Deep Learning via Semi-Supervised Embedding

Jason Weston
NEC Labs America, Princeton, USA

Joint work with Ronan Collobert, Frederic Ratle, Hossein Mobahi, Pavel Kuksa and Koray Kavukcuoglu.

Summary

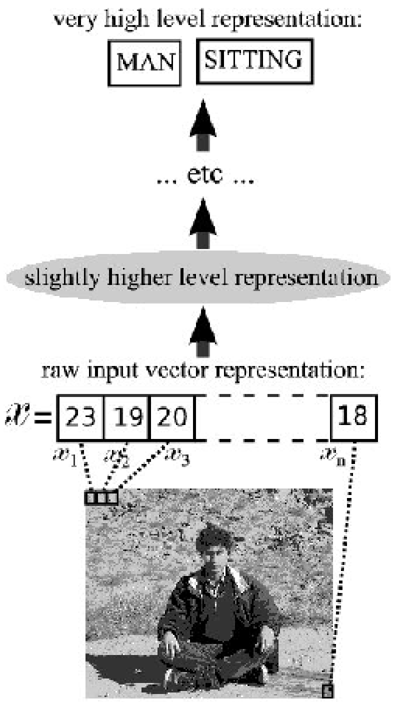
- We pose deep learning as multi-tasking at different layers with auxiliary tasks.
- Hinton, LeCun and Bengio approaches use encoder-decoder models as the auxiliary task.
- We propose simple "encoder only" methods: easy, simple, fast, works well.
- Experiments: can train very deep networks (15 layers) with better results than shallow networks ($\leq$4 layers) (including SVMs = 1 layer!)

Apply this to:

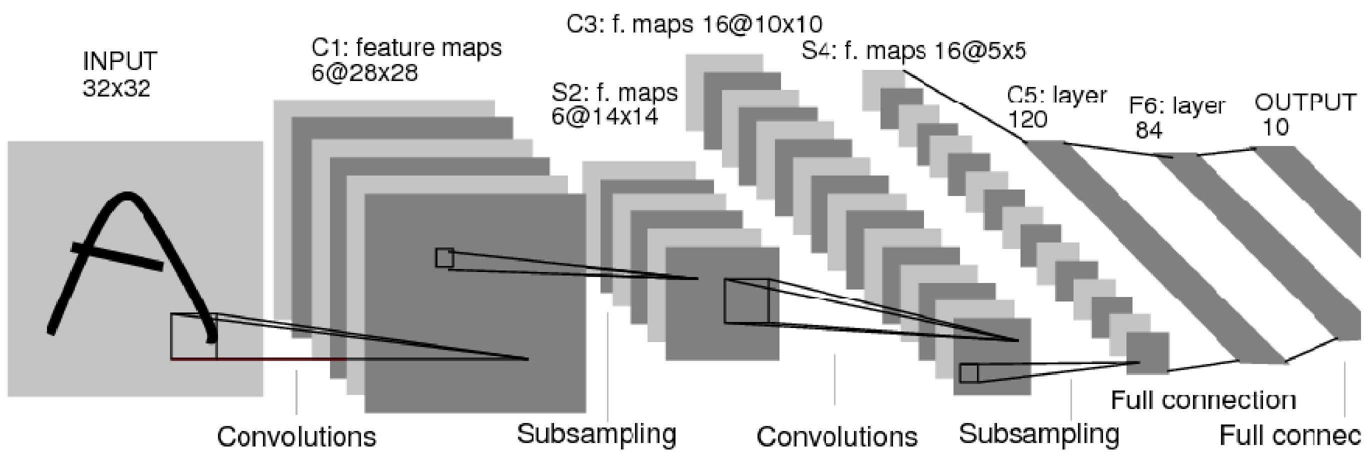
- Video: unlabeled video helps object recognition.
- Text: unlabeled text (600 million examples) helps tagging tasks.

Deep Learning with Neural Networks [Images: Y. Bengio, Y. LeCun]

NN:



CNN:



Deep = lot of layers. Powerful systems.

Standard backpropagation doesn’t always give great results.

Some Deep Training Methods That Exist

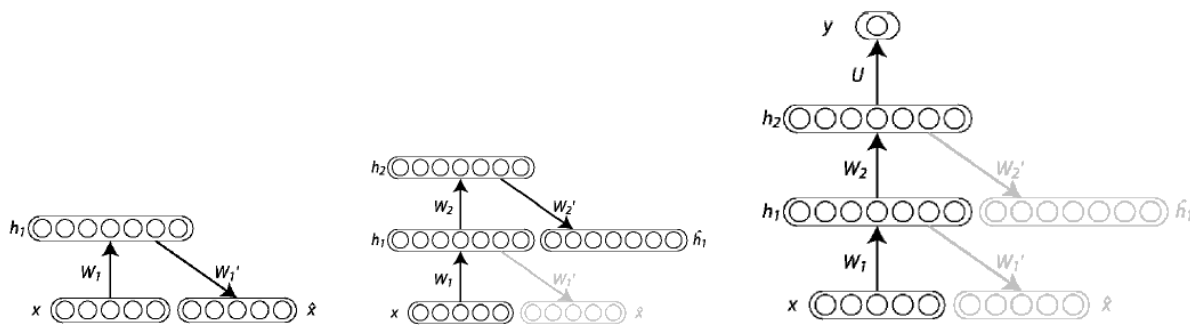
Hinton's group: DBNs – special kind of an encoder+decoder.

Y. Bengio's group propose using "classical" autoencoders *or* denoising encoder+decoders.

LeCun's group: sparse encoder-decoders.

- Pre-train with unlabeled data: *"afterwards parameters in a region of space where good optimum can be reached by local descent."*

- Pre-training: greedy layer-wise [Image: Larochelle et al. 2007]



(a) Reconst. $\mathbf{x}$ (b) Reconst. $\mathbf{h}^1$ (c) Predict y

- "Fine-tune" network afterwards using backprop.

Deep and Shallow Research

Deep Researchers (DRs) believe:

- Learn sub-tasks in layers. Essential for *hard* tasks.
- Natural for multi-task learning.
- Non-linearity is efficient compared to n^3 shallow methods.

Shallow Researchers believe:

- NNs were already complicated and messy.
- New deep methods are *even more complicated and messy*.
- Shallow methods: clean and give valuable insights into what works.

My p.o.v. $\rightarrow$ borrow from shallow research, place into deep algorithms

Deep NNs: Multitask with auxiliary unsupervised tasks

- Define "pseudo-supervised" tasks for unlabeled data [Ando & Zhang, 2005]   EXAMPLE: predict middle word given a window
- Multi-task labeled + unlabeled tasks, acts as regularizer

Convex learning:

- must train labeled + unlabeled at same time.

Non-convex:

- train sequentially, might still help $\rightarrow$ explains autoencoders.
- multi-layer nets can be multitasked at each layer.

*We will consider multi-tasking with a pairwise embedding algorithm...*

Existing Embedding Algorithms

Many existing ("shallow") embedding algorithms optimize:

$$\min \sum_{i,j=1}^{U} L(f(x_i), f(x_j), W_{ij}), \quad f_i \in \mathbb{R}^d$$

MDS: minimize $(||f_i - f_j|| - W_{ij})^2$

ISOMAP: same, but W defined by shortest path on neighborhood graph.

Laplacian Eigenmaps: minimize

$$\sum_{ij} W_{ij} ||f_i - f_j||^2$$

subject to "balancing constraint": $f^\top D f = I$ and $f^\top D 1 = 0$.

Siamese Networks: functional embedding

Similar to Lap. Eigenmaps but $f(x)$ is a NN.

DrLIM [Hadsell et al.,'06]:

$$L(f_i, f_j, W_{ij}) = \begin{cases} ||f_i - f_j||^2 & \text{if } W_{ij} = 1, \\ \max(0, m - ||f_i - f_j||)^2 & \text{if } W_{ij} = 0. \end{cases}$$

$\rightarrow$ *neighbors close, others have distance of at least* m

- Avoid trivial solution using $W_{ij} = 0$ case $\rightarrow$ easy online optimization

- $f(x)$ not just a lookup-table $\rightarrow$ control capacity, add prior knowledge, no out-of-sample problem

Shallow Semi-supervision

SVM: $\min_{w,b} \gamma ||w||^2 + \sum_{i=1}^{L} H(y_i f(x_i))$

Add embedding regularizer: unlabeled neighbors have same output:

- LapSVM [Belkin et al.]:

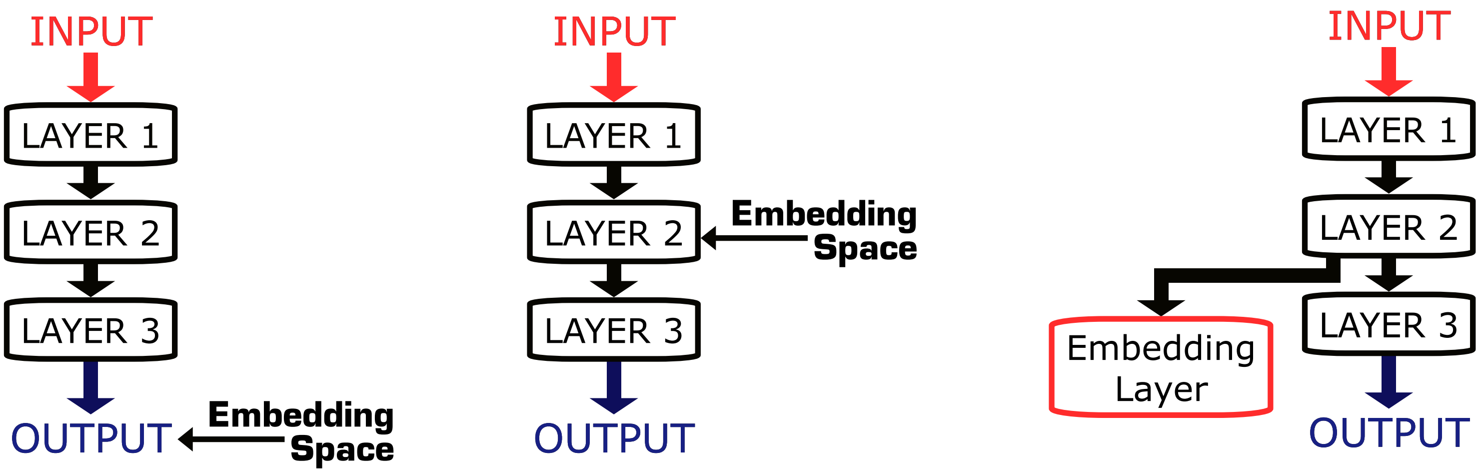
$$\text{SVM} + \lambda \sum_{i,j=1}^{U} W_{ij} ||f(x_i^*) - f(x_j^*)||^2$$

e.g. $W_{ij} = 1$ if two points are neighbors, 0 otherwise.

- "Preprocessing":

  Using ISOMAP vectors as input to SVM [Chapelle et al.]…

New regularizer for NNs: Deep Embedding



- Define Neural Network: $f(x) = h^3(h^2(h^1(x))$
- Supervised Training: minimize $\sum_i \ell(f(x_i), y_i)$
- Add Embedding Regularizer(s) to training:

Output: $\sum_i L(f(x_i), f(x_j), W_{ij})$ or

Internal: $\sum_i L(h^2(h^1((x_i)), h^2(h^1(x_j)), W_{ij})$

Aux.: $\sum_i L(e(x_i), e(x_j), W_{ij})$, where $e(x) = e^3(h^2(h^1(x)))$

Deep Semi-Supervised Embedding

Input: labeled data (x_i, y_i) and unlabeled data x_i^*, and matrix W

repeat

  Pick random labeled example (x_i, y_i)

  Gradient step for $H(y_i f(x_i))$

  for each embedding layer **do**

    Pick a random pair of neighbors x_i^*, x_j^*.

    Gradient step for $L(x_i^*, x_j^*, 1)$

    Pick a random pair x_i^*, x_k^*.

    Gradient step for $L(x_i^*, x_k^*, 0)$

  end for

until stopping criteria

Pairwise Example Prior: more general than using k-NN

- Standard way: k-nn with Euclidean distance.
  - many methods to make it fast.
  - …but Euclid. might suck.
- Sequences: text, images (video), speech (audio)
  - video: patch in frames t & $t+1 \longrightarrow$ same label
  - audio: consecutive audio frames $\longrightarrow$ same speaker + word ..
  - text: word + neighbors $\longrightarrow$ same topic
- Web data:
  - use links/click-through information to collect neighbors
  - images and text on same page

Some Perspectives

- General [Ando & Zhang '05] framework: sometimes difficult to define the task?
- Embedding is a class of auxiliary task, still free to define pairs.
- Encoder+Decoders= another class: learn regions of space that are densely populated (support of density?).
  Pairwise Embedding does something similar (encoder without decoder?).
- Pairwise Embedding has no decoder: for sparse inputs (e.g. bag of words) this is much faster than dense decoding.
- Another way: [Yu et al. '08] proposed NN auxiliary task approximating a *known* useful distance metric given by a hand-engineered kernel.

Our method should help when the "auxiliary" embedding matrix W is correlated to the supervised task.

Some Experiments: Small Semi-Supervised Setup

Typical *shallow semi-supervised* datasets:

data set	classes	dims	points	labeled
g50c	2	50	500	50
Text	2	7511	1946	50
Uspst	10	256	2007	50
Mnist1h	10	784	70k	100
Mnist6h	10	784	70k	600
Mnist1k	10	784	70k	1000

- First experiment: Only consider two-layer nets.

Deep Semi-Supervised Results

	g50c	Text	Uspst
SVM	8.32	18.86	23.18
SVMLight-TSVM	6.87	7.44	26.46
∇TSVM	5.80	5.71	17.61
LapSVM*	5.4	10.4	12.7
NN	8.54	15.87	24.57
*Embed*NN^{O}	5.66	5.82	15.49

	Mnist1h	Mnist6h	Mnist1k
SVM	23.44	8.85	7.77
TSVM	16.81	6.16	5.38
RBM$^{(*)}$	21.5	-	8.8
SESM$^{(*)}$	20.6	-	9.6
DBN-rNCA$^{(*)}$	-	8.7	-
NN	25.81	11.44	10.70
$Embed^{O}$NN	17.05	5.97	5.73
$Embed^{I1}$NN	16.86	9.44	8.52
$Embed^{A1}$NN	17.17	7.56	7.89
CNN	22.98	7.68	6.45
$Embed^{O}$CNN	11.73	3.42	3.34
$Embed^{I5}$CNN	7.75	3.82	2.73
$Embed^{A5}$CNN	7.87	3.82	2.76

Really Deep Results

Same MNIST1h dataset, but training 2-15 layer nets (50HUs each):

layers=	2	4	6	8	10	15
NN	26.0	26.1	27.2	28.3	34.2	47.7
$Embed\mathrm{NN}^{O}$	19.7	15.1	15.1	15.0	13.7	11.8
$Embed\mathrm{NN}^{ALL}$	18.2	12.6	7.9	8.5	6.3	9.3

- $Embed\mathrm{NN}^{O}$: auxiliary 10-dim embedding on output layer
- $Embed\mathrm{NN}^{ALL}$: auxiliary 10-dim embedding on every layer.
- Trained jointly with supervised signal, as before.
- (NOTE: Train error of NN can easily achieve 0.)
- SVM: 23.4% ,   TSVM: 16.8%

Conclusions (so far)

*Embed*NN generalizes shallow semi-supervised embedding.

- Easy to train.
- No pre-training, no decoding step = simple, fast.
- Seems to train very deep networks.

NOW. . . we will apply this to:

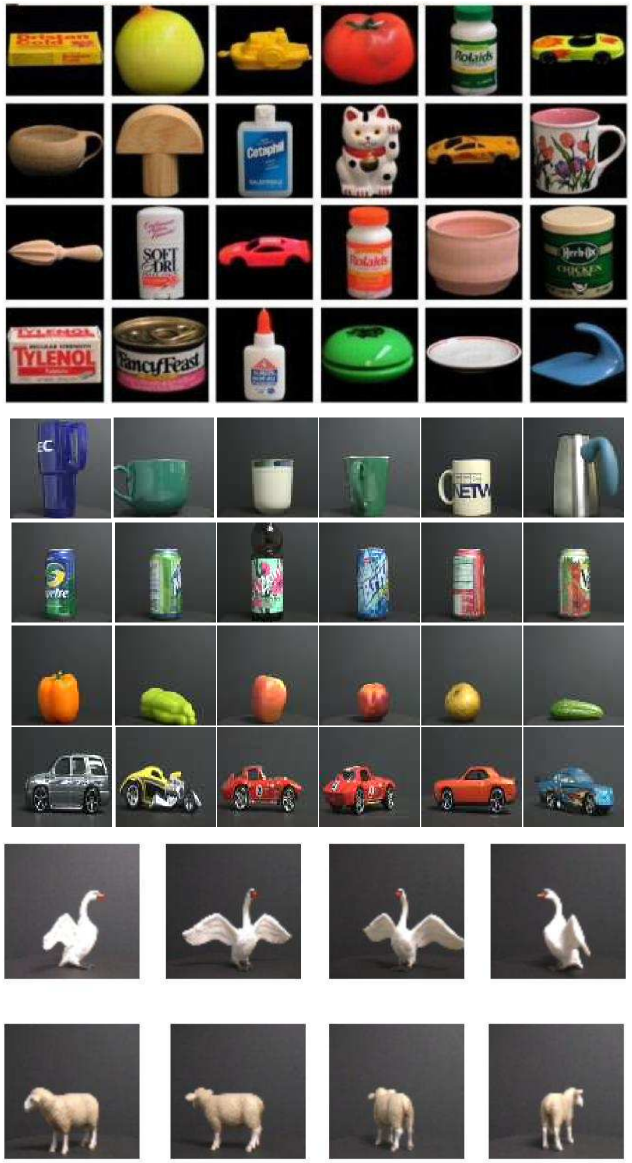
- Video: unlabeled video helps object recognition.
- Text: unlabeled text (600 million examples) helps tagging tasks.

Deep Learning For Video

APPLICATION: LEARNING FROM VIDEO

- Two consecutive frames likely to contain the same object or objects.
- Improve deep layers (internal representation of images):

  learn invariance to pose, illumination, background or clutter, deformations (e.g. facial expressions) or occlusions.
- Video collections obtained without human annotation.
- We show this works for varying video sources.
- Biologically, supervised learning isn't so plausible, but this might be..



- COIL-100 database.
  - 100 objects, 72x72 pixels.
  - 72 different poses.
- COIL-Like database.
  - 40 objects, 72 views.
  - 4 types (fruits, cars, cups, cans).
  - videostream
  - collected to look like COIL.
- Animal database.
  - 60 animals (horses, rabbits,. . .)
  - videostream
  - no objects in common with COIL.

Experimental setup

- Supervised task from COIL: 4 views for train, 68 for test. 30 or 100 objects for train/test following [Wersing, 2003].
- COIL video: transductive (100 objects) and semi-supervised (70 object) settings + COIL-Like and Animal videos.
- *Methods:*
  - Baseline methods: SVM, Nearest neighbors,. . . .
  - Baseline CNN
  - strongly engineered Neural Net (VTU) [Wersing et. al., 2003][a]
  - Our *video*CNN with different video sources.

[a]The VTU method builds a hierarchy of biologically inspired feature detectors. It applies Gabor filters at four orientations, followed by spatial pooling, and learns receptive field profiles using a special type of sparse coding algorithm with invariance constraints.

Test Accuracy Performance on COIL100 in various settings.

Method	30 objects	100 objects
Nearest Neighbor	81.8	70.1
SVM	84.9	74.6
SpinGlass MRF	82.8	69.4
Eigen Spline	84.6	77.0
VTU	89.9	79.1
Standard CNN	84.88	71.49
*video*CNN V:COIL100	-	92.25
*video*CNN V:COIL“70”	95.03	-
*video*CNN V:COIL-Like	-	79.77
*video*CNN V:Animal	-	78.67

Outperforms baselines without using engineered features.

Deep Learning For Text

NLP Tasks

- Part-Of-Speech Tagging (POS): syntactic roles (noun, adverb...)
- Chunking: syntactic constituents (noun phrase, verb phrase...)
- Name Entity Recognition (NER): person/company/location...
- Semantic Role Labeling (SRL): semantic role

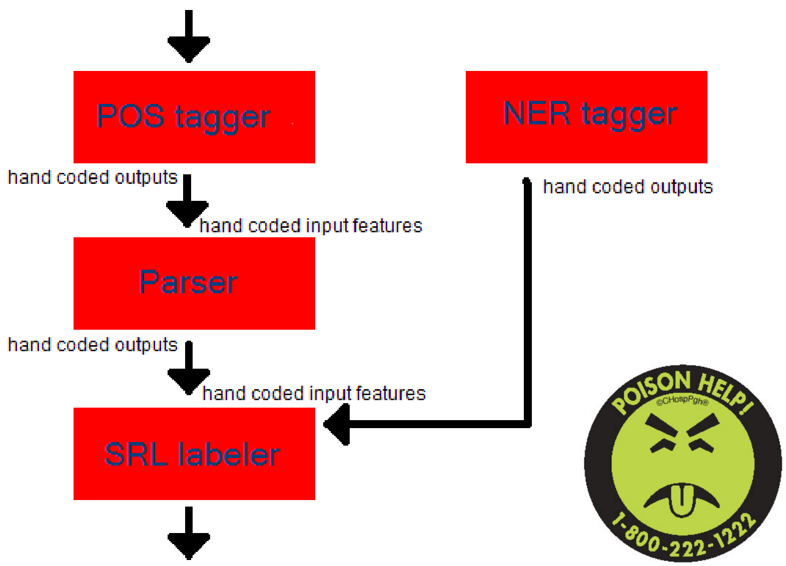
  [John]$_{ARG0}$ [ate]$_{REL}$ [the apple]$_{ARG1}$ [in the garden]$_{ARGM-LOC}$

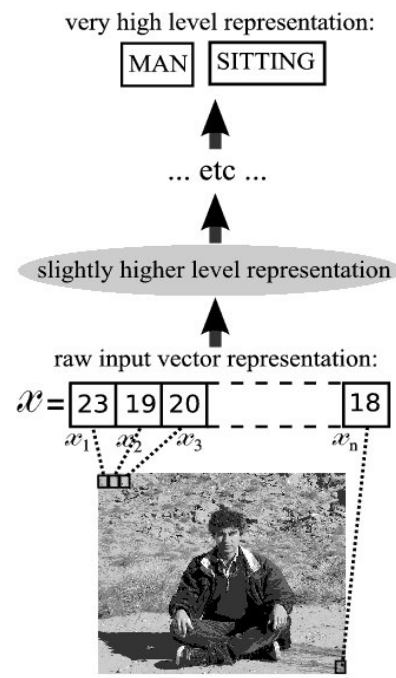
Labeled data: Wall Street Journal ($\sim 1M$ words)

The "Brain Way"

Deep learning seems radically different to the traditional NLP approach:

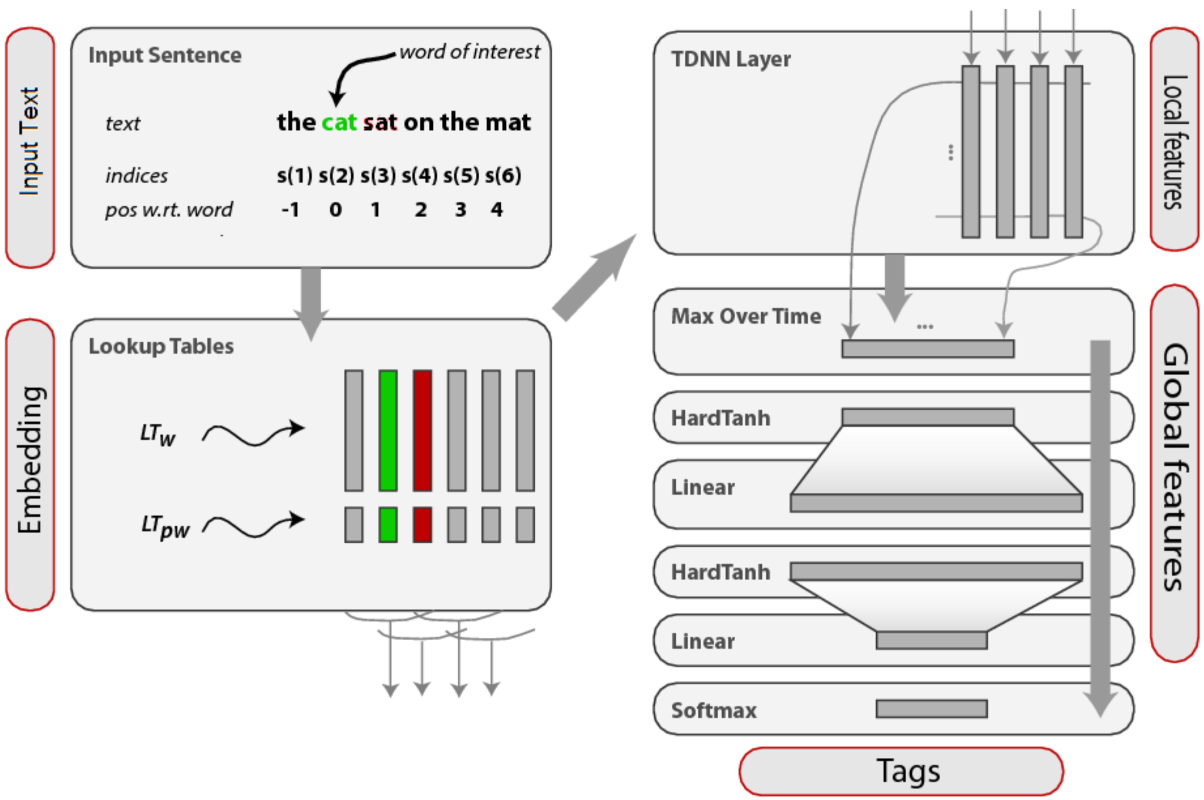
- Avoid building a parse tree. Humans don't need this to talk.
- We try to avoid all hand-built features → monolithic systems.
- Humans implicitly learn these features. Neural networks can too. . . ?





→ End-to-end system + Fast predictions (0.02 sec/sentence)

The Deep Learning Way



INPUT: lower case words

LEARN: word feature vectors using *auxiliary embedding.*

Using Unlabeled Data

- Language Model: "*is (part of) a sentence actually english or not?*"

  Implicitly captures

  - syntax
  - semantics

  Trained over Wikipedia ($\sim 631M$ words)

- Bengio & Ducharme (2001)

  Probability of next word given previous words

- Pick word + neighborhood $\longrightarrow W_{ij} = 1$ (push together) **+ve pair**

  "The cat sat on the " $\longrightarrow \longleftarrow$ "mat"

- Same neighborhood + random word $\longrightarrow W_{ij} = 0$ (push apart)

  "The cat sat on the" $\longleftarrow \longrightarrow$ "DBN" **-ve pair**

Language Model: Embedding

FRANCE 454	JESUS 1973	XBOX 6909	REDDISH 11724	SCRATCHED 29869
SPAIN	CHRIST	PLAYSTATION	YELLOWISH	SMASHED
ITALY	GOD	DREAMCAST	GREENISH	RIPPED
RUSSIA	RESURRECTION	PS2	BROWNISH	BRUSHED
POLAND	PRAYER	SNES	BLUISH	HURLED
ENGLAND	YAHWEH	WII	CREAMY	GRABBED
DENMARK	JOSEPHUS	NES	WHITISH	TOSSED
GERMANY	MOSES	NINTENDO	BLACKISH	SQUEEZED
PORTUGAL	SIN	GAMECUBE	SILVERY	BLASTED
SWEDEN	HEAVEN	PSP	GREYISH	TANGLED
AUSTRIA	SALVATION	AMIGA	PALER	SLASHED

Deep Text Results

WSJ for POS, CHUNK (CoNLL 2000) & SRL (CoNLL 2005)

Reuters (CoNLL 2003) for NER

Approach	POS (% Err)	CHUNK (F1)	NER (F1)	SRL (F1)
Top Systems	2.76	94.39/94.13	89.31/88.76	77.92‡/74.76†
CNN	3.15	88.82	81.61	51.16
*Embed*CNN	2.78	94.18	88.88	71.81⋆/74.55†

Top Systems:

Toutanova et al. ('03) for POS

Ando & Zhang ('05) and Florian et al. for NER,

Sha et al. ('03) for CHUNK

Punyakanok et al. (2005) for SRL

‡ Uses the Charniak top-5 parse trees, and the Collins parse tree † Uses the Charniak parse tree only

Final Conclusion (really)

- New Deep Learning Method :
  - Unsupervised pairwise embedding.
  - Improves internal representation in NN.
- Applications: images, text, . . . web ?
- Software: `http://torch5.sourceforge.net`

Thanks!