# Progress Report
# 1-18 October 2012

Amin Karbasi
E-mail: amin,karbasi@epfl.ch
Amir Hesam Salavati
E-mail: hesam.salavati@epfl.ch

Supervisor: Prof. Amin Shokrollahi
E-mail: amin.shokrollahi@epfl.ch

Algorithmics Laboratory (ALGO)
Ecole Polytechnique Federale de Lausanne (EPFL)

October 24, 2012

## 1 Summary

In the past two weeks, I was mainly working on applying our method to a database of natural images. I have played with different models and tested a few different ideas to find the best result. I will explained these approaches, most of which failed, as well as some ideas I am going to work for future.

I also worked on finalizing the journal version of the ITW paper. The newest version of this paper can be accessed separately.

## 2 Applying our model to datasets of natural images

To apply our neural error correcting model to any dataset, we must find a suitable representations such that the (sub-) patterns in this representation form a subspace. Therefore, to apply our algorithm to a dataset of images, we are looking for such a representation. As explained in previous reports, we have considered pixel-level representation of images and it did not work. Therefore, in this report we will explore various different forms of representations:

1. Pixel-level representation but for the pre-processed images

2. Sparse coding technique: Find the "natural" bases of representing an image and represent each image as a set of coefficients in that basis. This approach is inspired by the *sparse coding* method of [1] and utilizes the code desgined by the authors to learn the "natural" bases and the coefficients of each input image with respect to this set of bases.

3. Sparse filtering: we learn the output of a suitably chosen *sparse filter* for a set of patterns. This approach is inspired by [2] and uses the code offered by the authors to learn both the filter itself and the coefficients corresponding to each input image.

We will discuss each of the above approaches in some more details below.

### 2.1 Pixel-level representation of Gabor-filtered images

This idea corresponds to the model we introduced in the last report. In this model, as shown by figure 1, we first apply 16 Gabor wavelets to an image and reconstruct a rough version of the image using the output of the filters. We then divide the filtered image into *non-overlapping* smaller patches. We then applying PCA to these small patches to get a set of features as well as the corresponding coefficients for each patch of each image in the dataset. We store these coefficients for different patches of an image as a new pattern. We divide each of these patterns into random clusters and then apply our learning algorithm to learn a dual matrix $W^{(i)}$, where $i$ is the cluster index.

**Result:** Failure. We have applied this method to the class 1 of images in the STL-10 dataset[1]. However, the learning algorithm failed in the sense that the output matrix for the clusters contained many 0 columns. In other words, the algorithm only learned the constraints among only a few of the pattern nodes present in that cluster.

[1]Available at `http://www.stanford.edu/~acoates//stl10/`.

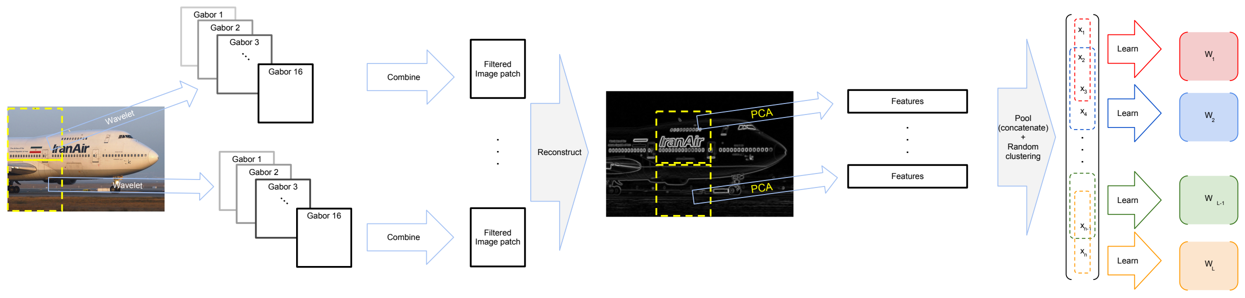

Figure 1: An example of reversible feature mapping. The mapping is reversible upto the wavelet reconstruction of the image (in black and white).

## 2.2 Sparse Gabor-filtered images

We also tried applying the above approach to sparse version of the filtered image, i.e. by setting small values to zero.

**Result:** Failure. Again, the learning algorithm failed in the sense that the output matrix for the clusters contained many 0 columns.

Obviously, this is not desirable and we had to seek an alternative approach.

## 2.3 Sparse coding technique

So if representing images at the pixel level does not work, why not express images in terms of a set of proper bases and then apply the learning algorithm to the coefficients of images with respect to these bases? In other words, instead of learning an image $I(x, y)$, we express it in terms of a set of bases as $I(x, y) \approx \sum_i a_i \phi(x, y)$, where $\phi_i$'s are the bases, and then learn the coefficients $a_i$. This is in fact very similar to what we did for the dataset of spoken English words as we learned the Fourier coefficients of an audio signal instead of the signal itself.

At this point, the bases that we choose become crucial. However, Olshausen and Field [1] have proposed a very interesting approach that learns both a set of proper bases and the corresponding coefficients such that:

1. The reconstruction error between the original image $I(x, y)$ and its reconstructed version $\sum_i a_i \phi(x, y)$ is minimized.
2. The coefficients $a_i$ are designed to be sparse.

Furthermore, the authors show that using this method, the set of bases converge to what simple cells in human visual system perform, i.e. a form of Gabor-like wavelet transform for edge detection.

Thus, we first apply their method to our database of natural images in order to determine the set of coefficients $a_i$ and then apply our learning algorithm to these coefficients.

Thankfully, the authors have shared their code on the web[2] so with the help of Mr. Masoud Alipour, we could easily run the code and get the corresponding coefficients. However, we should first preprocess the images according to the directions provided in their website. Here are the necessary steps (quoted from the website `http://redwood.berkeley.edu/bruno/sparsenet/`):

[2]Available at `http://redwood.berkeley.edu/bruno/sparsenet/`.

1. First, you should make sure all images have approximately the same overall contrast. One way of doing this is to normalize each image so that the variance of the pixels is the same (i.e., 1).

2. Then you will need to prewhiten the images. Basically, to whiten an image of size $N \times N$, you multiply by the filter $f * e^{-(f/f_0)^4}$ *in the frequency domain*, where $f_0 = 0.4N$ ($f_0$ is the cutoff frequency of a lowpass filter that is combined with the whitening filter). For a full explanation of whitening see [3].

3. Once you have preprocessed a number of images this way, all the same size, then you should combine them into one big $N \times N \times M$ matrix, where $M$ is the number of images.

4. Then rescale this matrix so that the average image variance is 0.1 (the parameters in the sparsenet code are set assuming that the data variance is 0.1).

5. Name this array IMAGES, save it to a file for future use, and you should be off and running.

Thankfully, the code for the pre-processing was available as well. After pre-processing the images, we apply their code and here is a summary of the procedure their algorithm performs:

1. Initialize the matrix of bases $\Phi$ randomly.

2. Then load an image randomly from the dataset.

3. Now extract 100 *random* $8 \times 8$ patches from this image. Put them in a $64 \times 100$ matrix $X$.

4. Using the conjugate gradient method, find the set of coefficients $A$ such that $\|E\|^2 = \|X - \Phi A\|^2$ is minimized and $A$ is rather sparse. Here, the size of $A$ is $m \times 100$ and the size of $\Phi$ is $64 \times m$, where $m$ is the number of bases.

5. Now update the bases: compute $\Delta\phi_i = \sum_{j=1}^{100} E(j,:)A(j,i)A$ and set $\phi_i = \phi_i + \eta\Delta\phi_i$. This is inline with equation (6) of their paper [1].

6. Now adjust the variance of the bases such that they lie within the desired bounds (to make sure that they do not diverge).

7. Go back to step 2 and repeat this process for all the images in the dataset several times until the bases are learned properly.

Figure 2 illustrates the 64 oriented edge-detectors learned by the *sparsenet* code.

After learning the bases, we repeat the above procedure to learn the coefficients for each image, with two modifications:

1. Step 1 and 5 will not be executed since $\Phi$ is fixed now.

2. In step 3, instead of taking 100 *random* patches, we systematically divide each image into *non-overlapping* $8 \times 8$ patches. We then learn the coefficients for each patch and store these coefficients (for all patches) as the new pattern.

In our experiments, we set $m$, the number of bases, to 64. Therefore, for each $8 \times 8$ patc, we get 64 coefficients $a_i$, which could again be arranged into a patch of $8 \times 8$. As a result, the coefficients of an $N \times N$ image could also be arranged as an $N \times N$ matrix, which could then be used as the input to our learning algorithm.

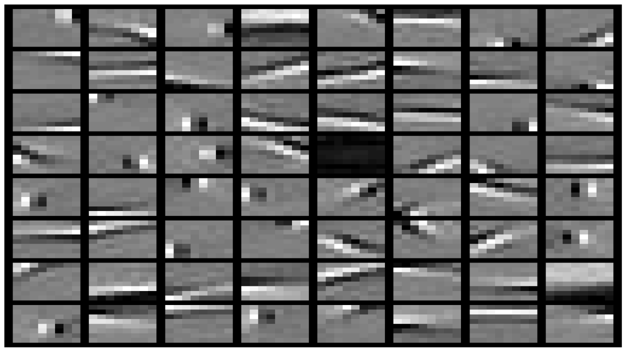

Figure 2: The set of 64 learned $8 \times 8$ bases in gray-scale. Note that each filter is sensitive to an oriented edge.

**Result:** Failure. The overall approach did not work as the learning algorithm was not able to learn a matrix orthogonal to the set of coefficients. We also tried different quantization techniques to make the coefficients sparser but the learning algorithm could neither learn the quantized values nor their indices (i.e. if we try to learn the indices of the quantized values instead of the values themselves).

## 2.4 Pixel-level representation of pre-processed images

Having the pre-processing method by [3], one might try also to learn just the pre-processed filters from the pixel-level.

**Modified (reversible) approach:** Failed. We have tried this approach and like the Gabor-filtered images, this one failed as well.

## 2.5 Sparse filtering

As another method, we used the sparse filtering technique for unsupervised feature learning proposed in [2]. Thankfully, the authors have shared their code on the web[3]. We gave our database of pre-processed images (according tot he procedure explained above) as the input[4] and the code returns two sets of learned coefficients for each input image: one for the first layer of feature-extraction, applied directly to the images, and the second one for the output of the second layer of feature extraction, applied to the output of the first layer.

Once the coefficients are returned by the algorithm, we assemble them again to an image and then divide this image into *overapping* patches which are then used as the input to our neural approach.

In its original form, this approach leads to irreversible feature mapping, i.e. one can not reconstruct the original image from the set of images because of the normalizations done in each stage of learning. However, one might be able to reconstruct a visually *similar* image even after

[3]Available at `https://github.com/jngiam/sparseFiltering`

[4]Note that the input to the algorithm should be in the form of *patches* of the images. So for instance if we have $N$ images of size $96 \times 96$, each image contains 36 patches of size $16 \times 16$ and we should store these patches in a big $36N \times 256$ matrix and give it to the code as the input.

normalizations. This is the subject of further investigations though. However, up to now, we have tested two versions of this approach: its original form and the one without normalization in each stage of feature extraction.

**Original approach:** Succeeded. This approach somewhat worked as we applied our learning algorithm to the features returned by the second stage of the feature extraction. However, as stated above, this whole scheme would be irreversible and this model would be suitable for *feature denoising*. Note that this might be acceptable for improving classification of noisy images. So we will investigate this opportunity more and even make an attempt to reconstruct visually similar images from the features (it seems that it shouldn't be impossible!).

**Modified (reversible) approach:** Failed. This approach failed because the returned coefficients by the feature extraction method was quite large. Therefore, it was very difficult for the dual learning approach to learn the coefficients. We will make another attempt in the upcoming weeks though.

## 3 Conclusions and future works

In the upcoming weeks, we are hopefully going to test a bunch of other techniques to find the best model for our framework. To list a few:

1. Apply 2D wavelet transform to images and learn the resulting coefficients.
2. Continue working on the sparse filtering approach [2].
    - For this approach, we either try to find a way to reconstruct the original image from the features, Or
    - try to use our model to design a classification system to improve the accuracy of the classifiers for noisy images by denoising the images first and then classify them.
3. finally, if the above approaches failed, we try to switch the dataset and find a more suitable dataset for our approach.

## References

[1] B. Olshausen, D. Field, *Emergence of simple-cell receptive eld properties by learning a sparse code for natural images.* Nature, 1996

[2] J. Ngiam, P. Koh, Z. Chen, S. Bhaskar, A.Y. Ng., *Sparse filtering*, NIPS 2011.

[3] B. A. Olshausen BA, D. J. Field, *Sparse coding with an overcomplete basis set: a strategy employed by V1?*, Vision Research, Vol. 37, 1997, pp. 3311-3325.