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 (≤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

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. \(x\)  
  (b) Reconst. \(h^1\)  
  (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.* → *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 → 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:

\[
\min_{i,j=1}^{U} \sum L(f(x_i), f(x_j), W_{ij}), \quad f_i \in \mathbb{R}^d
\]

**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”: \(f^\top Df = I\) and \(f^\top D1 = 0\).
**Siamese Networks: functional embedding**

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

**DrLIM** [Hadsell et al.,’06 ]:

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

→ neighbors close, others have distance of at least $m$

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

- $f(x)$ not just a lookup-table → 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 $t \& t + 1 \rightarrow$ same label
  - audio: consecutive audio frames $\rightarrow$ same speaker + word ..
  - text: word + neighbors $\rightarrow$ 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:

<table>
<thead>
<tr>
<th>data set</th>
<th>classes</th>
<th>dims</th>
<th>points</th>
<th>labeled</th>
</tr>
</thead>
<tbody>
<tr>
<td>g50c</td>
<td>2</td>
<td>50</td>
<td>500</td>
<td>50</td>
</tr>
<tr>
<td>Text</td>
<td>2</td>
<td>7511</td>
<td>1946</td>
<td>50</td>
</tr>
<tr>
<td>Uspst</td>
<td>10</td>
<td>256</td>
<td>2007</td>
<td>50</td>
</tr>
<tr>
<td>Mnist1h</td>
<td>10</td>
<td>784</td>
<td>70k</td>
<td>100</td>
</tr>
<tr>
<td>Mnist6h</td>
<td>10</td>
<td>784</td>
<td>70k</td>
<td>600</td>
</tr>
<tr>
<td>Mnist1k</td>
<td>10</td>
<td>784</td>
<td>70k</td>
<td>1000</td>
</tr>
</tbody>
</table>

- First experiment: Only consider two-layer nets.
## Deep Semi-Supervised Results

<table>
<thead>
<tr>
<th>Method</th>
<th>g50c</th>
<th>Text</th>
<th>Uspst</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>8.32</td>
<td>18.86</td>
<td>23.18</td>
</tr>
<tr>
<td>SVMLight-TSVM</td>
<td>6.87</td>
<td>7.44</td>
<td>26.46</td>
</tr>
<tr>
<td>∇TSVM</td>
<td>5.80</td>
<td>5.71</td>
<td>17.61</td>
</tr>
<tr>
<td>LapSVM*</td>
<td>5.4</td>
<td>10.4</td>
<td>12.7</td>
</tr>
<tr>
<td>NN</td>
<td>8.54</td>
<td>15.87</td>
<td>24.57</td>
</tr>
<tr>
<td>EmbedNN*</td>
<td>5.66</td>
<td>5.82</td>
<td>15.49</td>
</tr>
<tr>
<td>Method</td>
<td>Mnist1h</td>
<td>Mnist6h</td>
<td>Mnist1k</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>SVM</td>
<td>23.44</td>
<td>8.85</td>
<td>7.77</td>
</tr>
<tr>
<td>TSVM</td>
<td>16.81</td>
<td>6.16</td>
<td>5.38</td>
</tr>
<tr>
<td>RBM(*)</td>
<td>21.5</td>
<td>-</td>
<td>8.8</td>
</tr>
<tr>
<td>SESM(*)</td>
<td>20.6</td>
<td>-</td>
<td>9.6</td>
</tr>
<tr>
<td>DBN-rNCA(*)</td>
<td>-</td>
<td>8.7</td>
<td>-</td>
</tr>
<tr>
<td>NN</td>
<td>25.81</td>
<td>11.44</td>
<td>10.70</td>
</tr>
<tr>
<td>Embed^O NN</td>
<td>17.05</td>
<td>5.97</td>
<td>5.73</td>
</tr>
<tr>
<td>Embed^I1 NN</td>
<td>16.86</td>
<td>9.44</td>
<td>8.52</td>
</tr>
<tr>
<td>Embed^A1 NN</td>
<td>17.17</td>
<td>7.56</td>
<td>7.89</td>
</tr>
<tr>
<td>CNN</td>
<td>22.98</td>
<td>7.68</td>
<td>6.45</td>
</tr>
<tr>
<td>Embed^O CNN</td>
<td>11.73</td>
<td>3.42</td>
<td>3.34</td>
</tr>
<tr>
<td>Embed^I5 CNN</td>
<td>7.75</td>
<td>3.82</td>
<td>2.73</td>
</tr>
<tr>
<td>Embed^A5 CNN</td>
<td>7.87</td>
<td>3.82</td>
<td>2.76</td>
</tr>
</tbody>
</table>
## Really Deep Results

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

<table>
<thead>
<tr>
<th>layers=</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NN</strong></td>
<td>26.0</td>
<td>26.1</td>
<td>27.2</td>
<td>28.3</td>
<td>34.2</td>
<td>47.7</td>
</tr>
<tr>
<td>EmbedNN(^O)</td>
<td>19.7</td>
<td>15.1</td>
<td>15.1</td>
<td>15.0</td>
<td>13.7</td>
<td>11.8</td>
</tr>
<tr>
<td>EmbedNN(^ALL)</td>
<td>18.2</td>
<td>12.6</td>
<td>7.9</td>
<td>8.5</td>
<td>6.3</td>
<td>9.3</td>
</tr>
</tbody>
</table>

- EmbedNN\(^O\): auxiliary 10-dim embedding on output layer
- EmbedNN\(^ALL\): 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)

EmbedNN 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 *videoCNN* with different video sources.

---

aThe 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.
<table>
<thead>
<tr>
<th>Method</th>
<th>30 objects</th>
<th>100 objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Neighbor</td>
<td>81.8</td>
<td>70.1</td>
</tr>
<tr>
<td>SVM</td>
<td>84.9</td>
<td>74.6</td>
</tr>
<tr>
<td>SpinGlass MRF</td>
<td>82.8</td>
<td>69.4</td>
</tr>
<tr>
<td>Eigen Spline</td>
<td>84.6</td>
<td>77.0</td>
</tr>
<tr>
<td>VTU</td>
<td>89.9</td>
<td>79.1</td>
</tr>
<tr>
<td>Standard CNN</td>
<td>84.88</td>
<td>71.49</td>
</tr>
<tr>
<td>videoCNN V:COIL100</td>
<td>-</td>
<td>92.25</td>
</tr>
<tr>
<td>videoCNN V:COIL“70”</td>
<td>95.03</td>
<td>-</td>
</tr>
<tr>
<td>videoCNN V:COIL-Like</td>
<td>-</td>
<td>79.77</td>
</tr>
<tr>
<td>videoCNN V:Animal</td>
<td>-</td>
<td>78.67</td>
</tr>
</tbody>
</table>

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

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

**Labeled data:** Wall Street Journal (~ 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 (∼ 631M words)

Bengio & Ducharme (2001)
Probability of next word given previous words

Pick word + neighborhood → $W_{ij} = 1$ (push together) +ve pair
“The cat sat on the” → ← “mat”

Same neighborhood + random word → $W_{ij} = 0$ (push apart)
“The cat sat on the” ← → “DBN” -ve pair
## Language Model: Embedding

<table>
<thead>
<tr>
<th>Country</th>
<th>Mythological Entity</th>
<th>Console</th>
<th>Colour</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRANCE</td>
<td>JESUS</td>
<td>XBOX</td>
<td>REDDISH</td>
<td>SCRATCHED</td>
</tr>
<tr>
<td></td>
<td>454</td>
<td>1973</td>
<td>6909</td>
<td>11724</td>
</tr>
<tr>
<td>SPAIN</td>
<td>CHRIST</td>
<td>PLAYSTATION</td>
<td>YELLOWISH</td>
<td>SMASHED</td>
</tr>
<tr>
<td>ITALY</td>
<td>GOD</td>
<td>DREAMCAST</td>
<td>GREENISH</td>
<td>RIPPED</td>
</tr>
<tr>
<td>RUSSIA</td>
<td>RESURRECTION</td>
<td>PS2</td>
<td>BROWNISH</td>
<td>BRUSHED</td>
</tr>
<tr>
<td>POLAND</td>
<td>PRAYER</td>
<td>SNES</td>
<td>BLUISH</td>
<td>HURLED</td>
</tr>
<tr>
<td>ENGLAND</td>
<td>YAHWEH</td>
<td>WII</td>
<td>CREAMY</td>
<td>GRABBED</td>
</tr>
<tr>
<td>DENMARK</td>
<td>JOSEPHUS</td>
<td>NES</td>
<td>WHITISH</td>
<td>TOSSED</td>
</tr>
<tr>
<td>GERMANY</td>
<td>MOSES</td>
<td>NINTENDO</td>
<td>BLACKISH</td>
<td>SQUEEZED</td>
</tr>
<tr>
<td>PORTUGAL</td>
<td>SIN</td>
<td>GAMECUBE</td>
<td>SILVERY</td>
<td>BLASTED</td>
</tr>
<tr>
<td>SWEDEN</td>
<td>HEAVEN</td>
<td>PSP</td>
<td>GREYISH</td>
<td>TANGLED</td>
</tr>
<tr>
<td>AUSTRIA</td>
<td>SALVATION</td>
<td>AMIGA</td>
<td>PALER</td>
<td>SLASHED</td>
</tr>
</tbody>
</table>
## Deep Text Results

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

**Reuters** (CoNLL 2003) for NER

<table>
<thead>
<tr>
<th>Approach</th>
<th>POS (% Err)</th>
<th>CHUNK (F1)</th>
<th>NER (F1)</th>
<th>SRL (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Systems</td>
<td>2.76</td>
<td>94.39/94.13</td>
<td>89.31/88.76</td>
<td>77.92‡/74.76†</td>
</tr>
<tr>
<td>CNN</td>
<td>3.15</td>
<td>88.82</td>
<td>81.61</td>
<td>51.16</td>
</tr>
<tr>
<td>EmbedCNN</td>
<td>2.78</td>
<td>94.18</td>
<td>88.88</td>
<td>71.81*†</td>
</tr>
</tbody>
</table>

**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!