Deep Learning with Cognitive Toolkit

Sayan Pathak
Principal ML Scientist

Chris Basoglu
Partner Dev Manager

+ 120 contributors

#MSBuild
Deep learning at Microsoft

- Microsoft Cognitive Services
- Skype Translator
- Cortana
- Bing
- HoloLens
- Microsoft Research
How-Old.net

How old do I look?

The magic behind How-Old.net

Try Another Photo!

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 but a communication module which fits into Liebherr refrigerators and freezers, connecting them to the internet. The modular unit can be integrated and upgraded at any time, like an "intelligent appliance" 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.
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.agmrc.org/commodities_products/fruits/

Shop for bugs bunny books

Your Complete Bugs... $29.95
Bugs Bunny Calling A... $3.79
Bugs Bunny And The... $3.59
Bugs Bunny & Little... $3.79
Little Big Book Bugs... $7.99

Microsoft Cognitive Toolkit
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
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.
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
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
CTNK – 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></td>
<td></td>
<td>(44.917ms)</td>
<td>(284.898ms)</td>
<td></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

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 Tesla 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 N-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 benefitting from deeper insight on extreme amounts of data.
Microsoft, Cray claim deep learning breakthrough on supercomputers

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.
What is new in CNTK 2.0?

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 handwritten digits with
  - 60,000 training images
  - 10,000 test images
- Each image is: 28 x 28 pixels
Multi-layer perceptron

784 pixels (x)

D

Di = 784

O = 400

a = relu

Di = 400

O = 200

a = relu

Di = 200

O = 10

a = None

Deep Model

Weights

784

400

+ 400 bias

784

400

+ 200 bias

200

400

+ 10 bias

10 nodes

softmax

\[ \frac{e^{z_i}}{\sum_{j=0}^{9} e^{z_j}} \]
Loss function

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

Label One-hot encoded (Y)

Model (w, b)

Predicted Probabilities (p)

Cross entropy error
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

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

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

and cross-entropy training criterion

\[
\begin{align*}
    ce &= y^T \log P
\end{align*}
\]

\[
\begin{align*}
    h1 &= \text{sigmoid} (x \ @ \ w1 \ + \ b1) \\
    h2 &= \text{sigmoid} (h1 \ @ \ w2 \ + \ b2) \\
    P &= \text{softmax} (h2 \ @ \ wout \ + \ bout)
\end{align*}
\]

\[
\begin{align*}
    ce &= \text{cross_entropy} (P, y)
\end{align*}
\]
h1 = sigmoid (x @ W1 + b1)

h2 = sigmoid (h1 @ W2 + b2)

P = softmax (h2 @ Wout + bout)

ce = cross_entropy (P, y)
• 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 → execution engine
• Editable, clonable

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

“model function”
- **features → predictions**
- defines the **model structure** & parameter initialization
- holds parameters that will be learned by training

“criterion function”
- (features, labels) → (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 arguments 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)
)
```
Script configure and executes through CNTK Python APIs...

reader
- minibatch source
- task-specific deserializer
- automatic randomization
- distributed reading

network
- model function
- criterion function
- CPU/GPU execution engine
- packing, padding

trainer
- SGD (momentum, Adam, ...)
- minibatching

Corpus -> Reader -> Network -> Trainer -> Model
As easy as 1-2-3

from cntk import *

# reader
def create_reader(path, is_training):
    ...

# network
def create_model_function():
    ...
def create_criterion_function(model):
    ...

# trainer (and evaluator)
def train(reader, model):
    ...
def evaluate(reader, model):
    ...

# main function
model = create_model_function()
reader = create_reader(..., is_training=True)
train(reader, model)

reader = create_reader(..., is_training=False)
evaluate(reader, model)
Workflow

• prepare data
• configure reader, network, learner (Python)
• train:

  python my_cntk_script.py
def create_reader(map_file, mean_file, is_training):

    # deserializer
    return MinibatchSource(ImageDeserializer(map_file, StreamDefs(
        features = StreamDef(field='image', transforms=transforms), ')
    labels    = StreamDef(field='label', shape=num_classes)
    )), randomize=is_training, epoch_size = INFINITELY_REPEAT if is_training else
    FULL_DATA_SWEEP)
def create_reader(map_file, mean_file, is_training):
    # image preprocessing pipeline
    transforms = [
        ImageDeserializer.crop(crop_type='Random', ratio=0.8, jitter_type='uniRatio'),
        ImageDeserializer.scale(width=image_width, height=image_height, channels=num_channels, interpolations='linear'),
        ImageDeserializer.mean(mean_file)
    ]
    # deserializer
    return MinibatchSource(ImageDeserializer(map_file, StreamDefs(
        features = StreamDef(field='image', transforms=transforms), 'labels = StreamDef(field='label', shape=num_classes)
    )), randomize=is_training, epoch_size = INFINITELY_REPEAT if is_training else FULL_DATA_Sweep)

• automatic on-the-fly randomization important for large data sets
• readers compose, e.g. image → text caption
Distributed training

• prepare data
• configure reader, network, learner (Python)
• train:  --distributed!

    mpiexec --np 16 --hosts server1,server2,server3,server4  \
    python my_cntk_script.py
Workflow

• prepare data
• configure reader, network, learner (Python)
• train:
  mpiexec --np 16 --hosts server1,server2,server3,server4 \ python my_cntk_script.py

• deploy
  • offline (Python): apply model file-to-file
  • your code: embed model through C++ API
  • online: web service wrapper through C#/Java API
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)

\[ h_1 = \sigma(W_1 x + b_1) \]
\[ h_2 = \sigma(W_2 h_1 + b_2) \]
\[ P = \text{softmax}(W_{out} h_2 + b_{out}) \]
\[ ce = L^T \log P \]
\[ \sum_{\text{corpus}} ce = \text{max} \]
Symbolic Loops over Sequential Data

extend our example to a recurrent network (RNN)

\[
\begin{align*}
h_1(t) &= \sigma(W_1 x(t) + b_1) \\
h_2(t) &= \sigma(W_2 h_1(t) + b_2) \\
P(t) &= \text{softmax}(W_{out} h_2(t) + b_{out}) \\
ce(t) &= L^T(t) \log P(t) \\
\Sigma_{\text{corpus}} ce(t) &= \text{max}
\end{align*}
\]
Symbolic Loops over Sequential Data

extend our example to a recurrent network (RNN)

\[ h_1(t) = \sigma(W_1 x(t) + H_1 h_1(t-1) + b_1) \]
\[ h_2(t) = \sigma(W_2 h_1(t) + H_2 h_2(t-1) + b_2) \]
\[ P(t) = \text{softmax}(W_{out} h_2(t) + b_{out}) \]
\[ ce(t) = L^T(t) \log P(t) \]
\[ \sum_{\text{corpus}} ce(t) = \max \]
Symbolic Loops over Sequential Data

extend our example to a recurrent network (RNN)

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

\[ h_1 = \text{sigmoid}(x \ @ \ W_1 + \text{past_value}(h_1) \ @ \ H_1 + b_1) \]
\[ h_2 = \text{sigmoid}(h_1 \ @ \ W_2 + \text{past_value}(h_2) \ @ \ H_2 + b_2) \]
\[ P = \text{softmax}(h_2 \ @ \ W_{out} + b_{out}) \]
\[ ce = \text{cross_entropy}(P, L) \]
Symbolic Loops over Sequential Data

h1 = sigmoid(x \cdot W1 + past_value(h1) \cdot H1 + b1)

h2 = sigmoid(h1 \cdot W2 + past_value(h2) \cdot H2 + b2)

P = softmax(h2 \cdot Wout + bout)

ce = cross_entropy(P, L)

- CNTK automatically unrolls cycles → deferred computation
- Efficient and composable
Batch-Scheduling of Variable-Length Sequences

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

- 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

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

![Diagram showing time steps computed in parallel and parallel sequences.](image)
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>
</tbody>
</table>

time steps computed in parallel
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>
</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:
- **parallel sequences**
  - sequence 1
  - sequence 2
  - sequence 3
  - sequence 4

- **time steps computed in parallel**
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]
Batch-Scheduling of Variable-Length Sequences

- minibatches containing sequences of different lengths are automatically packed and padded
**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 \textit{and padded}

\begin{itemize}
  \item speed-up is automatic:
\end{itemize}
Data-Parallel Training

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

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

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

**communicate less often**

- **Automatic MB sizing** [F. Seide, H. Fu, J. Droppo, G. Li, D. Yu: "ON Parallelizability of Stochastic Gradient Descent...", ICASSP 2014]
- **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

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
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 (formerlyCNTK) 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 get 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 handwritten 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

---

Sayan Pathak, PhD

**Microsoft Principal Machine Learning Scientist**

Dr. Sayan Pathak, Principal Machine Learning Scientist at Microsoft, is also a Principal Investigator for NIH-funded projects in healthcare. He is on the faculty for Bioinformatics at Tufts University.

Roland Fernandez

**Microsoft Deep Learning Team Senior Researcher**

Roland Fernandez, Microsoft Deep Learning Team Senior Researcher, is also a Microsoft AI School Instructor. He has previously worked in areas of Information Technology, Human-Computer Interfaces.
Where to begin?

On GitHub:  [https://github.com/Microsoft/CNTK/wiki](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](https://www.cntk.ai/pythondocs/tutorials.html) 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](https://github.com/Microsoft/CNTK/tree/master/Tutorials) 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](https://github.com/Microsoft/CNTK/wiki).

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