</td>
<td>34.5</td>
<td>43.7</td>
</tr>
</tbody>
</table>

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
Where to begin?

On GitHub: https://github.com/Microsoft/CNTK/wiki

The Microsoft Cognitive Toolkit

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

The latest release of the Microsoft Cognitive Toolkit 2.0 is RC1 (release candidate 1). If you are a previous user of the toolkit, see this page for more information about (breaking) changes in this release.

It 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). 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:

• Setting up CNTK on your machine
• Tutorials, Examples, etc..
  • Try the tutorials on Azure Notebooks with pre-installed CNTK
• The CNTK Library APIs
  • Using CNTK from Python
  • Using CNTK from C++
• CNTK as a machine learning tool through BrainScript
• How to contribute to CNTK
• Give us feedback through these channels.

Seek help on Stack Overflow:
http://stackoverflow.com/search?q=cntk (please add cntk tag)
Where to begin?

Tutorials:
https://www.cntk.ai/pythondocs/tutorials.html (latest release)
https://github.com/Microsoft/CNTK/tree/master/Tutorials (latest)
Where to begin?

Azure Notebooks: Try for free pre-hosted
https://notebooks.azure.com/cntk/libraries/tutorials
Where to begin?

On GitHub: [https://github.com/Microsoft/CNTK/wiki](https://github.com/Microsoft/CNTK/wiki)

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

The latest release of the Microsoft Cognitive Toolkit 2.0 is RC1 (release candidate 1). If you are a previous user of the toolkit, see this page for more information about (breaking) changes in this release.

It 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). 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:

- Setting up CNTK on your machine
- Tutorials, Examples, etc.
  - Try the tutorials on Azure Notebooks with pre-installed CNTK
- The CNTK Library APIs
  - Using CNTK from Python
  - Using CNTK from C++
- CNTK as a machine learning tool through BrainScript
- How to contribute to CNTK
- Give us feedback through these channels.

Seek help on Stack Overflow: [http://stackoverflow.com/search?q=cntk](http://stackoverflow.com/search?q=cntk) (please add cntk tag)