Natural Images, Gaussian Mixtures and Dead Leaves Supplementary Material

Daniel Zoran
Interdisciplinary Center for Neural Computation
Hebrew University of Jerusalem
Israel
http://www.cs.huji.ac.il/ daniez

Yair Weiss
School of Computer Science and Engineering
Hebrew University of Jerusalem
Israel
http://www.cs.huji.ac.il/ yweiss

1 Details of model comparison

Here we will give full details of model compared in Section 2 of the main paper. In general, all training data were randomly sampled patches from the Berkeley Segmentation training set images, and all test data was sampled from the Berkeley test set images (that is, testing was always done on unseen patches). We removed the DC component of all patches.

The test set consists of a 1000 patches from which we removed patches with a standard deviation lower than 0.002, totaling in 992 patches. Training set sizes change with each model, see below for details - other than the GMM model (which is extremely parameter rich), the size of the training set has little effect on results (the minimum size we used was 50,000 patches). Log likelihood were averaged over the test set and normalized by the number of pixels minus 1 (because we remove the DC component). All log functions are base 2, so the result is in bits/pixel.

All models and code will be made available if the paper is accepted.

1.1 White Gaussian Noise (Ind. G)

Training

We fit the variance of each pixel σ_i^2 on a training set of 50,000 patches.

Log likelihood

The log likelihood of a patch $\mathbf{x}$ under this model is:

$$\log L(\mathbf{x}) = \sum_i \log \mathcal{N}(x_i; 0, \sigma_i^2)$$

Denoising

Each pixel is denoised independently using a Wiener filter such that:

$$\hat{x}_i = \frac{\sigma_i^2}{\sigma_i^2 + \sigma_n^2} y_i$$

where y_i is the noisy pixel and σ_n^2 the noise variance.

1.2 PCA/Gaussian (PCA G)

Training

We learn the covariance matrix Σ from the training set.

Log likelihood

The log likelihood of a patch $\mathbf{x}$ under this model is:

$$\begin{aligned} \mathbf{z} &= \mathbf{W}\mathbf{x} \\ \log L(\mathbf{x}) &= \sum_i \log \mathcal{N}(z_i; 0, 1) + \log |\mathbf{W}| \end{aligned}$$

where the sum is over the leading $N - 1$ coefficients (discarding the DC component).

Denoising

Denoising with this model is done using a Wiener filter in the whitened domain, coefficient by coefficient, and then transforming back to pixel space.

1.3 PCA/Laplace (PCA L)

Training

We learn the covariance matrix Σ from the training set.

Log likelihood

The log likelihood of a patch $\mathbf{x}$ under this model is:

$$\begin{aligned} \mathbf{z} &= \mathbf{W}\mathbf{x} \\ \log L(\mathbf{x}) &= \sum_i \log \mathcal{L}(z_i; 0, 1) + \log |\mathbf{W}| \end{aligned}$$

where $\mathcal{L}$ is the Laplace distribution and the sum is over the leading $N - 1$ coefficients (discarding the DC component).

Denoising

Denoising with this model is done using a Wiener filter in the whitened domain, coefficient by coefficient, and then transforming back to pixel space.

1.4 ICA

Training

We learn the covariance matrix Σ from the training set and use it to whiten the data. Then, to learn the rotation matrix $\mathbf{V}$ we use gradient ascent on the log likelihood using a factorial distribution with Laplace marginals.

Log likelihood

The log likelihood of a patch $\mathbf{x}$ under this model is:

$$\begin{aligned} \mathbf{z} &= \mathbf{V}\mathbf{W}\mathbf{x} \\ \log L(\mathbf{x}) &= \sum_i \log \mathcal{L}(z_i; 0, 1) + \log |\mathbf{W}| \end{aligned}$$

where $\mathcal{L}$ is the Laplace distribution and the sum is over the leading $N - 1$ coefficients (discarding the DC component). V is orthonormal so it does not change the log determinant part.

Denoising

We tried denoising in two ways (and took the better result). First, we tried to estimate the clean coefficient numerically by maximizing the posterior coefficient with gradient ascent. This produced reasonable results. Second, we used SPAMS [4] - a sparse coding code package which is very fast and efficient, and used it to estimate the clean coefficients and using them to produce the clean patches.

1.5 $2\times$Overcomplete sparse coding ($2\times$OCSC)

Training

We learn the model using the code by Culpepper et al.[1] - this is an efficient code package for learning these types of models. We also tried to learn this model by approximating the posterior with the Laplace approximation [3], results were similar.

Log likelihood

Since the partition function of this model is intractable we use Hamiltonian Annealed Importance Sampling [5] (HAIS) to estimate it and normalize the model. We have verified HAIS to give accurate estimates for the complete case (ICA above) and made sure we use enough particles to converge with the overcomplete case (see below). We use 10,000 particles for the estimation.

Denoising

We tried denoising in two ways (and took the better result). First, we tried to estimate the clean coefficient numerically by maximizing the posterior coefficient with gradient ascent. This produced reasonable results. Second, we used SPAMS [4] - a sparse coding code package which is very fast and efficient, and used it to estimate the clean coefficients and using them to produce the clean patches.

1.6 GSM

Training

We learn the covariance matrix Σ from the training set patches, and the scaling variables z_k and mixing coefficient π_k using plain EM. We learn a mixture of 16 components.

Log likelihood

The likelihood of a given patch $\mathbf{x}$ under this model is:

$$\log L(\mathbf{x}) = \log \left(\sum_k \pi_k \mathcal{N}(\mathbf{x}; \mathbf{0}, z_k^2 \Sigma) \right)$$

Denoising

We used the approximate MAP procedure we use for the GMM model, see below for details.

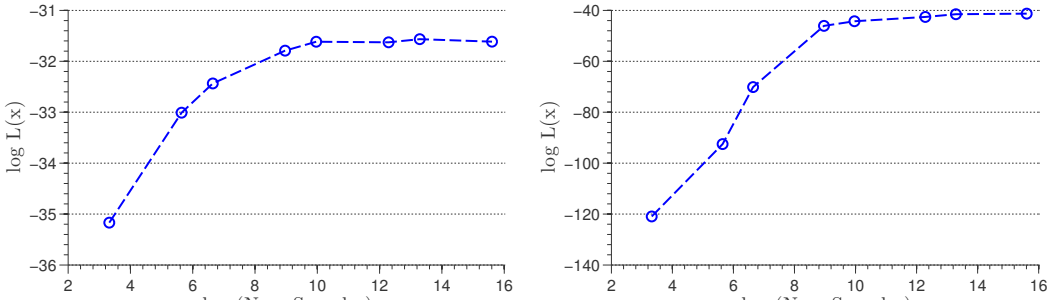
1.7 MoGSM

Training

We learn the MoGSM model using EM - this is similar to learning any mixture model, with the mixture components being GSMs as above. We learn a mixture of 5 GSMs, each having 8 scale components (as in [6]).

Log likelihood

The likelihood of a given patch is:



(a) 8×8 (b) 16×16

Figure 1: Estimated likelihood as a function of number of particles for the Karklin and Lewicki model.

$\log L(\mathbf{x}) = \log \left(\sum_k \pi_k \mathcal{GSM}(\mathbf{x}; \mathbf{0}, z_k^2 \Sigma) \right)$

where $\mathcal{GSM}$ is the likelihood of a GSM model as above.

Denoising

We used the approximate MAP procedure we use for the GMM model, see below for details.

1.8 Karklin and Lewicki (KL)

Training

We learn the model using the code supplied by the authors of the model [2] on our training set. We kept the same proportion of latent variables and dictionary elements to patch sizes as the original paper. For 8×8 model we used 25 latent variables and 160 dictionary elements. For the 16×16 model we used 96 latent variables and 640 dictionary elements.

Log likelihood

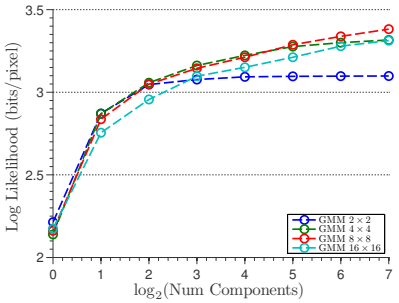
Since the partition function of this model is intractable we use Hamiltonian Annealed Importance Sampling [5] (HAIS) to estimate it and normalize the model. Figure 1 depicts the convergence of the estimation on 5 samples as a function of number of particles - note that it converges quite rapidly. In both experiments we used 10,000 particles which seem to give an accurate answer. See the original paper for details on this.

Denoising

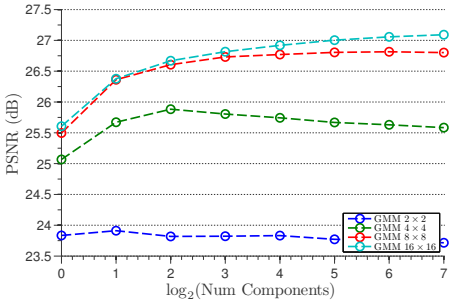
Since the model does not include a noise term calculating the MAP in the noisy case is very hard. What we do is a similar procedure to the GMM approximate MAP below. For each noisy patch $\mathbf{y}$ we estimate the latent variables by using the original MAP code supplied by the authors of the model. Then, using these latent variables we calculate the corresponding covariance matrix Σ_1 and use it to denoise the patch by Wiener filtering it:

$$\hat{\mathbf{x}}_1 = \left(\Sigma_1 + \Sigma_n \right)^{-1} \Sigma_1 \mathbf{y}$$

where Σ_n is the noise covariance. Using the estimated patch $\hat{\mathbf{x}}_1$we repeat the process to obtain a second covariance matrix Σ_2. We then us Σ_2 to perform another Wiener filtering on the original noisy patch to obtain the final clean estimate.



(a) Log Likelihood



(b) Log Likelihood

Figure 2: Likelihood and denoising performance as a function of number of components, similar to Figure **??** in the paper.

1.9 GMM

Training

We learn a 200 components GMM using a mini-batch version of EM. The GMM has full covariance matrices and zero means. At each iteration we sample a 20,000 or 50,000 patches (for the 8×8 and 16×16 models respectively) patches from the complete training set and calculate an E and an M step over it - then we update the current mixture model by taking a small step η towards the newly learned parameters. We regularize the covariance matrices by adding $\epsilon \mathbf{I}$ to the covariance of each component ($\epsilon = 10^{-5}$). We ran the learning for 4000 iterations, decreasing η slightly at each iteration. The code we use is based upon code available online[1].

Log likelihood

The likelihood of a given patch $\mathbf{x}$ under this model is:

$$\log L(\mathbf{x}) = \log \left(\sum_k \pi_k \mathcal{N}(\mathbf{x}; \mathbf{0}, \Sigma_k) \right)$$

Denoising

We used an approximate MAP procedure. We first calculate the assignment probability of each of the K mixture components. We then choose the most likely one k_{max} and use its covariance matrix to perform Wiener filter such that:

$$\hat{\mathbf{x}} = \left(\Sigma_{k_{max}} + \Sigma_n \right)^{-1} \Sigma_{k_{max}} \mathbf{y}$$

2 Number of components experiment

Figure 2 depicts the results for 2×2, 4×4 and 8×8 patches - note that the behavior is similar across patch sizes.

3 More components of the GMM

Figure 3 depicts some more components from the GMM, along with samples from each components. Components are ordered by order of mixing weight, in decreasing order (more likely ones come first). Note the varied different structures each component produces.

[1] http://www.mathworks.com/matlabcentral/fileexchange/26184-em-algorithm-for-gaussian-mixture-model

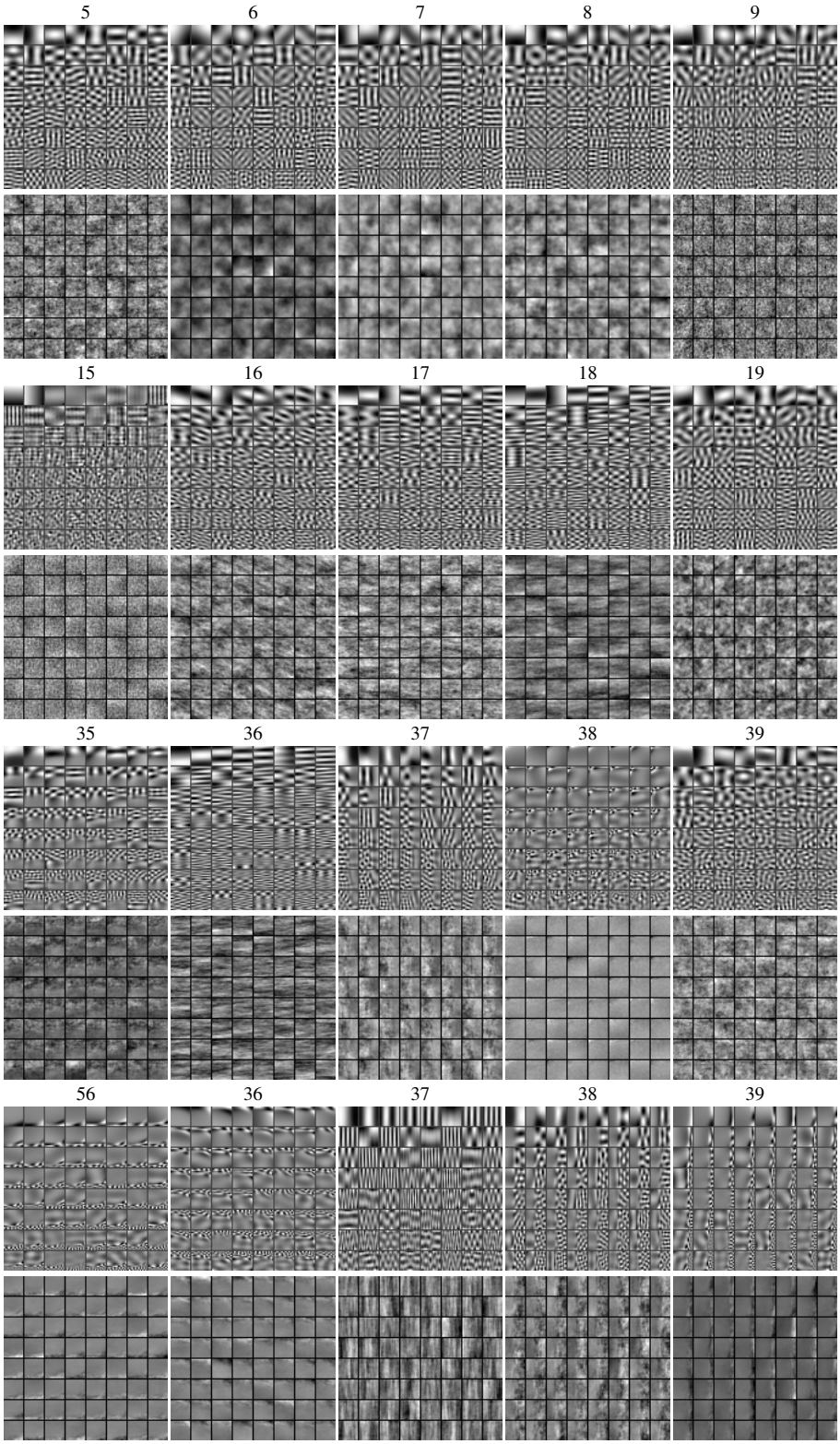


Figure 3: Eigenvectors and samples from different components of the 16×16 GMM model.

References

[1] B.J. Culpepper, J. Sohl-Dickstein, and B.A. Olshausen. Building a better probabilistic model of images by factorization. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011.

[2] Y. Karklin and M.S. Lewicki. Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*, November 2008.

[3] M.S. Lewicki and B.A. Olshausen. Probabilistic framework for the adaptation and comparison of image codes. *JOSA A*, 16(7):1587–1601, 1999.

[4] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[5] J Sohl-Dickstein and BJ Culpepper. Hamiltonian annealed importance sampling for partition function estimation. 2011.

[6] L. Theis, S. Gerwinn, F. Sinz, and M. Bethge. In all likelihood, deep belief is not enough. *The Journal of Machine Learning Research*, 999888:3071–3096, 2011.