Journal of Machine Learning Research 15 (2014) 2061-2086 Submitted 8/13; Revised 2/14; Published 6/14

The Student-t Mixture as a Natural Image Patch Prior with Application to Image Compression

Aäron van den Oord AARON.VANDENOORD@UGENT.BE
Benjamin Schrauwen BENJAMIN.SCHRAUWEN@UGENT.BE
Department of Electronics and Information Systems
Ghent University
Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium

Editor: Ruslan Salakhutdinov

Abstract

Recent results have shown that Gaussian mixture models (GMMs) are remarkably good at density modeling of natural image patches, especially given their simplicity. In terms of log likelihood on real-valued data they are comparable with the best performing techniques published, easily outperforming more advanced ones, such as deep belief networks. They can be applied to various image processing tasks, such as image denoising, deblurring and inpainting, where they improve on other generic prior methods, such as sparse coding and field of experts. Based on this we propose the use of another, even richer mixture model based image prior: the Student-t mixture model (STM). We demonstrate that it convincingly surpasses GMMs in terms of log likelihood, achieving performance competitive with the state of the art in image patch modeling. We apply both the GMM and STM to the task of lossy and lossless image compression, and propose efficient coding schemes that can easily be extended to other unsupervised machine learning models. Finally, we show that the suggested techniques outperform JPEG, with results comparable to or better than JPEG 2000.

Keywords: image compression, mixture models, GMM, density modeling, unsupervised learning

1. Introduction

Recently, there has been a growing interest in generative models for unsupervised learning. Especially latent variable models such as sparse coding, energy-based learning and deep learning techniques have received a lot of attention (Wright et al., 2010; Bengio, 2009). The research in this domain was for some time largely stimulated by the success of the models for discriminative feature extraction and unsupervised pre-training (Erhan et al., 2010). Although some of these techniques were advertised as better generative models, no experimental results could support these claims (Theis et al., 2011). Furthermore recent work (Theis et al., 2011; Tang et al., 2013) showed that many of these models, such as restricted Boltzmann machines and deep belief networks are outperformed by more basic models such as the Gaussian Mixture model (GMM) in terms of log likelihood on real-valued data.

Although arguably not as useful for the extraction of discriminative features, for the use of unsupervised pre-training, Gaussian mixture models have been shown to be very successful in various image processing tasks, such as denoising, deblurring and inpainting (Zoran and Weiss, 2011; Yu et al., 2012). Good density models are essential for these tasks, and the log likelihood measure of these models has shown to be a good proxy for their performance. Apart from being simple and efficient, GMMs are easily interpretable methods which allow us to learn more about the nature of images (Zoran and Weiss, 2012).

In this paper we suggest the use of a similar, simple model for modeling natural image patches: the Student-t mixture model (STM). The STM uses multivariate Student-t distributed components instead of normally distributed components. We will show that a Student-t distribution, although having only one additional variable (the number of degrees of freedom), is able to model stronger dependencies than solely the linear covariance of the normal distribution, resulting in a large increase in log likelihood. Although a GMM is a universal approximator for continuous densities (Titterington et al., 1985), we will see that the gap in performance between the STM and GMM remains substantial, as the number of components increases.

Apart from comparing these methods with other published techniques for natural image modeling in terms of log likelihood, we will also apply them to image compression by proposing efficient coding schemes based on these models. Like other traditional image processing applications, it is a challenging task to improve upon the well-established existing techniques. Especially in data compression, which is one of the older, more advanced branches of computer science, research has been going on for more than 30 years. Most modern image compression techniques are therefore largely the result of designing data-transformation techniques, such as as the discrete cosine transform (DCT) and the discrete wavelet transform (DWT), and combining them with advanced engineered entropy coding schemes (Wallace, 1991; Skodras et al., 2001).

We will demonstrate that simple unsupervised machine learning techniques such as the GMM and STM are able to perform quite well on image compression, compared with conventional techniques such as JPEG and JPEG 2000. Because we want to measure the density-modeling capabilities of these models, the amount of domain-specific knowledge induced in the proposed coding schemes is kept to a minimum. This also makes it relevant from a machine learning perspective as we can more easily apply the same ideas to other types of data such as audio, video, medical data, or more specific kinds of images, such as satellite, 3D and medical images.

In Section 2 we review some work on compression, in which related techniques were used. In Section 3 we give the necessary background for this paper on the GMM and STM and the expectation-maximization (EM) steps for training them. We will also elaborate on their differences and the more theoretical aspects of their ability to model the distribution of natural image patches. In Section 4 we present the steps for encoding/decoding images with the use of these mixture models for both lossy and lossless compression. The results and their discussion follow in Section 5. We conclude in Section 6.

2. Related Work

In this section we will review related work on image compression and density modeling.

2.1 Image Compression

The coding schemes (see section 4) we use to compare the GMM and STM, can be related with other published techniques in image compression, in the way they are designed. Although little research has been done on the subject we briefly review work based on vector quantization, sparse coding and subspace clustering. The lossy coding scheme we describe in this paper is based on a preliminary version that appeared in our previous work (van den Oord et al., 2013).

In vector quantization (VQ) literature, GMMs have been proposed for the modeling of low-dimensional speech signal parameters (Hedelin and Skoglund, 2000). In this setting, the GMMs' probability density function is suggested to be used to fit a large codebook of VQ centroids on (e.g., with a technique similar to k-means), instead of on the original data set. They were introduced to help against overfitting, which is a common problem with the design of vector quantizers when the training set is relatively small compared to the size of the codebook. The same idea has also been suggested for image compression (Aiyer et al., 2005). In contrast to these approaches we will apply a (learned) data transformation in combination with simple *scalar* uniform quantization, which reduces the complexity considerably given the relatively high dimensionality of image patches. This idea called transform coding (Goyal, 2001) is widely applied in most common image compression schemes, which use designed data-transforms such as the DCT and DWT.

By some authors (Hong et al., 2005) image compression has been suggested based on a subspace clustering model. The main contribution was a piecewise linear transformation for compression, which was also extended to a multiscale method. This is by some means similar to our lossy compression scheme as we also apply a piecewise linear transform, but based on the GMM/STM instead of a subspace clustering technique. They did not suggest quantization or entropy coding steps, and therefore only evaluated their approach in terms of energy compaction instead of rate-distortion.

Image compression based on sparse coding has been proposed (Horev et al., 2012) for images in general (Bryt and Elad, 2008; Zepeda et al., 2011) and for a specific class of facial images. Aside from being another unsupervised learning technique, sparse coding has been related with GMM in another way: Some authors (Yu et al., 2012; Zoran and Weiss, 2011) have suggested the interpretation of a GMM as a structured sparse coding model. This idea is based on the observation that data can often be represented well by one of the N Gaussian mixture components, thus when combining all the eigenvectors of their covariance matrices as an overcomplete dictionary, the sparsity is $\frac{1}{N}$. The main results in Horev et al. (2012) show that sparse coding outperforms JPEG, but it does not reach JPEG 2000 performance for a general class of images.

2.2 Models of Image Patches

Sparse coding approaches (Olshausen and Field, 1997) have also been successfully applied as an image prior on various image reconstruction tasks, such as denoising and demosaicing (Elad and Aharon, 2006; Mairal et al., 2009). These models have recently been shown to be outperformed by the GMM in both image denoising (Zoran and Weiss, 2011) as density modeling (Zoran and Weiss, 2012).

The Fields of Experts (FoE) framework is another approach for learning priors that can be used for image processing applications (Roth and Black, 2005; Weiss and Freeman, 2007). In a FoE, the linear filters of a Markov random field (MRF) are trained to maximize the log-likelihood of whole images in the training set. This optimization is done approximately with contrastive divergence, as computing the log likelihood itself is intractable. The potential functions that are used in the MRF are represented by a product of experts (PoE) (Hinton, 2002). The FoE is commonly used for image restoration tasks such as denoising and inpainting, but was also recently outperformed by GMMs with the expected patch log likelihood framework (EPLL) (Zoran and Weiss, 2012).

Recently similar models to the GMM have been proposed for image modeling, such as the Deep mixture of Factor analyzers (Deep MFA) (Tang et al., 2012). This technique is a deep generalization of the Mixture of Factor Analyzers model, which is similar to the GMM. The deep MFA has a tree structure in which every node is a factor analyzer, which inherits the low-dimensional latent factors from its parent.

Another model related to the GMM and STM is the Mixture of Gaussian scale mixtures (MoGSM) (Theis et al., 2011, 2012). Instead of a Gaussian, every mixture component is a Gaussian scale mixture distribution. The MoGSM has been used for learning multi-scale image representations, by modeling each level conditioned on the higher levels.

RNADE, a new deep density estimation technique for real valued data has a very different structure (Uria et al., 2013b,a). RNADE is an extension of the NADE technique for real-valued data, where the likelihood function is factored into a product of conditional likelihood functions. Each conditional distribution is fitted with a neural mixture density network, where one variable is estimated, given the other ones. Recently a new training method has allowed a factorial number of RNADE's to be trained at once within one model. It is currently one of the few deep learning methods with good density estimation results on real-valued data and is the current state of the art on image patch modeling.

3. Mixture Models as Image Patch Priors

Mixture models are among the most widely accepted methods for clustering and probability density estimation. Especially GMMs are well known and have widespread applications in different domains. However depending on the data used, other mixture models might be more suitable.

In this work we will denote the mixture component distribution as f_k and the mixture distribution as

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x),$$

where $\pi_k, k = 1 \dots K$ are the mixing weights. The two component distributions we study here are the multivariate normal distribution:

$$f_k(x) = \mathcal{N}(x|\mu_k, \Sigma_k) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)},$$

and the multivariate Student-t distribution, see Equation 1. In these equations, p is the dimensionality of x. We will train the GMM with the EM-algorithm: an iterative algorithm

for finding the maximum likelihood estimate of the parameters. For completeness we summarize the expectation and maximization steps for training a GMM with EM.
E-step:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}\left(x_n | \mu_k, \Sigma_k\right)}{\sum\limits_{j=1}^{K} \pi_j \mathcal{N}\left(x_n | \mu_j, \Sigma_j\right)}.$$

M-step:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}, \qquad \mu_k = \frac{\sum\limits_{n=1}^{N} \gamma_{nk} x_n}{\sum\limits_{n=1}^{N} \gamma_{nk}},$$

$$\Sigma_k = \frac{\sum\limits_{n=1}^{N} \gamma_{nk} \left(x_n - \mu_k\right) \left(x_n - \mu_k\right)^T}{\sum\limits_{n=1}^{N} \gamma_{nk}}.$$

One of the important reasons GMMs excel at modeling image patches is that the distribution of image patches has a multimodal landscape. A unimodal distribution such as the multivariate normal distribution is not able to capture this. When using a mixture however, each component can represent a different aspect or texture of the whole distribution. We can observe this by looking at the individual mixture components of a trained GMM model, see Figure 1.

Next to modeling different textures, the GMM also captures differences in contrast. It has been shown (Zoran and Weiss, 2012) that multiple components in the GMM describe a similar structure in the image, but each with a different level of contrast. The STM, however, can model different ratios of contrast within a single mixture component.

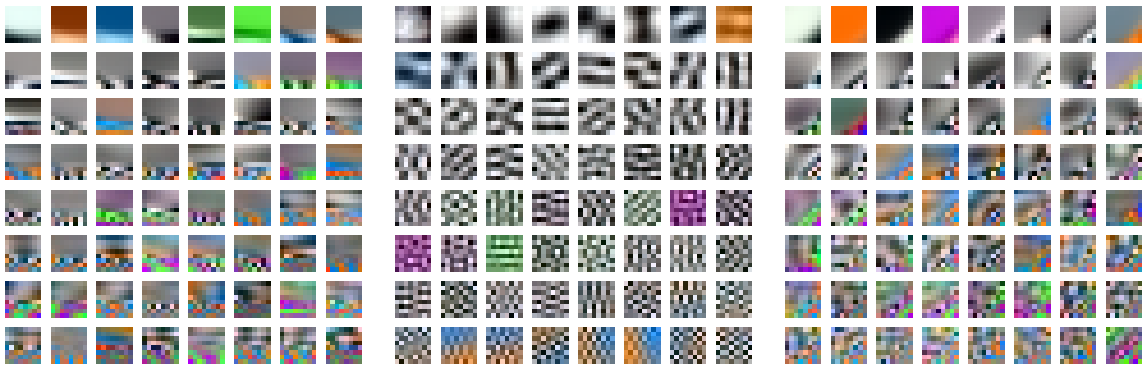
A multivariate Student-t distribution has the following density function (Kotz and Nadarajah, 2004):

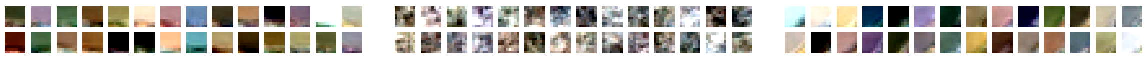
$$\mathcal{T}\left(x | \nu, \mu, \Sigma\right) = \frac{\Gamma\left(\frac{\nu+p}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right) \nu^{\frac{p}{2}} \pi^{\frac{p}{2}} \left|\Sigma\right|^{1/2}} \left[1 + \frac{1}{\nu} \left(x - \mu\right)^T \Sigma^{-1} \left(x - \mu\right)\right]^{-\frac{\nu+p}{2}}. \tag{1}$$

ν is an additional parameter which represents the number of degrees of freedom. Note that for $\nu \to \infty$ the Student-t distribution converges to the normal distribution. It is interesting to see how this distribution is constructed:
If Y is a multivariate normal random vector with mean 0 and covariance Σ, and if νT is a chi-squared random variable with degrees of freedom ν, independent of Y, and

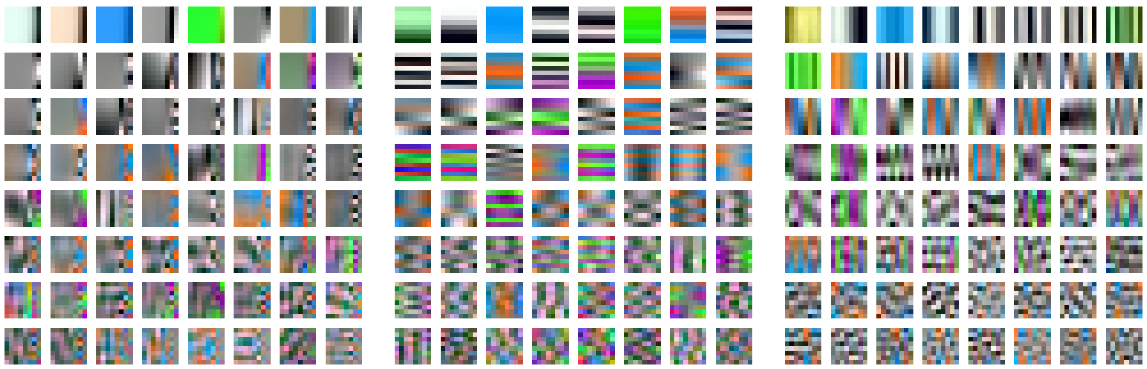
$$X = \frac{Y}{\sqrt{T}} + \mu,$$

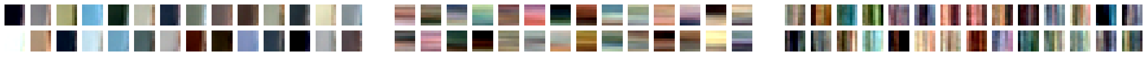
then X has a multivariate Student-t distribution with degrees of freedom ν, mean μ, and covariance matrix Σ. This also means $X|T = \tau$ is normally distributed with mean μ and covariance $\frac{\Sigma}{\tau}$. In the setting of modeling image patches, T can be interpreted to model the


The first 64 eigenvectors of the component's covariance matrix.


Patches generated by sampling from the component distribution.

Examples from the train set that are best represented by this component.


The first 64 eigenvectors of the component's covariance matrix.


Patches generated by sampling from the component distribution.

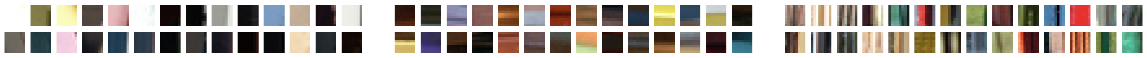
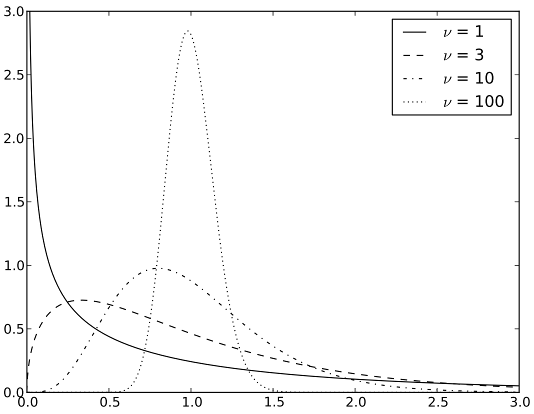

Examples from the train set that are best represented by this component.

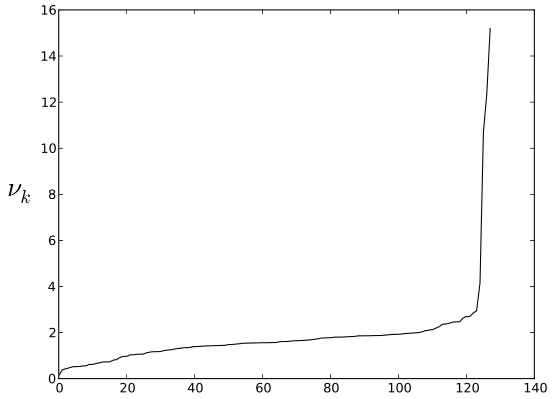
Figure 1: Six mixture components of the GMM are visualized here (the STM gives similar texture patterns). We first show the eigenvectors of the covariance matrix of each component, which show the structure of the image patches that the mixture component learns. These eigenvectors are sorted by their respective eigenvalues from large to small (left to right and top to bottom). Only the first 64 of 192 are shown. Next we show some samples that are generated by each component, and some examples from the train set that are best represented with this component (clustered with the GMM). Note that every component has specialized in a different aspect or texture, and that the samples generated by the component distributions are very similar to the real image patches. This figure is best viewed in color on the electronic version.

variety of contrast, for a given texture. The distribution of T is visualized in Figure 2(a), for different values of ν. If ν is small for a given component (texture), this means that the texture appears in natural images in a wide range of contrast. For $\nu \to \infty$ we get a Gaussian distribution and its contrast is more constrained. To obtain the same capacity with a GMM, one would need multiple components having scaled versions of the same covariance matrix.

In Figure 2(b) the value of ν is visualized for different components of a trained STM. This value differs substantially for each component, ranging from almost zero to 15. This means some component-distributions are very long-tailed (with small ν) and some are more normally distributed (higher ν). This means that some texture patterns appear in a wider range of contrast than others. However, in our experiments we saw that the STM does not learn significantly different structures compared to the GMM. The texture patterns learned by the STM were also very similar to those shown in Figure 1. This means the STM is better at generalizing to image patches with different levels of contrast, but might not be better at generalizing to different unseen texture patterns.



(a) The distribution of T in Equation 2, for different values of ν. As ν increases the distribution becomes more peaked and converges to a Dirac delta at 1.



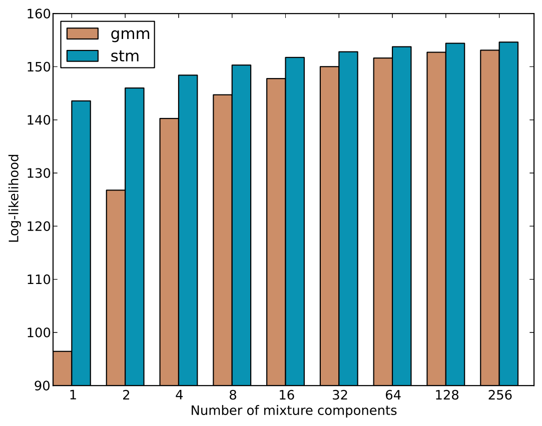
(b) The value of ν_k for each component k of a trained STM with 128 components, sorted from low to high.

Figure 2: The distribution of T for different ν and the value of ν for different mixture components.

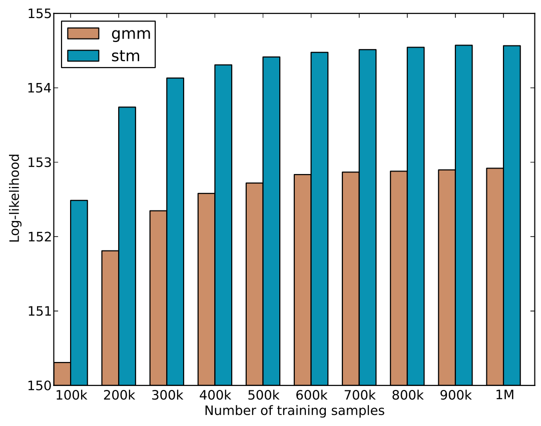
Given the fact that a GMM is universal approximator for continuous densities, the question that remains is if a STM still has the advantage over the GMM when the number of components increases. To this end we have trained a GMM and STM on a set of image patches for different numbers of mixture components and computed their log likelihood scores on a validation set, see Figure 3(a). Notice that the performance of a single Student-t is much better than that of a single Gaussian, and close to that of a GMM with K=4. This is in agreement with previously reported findings (Zoran and Weiss, 2012), which suggest that a GMM with a small number of components mainly learns contrast. Next we see that as N increases the gap in performance between the STM and GMM remains substantial. The most plausible explanation for this behavior is that the GMM needs more mixture components than the STM to have the same contrast modeling capabilities. However, with more mixture components the risk of overfitting also increases. If one would tie the parameters of some of these components together, so that they have scaled versions of the same covariance matrix, the risk of overfitting would decrease. This is exploited in the mixture of Gaussian scale mixtures (MoGSM) (Theis et al., 2012).

The idea of explicitly sharing covariance parameters between mixture components has also been applied to mixtures of factor analyzers, with the deep MFA model (Tang et al., 2012). They proposed a hierarchical structure in which the mixture components partially inherit the covariance structure of their parent in the hierarchy.

The Student-t has previously been used for modeling image patches in the PoE framework (Welling et al., 2002), where each expert models a differently linearly filtered version of the input with a univariate Student-t distribution.



(a)



(b)

Figure 3: The average patch log likelihood for the Gaussian mixture model (GMM) and Student-t mixture model (STM) in function of its number of mixture components (a) and number of training samples (b). The models were trained on 8x8 normalized gray scale image patches, extracted from the Berkeley data set (see Section 5.1.1).

We also train the STM with the EM algorithm (Peel and McLachlan, 2000; Dempster et al., 1977):
E-step:

$$\gamma_{nk} = \frac{\pi_k \mathcal{T}\left(x_n|\nu_k, \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{T}\left(x_n|\nu_j, \mu_j, \Sigma_j\right)}, \quad w_{nk} = \frac{\nu_k + p}{\nu_k + (x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)}.$$

M-step:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}, \qquad \mu_k = \frac{\sum_{n=1}^{N} \gamma_{nk} w_{nk} x_n}{\sum_{n=1}^{N} \gamma_{nk} w_{nk}}.$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma_{nk} w_{nk} \left(x_n - \mu_k\right)\left(x_n - \mu_k\right)^T}{\sum_{n=1}^{N} \gamma_{nk}}.$$

For the degrees of freedom, there is no closed form update rule. Instead ν_k gets updated as the solution of:

$$\begin{aligned} & -\psi\left(\frac{\nu_k}{2}\right) + \log\left(\frac{\nu_k}{2}\right) + 1 + \frac{1}{\alpha_k}\sum_{n=1}^{N}\gamma_{nk}\left(\log\left(w_{nk}\right) - w_{nk}\right) \\ & + \psi\left(\frac{\tilde{\nu_k} + p}{2}\right) - \log\left(\frac{\tilde{\nu_k} + p}{2}\right) = 0, \end{aligned}$$

where $\tilde{\nu_k}$ is the value of the current ν_k, $\alpha_k = \sum_{n=1}^{N} \gamma_{nk}$ and ψ is the digamma function. This scalar non-linear equation can be solved quickly with a root finding algorithm, such as Brent's method (Brent, 1973).

Note that the expectation and maximization steps are quite similar to those of the GMM. In our experiments, it did not take substantially longer to train a STM than a GMM. Typically 100 iterations were enough to train the STM or GMM, even though the log likelihood does keep improving a little bit after that (even after 500 iterations). For a big mixture model of 256 components, trained on 500.000 samples of 8x8 gray scale patches, this took about 20 hours on a standard desktop computer with four cores. For the STM, it took 21 hours. On this scale, the CPU time is linear in both the number of training samples and the number of components. Training on image patches proved to be quite stable: no components needed to be reinitialized during training.

The code for training a Student-t mixture is included in the supplementary material of this paper.

4. Compression with Mixture Models

Both the lossy and lossless algorithms we propose are patch/block based. This means they will encode each patch of an image separately. During training we randomly sample a large set of image patches from the training images. These are used to fit the GMM and STM models. Once training is finished, these density models can be used to encode the test images. Each test image is viewed as a grid of non-overlapping patches. The encoder loops over all patches, which are extracted, flattened and encoded one by one.

To speed up the algorithms, each patch will be encoded using the distribution and parameters of only one of the mixture components. We choose the mixture component which represents the given patch with the highest likelihood:

$$\beta = \arg\max_k f_k\left(x_n\right).$$

This will only slightly reduce the performance, because the "overlap" between the individual mixture components is relatively small. We can easily validate this with a simple intermediate experiment. In Table 1 we have computed the log likelihood for a trained GMM and STM

	GMM	STM
Log likelihood	152.86	154.51
Highest mixture component log likelihood	152.66	154.15

Table 1: Average patch log likelihood compared with the average highest component patch log likelihood: How well can a sample be represented by using a single mixture component? (See text)

on a validation set. We have also computed the average log likelihood when only one of the mixture components is used for each example: $\frac{1}{N}\sum_{n=1}^{N}\log\left(\max_k\left(\pi_k f_k\left(x_n\right)\right)\right)$. Note that this is strictly lower than the actual average log likelihood: $\frac{1}{N}\sum_{n=1}^{N}\log\left(\sum_{k=1}^{K}\pi_k f_k\left(x_n\right)\right)$. But as can be seen from Table 1, the difference is small.

4.1 Arithmetic Coding

Most commonplace image compression schemes follow three main steps: transformation, quantization and entropy coding (Goyal, 2001). Transformation decorrelates the data, quantization maps the values of the decorrelated continuous variables onto discrete values from a relatively small set of symbols (such as integers) and entropy coding encodes these discrete quantized values into a bit sequence. In this paper, transformation and quantization will only be used for lossy compression and not for lossless compression. However, in both cases we employ arithmetic coding (AC) for the entropy coding step.

Entropy coding is a family of algorithms that take as input a sequence of discrete values, and give as output the encoded binary sequence. Based on the statistical properties of the input, the goal is to minimize the expected length of the bit sequence (e.g., by assigning more bits to a rare symbol and less bits to a common symbol). The theoretical limit of the encoding scheme is bounded by the entropy of the input signal, which explains the name entropy coding. Arithmetic coding is a form of entropy coding, which requires a list of probabilities $\alpha_i, i = 1 \dots N$ that describe the discrete distribution $P\left(s_j\right) = \alpha_j$ of a symbol s_j occurring in an input sequence. Based on these probabilities, the algorithm will on average spend fewer bits on common symbols, than on rare ones. However, with AC it is also possible to use different probabilities for each time step t in the sequence: $P\left(s_j^{(t)}\right) = \alpha_{jt}$, and even adapt them during the encoding/decoding based on the values of the previously encoded symbols. This is also called adaptive arithmetic coding.

4.2 Lossless Compression

In lossless compression, the image should be preserved perfectly so that after decompression the output image is identical to the input image. Because we have a probabilistic model for an image patch, the most natural way to approach this task is to use lossless predictive coding (Pearlman and Said, 2011). The idea is to predict the value (integer) of each sample within an image patch, using the values of its neighboring samples that are already encoded, based on the correlations between them. In this case, the prediction will actually consist of a discrete probability distribution over the possible values of the current sample, which can directly be used to perform arithmetic coding.

To carry out arithmetic coding on a patch x_i, one needs to compute a list of probabilities (probability table) for each of its elements $x_{i,j}$: $P(x_{i,j} = l)$ for $l = 0 \dots L$. More specifically, because arithmetic coding can adapt the probability tables to the information of the previous symbols $x_{i,1} \dots x_{i,j-1}$ it is possible to encode every symbol conditionally with respect to the ones already encoded: $P(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$. As the image patches are modeled by continuous probability densities, this can computed as follows:

$$P(x_{i,j} = l | x_{i,1} \dots x_{i,j-1}) = \int_{l-\frac{1}{2}}^{l+\frac{1}{2}} f\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \mathrm{d}x_{i,j}. \tag{2}$$

This scheme for performing lossless image compression can be used in combination with any density model, provided that we can compute Equation 2. This way arithmetic coding can be applied to the image using the statistics of the trained model. Algorithm 1 gives a summary for lossless compression with a mixture model.

As already mentioned, when using an mixture model, it is more efficient to use a single component for the encoding of a patch than the whole mixture. For both normally and Student-t distributed variables, the expressions for Equation 2 can be derived from their conditional distributions.
For the normal distribution this becomes:

$$\int_{l-\frac{1}{2}}^{l+\frac{1}{2}} \mathcal{N}\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \mathrm{d}x_{i,j} = F_n\left(l + \frac{1}{2} | \tilde{\mu_j}, \tilde{\sigma_j}^2\right) - F_n\left(l - \frac{1}{2} | \tilde{\mu_j}, \tilde{\sigma_j}^2\right),$$

with F_n the cumulative distribution function (CDF) of the univariate normal distribution, and where

$$\tilde{\mu_j} = \mu_j + \Sigma_{j,1:j-1}\Sigma^{-1}_{1:j-1,1:j-1}\left(x_{1:j-1} - \mu_{1:j-1}\right),$$
$$\tilde{\sigma_j}^2 = \Sigma_{j,j} - \Sigma_{j,1:j-1}\Sigma^{-1}_{1:j-1,1:j-1}\Sigma_{1:j-1,j}.$$

For the multivariate Student-t distribution the equations are similar:

$$\int_{l-\frac{1}{2}}^{l+\frac{1}{2}} \mathcal{T}\left(x_{i,j} | x_{i,1} \dots x_{i,j-1}\right) \mathrm{d}x_{i,j} = F_t\left(l + \frac{1}{2} | \tilde{\nu_j}, \tilde{\mu_j}, \tilde{s_j}^2\right) - F_t\left(l - \frac{1}{2} | \tilde{\nu_j}, \tilde{\mu_j}, \tilde{s_j}^2\right),$$

with F_n the CDF of the non-standardized univariate Student-t distribution (which has a location and scale parameter), and where

$$\tilde{\nu_j} = \nu_j + j - 1,$$
$$\tilde{\mu_j} = \mu_j + \Sigma_{j,1:j-1}\Sigma^{-1}_{1:j-1,1:j-1}\left(x_{1:j-1} - \mu_{1:j-1}\right),$$
$$\tilde{s_j}^2 = \left(\frac{\nu + x^T_{1:j-1}\Sigma^{-1}_{1:j-1,1:j-1}x_{1:j-1}}{\nu + j - 1}\right)\left(\Sigma_{j,j} - \Sigma_{j,1:j-1}\Sigma^{-1}_{1:j-1,1:j-1}\Sigma_{1:j-1,j}\right).$$

When using a form of entropy coding, such as arithmetic coding, the theoretical optimal code length for a symbol i is dependent on the probability P_i of it occurring: $-\log(P_i)$. Therefore, the lower bound on the expected rate (bits per symbol) is: $-\frac{1}{N}\sum_{i=1}^{N} P_i \log(P_i)$. Because P_i is calculated by a density model (Equation 2), the log likelihood score of this model is a good indication for how well it performs on lossless compression.

Algorithm 1: Lossless image compression with a mixture model. [AC] stands for arithmetic coding.

```
Encoder:
for each patch $x_i$ in image do
    $\beta = \arg\max_k f_k(x_i)$
    [AC] Encode symbol $\beta$ with probability table $\pi$ (mixing weights)
    for each $x_{ij},\ j = 1 \ldots p$ do
        Use Equation 2 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | z_{i,1} \ldots x_{i,j-1})$
        [AC] Encode symbol $x_{ij}$ with probability table $\alpha_{i,j}$
    end
end

Decoder:
while not at end of bit stream do
    [AC] Decode symbol $\beta$ with probability table $\pi$ (mixing weights)
    initialize $x_i$
    for $j = 1 \ldots p$ do
        Use Equation 2 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | z_{i,1} \ldots x_{i,j-1})$
        [AC] Decode symbol $x_{ij}$ with probability table $\alpha_{i,j}$
    end
end
Reconstruct image from patches $x_i,\ i = 1 \ldots N$.
```

4.3 Lossy Compression

For lossy compression, the image reconstruction after decompression does not have to be identical to the original, but should match it very closely. The strength of compression should be as high as possible, given a certain tolerable amount of distortion. This freedom evidently allows stronger compression than with lossless algorithms.

Lossy image compression algorithms typically use quantization to reduce the amount of information that needs to be entropy coded. Quantization decreases the number of states of the data variables to a smaller discrete set. As mentioned above we will use simple scalar quantization as the number of variables in a patch is relatively high and vector quantization would simply be impractical. Instead of VQ, we will combine scalar quantization with a data transform step, as is done in most image compression schemes.

The main reason of a data transform step in compression schemes is to decorrelate the input, so that the different coefficients can be handled more independently afterwards. Especially when using scalar quantization it is important to use a form of transformation first, as this reduces the amount of redundancy in the data that has to be encoded. Moreover, if one would quantize the image in the original pixel domain, the reconstruction artifacts would be very obtrusive. Because a Gaussian or Student-t mixture component already models covariance, decorrelation is fairly straightforward. The transform step is as follows:

$$y_i = W^T\left(x_i - \mu\right), \tag{3}$$

where W is the eigenvector matrix of the covariance matrix Σ of the Gaussian/Student-t mixture component: $WJW^T = \Sigma$. J is the diagonal eigenvalue matrix of Σ. Subsequently, the transformed values are quantized with a uniform quantizer:

$$z_i = \text{round}\left(\frac{y_i}{\lambda}\right). \tag{4}$$

The strength of the quantization only depends on λ. When it is high, the quality of the encoded image will be low, but the compression ratio will be high.

Once quantization is done, arithmetic coding is carried out in a similar fashion as with lossless compression: we have to be able to compute Equation 2. Because the data is transformed (Equation 3), the mean of y becomes 0: $\mu_y = 0$ and the covariance matrix reduces to: $\Sigma_y = J$. The equations for calculating the conditional probabilities from before can now be simplified.
For the normal distribution:

$$\begin{aligned} P\left(z_{i,j} = l | z_{i,1}, \ldots, z_{i,j-1}\right) &= \int_{\lambda\left(l-\frac{1}{2}\right)}^{\lambda\left(l+\frac{1}{2}\right)} \mathcal{N}\left(y_{i,j} | \tilde{y}_{i,1} \ldots \tilde{y}_{i,j-1}\right) \mathrm{d}y_{i,j} \\ &= F_n\left(\lambda\left(l+\frac{1}{2}\right)|0, J_j\right) - F_n\left(\lambda\left(l-\frac{1}{2}\right)|0, J_j\right), \end{aligned} \tag{5}$$

with F_n the cumulative distribution function (CDF) of the univariate normal distribution, and $\tilde{y}_{i,*}$ is the reconstruction of $y_{i,*}$ (as we will discuss later).

For the Student-t distribution:

$$\begin{aligned} P\left(z_{i,j}=l|z_{i,1},\ldots,z_{i,j-1}\right) &= \int_{\lambda\left(l-\frac{1}{2}\right)}^{\lambda\left(l+\frac{1}{2}\right)} \mathcal{T}\left(y_{i,j}|\tilde{y}_{i,1}\ldots\tilde{y}_{i,j-1}\right)\mathrm{d}y_{i,j} \qquad (6)\\ &= F_t\left(\lambda\left(l+\frac{1}{2}\right)|\tilde{\nu_j},0,\tilde{s_j}^2\right)-F_t\left(\lambda\left(l-\frac{1}{2}\right)|\tilde{\nu_j},0,\tilde{s_j}^2\right), \end{aligned}$$

with F_t the CDF of the non-standardized univariate Student-t distribution (which has a location and scale parameter), and where

$$\tilde{\nu_j}=\nu_j+j-1,$$

$$\tilde{s_j}=\left(\frac{\nu_j+\sum_{m=1}^{j-1}\frac{\tilde{y}_{i,m}^2}{J_m}}{\nu_j+j-1}\right)J_j.$$

Because of the two additional steps (Transformation and Quantization) during compression, the decoder has to dequantize and subsequently detransform the data after arithmetic coding.
Dequantization:

$$\tilde{y_i}=\lambda z_i. \qquad (7)$$

Inverse transform:

$$\tilde{x_i}=W\tilde{y_i}+\mu. \qquad (8)$$

4.3.1 Uniform Threshold Quantization

It is important to note that Equation 7 might not be the best choice for reconstruction. It is indeed possible to increase the quality of dequantization by using prior knowledge of the scalar input distribution. This concept is called uniform threshold quantization (Pearlman and Said, 2011). Figure 4 shows the difference with regular uniform quantization.

Depending on the assumed distribution of the source it is possible to minimize the expected distortion: $\left(\tilde{y}_{i,j}-y_{i,j}\right)^2$ (other measures of distortion can also be used). This comes down to solving the following optimization problem:

$$\tilde{y}_{i,j}=\arg\min_y\int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)}\|y-x\|^2 f(x)\,\mathrm{d}x,$$

which can be simplified to:

$$\tilde{y}_{i,j}=\frac{\int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)}xf(x)\,\mathrm{d}x}{\int_{\lambda\left(z_{i,j}-\frac{1}{2}\right)}^{\lambda\left(z_{i,j}+\frac{1}{2}\right)}f(x)\,\mathrm{d}x}.$$

This is actually nothing more than the centroid in that region (see Figure 4). Because we are using a probabilistic method, this improved reconstruction almost comes for free: The

Algorithm 2: Lossy image compression with a mixture model. [AC] stands for arithmetic coding.

```
Encoder:

for each patch $x_i$ in image do
    $\beta = \arg\max_k f_k(x_i)$
    [AC] Encode symbol $\beta$ with probability table $\pi$ (mixing weights)
    Transform $x_i$ with Equation 3 using the $\beta$-th component
    Quantize $x_i$ with Equation 4
    for each $x_{ij}$, $j = 1 \dots p$ do
        Use Equation 5 or 6 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$
        [AC] Encode symbol $x_{ij}$ with probability table $\alpha_{i,j}$
    end
end

Decoder:

while not at end of bitstream do
    [AC] Decode symbol $\beta$ with probability table $\pi$ (mixing weights)
    initialize $x_i$
    for $j = 1 \dots p$ do
        Use Equation 5 or 6 to compute: $\alpha_{i,j,l} = P_\beta(x_{i,j} = l | x_{i,1} \dots x_{i,j-1})$
        [AC] Decode symbol $x_{ij}$ with probability table $\alpha_{i,j}$
    end
    Dequantize $x_i$ with Equation 7 or 9
    Inverse transform $x_i$ with Equation 8
end
Reconstruct image from patches $x_i$, $i = 1 \dots N$.
```

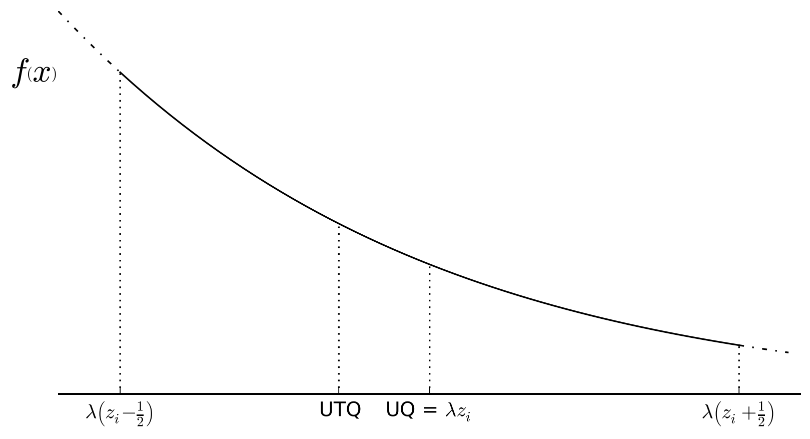


Figure 4: Uniform quantization versus uniform threshold quantization. During dequantization, UQ will reconstruct the input with the centers of the quantization intervals. UTQ uses the centroids instead.

compression scheme remains the same, only the decompression is improved. For a normally distributed variable the reconstruction is

$$\tilde{y}_{i,j} = \frac{\sqrt{\frac{J_j}{2\pi}} \left[\exp\left(-\frac{\lambda^2 \left(z_{i,j} - \frac{1}{2}\right)^2}{2J_j} \right) - \exp\left(-\frac{\lambda^2 \left(z_{i,j} + \frac{1}{2}\right)^2}{2J_j} \right) \right]}{F_n\left(\lambda\left(z_{i,j} + \frac{1}{2}\right) | 0, J_j\right) - F_n\left(\lambda\left(z_{i,j} - \frac{1}{2}\right) | 0, J_j\right)},$$

and for a Student-t distributed variable it is

$$\tilde{y}_{i,j} = \frac{\frac{\Gamma\left(\frac{\tilde{\nu_j}+1}{2}\right)}{\sqrt{\pi}\Gamma\left(\frac{\tilde{\nu_j}}{2}\right)} \frac{\tilde{s}_j^{\tilde{\nu_j}} \tilde{\nu_j}^{\frac{\tilde{\nu_j}}{2}}}{\tilde{\nu_j} - 1} \left[\left(\tilde{\nu_j} \tilde{s}_j^2 + \lambda^2 \left(z_{i,j} - \frac{1}{2}\right)^2 \right)^{\frac{1-\tilde{\nu_j}}{2}} - \left(\tilde{\nu_j} \tilde{s}_j^2 + \lambda^2 \left(z_{i,j} + \frac{1}{2}\right)^2 \right)^{\frac{1-\tilde{\nu_j}}{2}} \right]}{F_t\left(\lambda\left(z_{i,j} + \frac{1}{2}\right) | \tilde{\nu_j}, 0, \tilde{s_j}^2\right) - F_t\left(\lambda\left(z_{i,j} - \frac{1}{2}\right) | \tilde{\nu_j}, 0, \tilde{s_j}^2\right)}.$$

5. Results and Discussion

In this section we will discuss the experimental results of the STM on density modelling and image compression tasks.

5.1 Data Sets and Methods

We will first introduce the data sets that we used for our experiments and also discuss the image compression standards (JPEG, JPEG 2000) which will be used to compare the compression results with.

5.1.1 Berkeley Segmentation Data Set

The Berkeley Segmentation Data set (Martin et al., 2001) consists of 200 training and 100 test JPEG-encoded images, originally intended to be used as a segmentation benchmark.

Some samples can be seen on Figure 5. This data set has been used by several authors to measure the unsupervised learning performance of their model on image patches (Zoran and Weiss, 2011; Tang et al., 2013; Uria et al., 2013a). We adopt the use of this data set for measuring density modeling performance, but not for image compression, as these images were already encoded with JPEG and will already contain some quantization noise.

5.1.2 UCID Data Set

Although images are abundant on the world wide web, large data sets containing losslessly encoded images are rather hard to find. In image processing most authors have grown to rely on a particular set of standard images, such as Lena, Baboon, Peppers, etc. [1] to measure their algorithms' performance. Although each of these images have specific features that make them interesting to test a new method on, results on this small set likely will not generalize to a wide range of images. Furthermore, because there is no clear distinction between a training and test set for these images, there is a high risk of overfitting (even when engineering a compression scheme). Finally most of these images are relatively old and noisy, so they are hardly representative for images of modern photography.

On of the few publicly available data sets is UCID (Schaefer and Stich, 2003) (Uncompressed Colour Image Data set). The UCID database consists of 1338 TIFF images on a variety of topics including natural scenes and man-made objects, both indoors and outdoors. The camera settings were all set to automatic as this resembles what the average user would do. All the images have sizes 512x384 or 384x512. The images are in true color (24-bit RGB, each color channel having 256 possible values per pixel). Some sample UCID images can be seen on Figure 6.

As the images are not in random order, we have included every 10th image (10, 20, 30, ..., 1330) of the data set in our test set, and the others were used for training. This results in 1205 images for training and 133 images for testing. We randomly sample a large set of image patches (two million) for training the mixture models. We then encode every test image with a number of different quantization strengths (only for lossy compression), and measure their compression performance and the distortion of their reconstruction.

5.1.3 JPEG and JPEG 2000

For comparison we added two image compression standards as benchmark: JPEG (Wallace, 1991) and JPEG 2000 (Skodras et al., 2001).

JPEG is a patch based compression standard which uses the DCT as its transform, with quantization and entropy coding optimized for this transform. For the JPEG standard, we employed the widely used libjpeg implementation (`ijg.org`). Optimization of the JPEG entropy encoding parameters was enabled for better performance. The quality parameter was swept from 0 (worst) to 100 (best) in steps of 5.

JPEG 2000 is a wavelet-based compression standard and because of its multiresolution decomposition structure, it is able to exploit wider spatial correlations than JPEG and our method (which are patch based). For JPEG 2000, the kakadu implementation was

1. Most of these standard images can be found here: `http://sipi.usc.edu/database/database.php?volume=misc`

used (`kakadusoftware.com`). To make a fair comparison, command line parameters were enabled to optimize the PSNR instead of perceptual error measures.

For both methods we did not take the meta information of the header into account when measuring the performance of compression.

Figure 5: Sample images from the Berkeley Segmentation Data set.

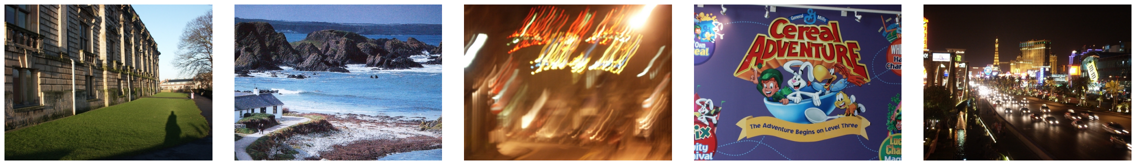


Figure 6: Sample images from the UCID data set (Uncompressed Colour Image Data set).

5.2 Average Patch Log Likelihood Comparison

Two million 8x8 patches were extracted from the training images, and 50,000 were extracted from the testing images (Berkeley data set). For every sample the mean was subtracted (DC component). Because the test patches we extracted could differ from those used by other authors, we report the mean and standard deviation across 10 randomly sampled sets of 50,000 patches. The results are listed in Table 2. The GMM and STM had 200 mixture components. As expected our GMM has a comparable result to that reported in literature. The proposed method STM significantly outperforms other methods.

We also compare our result with the recently proposed RNADE model (Uria et al., 2013b,a). Because the authors preprocess the gray scale patches differently, the results are not comparable to the ones reported in Table 2. Before subtracting the sample mean, small uniform noise (between 0 and 1) is added to the pixel values (between 0 and 255), which are then normalized by dividing by 256. Afterwards, they remove the last pixel, so that the number of variables of each datapoint equals 63. For this task we used four million patches during training and evaluated on one million patches from the test set. The results are shown in Table 3. The STM outperforms the deep RNADE model of 6 layers, but is on its turn outperformed by the ensemble of RNADE models (EoRNADE).

5.3 Lossless Compression

Because JPEG does not natively support lossless compression we excluded this benchmark for this test. For our methods we used patch size 8x8 and 128 mixture components. The results are listed in Table 4. As explained above (see Section 4.2), there is a connection

Indp. Pixel	ICA	GRBM	DBN	MFA
78.3	135.7	137.8	144.4	166.5
Deep MFA	MTA	GMM	**GMM**	**STM**
169.3	158.2	167.2	**166.97 ± 0.36**	**172.13 ± 0.42**

Table 2: Average log-likelihood (higher is better). Own results are marked in bold. The results of other methods are taken from Zoran and Weiss (2011); Tang et al. (2013). ICA: independent component analysis, GRBM: Gaussian restricted Boltzmann machine, DBN: Deep belief network, MFA: mixture of factor analyzers, MTA: Mixture of Tensor analyzers.

RNADE:	
1hl, 2hl, 3hl	143.2, 149,2, 152.0
4hl, 5hl, 6hl	153.6, 154.7, 155.2
EoRNADE (6hl)	157.0
GMM	**153.7**
STM	**155.3**

Table 3: Average log-likelihood comparison with RNADE (Uria et al., 2013a) in function of the number of hidden layers (hl). Our results are marked in bold. *These results are obtained from differently processed patches than those in Table 2 (see text).* EoRNADE stands for an ensemble of RNADE's.

between average log likelihood and the expected lossless compression strength. The STM also outperforms the GMM on this task, and both methods outperform JPEG 2000.

5.4 Lossy Compression

We will first analyze the influence of the patch size on the lossy compression performance. The results are visualized in Figure 7. All mixture models were trained for 500 iterations and consist of 128 components. The reconstruction quality of an image is measured in peak signal-to-noise ratio: $\mathrm{PSNR} = 10 \log_{10} \frac{R^2}{\mathrm{MSE}}$, with R being the largest possible pixel value (255 in this case) and MSE being the average mean squared error.

Bigger patch sizes show better results for low bit rates and vice versa. This can be explained by the fact that when using larger patch sizes, covariance between more pixels

JPEG 2000	**GMM**	**STM**
12.40	**12.07**	**11.83**

Table 4: Lossless compression rate (in bits per pixel - lower is better). Naive encoding would result in 24 bits per pixel (true color images).

can be modeled simultaneously. This way the transform has the ability to decorrelate better, which is important for low bit rates. For higher bit rates, we approach a near-lossless region, where the log likelihood performance of the model is crucial. When modeling smaller patch sizes, the algorithm is less prone to overfit, resulting in better performance. We can see that these high-rate effects are most apparent for the GMM. The STM, which is more robust to overfitting, is able to model larger patch sizes.

Because the 8x8 patch size has a good performance in general for both methods, our final experiments are computed with this setting. Note that JPEG also uses 8x8 patches for its compression scheme. For different compression strengths we have computed the average PSNR of the reconstructed images. For some images, JPEG or JPEG 2000 was unable to encode them at a given rate (1, 2 and 10 images for 3, 4 and 5 bpp respectively), so these images where not taken into account at those rates.

The final results are shown on Figure 8. For all compression rates, JPEG is outperformed by the other methods. The proposed compression schemes are competitive with JPEG 2000, and relatively to JPEG they score quite similar. In all experiments uniform threshold quantization improves on standard uniform quantization. The GMM is always outperformed by the STM, and the difference increases for larger bit rates. JPEG 2000 slightly exceeds the performance of the GMM in all experiments, but is in turn surpassed by the STM, with the exception of the lowest bit rate. At low bit rates, correlations on a more global scale become more important, which is why the multiresolution wavelet transform of JPEG 2000 achieves a better performance than our patch-based approach in this setting. Extending our approach to a multiscale technique might therefore be a promising direction of future research.

In Figure 9 we have visualized some reconstructed images after compression with JPEG, JPEG 2000 and the proposed method (GMM and STM), for varying levels of compression strength: 1, 2.5 and 5 bits per pixel (bpp). This Figure is best viewed on the electronic version by zooming in on the different images. Because JPEG and the proposed method are block based, they have blocking artifacts that JPEG 2000 does not. The latter has more blurring artifacts. The proposed method seems to have the strongest visual artifacts in low-frequency regions, but performs well in high-frequency regions such as trees and leaves. This can be attributed to the fact that the compression method does not take into account the properties of human visual perception and therefore quantizes both high as low frequency regions equally strongly. One could improve the visual results by adding prior knowledge about the perceptual system, using a deblocking filter (or using an image reconstruction algorithm based on GMM/STMs with the expected patch log likelihood (EPLL) framework), extending the model so that it works with overlapping blocks (with the MDCT transform for example) or by making it multi-scale. However, these extensions are outside the scope of this work.

6. Conclusion and Future Work

The presented work consists of two main contributions: the introduction and analysis of the Student-t mixture as an image patch modeling technique, and the proposal of lossless and lossy image compression techniques based on mixture models.

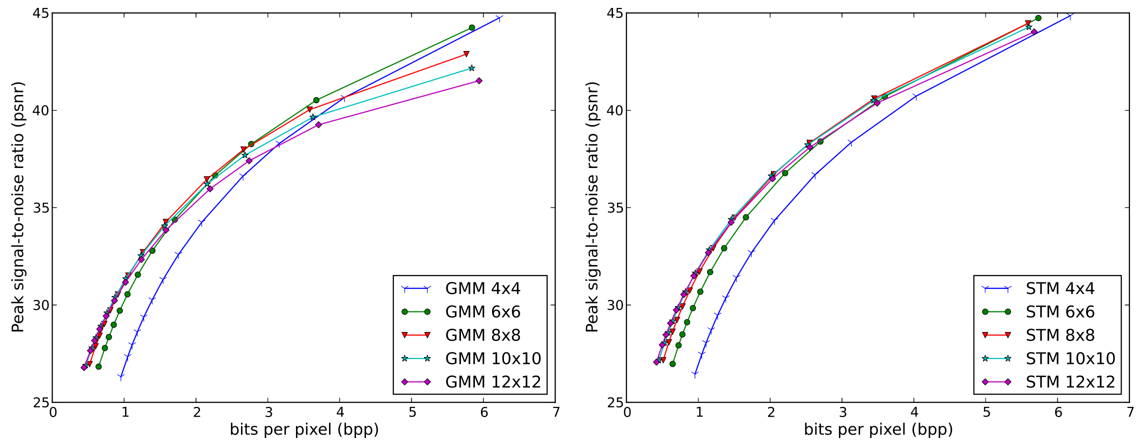


Figure 7: Average patch Quality (PSNR) - Rate (bpp) curves for different patch-sizes (GMM left, STM right). This Figure is best viewed in color.

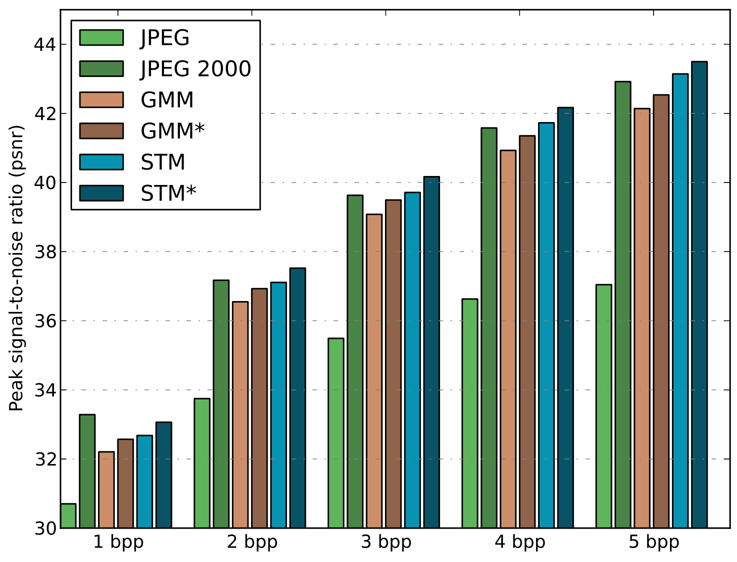


Figure 8: Results for lossy compression of colored images. Average quality (PSNR) in function of average rate (bits per pixel). Methods marked with a asterisk (*) use uniform *threshold* quantization, and thus have a better reconstruction error.

In the first part we have proposed the STM as an image patch prior. This method significantly outperformed the GMM for density modeling of image patches, with results competitive to the state-of-the-art on this task. This performance could largely be attributed to the fact that a Student-t mixture is able to model contrast in addition to linear dependencies within a single mixture component. For future work it would also be very interesting to see how it matches up with other methods on other types of data. Another possibility would be to study the Student-t mixture model for the use of image reconstruction applications (denoising, deblurring, inpainting), as was recently proposed with GMMs (Zoran and Weiss, 2011).

In the second part both the GMM and STM have been examined in this paper for the task of image compression. Lossless and lossy coding schemes were presented, which could easily be adapted for other unsupervised learning techniques. For lossy compression, experimental results demonstrated that the proposed techniques consistently outperform JPEG, with results similar to those of JPEG 2000. With the exception of the lowest bit rate, the STM has the advantage over JPEG 2000 in terms of rate-distortion. In lossless compression both the GMM and STM outperform JPEG 2000, which is mainly due to the fact that this task is even more connected with density estimation. In future work, even more advanced techniques will be considered. Moving beyond the 8x8 patch size, with for example multiscale techniques, is an especially promising direction.

One of the most important conclusions we can draw here is that relatively simple machine learning techniques can perform quite well on the task of image compression. We saw that their performance could largely be attributed to their density modeling capabilities. It would therefore be interesting to apply machine learning to compression of different types of data, such as audio, video, EEG, etc. and more specific types of data such as facial or satellite images. We also propose for compression to be used more in machine learning as a benchmark to compare models. Given the recent progress in unsupervised machine learning we expect that even better results will follow.

References

Anuradha Aiyer, Kyungsuk Pyun, Ying-zong Huang, Deirdre B OBrien, and Robert M Gray. Lloyd clustering of Gauss mixture models for image compression and classification. *Signal Processing: Image Communication*, 20(5):459–485, 2005.

Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

Richard P Brent. *Algorithms for Minimization without Derivatives.* Courier Dover Publications, 1973.

Ori Bryt and Michael Elad. Compression of facial images using the k-svd algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

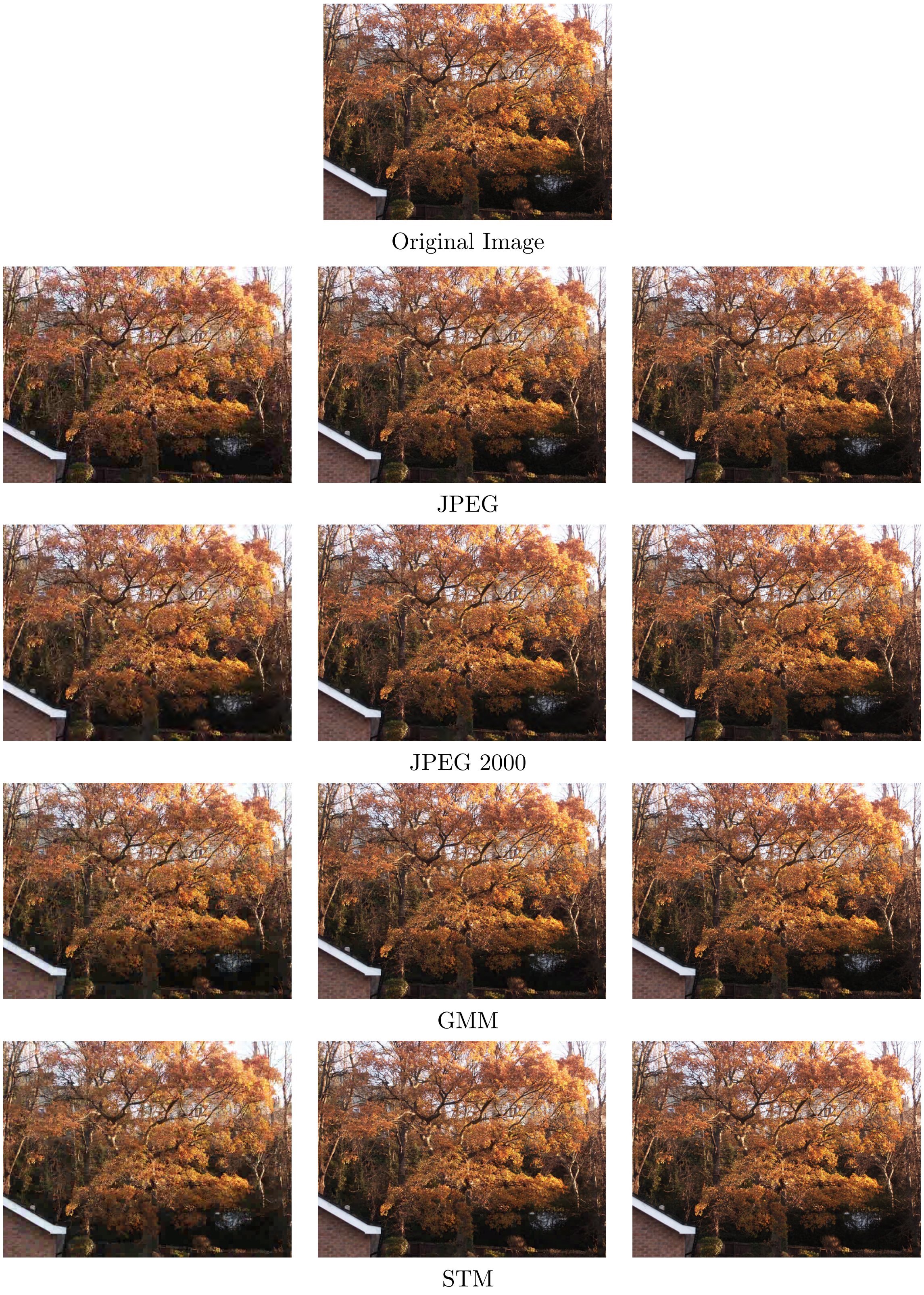


Figure 9: Reconstructions after compression by JPEG, JPEG 2000 or the proposed method with a GMM of STM. The rates were 1 bpp (left), 2.5 bpp (middle), 5 bpp (right). This figure is best viewed on the electronic version by zooming in on the images.

Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Transactions on Image Processing*, 15(12):3736–3745, 2006.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

Vivek K Goyal. Theoretical foundations of transform coding. *Signal Processing Magazine*, 18(5):9–21, 2001.

Per Hedelin and Jan Skoglund. Vector quantization based on Gaussian mixture models. *Transactions on Speech and Audio Processing*, 8(4):385–401, 2000.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

Wei Hong, John Wright, Kun Huang, and Yi Ma. A multiscale hybrid linear model for lossy image representation. In *International Conference on Computer Vision*, volume 1, pages 764–771. IEEE, 2005.

Inbal Horev, Ori Bryt, and Ron Rubinstein. Adaptive image compression using sparse dictionaries. In *International Conference on Systems, Signals and Image Processing*, pages 592–595. IEEE, 2012.

Samuel Kotz and Saralees Nadarajah. *Multivariate t-Distributions and their Applications.* Cambridge University Press, 2004.

Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision*, pages 2272–2279. IEEE, 2009.

David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

William A Pearlman and Amir Said. *Digital Signal Compression: Principles and Practice.* Cambridge University Press, 2011.

David Peel and Geoffrey J McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000.

Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Computer Vision and Pattern Recognition, 2005.*, volume 2, pages 860–867. IEEE, 2005.

Gerald Schaefer and Michal Stich. UCID: an uncompressed color image database. In *Electronic Imaging 2004*, pages 472–480. International Society for Optics and Photonics, 2003.

Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *Signal Processing Magazine*, 18(5):36–58, 2001.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep mixtures of factor analysers. In *International Conference on Machine Learning*, pages 505–512, 2012.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Tensor analyzers. In *International Conference on Machine Learning*, 2013.

Lucas Theis, Sebastian Gerwinn, Fabian Sinz, and Matthias Bethge. In all likelihood, deep belief is not enough. *The Journal of Machine Learning Research*, 12:3071–3096, 2011.

Lucas Theis, Reshad Hosseini, and Matthias Bethge. Mixtures of conditional Gaussian scale mixtures applied to multiscale image representations. *PLoS ONE*, 7(7), Jul 2012. doi: 10.1371/journal.pone.0039857.

D Michael Titterington, Adrian FM Smith, Udi E Makov, et al. *Statistical Analysis of Finite Mixture Distributions*, volume 7. Wiley New York, 1985.

Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. *arXiv preprint arXiv:1310.1757*, 2013a.

Benigno Uria, Iain Murray, and Hugo Larochelle. Nade: The real-valued neural autoregressive density-estimator. *arXiv preprint arXiv:1306.0186*, 2013b.

Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Learning a piecewise linear transform coding scheme for images. In *2012 International Conference on Graphic and Image Processing*, pages 876844–876844. International Society for Optics and Photonics, 2013.

Gregory K Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

Yair Weiss and William T Freeman. What makes a good model of natural images? In *Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

Max Welling, Simon Osindero, and Geoffrey E Hinton. Learning sparse topographic representations with products of Student-t distributions. In *Advances in Neural Information Processing Systems*, pages 1359–1366, 2002.

John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *Transactions on Image Processing*, 21(5):2481–2499, 2012.

Joaquin Zepeda, Christine Guillemot, and Ewa Kijak. Image compression using sparse representations and the iteration-tuned and aligned dictionary. *Journal of Selected Topics in Signal Processing*, 5(5):1061–1073, 2011.

Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*, pages 479–486. IEEE, 2011.

Daniel Zoran and Yair Weiss. Natural images, Gaussian mixtures and dead leaves. In *Advances in Neural Information Processing Systems*, volume 25, pages 1745–1753, 2012.