Large-Scale Feature Learning with Spike-and-Slab Sparse Coding

Ian J. Goodfellow, Aaron Courville, Yoshua Bengio

ICML 2012

Presented by Xin Yuan

January 17, 2013

Outline

- Contributions
- Spike-and-Slab Sparse Coding (S3C) Model
- Algorithm: Variational EM for S3C
- Results

Contributions

- Introduce *a new feature learning and extraction procedure* based on a factor model, *Spike-and-Slab Sparse Coding* (S3C).
- Overcome *two major scaling challenges*.

1) Scale inference in the spike-and-slab coding model to work for the large problem sizes required for object recognition.
2) Use the enhanced regularization properties of spike-and-slab sparse coding to scale object recognition techniques to work with large numbers of classes and small amounts of labeled data.

The Spike-and-Slab Sparse Coding Model

1. *Latent binary* spike variables $h \in \{0,1\}^N$
2. *Latent real*-valued slab variables, $s \in \mathbb{R}^N$
3. *Real*-valued D-dimensional visible vector $v \in \mathbb{R}^D$

$$\boxed{\begin{aligned} &\forall i \in \{1,\ldots,N\}, d \in \{1,\ldots,D\},\\ &p(h_i = 1) = \sigma(b_i)\\ &p(s_i \mid h_i) = \mathcal{N}(s_i \mid h_i \mu_i, \alpha_{ii}^{-1})\\ &p(v_d \mid s, h) = \mathcal{N}(v_d \mid W_{d:}(h \circ s), \beta_{dd}^{-1}) \end{aligned}}$$

σ : the logistic sigmoid function b : a set of biases on the spike variables

μ : governs the linear dependence of s on h. $h \circ s$: the element-wise product of h and s.

W: governs the linear dependence of v on s. The columns of W have unit norm.

α and β are diagonal precision matrices. The state of a hidden unit is best understood as $h_i s_i$

Comparison to Sparse Coding

1. One can derive the S3C model from sparse coding by replacing the factorial Cauchy or Laplace prior with a spike-and-slab prior.
2. One drawback of sparse coding is that the latent variables are not merely encouraged to be sparse; they are encouraged to remain close to 0, even when they are active. S3C uses b and s.
3. Another drawback of sparse coding is that the factors are not actually sparse in the generative distribution.
4. Sparse coding is also difficult to integrate into a deep generative model of data such as natural images.

Comparison to Restricted Boltzmann Machines

- Energy based model:

$$
\begin{aligned}
E(v,s,h) &= \frac{1}{2}\left(v - \sum_i W_i s_i h_i\right)^T \beta \left(v - \sum_i W_i s_i h_i\right) \\
&+ \frac{1}{2}\sum_{i=1}^{N} \alpha_i (s_i - \mu_i h_i)^2 - \sum_{i=1}^{N} b_i h_i, \qquad (2)
\end{aligned}
$$

S3C moves from an *undirected* ssRBM model to the *directed* graphical model.

$$\boxed{\tfrac{1}{2}(h \circ s)^T W^T \beta W (h \circ s)}$$

A variant of the μ-ssRBM

$$
\begin{aligned}
E(v,s,h) &= -\sum_{i=1}^{N} v^T \beta W_i s_i h_i + \frac{1}{2} v^T \beta v \\
&+ \frac{1}{2}\sum_{i=1}^{N} \alpha_i (s_i - \mu_i h_i)^2 - \sum_{i=1}^{N} b_i h_i, \quad (3)
\end{aligned}
$$

Effects:
1. Partition function
2. Posterior
3. Prior

Review of RBM(1)

A **restricted Boltzmann machine** (**RBM**) is a generative stochastic neural network that can learn a probability distribution over its set of inputs.

RBMs are a variant of Boltzmann machines, with the restriction that their neurons must form a bipartite graph: they have input units, corresponding to features of their inputs, hidden units that are trained, and each connection in an RBM must connect a visible unit to a hidden unit.

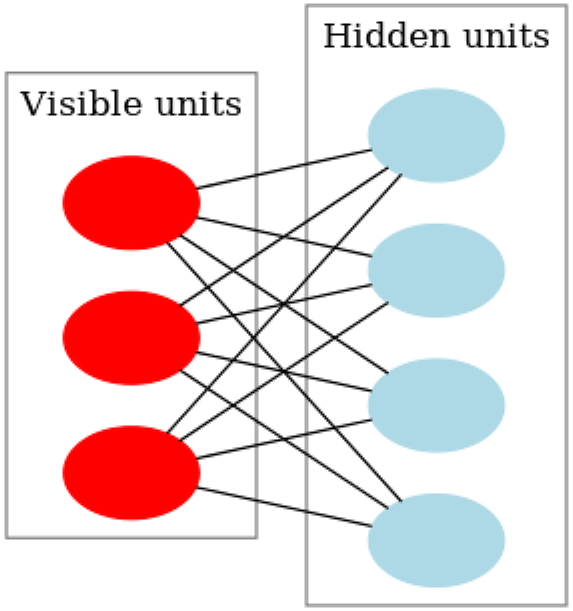


Diagram of a restricted Boltzmann machine with three visible units and four hidden units (no bias units).

RBMs have found applications in dimensionality reduction, classification, collaborative filtering and topic modeling. They can be trained in either supervised or unsupervised ways, depending on the task.

Review of RBM(2)

A joint conguration, (*v; h*) of the visible and hidden units has an energy given by:

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i\in\text{visible}} a_i v_i - \sum_{j\in\text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function:

$$p(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \qquad Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

The probability that the network assigns to a visible vector,

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model} \qquad \Delta w_{ij} = \epsilon(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model})$$

$$p(h_j = 1 \mid \mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}) \qquad p(v_i = 1 \mid \mathbf{h}) = \sigma(a_i + \sum_j h_j w_{ij})$$

Geoffrey Hinton (2010). *A Practical Guide to Training Restricted Boltzmann Machines*. UTML TR 2010–003, University of Toronto.

Effects: from Undirected Modeling to Directed Modeling

1. Partition function

   The partition function becomes tractable.

2. Posterior

   RBMs have a factorial posterior, but S3C and sparse coding have a complicated posterior due to the “explaining away” effect.

   Being able to selectively activate a small set of features that cooperate to explain the input likely provides S3C a major advantage in discriminative capability.

3. Prior

   S3C introduces a factorial prior.

   This probably makes it a poor generative model, but this is not a problem for the purpose of feature discovery.

Variational EM for S3C

- **A variant of the EM algorithm**

1) E-step, compute a variational approximation to the posterior rather than the posterior itself.

2) M-step, a closed-form solution.

Online learning with small gradient steps on the M-step objective worked better in practice.

The goal of the variational E-step is to maximize the energy functional with respect to a distribution Q over the unobserved variables.

Selecting Q that minimizes the KL divergence: $\mathcal{D}_{KL}(Q(h,s) \| P(h,s|v))$

$Q(h,s)$ is drawn from a restricted family of distributions.
This family can be chosen to ensure that Q is tractable.

$$Q(h_i) = \hat{h}_i,$$
$$Q(s_i \mid h_i) = \mathcal{N}(s_i \mid h_i \hat{s}_i,\ (\alpha_i + h_i W_i^T \beta W_i)^{-1})$$

Parallel Updates to All Units (1)

1) Start each iteration by partially minimizing the KL divergence with respect to $\hat{s}$.

The terms of the KL divergence that depend on $\hat{s}$ make up a quadratic function so this can be minimized via conjugate gradient descent.

Implement conjugate gradient descent efficiently by using the **R-operator** to perform *Hessian vector* products rather than computing the entire Hessian explicitly.

This step is guaranteed to improve the KL divergence on each iteration.

2) Next update $\hat{h}$ in parallel, shrinking the update by a damping coefficient.

This approach is not guaranteed to decrease the KL divergence on each iteration but it is a widely applied approach.

Pearlmutter, B. A. (1994). Fast exact multiplication by the Hessian. *Neural Computation, 6*(1), 147–160.

Parallel Updates to All Units (2)

Replace the conjugate gradient update to $\hat{s}$ with a more heuristic approach.

In practice *faster* convergence, reaching equally good solutions.

Use a parallel damped update on $\hat{s}$ much like $\hat{h}$.

An additional heuristic modication to the update rule which is made necessary by the unbounded nature of $\hat{s}$.
Clip the update to $\hat{s}$ so that if $\hat{s}_{\text{new}}$ has the opposite sign from $\hat{s}$, its magnitude is at most $\hat{s}$.
This prevents a case where multiple mutually inhibitory s units inhibit each other so strongly that rather than being driven to 0 they change sign and actually increase in magnitude. This case is a failure mode of the parallel updates that can result in $\hat{s}$ amplifying without bound if clipping is not used.

Algorithm 1 Fixed-Point Inference

```
Let K be a user-defined number of inference updates to run (e.g. 20)
Initialize $\hat{h}^{(0)} = \sigma(b)$ and $\hat{s}^{(0)} = \mu$.
for k=0:K do
    Compute the individually optimal value $\hat{s}_i^*$ for each $i$ simultaneously:
        $$\hat{s}_i^* = \frac{\mu_i \alpha_{ii} + v^T \beta W_i - W_i \beta \left[\sum_{j \neq i} W_j \hat{h}_j \hat{s}_j^{(k)}\right]}{\alpha_{ii} + W_i^T \beta W_i}$$
    Clip reflections by assigning
        $$c_i = \rho \text{sign}(\hat{s}_i^*) |\hat{s}_i^{(k)}|$$
    for all $i$ such that $\text{sign}(\hat{s}_i^*) \neq \text{sign}(\hat{s}_i^{(k)})$ and $|\hat{s}_i^*| > \rho |\hat{s}_i^{(k)}|$, and
    assigning $c_i = \hat{s}_i^*$ for all other $i$.
    Damp the updates by assigning
        $$\hat{s}_i^{(k+1)} = \eta c + (1 - \eta) \hat{s}^{(k)}$$
    where $\eta \in (0, 1]$.
    Compute the individually optimal values for $\hat{h}$:
        $$\hat{h}_i^* = \sigma\left(\left(v - \sum_{j \neq i} W_j \hat{s}_j^{(k+1)} \hat{h}_j^{(k)} - \frac{1}{2} W_i \hat{s}_i^{(k+1)}\right)^T \beta W_i \hat{s}_i^{(k+1)} + b_i - \frac{1}{2} \alpha_{ii} (\hat{s}_i^{(k+1)} - \mu_i)^2 - \frac{1}{2} \log(\alpha_{ii} + W_i^T \beta W_i) + \frac{1}{2} \log(\alpha_{ii})\right)$$
    Damp the update to $\hat{h}$:
        $$\hat{h}^{(k+1)} = \eta \hat{h}^* + (1 - \eta) \hat{h}^{(k)}$$
end for
```

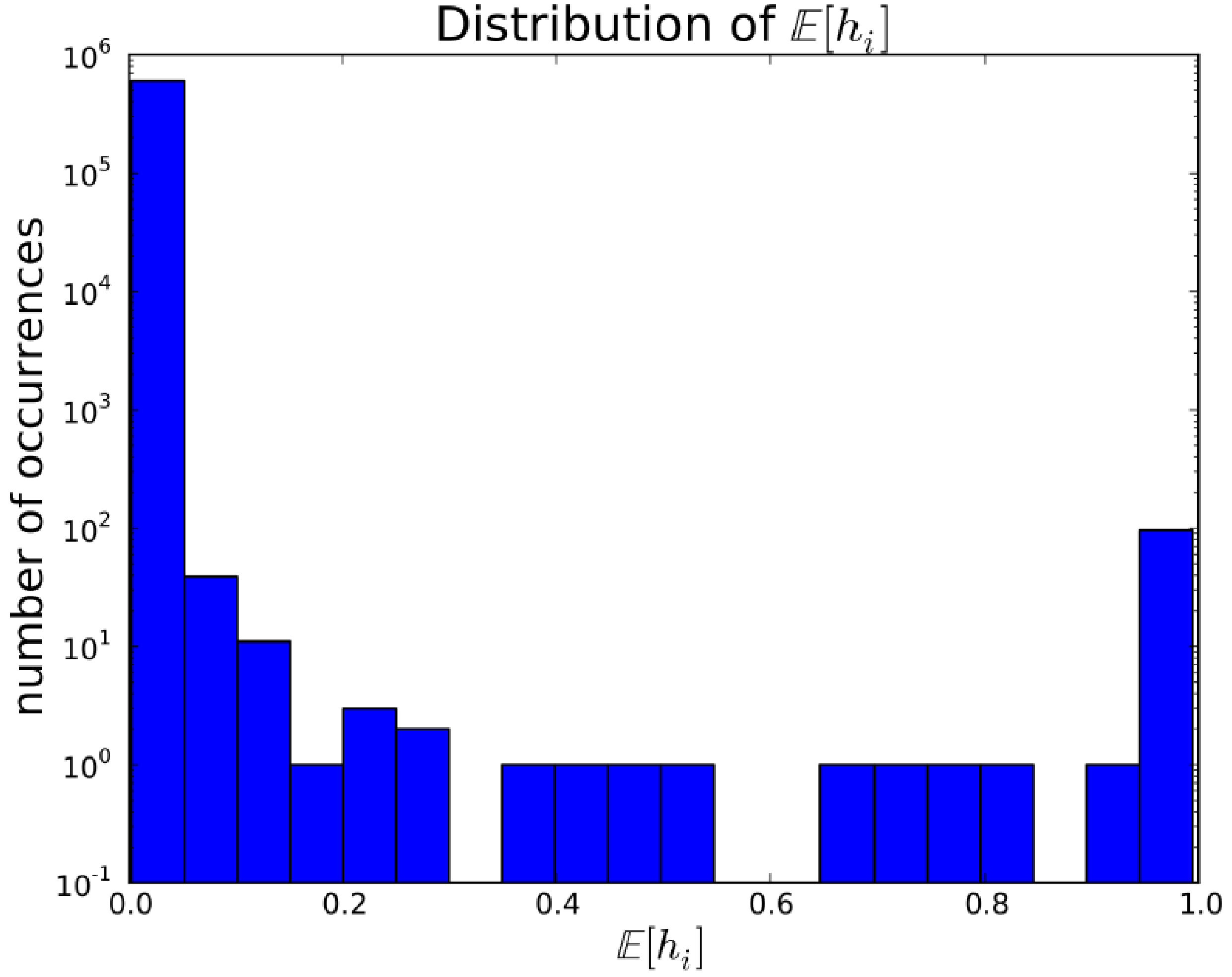


Figure 1. Q imposes a sparse distribution on h; $Q(h_i) < .01$ 99.7% of the time in this example histogram of values of $Q(h_i)$ for 6,000 different hidden units from a trained model applied to 100 different 6×6 image patches.

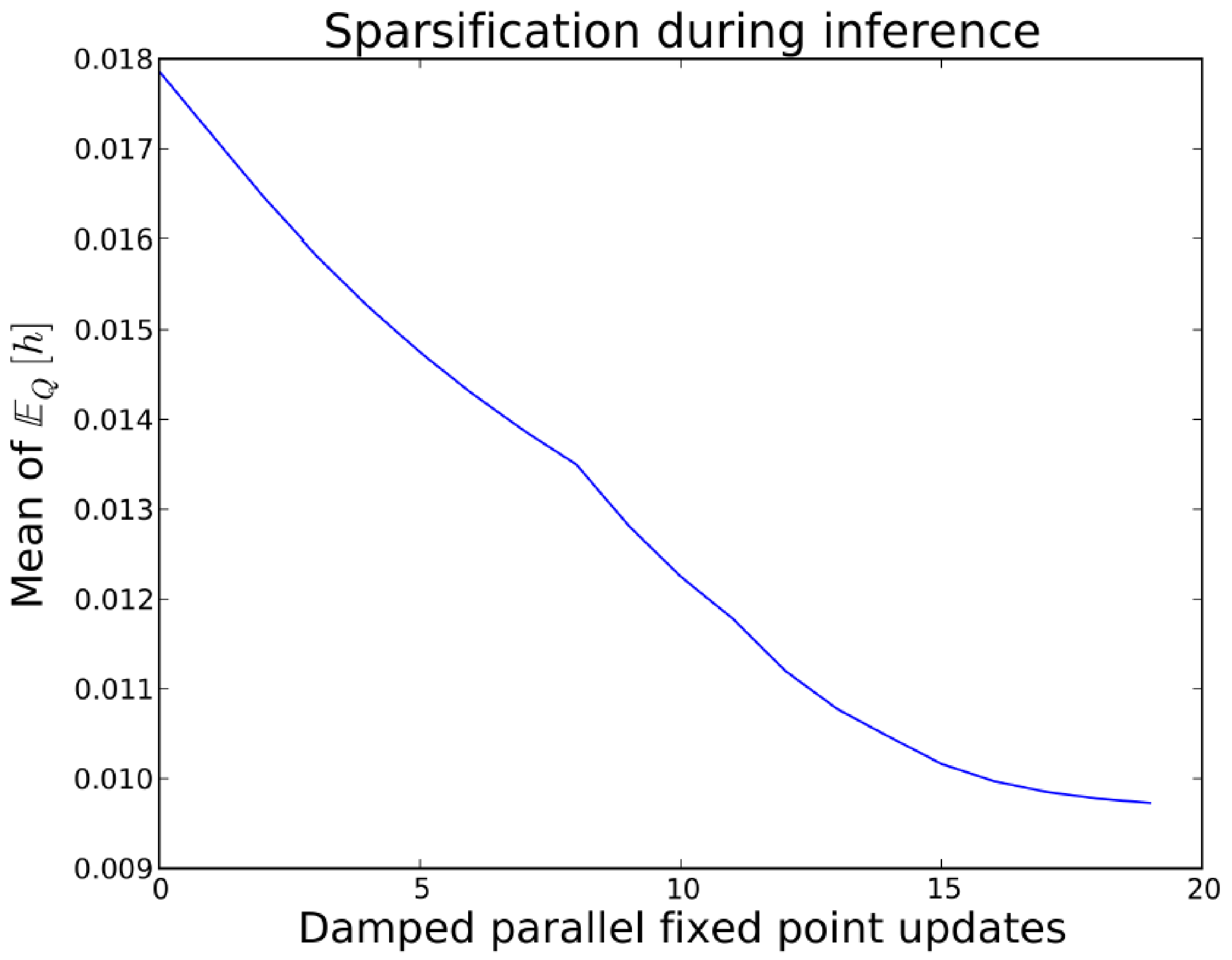


Figure 2. The inference procedure sparsifies the representation due to the explaining-away effect.

Performance Results

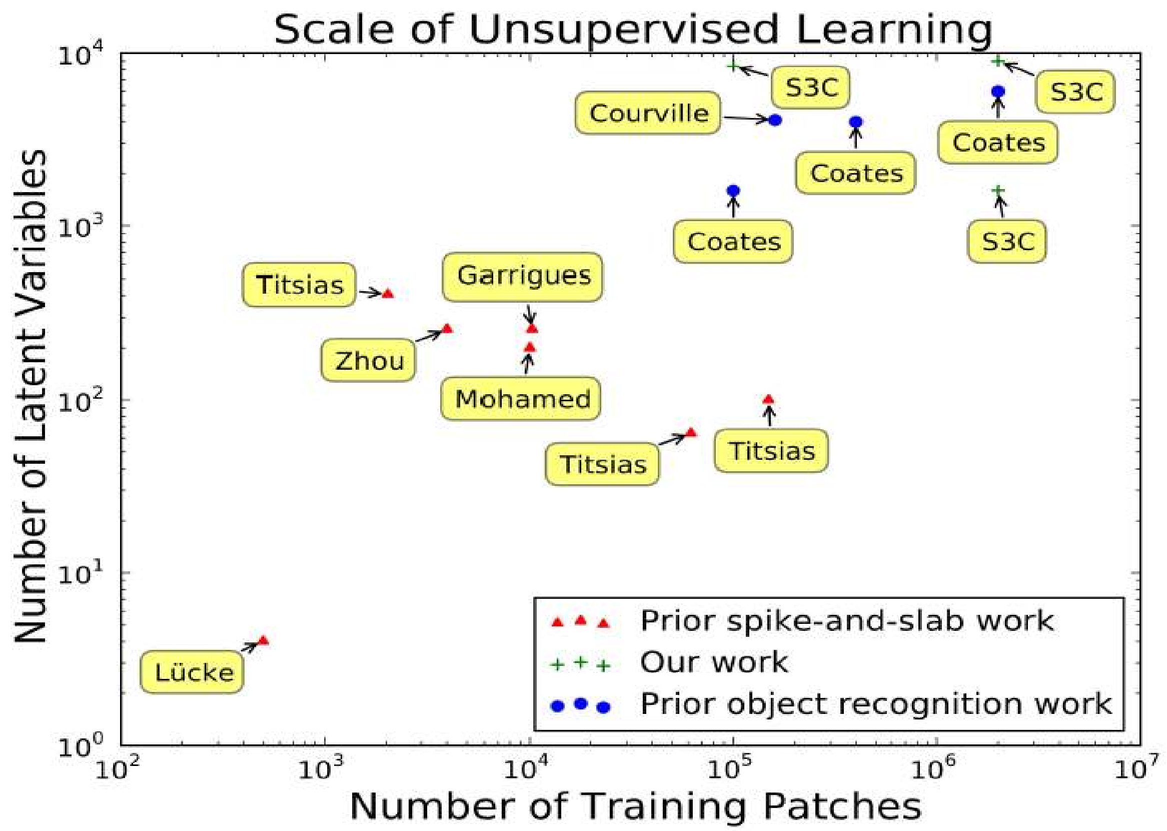


Figure 4. Our inference scheme enables us to extend spike-and-slab modeling from small problems to the scale needed for object recognition. Previous object recognition work is from (Coates and Ng, 2011; Courville et al., 2011a). Previous spike-and-slab work is from (Mohamed et al., 2011; Zhou et al., 2009; Garrigues and Olshausen, 2008; Lücke and Sheikh, 2011; Titsias and Lázaro-Gredilla, 2011).

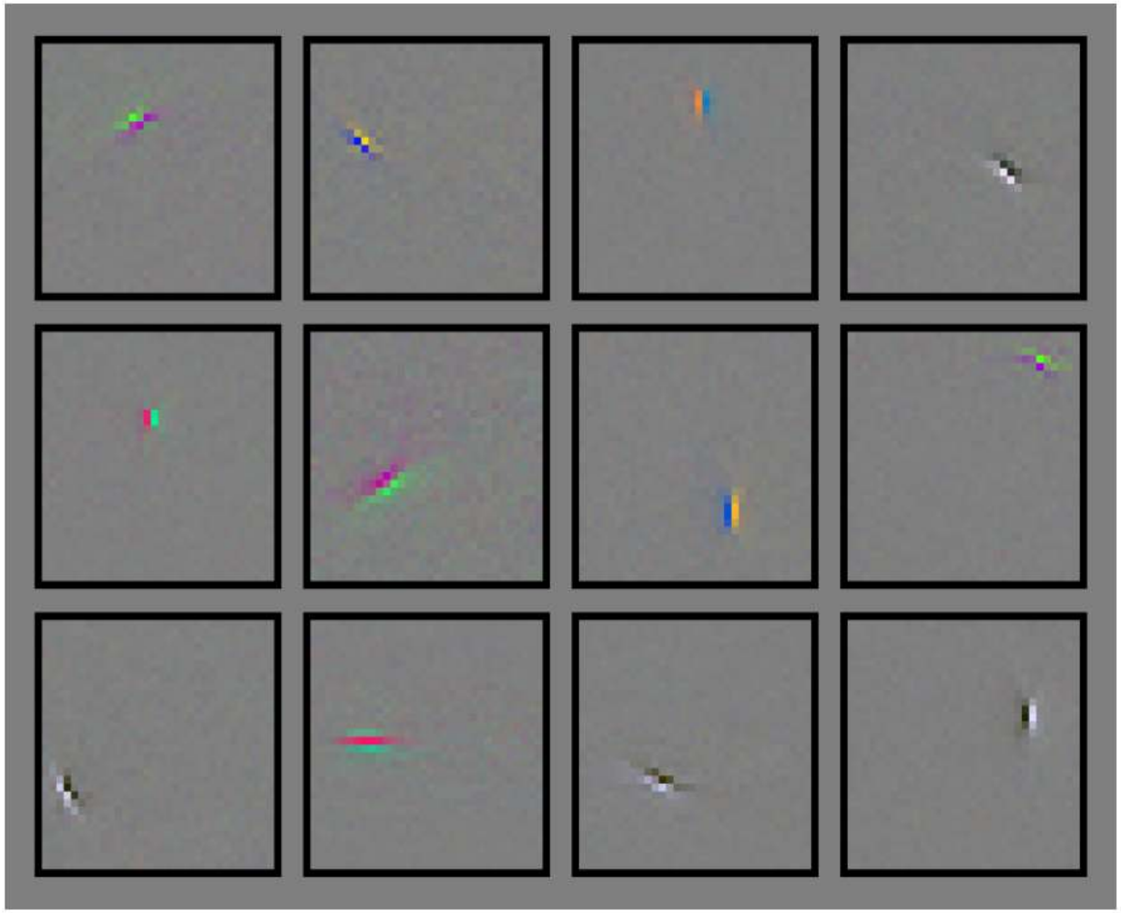


Figure 5. Example filters from a dictionary of over 8,000 learned on full 32x32 images.

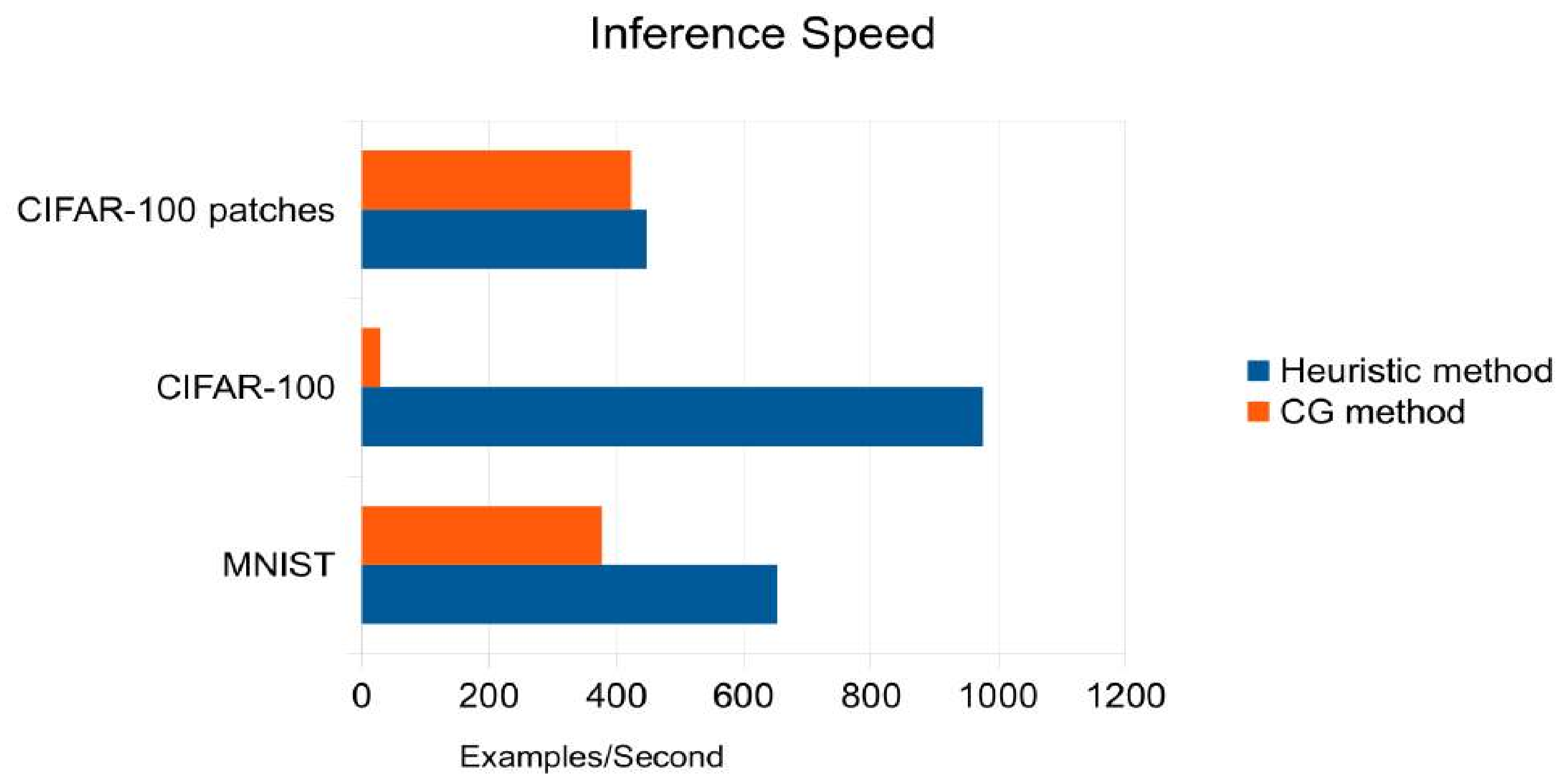


Figure 6. The heuristic method is consistently faster than the conjugate gradient method. The inference speed for each method was computed based on the inference time for the same set of 100 examples from each dataset. In all cases we report the best speed after searching over hyper-parameters controlling the amount of damping / conjugate gradient steps to apply at each update.

Classification Results

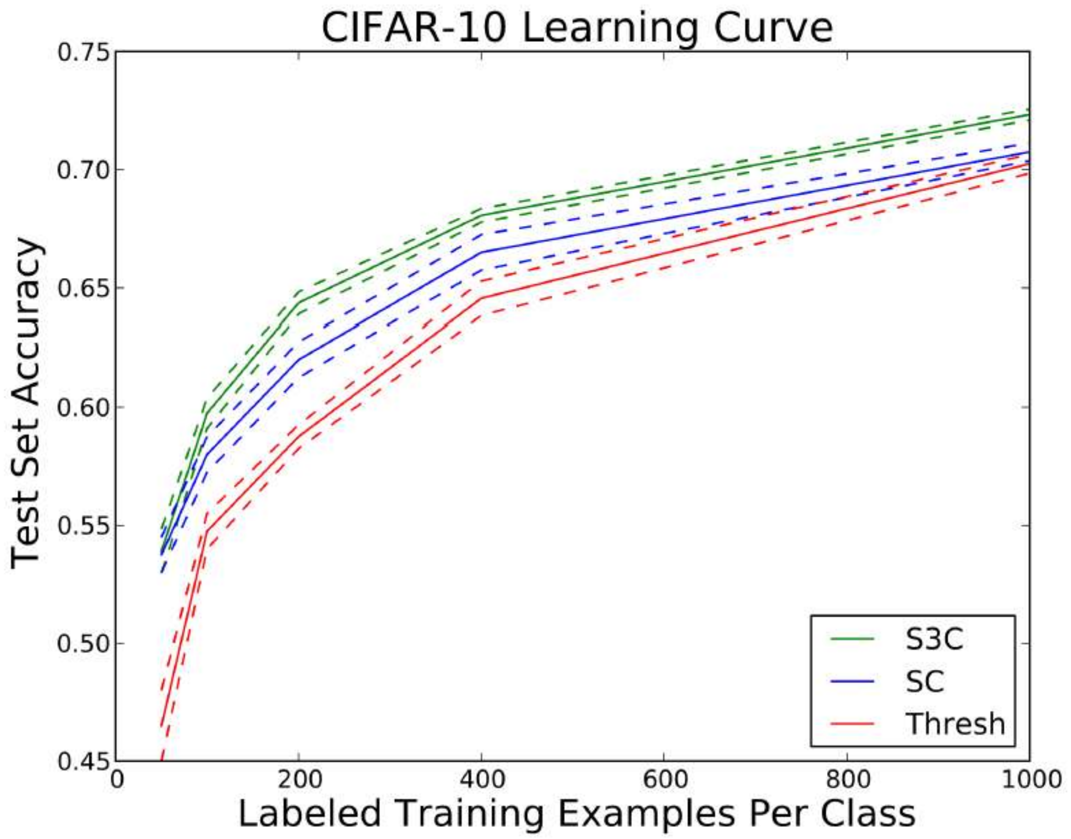
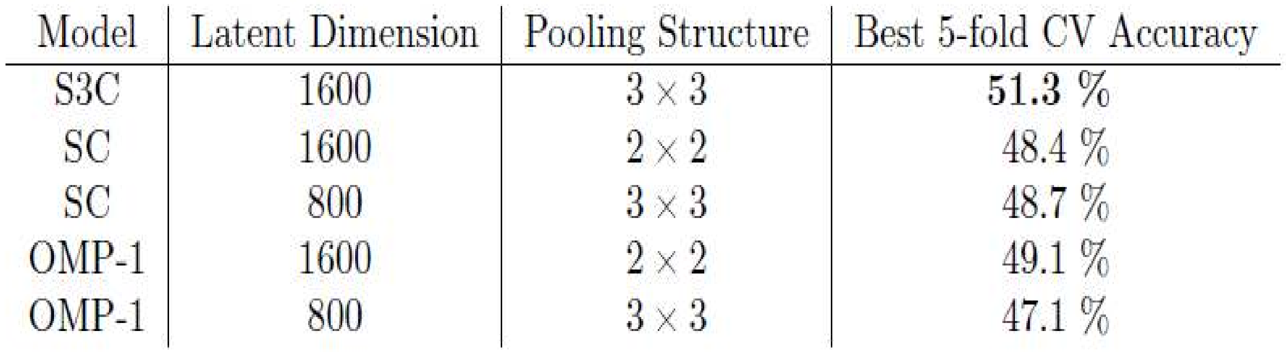


Figure 3. Semi-supervised classification accuracy on subsets of CIFAR-10. Thresholding, the best feature extractor on the full dataset, performs worse than sparse coding when few labels are available. S3C improves upon sparse coding's advantage.

Classification Results

- CIFAR-100



Model	Latent Dimension	Pooling Structure	Best 5-fold CV Accuracy
S3C	1600	3×3	**51.3 %**
SC	1600	2×2	48.4 %
SC	800	3×3	48.7 %
OMP-1	1600	2×2	49.1 %
OMP-1	800	3×3	47.1 %

Table 1. CIFAR-100 validation accuracy. These results demonstrate that using S3C for the detection layer improves upon OMP-1 (the best detector layer for CIFAR-10) and sparse coding.

The CIFAR-10 and CIFAR-100 are labeled subsets of the 80 million tiny images dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The CIFAR-100 has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

Transfer Learning Challenge

NIPS 2011 Workshop on Challenges in Learning Hierarchical Models.

Dataset: 32x32 color images, including 100,000 unlabeled examples, 50,000 labeled examples of 100 object classes not present in the test set, and 120 labeled examples of 10 object classes present in the test set.

Results: A test set accuracy of 48.6 %. This approach disregards the 50,000 labels and treats this transfer learning problem as a semi-supervised learning problem.

Experimented with some transfer learning techniques but the transfer-free approach performed best on leave-one-out cross-validation on the 120 example training set, so we chose to enter the transfer-free technique in the challenge.