Overview for today

• Natural Language Processing with NNs [~15m]
  – Supervised models

• Unsupervised Learning [~45m]

• Memory in Neural Nets [~30m]
Natural Language Processing

Slides from:

Jason Weston  Tomas Mikolov  Antoine Bordes  Wojciech Zaremba
NLP

• Many different problems
  – Language modeling
  – Machine translation
  – Q & A

• Recent attempts to address with neural nets
  – Yet to achieve same dramatic gains as vision/speech
Language modeling

- Natural language is a sequence of sequences
- Some sentences are more likely than others:
  - “How are you?” has a high probability
  - “How banana you?“ has a low probability
Recurrent Neural Network Language Models

**Key idea:** input to predict next word is current word plus context fed-back from previous word (i.e. remembers the past with recurrent connection).

Figure: *Recurrent neural network based LM*

Recurrent neural network based language model. Mikolov et al., Interspeech, '10.

[Slide: Antoine Border & Jason Weston, EMNLP Tutorial 2014]
Recurrent neural networks - schema

My name is Wojciech

name

is

Wojciech
Backpropagation through time

• The intuition is that we unfold the RNN in time

• We obtain deep neural network with shared weights $U$ and $W$
Backpropagation through time

• We train the unfolded RNN using normal backpropagation + SGD

• In practice, we limit the number of unfolding steps to 5 – 10

• It is computationally more efficient to propagate gradients after few training examples (batch mode)
**NNLMS vs. RNNS: Penn Treebank Results (Mikolov)**

<table>
<thead>
<tr>
<th>Model</th>
<th>Weight</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-gram with Good-Turing smoothing (GT3)</td>
<td>0</td>
<td>165.2</td>
</tr>
<tr>
<td>5-gram with Kneser-Ney smoothing (KN5)</td>
<td>0</td>
<td>141.2</td>
</tr>
<tr>
<td>5-gram with Kneser-Ney smoothing + cache</td>
<td>0.0792</td>
<td>125.7</td>
</tr>
<tr>
<td>Maximum entropy model</td>
<td>0</td>
<td>142.1</td>
</tr>
<tr>
<td>Random clusterings LM</td>
<td>0</td>
<td>170.1</td>
</tr>
<tr>
<td>Random forest LM</td>
<td>0.1057</td>
<td>131.9</td>
</tr>
<tr>
<td>Structured LM</td>
<td>0.0196</td>
<td>146.1</td>
</tr>
<tr>
<td>Within and across sentence boundary LM</td>
<td>0.0838</td>
<td>116.6</td>
</tr>
<tr>
<td>Log-bilinear LM</td>
<td>0</td>
<td>144.5</td>
</tr>
<tr>
<td>Feedforward NNLM</td>
<td>0</td>
<td>140.2</td>
</tr>
<tr>
<td>Syntactical NNLM</td>
<td>0.0828</td>
<td>131.3</td>
</tr>
<tr>
<td>Combination of static RNNLMs</td>
<td>0.3231</td>
<td>102.1</td>
</tr>
<tr>
<td>Combination of adaptive RNNLMs</td>
<td>0.3058</td>
<td>101.0</td>
</tr>
<tr>
<td><strong>ALL</strong></td>
<td>1</td>
<td><strong>83.5</strong></td>
</tr>
</tbody>
</table>

Recent uses of NNLMs and RNNs to improve machine translation:
Also Kalchbrenner ’13, Sutskever et al., ’14., Cho et al., ’14. .
Language modelling – RNN samples

The meaning of life is that only if an end would be of the whole supplier. Widespread rules are regarded as the companies of refuses to deliver. In balance of the nation’s information and loan growth associated with the carrier thrifts are in the process of slowing the seed and commercial paper.

[Slide: Wojciech Zaremba]
More depth gives more power
LSTM - Long Short Term Memory

[Hochreiter and Schmidhuber, Neural Computation 1997]

- Ad-hoc way of modelling long dependencies
- Many alternative ways of modelling it
- Next hidden state is modification of previous hidden state (so information doesn’t decay too fast).

For simple explanation, see [Recurrent Neural Network Regularization, Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals, arXiv 1409.2329, 2014]
RNN-LSTMs for Machine Translation

Sequence to Sequence Learning with Neural Networks,
Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014


[Slide: Wojciech Zaremba]
Visualizing Internal Representation

t-SNE projection of network state at end of input sentence

Sequence to Sequence Learning with Neural Networks, Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014
Translation - examples

- FR: Les avionneurs se querellent au sujet de la largeur des sièges alors que de grosses commandes sont en jeu

- Google Translate: Aircraft manufacturers are quarreling about the seat width as large orders are at stake

- LSTM: Aircraft manufacturers are concerned about the width of seats while large orders are at stake

- Ground Truth: Jet makers feud over seat width with big orders at stake

[Sequence to Sequence Learning with Neural Networks, Ilya Sutskever, Oriol Vinyals, Quoc Le, NIPS 2014]

[Slide: Wojciech Zaremba]
Many recent works on this:
• Baidu/UCLA: Explain Images with Multimodal Recurrent Neural Networks
• Toronto: Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models
• Berkeley: Long-term Recurrent Convolutional Networks for Visual Recognition and Description
• Google: Show and Tell: A Neural Image Caption Generator
• Stanford: Deep Visual-Semantic Alignments for Generating Image Description
• UML/UT: Translating Videos to Natural Language Using Deep Recurrent Neural Networks
• Microsoft/CMU: Learning a Recurrent Visual Representation for Image Caption Generation
• Microsoft: From Captions to Visual Concepts and Back
Image Captioning Examples

Unsupervised Learning
Motivation

• Most successes obtained with supervised models, e.g. Convnets

• Unsupervised learning methods less successful

• But likely to be very important in long-term
Historical Note

• Deep Learning revival started in ~2006
  – Hinton & Salakhudinov Science paper on RBMs

• Unsupervised Learning was focus from 2006–2012

• In ~2012 great results in vision, speech with supervised methods appeared
  – Less interest in unsupervised learning
Arguments for Unsupervised Learning

- Want to be able to exploit unlabeled data
  - Vast amount of it often available
  - Essentially free

- Good regularizer for supervised learning
  - Helps generalization
  - Transfer learning
  - Zero / one-shot learning
Another Argument for Unsupervised Learning

When we’re learning to see, nobody’s telling us what the right answers are — we just look. Every so often, your mother says “that’s a dog”, but that’s very little information.

You’d be lucky if you got a few bits of information — even one bit per second — that way. The brain’s visual system has $10^{14}$ neural connections. And you only live for $10^9$ seconds.

So it’s no use learning one bit per second. You need more like $10^5$ bits per second. And there’s only one place you can get that much information: from the input itself.

— Geoffrey Hinton, 1996
Taxonomy of Approaches

- Autoencoder (most unsupervised Deep Learning methods)
  - RBMs / DBMs
  - Denoising autoencoders
  - Predictive sparse decomposition
- Decoder-only
  - Sparse coding
  - Deconvolutional Nets
- Encoder-only
  - Implicit supervision, e.g. from video
- Adversarial Networks

Loss involves some kind of reconstruction error
Auto-Encoder

Input (Image/Features)

Decoder

Encoder

Output Features

Feed-back / generative / top-down path

Feed-forward / bottom-up path
Auto-Encoder Example 1

- Restricted Boltzmann Machine [Hinton ’02]

\[
\begin{align*}
\sigma(W^T z) & \quad \text{(Binary) Features } z \\
\sigma(Wx) & \quad \text{(Binary) Input } x
\end{align*}
\]
Auto-Encoder Example 2

- Predictive Sparse Decomposition [Ranzato et al., '07]

\[ \sigma(Wx) \]

\[ Dz \]

\[ z \]

Sparse Features \( z \)

Input Patch \( x \)

Encoder filters \( W \)

Sigmoid function \( \sigma(.) \)

Decoder filters \( D \)

\( L_1 \) Sparsity
Auto-Encoder Example 2

- Predictive Sparse Decomposition  [Kavukcuoglu et al., ‘09]

Training

\[
\min_{D,W,z} \left( \| Dz - x \|_2^2 + \lambda |z|_1 + \| \sigma(Wx) - z \|_2^2 \right)
\]
Stacked Auto-Encoders

Two phase training:

1. Unsupervised layer-wise pre-training
2. Fine-tuning with labeled data

[Hinton & Salakhutdinov Science '06]
Training phase 2: Supervised Fine-Tuning

- Remove decoders
- Use feed-forward path
- Gives standard (Convolutional) Neural Network
- Can fine-tune with backprop

[Hinton & Salakhutdinov Science '06]
Effects of Pre-Training

- From [Hinton & Salakhudinov, Science 2006]

Big network

![Big network graph]

Small network

![Small network graph]

See also: Why Does Unsupervised Pre-training Help Deep Learning? Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol
PIERRE-Pascal Vincent, Sammy Bengio, JMLR 2010
Deep Boltzmann Machines

Undirected model

- Class label
- Features
- Features
- Input Image

- Decoder
- Decoder
- Decoder

- Encoder
- Encoder
- Encoder

Both pathways used at train & test time

TD modulation of BU features

Salakhutdinov & Hinton
AISTATS’09
Weaknesses, but all share the fundamental challenge of im-

Grid MRFs:


samples that differ from images in the training dataset.

understand the composition of the shape, it is impossible to

large database of template shapes [5].

shapes while allowing for the right degree of flexibility to

parameterized contour. These have different strengths and

In this section we will review several undirected models

A binary grid-structured MRF defines a distri-

bution, as captured by a Principal Components Analysis

7] that learning will get stuck in local optima. Hence the DBM

parameters (1 + \exp(wv2), but remain computationally tractable. The

Energy of a DBM with two

and so can learn about complex structure in the data using

dependencies between the hidden variables of previous layers

formulation of the RBM in terms of high-order potentials

This involves the same sample-based approximation to the

dataset [10], for instance, are defined in

Learning: (a) RBM on the observed data using stochastic maximum

ence of latent variables; and (3) the tendency of learning to

properties into the two hidden layers.

second layer is to impose global constraints, e.g. with respect

architecture, as captured by a Principal Components Analysis

Figure 3. Sampled shapes.

y=1

shown in Fig.

1 + \exp(wv2), and all

f

h

rbm

h

v

v

v

v

h

h

v

h

v

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v

h

h

v

h

v

v
Variational Auto-Encoder

- [Kingma & Welling, ICLR 2014]

\[ \text{Maximize } \log p(x) - D_{KL} (q(z) \| p(z | x)) \]

(Kingma and Welling, 2014, Rezende et al 2014)

[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]
Decoder-Only Models

• Examples:
  – Sparse coding
  – Deconvolutional Networks [Zeiler & Fergus, ‘10]

• No encoder to compute features

• So need to perform optimization
  – Can be relatively fast
Sparse Coding (Patch-based)

- Over-complete linear decomposition of input $y$ using dictionary $D$

Input

\[ y = 0.3 \times \text{image} + 0.5 \times \text{image} + 0.2 \times \text{image} \]

\[ C(y, D) = \arg\min_z \frac{\lambda}{2} \| Dz - y \|_2^2 + |z|_1 \]

- $\ell_1$ regularization yields solutions with few non-zero elements

- Output is sparse vector: $z = [0, 0.3, 0, \ldots, 0.5, \ldots, 0.2, \ldots, 0]$
Deconvolutional Network Layer

- Convolutional form of sparse coding
  [Zeiler & Fergus, CVPR 2010].
  Also Kavukcuoglu et al. NIPS 2010

\[ y_1 = \sum f_{1,1} z_1 + f_{1,c} z_{1,c} + \ldots + f_{K,1} z_{K,1} + f_{K,c} z_{K,c} \]

Feature Maps

\[ |\cdot|_1 \text{ Sparsity} \]

Input Image Planes

\[ y_1, y_c \]

Filters
Overall Architecture (2 layers)
Generative Models using Convnets

- Learning to Generate Chairs with Convolutional Neural Networks, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, 1411.5928, 2014

- Supervised training of convnet to draw chairs
Some other interesting generative models


Encoder-Only Models

• In vision setting, essentially a convnet trained without explicit class labels

• Learn invariances
  • Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014

• Learn from video
Unsupervised Learning of Transformations

[Unsupervised feature learning by augmenting single images, Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox, NIPS 2014]

• Take patches from images
• For each patch, make lots of perturbed versions
• Treat each patch + perturbed copies as a separate classss
• Train supervised convnet

<table>
<thead>
<tr>
<th>Method</th>
<th>STL-10</th>
<th>CIFAR-10-reduced</th>
<th>CIFAR-10</th>
<th>Caltech-101</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means [6]</td>
<td>60.1 ± 1</td>
<td>70.7 ± 0.7</td>
<td>82.0</td>
<td>—</td>
</tr>
<tr>
<td>Multi-way local pooling [5]</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>77.3 ± 0.6</td>
</tr>
<tr>
<td>Slowness on videos [25]</td>
<td>61.0</td>
<td>—</td>
<td>—</td>
<td>74.6</td>
</tr>
<tr>
<td>Receptive field learning [16]</td>
<td>—</td>
<td>—</td>
<td>[83.11]</td>
<td>75.3 ± 0.7</td>
</tr>
<tr>
<td>Hierarchical Matching Pursuit (HMP) [3]</td>
<td>64.5 ± 1</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Multipath HMP [4]</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>82.5 ± 0.5</td>
</tr>
<tr>
<td>Sum-Product Networks [8]</td>
<td>62.3 ± 1</td>
<td>—</td>
<td>[83.96]</td>
<td>—</td>
</tr>
<tr>
<td>View-Invariant K-means [15]</td>
<td>63.7</td>
<td>72.6 ± 0.7</td>
<td>81.9</td>
<td>—</td>
</tr>
<tr>
<td>This paper</td>
<td>67.4 ± 0.6</td>
<td>69.3 ± 0.4</td>
<td>77.5</td>
<td>76.6 ± 0.7</td>
</tr>
</tbody>
</table>
Unsupervised Learning from Video


Abstract

Is strong supervision necessary for learning a good visual representation? Do we really need millions of semantically-labeled images to train a ConvNet? In this paper, we present a simple yet surprisingly powerful approach for unsupervised learning of ConvNets. Specifically, we use hundreds of thousands of unlabeled videos from the web to learn visual representations. Our key idea is that we track millions of patches in these videos. Visual tracking provides the key supervision. That is, two patches connected by a track should have similar visual representation in deep feature space since they probably belong to same object or object part. We design a Siamese-triplet network with a ranking loss function to train this ConvNet representation. Without using a single image from ImageNet, just using 100K unlabeled videos and the VOC 2012 dataset, we train an ensemble of unsupervised networks that achieves 52% mAP (no bounding box regression). This performance comes tantalizingly close to its ImageNet-supervised counterpart, an ensemble which achieves a mAP of 54.4%.

1. Introduction

What is a good visual representation and how can we learn it? At the start of this decade, most computer vision research focused on "what" and used hand-defined features such as SIFT [29] and HOG [5] as the underlying visual representation. Learning was often the last step where these low-level feature representations were mapped to semantic/3D/functional categories. However, the last three years have seen the resurgence of learning visual representations directly from pixels themselves using the deep learning and ConvNets [25, 21, 20]. At the heart of ConvNets is a completely supervised learning paradigm. Often millions of examples are first labeled using Mechanical Turk followed by data augmentation to create tens of millions of training instances. ConvNets are then trained using gradient descent…

![Diagram](image_url)

(a) Unsupervised Tracking in Videos

(b) Siamese-triplet Network

(c) Ranking Objective

Learning to Rank

Conv Net

Conv Net

Conv Net

Query (First Frame)

Tracked (Last Frame)

Negative (Random)

Distance in deep feature space

$D$: Distance in deep feature space

$D$, $D$, $D$, $D$, $D$

$\langle D \rangle$

$\langle D \rangle$

$\langle D \rangle$

$\langle D \rangle$
Generative Adversarial Networks

Generative Adversarial Networks


- Minimax value function:

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]
\]

Discriminator pushes up
Generator pushes down
Discriminator’s ability to recognize data as being real
Discriminator’s ability to recognize generator samples as being fake

[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]
Generative Adversarial Networks


\[ D(x) \]

Data distribution

Model distribution

Poorly fit model

After updating D

After updating G

Mixed strategy equilibrium

[Slide: Ian Goodfellow, Deep Learning workshop, ICML 2015]
Adversarial Network Samples

MNIST

CIFAR-10 (fully connected)

TFD

CIFAR-10 (convolutional)
Adversarial Network using Laplacian Pyramid

Adversarial Network using Laplacian Pyramid

Memory in Neural Networks

Sainbayar Sukhbaatar
Introduction

• Recently, there has been lot of interest in incorporating memory and attention to neural networks
  – Memory Networks, NTM, Learning to attend …
• Neural networks are not good at remembering things, especially when input is large but only part of it is relevant
• Adding external memory and learning to attend on important part is key
Outline

• Implicit Internal memory
  – RNN, LSTM

• Explicit External memory
  – MemNN, NTM

• Attention models
  – MT, Speech, Image, Pointer Network
Implicit Internal Memory

- Internal state of the model can be used for memory
  - Recurrent Neural Networks (RNNs)

- Computation and memory is mixed
  - Complex computation requires many layers of non-linearity
  - But some information is lost with each non-linearity
  - Gradient vanishing, Catastrophic forgetting
Ways to Prevent Forgetting in RNNs

• Split state into fast and slow changing parts: structurally constrained recurrent nets (Mikolov et al., 2014)
  – Fast changing part is good for computation
  – Slow changing part is good for storing information
• Gated units for internal state
  – Control when to forget/write using gates
  – Long-short term memory (LSTM) (see Graves, 2013)
  – Simpler Gated Recurrent Unit (GRU) (Cho et al., 2014)
• Other problems
  – Memory capacity is fixed and limited by the dimension of state vector (computation is $O(N^2)$ where $N$ is memory capacity)
  – Vulnerable to distractions in inputs
  – Restricted to sequential inputs
Stack memory for RNN
(Joulin et al., 2014)

- Added a stack module to RNN, which can hold a list of vectors
- Action on stack: push, pop and no-op
- More powerful with multiple stacks
- Stack are updated in continuous manner → differentiable → trainable by backpropagation + search
- Applied to counting, memorization, binary addition
External Global Memory

- Separate memory from computation
  - Add separate memory module for storage
  - Memory contains list/set of items

- Main module can read and write to the memory
- Advantage: long-term, scalable, flexible
Selective Addressing is Key for Memory

- Often, you only want to interact with few items in memory at once
  - Memory needs some addressing mechanism

- Memory addressing types
  - Soft or hard addressing
    - Soft addressing can be trained by backpropagation
    - Hard addressing is not differentiable (e.g. can be trained with reinforcement learning or additional training signal for where to attend)
  - Context and Location based addressing
    - When input is ordered in some way, location based addressing is useful
    - Location addressing is same as context if location is embedded in the context (e.g. MemN2N)
Memory Networks
(Weston et al., 2014)

• Neural network with large external memory
• Writes everything to the memory, but reads only relative information
• Hard addressing: max of the inner product between then internal state and memory contents
• Location based addressing: can compare two memory items by their relative location
• Can perform multiple memory lookups (hops) before producing an output
• Requires additional training signals for training hard addressing
• Applied to toy and large-scale QA tasks
John is in office

Mary is in garden

John is in office

Bob is in kitchen

Where is John

Input text

Internal state vector

Decoder

Output

Memory

MAX

Embed
End-to-end Memory Networks (Sukhbaatar et al., 2015)

• Soft addressing: replaced hard max with \texttt{softmax}

• End-to-end training: \texttt{softmax} is differentiable \rightarrow can train with backpropagation

• Location addressing: location/time is embedded into the context (special words for “Time=4”)

• Applied to toy QA and language modeling
End-to-end Memory Networks (Sukhbaatar et al., 2015)

We describe our model in the single layer case, which implements a single memory hop operation. We then show it can be stacked to give multiple hops in memory.

Input memory representation:
Suppose we are given an input set \( x_1, \ldots, x_i \) to be stored in memory. The memory vector \( m_i \) of dimension \( d \) is computed by first embedding each \( x_i \) in a continuous space, in the simplest case, using an embedding matrix \( A \) (of size \( d \times V \)). Thus, the entire set of \( \{x_i\} \) are converted into memory vectors \( \{m_i\} \). The query \( q \) is also embedded (again, in the simplest case via another embedding matrix \( B \) with the same dimensions as \( A \)) to obtain an internal state \( u \). In the embedding space, we compute the match between \( u \) and each memory \( m_i \) by taking the inner product followed by a softmax:

\[
p_i = \text{Softmax}(u^T m_i)
\]

where \( \text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j} \). Defined in this way \( p_i \) is a probability vector over the inputs.

Output memory representation:
Each \( x_i \) has a corresponding output vector \( c_i \) (given in the simplest case by another embedding matrix \( C \)). The response vector from the memory \( o \) is then a sum over the \( c_i \), weighted by the probability vector from the input:

\[
o = \sum_i p_i c_i
\]

Because the function from input to output is smooth, we can easily compute gradients and back-propagate through it. Other recently proposed forms of memory or attention take this approach, notably Bahdanau et al. [2] and Graves et al. [8], see also [9].

Generating the final prediction:
In the single layer case, the sum of the output vector \( o \) and the input embedding \( u \) is then passed through a final weight matrix \( W \) (of size \( V \times d \)) and a softmax to produce the predicted label:

\[
\hat{a} = \text{Softmax}(W(o + u))
\]

The overall model is shown in Fig. 1(a). During training, all three embedding matrices \( A, B, \) and \( C, W \) are jointly learned by minimizing a standard cross-entropy loss between \( \hat{a} \) and the true label \( a \). Training is performed using stochastic gradient descent (see Section 4.2 for more details).

(a)

Single Memory Lookup
End-to-end Memory Networks
(Sukhbaatar et al., 2015)

2.1 Single Layer

We start by describing our model in the single layer case, which implements a single memory hop operation. We then show it can be stacked to give multiple hops in memory.

Input memory representation:

Suppose we are given an input set \( \{x_i\}\) to be stored in memory. The memory vector \( m_i \) of dimension \( d \) is computed by first embedding each \( x_i \) in a continuous space, in the simplest case, using an embedding matrix \( A \) (of size \( d \times V \)). Thus, the entire set of \( \{x_i\} \) are converted into memory vectors \( \{m_i\} \). The query \( q \) is also embedded (again, in the simplest case via another embedding matrix \( B \) with the same dimensions as \( A \)) to obtain an internal state \( u \). In the embedding space, we compute the match between \( u \) and each memory \( m_i \) by taking the inner product followed by a softmax:

\[
p_i = \text{Softmax}(u^T m_i)
\]

(1)

where \( \text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j} \). Defined in this way \( p \) is a probability vector over the inputs.

Output memory representation:

Each \( x_i \) has a corresponding output vector \( c_i \) (given in the simplest case by another embedding matrix \( C \)). The response vector from the memory \( o \) is then a sum over the \( c_i \), weighted by the probability vector from the input:

\[
o = \sum_i p_i c_i
\]

(2)

Because the function from input to output is smooth, we can easily compute gradients and back-propagate through it. Other recently proposed forms of memory or attention take this approach, notably Bahdanau et al. [2] and Graves et al. [8], see also [9].

Generating the final prediction:

In the single layer case, the sum of the output vector \( o \) and the input embedding \( u \) is then passed through a final weight matrix \( W \) (of size \( V \times d \)) and a softmax to produce the predicted label:

\[
\hat{a} = \text{Softmax}(W(o + u))
\]

(3)

The overall model is shown in Fig. 1(a). During training, all three embedding matrices \( A, B \) and \( C \), as well as \( W \) are jointly learned by minimizing a standard cross-entropy loss between \( \hat{a} \) and the true label \( a \). Training is performed using stochastic gradient descent (see Section 4.2 for more details).

Figure 1: (a): A single layer version of our model. (b): A three layer version of our model. In practice, we can constrain several of the embedding matrices to be the same (see Section 2.2).

2.2 Multiple Layers

We now extend our model to handle \( K \) hop operations. The memory layers are stacked in the following way:

- Single Memory Lookup
- Multiple Memory Lookup

(Sukhbaatar et al., 2015)
RNN viewpoint of End-to-End MemNN

Plain RNN

Inputs are fed to RNN one-by-one in order. RNN has only one chance to look at a certain input symbol.

Memory Network

Place all inputs in the memory. Let the model decide which part it reads next.
Attention during memory lookups

Samples from toy QA tasks (bAbI dataset)

<table>
<thead>
<tr>
<th>Story (1: 1 supporting fact)</th>
<th>Support</th>
<th>Hop 1</th>
<th>Hop 2</th>
<th>Hop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daniel went to the bathroom.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.03</td>
<td></td>
</tr>
<tr>
<td>Mary travelled to the hallway.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>John went to the bedroom.</td>
<td>0.37</td>
<td>0.02</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>John travelled to the bathroom.</td>
<td>yes</td>
<td>0.60</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>Mary went to the office.</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Where is John? Answer: bathroom Prediction: bathroom

<table>
<thead>
<tr>
<th>Story (2: 2 supporting facts)</th>
<th>Support</th>
<th>Hop 1</th>
<th>Hop 2</th>
<th>Hop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>John dropped the milk.</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>John took the milk there.</td>
<td>0.88</td>
<td>1.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Sandra went back to the bathroom.</td>
<td>yes</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>John moved to the hallway.</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>Mary went back to the bedroom.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Where is the milk? Answer: hallway Prediction: hallway

<table>
<thead>
<tr>
<th>Story (16: basic induction)</th>
<th>Support</th>
<th>Hop 1</th>
<th>Hop 2</th>
<th>Hop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian is a frog.</td>
<td>yes</td>
<td>0.00</td>
<td>0.98</td>
<td>0.00</td>
</tr>
<tr>
<td>Lily is gray.</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Brian is yellow.</td>
<td>yes</td>
<td>0.07</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Julius is green.</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Greg is a frog.</td>
<td>yes</td>
<td>0.76</td>
<td>0.02</td>
<td>0.00</td>
</tr>
</tbody>
</table>

What color is Greg? Answer: yellow Prediction: yellow

<table>
<thead>
<tr>
<th>Story (18: size reasoning)</th>
<th>Support</th>
<th>Hop 1</th>
<th>Hop 2</th>
<th>Hop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The suitcase is bigger than the chest.</td>
<td>yes</td>
<td>0.00</td>
<td>0.88</td>
<td>0.00</td>
</tr>
<tr>
<td>The box is bigger than the chocolate.</td>
<td>0.04</td>
<td>0.05</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>The chest is bigger than the chocolate.</td>
<td>yes</td>
<td>0.17</td>
<td>0.07</td>
<td>0.90</td>
</tr>
<tr>
<td>The chest fits inside the container.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>The chest fits inside the box.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Does the suitcase fit in the chocolate? Answer: no Prediction: no

<table>
<thead>
<tr>
<th>Result</th>
<th>Test error</th>
<th>Failed tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemNN</td>
<td>6.7%</td>
<td>4</td>
</tr>
<tr>
<td>LSTM</td>
<td>51%</td>
<td>20</td>
</tr>
<tr>
<td>MemN2N</td>
<td>12.4%</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 2: The perplexity on the test sets of Penn Treebank and Text8 corpora. Note that increasing the number of memory hops improves performance.

Language Modeling Experiments

The goal in language modeling is to predict the next word in a text sequence given the previous words $x$. We now explain how our model can easily be applied to this task.

We now operate on word level, as opposed to the sentence level. Thus the previous $N$ words in the sequence (including the current) are embedded into memory separately. Each memory cell holds only a single word, so there is no need for the BoW or linear mapping representations used in the QA tasks. We employ the temporal embedding approach of Section 4.1.

Since there is no longer any question, $q$ in Fig. 1 is fixed to a constant vector $0.1$ (without embedding). The output softmax predicts which word in the vocabulary (of size $V$) is next in the sequence. A cross-entropy loss is used to train model by backpropagating the error through multiple

Figure 3: Average activation weight of memory positions during 6 memory hops. White color indicates where the model is attending during the $k$th hop. For clarity, each row is normalized to have maximum value of 1. A model is trained on (left) Penn Treebank and (right) Text8 dataset.

Penn Treebank
Text8
# of # of memory Valid. Test
# of # of memory Valid. Test

<table>
<thead>
<tr>
<th>Model</th>
<th>hidden hops size perp.</th>
<th>perp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>-</td>
</tr>
<tr>
<td>LSTM</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>-</td>
</tr>
<tr>
<td>SCRN</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>-</td>
</tr>
<tr>
<td>MemN2N</td>
<td>150</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3: The perplexity on the test sets of Penn Treebank and Text8 corpora. Note that increasing the number of memory hops improves performance.
Neural Turing Machine
(Graves et al., 2014)

• Learns how to write to the memory
• Soft addressing $\rightarrow$ backpropagation training
• Location addressing: small continuous shift of attention
• Complex addressing mechanism: need to sharpen after convolution
• Controller can be LSTM-RNN or feed-forward neural network
• Applied to learn algorithms such as sort, associative recall and copy.
• Hard addressing with reinforcement learning (Zaremba et al., 2015)
RNNsearch: Attention in Machine Translation (Bahdanau et al., 2015)

• RNN based encoder and decoder model
• Decoder can look at past encoder states using soft attention
• Attention mechanism is implemented by a small neural network
  – It takes the current decoder state and a past encoder state and outputs a score. Then the all scores are fed to softmax to get attention weights
• Applied to machine translation. Significant improvement in translation of longer sentences

![Attention weights during English to French machine translation](image)

![Significant improvement on long sentences](image)
Image caption generation with attention (Xu et al., 2015)

- Encoder: lower convolutional layer of a deep ConvNet (because need spatial information)
- Decoder: LSTM RNN with soft spatial attention
  - Decoder state and encoder state at single location are fed to small NN to get score at that location
- Network attends to the object when it is generating a word for it
- Also hard attention is tried with reinforcement learning

A woman is throwing a **frisbee** in a park.

A dog is standing on a **hardwood floor**.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a **teddy bear**.

A group of **people** sitting on a boat in the water.

A giraffe standing in a forest with **trees** in the background.
Video description generation
(Yao et al., 2015)

+Local+Global: A man and a woman are talking on the road
Ref: A man and a woman ride a motorcycle

+Local+Global: Someone is frying a fish in a pot
Ref: A woman is frying food

(bottom: ground truth)

Location-aware attention for speech
(Chorowski et al., 2015)

- RNN based encoder-decoder model with attention (similar to RNNsearch)
- Location based addressing: previous attention weights are used as feature for the current attention (good when subsequent attention locations are highly correlated)
- Improvement with sharpening and smoothing of memory addressing
Pointer Network: attention as an output (Vinyals et al., 2015)

• RNN based encoder-decoder model for discrete optimization problems
• Decoder can attend to previous encoder states (similar to RNNsearch, content based soft attention by a small NN)
• Rather than fixed output classes, attention weights determine output
• Input to the most attended encoder state becomes an output \( \Rightarrow \) can output any sequence of inputs

Ground Truth: tour length is 3.518

Predictions: tour length is 3.523
Resources

• EMNLP 2014 tutorial
  – http://emnlp2014.org/tutorials.html#embedding

• CVPR2014 deep learning tutorial
  – https://sites.google.com/site/deeplearningcvpr2014/

• ICML 2013 deep learning tutorial
Software

• Caffe (http://caffe.berkeleyvision.org/)
  – Vision-centric

• Torch (http://torch.ch/)
  – Lua-based library for Deep Learning
  – Currently used by FAIR and Google Deep Mind

• Theano (http://deeplearning.net/software/theano/)
  – Automatic differentiation
  – Python-based
Thanks!

Facebook AI Research colleagues & NYU PhD students:

Manohar Paluri
Antoine Bordes
Yaniv Taigman
Soumith Chintala
Emily Denton

Jason Weston
Tomas Mikolov
Ronan Collobert
Marc’Aurelio Ranzato
Sainbayar Sukhbaatar
FAIR Overview
Facebook AI Research

- Toward Artificial Intelligence (AI), with Machine Learning.
- Established Dec 2013 (1.5 year old)
  - initiative of CEO and CTO
  - lead by Yann Lecun
FAIR Overview
Facebook AI Research

- ~35 researcher scientists
  - Machine Learning, Computer Vision and Natural Language Processing

- ~15 research engineers
  - Software support, prototyping, interaction with product teams...

- Locations:
  - New York City
  - Menlo Park (HQ)
  - Paris
FAIR Mission

Facebook AI Research

- Advance the state of the art of AI
  - Publish research in best conferences and journals
  - Open-source code release
- Produce software tools for AI research and applications
- Help FB products to leverage advances in AI
  - Software prototyping, architecting, interaction with product teams...
FAIR Impact
Machine Learning @ FB

- Computer Vision
  - Face detection and identification
  - Object detection, scene classification
  - Video classification

- Natural Language
  - Tag prediction for search, feed ranking, ad targeting

- Computational Advertising
  - Ads targeting
  - User interest modeling
Huge Scale Deployment of Machine Learning

- 1.4 billion monthly active users
- 850 million daily active users (1 in 7 people on Earth)
- More images uploaded than any other website
  - 400M+ new Facebook photos/day (no labels)
  - 60M+ Instagram images/day (most with hashtags)
  - ~500 Billion photos total
- Face and Object recognition models applied to every image
- 5M video uploads/day & growing rapidly
  - More video playback than YouTube
We are hiring!

• Internships
  - https://www.facebook.com/careers/department?dept=grad&req=a0IA000000CzCGuMAN

• Postdoc positions
  - Ex-postdocs now faculty at Berkeley, Harvard

• Full-time positions

• https://research.facebook.com/ai
Memory References

Memory References

• K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. ICML, 2015
Neuroscience of memory

• hippocampus
  – Densely connected
  – Vital for new memory formation
  – From few days to few years
  – Place / grid cells

• Neo-cortex
  – Can keep memory much longer
Memory types

• Short-term memory (working memory)
  – Limited capacity

• Long term memory
  – Explicit / Declarative
    • Semantic memory
    • Episodic memory
  – Implicit
    • Procedural memory
    • Priming