Deeply-Supervised Nets
AISTATS, 2015
Deep Learning Workshop, NIPS 2014

Zhuowen Tu
Department of Cognitive Science
Department of Computer Science and Engineering (affiliate)
University of California, San Diego (UCSD)

with Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zengyou Zhang

Funding support: NSF IIS-1360566, NSF IIS-1360568
Artificial neural networks: a brief history

Visual representation

**Frontal lobe:**
- motor control,
- decisions and judgments, emotions
- language production

**Temporal lobe:**
- Visual perception, object recognition, auditory processing
- Memory
- Language comprehension

**Parietal lobe:**
- Attention
- Spatial cognition
- Perception of stimuli related to touch, pressure, temperature, pain

**Occipital lobe:**
- Vision

**dorsal stream:** “where”

**ventral stream:** “what”

Hubel and Wiesel Model

Visual representation

- Simple units combine afferent units with different selectivities (here different orientations).
- Complex units combine afferent units with the same selectivities but slightly different positions and scales.
- V2/V4-like simple units tuned to the non-linear combination of multiple orientations.
- V1-like complex unit tuned to a single orientation and tolerant with respect to the exact position and scale of the bar within its receptive field.
Visual cortical areas - human

HMax Framework (Serre et al.)

Serre, Oliva, and Poggio 2007

Kobatake and Tanaka, 1994
Motivation

• Make feature representation learnable instead of hand-crafting it.

SIFT [1]

HOG [2]

[1] Lowe, David G. "Object recognition from local scale-invariant features". ICCV 1999
Motivation

- Make feature representation learnable instead of hand-crafting it.

History of ConvNets

Fukushima 1980
Neocognitron

Rumelhart, Hinton, Williams 1986
“T” versus “C” problem

LeCun et al. 1989-1998
Hand-written digit reading

Krizhevksy, Sutskever, Hinton 2012
ImageNet classification breakthrough
“SuperVision” CNN

From Ross Girshick
Problem of Current CNN

• Current CNN architecture is mostly based on the one developed in 1998.
• Hidden layers of CNN lack transparency during training.
• Exploding and vanishing gradients presence during back propagation training [1,2].

Deeply-Supervised Nets

To boost the classification performance by focusing three aspects:

- Robustness and discriminativeness of the learned features.
- Transparency of the hidden layers on the overall classification.
- Training difficulty due to the “exploding” and "vanishing" gradients.
Some Definitions

Input training set: \( S = \{(X_i, y_i), i = 1..N\} \)

\[ X_i \in \mathbb{R}^n, y_i \in \{1..K\} \]

Recursive functions:

Features: \( Z^{(m)} = f(Q^{(m)}), \text{and } Z^{(0)} \equiv X \)

\[ Q^{(m)} = W^{(m)} \ast Z^{(m-1)} \]

Summarize all the parameters as:

\[ W = (W^{(1)}, ..., W^{(\text{out})}) \]

In addition, we have SVM weights (to be discarded after training)

\[ w = (w^{(1)}, ..., w^{(M-1)}) \]
Proposed Method

- Deeply-Supervised Nets (DSN)
- Direct supervision to intermediate layers to learn weights $W, w$
Formulations

standard objective function for SVM

\[ \|w^{(out)}\|^2 + \mathcal{L}(W, w^{(out)}) + \sum_{m=1}^{M-1} \alpha_m [\|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma]^+ \]

\[ \mathcal{L}(W, w^{(out)}) = \sum_{y_k \neq y} [1 - <w^{(out)}, \phi(Z^{(M)}, y) - \phi(Z^{(M)}, y_k)>]^2_+ \]

\[ \ell(W, w^{(m)}) = \sum_{y_k \neq y} [1 - <w^{(m)}, \phi(Z^{(m)}, y) - \phi(Z^{(m)}, y_k)>]^2_+ \]

Hidden layer supervision

Multi-class hinge loss between responses $Z$ and true label $y$
Formulations

• The gradient of the objective function w.r.t the weights:

\[
\frac{\partial F}{\partial w^{(out)}} = 2w^{(out)} - 2 \sum_{y_k \neq y} [\phi(z^{(M)}, y) - \phi(z^{(M)}, y)] [1 - \langle w^{(out)}, \phi(z^{(M)}, y) - \phi(z^{(M)}, y_k) \rangle] +
\]

\[
\frac{\partial F}{\partial w^{(m)}} = \begin{cases} 
0, \text{ when } \|w^{(m)}\|^2 + \ell(W, w^{(m)}) \leq \gamma \\
\alpha_m \left\{ 2w^{(m)} - 2 \sum_{y_k \neq y} [\phi(z^{(m)}, y) - \phi(z^{(m)}, y_k)] [1 - \langle w^{(m)}, \phi(z^{(m)}, y) - \phi(z^{(m)}, y_k) \rangle] + \right\}, \text{ otherwise.}
\end{cases}
\]

• Apply the computed gradients to perform stochastic gradient descend and then iteratively train our DSN model.
Greedy layer-wise supervised pretraining
(Bengio et al. 2007)

Essentially shown to be ineffective (worse than unsupervised pre-training).

<table>
<thead>
<tr>
<th></th>
<th>Experiment 2</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>train.</td>
<td>valid.</td>
<td>test</td>
<td>train.</td>
<td>valid.</td>
</tr>
<tr>
<td>DBN, unsupervised pre-training</td>
<td>0%</td>
<td>1.2%</td>
<td>1.2%</td>
<td>0%</td>
<td>1.5%</td>
<td>1.5%</td>
</tr>
<tr>
<td>Deep net, auto-associator pre-training</td>
<td>0%</td>
<td>1.4%</td>
<td>1.4%</td>
<td>0%</td>
<td>1.4%</td>
<td>1.6%</td>
</tr>
<tr>
<td>Deep net, supervised pre-training</td>
<td>0%</td>
<td>1.7%</td>
<td>2.0%</td>
<td>0%</td>
<td>1.8%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Deep net, no pre-training</td>
<td>.004%</td>
<td>2.1%</td>
<td>2.4%</td>
<td>.59%</td>
<td>2.1%</td>
<td>2.2%</td>
</tr>
<tr>
<td>Shallow net, no pre-training</td>
<td>.004%</td>
<td>1.8%</td>
<td>1.9%</td>
<td>3.6%</td>
<td>4.7%</td>
<td>5.0%</td>
</tr>
</tbody>
</table>
Deeply-supervised nets

\[ \|w^{(out)}\|^2 + \mathcal{L}(W, w^{(out)}) + \sum_{m=1}^{M-1} \alpha_m [\|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma] + \]

\[ F(W) \equiv \mathcal{P}(W) + \mathcal{Q}(W) \]

\[ \mathcal{P}(W) \equiv \|w^{(out)}\|^2 + \mathcal{L}(W, w^{(out)}) \]

\[ \mathcal{Q}(W) \equiv \sum_{m=1}^{M-1} \alpha_m [\|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma] + \]
With a loose assumption

\[ F(W') \geq F(W) + <g, W' - W> + \frac{\lambda}{2} \|W' - W\|^2 \]

\[ F(W) \equiv \mathcal{P}(W) + \mathcal{Q}(W) \]

**Definition** We denote by \( S_\gamma(F) = \{W \mid F(W) \leq \gamma\} \) the \( \gamma \)-sublevel set, stated here for the function \( F(W) \equiv \mathcal{P}(W) + \mathcal{Q}(W) \).

**Lemma 1** \( \forall m, m' = 1 \ldots M - 1, \text{ and } m' > m \) if

\[ \|w^{(m)}\|^2 + \ell((\hat{W}^{(1)}, \ldots, \hat{W}^{(m)}), w^{(m)}) \leq \gamma \]

then there exists \((\hat{W}^{(1)}, \ldots, \hat{W}^{(m)}, \ldots, \hat{W}^{(m')})\) such that

\[ \|w^{(m')}\|^2 + \ell((\hat{W}^{(1)}, \ldots, \hat{W}^{(m)}, \ldots, \hat{W}^{(m')}), w^{(m')}) \leq \gamma. \]
With a loose assumption

\[ F(W') \geq F(W) + \langle g, W' - W \rangle + \frac{\lambda}{2} \|W' - W\|^2 \]

\[ F(W) \equiv P(W) + Q(W) \]

**Lemma 2** Suppose \( \mathbb{E}[\|\hat{g}p_t\|^2] \leq G^2 \) and \( \mathbb{E}[\|\hat{g}q_t\|^2] \leq G^2 \), and we use the update rule of \( W_{t+1} = \Pi_{\mathcal{W}}(W_t - \eta_t(\hat{g}p_t + \hat{g}q_t)) \) where \( \mathbb{E}[\hat{g}p_t] = gp_t \) and \( \mathbb{E}[\hat{g}q_t] = gq_t \). If we use \( \eta_t = 1/(\lambda_1 + \lambda_2)t \), then at iteration \( T \)

\[ \mathbb{E}[\|W_T - W^*\|^2] \leq \frac{4G^2}{(\lambda_1 + \lambda_2)^2T} \]

A loose assumption

\[ F(W') \geq F(W) + \langle g, W' - W \rangle + \frac{\lambda}{2} \|W' - W\|^2 \]

\[ F(W) \equiv \mathcal{P}(W) + \mathcal{Q}(W) \]

**Lemma 3** We follow the assumptions in lemma 2, with the exception that we assume \( \eta_t = 1/t \) since \( \lambda_1 \) and \( \lambda_2 \) are not always readily available; as we discuss in the appendix, we also expect the combined \( \lambda \) to be small. When we begin in the region \( \|W_1 - W^*\|^2 \leq D \), the convergence rate is bounded by

\[
\mathbb{E}[\|W_T - W^*\|^2] \leq e^{-2\lambda(\ln(T-1) + 0.578)} D + 4G^2 \sum_{t=1}^{T-1} \frac{1}{t^2} \left( \frac{t}{T-1} \right)^{2\lambda}
\]
A loose assumption

\[ \|w^{(out)}\|^2 + \mathcal{L}(\mathcal{W}, w^{(out)}) + \sum_{m=1}^{M-1} \alpha_m [\|w^{(m)}\|^2 + \ell(\mathcal{W}, w^{(m)}) - \gamma] + \]

\[ F(\mathcal{W}) \equiv \mathcal{P}(\mathcal{W}) + \mathcal{Q}(\mathcal{W}) \]

**Theorem 1** Let \( \mathcal{P}(\mathcal{W}) \) be \( \lambda_1 \)-strongly convex and \( \mathcal{Q}(\mathcal{W}) \) be \( \lambda_2 \)-strongly convex near optimal \( \mathcal{W}^* \) and denote by \( \mathcal{W}_T^{(F)} \) and \( \mathcal{W}_T^{(P)} \) the solution after \( T \) iterations when following SGD on \( F(\mathcal{W}) \) and \( \mathcal{P}(\mathcal{W}) \), respectively. Then DSN framework improves the relative convergence speed \( \frac{\mathbb{E}[\|W_T^{(P)} - W^*\|^2]}{\mathbb{E}[\|W_T^{(F)} - W^*\|^2]} \), viewed from the ratio of their upper bounds as \( \Theta(\frac{(\lambda_1 + \lambda_2)^2}{\lambda_1^2}) \), when \( \eta_t = 1/\lambda t \).
A loose assumption

\[ F(W') \geq F(W) + \langle g, W' - W \rangle + \frac{\lambda}{2} \|W' - W\|^2 \]

\[ F(W) = \mathcal{P}(W) + \mathcal{Q}(W) \]

\[ \mathbb{E}[\|W_T - W^*\|^2] \leq e^{-2\lambda(\ln(T-1) + 0.578)} D + 4G^2 \sum_{t=1}^{T-1} \frac{1}{t^2} \left( \frac{t}{T-1} \right)^{2\lambda} \]

Remark: When \( \eta_t = 1/t \), \( T \) is large, and the first term dominates, the upper bound ratio for \( \frac{\mathbb{E}[\|W_{T}^{(\mathcal{P})} - W^*\|^2]}{\mathbb{E}[\|W_{T}^{(\mathcal{F})} - W^*\|^2]} \) is roughly at the order of \( \Theta(e^{2\ln(T)\lambda_2}) \). If the second term dominates, the convergence of \( F(W) \) over \( \mathcal{P}(W) \) is also advantageous with the ratio at an order of \( \Theta(e^{2\ln((T-1)/(T-2))\lambda_2}) \) loosely.
Network-in-Network
(M. Lin, Q. Chen, and S. Yan, ICLR 2014)
Some alternative formulations

1. Constrained optimization:

\[
\text{minimize } \| w^{\text{(out)}} \|^2 + \mathcal{L}(W, w^{\text{(out)}}) \\
\text{subject to } \| w^{(m)} \|^2 + \ell(W, w^{(m)}) \leq \gamma, m = 1..M - 1
\]

2. Fixed \( \alpha(m) \):

\[
\text{minimize } \| w^{\text{(out)}} \|^2 + \mathcal{L}(W, w^{\text{(out)}}) + \sum_{m=1}^{M-1} \alpha_m \left( \| w^{(m)} \|^2 + \ell(W, w^{(m)}) - \gamma \right)_{+}
\]

3. Decay function for \( \alpha(m) \):

\[
\text{minimize } \| w^{\text{(out)}} \|^2 + \mathcal{L}(W, w^{\text{(out)}}) + \sum_{m=1}^{M-1} \alpha_m \left( \| w^{(m)} \|^2 + \ell(W, w^{(m)}) - \gamma \right)_{+}
\]

\[
\alpha(m) \equiv c(m) \left( 1 - \frac{t}{N} \right)
\]
Experiment on the MNIST dataset

MNIST testing error

- CNN–Softmax [0.56%]
- DSN–Softmax (ours) [0.51%]
- CNN–SVM [0.48%]
- DSN–SVM (ours) [0.39%]

Error (%) vs. epoch

- CNN–Softmax
- DSN–Softmax (ours)
- CNN–SVM
- DSN–SVM (ours)

Training and testing error

- CNN–SVM: Training error [0.03%]
- CNN–SVM: Testing error [0.50%]
- DSN–SVM: Training error [0.03%]
- DSN–SVM: Testing error [0.39%]
Some empirical results
## Experiment on the MNIST dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.53</td>
</tr>
<tr>
<td>Stochastic Pooling</td>
<td>0.47</td>
</tr>
<tr>
<td>Network in Network</td>
<td>0.47</td>
</tr>
<tr>
<td>Maxout Network</td>
<td>0.45</td>
</tr>
<tr>
<td>CNN (layer-wise pre-training)</td>
<td>0.43</td>
</tr>
<tr>
<td>DSN (ours)</td>
<td>0.39</td>
</tr>
</tbody>
</table>

## MNIST training details

<table>
<thead>
<tr>
<th>layer</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>stride 2, kernel 5x5, relu, channel_output 32</td>
</tr>
<tr>
<td>+ L2SVM</td>
<td>input conv1 (after max pooling), squared hinge loss</td>
</tr>
<tr>
<td>conv2</td>
<td>stride 2, kernel 5x5, relu, channel_output 64</td>
</tr>
<tr>
<td>+ L2SVM</td>
<td>input conv2 (after max pooling), squared hinge loss</td>
</tr>
<tr>
<td>fc3</td>
<td>relu, channel_output 500, dropout rate 0.5</td>
</tr>
<tr>
<td>fc4</td>
<td>channel_output 10</td>
</tr>
<tr>
<td><strong>Output layer: L2SVM</strong></td>
<td>squared hinge loss</td>
</tr>
</tbody>
</table>

**110 epochs**

\[
Q(W) \equiv \sum_{m=1}^{M-1} \alpha_m \left[ \|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma \right]_+.
\]

- Base learning rate = 0.4.
- \( \alpha_m = 0.1 \times \left(1 - \frac{t}{N}\right) \)
## CIFAR results

### CIFAR-10 classification error.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Data Augmentation</td>
<td></td>
</tr>
<tr>
<td>Stochastic Pooling</td>
<td>15.13</td>
</tr>
<tr>
<td>Maxout Networks</td>
<td>11.68</td>
</tr>
<tr>
<td>Network in Network (NIN)</td>
<td>10.41</td>
</tr>
<tr>
<td>NIN (layer-wise pre-training)</td>
<td>9.92</td>
</tr>
<tr>
<td><strong>DSN (ours)</strong></td>
<td><strong>9.69</strong></td>
</tr>
</tbody>
</table>

### CIFAR-100 classification error.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Pooling</td>
<td>42.51</td>
</tr>
<tr>
<td>Maxout Networks</td>
<td>38.57</td>
</tr>
<tr>
<td>Tree based Priors</td>
<td>36.85</td>
</tr>
<tr>
<td>Network in Network</td>
<td>35.68</td>
</tr>
<tr>
<td><strong>DSN (ours)</strong></td>
<td><strong>34.57</strong></td>
</tr>
</tbody>
</table>

### With Data Augmentation

<table>
<thead>
<tr>
<th>Method</th>
<th>Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxout Networks</td>
<td>9.38</td>
</tr>
<tr>
<td>Dropconnect</td>
<td>9.32</td>
</tr>
<tr>
<td>Network in Network</td>
<td>8.81</td>
</tr>
<tr>
<td><strong>DSN (ours)</strong></td>
<td><strong>7.97</strong></td>
</tr>
</tbody>
</table>

DSN on CIFAR-10 training details

<table>
<thead>
<tr>
<th>layer</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>stride 2, kernel 5x5, channel_output 192</td>
</tr>
<tr>
<td>+ L2SVM</td>
<td>input conv1 (before relu), squared hinge loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 160, 96, dropout 0.5</td>
</tr>
<tr>
<td>conv2</td>
<td>stride 2, kernel 5x5, channel_output 192</td>
</tr>
<tr>
<td>+ L2SVM</td>
<td>input conv2 (before relu), squared hinge loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 192, 192, dropout rate 0.5</td>
</tr>
<tr>
<td>conv3</td>
<td>stride 1, kernel 3x3, relu, channel_output 192</td>
</tr>
<tr>
<td>+ L2SVM</td>
<td>input conv3 (before relu), squared hinge loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 192, 192, dropout rate 0.5, global average pooling</td>
</tr>
<tr>
<td>Output layer: L2SVM</td>
<td>input global average pooling, squared hinge loss</td>
</tr>
</tbody>
</table>

400 epochs

\[ Q(W) = \sum_{m=1}^{M-1} \alpha_m [\|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma]_+ \]

- Base learning rate = 0.025, reduce learning rate twice by a factor of 20.
- \( \alpha_m = 0.001 \) fixed for all companion objectives.
- The companion objectives vanish after 100 epochs \( \equiv \gamma(0.8, 0.8, 1.4) \) for each layer,
### DSN on CIFAR-100 training details

<table>
<thead>
<tr>
<th>layer</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>stride 2, kernel 5x5, channel_output 192</td>
</tr>
<tr>
<td>+ SOFTMAX</td>
<td>input conv1 (before relu), softmax loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 160, 96, dropout 0.5</td>
</tr>
<tr>
<td>conv2</td>
<td>stride 2, kernel 5x5, channel_output 192</td>
</tr>
<tr>
<td>+ SOFTMAX</td>
<td>input conv2 (before relu), softmax loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 192, 192, dropout rate 0.5</td>
</tr>
<tr>
<td>conv3</td>
<td>stride 1, kernel 3x3, relu, channel_output 192</td>
</tr>
<tr>
<td>+ SOFTMAX</td>
<td>input conv3 (before relu), softmax loss</td>
</tr>
<tr>
<td>2 NIN layers</td>
<td>1x1 conv, channel_output 192, 10, dropout rate 0.5, global average pooling</td>
</tr>
<tr>
<td>Output layer: SOFTMAX</td>
<td>input global average pooling, softmax loss</td>
</tr>
</tbody>
</table>

400 epochs: \( Q(W) = \sum_{m=1}^{M-1} \alpha_m \|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma \).+.

- Hyper-parameters and epoch schedules are identical to those in CIFAR-10
- The only difference is using Softmax classifiers instead of L2SVM classifiers
## Result on the SVHN dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Pooling</td>
<td>2.80</td>
</tr>
<tr>
<td>Maxout Networks</td>
<td>2.47</td>
</tr>
<tr>
<td>Network in Network</td>
<td>2.35</td>
</tr>
<tr>
<td>Dropconnect</td>
<td>1.94</td>
</tr>
<tr>
<td>DSN (ours)</td>
<td>1.92</td>
</tr>
</tbody>
</table>

Visualization of learned features

**ImageNet**

## Results on ImageNet

<table>
<thead>
<tr>
<th>Method</th>
<th>top-1 val. error(%)</th>
<th>top-5 val. error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN 8-layer</td>
<td>40.7</td>
<td>18.2</td>
</tr>
<tr>
<td>DSN 8-layer (ours)</td>
<td>39.6</td>
<td>17.8</td>
</tr>
<tr>
<td>CNN 11-layer</td>
<td>34.5</td>
<td>13.9</td>
</tr>
<tr>
<td>DSN 11-layer (ours)</td>
<td>33.6</td>
<td>13.1</td>
</tr>
</tbody>
</table>
## DSN on ImageNet 2012 training details

<table>
<thead>
<tr>
<th>layer</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>stride 4, kernel 11x11, relu, channel_output 64</td>
</tr>
<tr>
<td>conv2</td>
<td>stride 1, kernel 5x5, relu, channel_output 192</td>
</tr>
<tr>
<td>conv3</td>
<td>stride 1, kernel 3x3, relu, channel_output 384</td>
</tr>
<tr>
<td>conv4</td>
<td>stride 1, kernel 3x3, relu, channel_output 256</td>
</tr>
<tr>
<td>+ SOFTMAX</td>
<td>softmax loss</td>
</tr>
<tr>
<td>conv5</td>
<td>stride 1, kernel 3x3, relu, channel_output 256</td>
</tr>
<tr>
<td>fc6</td>
<td>channel_output 4096, dropout rate 0.5</td>
</tr>
<tr>
<td>fc7</td>
<td>channel_output 4096, dropout rate 0.5</td>
</tr>
<tr>
<td>fc8</td>
<td>channel_output 1000</td>
</tr>
<tr>
<td>Output layer</td>
<td>softmax loss</td>
</tr>
</tbody>
</table>

N=90 epochs

- Base learning rate = 0.01 with decay factor 0.1. Learning rate is decayed whenever validation error stop decreasing until it reaches $10^{-5}$
- The companion objectives are weighted by $\alpha = 0.4$ with decay factor $= \left(1 - \frac{t}{N}\right)$ where $t$ is current epoch index and $N$ is the number of total epoch.
Relation to prior work

- M. A. Carreira-Perpinan and W. Wang. "Distributed optimization of deeply nested systems: \( \|w^{(out)}\|^2 + \mathcal{L}(W, w^{(out)}) + \sum_{m=1}^{M-1} \alpha_m \|w^{(m)}\|^2 + \ell(W, w^{(m)}) - \gamma\) +,


Figure 1. Three modes of embedding in deep architectures.
Conclusions of DSN

• For relatively shallow networks, DSN provides a strong regularization to reduce the test error.

• For very deep networks, DSN greatly relieves the vanishing gradient problem that makes the learning process otherwise very hard to train.

• We provide a new DL formulation and analysis, but many problems remain open.
Thank you!

Questions?