Scalable deep learning with Cognitive Toolkit (CNTK) - Lab

Sayan Pathak
Principal ML Scientist

With many contributors:
Agenda

• Cognitive Toolkit Intro (morning session)
• Tutorials –
  ▪ Intro Read-Model-Train-Test-Predict (Basic Conv Net)
  ▪ Image recognition:
    ✓ VGG and ResNet (Hands-on)
    ✓ DCGAN (code-walk through)
  ▪ Sequence handling
    ✓ Intro to recurrence
    ✓ Text classification (Hands-on)
    ✓ Sequence to Sequence with Attention (Code-walk through)
    ✓ Machine comprehension with ReasoNet (Time permitting)
MNIST Handwritten Digits (OCR)

Handwritten Digits

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

Data set of handwritten 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 %

Corresponding Labels

{ 1 5 4 3
  5 3 5 3
  5 9 0 6 }
Multi-layer perceptron

Deep Model

Weights

0.08 0.08 0.10 0.17 0.11 0.09 0.08 0.08 0.13 0.01

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

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

Label One-hot encoded (\( \gamma \))

Predicted Probabilities (\( p \))

Model (\( w, b \))

Cross entropy error

Label: 1 5 4 3
5 3 5 3
5 9 0 6

Predicted Probabilities:
0.08 0.08 0.10 0.17 0.11 0.09 0.08 0.08 0.13 0.01

Cross-entropy error formula:
\[ ce = -\sum_{j=0}^{9} y_j \log(p_j) \]
Train workflow

- **Data Sampler**: Features \(x\), Labels \(Y\)
- **Training Data**
- **Train (learner)**
- **Reporting**
- **Model**: \(z(\text{params})\)
- **Train more?**

Flow:
- Data Sampler to Training Data
- Training Data to Train (learner)
- Train (learner) to Model \(z(\text{params})\)
- Model \(z(\text{params})\) to params
- params to Train (learner)
- params to update params
- Report

Parameters:
- \(\text{params}\)
- \(\text{iterations}\)
- \(\text{Loss}\)
Train workflow

Input feature (X: 128 x 784)

Model Parameters

weights

Bias

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(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 - W & b
Train (learner) → Data Sampler (Features (x), Labels (Y)) → Train (learner) → Reporting → Test workflow → Model (z(params)) → trained params → Test (Y) → Reporting → Train more? → Data Sampler (Features (x), Labels (Y)) → Data Sampler (Features (x), Labels (Y)) → Test (Y) → Reporting → Train more?
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(z, Y)

Error
   classification_error(z, Y)

Trainer.test_minibatch({X, Y})
Prediction workflow

Input feature (new $X: 1 \times 784$)

Any MNIST

Model $(w, b)$

Model.eval(new $X$)

Predicted Softmax Probabilities (predicted_label)

[numpy.argmax(predicted_label) for predicted_label in predicted_labels]

[9]
Prediction workflow

Input feature (new X: 25 x 784)

Model (w, b)

Model.eval(new X)

Predicted Softmax Probabilities (predicted_label)

\[
[\text{numpy.argmax(predicted_label)} \text{ for predicted_label in predicted_labels}]
\]

\[ [9, 5, 8, \ldots, 2] \]
Recurrence Primer
Sequences (many to one)

Problem: Time series prediction with IOT data

Output
(Y: n x future prediction)

Model
Rec = Recurrence

Input feature
(X: n x 14 data pnts)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Sequences (many to many + 1:1)

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

0 0 0 Date

Rec Rec Rec Rec Rec

show burbank to seattle flights tomorrow

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Forecasting

\[ y^* = mx + b \]

- Solar panel output (in W)
- Average day temperature (in °F)

**Model**

- Day-1 (history)
- \( y \) (Sunday)
- \( x \) (Monday)
- \( y \) (Monday)
Recurrence

\[ \tilde{y}(t=2) \]
\[ h(t=1) \rightarrow x(t=1) \]
\[ \text{Model} \]
\[ \tilde{y}(t=3) \]
\[ h(t=2) \rightarrow x(t=2) \]
\[ \text{Model} \]
\[ \tilde{y}(t=4) \]
\[ h(t=3) \rightarrow x(t=3) \]
\[ \text{Model} \]
\[ \ldots \]
\[ \tilde{y}(t=10) \]
\[ h(t=9) \rightarrow x(t=9) \]
\[ \text{Model} \]

\[ x(t) \]: Input \((n\text{-dimensional array})\) at time \(t\)
\[ \tilde{y}(t) \]: Output \((c\text{-dimensional array})\) at time \(t\)
\[ h(t) \]: Internal State \([m\text{-dimensional array}]\) at time \(t\) a.k.a history

**Input:**

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

Model \( x(t) \) \( \rightarrow \) \( y(t) \)

\( x(t=1) \) \( \rightarrow \) \( h(t=1) \) \( \rightarrow \) \( x(t=2) \) \( \rightarrow \) \( h(t=2) \) \( \rightarrow \) \( x(t=3) \)
Recurrence

\[
\tilde{h}(t) = D_i = n + m \\
O = m \\
a = \tanh
\]

\[
\tilde{h}(t - 1) = (m - \text{dim})
\]

\[
\tilde{y}(t) = \text{softmax}
\]

\[
\tilde{x}(t) = (\tilde{x}(t) | \tilde{h}(t - 1))
\]

\[
\tilde{x}^* = (\tilde{x}(t) | \tilde{h}(t - 1))
\]

\[
W \tilde{x}^* + \tilde{b}
\]

\[
(W, \tilde{b}) \text{ Same parameters are shared and updated across time steps}
\]

Internal State \( \tilde{h}(t - 1) \) (m-dim)
Recurrence (Vanishing Gradients)
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 BBC is a single set of \((W, \hat{b})\) has limited memory.
Language Understanding
Sequences (many to many)

Problem: Tagging entities in Air Traffic Controller (ATIS) data
ATIS data
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
<table>
<thead>
<tr>
<th>Sequence Id</th>
<th>Input Word (sample)</th>
<th>Word Index (in vocabulary)</th>
<th>Word Label</th>
<th>Label Index (S2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td># BOS</td>
<td>178:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># please</td>
<td>688:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># give</td>
<td>449:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># me</td>
<td>581:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># the</td>
<td>827:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># flights</td>
<td>429:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># from</td>
<td>444:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># boston</td>
<td>266:1</td>
<td># B-fromloc.city_name</td>
<td>48:1</td>
</tr>
<tr>
<td>19</td>
<td># to</td>
<td>851:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># pittsburgh</td>
<td>682:1</td>
<td># B-toloc.city_name</td>
<td>78:1</td>
</tr>
<tr>
<td>19</td>
<td># on</td>
<td>654:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># thursday</td>
<td>845:1</td>
<td># B-depart_date.day_name</td>
<td>26:1</td>
</tr>
<tr>
<td>19</td>
<td># of</td>
<td>646:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># next</td>
<td>621:1</td>
<td># B-depart_date.date_relative</td>
<td>25:1</td>
</tr>
<tr>
<td>19</td>
<td># week</td>
<td>910:1</td>
<td># O</td>
<td>128:1</td>
</tr>
<tr>
<td>19</td>
<td># EOS</td>
<td>179:1</td>
<td># O</td>
<td>128:1</td>
</tr>
</tbody>
</table>

**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 943 words)  
**Label Index:** `###:1` indicates the position of the corresponding tag in tag index (total 129 tags)
Sequence Tagging (Input / Label Pre-processing)

Create a numerical representation of the input words

For MNIST data:

For each word - One-hot representation is a vector with 943 elements

For each label - one-hot representation is a vector with 129 elements
Embedding

One-hot Encoding
Numerical representation of text

Word Embedding
Technique to map words or phrases to vector of real numbers.
Maps one-hot encoded vector to a lower dimensional space

Linear Embedding
Multiply a matrix with one-hot encoded vector \((W_e \vec{X}^T)\)
- \(\vec{X}^T\): vector of size 1 x 943
- \(W_e\): matrix of size 150 x 943

Popular Embedding
- GloVe (https://en.wikipedia.org/wiki/GloVe_(machine_learning))
Model
\[ y(t) \text{ class label} \]

Model

\[ \tilde{x}(t) \text{ Text token} \]

\[ h(t-1) \]

Dense

LSTM

Embedding

\[ E \]

Recurrence

\[ \tilde{h}(t) \]

\[ y(t) \]

\[ \tilde{x}(t) \]

\[ 0 \quad 0 \quad \cdots \quad 1 \quad \cdots \quad 0 \]

\[ d = \text{sigmoid} \quad h(t) \]

\[ l = 300 \quad 0 = 129 \quad d = \text{sigmoid} \]

\[ i = 150 \quad o = 200 \]

\[ i = 943 \quad o = 150 \]
Text classification

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

BOS from boston to pittsburgh on thursday of next week
Text classification

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

<table>
<thead>
<tr>
<th>#</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS</td>
<td># O</td>
</tr>
<tr>
<td>from</td>
<td># O</td>
</tr>
<tr>
<td>boston</td>
<td># B-fromloc.city_name</td>
</tr>
<tr>
<td>to</td>
<td># O</td>
</tr>
<tr>
<td>pittsburgh</td>
<td># B-toloc.city_name</td>
</tr>
<tr>
<td>on</td>
<td># O</td>
</tr>
<tr>
<td>thursday</td>
<td># B-depart_date.day_name</td>
</tr>
<tr>
<td>of</td>
<td># O</td>
</tr>
<tr>
<td>next</td>
<td># B-depart_date.date_relative</td>
</tr>
<tr>
<td>week</td>
<td># O</td>
</tr>
<tr>
<td>EOS</td>
<td># O</td>
</tr>
</tbody>
</table>

'BOS from boston to Pittsburgh on Thursday of next week EOS'

Input feature (1 x 11 x (1x943))
Loss function:

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

Label One-hot encoded (\( \hat{y}(t) \))

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>...</th>
<th>0</th>
<th>0</th>
<th>...</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Model

\( \hat{x}(t) \)

Predicted Probabilities (p)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>...</th>
<th></th>
<th></th>
<th>...</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

943
Train workflow

Data Sampler
Features (x), Labels (Y)

Training Data

Train (learner)

Model
z(\text{params})

Reporting

Train more?

\text{params}

\text{update params}

Loss

iterations

\text{Y}
Train Workflow

**Input feature** ($96 \times \tilde{x}(t)$)

- #1
- #2
- #3
- #96

One-hot encoded Label

- $Y: 96 \times 129$ sample
- Or word in sequence

**ATIS Train**

96 samples (mini-batch)

$$z = \text{model}():$$
$$\text{return}$$
$$\text{Sequential([}$$
$$\text{Embedding(emb_dim=150),}$$
$$\text{Recurrence(LSTM(hidden_dim=300),}$$
$$\text{go_backwards=False),}$$
$$\text{Dense(num_labels = 129)}$$
$$\text{])}$$

**Loss**
$$\text{cross_entropy_with_softmax}(z, Y)$$

**Error**
$$\text{classification_error}(z, Y)$$

**Trainer** ($\text{model}, (\text{loss, error), learner}$)

- $$\text{Trainer}.\text{train_minibatch}({X, Y})$$
- **Learner**
  - Adam, adagrad etc, are solvers to estimate
Train more?

Data Sampler
Features (x), Labels (Y)

Train (learner)

Reporting

Model
$z(\text{params})$

Loss

iterations

Model final

Test more?

Data Sampler
Features (x), Labels (Y)

Test Data

Test workflow

Data Sampler
Features (x), Labels (Y)

Train Data

Test

Reporting

Test

Labels (Y)

trained params

update params

params

Train more?
Test workflow

Input feature (32 x \( \vec{x}(t) \))

One-hot encoded Label
\( \vec{Y}: 32 \times 129/\text{sample} \)
Or word in sequence)

\[
z = \text{model}():
\begin{align*}
&\quad \text{return} \\
&\quad \text{Sequential}([ \\
&\quad\quad \text{Embedding(emb_dim=150),} \\
&\quad\quad \text{Recurrence(LSTM(hidden_dim=300),} \\
&\quad\quad\quad \text{go_backwards=False),} \\
&\quad\quad \text{Dense(num_labels = 129)}]
\])
\end{align*}
\]

Loss
\[
\text{cross_entropy_with_softmax}(z, \vec{Y})
\]

Error
\[
\text{classification_error}(z, \vec{Y})
\]

Trainer.test_minibatch({\( \vec{X}, \vec{Y} \)} Lei-Wei Component
Prediction workflow

Any Data string

'BOS flights from new york to seattle EOS'

Input feature (new $X: 1 \times 8 \times (1\times943)$)

Model.eval(new $X$)

Predicted Softmax Probabilities

Output prediction ($1 \times 8 \times (1\times129)$)
Sequence to Sequence
Sequences (many to many)

```
<s>
hallo
wie
geht
es
dir
</s>
```

[Diagram showing the process of translating from English to German using RNNs]

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
First described in the context of machine translation

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:

✓ 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
  • The decoder’s hidden state is then initialized with this thought vector
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>"

```python
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 \( h \) is 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:
     $$a_i = \text{softmax}(u_i)$$
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:
  \[ D = d + d' \]
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 ReasoNet
Problem Definition

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

<table>
<thead>
<tr>
<th>Query</th>
<th>Who is the producer of Doctor Who?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passage</td>
<td>Doctor Who is a British science-fiction television programme produced by the <strong>BBC</strong> 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.</td>
</tr>
<tr>
<td>Answer</td>
<td><strong>BBC</strong></td>
</tr>
</tbody>
</table>
Related Work

**Single Step Reasoning**

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

**Multiple Step Reasoning**


How many steps?
## Different levels of complexity

<table>
<thead>
<tr>
<th>Easier</th>
<th><strong>Query</strong></th>
<th>Who was the 2015 NFL MVP?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Passage</strong></td>
<td>The Panthers finished the regular season with a 15–1 record, and quarterback <strong>Cam Newton</strong> was named the NFL Most Valuable Player (MVP).</td>
<td></td>
</tr>
<tr>
<td><strong>Answer</strong></td>
<td>Cam Newton</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Harder</th>
<th><strong>Query</strong></th>
<th>Who was the #2 pick in the 2011 NFL Draft?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Passage</strong></td>
<td>Manning was the #1 selection of the 1998 NFL draft, while <strong>Newton</strong> was picked first in 2011. The matchup also pits the top two picks of the 2011 draft against each other: Newton for Carolina and <strong>Von Miller</strong> for Denver.</td>
<td></td>
</tr>
<tr>
<td><strong>Answer</strong></td>
<td>Von Miller</td>
<td></td>
</tr>
</tbody>
</table>
ReasoNet: Learning to Stop Reading

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

\[
\nabla_{\theta} J(\theta) = \sum_{(t_1:T, a_T) \in \mathbb{A}^\dagger} \pi(t_1:T, a_T; \theta) \left[ \nabla_{\theta} \log \pi(t_1:T, a_T; \theta) (r_T - b) \right]
\]

\[
b = \sum_{(t_1:T, a_T) \in \mathbb{A}^\dagger} \pi(t_1:T, a_T; \theta) r_T
\]

Instance-based baseline
### CNN / Daily Mail Reading Comprehension Task

<table>
<thead>
<tr>
<th>Query</th>
<th>passenger @placeholder , 36 , died at the scene</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passage</td>
<td>( @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. &quot;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,&quot; 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.</td>
</tr>
<tr>
<td>Answer</td>
<td>@entity14</td>
</tr>
</tbody>
</table>
Termination Step Histogram

CNN Dataset

Step

0 500 1000 1500 2000 2500

1 2 3 4 5
### Results

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>CNN</th>
<th>Daily Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attentive Reader [Hermann et al. 15]</td>
<td>63.0</td>
<td>69.0</td>
</tr>
<tr>
<td>AS Reader [Kadlec et al. 16]</td>
<td>69.5</td>
<td>73.9</td>
</tr>
<tr>
<td>Stanford AR [Chen et al. 16]</td>
<td>72.4</td>
<td>75.8</td>
</tr>
<tr>
<td>Iterative AR [Sordoni et al. 16]</td>
<td>73.3</td>
<td>-</td>
</tr>
<tr>
<td>EpiReader [Trischler et al. 16]</td>
<td>74.0</td>
<td>-</td>
</tr>
<tr>
<td>GA Reader [Dhingra et al. 16]</td>
<td>73.8</td>
<td>75.7</td>
</tr>
<tr>
<td><strong>ReasoNet</strong> (Sep 17 2016)</td>
<td>74.7</td>
<td>76.6</td>
</tr>
</tbody>
</table>

**Single step**

**Multiple steps**
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

• 1\textsuperscript{st}-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

Ask Questions: www.stackoverflow.com with cntk tag