Sparse Autoencoder

Akhil P M
SC14M044

Autoenoder

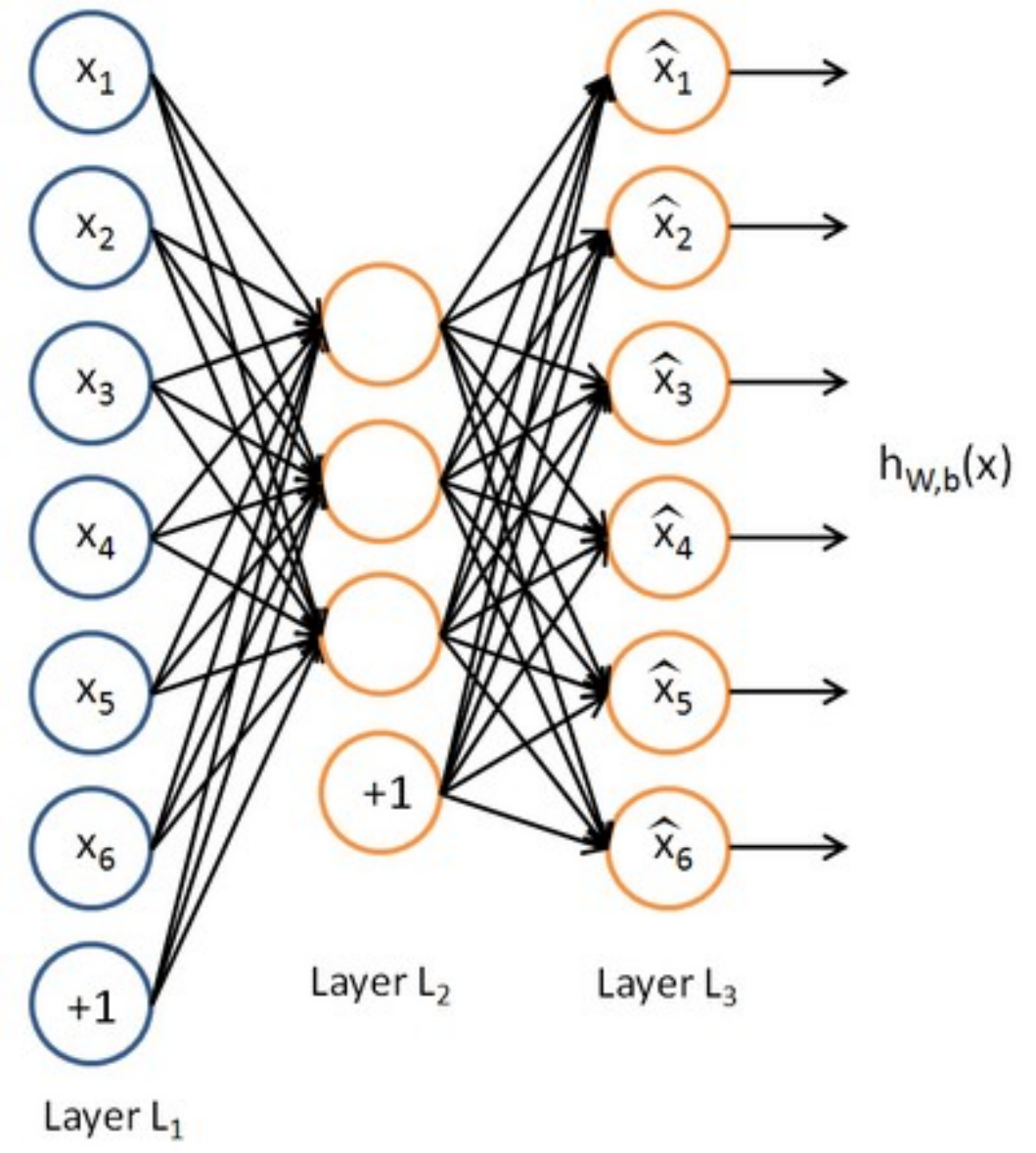
- An autoencoder is a ***neural network*** trained with backpropagation algorithm (using unsupervised learning) by gradient descent.
- The aim of an auto-encoder is to learn a ***compressed, distributed representation*** (encoding) for a set of data, typically for the purpose of dimensionality reduction.
- The autoencoder learns a nonlinear approximation of the identity function which reconstructs the output from its input itself.

Autoencoder

let $\mathrm{x}^{(1)}, x^{(2)}, \ldots$ are a set of unlabelled training examples where each $x^{(i)} \in \mathbb{R}^n$.An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, by setting the target values to be equal to the inputs. I.e., it uses $y^{(i)} = x^{(i)}$

Here is an example

Autoencoder



Autoencoder

- by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data.
- Autoencoders can implement any of several variations of dimensionality reduction. A *linear autoencoder* will learn the principal variance directions (Eigenvectors) of the data, equivalent to ***applying PCA to the inputs***.
- A nonlinear autoencoder is capable of discovering more complex, multi-modal structure in the data.
- Hinton and Salakhutdinov have shown that nonlinear autoencoder networks can produce low-dimensional codes that outperform PCA for handwriting and face recognition

Example

suppose the inputs x are the pixel intensity values from a 10×10 image (100 pixels) so that $n = 100$, and there are $s_2 = 50$ hidden units in layer L_2. For the autoencoder $y \in \Re^{100}$. Since there are only 50 hidden units, the network is forced to learn a compressed representation of the input. I.e., given only the vector of hidden unit activations $a^{(2)} \in \Re^{50}$, it must try to reconstruct the 100 pixel input x.

Autoencoder

- even when the number of hidden units is large (perhaps even greater than the number of input pixels), we can still discover interesting structure, by imposing other constraints on the network. In particular, if we impose a ***sparsity constraint on the hidden units***, then the autoencoder will still discover interesting structure in the data, even if the number of hidden units is large.

Sparse coding

- A coding is said to be ***sparse*** when each item is encoded by the strong activation of a relatively small set of neurons.
- By enforcing sparse coding requirement, we can limit the *number of hidden units* that can be activated by each input pattern.
- Given a potentially large set of input patterns, sparse coding algorithms (e.g. Sparse Autoencoder) attempt to automatically find a small number of representative patterns which, when combined in the right proportions, reproduce the original input patterns. The sparse coding for the input then consists of those representative patterns.

Sparse coding

- For example, the very large set of English sentences can be encoded by a small number of symbols (i.e. letters, numbers, punctuation, and spaces) combined in a particular order for a particular sentence, and so a sparse coding for English would be those symbols.
- A sparse autoencoder includes a sparseness constraint on hidden layer activation that is enforced by the addition of the ***Kullback-Leibler (KL) divergence*** term to the objective function.

Sparse Autoencoder

let $a_j^{(2)}(x)$ denote the activation of hidden unit j when the network is given a specific input x. Further, let

$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j^{(2)}(x^{(i)}) \right]$

be the average activation of hidden unit j(averaged over the training set).We would like to enforce the constraint

$\hat{\rho}_j = \rho$

where ρ is a sparsity parameter, typically a small value close to zero (say$\rho = 0.05$).

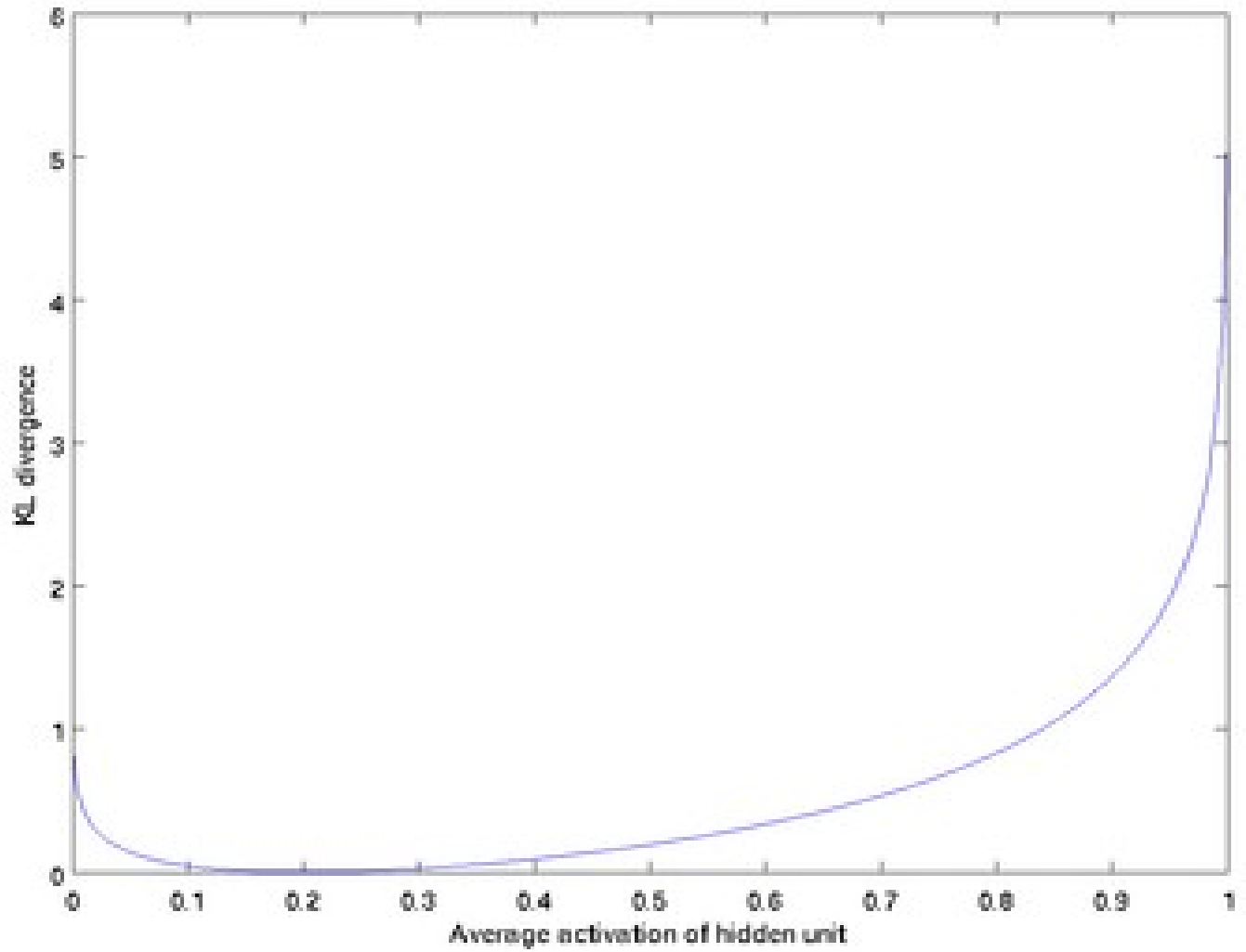
Sparse Autoencoder

In other words, we would like the average activation of each hidden neuron j to be close to 0.05. To achieve this, we will add an extra penalty term to our optimization objective that penalizes $\hat{\rho_j}$ deviating significantly from ρ. The penalty term is

$$\sum_{j=1}^{s_2} \mathrm{KL}(\rho||\hat{\rho}_j) = \sum_{j=1}^{s_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

where s_2 is the no of nodes in hidden layer.
This penalty function has the property that $\mathrm{KL}(\rho||\hat{\rho}_j) = 0$ if $\hat{\rho_j} = \rho$, and otherwise it increases monotonically as $\hat{\rho_j}$ diverges from ρ.

Sparse Autoencoder



Sparse Autoencoder

For example, in the figure above, we have set $\rho = 0.2$, and plotted $\mathrm{KL}(\rho||\hat{\rho}_j)$for a range of values of $\hat{\rho}_j$.
We see that the KL-divergence reaches its minimum of 0 at $\hat{\rho}_j = \rho$, and blows up as $\hat{\rho}_j$ approaches 0 or 1. Thus, minimizing this penalty term has the effect of causing $\hat{\rho}_j$ to be close to ρ.
Then the overall cost function now becomes

$$J_{\text{sparse}}(W, b) = J(W, b) + \beta \sum_{j=1}^{s_2} \mathrm{KL}(\rho||\hat{\rho}_j),$$

where $J(W, b)$is the usual NN cost function defined as

Sparse Autoencoder

$$\mathrm{J(W,b)} = \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\ \sum_{i=1}^{s_l}\ \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)}\right)^2$$

$$= \left[\frac{1}{m}\sum_{i=1}^{m} \left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\ \sum_{i=1}^{s_l}\ \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)}\right)^2$$

here λ is the regularization parameter.
The parameter β controls the weight of the sparsity penalty term.

Algorithm

1. Perform a feedforward pass, computing the activations for hidden and output layers
2. for the output layer set

$$\delta^{(3)} = -(y - a^{(3)}) \bullet f'(z^{(3)})$$

2. for the hidden layer set

$$\delta^{(2)} = \left((W^{(2)})^T \delta^{(3)} + \beta \left(-\frac{\rho}{\hat{\rho}} + \frac{1-\rho}{1-\hat{\rho}} \right) \right) \bullet f'(z^{(2)})$$

3. compute the desired partial derivatives over all x, y

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} (a^{(l)})^T,$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)}$$

Algorithm

4. Sum the partial derivatives over all x, y

$$\text{Set } \Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$$

$$\text{Set } \Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$$

5. Update the parameters

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

6. Repeat the process untill convergence

Lessons from MNIST dataset

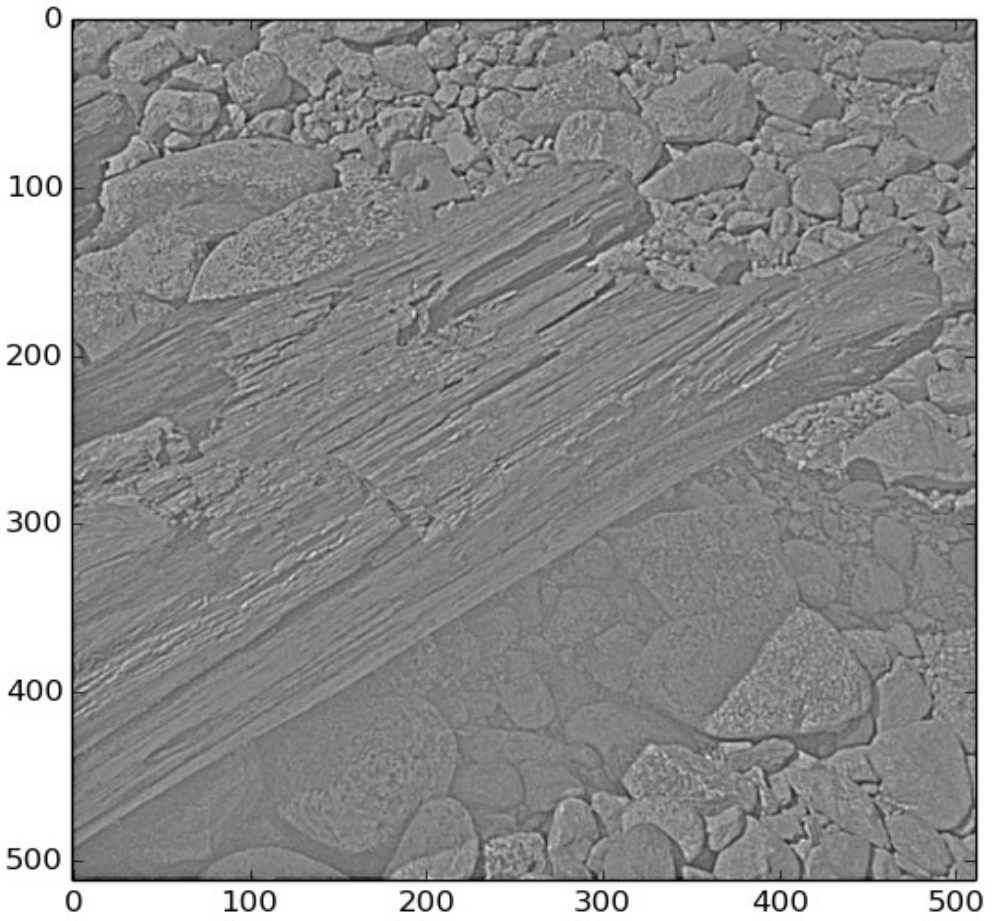
The Dataset

Lessons from MNIST dataset

Trained autoencoder output

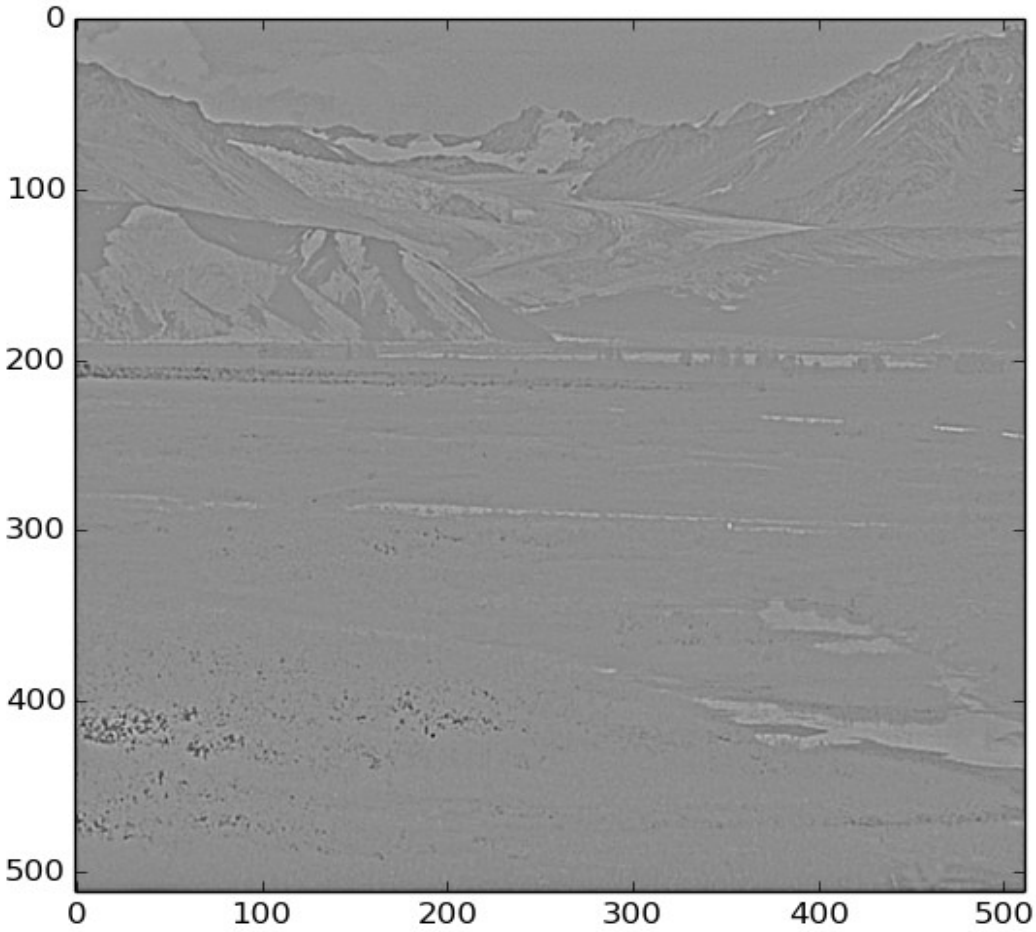
Natural Images Dataset

Sample Images



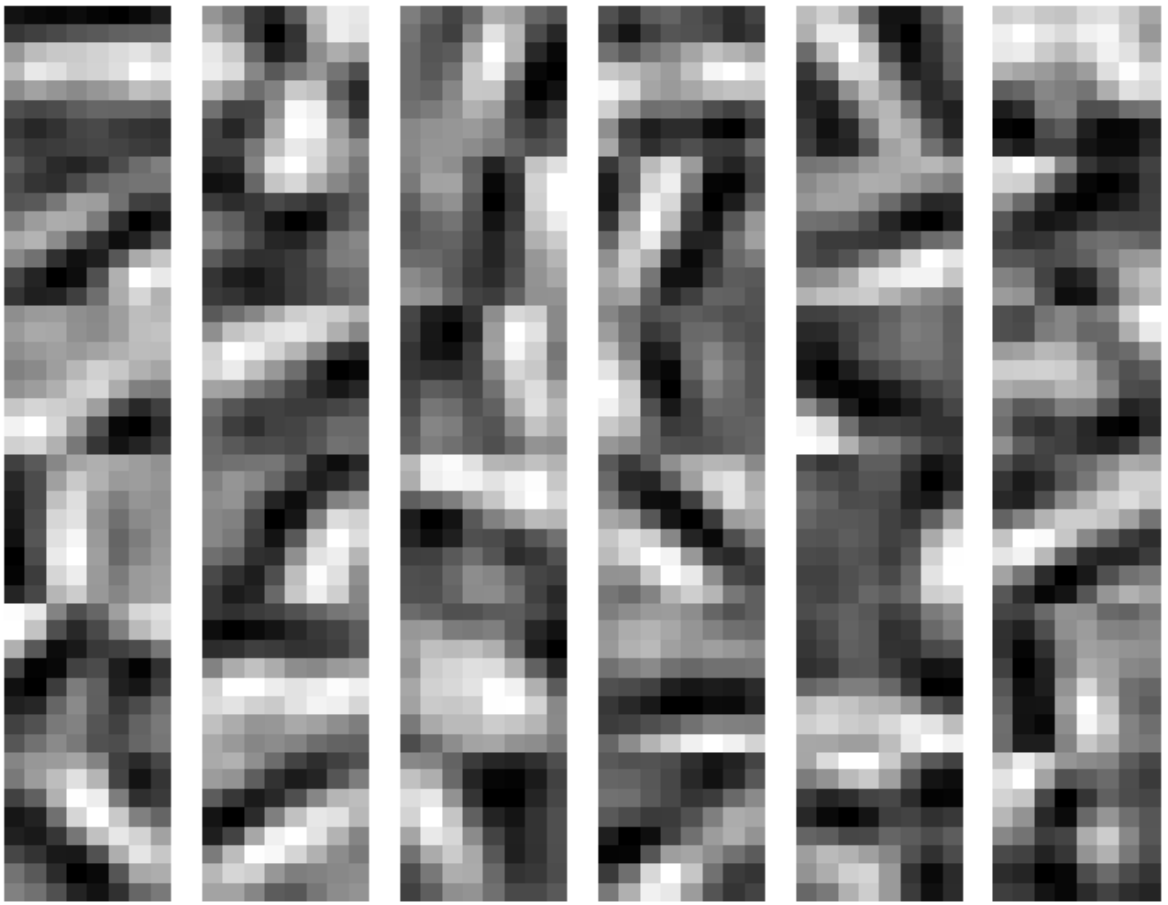
Natural Images Dataset

Sample Images



Natural Images Dataset

Trained Autoencoder Output



References

1. Michelle Shu, Alona Fyshe “ *Sparse Autoencoders for Word Decoding from Magnetoencephalography* ”, Proceedings of the third NIPS Workshop on Machine Learning - 2013 \

2. UFLDL tutorial, Stanford University
http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

3. Wikipedia – Autoencoder, Sparse coding

https://en.wikipedia.org/wiki/Autoencoder

https://en.wikipedia.org/wiki/Neural_coding#Sparse_coding


Thank
You