



Microsoft Cognitive Toolkit

aka.ms/CognitiveToolkit



Scalable deep document / sequence reasoning with Cognitive Toolkit (CNTK)

Sayan Pathak, Pengcheng He and William Darling

With many contributors:

A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, W. Darling, J. Droppo, A. Eversole, B. Guenter, P. He, M. Hillebrand, X. Huang, Z. Huang, R. Hoens, V. Ivanov, A. Kamenev, N. Karampatziakis, P. Kranen, O. Kuchaiev, W. Manousek, C. Marschner, A. May, B. Mitra, O. Nano, G. Navarro, A. Orlov, M. Radmilac, A. Reznichenko, P. Parthasarathi, S. Pathak, B. Peng, A. Reznichenko, W. Richert, F. Seide, M. Seltzer, M. Slaney, A. Stolcke, T. Will, H. Wang, Z. Wang, W. Xio. Yao, D. Yu, C. Zhang, Y. Zhang, G. Zweig



Agenda

- Cognitive Toolkit Overview
- Deep dive on key features
- Intro to recurrence and LSTM/GRU
- Hands-on:
 - Language understanding
 - Sequence to Sequence with Attention
 - Machine comprehension with ReasonNet

<https://github.com/Microsoft/CNTK/wiki/WWW-2017-Tutorial>

deep learning at Microsoft






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






Vision	Speech	Language	Knowledge	Search
Computer Vision	Bing Speech	Bing Spell Check	Academic	Bing Autosuggest
Content Moderator	Custom Recognition	Language Understanding	Entity Linking	Bing Image Search
Emotion	Speaker Recognition	Linguistic Analysis	Knowledge Exploration	Bing News Search
Face		Text Analytics	QnA Maker	Bing Video Search
Video		Translator	Recommendations	Bing Web Search
		WebLM		

Still looking for the right API? [See the entire collection >](#)

Vision

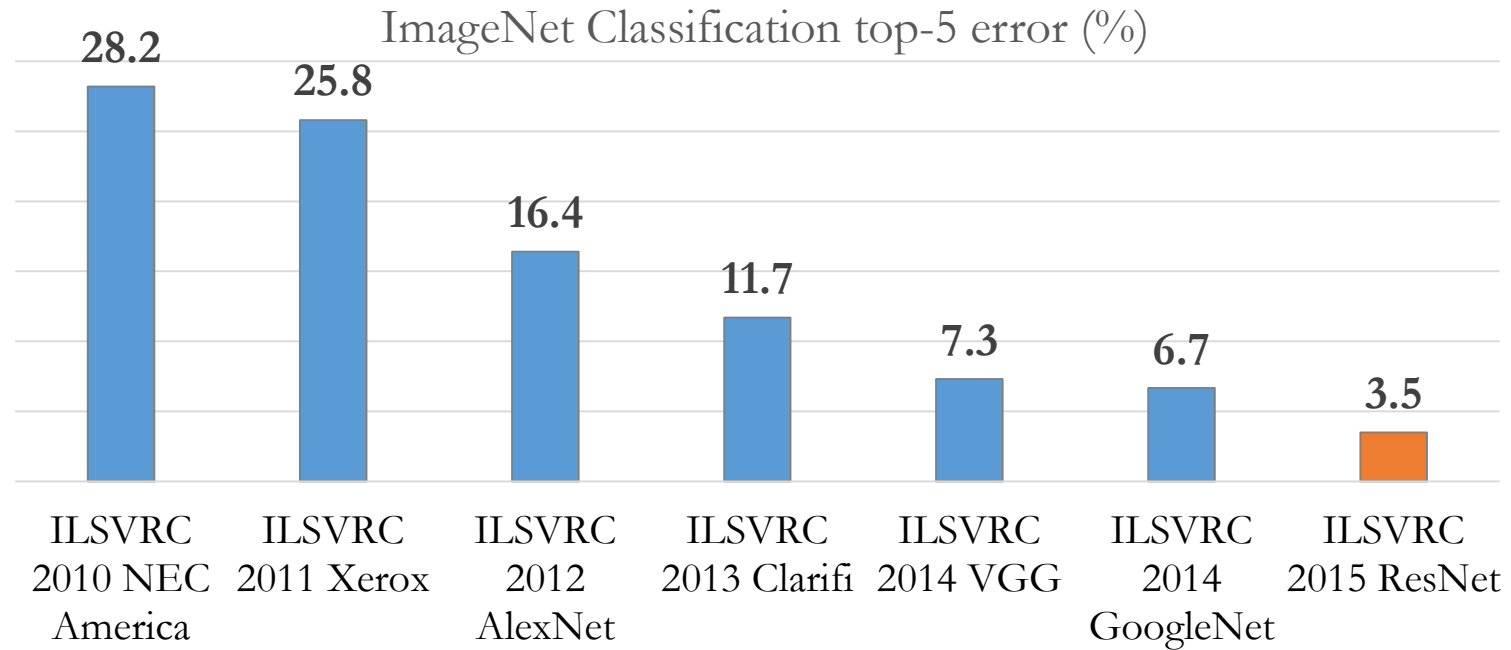
 Computer Vision API <i>Preview</i> Distill actionable information from images	 Content Moderator <i>Preview</i> Automatically moderate text, images, and videos, augmented with human review tools.	 Emotion API <i>Preview</i> Personalize experiences with emotion recognition
 Face API <i>Preview</i> Detect, identify, analyze, organize, and tag faces in photos	 Video API <i>Preview</i> Analyze, edit, and process videos within your app	

Speech

 Bing Speech API <i>Preview</i> Convert speech to text and back again, and understand its intent	 Custom Recognition Intelligent Service (CRIS) <i>Private Preview</i> Fine-tune speech recognition for anyone, anywhere	 Speaker Recognition API <i>Preview</i> Give your app the ability to know who's talking
--	--	--



ImageNet: Microsoft 2015 ResNet



Microsoft had all **5 entries** being the 1-st places this year: ImageNet classification, ImageNet localization, ImageNet detection, COCO detection, and COCO segmentation



How-Old.net

How old do I look? #HowOldRobot



Sorry if we didn't quite get it right - [we are still improving this feature.](#)

Try Another Photo!



P.S. We don't keep the photo

Share 2.3M Tweet

The magic behind How-Old.net

Privacy & Cookies | Terms of Use | View Source



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?





MUST READ PIXEL, GALAXY, IPHONE, OH MY! WHY PAY A PREMIUM WHEN EVERY PHONE RUNS THE SAME APPS?

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

Share buttons for Facebook (23), LinkedIn (3), Twitter, and Email.

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

SHARING ECONOMY



RECOMMENDED FOR YOU

Software Defined Networking Service (Japanese)

White Papers provided by IBM

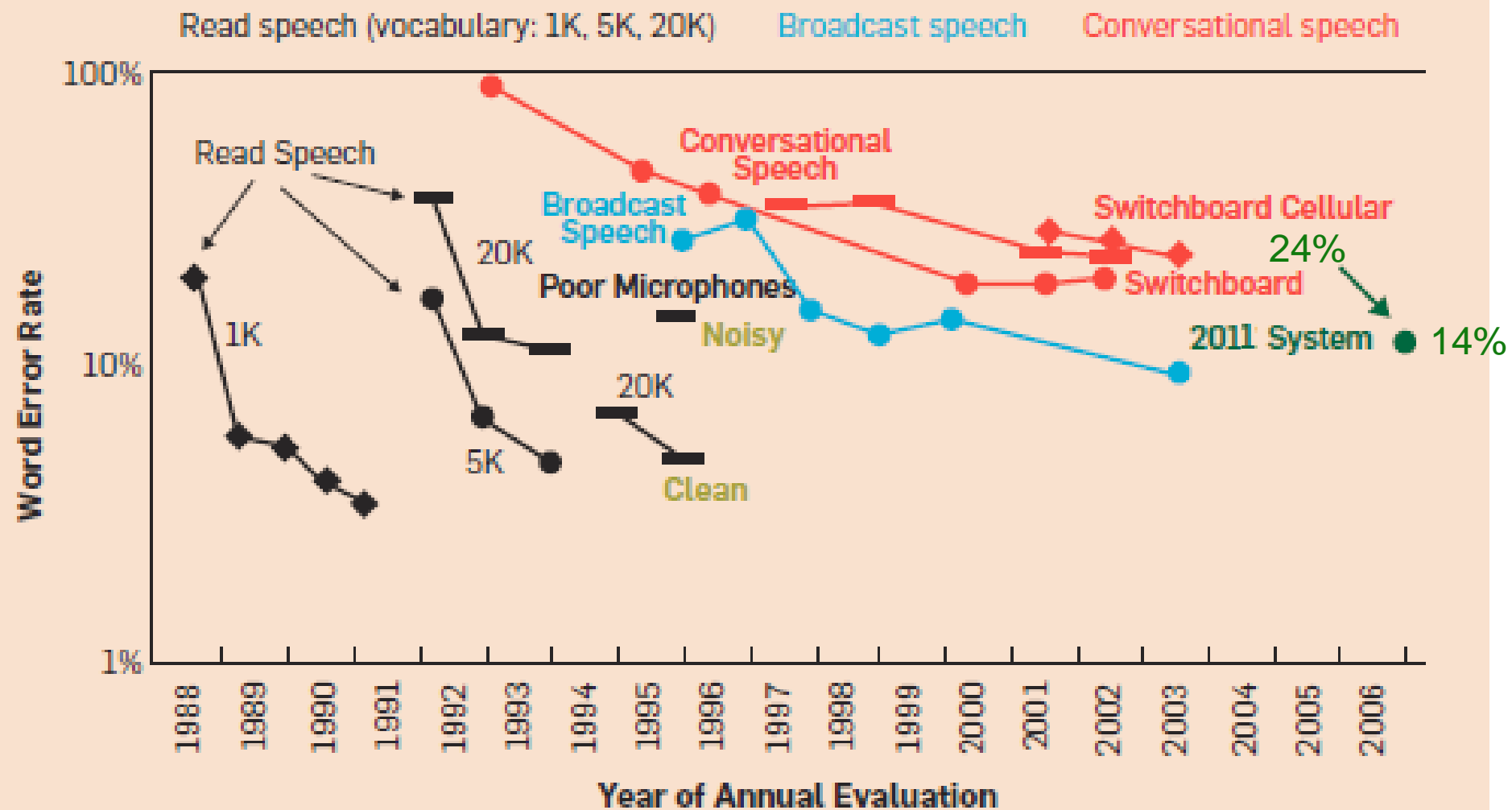
DOWNLOAD NOW

RELATED STORIES



Innovation Victoria partners with Bosch for self-driving vehicle development

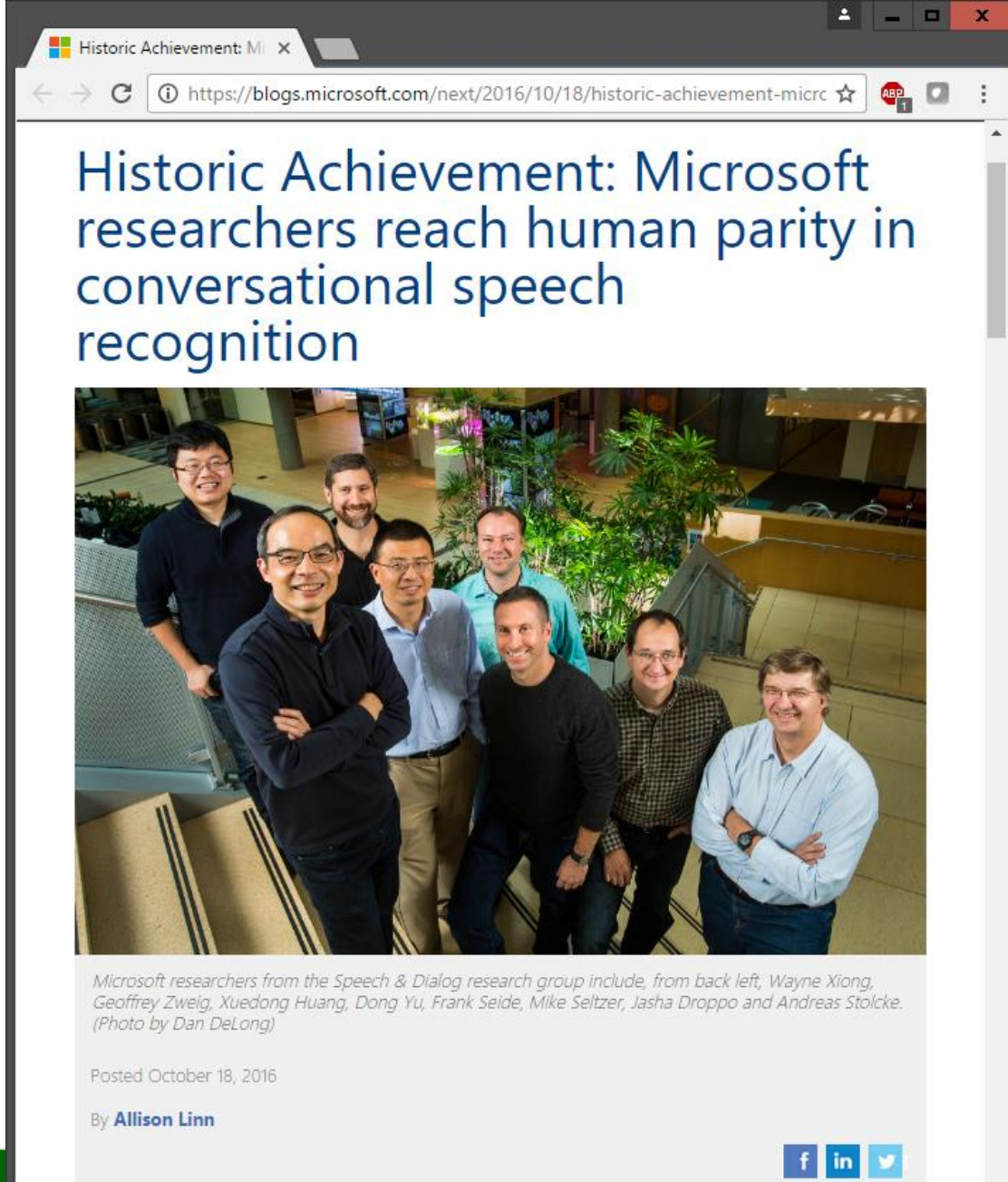
Figure 1. Historical progress of speech recognition word error rate on more and more difficult tasks.¹⁰ The latest system for the switchboard task is marked with the green dot.



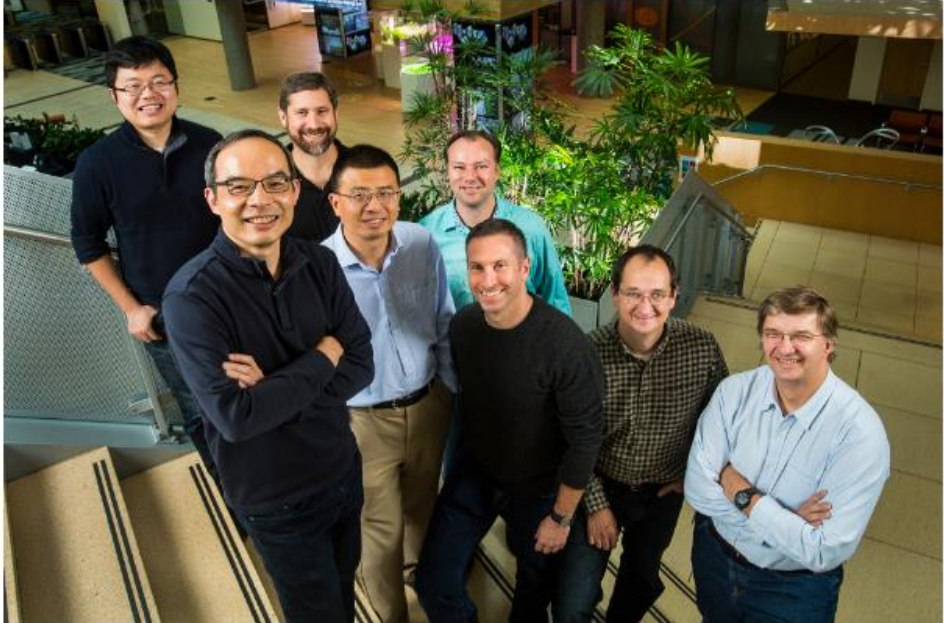
Microsoft's historic speech breakthrough

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

[W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig: "Achieving Human Parity in Conversational Speech Recognition," <https://arxiv.org/abs/1610.05256>]



Historic Achievement: Microsoft researchers reach human parity in conversational speech recognition



Microsoft researchers from the Speech & Dialog research group include, from back left, Wayne Xiong, Geoffrey Zweig, Xuedong Huang, Dong Yu, Frank Seide, Mike Seltzer, Jasha Droppo and Andreas Stolcke. (Photo by Dan DeLong)

Posted October 18, 2016

By [Allison Linn](#)

[f](#) [in](#) [t](#)

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.

[Youtube Link](#)



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.



Hello!

Suggestion

Connect to a projector

<https://windows.microsoft.com/en-us/windows7/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. NoteProjectors 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!

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 (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.

Enter your response here



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 ...)

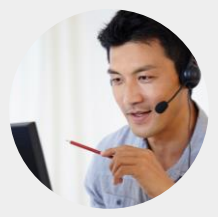


Hello!

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.

“CNTK is production-ready: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.”

Benchmarking on a single server by HKBU

seconds per minibatch on G1080 (**G980**) GPU; lower=better

	FCN-8	AlexNet		ResNet-50		LSTM-64
CNTK	0.037	0.040	(0.054)	0.207	(0.245)	0.122
Caffe	0.038	0.026	(0.033)	0.307	(-)	-
TensorFlow	0.063	-	(0.058)	-	(0.346)	0.144
Torch	0.048	0.033	(0.038)	0.188	(0.215)	0.194

[“Benchmarking State-of-the-Art Deep Learning Software Tools,” <http://arxiv.org/pdf/1608.07249v5.pdf>]

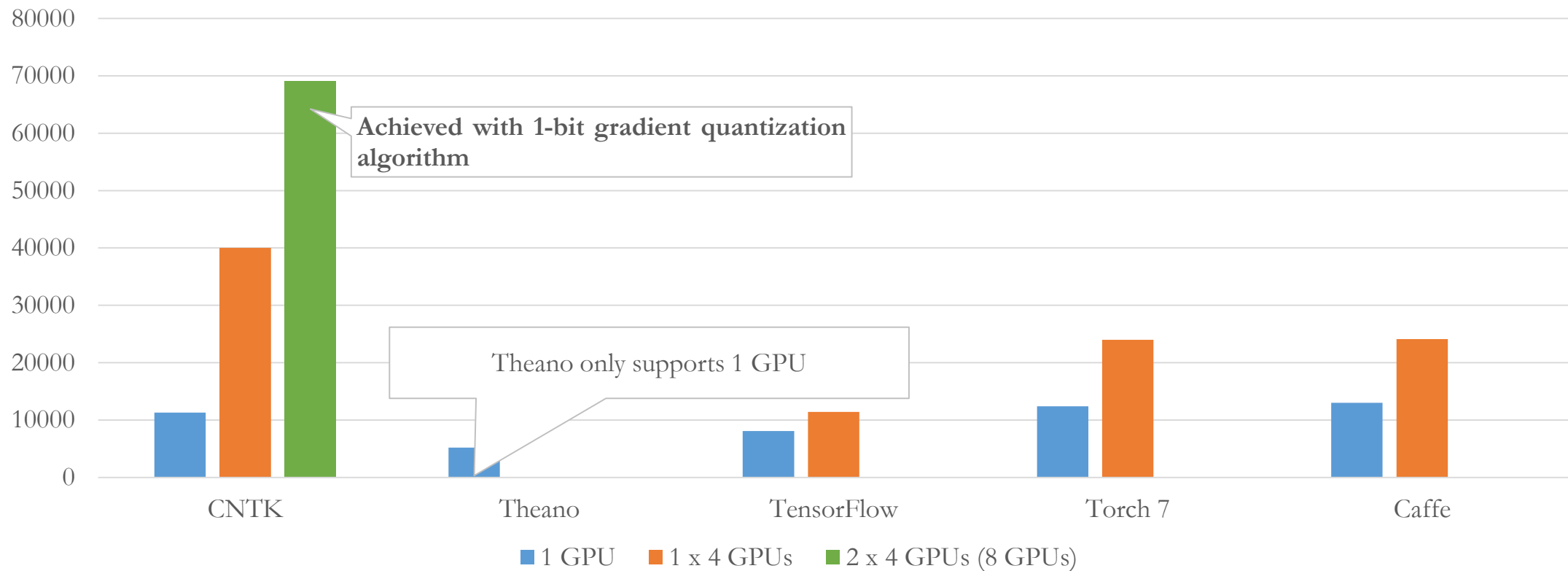
Recent update

- With a single GPU platform:
 - Caffe, **CNTK** and Torch perform better than MXNet and TensorFlow on FCNs
 - MxNet is outstanding in CNNs, while Caffe and CNTK also achieve good performance.
 - For RNN of LSTM, **CNTK** obtains excellent time efficiency, which is up to **5-10 times better than other tools**.
 - CNTK out performs TensorFlow on all categories often by a large margin.

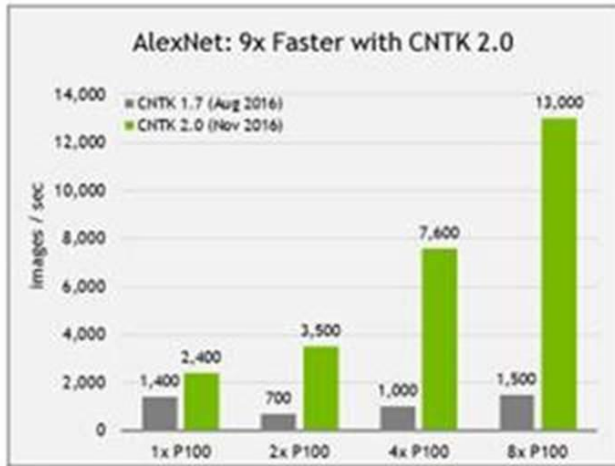
“CNTK is production-ready: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.”

speed comparison (samples/second), higher = better

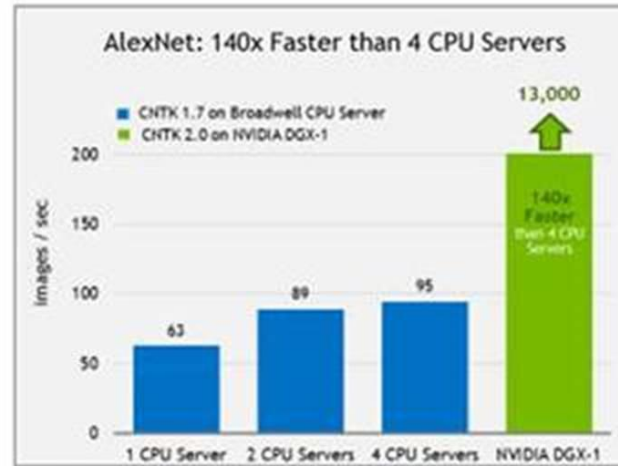
[note: December 2015]



Superior performance



AlexNet training batch size: 128.
 CNTK 1.7 includes: cuDNN 5.1, NVLink enabled
 CNTK 2.0 includes: cuDNN 5.1.8, NCCL 1.6.1, NVLink enabled



Global batch size: 1024. Each dual socket E5-2698v4 CPUs @2.20 GHz, Quad rail EDR InfiniBand
 CNTK 2.0 includes: cuDNN 5.1.8, NCCL 1.6.1, NVLink enabled



News

NVIDIA and Microsoft Accelerate AI Together

Monday, November 14, 2016

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® 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 194x 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.

Superior performance

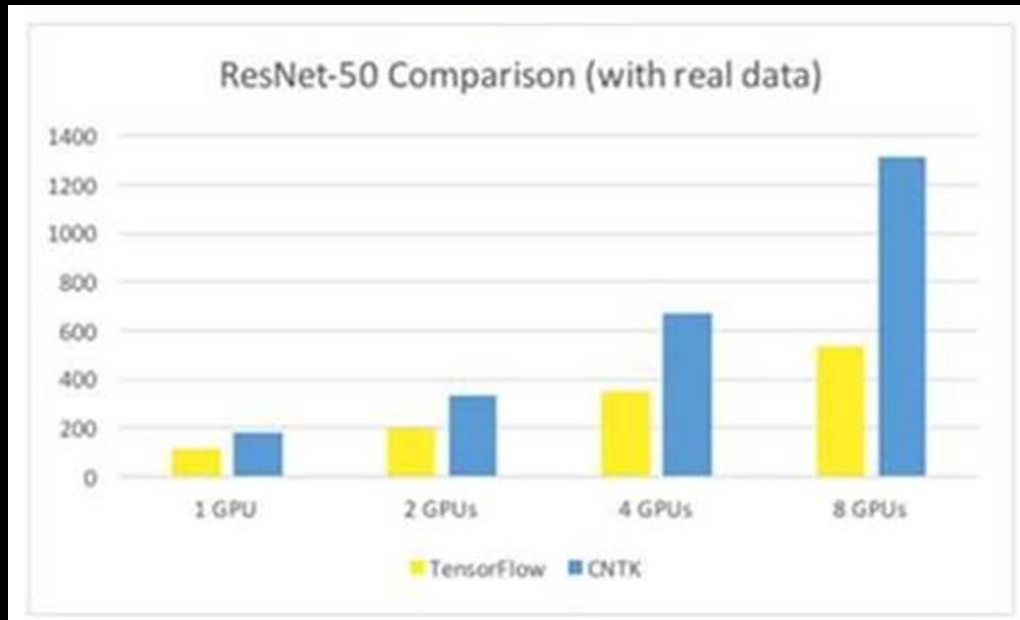


Image: Cray

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](#).

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.

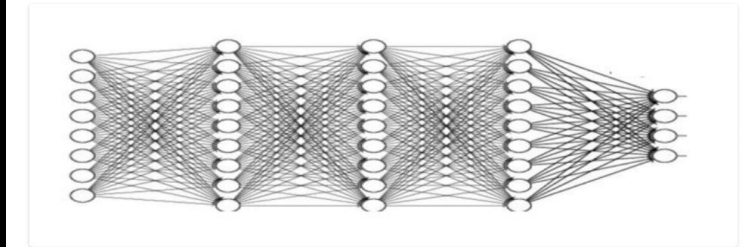
There are two major changes from the first release that you will see when you begin to look at the new release. First is that CNTK now has a very nice Python API and, second, the documentation and examples are excellent.

Installing the software from the binary builds is very easy on both Ubuntu Linux and Windows.

The eScience Cloud

Cloud and HPC Solutions for Science

[HOME](#) [ABOUT](#) [CLOUD CONCEPTS](#) [SOFTWARE, TOOLS AND ALGORITHMS](#)



CNTK Revisited. A New Deep Learning Toolkit Release from Microsoft

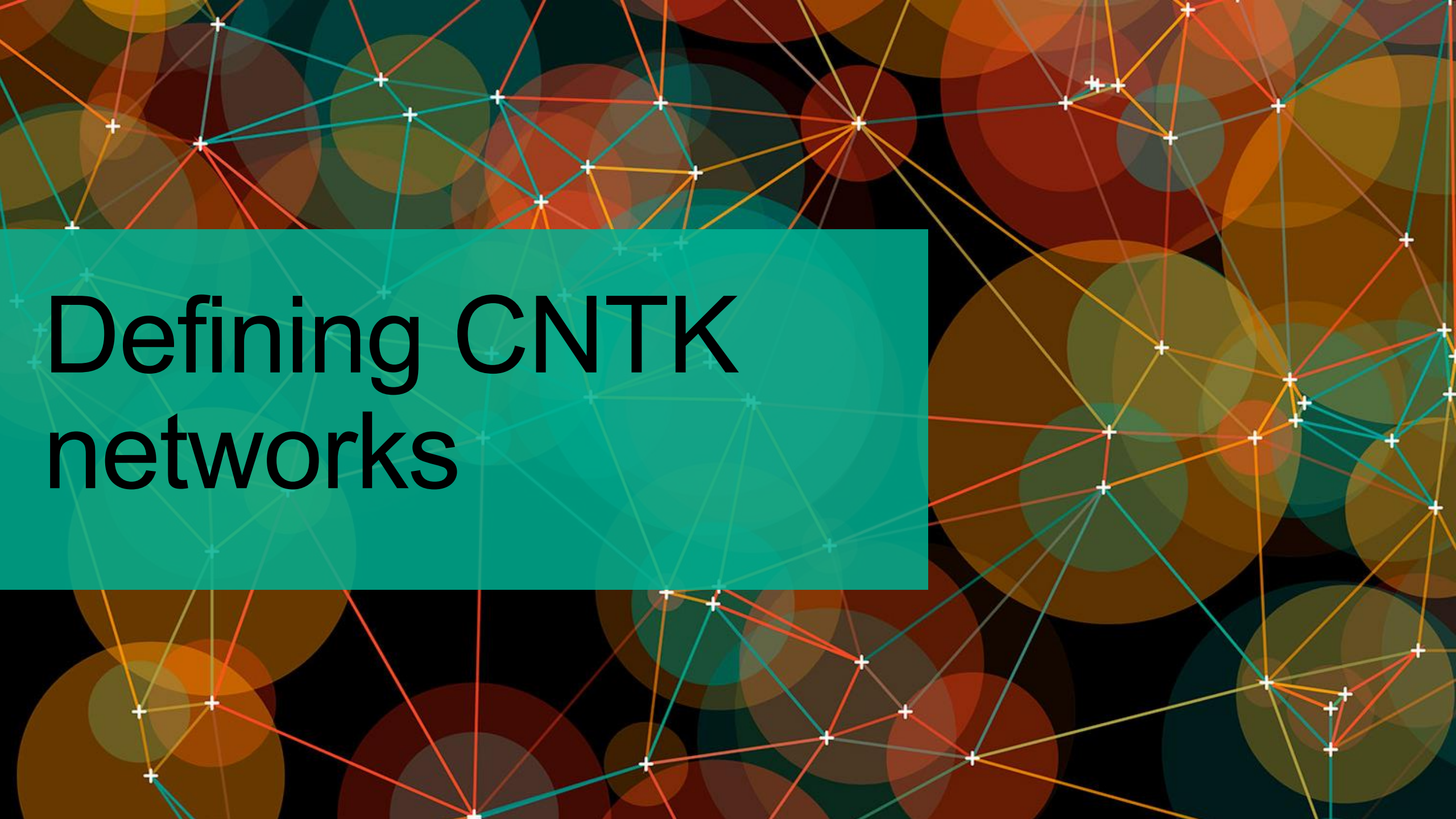
[Leave a reply](#)

In a pair of articles from last winter ([first article](#), [second article](#)) we looked at Microsoft's "Computational Network Toolkit" and compared it to Google's Tensorflow. 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. Because these older articles still get a fair amount of web traffic we wanted to provide a proper update.

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. The core concepts are the same as in the initial release. The original programming model was based on configuration scripts and that is still there, but it has been improved and renamed as "Brain Script". Brain Script is still an excellent way to build custom networks, but we will focus on the Python API which is very well [documented](#).

Installing the software from the binary builds is very easy on both Ubuntu Linux and Windows. The process is described in the CNTK [github site](#). On a Linux machine, simply download the gzipped tarred binary and execute the installer.

<https://esciencegroup.com/2016/11/10/cntk-revisited-a-new-deep-learning-toolkit-release-from-microsoft/>



Defining CNTK networks

The Microsoft Cognitive Toolkit (CNTK)

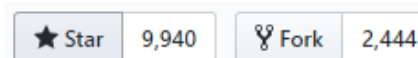
- CNTK expresses (nearly) **arbitrary neural networks** by composing simple building blocks into complex **computational networks**, supporting relevant network types and applications.
- CNTK is **production-ready**: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.

“CNTK is Microsoft’s open-source, cross-platform toolkit for learning and evaluating deep neural networks.”

- open-source model inside and outside the company
 - created by Microsoft Speech researchers (Dong Yu et al.) in 2012, “Computational Network Toolkit”
 - open-sourced (CodePlex) in early 2015
 - on GitHub since Jan 2016 under permissive license
 - Python support since Oct 2016 (beta), rebranded as “Cognitive Toolkit”
 - used by Microsoft product groups; but code development is out in the open
 - external contributions e.g. from MIT and Stanford
- Linux, Windows, docker, cudnn5, CUDA 8
- Python and C++ API (beta; C#/.Net on roadmap)
- Keras integration in progress

Microsoft Cognitive Toolkit, CNTK

- CNTK is a library for deep neural networks
 - model definition
 - scalable training
 - efficient I/O
- easy to author, train, and use neural networks
 - think “what” not “how”
 - focus on composability
- Python, C++, C#, Java
- open source since 2015 <https://github.com/Microsoft/CNTK>
 - created by Microsoft Speech researchers (Dong Yu et al.) in 2012, “Computational Network Toolkit”
 - contributions from MS product groups and external (e.g. MIT, Stanford), development is visible on Github
 - Linux, Windows, docker, cudnn5, CUDA 8

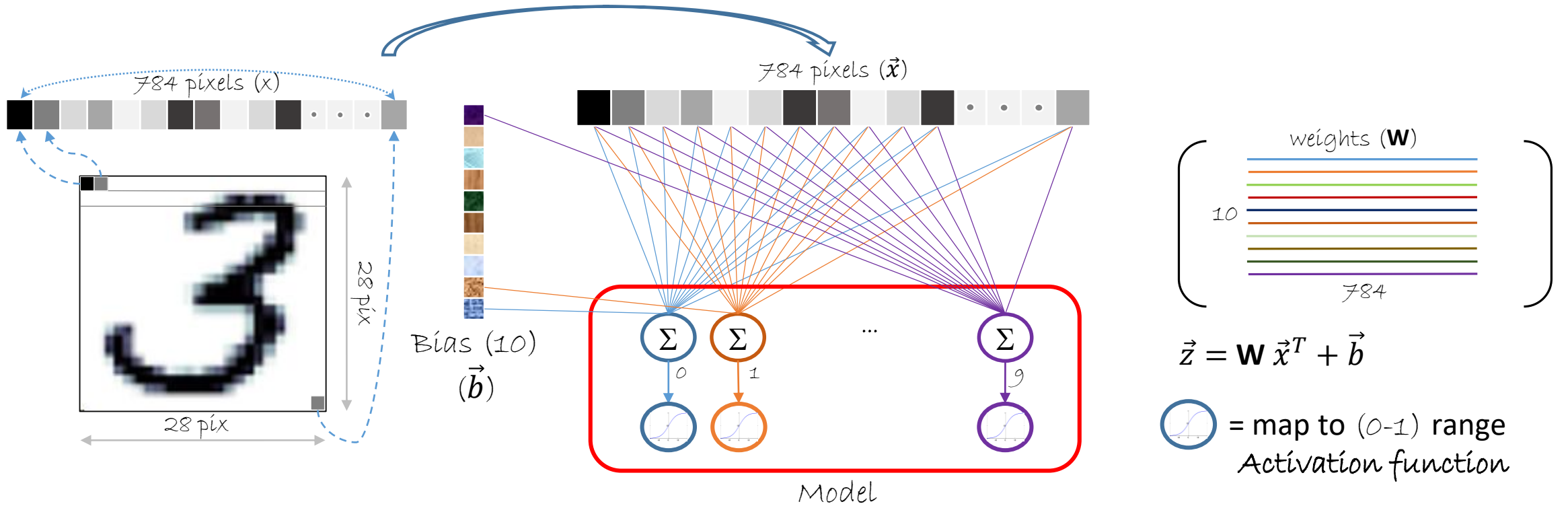


MNIST Handwritten Digits (OCR)

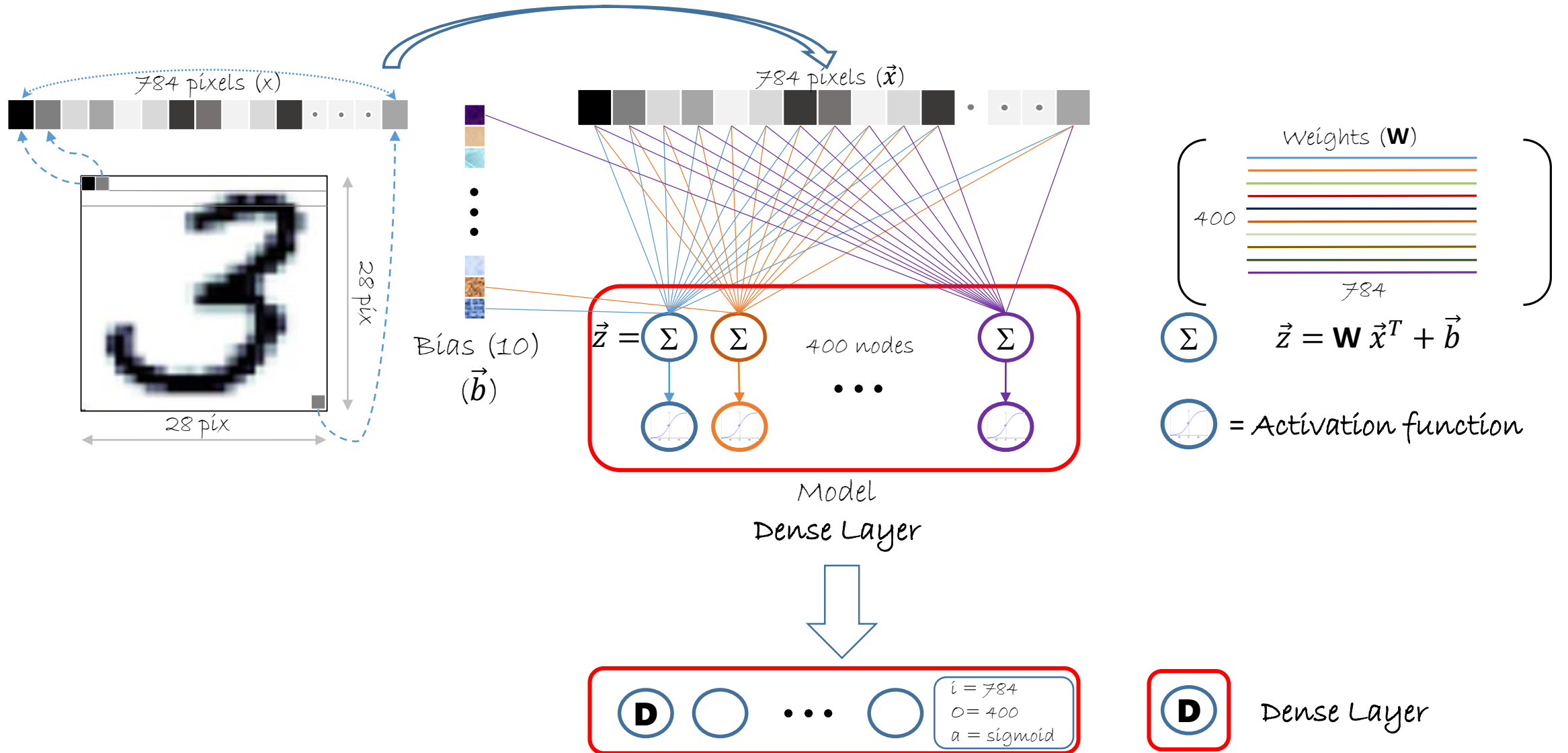


- Data set of hand written digits with
 - ✓ 60,000 training images
 - ✓ 10,000 test images
- Each image is: 28 x 28 pixels
- Performance with different classifiers (error rate):
 - ✓ Neural nets (2-layers): 1.6 %
 - ✓ Deep nets (6-layers): 0.35 %
 - ✓ Conv nets (different): 0.21% - 0.31%

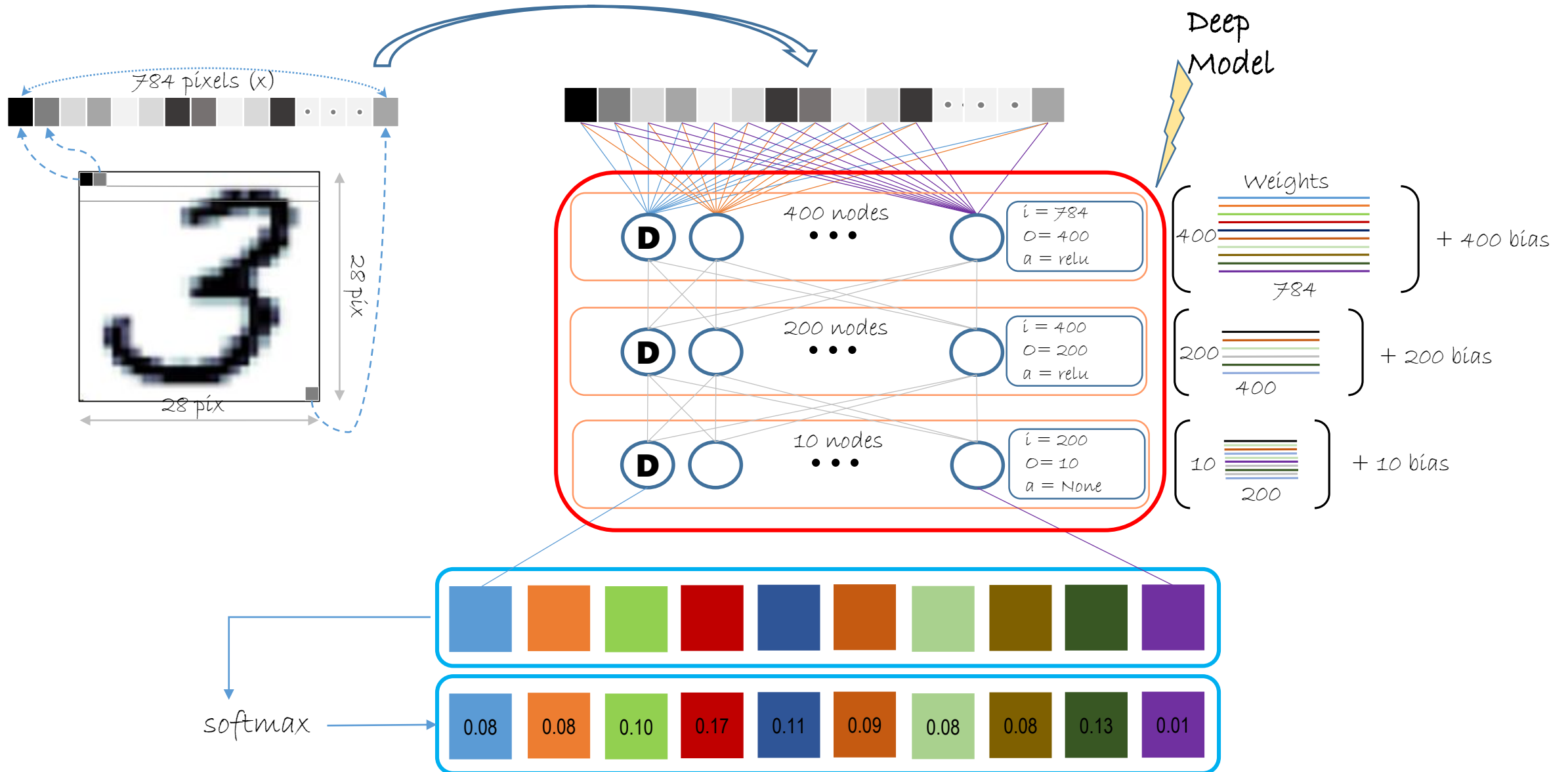
Logistic Regression



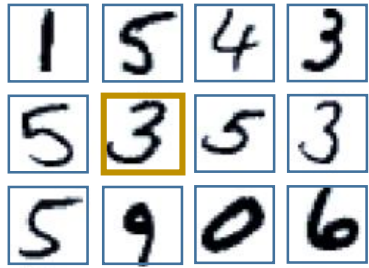
Single-Layer Perceptron



Multi-layer Perceptron

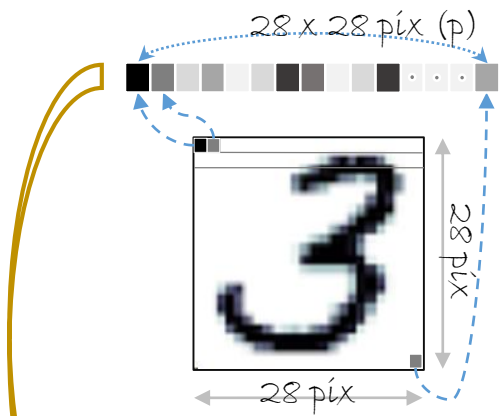
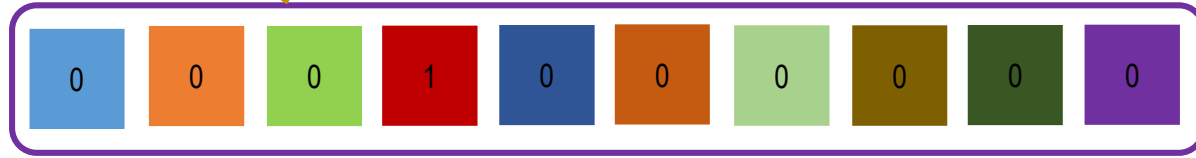


Error or Loss Function



1	5	4	3
5	3	5	3
5	9	0	6

Label One-hot encoded (Y)



Loss function

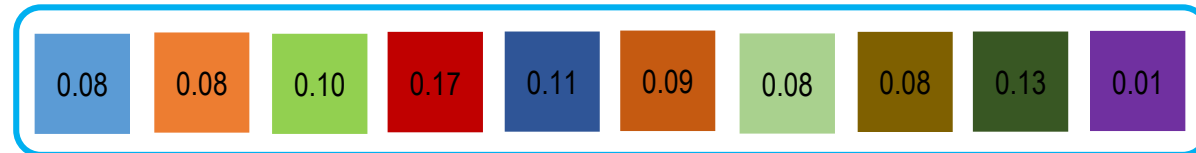
$$se = \sum_{j=0}^9 (y_j - p_j)^2$$

Squared error

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

Cross entropy error

Predicted Probabilities (p)

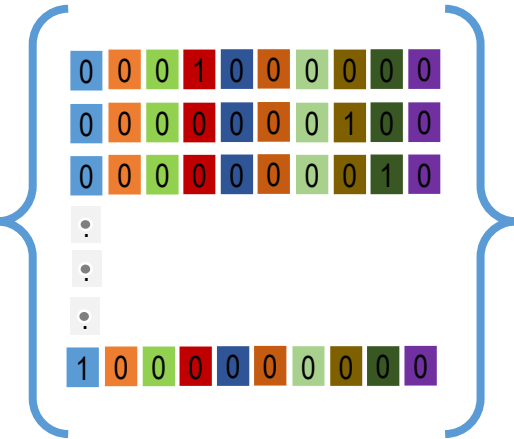
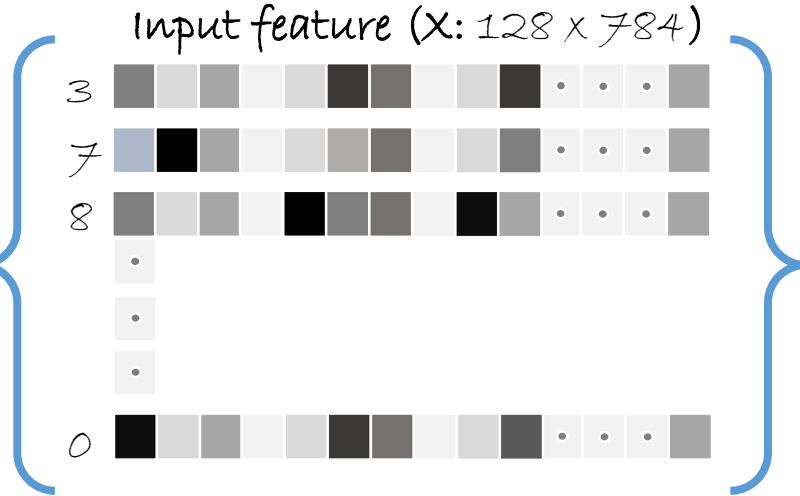


Model
(w, b)

Train Workflow

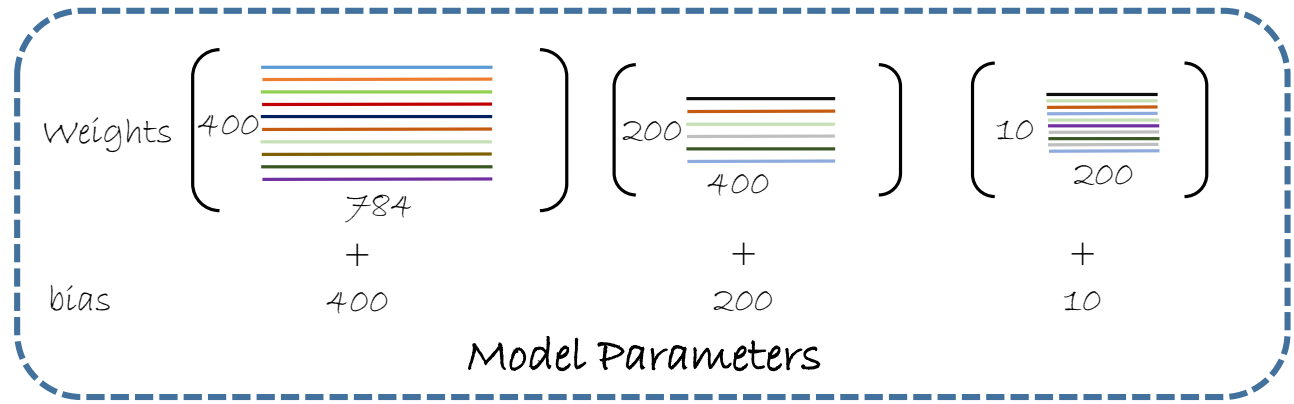


128 samples
(mini-batch)



One-hot
encoded
Label
(Y: 128 x 10)

Model



```
z = model(X):
    h1 = Dense(400, act = relu)(X)
    h2 = Dense(200, act = relu)(h1)
    r = Dense(10, act = None)(h2)
    return r
```

Loss

```
cross_entropy_with_softmax(p, Y)
```

Error
(optional)

```
classification_error(p, Y)
```

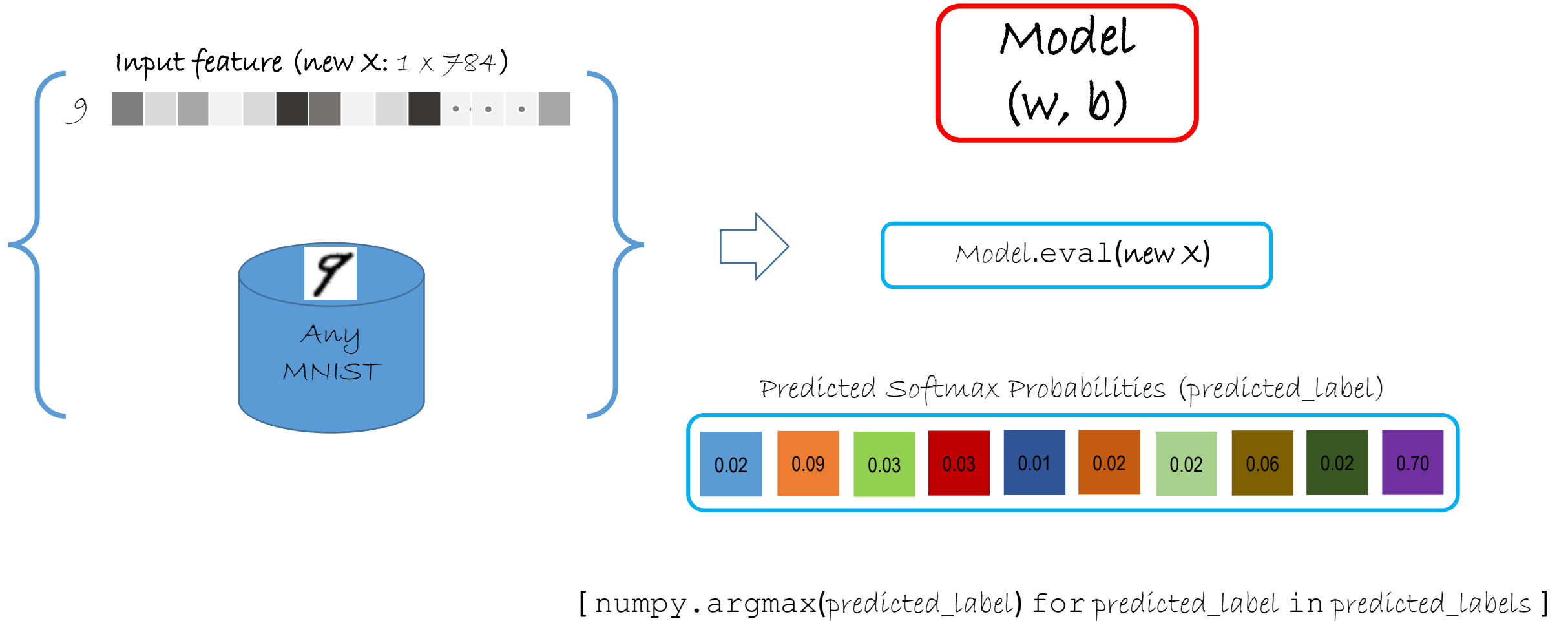
Trainer(model, (loss, error), learner)

Trainer.train_minibatch({X, Y})

Learner

sgd, adagrad etc, are solvers to estimate - w & b

Prediction Workflow



[9]

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

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$

$$h_2 = \sigma(W_2 h_1 + b_2)$$

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

with input $x \in \mathbf{R}^M$

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

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$

$$h_2 = \sigma(W_2 h_1 + b_2)$$

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

with input $x \in \mathbf{R}^M$ and one-hot label $y \in \mathbf{R}^J$
and cross-entropy training criterion

$$ce = y^T \log P$$

$$\sum_{\text{corpus}} ce = \max$$

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

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$

$$h_2 = \sigma(W_2 h_1 + b_2)$$

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



$$h1 = \text{sigmoid} (x @ W1 + b1)$$

$$h2 = \text{sigmoid} (h1 @ W2 + b2)$$

$$P = \text{softmax} (h2 @ Wout + bout)$$

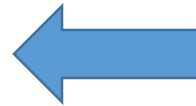
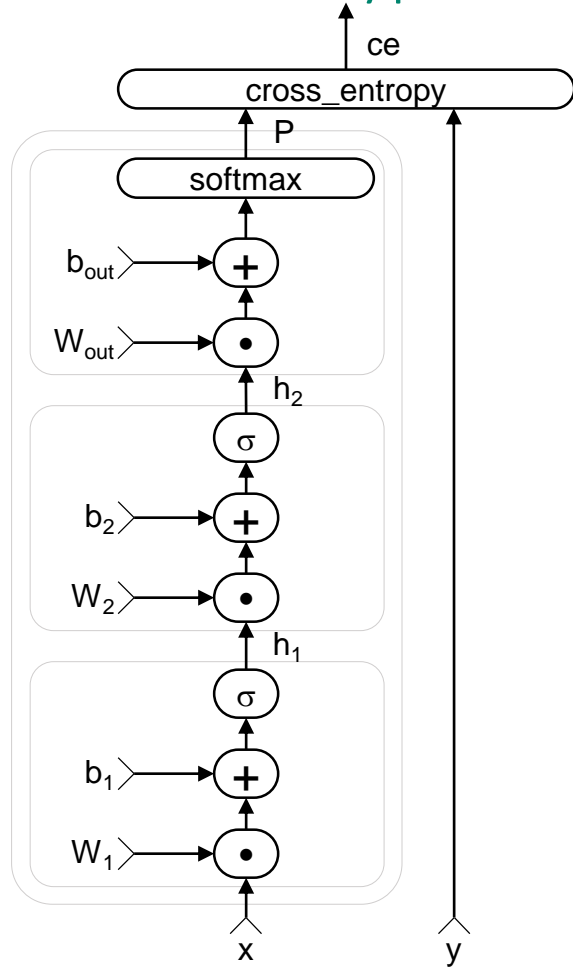
with input $x \in \mathbb{R}^M$ and one-hot label $y \in \mathbb{R}^J$
and cross-entropy training criterion

$$ce = y^T \log P$$

$$\sum_{\text{corpus}} ce = \max$$

$$ce = \text{cross_entropy} (P, y)$$

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



$$h_1 = \text{sigmoid} (x @ W_1 + b_1)$$
$$h_2 = \text{sigmoid} (h_1 @ W_2 + b_2)$$
$$P = \text{softmax} (h_2 @ W_{out} + b_{out})$$
$$ce = \text{cross_entropy} (P, y)$$

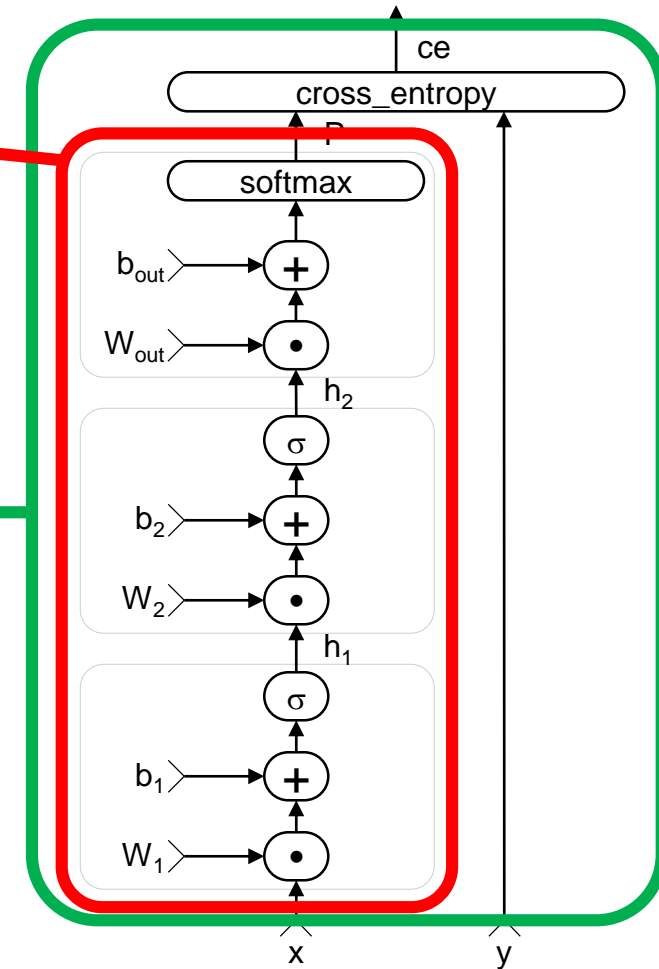
authoring networks as functions

- “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

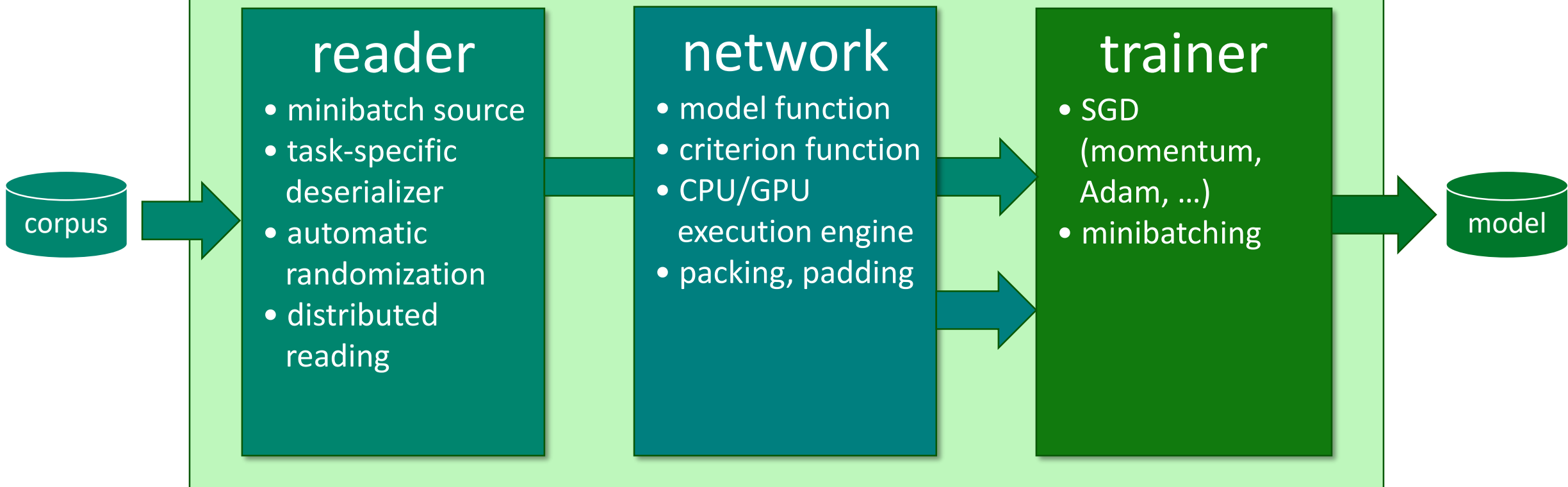


authoring networks as functions

- **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

Microsoft Cognitive Toolkit, CNTK

Script configure and executes through CNTK Python APIs...



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):
    ...

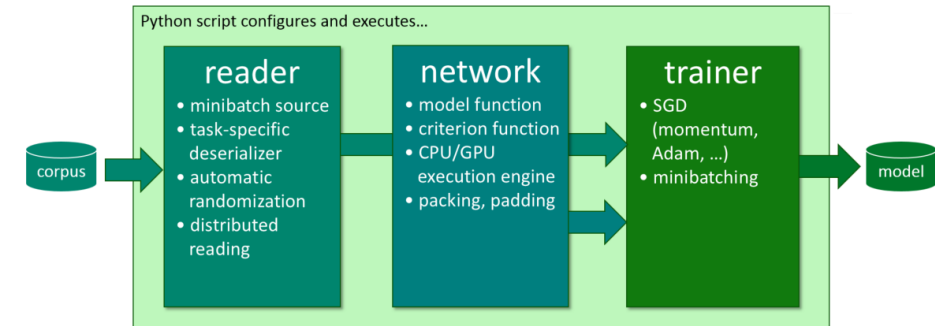
# trainer (and evaluator)
def train(reader, model):
    ...
def evaluate(reader, model):
    ...

# main function
model = create_model_function()

reader = create_reader(..., is_training=True)
train(reader, model)

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

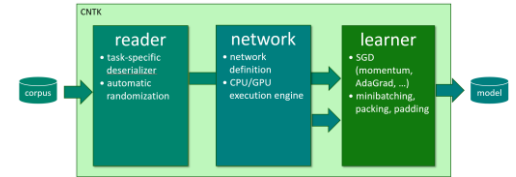
```



workflow

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

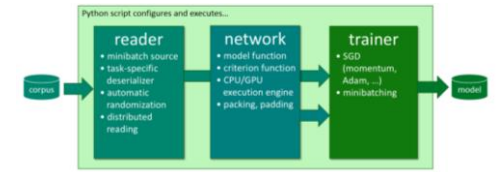
```
python my_cntk_script.py
```



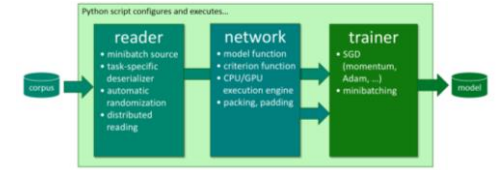
how to: reader

```
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)
```



how to: reader



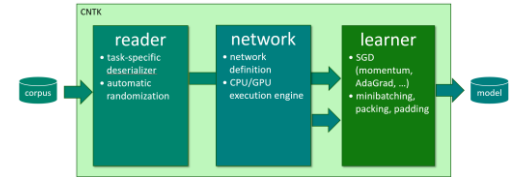
```
def create_reader(map_file, mean_file, is_training):  
    # image preprocessing pipeline  
    transforms = [  
        ImageDeserializer.crop(crop_type='Random', ratio=0.8, jitter_type='uniRatio')  
        ImageDeserializer.scale(width=image_width, height=image_height, channels=num_channels,  
                                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

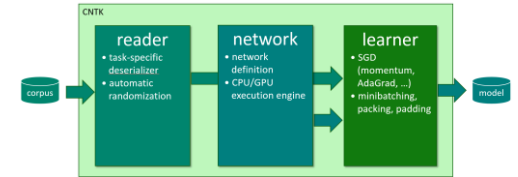
workflow

- 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

CNTK performs!



TABLE 7. COMPARATIVE EXPERIMENT RESULTS (TIME PER MINI-BATCH IN SECOND)

		Desktop CPU (Threads used)				Server CPU (Threads used)						Single GPU		
		1	2	4	8	1	2	4	8	16	32	G980	G1080	K80
FCN-S	Caffe	1.324	0.790	0.578	-	1.355	0.997	0.745	0.573	0.608	1.130	0.041	0.030	0.071
	CNTK	1.227	0.660	0.435	-	1.340	0.909	0.634	0.488	0.441	1.000	0.045	0.033	0.074
	TF	7.062	4.789	2.648	1.938	9.571	6.569	3.399	1.710	0.946	0.630	0.060	0.048	0.109
	MXNet	4.621	2.607	2.162	1.831	5.824	3.356	2.395	2.040	1.945	2.670	-	0.106	0.216
	Torch	1.329	0.710	0.423	-	1.279	1.131	0.595	0.433	0.382	1.034	0.040	0.031	0.070
AlexNet-S	Caffe	1.606	0.999	0.719	-	1.533	1.045	0.797	0.850	0.903	1.124	0.034	0.021	0.073
	CNTK	3.761	1.974	1.276	-	3.852	2.600	1.567	1.347	1.168	1.579	0.045	0.032	0.091
	TF	6.525	2.936	1.749	1.535	5.741	4.216	2.202	1.160	0.701	0.962	0.059	0.042	0.130
	MXNet	2.977	2.340	2.250	2.163	3.518	3.203	2.926	2.828	2.827	2.887	0.020	0.014	0.042
	Torch	4.645	2.429	1.424	-	4.336	2.468	1.543	1.248	1.090	1.214	0.033	0.023	0.070
RenNet-50	Caffe	11.554	7.671	5.652	-	10.643	8.600	6.723	6.019	6.654	8.220	-	0.254	0.766
	CNTK	-	-	-	-	-	-	-	-	-	-	0.240	0.168	0.638
	TF	23.905	16.435	10.206	7.816	29.960	21.846	11.512	6.294	4.130	4.351	0.327	0.227	0.702
	MXNet	14.035	16.053	16.162	15.000	17.910	19.277	19.123	18.898	19.048	19.355	0.059	0.041	0.132
	Torch	13.178	7.500	4.736	4.948	12.807	8.391	5.471	4.164	3.683	4.422	0.208	0.144	0.523
FCN-R	Caffe	2.476	1.499	1.149	-	2.282	1.748	1.403	1.211	1.127	1.127	0.025	0.017	0.055
	CNTK	1.845	0.970	0.661	0.571	1.592	0.857	0.501	0.323	0.252	0.280	0.025	0.017	0.053
	TF	2.647	1.913	1.157	0.919	3.410	2.541	1.297	0.661	0.361	0.325	0.033	0.020	0.063
	MXNet	1.914	1.072	0.719	0.702	1.609	1.065	0.731	0.534	0.451	0.447	0.029	0.019	0.060
	Torch	1.670	0.926	0.565	0.611	1.379	0.915	0.662	0.440	0.402	0.366	0.025	0.016	0.051
AlexNet-R	Caffe	3.558	2.587	2.157	2.963	4.270	3.514	3.381	3.364	4.139	4.930	0.041	0.027	0.137
	CNTK	9.956	7.263	5.519	6.015	9.381	6.078	4.984	4.765	6.256	6.199	0.045	0.031	0.108
	TF	4.535	3.225	1.911	1.565	6.124	4.229	2.200	1.396	1.036	0.971	0.227	0.317	0.385
	MXNet	13.401	12.305	12.278	11.950	17.994	17.128	16.764	16.471	17.471	17.770	0.060	0.032	0.122
	Torch	5.352	3.866	3.162	3.259	6.554	5.288	4.365	3.940	4.157	4.165	0.069	0.043	0.141
RenNet-56	Caffe	6.741	5.451	4.989	6.691	7.513	6.119	6.232	6.689	7.313	9.302	-	0.116	0.378
	CNTK	-	-	-	-	-	-	-	-	-	-	0.206	0.138	0.562
	TF	-	-	-	-	-	-	-	-	-	-	0.225	0.152	0.523
	MXNet	34.409	31.255	30.069	31.388	44.878	43.775	42.299	42.965	43.854	44.367	0.105	0.074	0.270
	Torch	5.758	3.222	2.368	2.475	8.691	4.965	3.040	2.560	2.575	2.811	0.150	0.101	0.301
LSTM	Caffe	-	-	-	-	-	-	-	-	-	-	-	-	-
	CNTK	0.186	0.120	0.090	0.118	0.211	0.139	0.117	0.114	0.114	0.198	0.018	0.017	0.043
	TF	4.662	3.385	1.935	1.532	6.449	4.351	2.238	1.183	0.702	0.598	0.133	0.065	0.140
	MXNet	-	-	-	-	-	-	-	-	-	-	0.089	0.079	0.149
	Torch	6.921	3.831	2.682	3.127	7.471	4.641	3.580	3.260	5.148	5.851	0.399	0.324	0.560

["Benchmarking State-of-the-Art Deep Learning Software Tools," HKBU, <https://arxiv.org/pdf/1608.07249v6.pdf>]

Layers API

- **basic blocks:**
 - LSTM(), GRU(), RNNUnit()
 - Stabilizer(), identity
 - ForwardDeclaration(), Tensor[], SparseTensor[], Sequence[], SequenceOver[]
- **layers:**
 - Dense(), Embedding()
 - Convolution(), Convolution1D(), Convolution2D(), Convolution3D(), Deconvolution()
 - MaxPooling(), AveragePooling(), GlobalMaxPooling(), GlobalAveragePooling(), MaxUnpooling()
 - BatchNormalization(), LayerNormalization()
 - Dropout(), Activation()
 - Label()
- **composition:**
 - Sequential(), For(), operator >>, (function tuples)
 - ResNetBlock(), SequentialClique()
- **sequences:**
 - Delay(), PastValueWindow()
 - Recurrence(), RecurrenceFrom(), Fold(), UnfoldFrom()
- **models:**
 - AttentionModel()



Search docs

Setup

Getting Started

Working with Sequences

Tutorials

Examples

Layers Library Reference

General patterns

Specifying the same options to multiple layers

Weight sharing

Example models

Dense()

Convolution()

MaxPooling(), AveragePooling()

[Docs](#) » Layers Library Reference

[View page source](#)

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 `()` to pass construction parameters or attributes.

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

```
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:



Distinguishing Features

Differentiating features

- higher-level features:
 - auto-tuning of learning rate and minibatch size
 - memory sharing
 - implicit handling of time
 - minibatching of variable-length sequences
 - data-parallel training
- you can do all this with other toolkits, but must write it yourself

deep dive: handling of time

extend our example to a recurrent network (RNN)

$$h_1 = \sigma(W_1 x + b_1)$$

$$h_2 = \sigma(W_2 h_1 + b_2)$$

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

$$ce = L^T \log P$$

$$\sum_{\text{corpus}} ce = \max$$



deep dive: handling of time

extend our example to a recurrent network (RNN)

$$h_1(t) = \sigma(W_1 x(t) + b_1)$$

$$h_2(t) = \sigma(W_2 h_1(t) + b_2)$$

$$P(t) = \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}})$$

$$ce(t) = L^T(t) \log P(t)$$

$$\sum_{\text{corpus}} ce(t) = \max$$



deep dive: handling of time

extend our example to a recurrent network (RNN)

$$h_1(t) = \sigma(W_1 x(t) + H_1 h_1(t-1) + b_1)$$

$$h_2(t) = \sigma(W_2 h_1(t) + H_2 h_2(t-1) + b_2)$$

$$P(t) = \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}})$$

$$ce(t) = L^T(t) \log P(t)$$

$$\sum_{\text{corpus}} ce(t) = \max$$

deep dive: handling of time

extend our example to a recurrent network (RNN)

$$h_1(t) = \sigma(W_1 x(t) + H_1 h_1(t-1) + b_1)$$

$$h1 = \text{sigmoid}(x @ W1 + \text{past_value}(h1) @ H1 + b1)$$

$$h_2(t) = \sigma(W_2 h_1(t) + H_2 h_2(t-1) + b_2)$$

$$h2 = \text{sigmoid}(h1 @ W2 + \text{past_value}(h2) @ H2 + b2)$$

$$P(t) = \text{softmax}(W_{\text{out}} h_2(t) + b_{\text{out}})$$

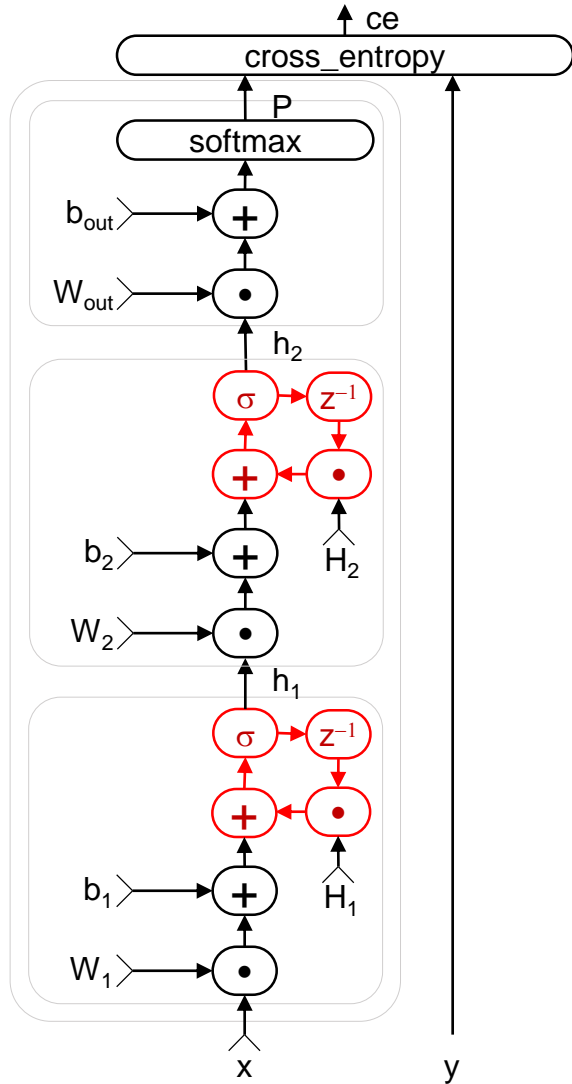
$$P = \text{softmax}(h2 @ W_{\text{out}} + b_{\text{out}})$$

$$ce(t) = L^T(t) \log P(t)$$

$$ce = \text{cross_entropy}(P, L)$$

$$\sum_{\text{corpus}} ce(t) = \max$$

deep dive: handling of time



$$h_1 = \text{sigmoid}(x @ W_1 + \text{past_value}(h_1) @ H_1 + b_1)$$

$$h_2 = \text{sigmoid}(h_1 @ W_2 + \text{past_value}(h_2) @ H_2 + b_2)$$

$$P = \text{softmax}(h_2 @ W_{out} + b_{out})$$

$$ce = \text{cross_entropy}(P, L)$$

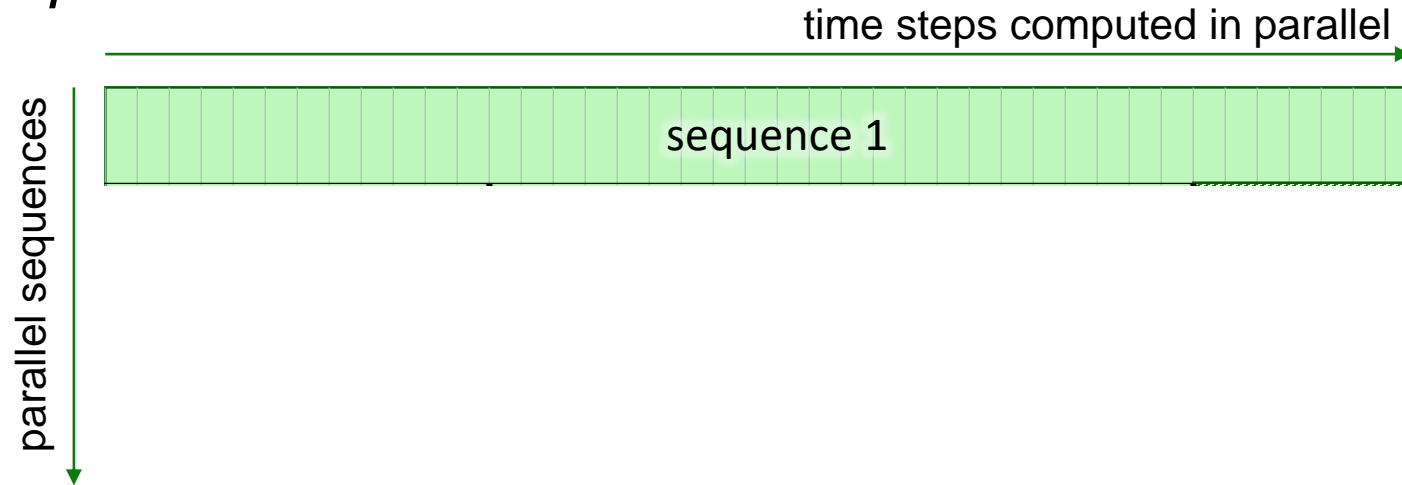
- CNTK automatically unrolls **cycles** \rightarrow deferred computation
- Efficient and composable

deep dive: variable-length sequences

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

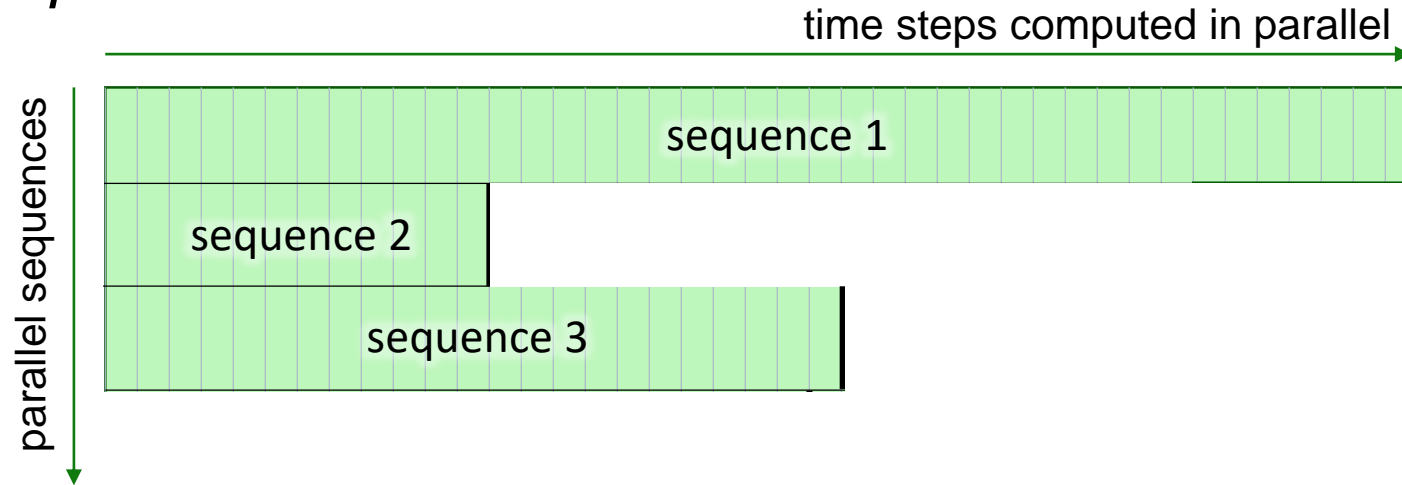
deep dive: variable-length sequences

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



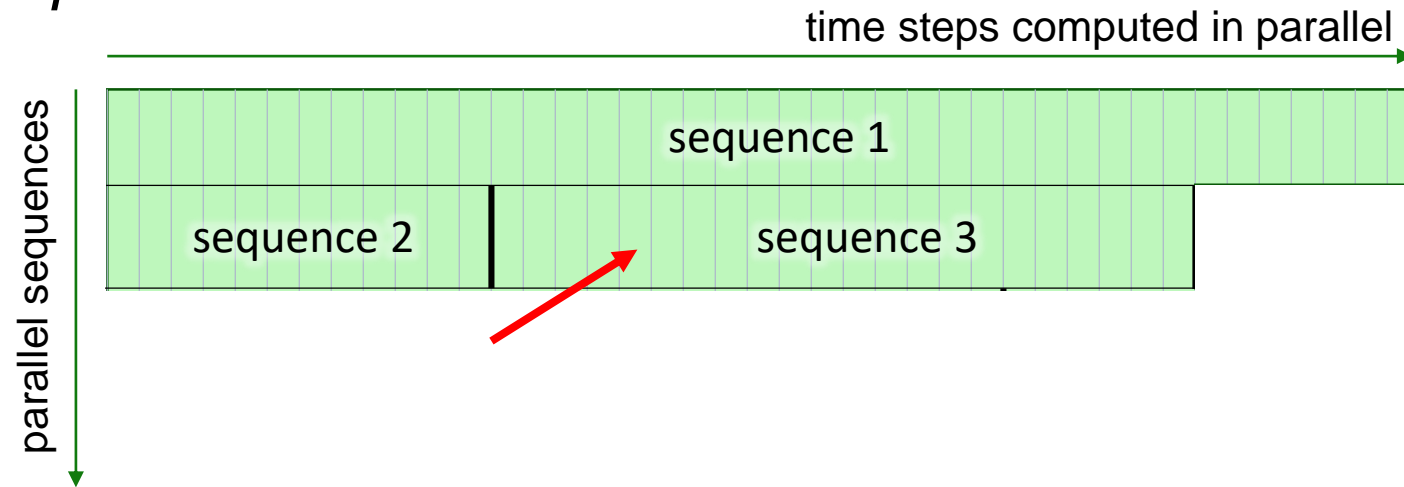
deep dive: variable-length sequences

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



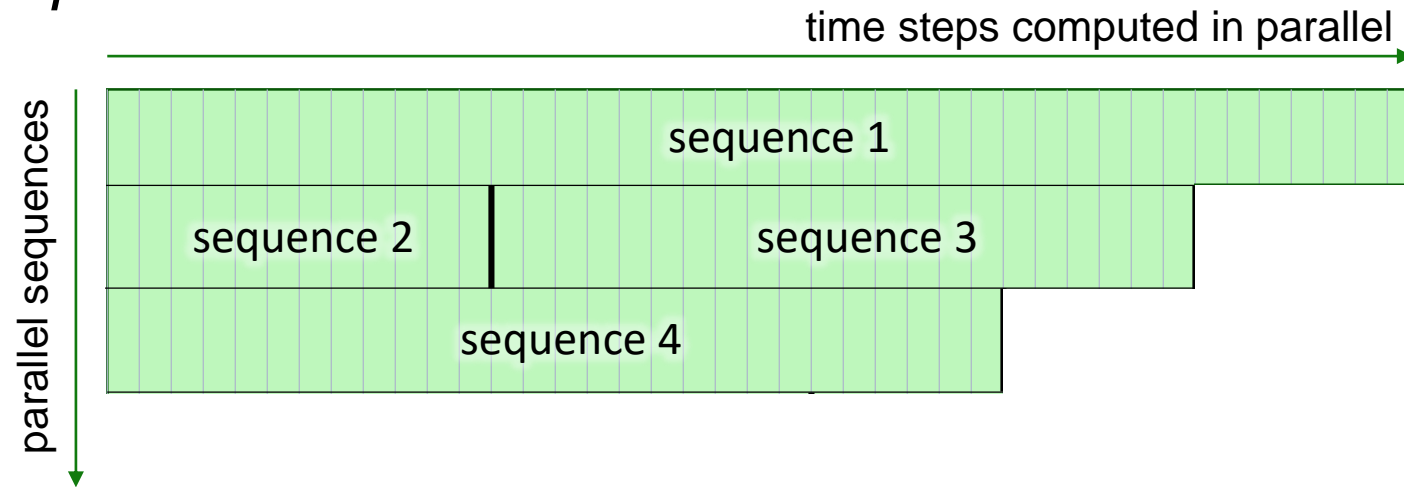
deep dive: variable-length sequences

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



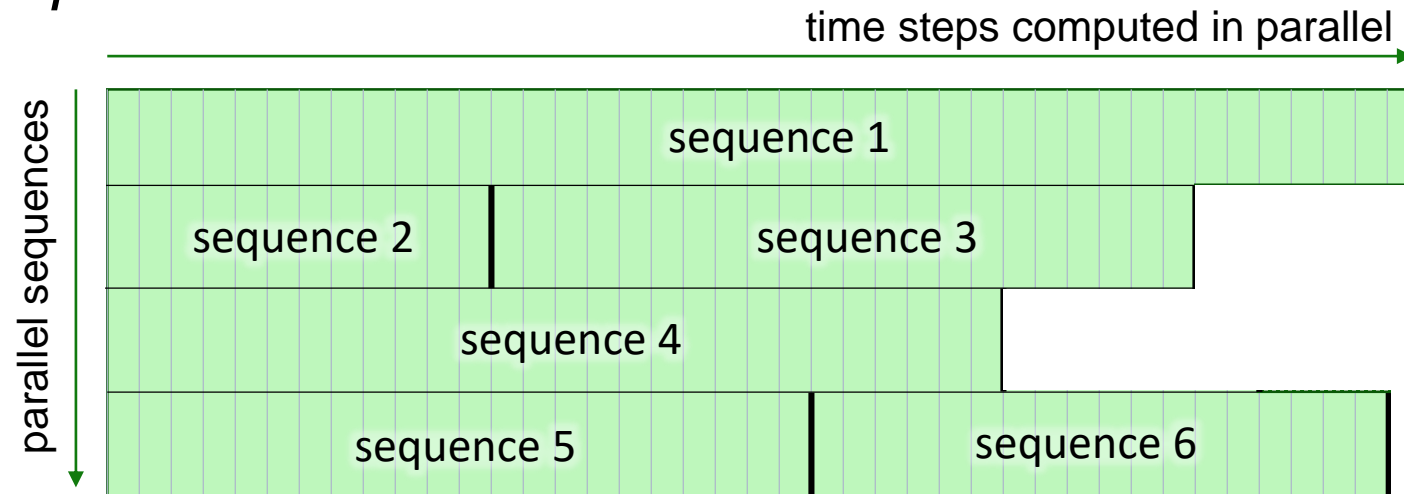
deep dive: variable-length sequences

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



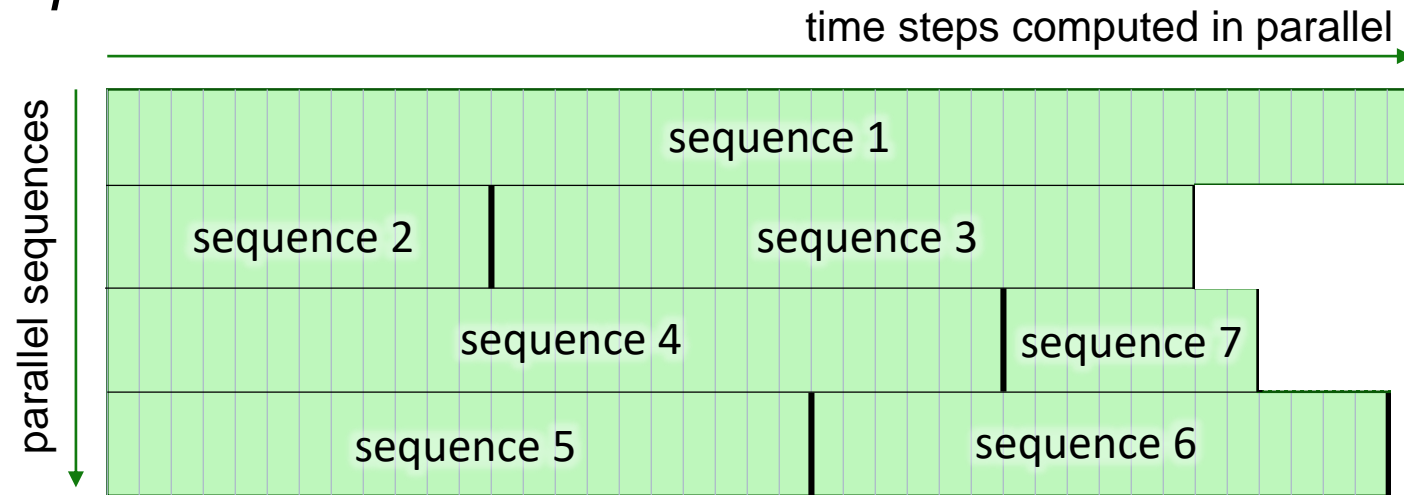
deep dive: variable-length sequences

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



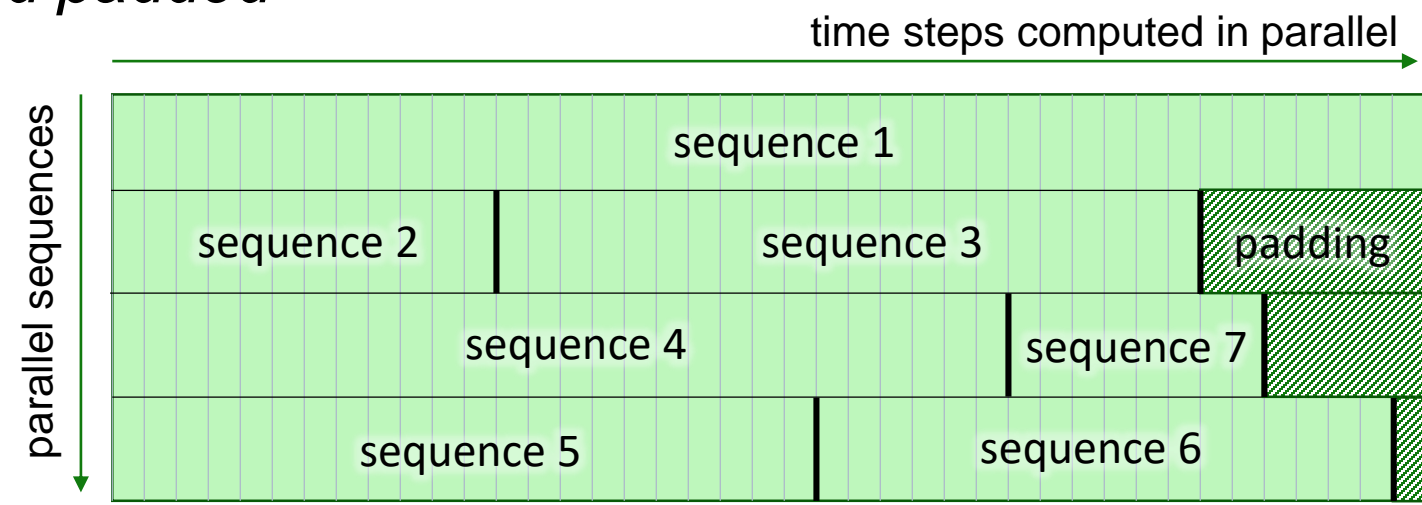
deep dive: variable-length sequences

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



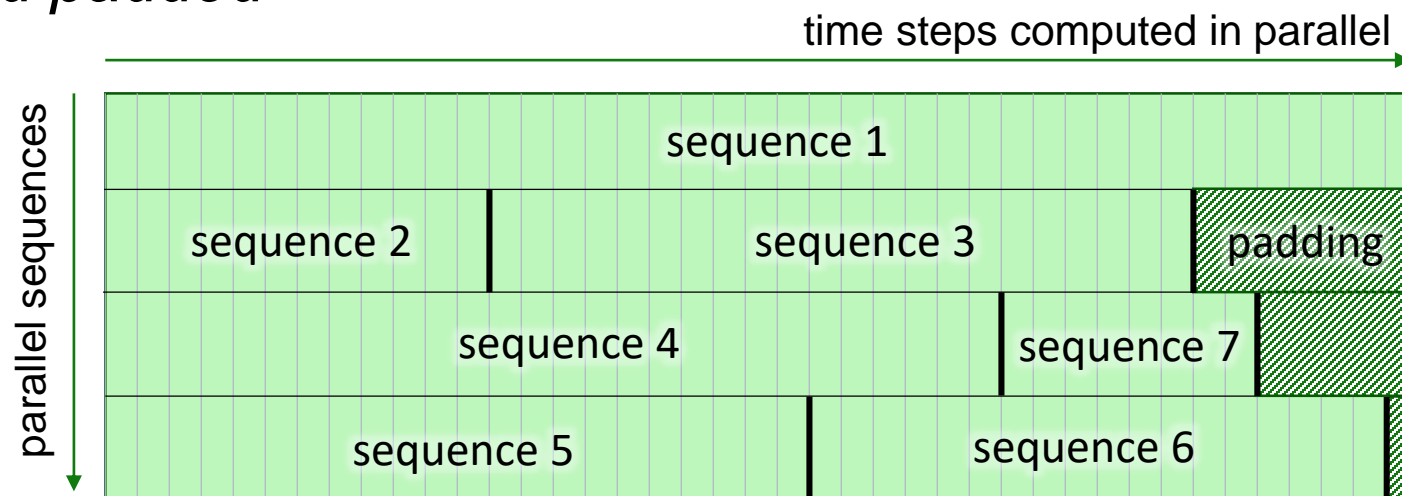
deep dive: variable-length sequences

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



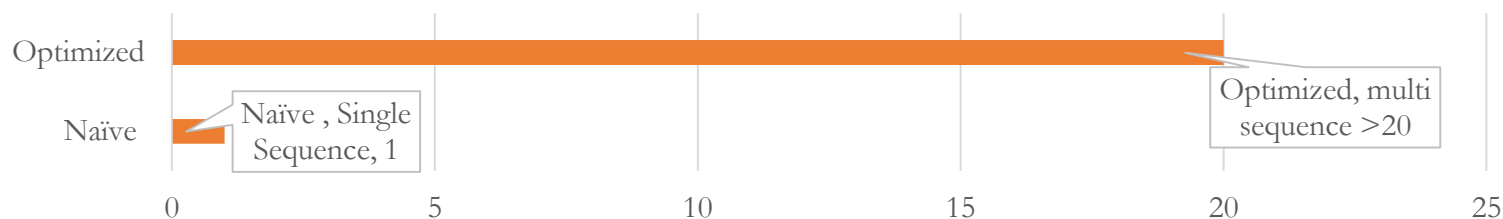
deep dive: variable-length sequences

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



- speed-up is automatic:

Speed comparison on RNNs



Deep dive: data-parallel training

- data-parallelism: distribute each minibatch over workers, then aggregate
- challenge: communication cost
- example: DNN, MB size 1024, 160M model parameters
 - compute per MB: → 1/7 second
 - communication per MB: → 1/9 second (640M over 6 GB/s)
 - can't even parallelize to 2 GPUs: communication cost already dominates!

Deep dive: 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

data-parallel training

how to reduce communication cost:

communicate less each time

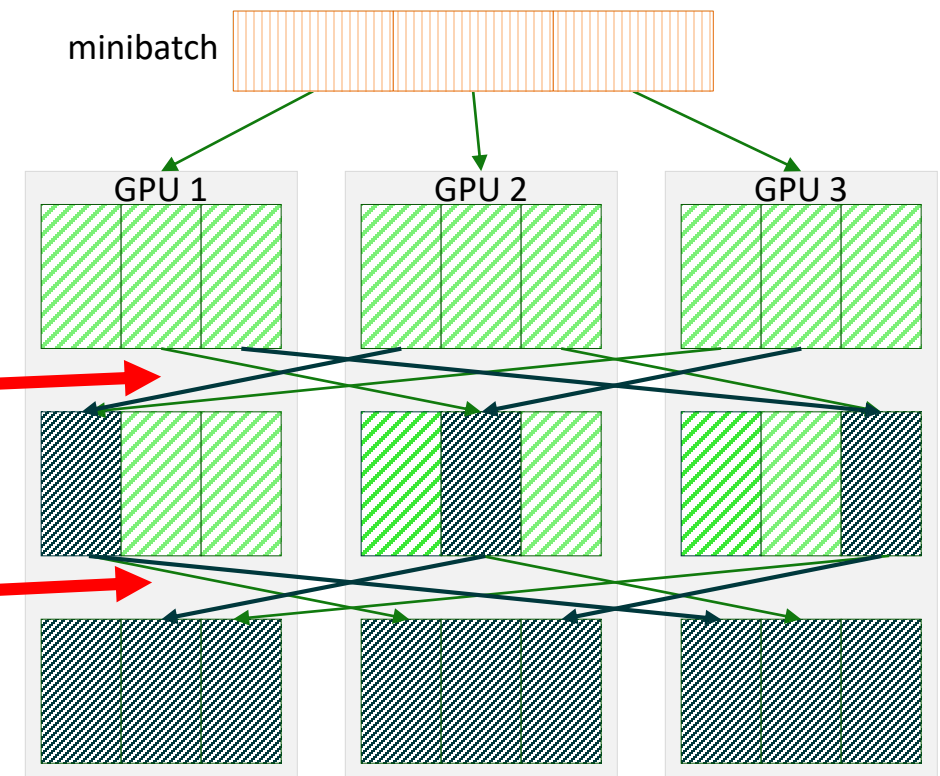
- 1-bit SGD:

[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

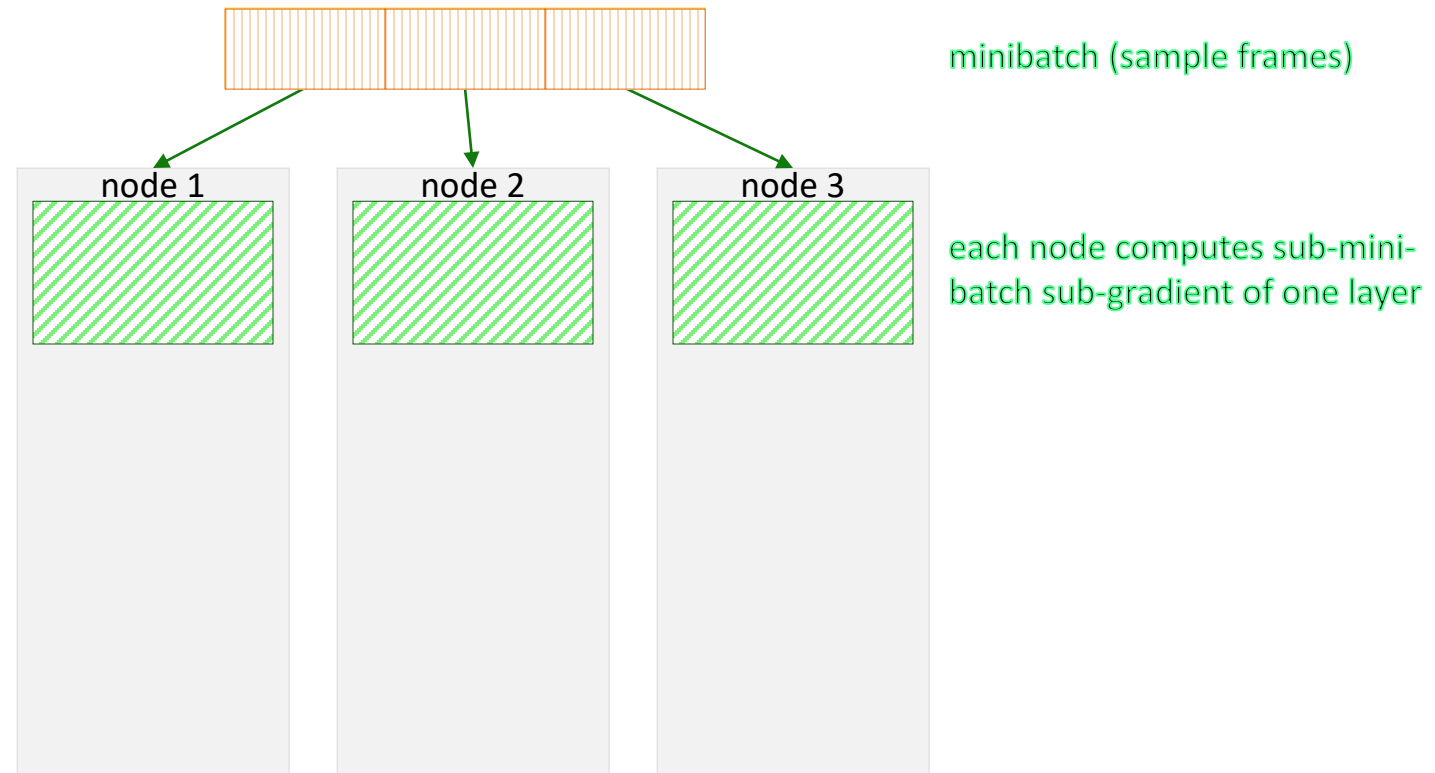
1-bit quantized with residual

1-bit quantized with residual



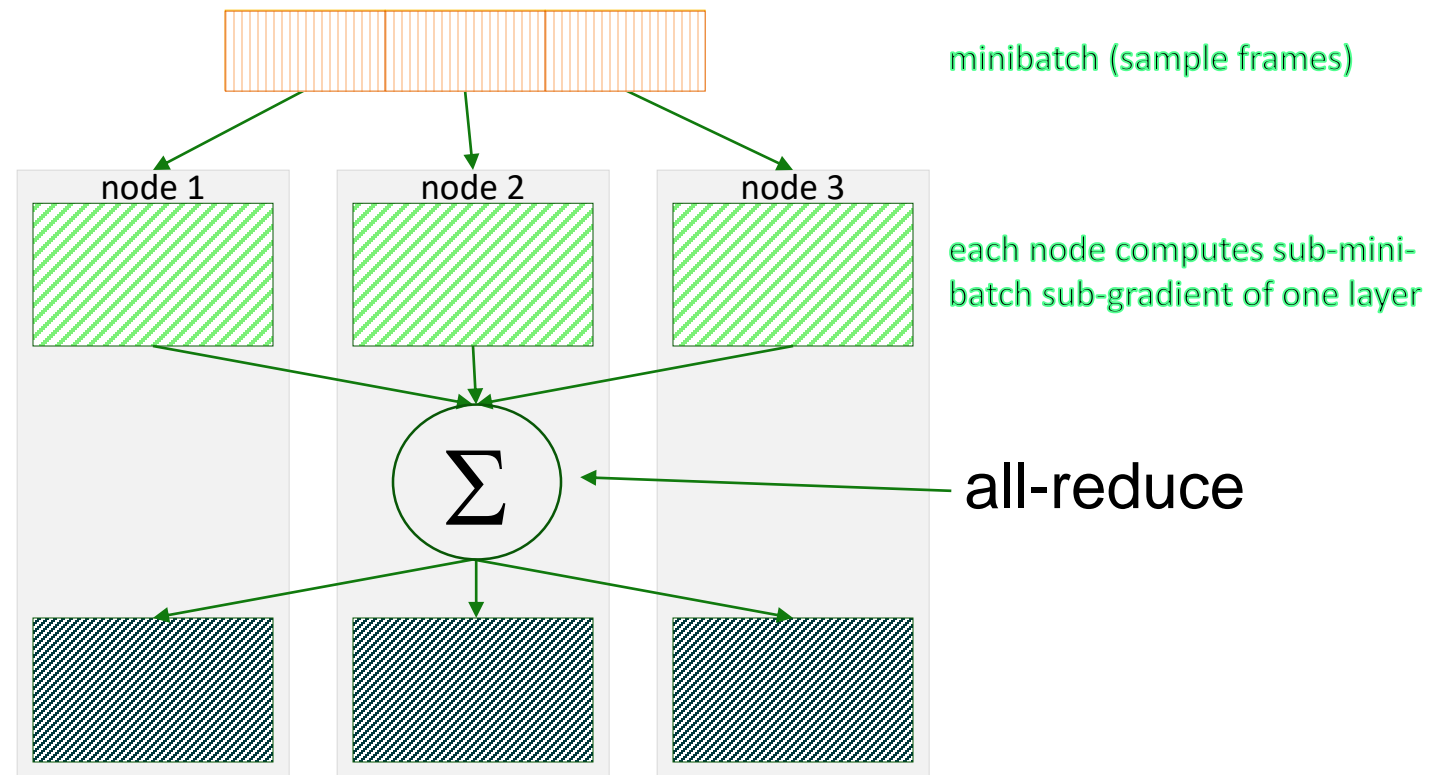
data-parallel training

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



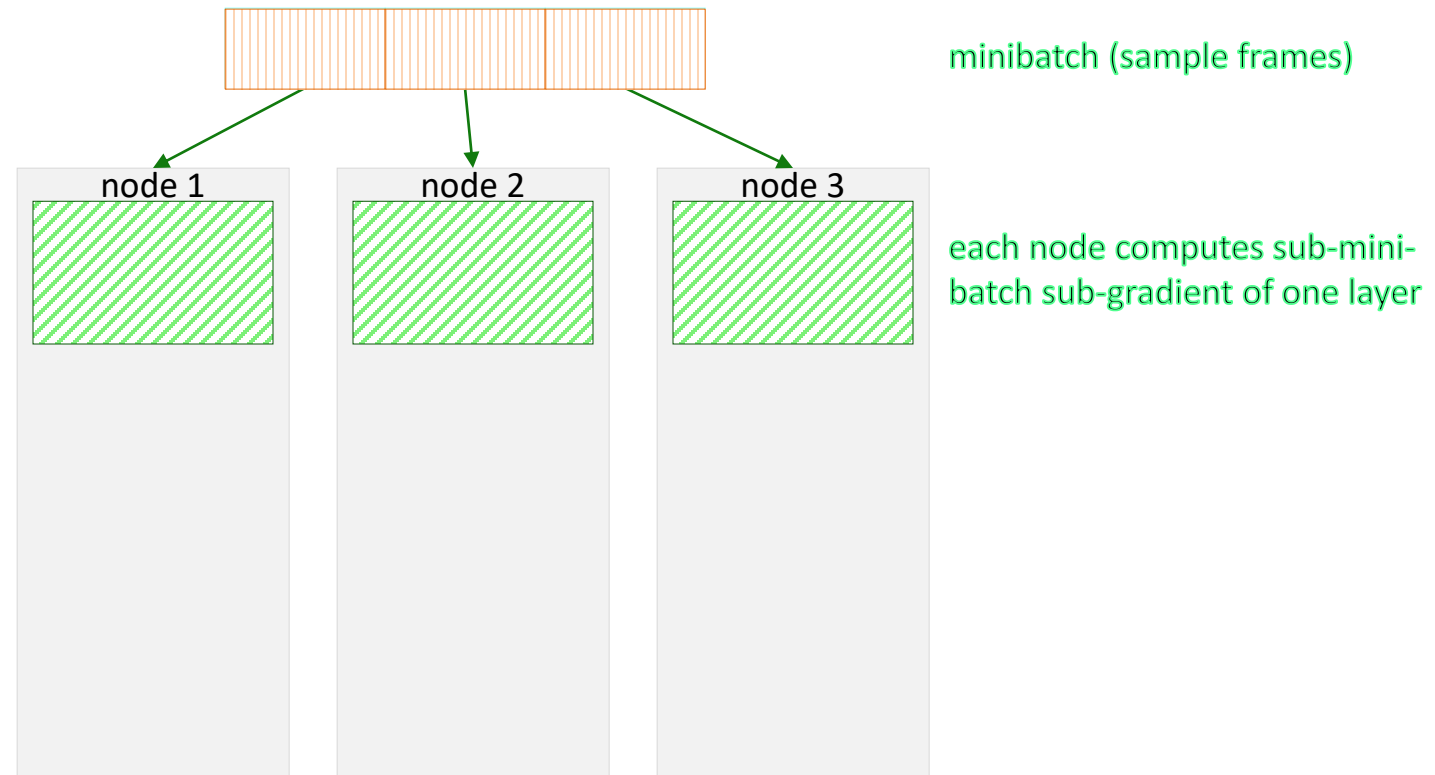
data-parallel training

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



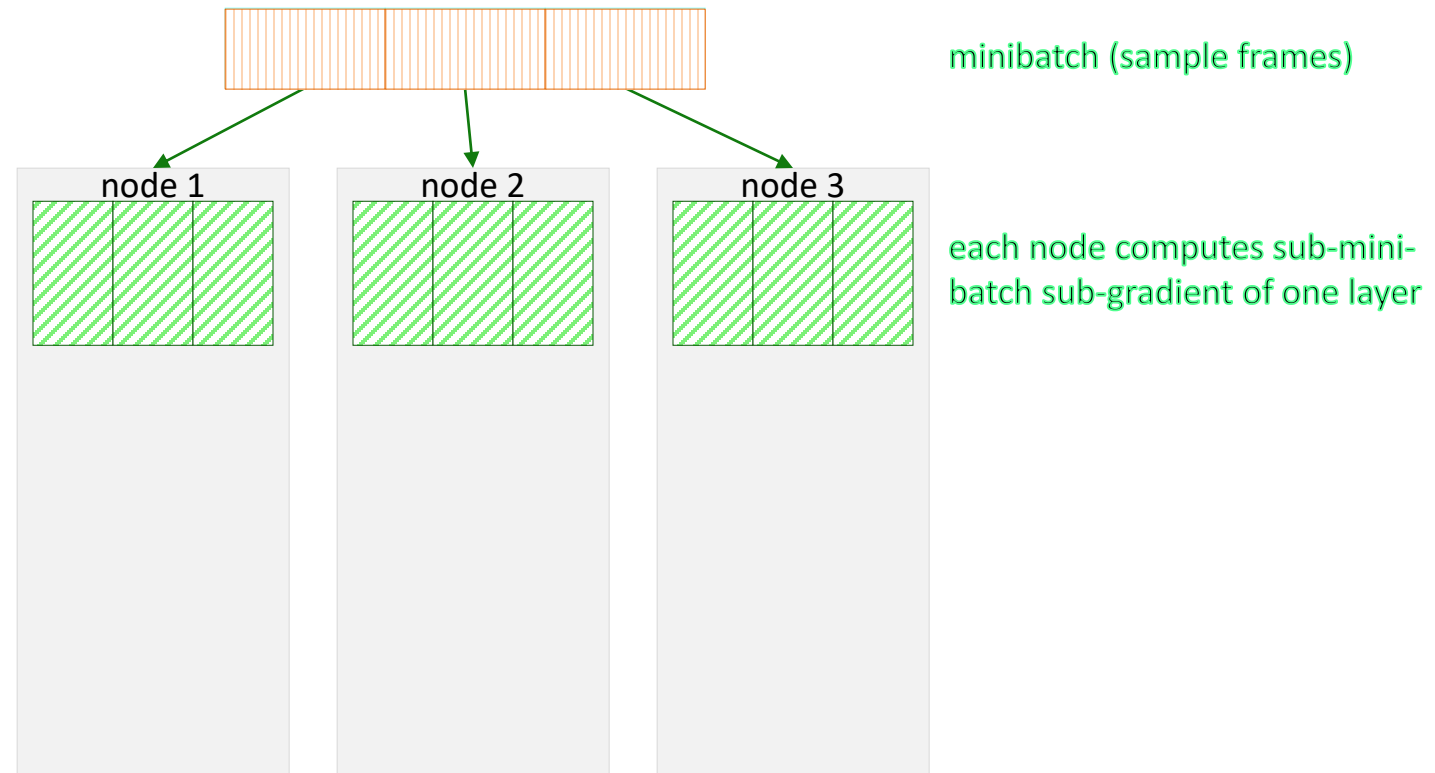
data-parallel training

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



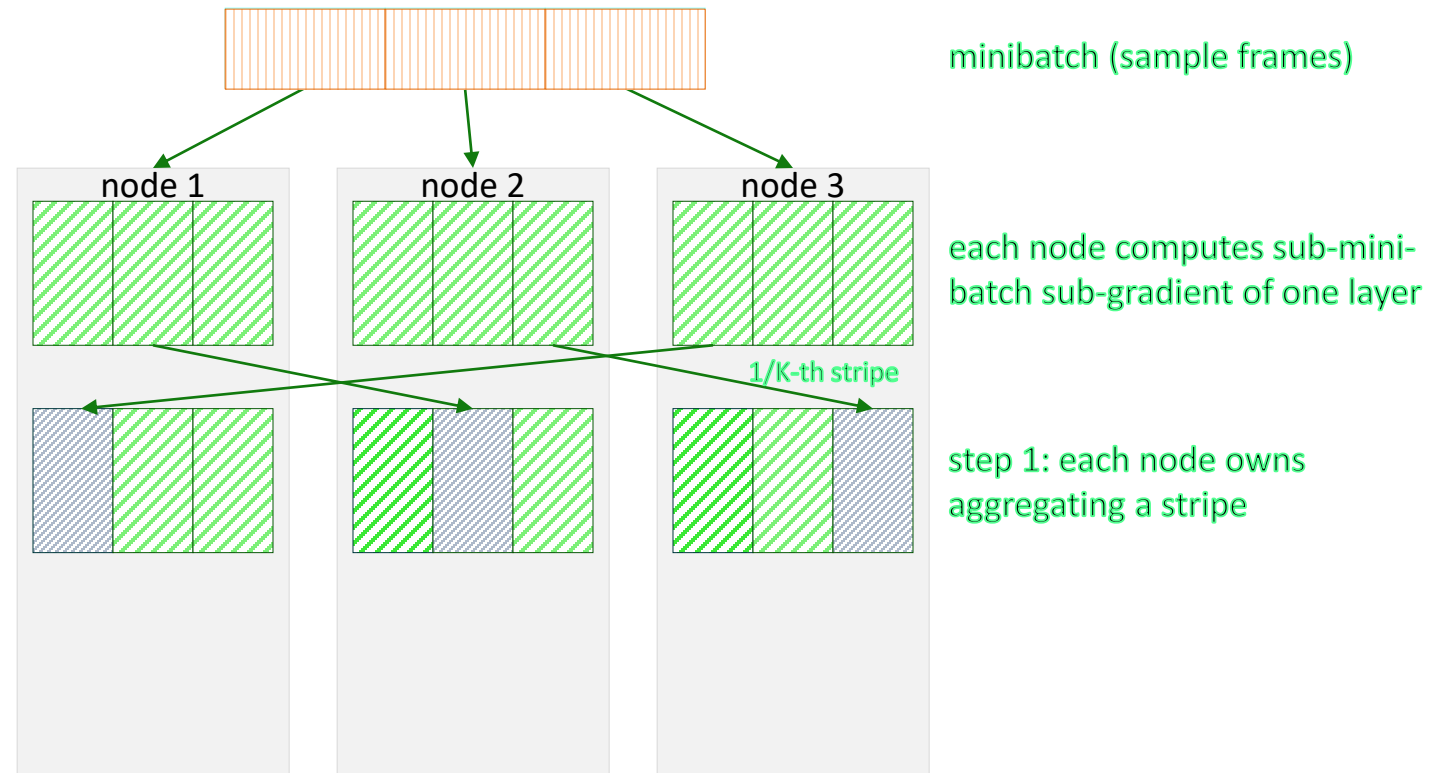
data-parallel training

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



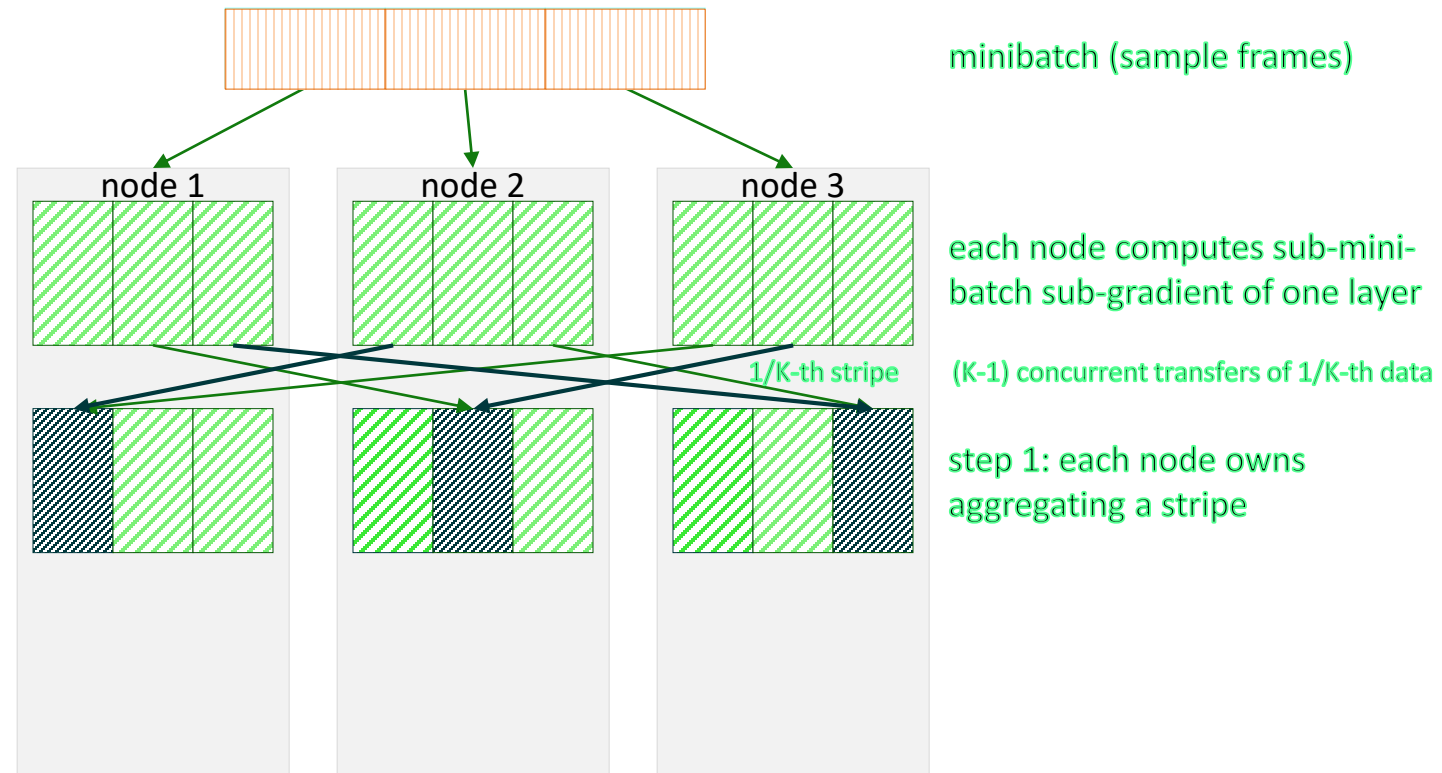
data-parallel training

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



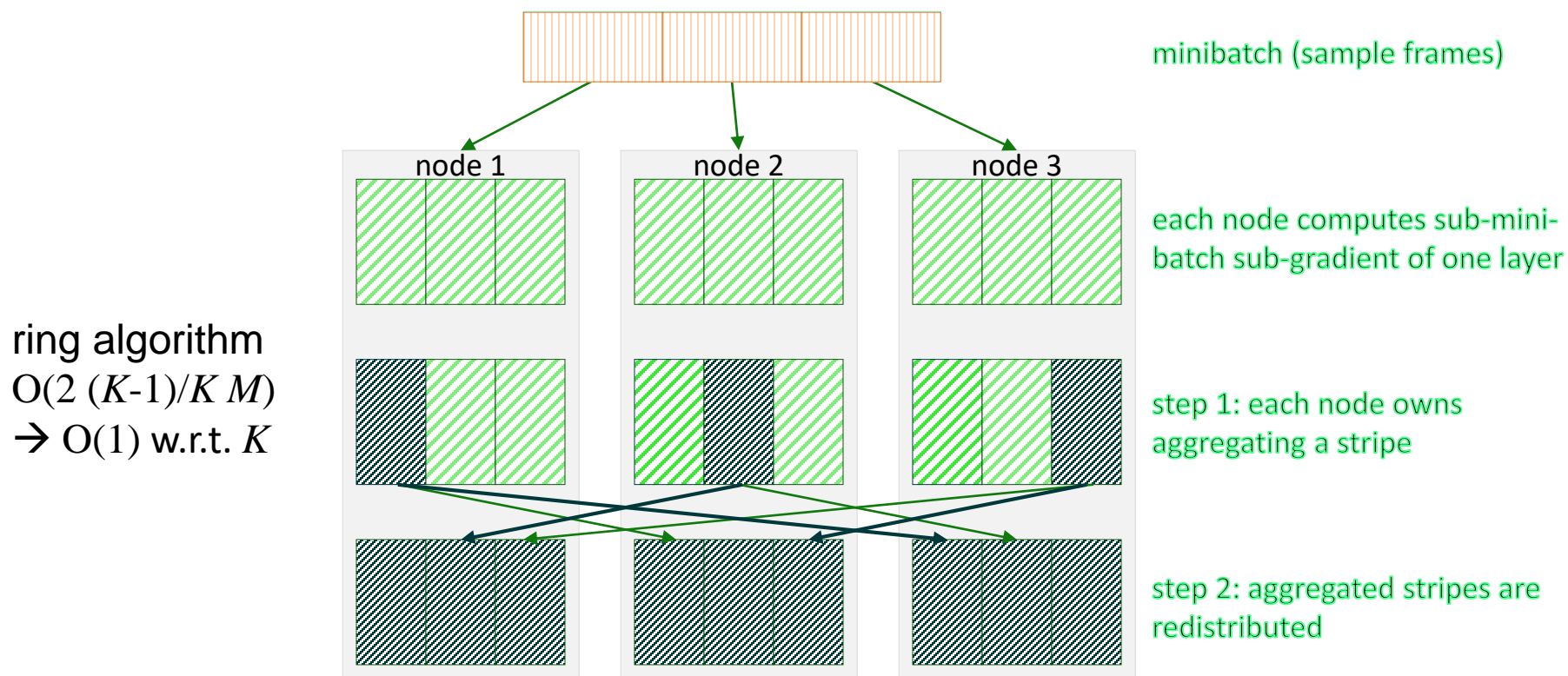
data-parallel training

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



data-parallel training

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



Deep dive: 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

Batch Momentum

- **Incremental Block Training (IBT)**
 - Training dataset is processed block-by-block
- **Intra-Block Parallel Optimization (IBPO)**
 - Master-slave architecture to exploit data parallelism
 - Each worker works independently on a split of data block
 - Local model-updates are **aggregated appropriately**
 - MPI-like framework to coordinate parallel job scheduling and communication
 - Redundant workers to reduce wasted time for synchronization of multiple workers
- **Blockwise Model-Update Filtering (BMUF)**
 - Use **historic model-update information** to guide learning process

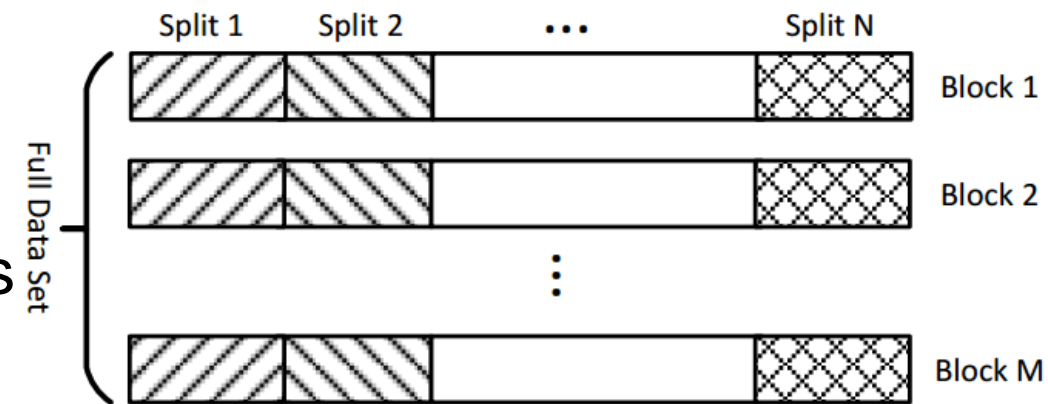
Data Partition

- Partition randomly training dataset \mathcal{D} into S mini-batches

$$\mathcal{D} = \{\mathcal{B}_i | i = 1, 2, \dots, S\}$$

- Group every τ mini-batches to form a split
- Group every N splits to form a data block
- Training dataset \mathcal{D} consists of M data blocks

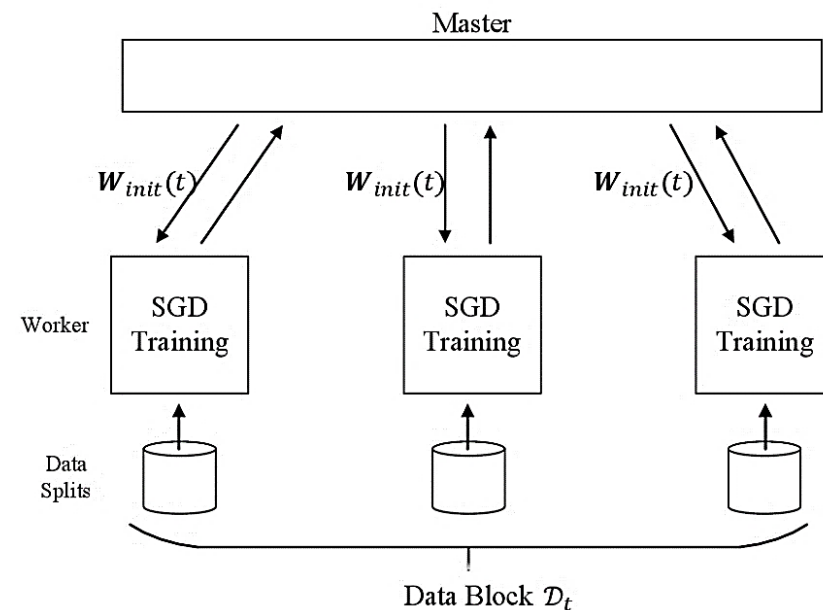
$$S = M \times N \times \tau$$



- Training dataset is processed block-by-block
→ **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 $W_{init}(t)$, each worker optimizes its local model independently by 1-sweep mini-batch SGD with momentum trick
- Average N optimized local models to get $\overline{W}(t)$



Blockwise Model-Update Filtering (BMUF)

- Generate model-update resulting from data block \mathcal{D}_t :

$$\mathbf{G}(t) = \overline{\mathbf{W}}(t) - \mathbf{W}_{init}(t)$$

- Calculate global model-update:

$$\Delta(t) = \eta_t \cdot \Delta(t-1) + \zeta_t \cdot \mathbf{G}(t)$$

- ζ_t : Block Learning Rate (BLR)
- η_t : **Block Momentum** (BM)
- When $\zeta_t = 1$ 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)

$$\mathbf{W}_{init}(t+1) = \mathbf{W}(t) + \eta_{t+1} \cdot \Delta(t)$$

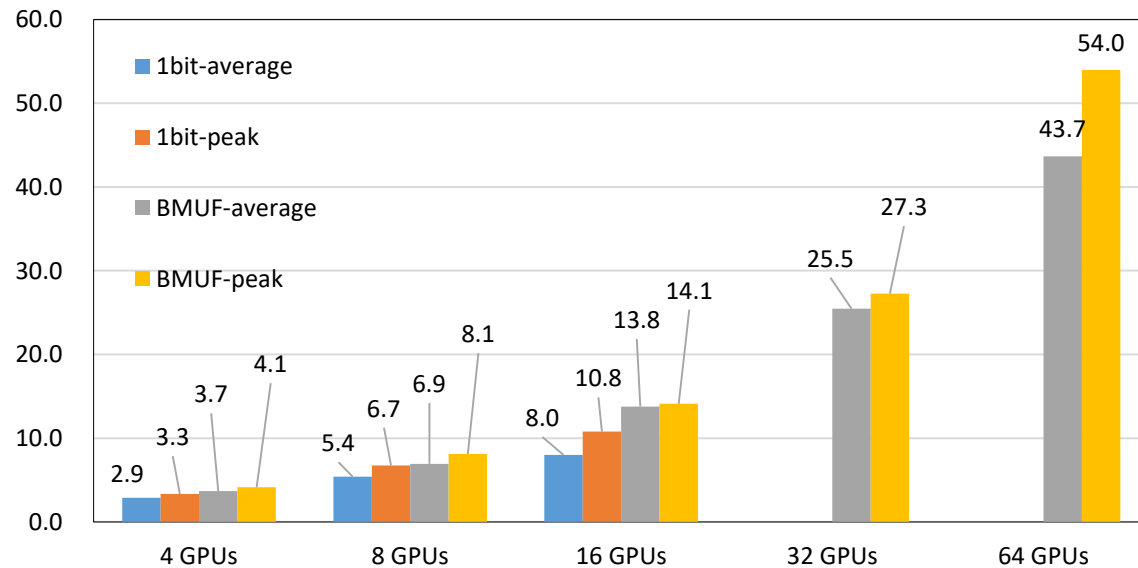
Iteration

- 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}

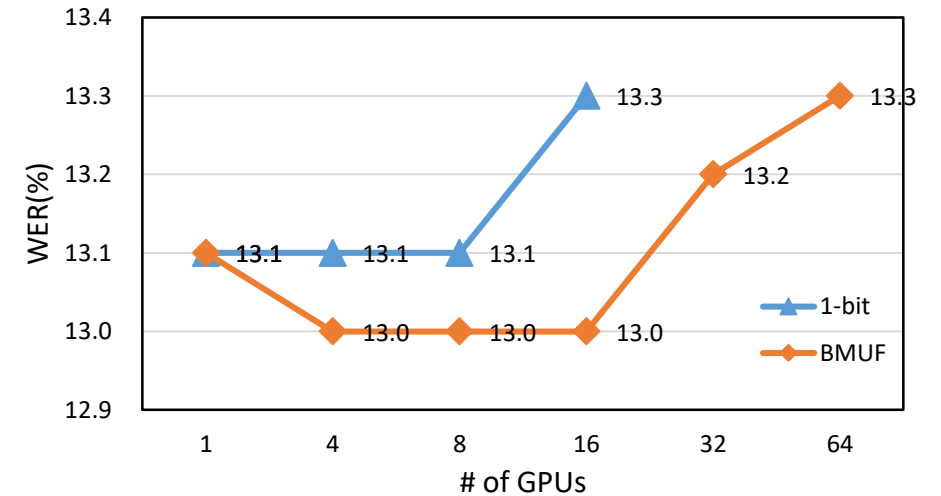
Benchmark Result of Parallel Training on CNTK

- 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

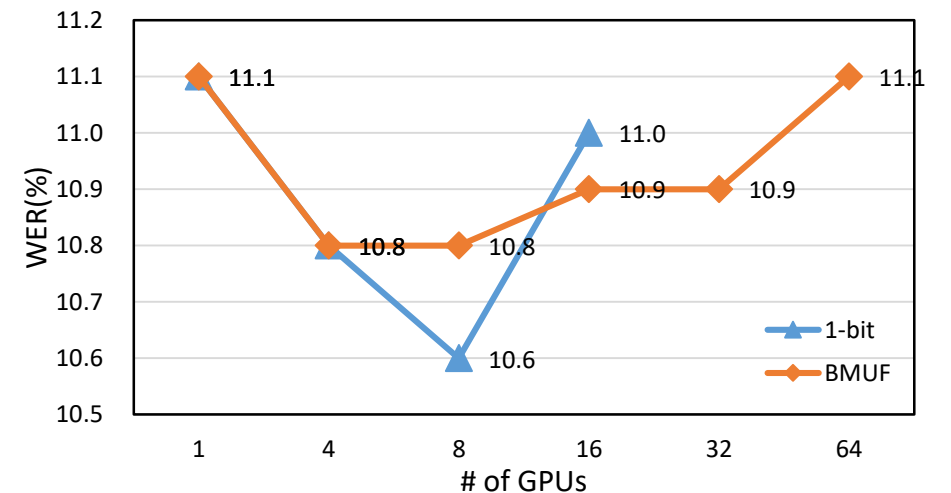
1bit/BMUF Speedup Factors in LSTM Training



WER of CE-trained DNN with different # of GPUs



WER of CE-trained LSTM with different # of GPUs



Impact

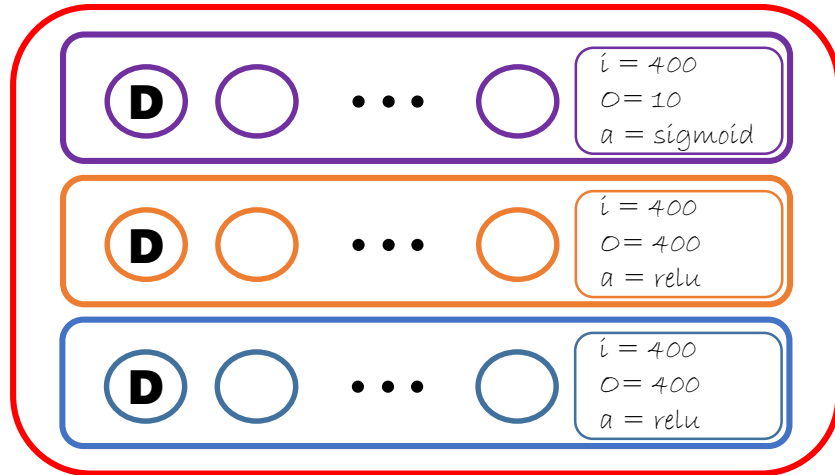
- Achievement
 - Almost linear speedup without degradation of model quality
 - Verified for training DNN, CNN, LSTM up to **64 GPUs** for speech recognition, image classification, OCR, and click prediction tasks
- [Released](#) in [CNTK](#) as a critical differentiator
- Used for enterprise scale production data loads
- Production tools in other companies such as iFLYTEK and Alibaba



Recurrence Primer

Sequences (one to one)

Problem: Optical character recognition of MNIST data

$$\left\{ \begin{array}{cccc} 1 & 5 & 4 & 3 \\ 5 & 3 & 5 & 3 \\ 5 & 9 & 0 & 6 \end{array} \right\}$$


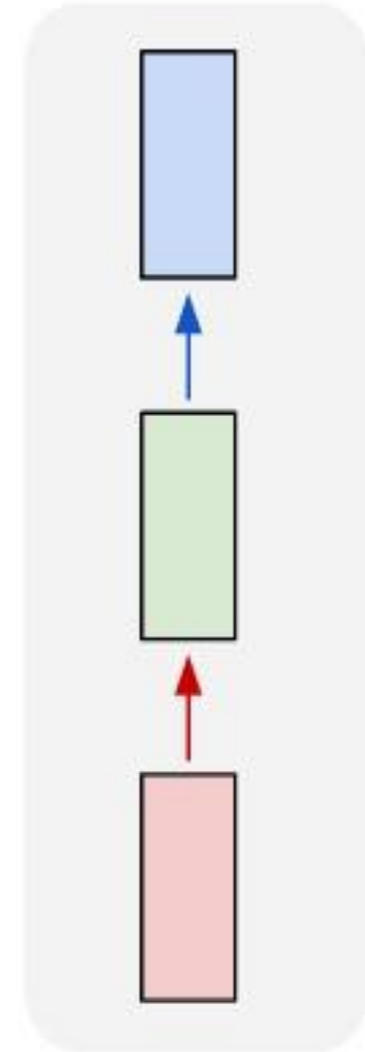
1 5 4 3
5 3 5 3
5 9 0 6

Output Labels
($Y: 12 \times 10$)

Model
Hidden
parameters

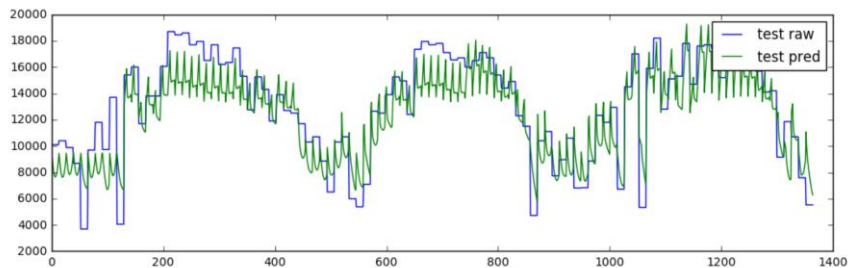
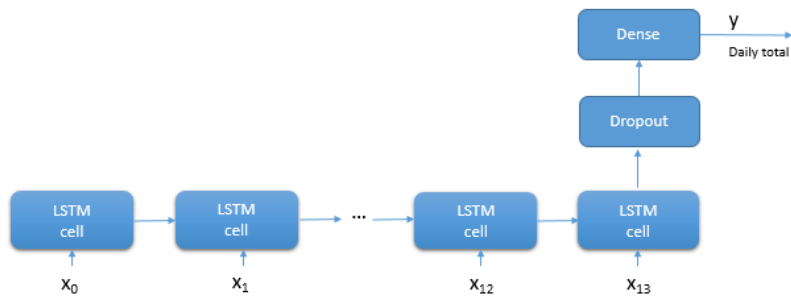
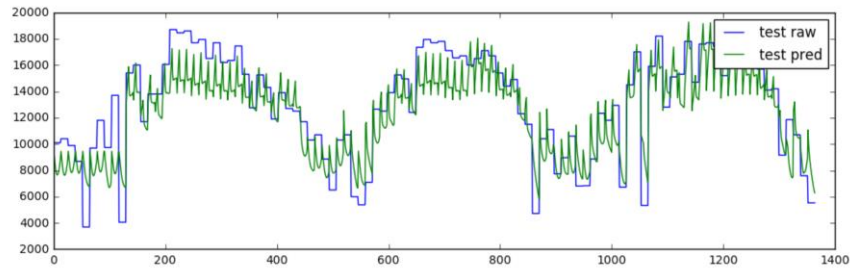
Input feature
($X: 12 \times 784$)

one to one



Sequences (many to one)

Problem: Time series prediction with IOT data

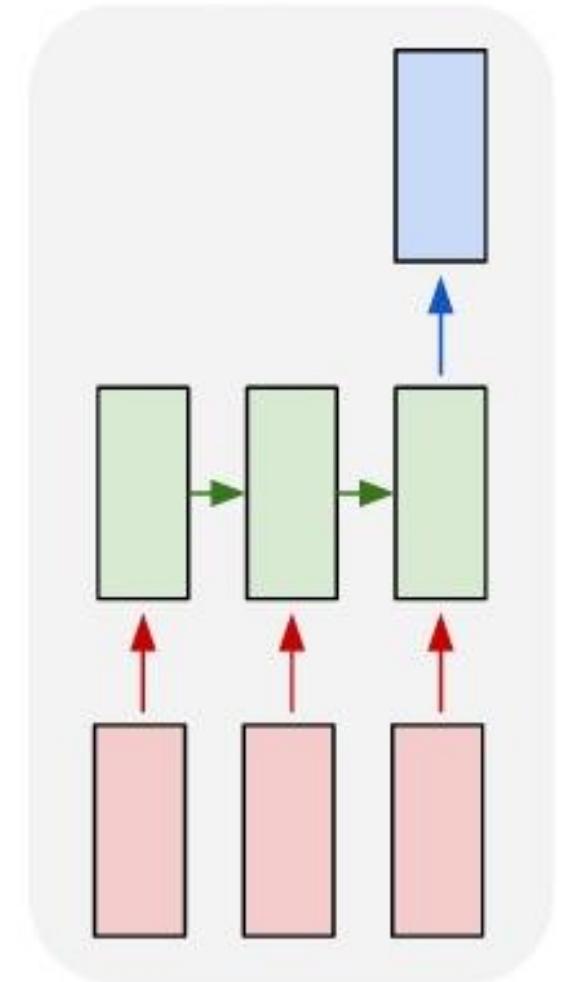


Output Labels
($Y: n \times \text{future prediction}$)

Model

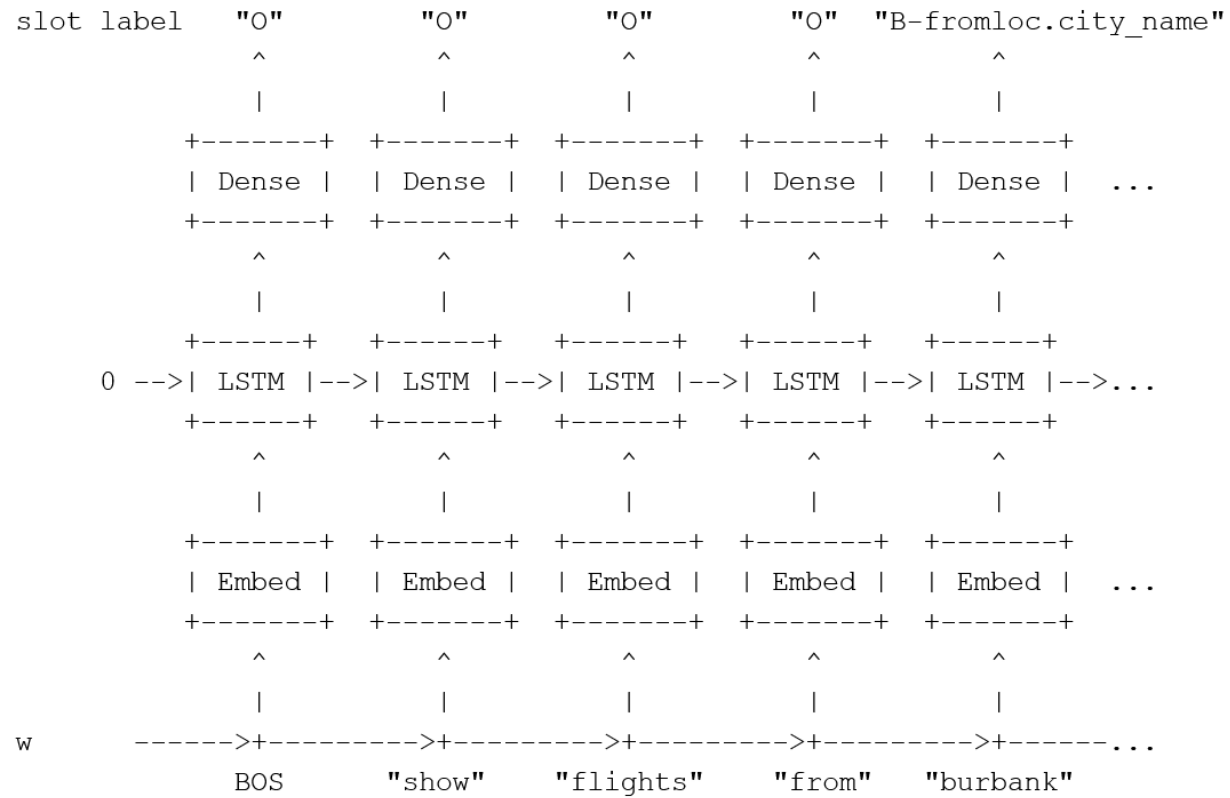
Input feature
($X: n \times 14 \text{ data pnts}$)

many to one

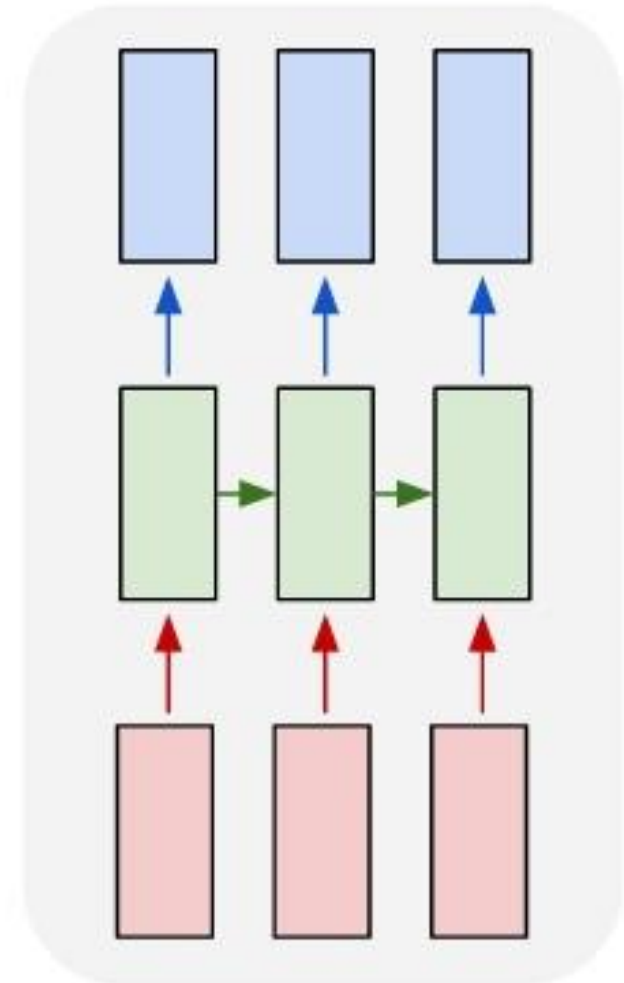


Sequences (many to many)

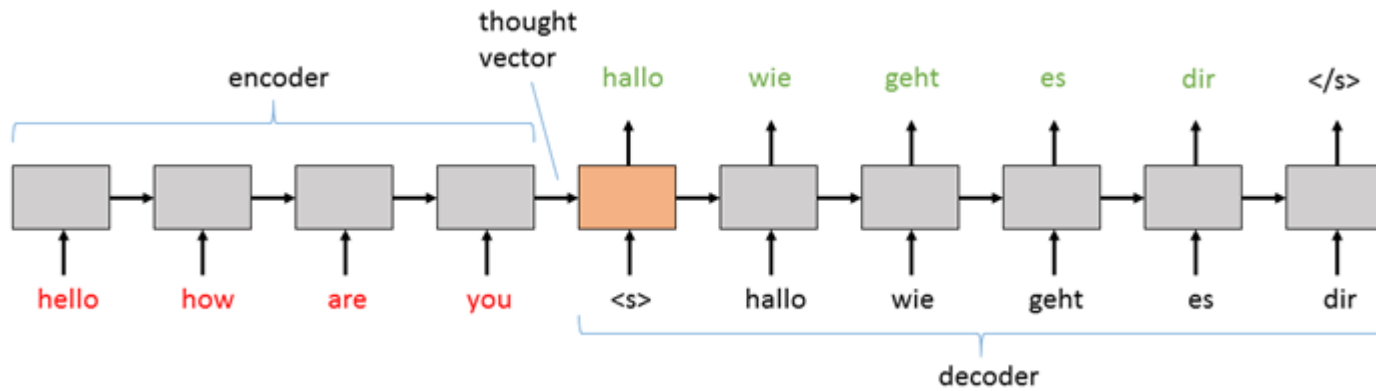
Problem: Tagging entities in Air Traffic Controller (ATIS) data



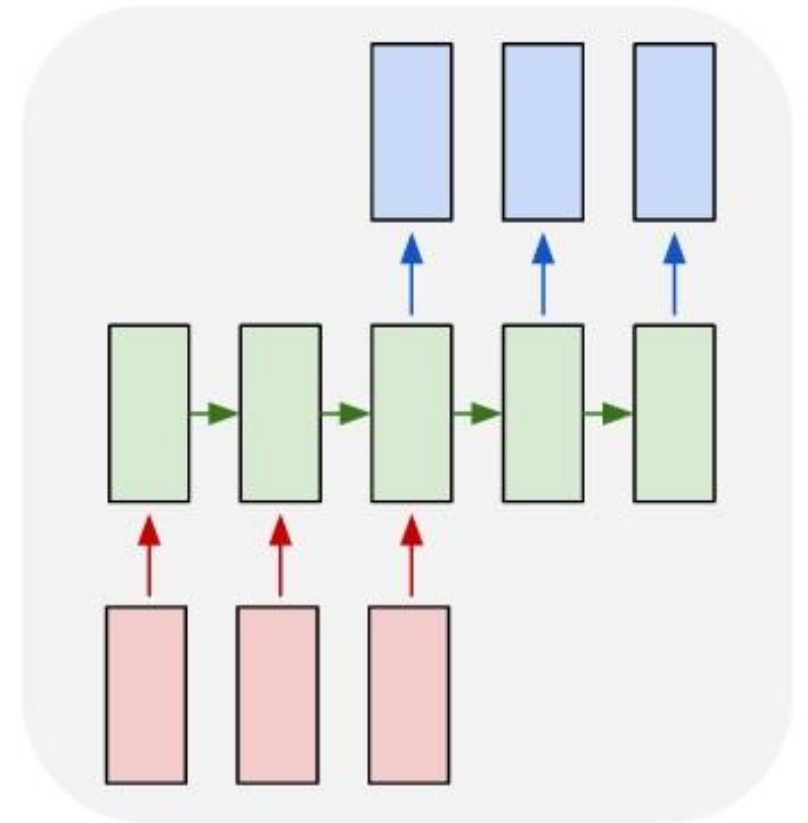
many to many



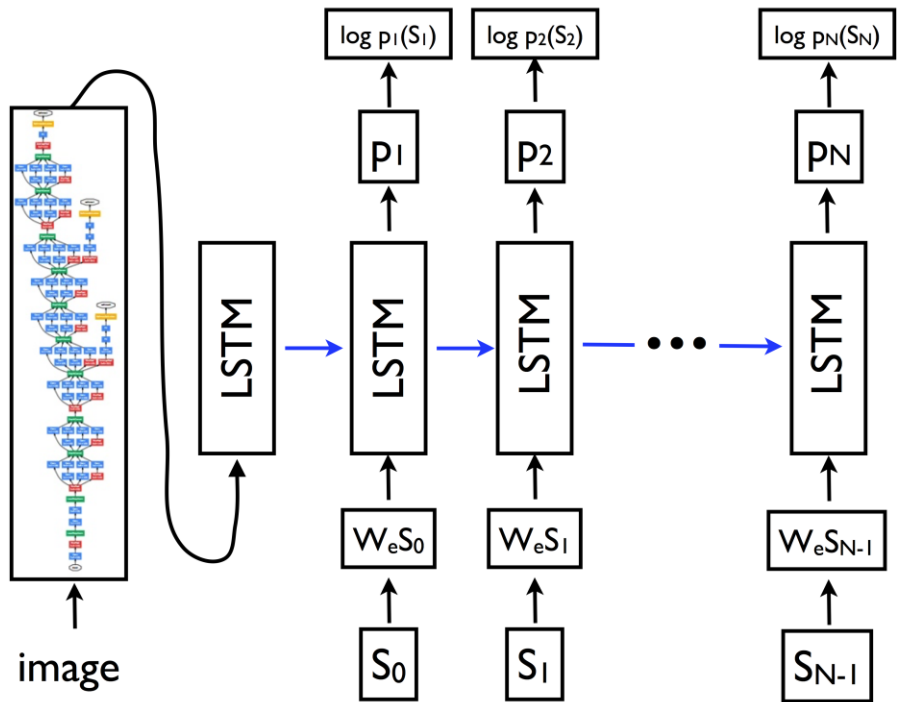
Sequences (many to many)



many to many



Sequences (one to many)



A person on a beach flying a kite.



A person skiing down a snow covered slope.



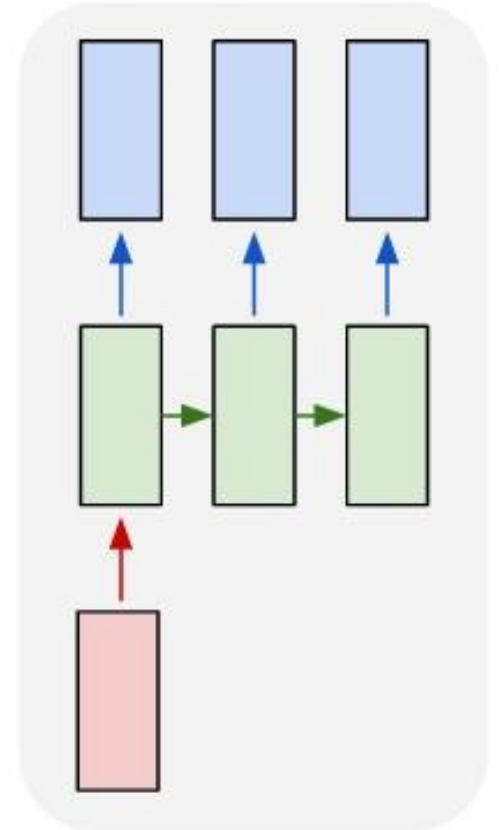
A black and white photo of a train on a train track.



A group of giraffe standing next to each other.

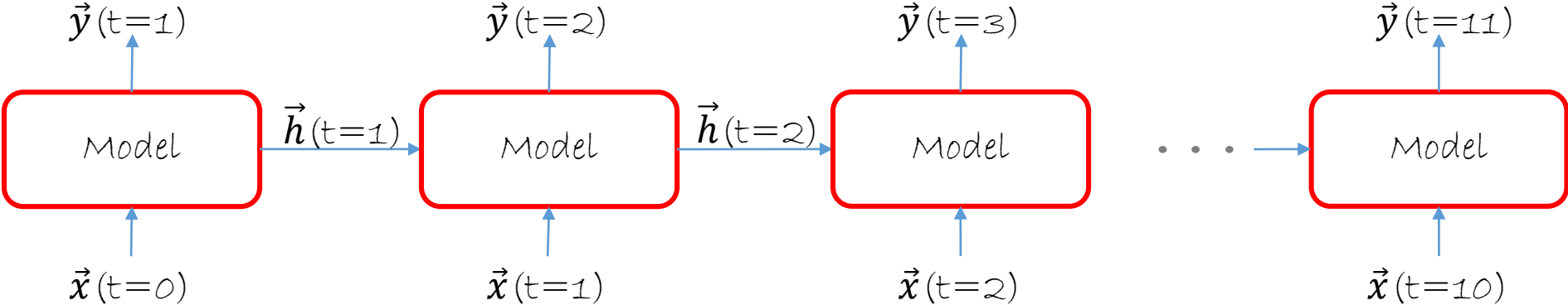


one to many



Recurrence

Problem: Predict the output of a solar panel for a day based on past N days

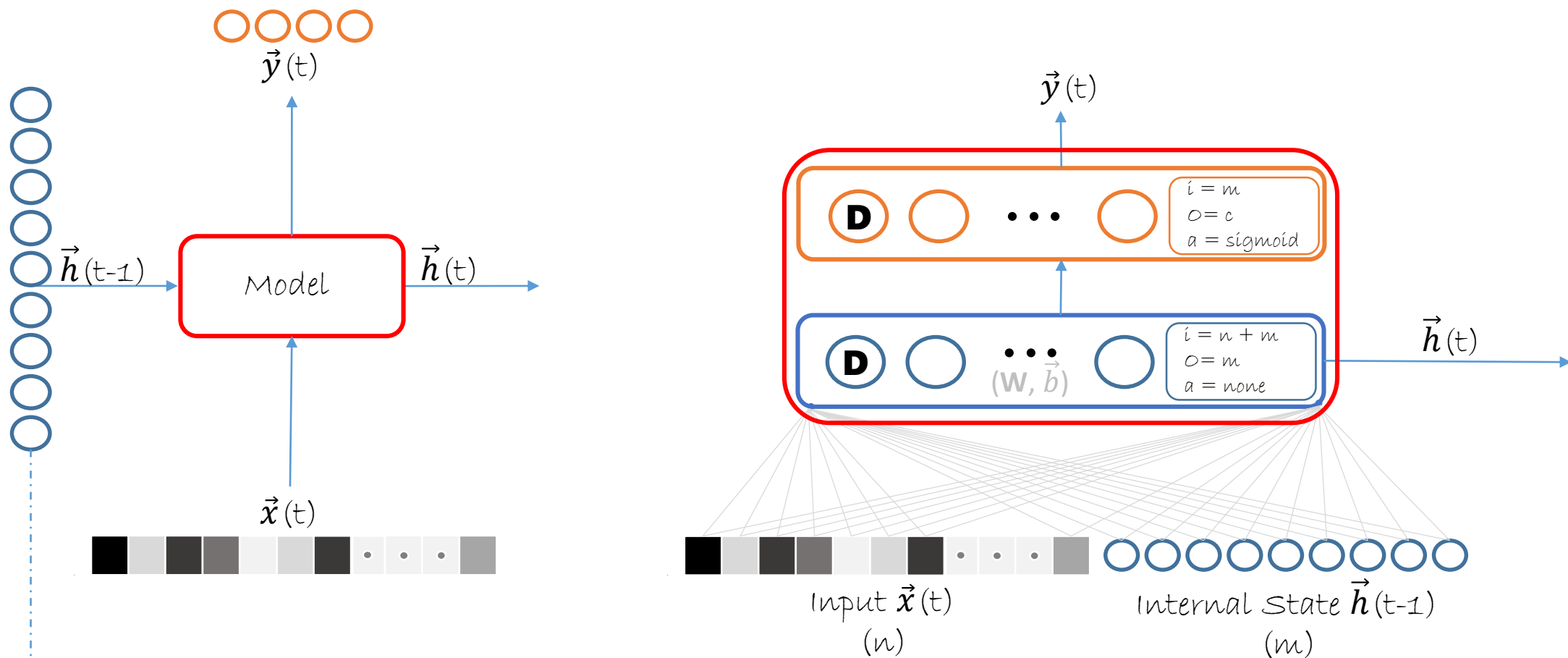


$\vec{x}(t)$: Input (n -dimensional array) at time t
 $\vec{h}(t)$: Internal State [m -dimensional array] at time t

D $\vec{h} = \mathbf{W} \vec{x}^T + \vec{b}$

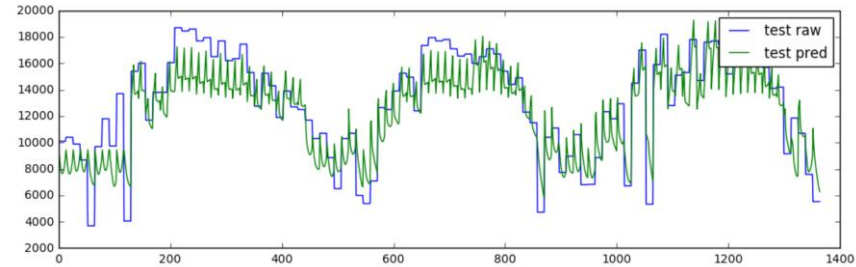
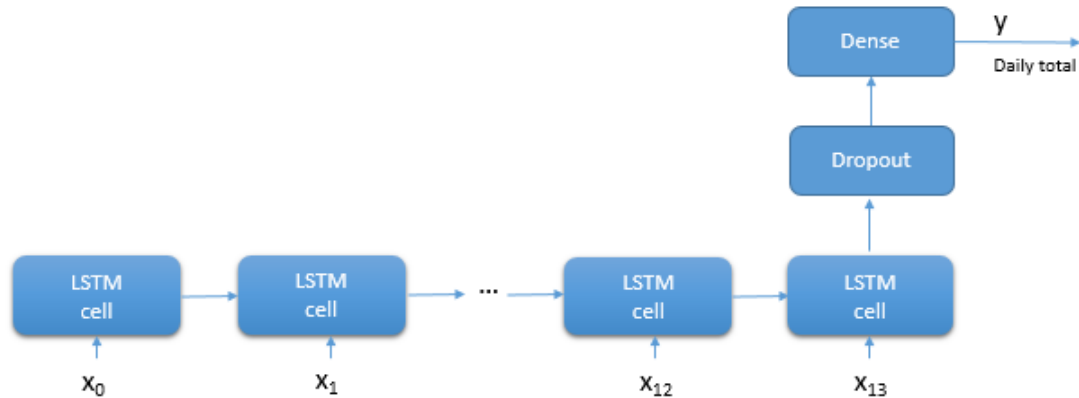
$\vec{y}(t)$: Output (c -dimensional array) at time t
 c : number of classes

Recurrence



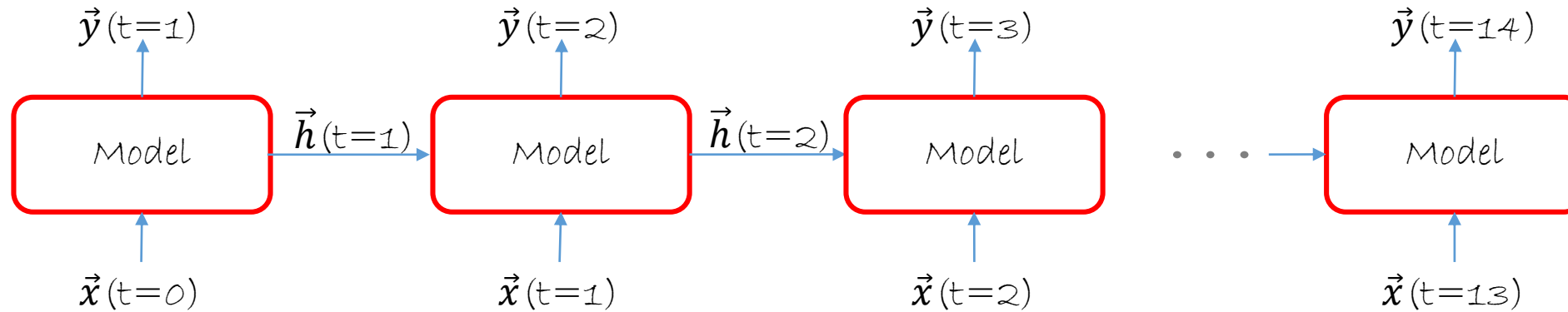
(\mathbf{W}, \vec{b}) = Parameters are share & updated across different time steps

Time-series Forecasting



```
def create_model(x):  
    """Create the model for time series prediction"""  
    with C.layers.default_options(initial_state = 0.1):  
        m = C.layers.Recurrence(C.layers.LSTM(TIMESTEPS))(x)  
        m = C.sequence.last(m)  
        m = C.layers.Dropout(0.2)(m)  
        m = cntk.layers.Dense(1)(m)  
    return m
```

Recurrence



$\vec{x}(t)$

For numeric:

Array of numeric values coming from different sensor

For an image:

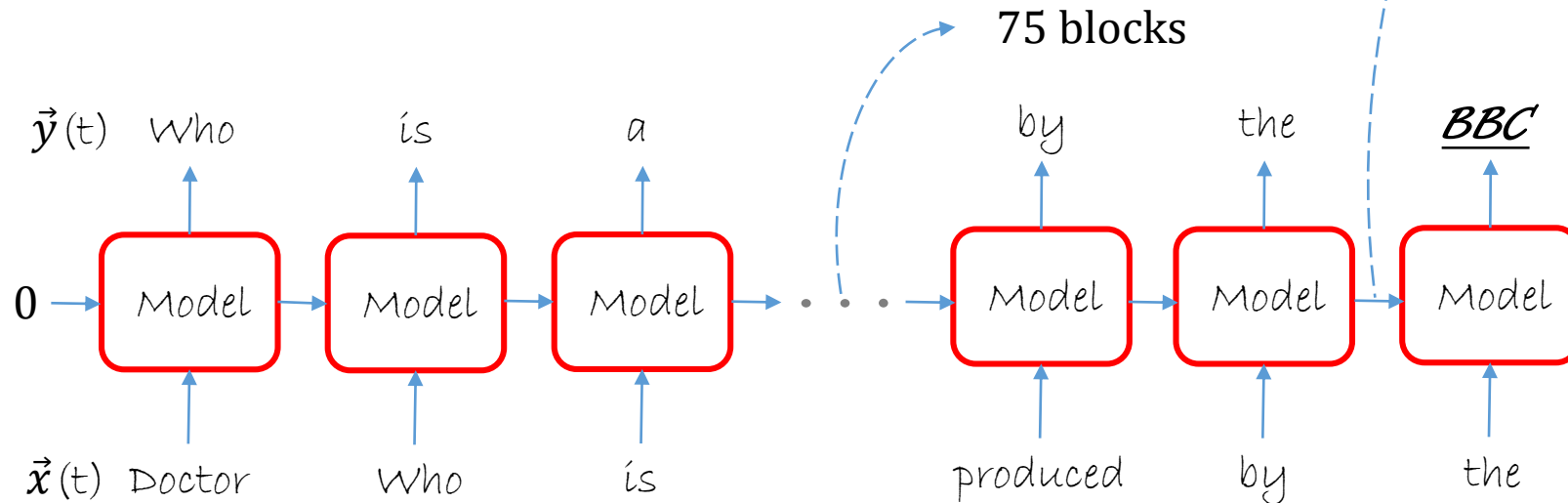
Pixels in an array, Map the image pixels to a compact representation (say n values)

For word in text:

Represent words as a numeric vector using embeddings (word2vec or GLOVE)

Recurrence (Vanishing Gradients)

Doctor Who is a British science-fiction television programme produced by the BBC since 1963. The programme depicts the adventures of the Doctor, a Time Lord—a space and time-travelling humanoid alien. He explores the universe in his TARDIS, a sentient time-travelling space ship. Accompanied by companions, the Doctor combats a variety of foes, while working to save civilizations and help people in need. This television series produced by the ?

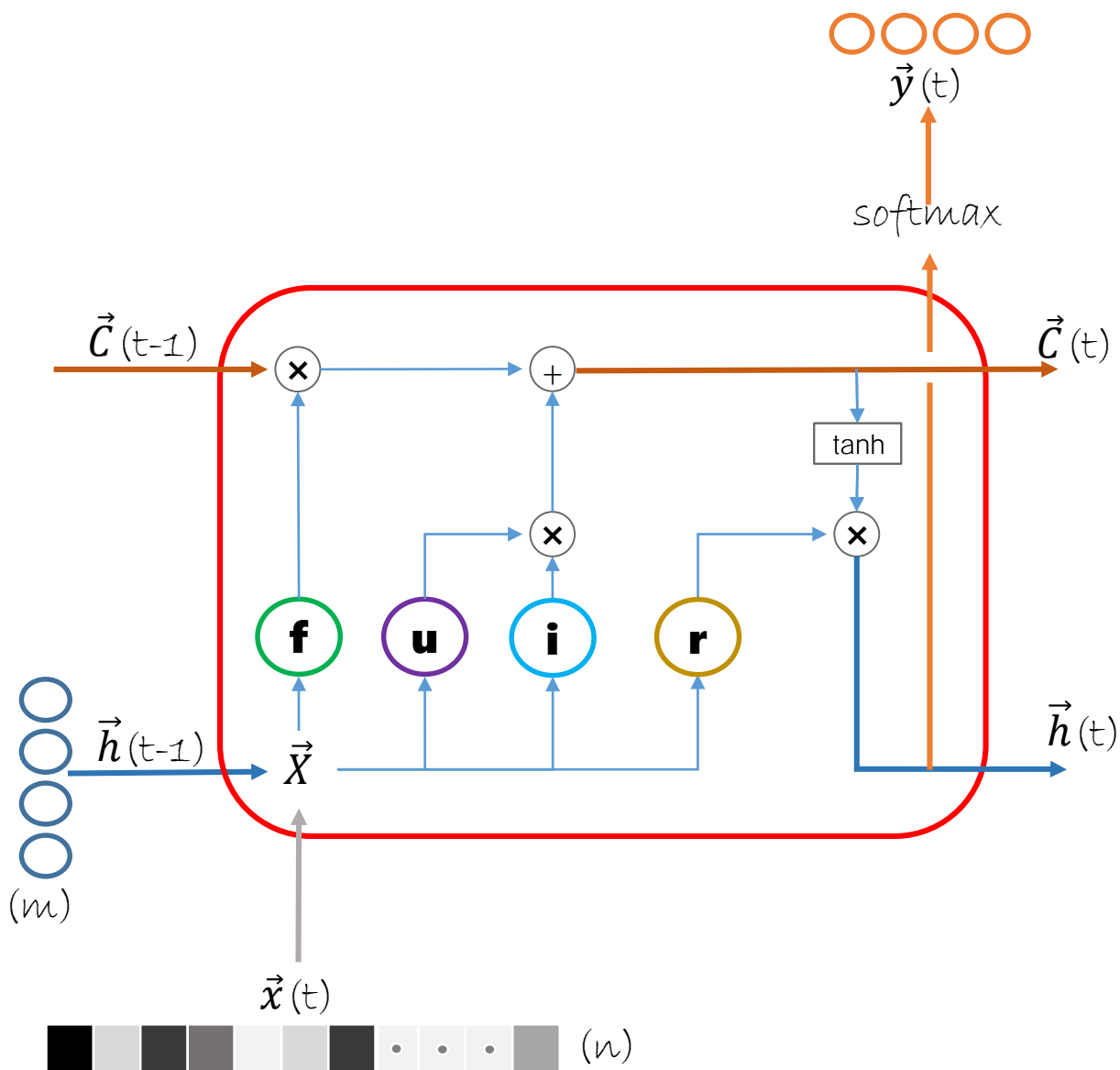


$$\vec{h} = \mathbf{W} \vec{x}^T + \vec{b}$$

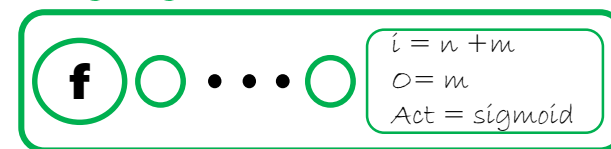


A single set of (\mathbf{W}, \vec{b})
has
limited memory

Long-Short Term Memory (LSTM)

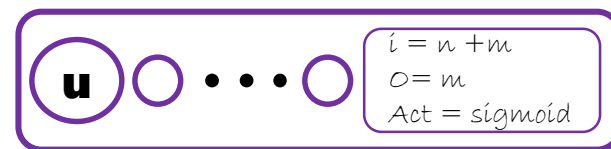


Forget gate



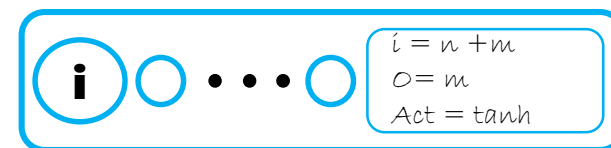
$$\vec{f} = \text{sigmoid}(\mathbf{W}_f \vec{X}^T + \vec{b}_f)$$

update gate



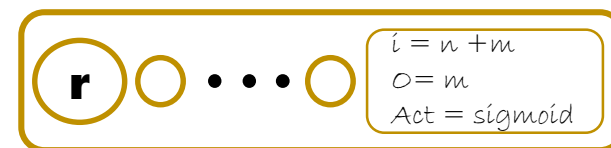
$$\vec{u} = \text{sigmoid}(\mathbf{W}_u \vec{X}^T + \vec{b}_u)$$

Input



$$\vec{X}^* = \text{tanh}(\mathbf{W}_i \vec{X}^T + \vec{b}_i)$$

Result gate



$$\vec{r} = \text{sigmoid}(\mathbf{W}_r \vec{X}^T + \vec{b}_r)$$

New cell memory

$$\vec{C}(t) = \vec{C}(t-1) \times \mathbf{f} + \mathbf{i} \times \mathbf{u}$$

New history

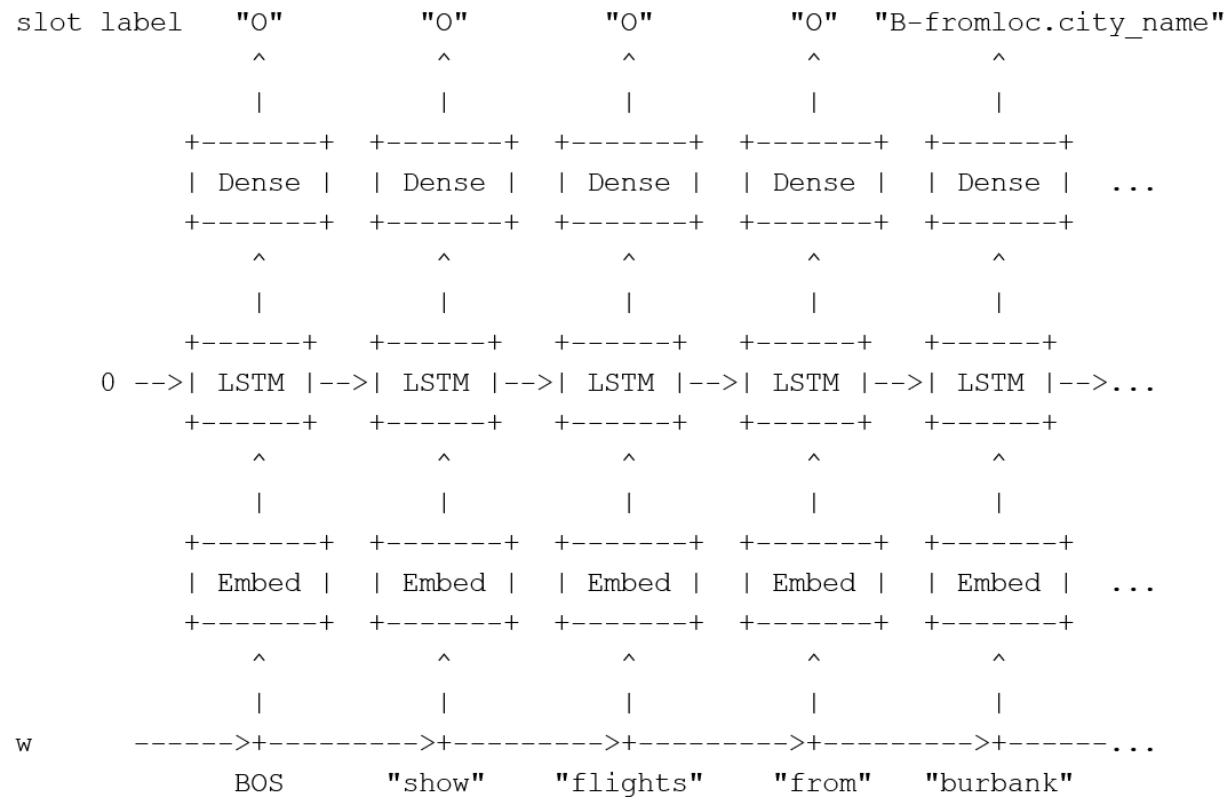
$$\vec{h}(t) = \text{tanh}(\vec{C}(t)) \times \mathbf{r}$$



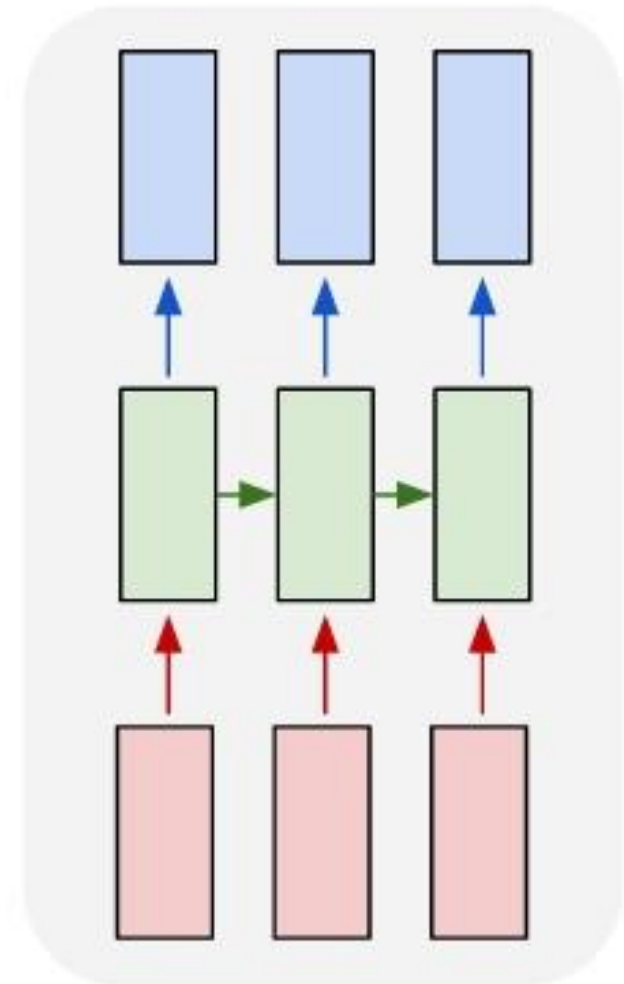
Language Understanding

Sequences (many to many) - Classification

Problem: Tagging entities in Air Traffic Controller (ATIS) data



many to many



ATIS Data

Domain:

- ✓ ATIS contains human-computer queries from the domain of Air Travel Information Services.

Data Summary:

- ✓ 943 unique words a.k.a. : Vocabulary
- ✓ 129 unique tags a.k.a.: Labels
- ✓ 26 intent tags: not used in this tutorial

Sequence Id	Input Word (sample)	Word Index (in vocabulary) S0	Word Label	Label Index (S2)
19	# BOS	178:1	# O	128:1
19	# please	688:1	# O	128:1
19	# give	449:1	# O	128:1
19	# me	581:1	# O	128:1
19	# the	827:1	# O	128:1
19	# flights	429:1	# O	128:1
19	# from	444:1	# O	128:1
19	# boston	266:1	# B-fromloc.city_name	48:1
19	# to	851:1	# O	128:1
19	# pittsburgh	682:1	# B-toloc.city_name	78:1
19	# on	654:1	# O	128:1
19	# thursday	845:1	# B-depart_date.day_name	26:1
19	# of	646:1	# O	128:1
19	# next	621:1	# B-depart_date.date_relative	25:1
19	# week	910:1	# O	128:1
19	# EOS	179:1	# O	128:1

Sequence Id: 19 indicates – this sentence is the 19th sentence in the data set

Word Index: ###:1 indicates the position of the corresponding word in the vocabulary (total 929 words)

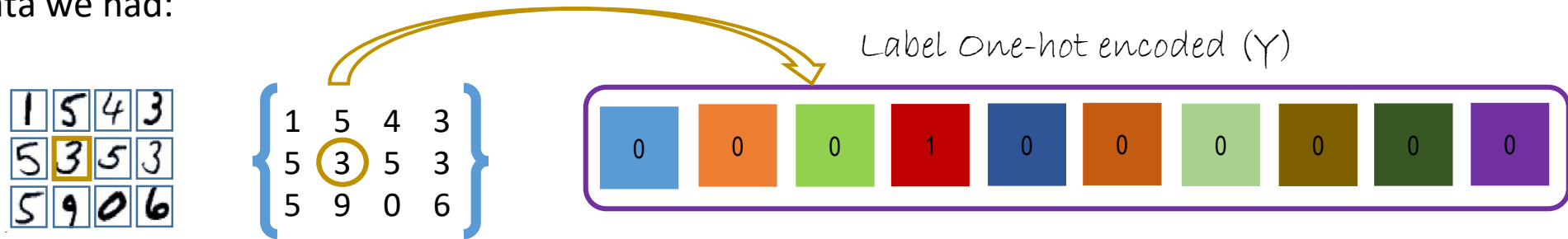
Label Index: ###:1 indicates the position of the corresponding tag in tag index (total 129 tags)

Sequence Tagging (Input / Label Preo

Vectorize Input Tokens (Step 1):

- Create a numerical representation of the input words
- This step is called *Embedding*

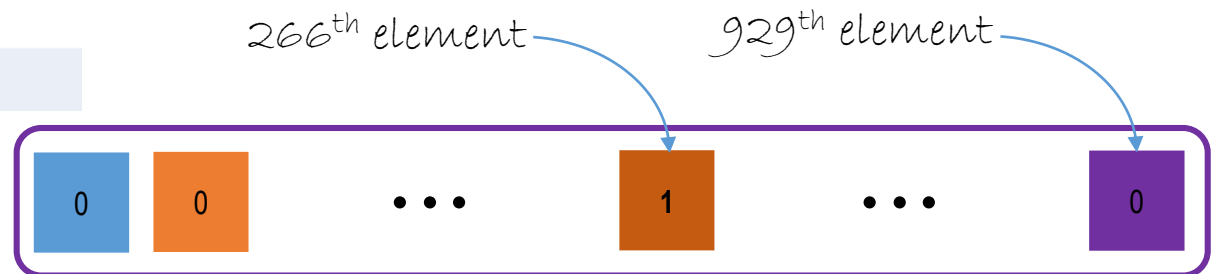
For MNIST data we had:



For Word data (one-hot encoding looks like)

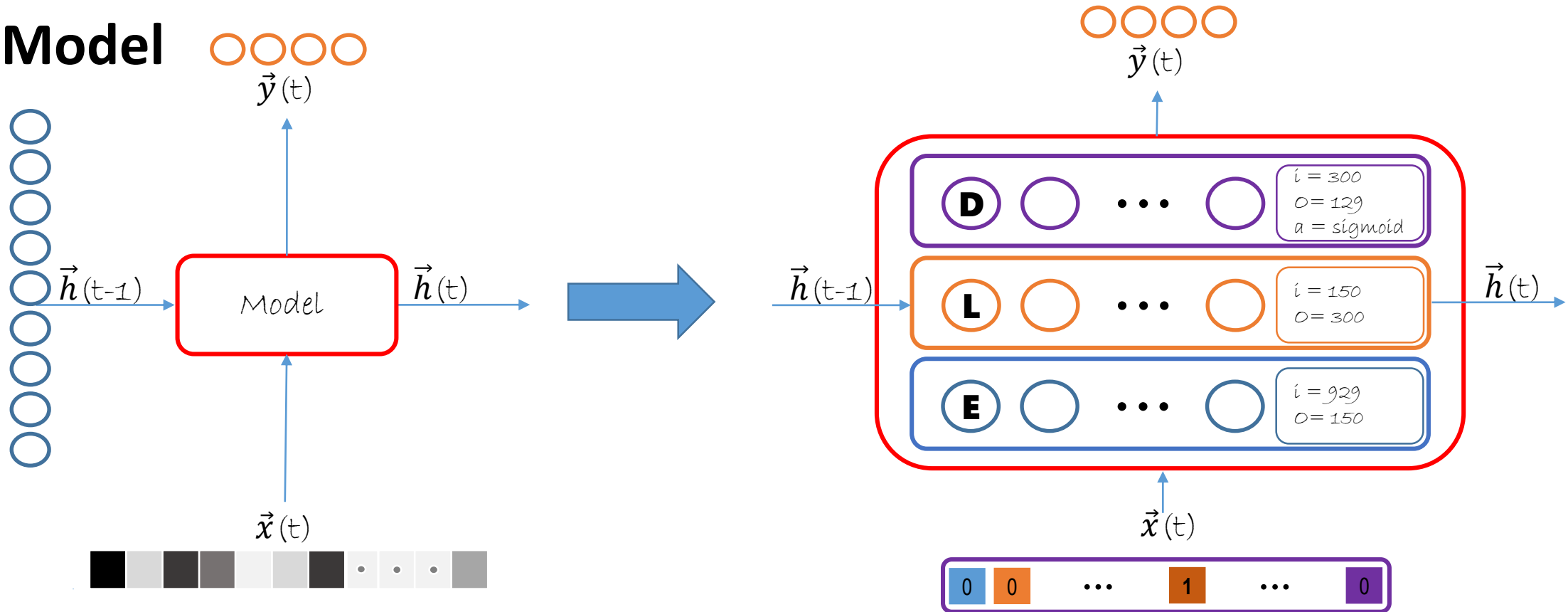
boston 266:1

- For vocabulary size of 929



For the label data – The one-hot representation is a 129 dimensional vector

Model



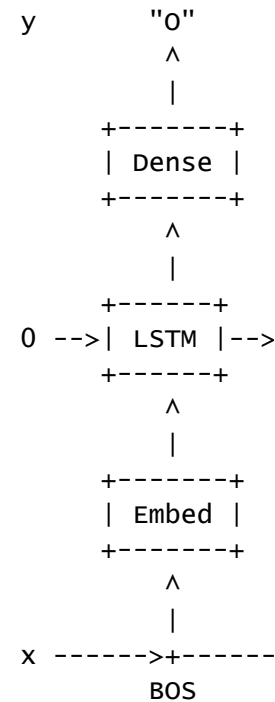
Embedding Layer (E):

- Projects a word in the input into a vector space: $\mathbf{W}_e \vec{X}^T$ (simple linear embedding)
- Here the weight matrix has dimension of 943 x 150
- Alternatively, more advanced embedding such as Glove can be used as \mathbf{W}_e

examples: language understanding

Task: Slot tagging with an LSTM

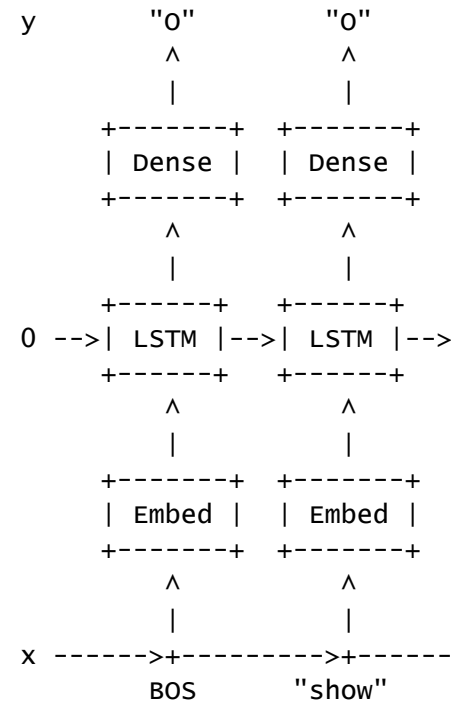
show	0
flights	0
from	0
burbank	B-fromloc.city_name
to	0
st.	B-toloc.city_name
louis	I-toloc.city_name
on	0
monday	B-depart_date.day_name



examples: language understanding

Task: Slot tagging with an LSTM

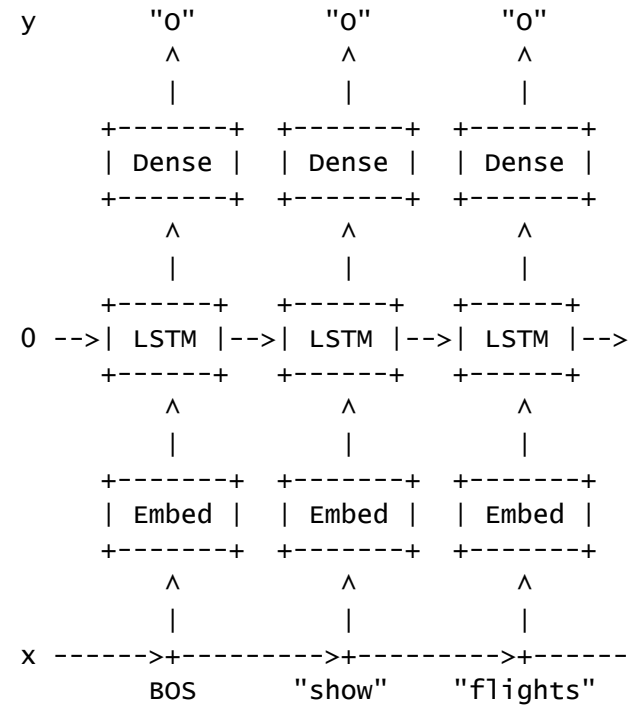
show	0
flights	0
from	0
burbank	B-fromloc.city_name
to	0
st.	B-toloc.city_name
louis	I-toloc.city_name
on	0
monday	B-depart_date.day_name



examples: language understanding

Task: Slot tagging with an LSTM

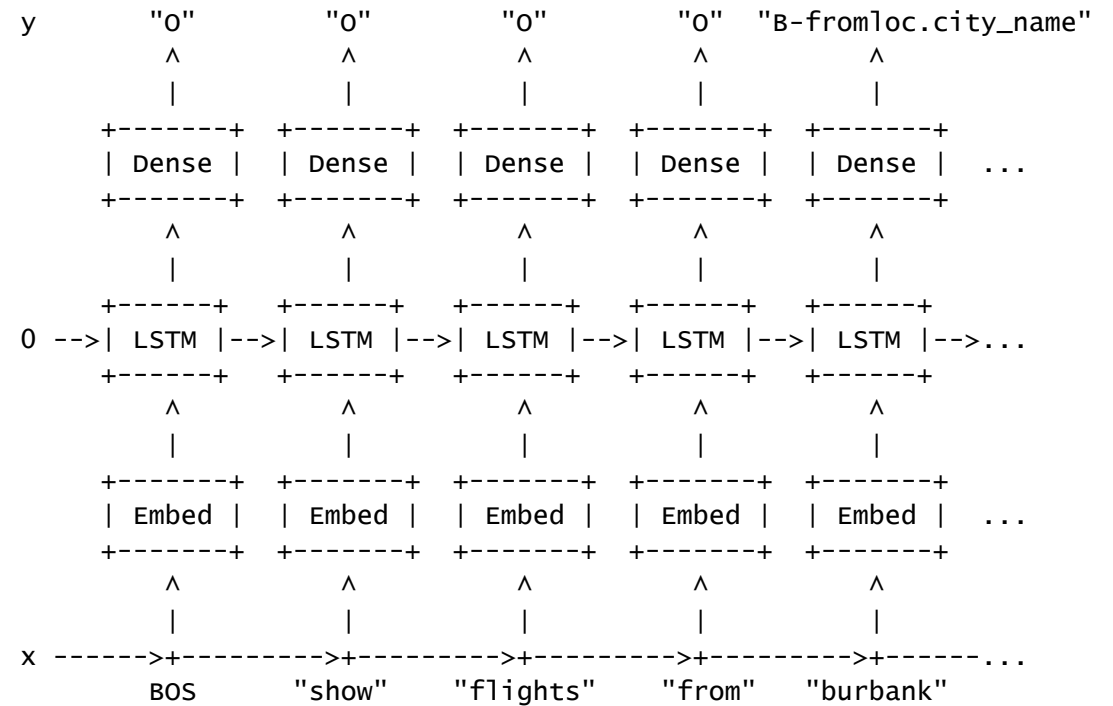
show	0
flights	0
from	0
burbank	B-fromloc.city_name
to	0
st.	B-toloc.city_name
louis	I-toloc.city_name
on	0
monday	B-depart_date.day_name



examples: language understanding

Task: Slot tagging with an LSTM

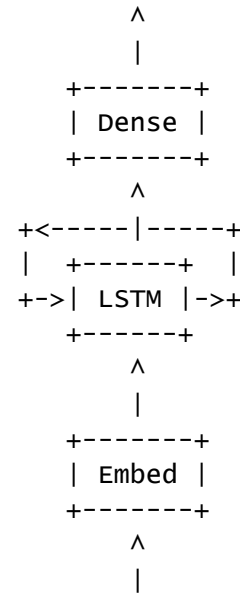
show	0
flights	0
from	0
burbank	B-fromloc.city_name
to	0
st.	B-toloc.city_name
louis	I-toloc.city_name
on	0
monday	B-depart_date.day_name



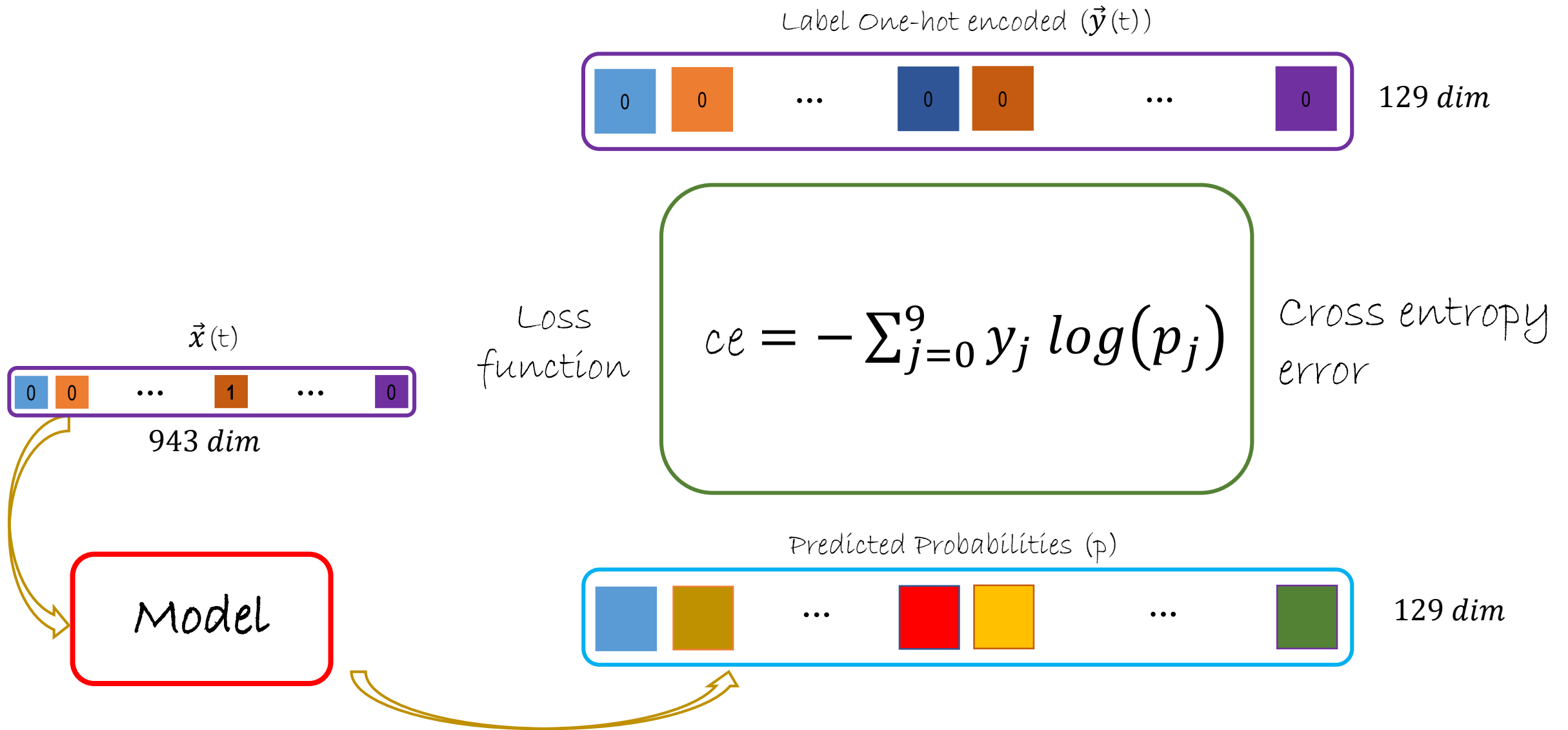
examples: language understanding

Task: Slot tagging with an LSTM

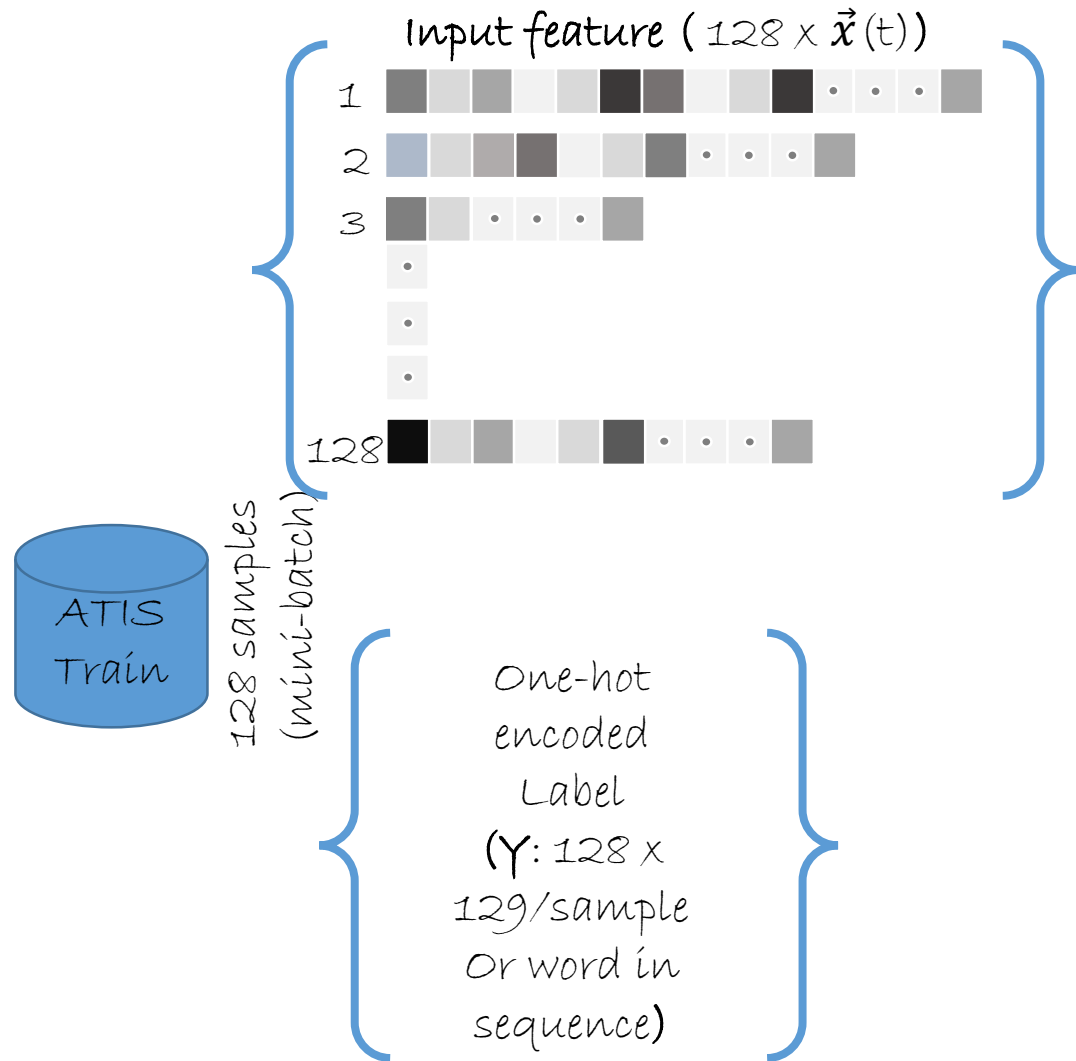
```
model = Sequential ([  
    Embedding(150),  
    RecurrentLSTM(300),  
    Dense(labelDim)  
)
```



Error or Loss Function



Train Workflow



```
z = model():  
    return  
        Sequential([  
            Embedding(emb_dim=150),  
            Recurrence(LSTM(hidden_dim=300),  
                        go_backwards=False),  
            Dense(num_labels = 129)  
        ])
```

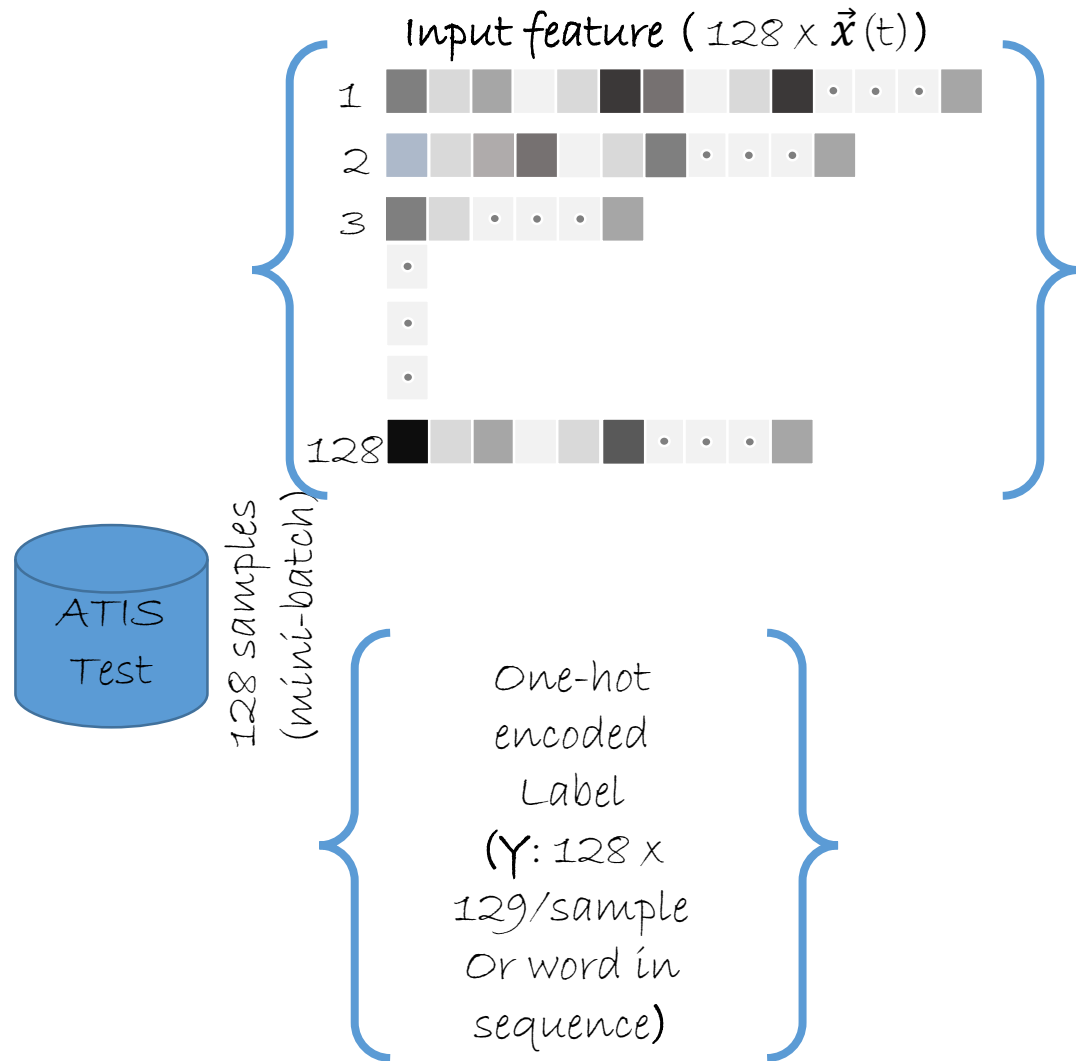


```
Loss cross_entropy_with_softmax(z, Y)  
Error (optional) classification_error(z, Y)
```

```
Trainer(model, (loss, error), learner)  
Trainer.train_minibatch({X, Y})
```

Learner
sgd, adagrad etc, are solvers to estimate

Test Workflow



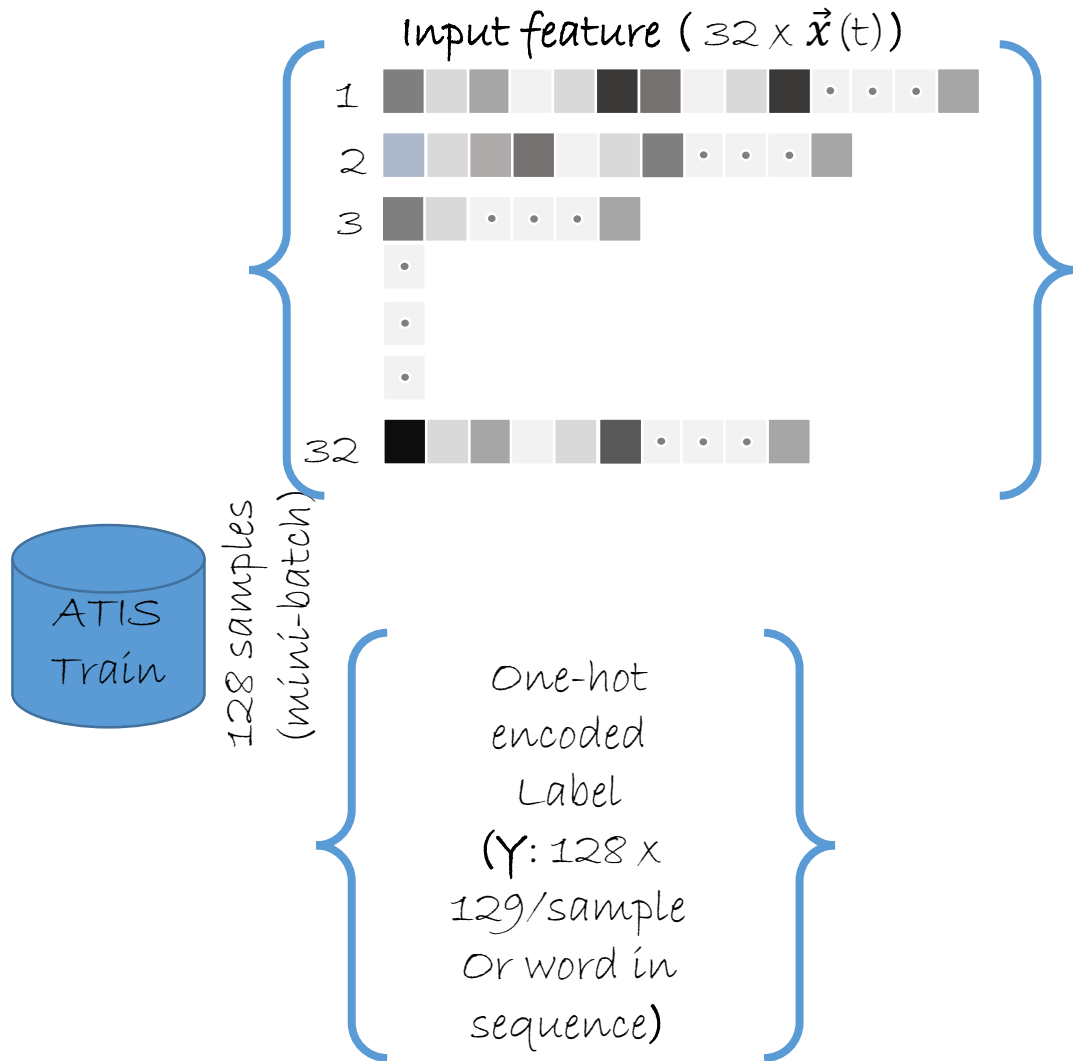
```
z = model():  
    return  
        Sequential([  
            Embedding(emb_dim=150),  
            Recurrence(LSTM(hidden_dim=300),  
                        go_backwards=False),  
            Dense(num_labels = 129)  
        ])
```



```
Loss cross_entropy_with_softmax(z,  $\gamma$ )  
Error (optional) classification_error(z,  $\gamma$ )
```

```
Trainer(model, (loss, error), learner)  
Trainer.test_minibatch({X,  $\gamma$ })  
Learner  
sgd, adagrad etc, are solvers to estimate
```

Test Workflow



```
z = model():  
    return  
        Sequential([  
            Embedding(emb_dim=150),  
            Recurrence(LSTM(hidden_dim=300),  
                        go_backwards=False),  
            Dense(num_labels = 129)  
        ])
```

Loss `cross_entropy_with_softmax(z, Y)`

Error (optional) `classification_error(z, Y)`

`Trainer(model, (loss, error), learner)`

`Trainer.train_minibatch({X, Y})`

Learner
sgd, adagrad etc, are solvers to estimate

Prediction Workflow



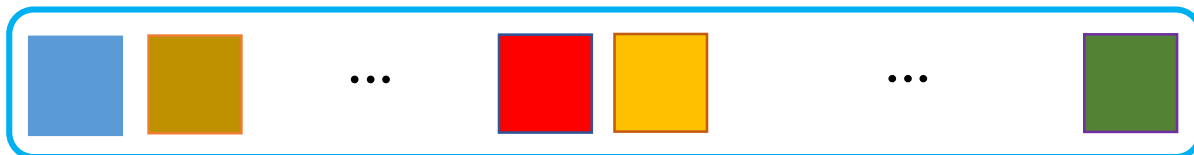
'BOS flights from new york to seattle EOS'

Input feature (new X: $1 \times 8 \times (1 \times 943)$)



`Model.eval(new X)`

Predicted Softmax Probabilities



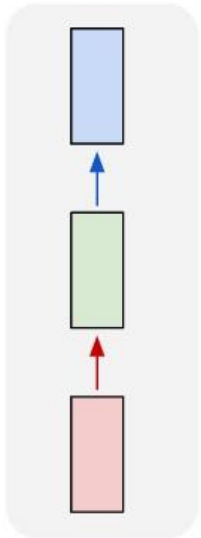
Output prediction ($: 1 \times 8 \times (1 \times 129)$)



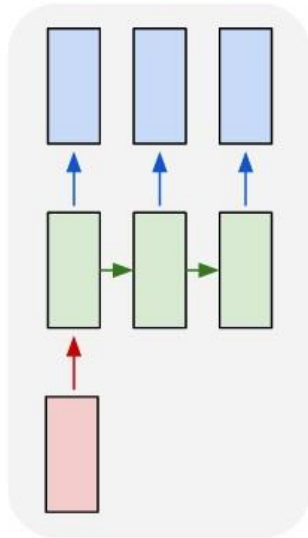
Sequence to Sequence

Neural network paradigms

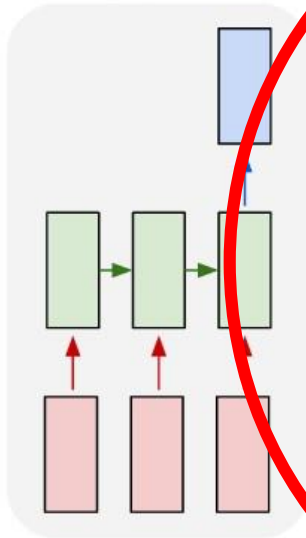
one to one



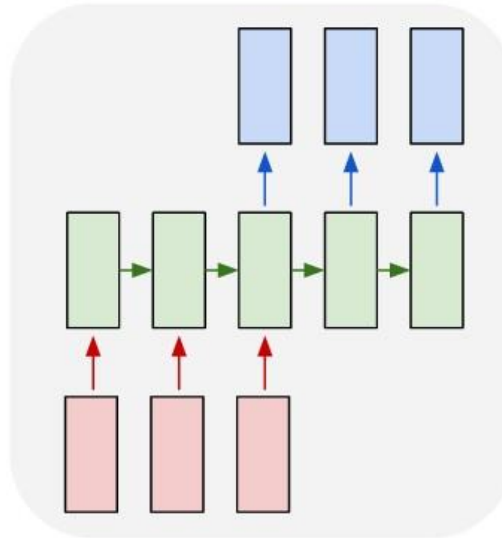
one to many



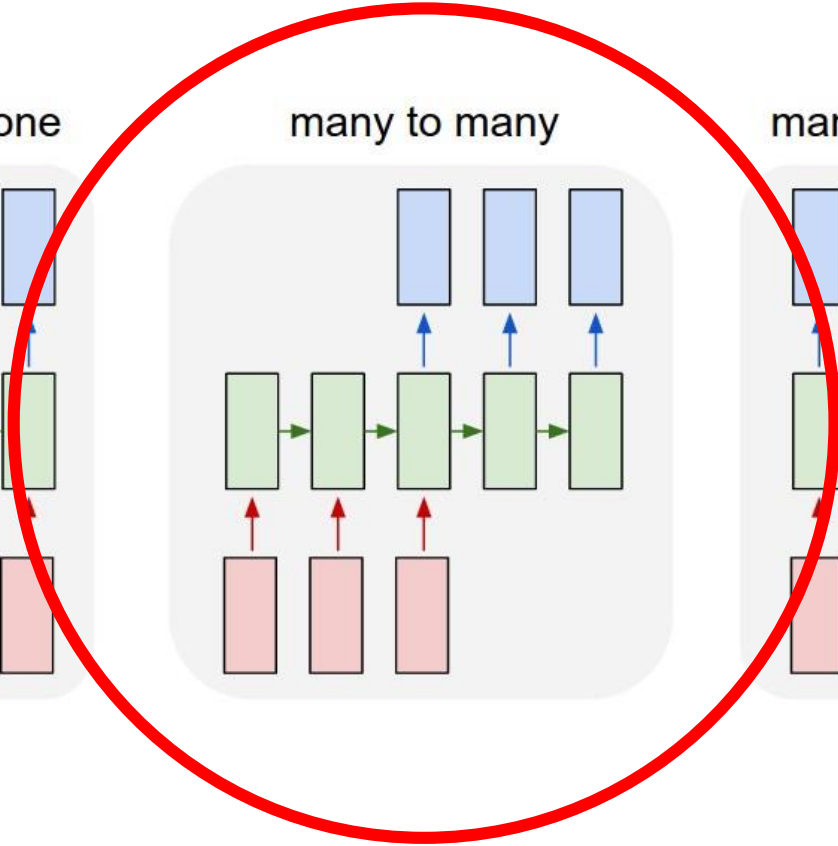
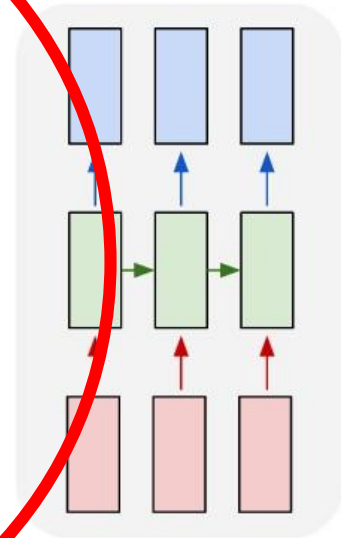
many to one



many to many



many to many



Background

First described in the context of machine translation

- ✓ Cho, et al., “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation” (2014). <https://arxiv.org/abs/1406.1078>.

It is a natural fit for:

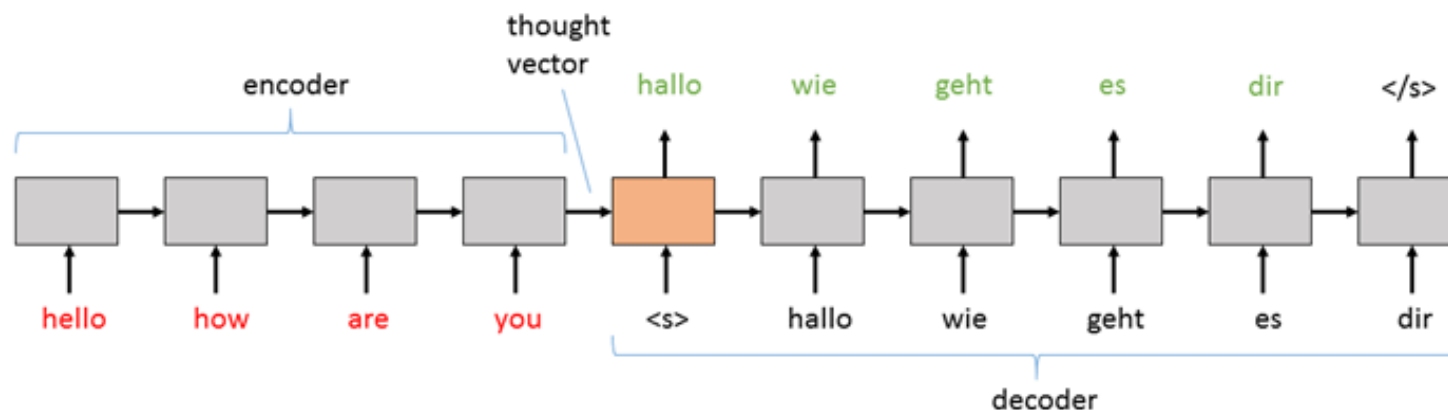
- ✓ Automatic text summarization:
 - Input sequence: full document
 - Output sequence: summary document
- ✓ Word to pronunciation models:
 - Input sequence: character [grapheme]
 - Output sequence: pronunciation[phoneme]
- ✓ Question – Answering models:
 - Input sequences: Query and document
 - Output sequence: Answer

Basic Theory

A sequence-to-sequence model consists of two main pieces:

- (1) an encoder,
- (2) a decoder, and
- (3) an attention module (optional)

Sequence to Sequence Mechanism:



- ✓ Encoder
 - Processes the input sequence into a fixed representation
 - This representation is fed into the decoder as a context a.k.a thought vector
- ✓ Decoder
 - Uses some mechanism to decode the processed information into an output sequence
 - This is a language model that is augmented with some "strong context"
 - Each symbol that it generates is fed back into the decoder for additional context

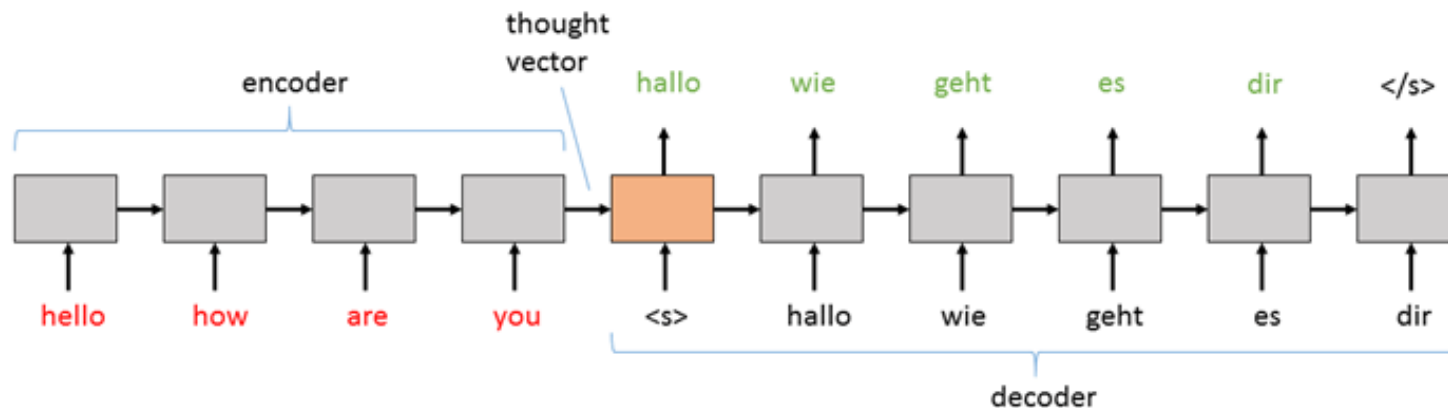
What is “thought-vector”

Term popularized by Geoffrey Hinton

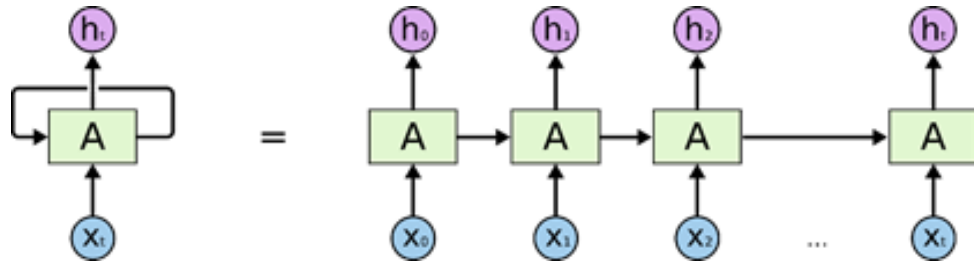
“What I think is going to happen over the next few years is this ability to turn sentences into *thought vectors* is going to rapidly change the level at which we can understand documents”

What is a thought-vector:

- ✓ Like an embedding similar to (word2vec & GloVe) but instead encodes several words, or ideas, or... a “thought”
- ✓ In basic sequence to sequence, the thought vector represents:
 - the encoded version of the input sequence after running it through the encoder RNN
 - the hidden state of the encoder after all of the words in the input sequence have passed through I
 - The decoder’s hidden state is then *initialized* with this thought vector



Example



For every input:

- the hidden state is updated
- some output is returned

$$h_t = \tanh(Ux_t + Wh_{t-1})$$

$$o_t = Vh_t$$

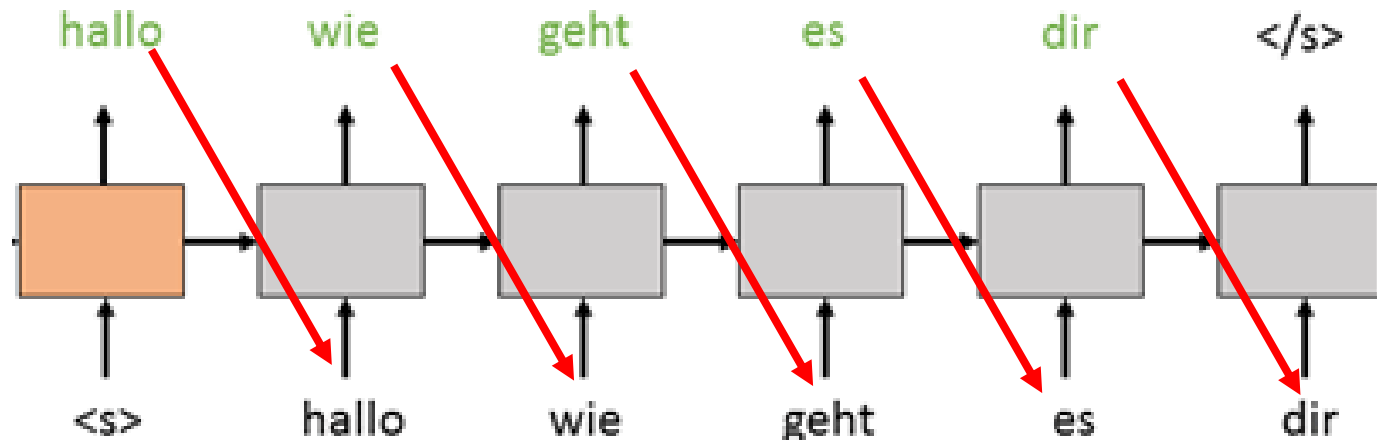


```
def step(x):  
    h = C.tanh(C.times(U, x) + C.times(W, h))  
    o = C.times(V, h)  
    return o
```

Sequence to Sequence Decoder

In the sequence-to-sequence decoder:

- ✓ Output o is projected through a dense layer and softmax function
- ✓ The resultant word is directed back into itself as the input for the next step
- ✓ This is a greedy-decoding approach



Sequence to Sequence Decoder

Steps in decoding:

- ✓ First step is to initialize the decoder RNN with the thought vector as its hidden state
- ✓ Use a "sequence start" tag (e.g. <s>) as input to prime the decoder to start generating an output sequence
- ✓ The decoder keeps generating outputs until it hits the special "end sequence" tag (e.g. </s>)

```
def model_greedy(input): # (input*) --> (word_sequence*)

    # Decoding is an unfold() operation starting from sentence_start.
    # We must transform s2smodel (history*, input* -> word_logp*)
    # into a generator (history* -> output*) which holds 'input' in its closure.
    unfold = UnfoldFrom(lambda history: s2smodel(history, input) >> hardmax,
                        # stop once sentence_end_index was max-scoring output
                        until_predicate=lambda w: w[... , sentence_end_index],
                        length_increase=length_increase)

    return unfold(initial_state=sentence_start, dynamic_axes_like=input)
```


Sequence to Sequence Problems

Squeezing all the input sequence information into a single vector

At each time step:

- ✓ the hidden state h gets updated with the most recent information, and
- ✓ therefore h gets "diluted" in information as it processes each token

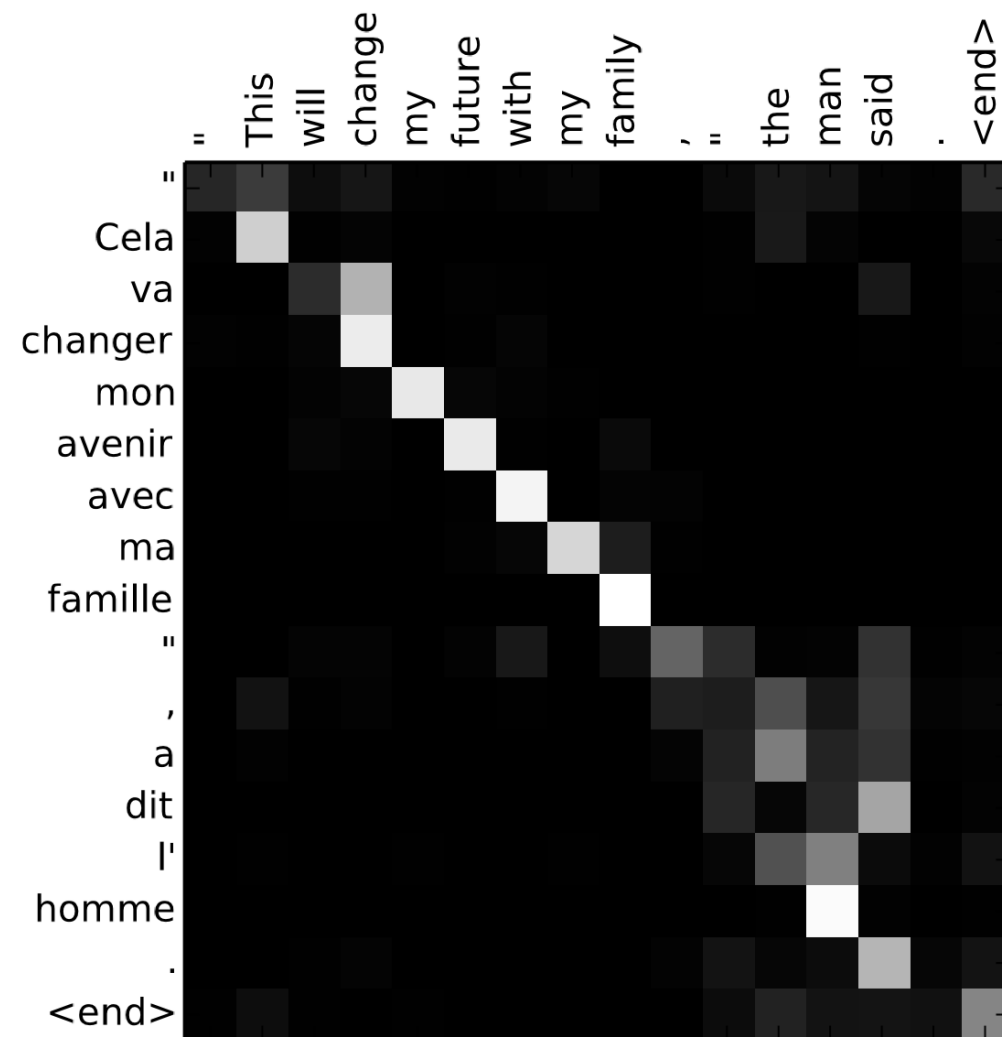
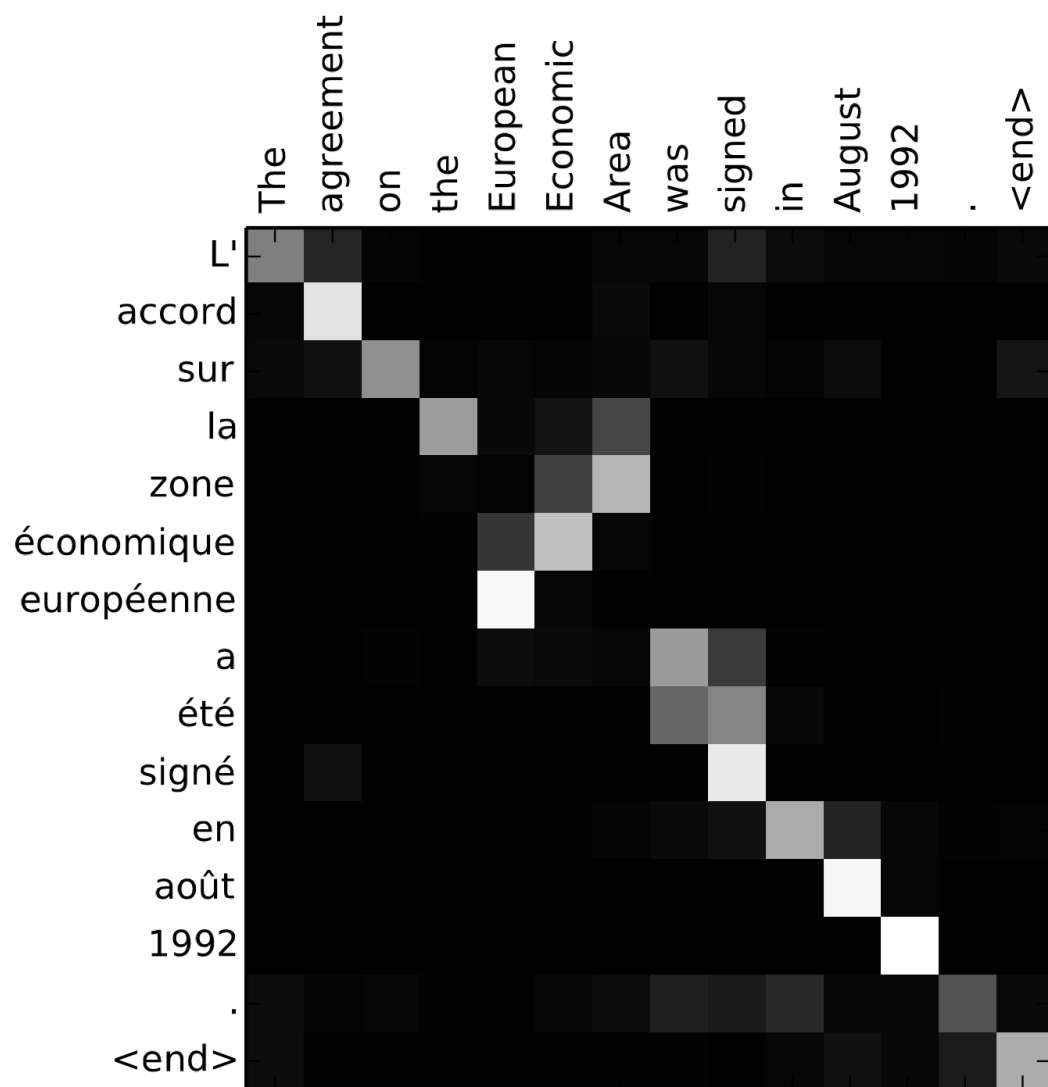
Token position influence

- ✓ Even with a relatively short sequence, the last token will always get the last say and
- ✓ therefore the thought vector is biased/weighted towards that last word

For Machine Translation:

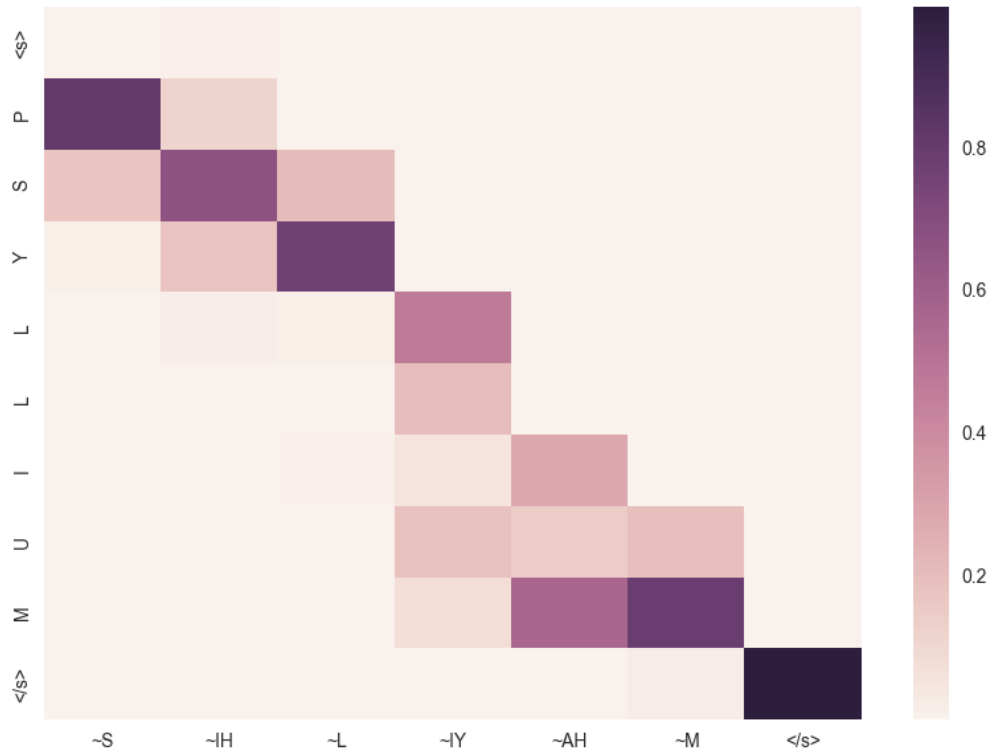
- ✓ We run the encoder backwards also to help mitigate this problem
- ✓ Need a more systematic approach

Attention Mechanism



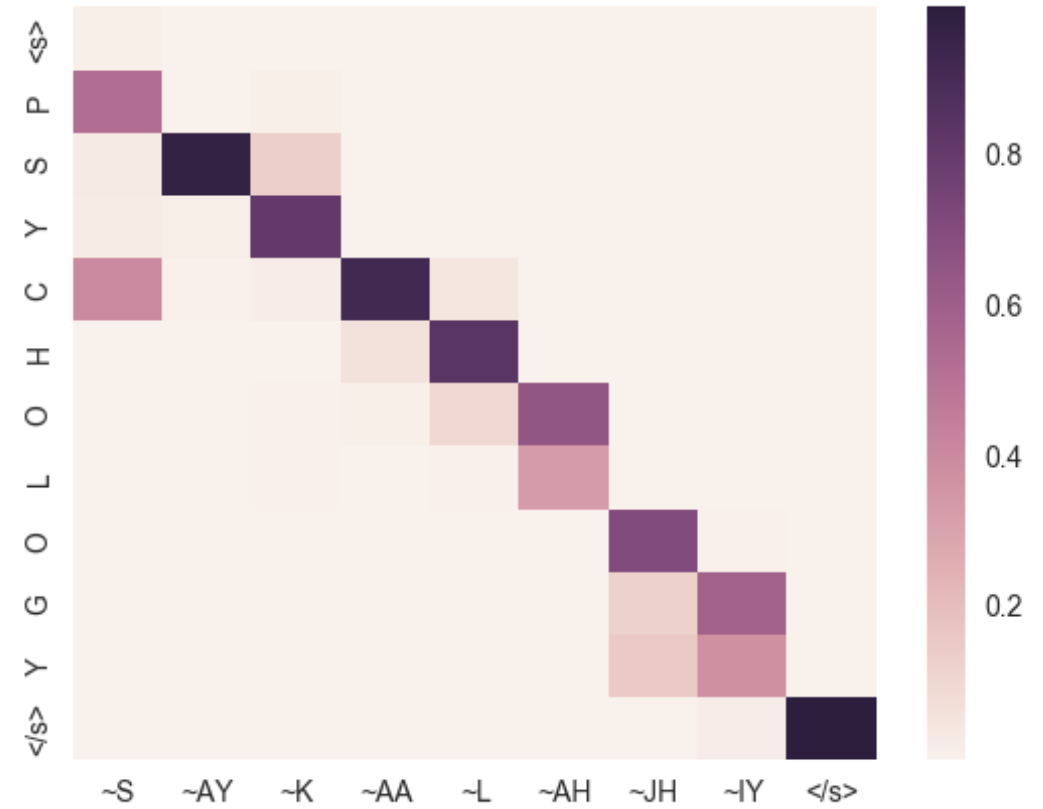
Attention Mechanism

psyllium



vs.

psychology



Attention Mechanism

Helps to solve the “long-sequence” and alignment problem

Replace single thought vector (and only as an initial context) with:

- ✓ Each decoding step directly use information from the encoder
- ✓ All of the hidden states from the encoder are available to us (instead of just the final one); and
- ✓ The decoder *learns* which weighted sum of hidden states, given the current context and input, to use

How is it done:

- ✓ Learn which encoder hidden states are important given current context and input; and
- ✓ Augment the decoder’s current hidden state with information from those states

Attention Mechanism

Key Idea: Learn which encoder states are important given current context and input

1. Compute similarity between different encoder states w.r.t. a given decoder state

✓ Dot product between h_i and d

✓ Cosine distance between h_i and d

✓ Projected similarity given by

$$u_i = v^T \tanh(W_1 h_i + W_2 d)$$

Where h_i is the hidden state for each encoding RNN unit and d is the corresponding decoder state

Note: v is a learnable vector parameter; W_1 and W_2 are learnable matrix parameters

✓ Finally the attention score for a given comparison can be computed as:

$$a_i = \text{softmax}(u_i)$$

Attention Mechanism

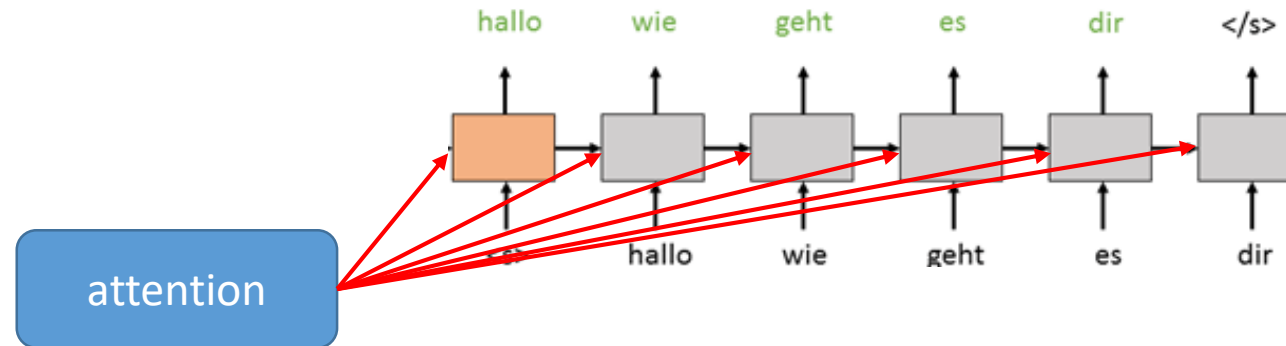
2. Augment the decoder's current hidden state with information from those states

- ✓ Create a vector in the same space as the hidden states that consists of a weighted sum of the encoder hidden states

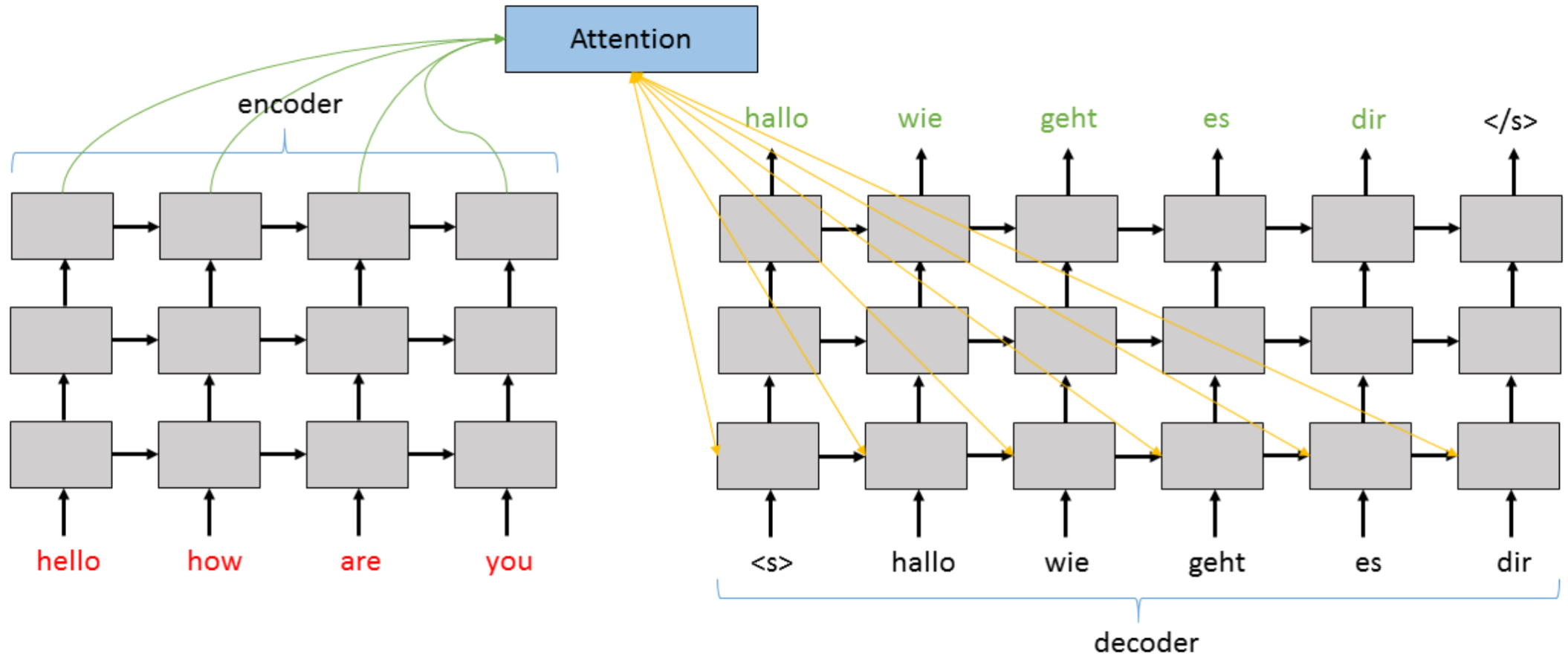
$$d' = \sum_{i=1}^{T_A} a_i h_i$$

- ✓ New hidden state for predicting current word:

$$D = d + d'$$



Attention Mechanism



Decoding with Attention

Take a greedy approach and output the most probable word at each step

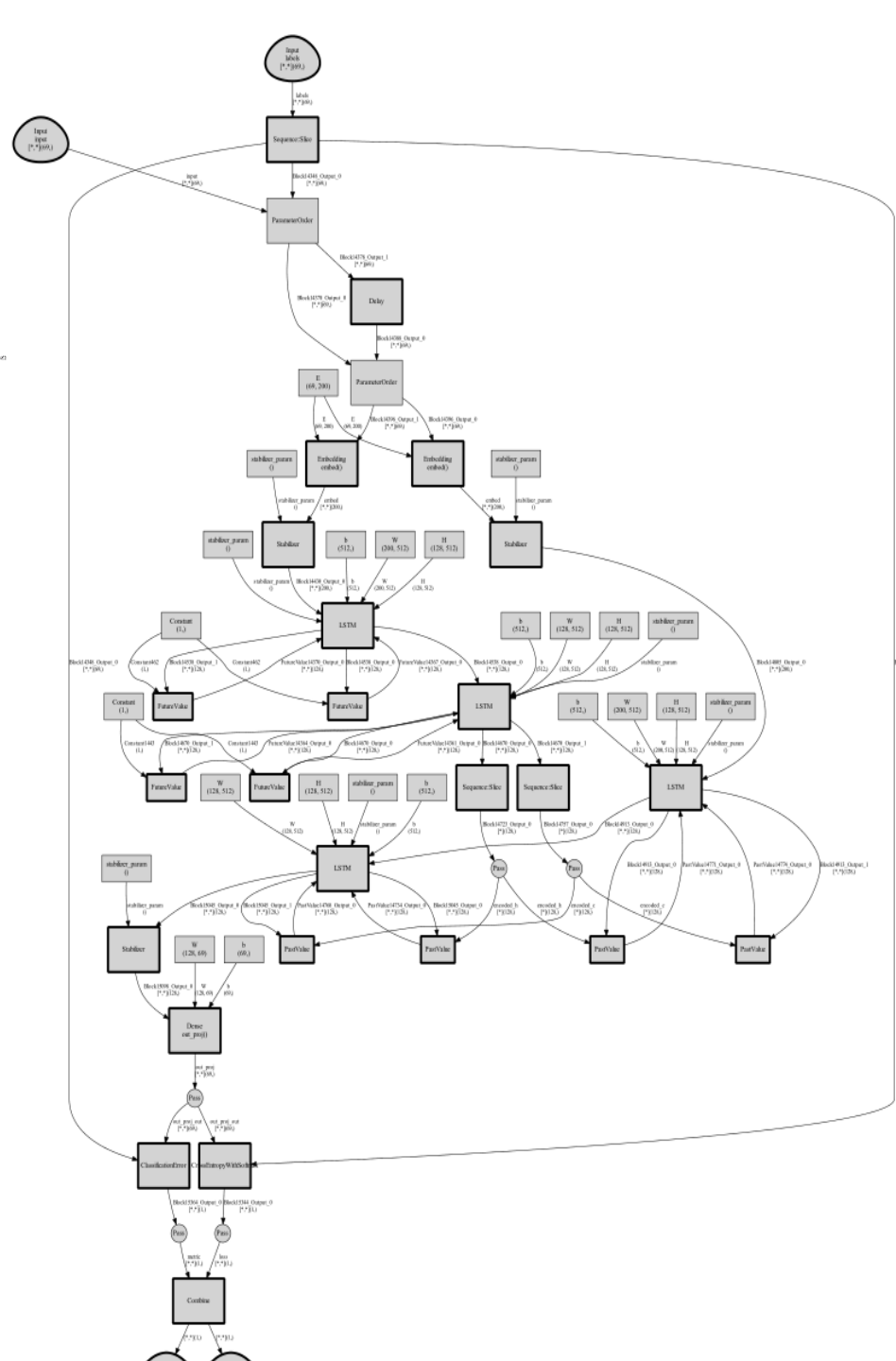
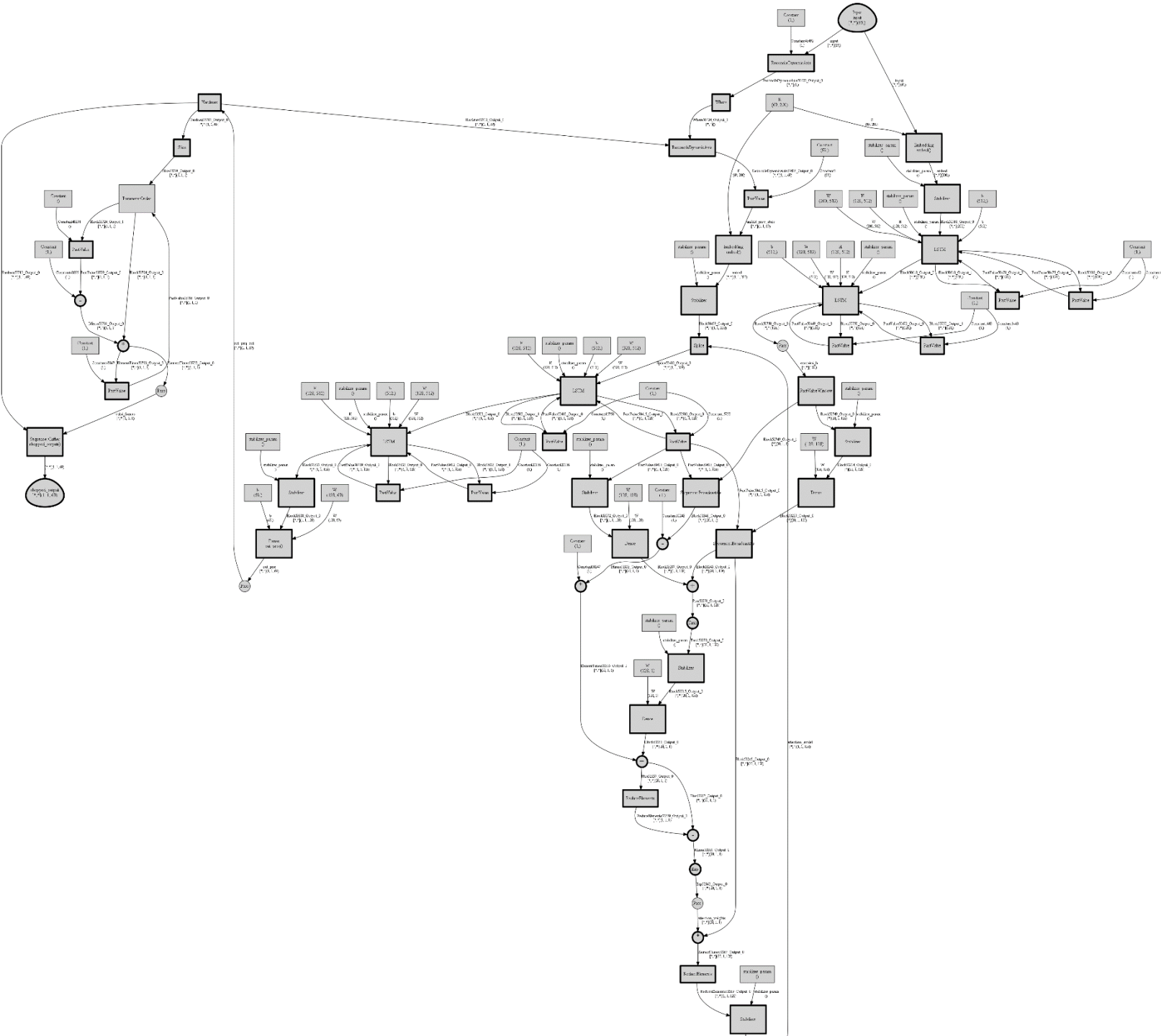
- ✓ Does not render well in practice

Consider every single combination at each step

- ✓ However, that is generally computationally intractable

Strike a compromise using beam search

- ✓ Instead, we use a beam search decoder with a given depth
- ✓ The depth parameter considers how many best candidate solutions to keep at each step
- ✓ This results in a heuristic for the global optimal that works quite well
- ✓ Indeed, a beam search of 3 gives very good results in most situations.





Machine Comprehension with ReasonNet

ReasonNet: Learning to Stop Reading in Machine Comprehension

Yelong Shen, Po-Sen Huang, Jianfeng Gao, Weizhu Chen



Microsoft Research

CNTK Tutorial: Pengcheng He, Amit Agarwal, Sayan Pathak

Problem Definition

- Machine Comprehension
 - Teach machine to answer questions given an input passage

Query

Who is the producer of Doctor Who?

Passage

Doctor Who is a British science-fiction television programme produced by the **BBC** since 1963. The programme depicts the adventures of the Doctor, a Time Lord—a space and time-travelling humanoid alien. He explores the universe in his TARDIS, a sentient time-travelling space ship. Its exterior appears as a blue British police box, which was a common sight in Britain in 1963 when the series first aired. Accompanied by companions, the Doctor combats a variety of foes, while working to save civilisations and help people in need.

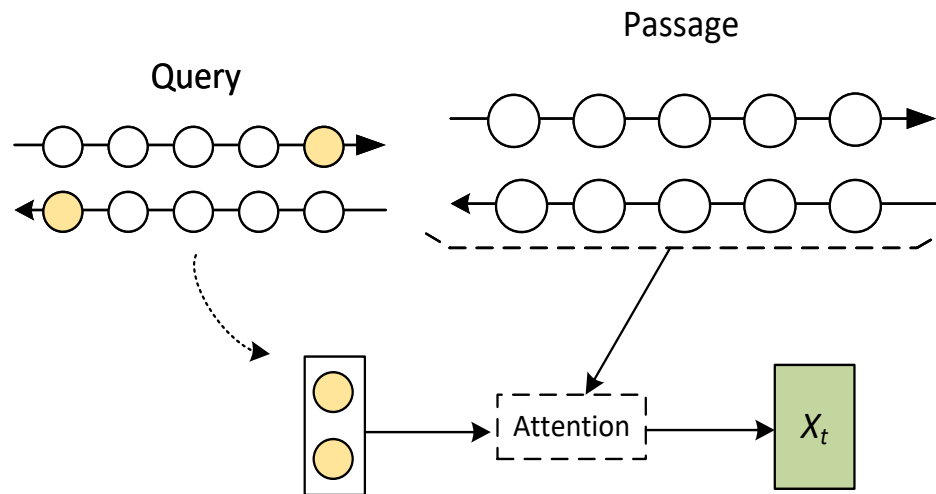
Answer

BBC

Related Work

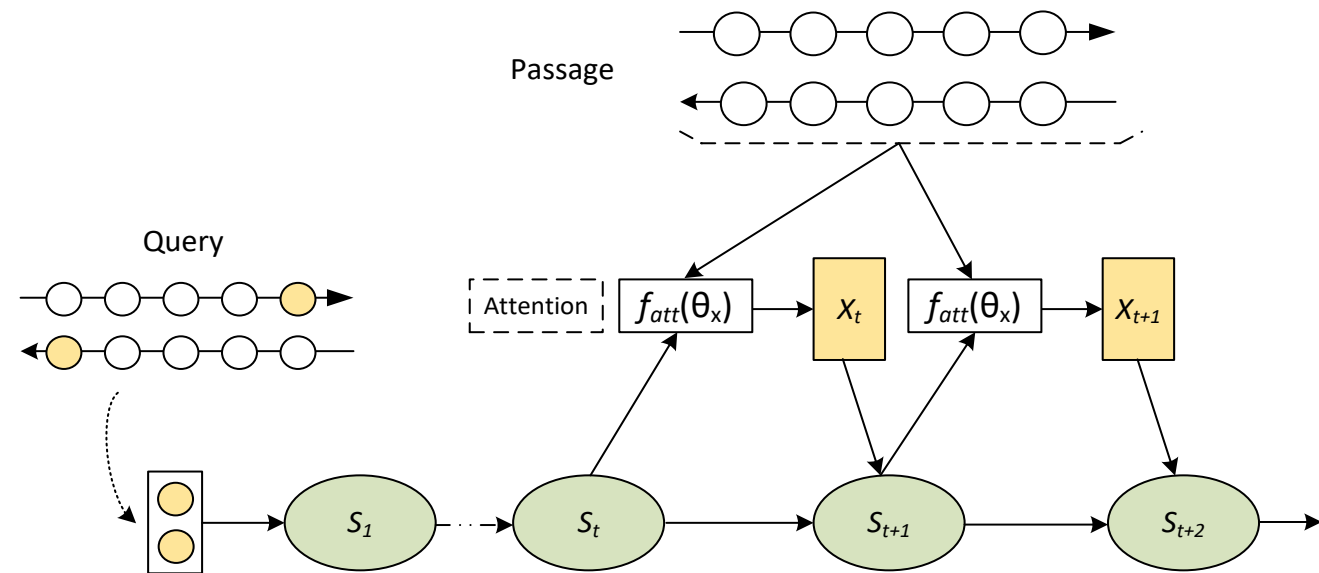
Single Step Reasoning

[Kadlec et al. 2016, Chen et al. 2016]



Multiple Step Reasoning

[Hill et al. 2016, Trischler et al. 2016, Dhingra et al. 2016, Sordoni et al. 2016, Kumar et al. 2016]



How many steps?

Different levels of complexity

Easier

Query

Who was the 2015 NFL MVP?

Passage

The Panthers finished the regular season with a 15–1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP).

Answer

Cam Newton

Harder

Query

Who was the #2 pick in the 2011 NFL Draft?

Passage

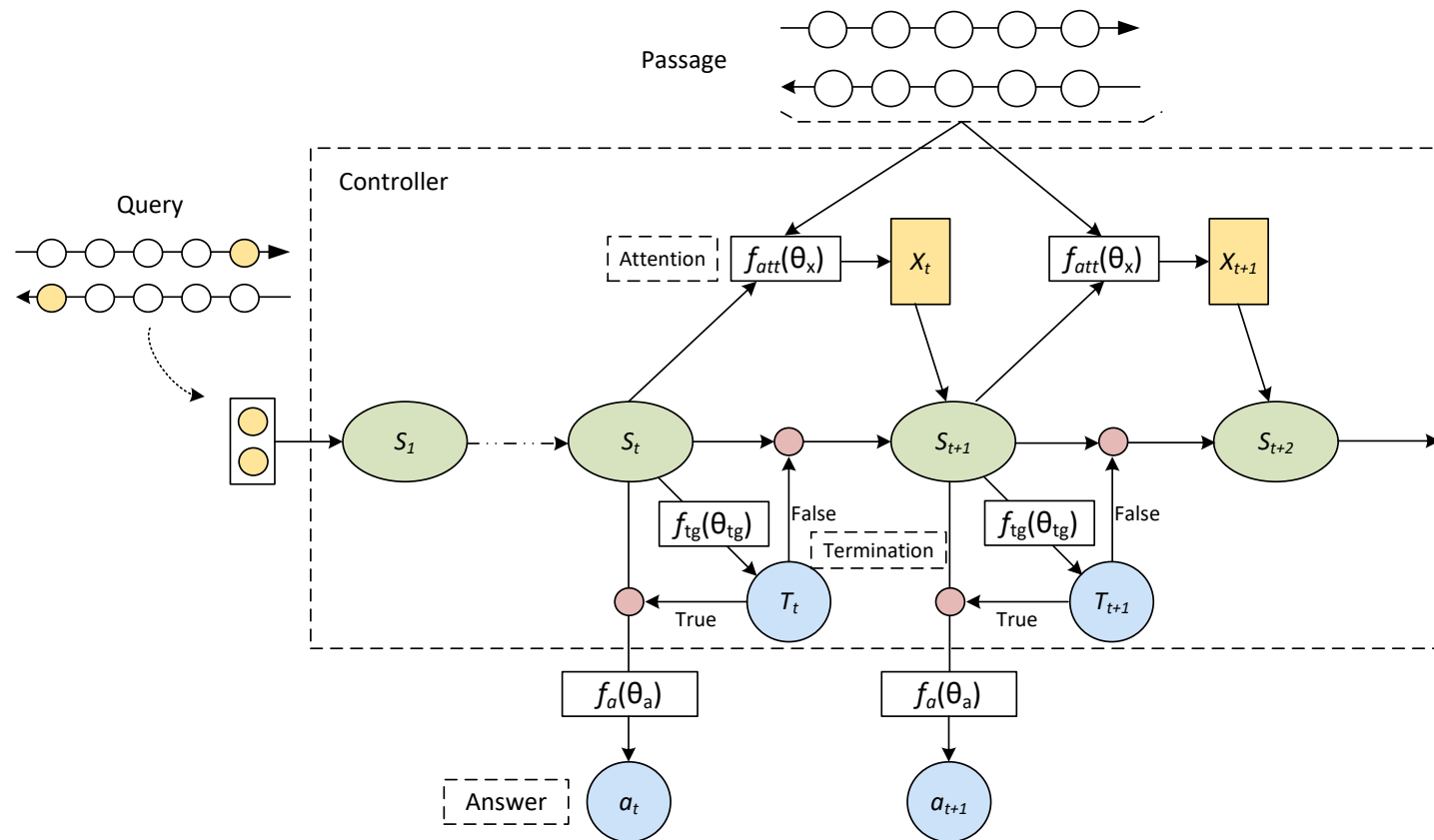
Manning was the #1 selection of the 1998 NFL draft, while Newton was picked first in 2011. The matchup also pits the top two picks of the 2011 draft against each other: Newton for Carolina and Von Miller for Denver.

Answer

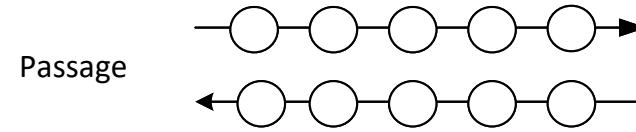
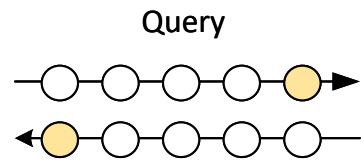
Von Miller

ReasonNet: Learning to Stop Reading

- Dynamic termination based on the complexity of query and passage
- Instance-based RL objectives



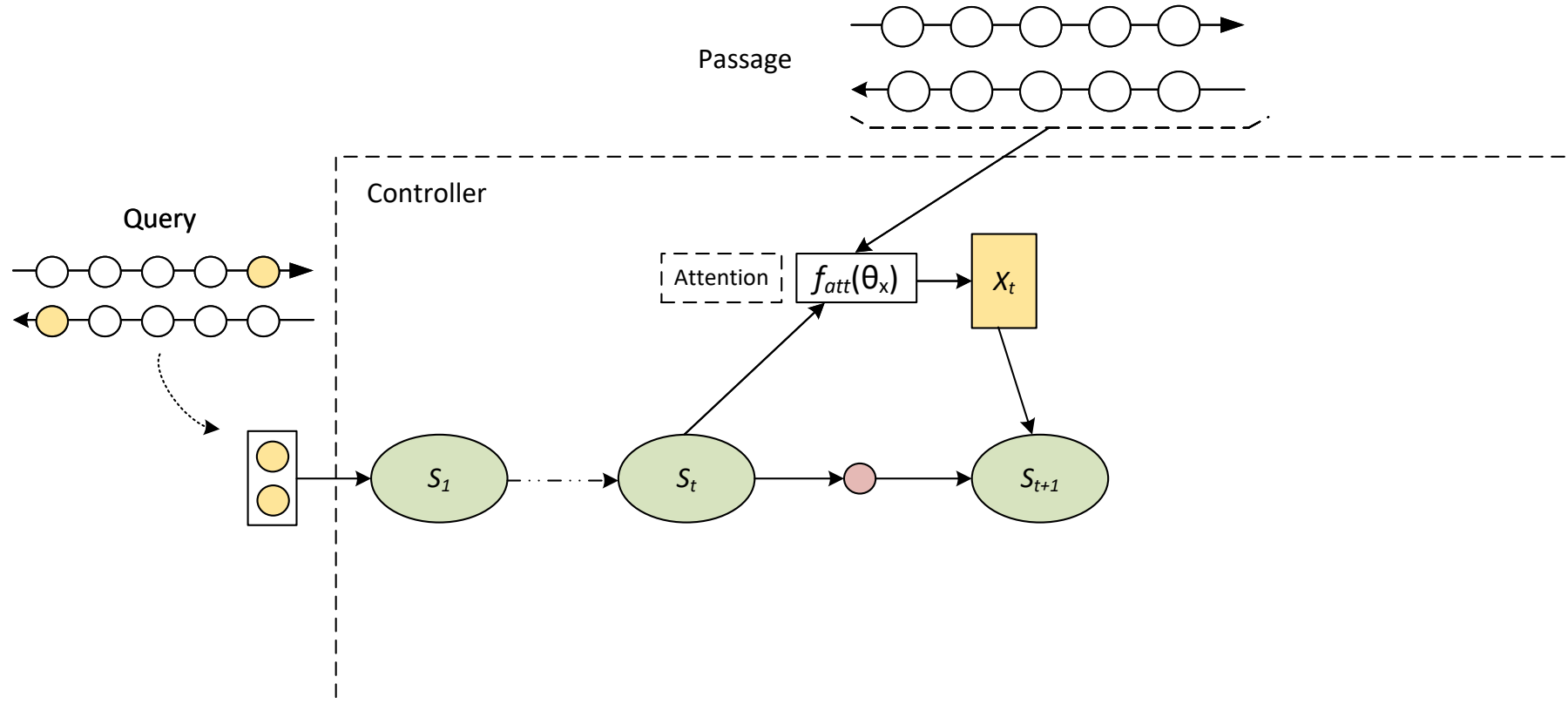
ReasonNet Architecture



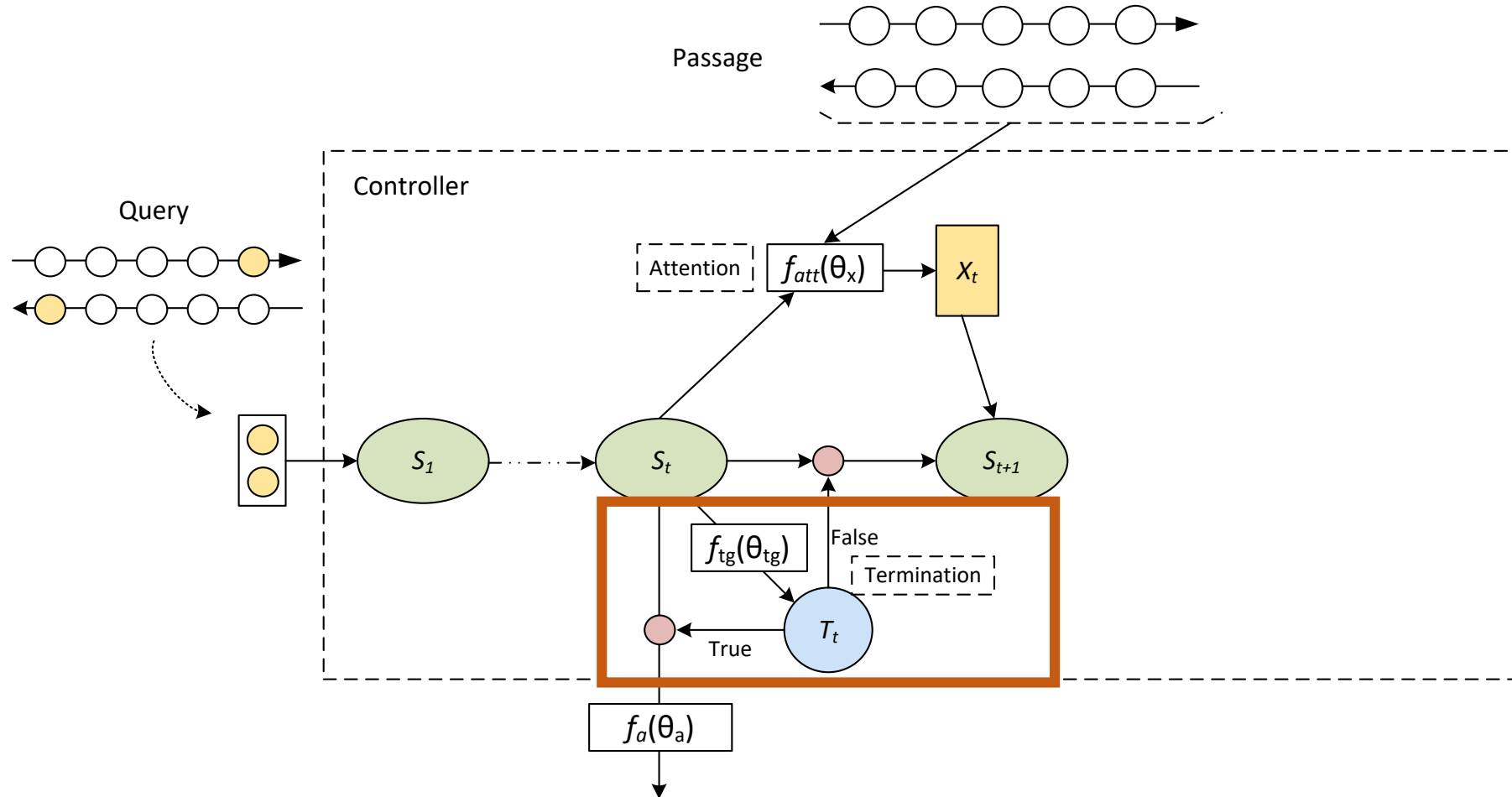
ReasonNet Architecture



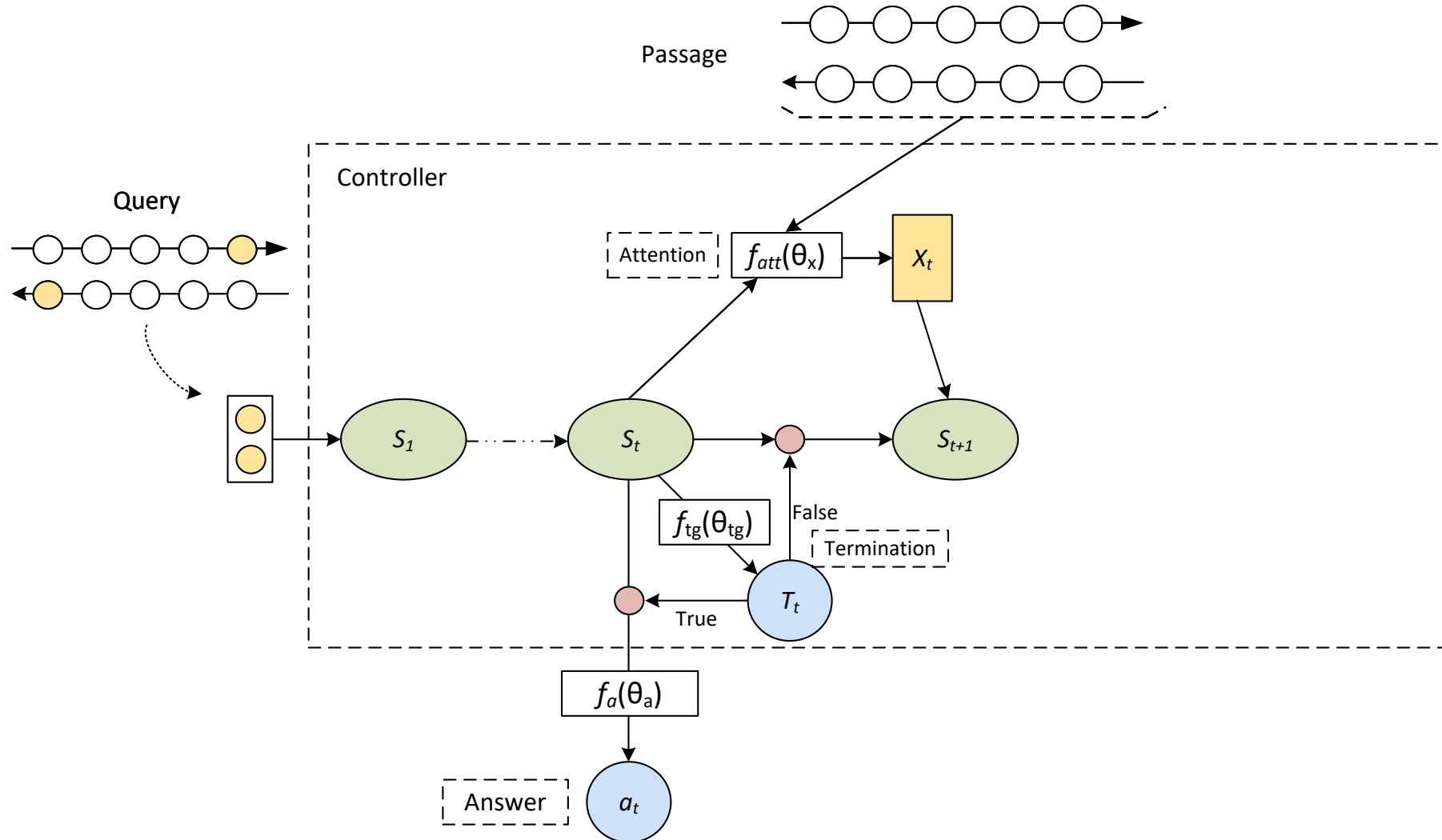
ReasonNet Architecture



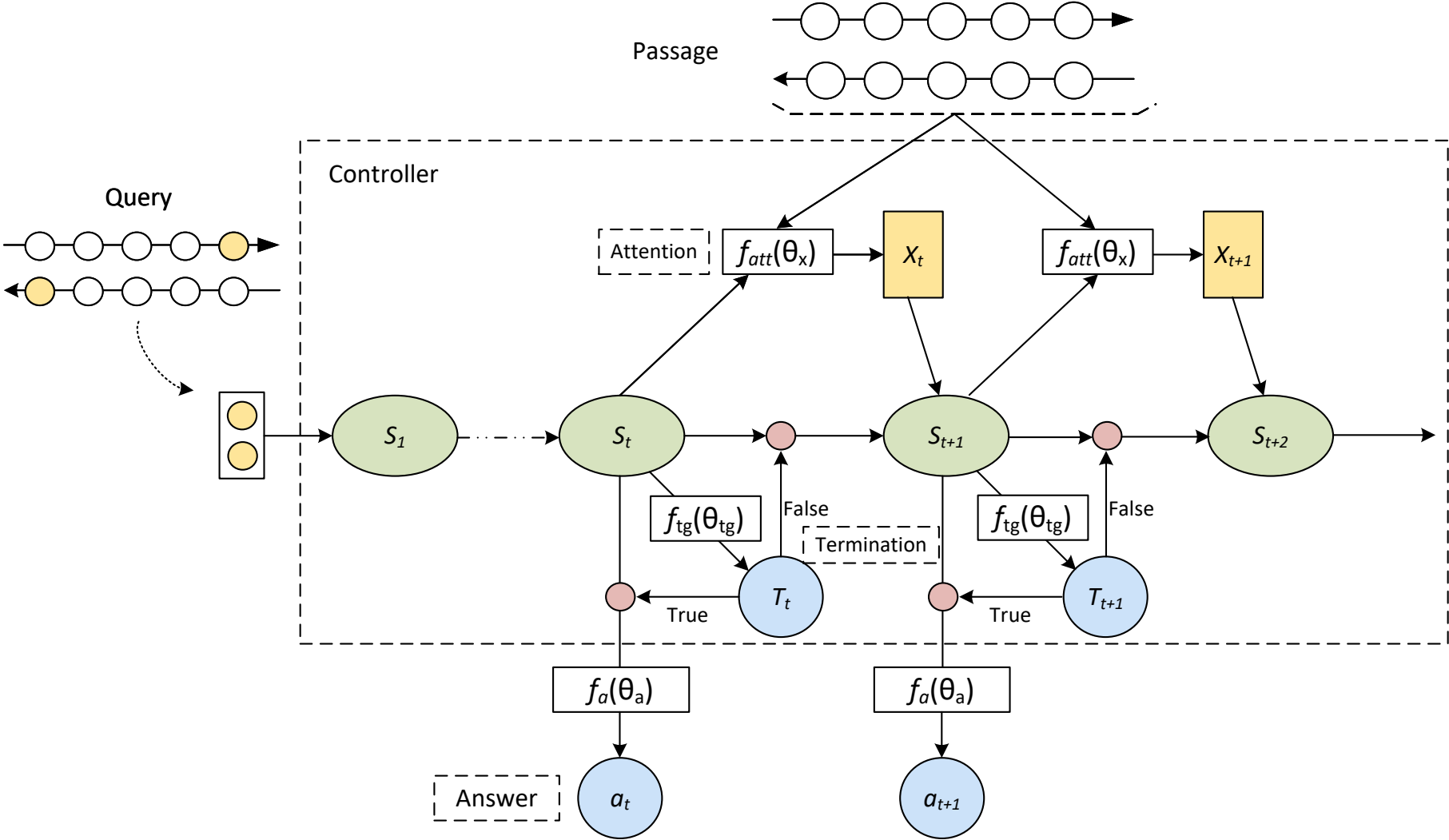
ReasonNet Architecture



ReasonNet Architecture



ReasonNet Architecture



RL Objectives

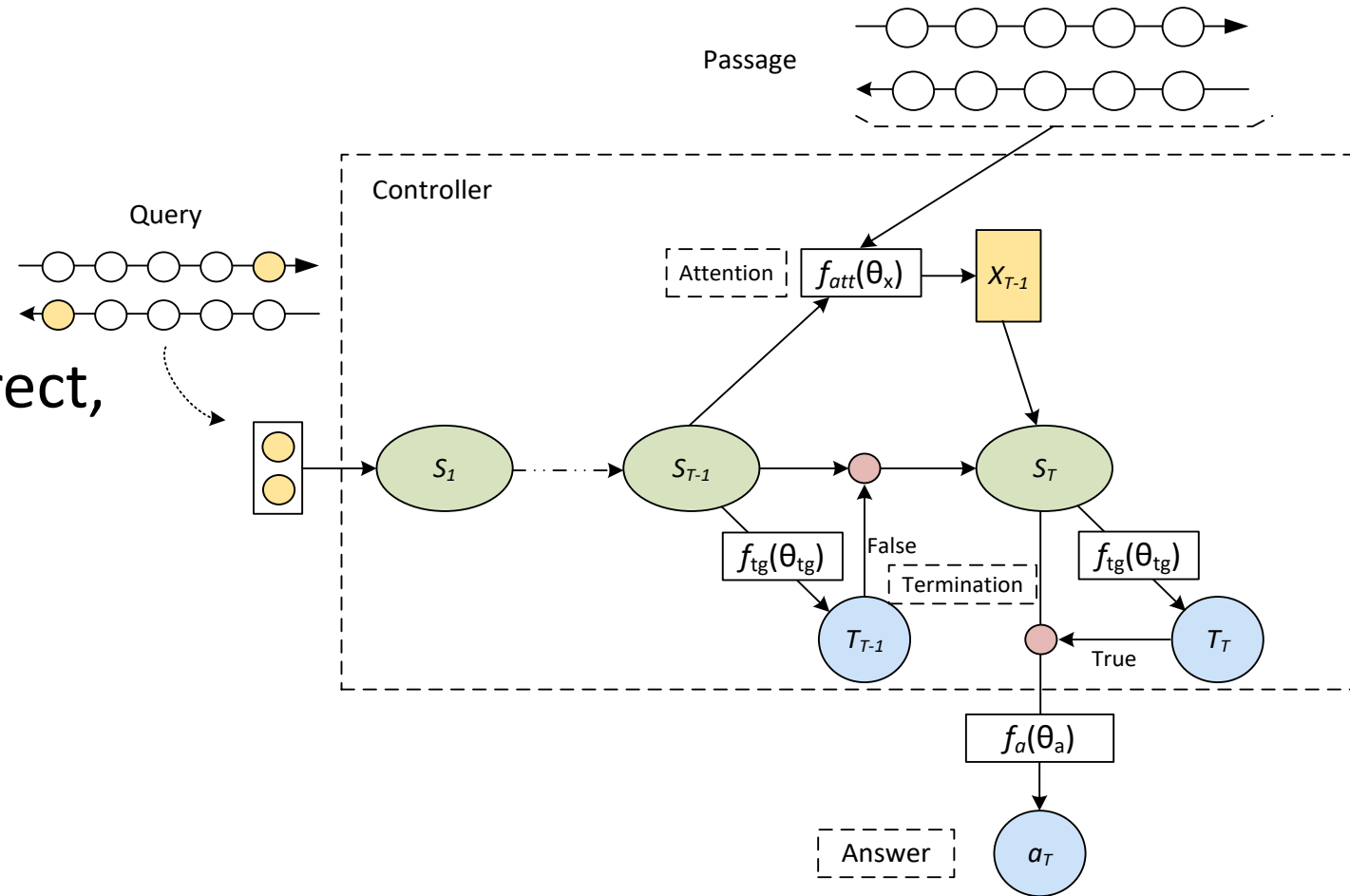
- Action: termination, answer
- Reward: 1 if the answer is correct, 0 otherwise (Delayed Reward)
- Expected total reward

$$J(\theta) = \mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\sum_{t=1}^T r_t \right]$$

- REINFORCE algorithm

$$\nabla_{\theta} J(\theta) = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^{\dagger}} \pi(t_{1:T}, a_T; \theta) [\nabla_{\theta} \log \pi(t_{1:T}, a_T; \theta) (r_T - b)]$$

$$b = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^{\dagger}} \pi(t_{1:T}, a_T; \theta) r_T \quad \text{Instance-based baseline}$$



CNN / Daily Mail Reading Comprehension Task

Query passenger [@placeholder](#) , 36 , died at the scene

Passage

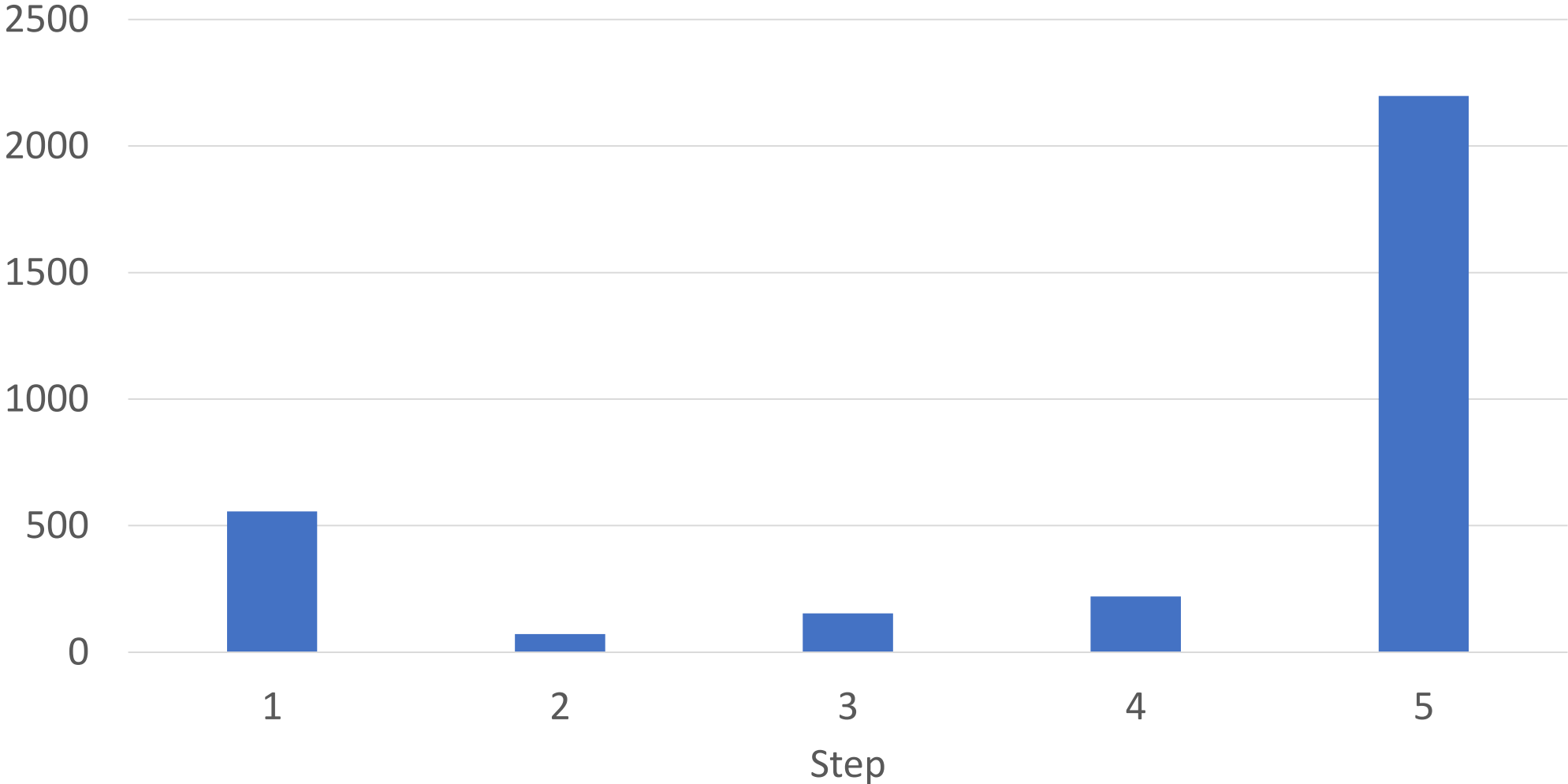
(@entity0) what was supposed to be a fantasy sports car ride at @entity3 turned deadly when a @entity4 crashed into a guardrail . the crash took place sunday at the @entity8 , which bills itself as a chance to drive your dream car on a racetrack . the @entity4 's passenger , 36 - year - old @entity14 of @entity15 , @entity16 , died at the scene , @entity13 said . the driver of the @entity4 , 24 - year - old @entity18 of @entity19 , @entity16 , lost control of the vehicle , the @entity13 said . he was hospitalized with minor injuries . @entity24 , which operates the @entity8 at @entity3 , released a statement sunday night about the crash . " on behalf of everyone in the organization , it is with a very heavy heart that we extend our deepest sympathies to those involved in today 's tragic accident in @entity36 , " the company said . @entity24 also operates the @entity3 -- a chance to drive or ride in @entity39 race cars named for the winningest driver in the sport 's history . @entity0 's @entity43 and @entity44 contributed to this report .

Answer

[@entity14](#)

Termination Step Histogram

CNN Dataset



Results

	Accuracy (%)	CNN	Daily Mail
Single step	Attentive Reader [Hermann et al. 15]	63.0	69.0
	AS Reader [Kadlec et al. 16]	69.5	73.9
	Stanford AR [Chen et al. 16]	72.4	75.8
	Iterative AR [Sordoni et al. 16]	73.3	-
Multiple steps	EpiReader [Trischler et al. 16]	74.0	-
	GA Reader [Dhingra et al. 16]	73.8	75.7
	ReasonNet (Sep 17 2016)	74.7	76.6

The background features a complex network of thin, multi-colored lines (red, orange, teal, and grey) connecting small white plus-sign nodes. This network is overlaid on a dark background with numerous overlapping, semi-transparent circles in various shades of brown, orange, and teal. A large, semi-transparent teal rectangle is positioned on the left side of the image, containing the word "Conclusion" in a black, sans-serif font.

Conclusion

CNTK's approach to the two key questions:

- **efficient network authoring**

- **networks as function objects**, well-matching the nature of DNNs
- focus on **what, not how**
- familiar syntax and flexibility in **Python**

- **efficient execution**

- graph → parallel program through **automatic minibatching**
- **symbolic loops** with dynamic scheduling
- unique **parallel training algorithms** (1-bit SGD, Block Momentum)



on our roadmap

- integration with C#/.Net, R, Keras, HDFS, and Spark
 - continued C#/.Net integration; R
 - Keras back-end
 - HDFS
 - Spark
- technology
 - handle models too large for GPU
 - optimized nested recurrence
 - ASGD
 - 16-bit support, ARM, FPGA



Cognitive Toolkit: deep learning like Microsoft product groups

- ease of use

- *what*, not *how*
- powerful library
- minibatching is automatic

- fast

- optimized for NVidia GPUs & libraries
- easy yet best-in-class multi-GPU/multi-server support

- flexible

- Python and C++ API, powerful & composable

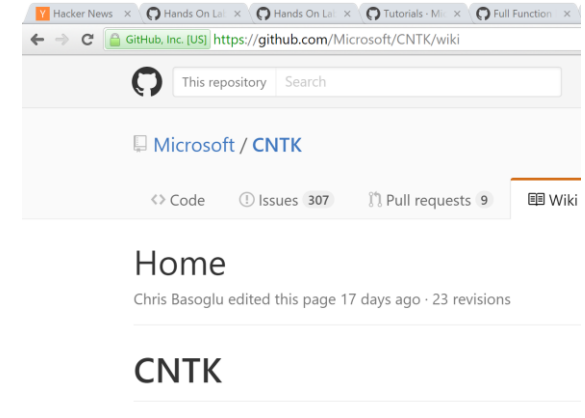
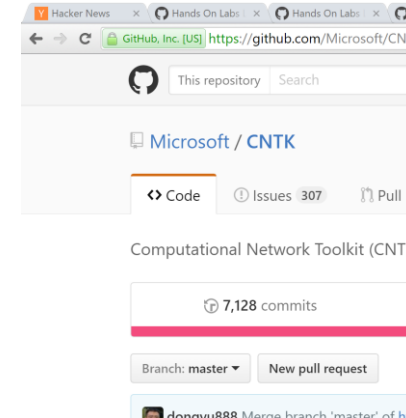
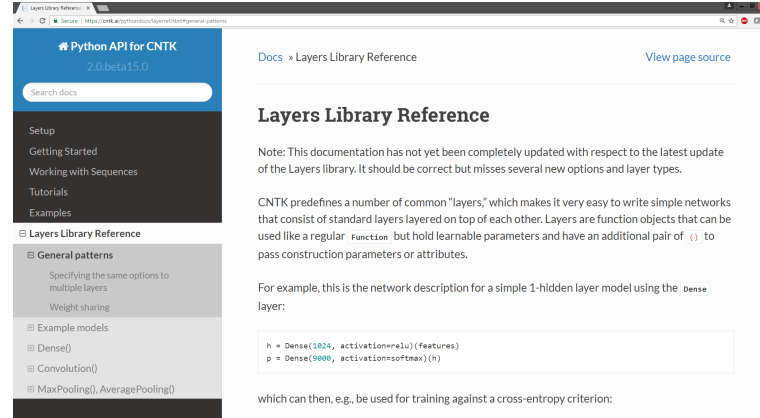
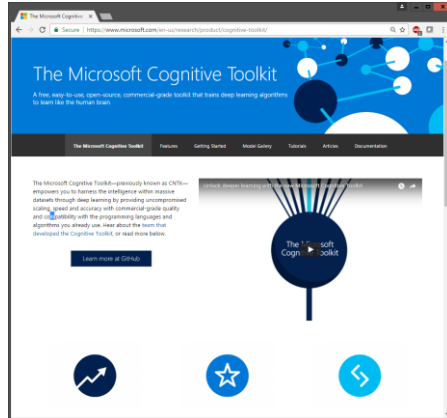
- 1st-class on Linux and Windows

- train like MS product groups: internal=external version



Cognitive Toolkit: democratizing the AI tool chain

- Web site: <https://cntk.ai/>
- Docs: <https://cntk.ai/pythondocs>
- Github: <https://github.com/Microsoft/CNTK>
- Wiki: <https://github.com/Microsoft/CNTK/wiki>



www.stackoverflow.com with cntk tag

