TRAINING A RESTRICTED BOLTZMANN MACHINE USING SPATIAL MARKOV RANDOM FIELD PRIORS ON WEIGHTS

by

Shweta Singhal

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

August 2017

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Shweta Singhal**
has been approved by the following supervisory committee members:

P. Thomas Fletcher,	Chair(s)	**05/03/2017** Date Approved
Vivek Srikumar,	Member	**05/03/2017** Date Approved
Jeffrey M. Phillips,	Member	**05/03/2017** Date Approved

by **Ross Whitaker**, Director of
the Department/College/School of **Computing**
and by **David B. Keida**, Dean of the Graduate School.

ABSTRACT

Recent developments have shown that restricted Boltzmann machines (RBMs) are useful in learning the features of a given dataset in an unsupervised manner. In the case of digital images, RBMs consider the image pixels as a set of real-valued random variables, disregarding their spatial layout. However, as we know, each image pixel is correlated with its neighboring pixels, and direct modeling of this correlation might help in learning. Therefore, this thesis proposes using a Markov random field prior on the weights of the RBM model, which is designed to model these correlations between neighboring pixels. We compared the test classification error of our model with that of a traditional RBM with no prior on the weights and with RBMs with L_1 and L_2 regularization prior on the weights. We used the MNIST dataset, which consists of images of handwritten digits for our experiments.

To my family & friends

CONTENTS

ABSTRACT . . . . **iii**

LIST OF FIGURES . . . . **vii**

LIST OF TABLES . . . . **viii**

NOTATION AND SYMBOLS . . . . **ix**

ACKNOWLEDGEMENTS . . . . **x**

CHAPTERS

1. INTRODUCTION **1**

2. BACKGROUND **4**

2.1 Graphical Models 4
2.1.1 Undirected Graphs 4
2.2 Markov Chain and Markov Chain Monte Carlo Techniques 5
2.2.1 Markov Chain 6
2.2.2 Gibbs Sampling 6
2.2.3 Metropolis-Hastings Algorithm 7
2.3 Energy-Based Models (EBM) 7
2.4 Restricted Boltzmann Machine 10
2.4.1 Neural Network With Binary Units 11
2.4.2 Log-Likelihood 11
2.4.3 Algorithms to Approximate Log-Likelihood Gradients 12
2.4.3.1 Contrastive Divergence 13
2.4.3.2 Persistent Contrastive Divergence (PCD) 14

3. MARKOV RANDOM FIELD PRIOR ON RBM **16**

3.1 Markov Random Field Prior 16
3.1.1 Evaluating Gradient of Log-Likelihood 18
3.2 Learning Algorithm 19

4. EXPERIMENTS **20**

4.1 Experimental Setup 20
4.2 Classification of MNIST Dataset 21

APPENDICES

A. TEST CLASSIFICATION ERROR RESULTS **27**

B. WEIGHT FILTERS OBTAINED **30**
REFERENCES **36**

LIST OF FIGURES

1.1 Boltzmann machine [5]. 2

1.2 restricted Boltzmann machine. 2

2.1 Undirected graphical model, with G = (V,E), $V \in \{v_1, \ldots, v_7\}$ where subset $\{v_4, v_5, v_6, v_7\}$ forms a maximal clique and $(v_4 \perp v_2 | v_1, v_5, v_7)$ shows conditional independence. 5

2.2 Gibbs sampling in RBM (referenced from [8]). 13

3.1 RBM weights leaned without any regularization with $h = 500$ after 15 epochs of the training set. 17

3.2 RBM weights. 17

4.1 Classification error on test dataset in traditional RBM with respect to increase in hidden units. 21

4.2 Classification error on test dataset. 22

4.3 Result of 10*10 matrix of filters learned with various models during their best results with h = 1000. 23

4.4 Learned hidden layer behaves as input to the logistic layer for classification of MNIST handwritten digits with different set of weights and biases for connection between the hidden layer of RBM and the logistic layer for classification. 24

B.1 Result of 10*10 matrix of filters learned with traditional RBM using various number of hidden units after 15 epochs. 31

B.2 Result of 10*10 matrix of filters learned with RBM with L_1 regularization after 15 epochs and $\lambda = 0.001$. 32

B.3 Result of 10*10 matrix of filters learned with RBM with L_2 regularization after 15 epochs and $\lambda = 0.00001$, also uniform weight initialization with seed = 1234. 33

B.4 Result of 10*10 matrix of filters learned with RBM with L2 prior and Gaussian initialization with std = 0.01 with $\lambda = 0.0001$ and obtained after 15 epochs. . . . 34

B.5 Result of 10*10 matrix of filters learned with RBM with spatial MRF prior after 15 epochs and $\lambda = 10^{-10}$ and uniform weight initialization. 35

LIST OF TABLES

4.1 MNIST test classification error. 24

4.2 Experimental analysis of test classification error with respect to random initialization (seed) factor. Experiments are initialized randomly with $h = 1000$, $\eta = 0.1$ and $\lambda = 0.0001$ for L_1 and L_2 regularization, and $\lambda = 10^{-10}$ for spatial MRF prior. 25

A.1 MNIST test dataset classification error in RBM with no prior. 27

A.2 MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.00001$. 27

A.3 MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.0001$. 28

A.4 MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.001$. 28

A.5 MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.01$. 28

A.6 MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.1$. 28

A.7 MNIST test dataset classification error in RBM with spatial MRF prior. 29

A.8 MNIST test dataset classification error in RBM with L_1 regularization. 29

A.9 MNIST test dataset classification error in RBM with L_2 regularization. 29

NOTATION AND SYMBOLS

E	Energy function
λ	weight decay
sig	sigmoid function
k	number of sampling steps in Gibbs sampling
σ	standard deviation of Gaussian distribution used for initialization of weight matrix
S	represents whole training set
S'	represents a subset of the training set
η	learning rate
$p(x)$	probability of random variable x; x is a vector
$P(X)$	marginal probability of random variable X, where X is a vector of size 28 x 28
s	number of nodes at visible layer
t	number of nodes at hidden layer
v	represents visible layer
h	represents hidden layer
$E(X)$	total energy of random variable X
Z	potential function or normalization constant
$\mathcal{L}$	likelihood function
l	loss-function
θ	represents all the hyper-parameters
$\mathcal{F}(x)$	free energy of random variable x
e	Euler′s number; approximately equal to 2.718
W	weight matrix of size sXt
b	bias vector at visible layer
c	bias vector at hidden layer
$\mathcal{R}$	real-number
w_{ij}	weight of edge between i^{th} visible node and j^{th} hidden node
Δ	represents hyper-parameter after update
∂	represents partial derivative symbol
K	Laplacian kernel
K'	bi-harmonic 13 stencil kernel
∇	gradient operator
$*$	convolution operator

ACKNOWLEDGEMENTS

I would like to acknowledge Tom Fletcher for all his guidance, discussions, time, and pointing out the various aspects of the already available successful model; Shireen Elhabian for her guidance on spatial prior and its evaluation; and Nelson H. F. Beebee and Karl Schwede provided much-needed support with LaTeX typesetting. I would also like to thank Alex Babu Karatha for accompanying me during image processing and convolution operator understanding. Last but not the least, my family and friends for supporting me in the journey.

CHAPTER 1

INTRODUCTION

In the last few years, many models have been designed using the idea behind the restricted Boltzmann machine (RBM) [8]. Such models have become popular in data analysis and pattern recognition, with various applications including image processing [14, 27, 35], classification [19, 21, 23, 24, 29, 30], feature learning [4, 11, 26], object detection [3, 13, 26], learning patterns [31,32], document representation [16,20,36], movie recommendation [28], and so on. RBMs are a special case of Boltzmann machine [4]. The Boltzmann machine is a type of Markov random field [12] with stochastic processing units. A Boltzmann machine is a network with symmetrically connected neuronlike units that make stochastic decisions about whether to be on or off. Boltzmann machines have a simple learning algorithm that allows them to discover interesting features that represent complex relationships in training data. The structure of a Boltzmann machine is shown in Figure 1.1. Here, the learning is very slow with many layers of feature detectors, but it is fast in RBMs [9] that have a single layer of feature detectors. Many hidden layers can be learned efficiently by composing RBMs with each layer feature activation behaving as input for the next layer.

In Boltzmann machines there are two types of units: one is the visible unit and the other is the hidden unit. RBMs have both types of units arranged in two layers. The visible units constitute the input layer, where the visible unit for each pixel is a digital image or a word representation from a sentence. The hidden layer learns the model dependencies between the observations (features of image pixels and their dependencies, in case of image pixels). They are viewed as nonlinear feature detectors. In the RBM setting, each neuron/unit in one layer is connected to all the units in the other layer. However, there is no connection between neurons in the same layer thus the name ”restricted.” RBM structure is shown in Figure 1.2.

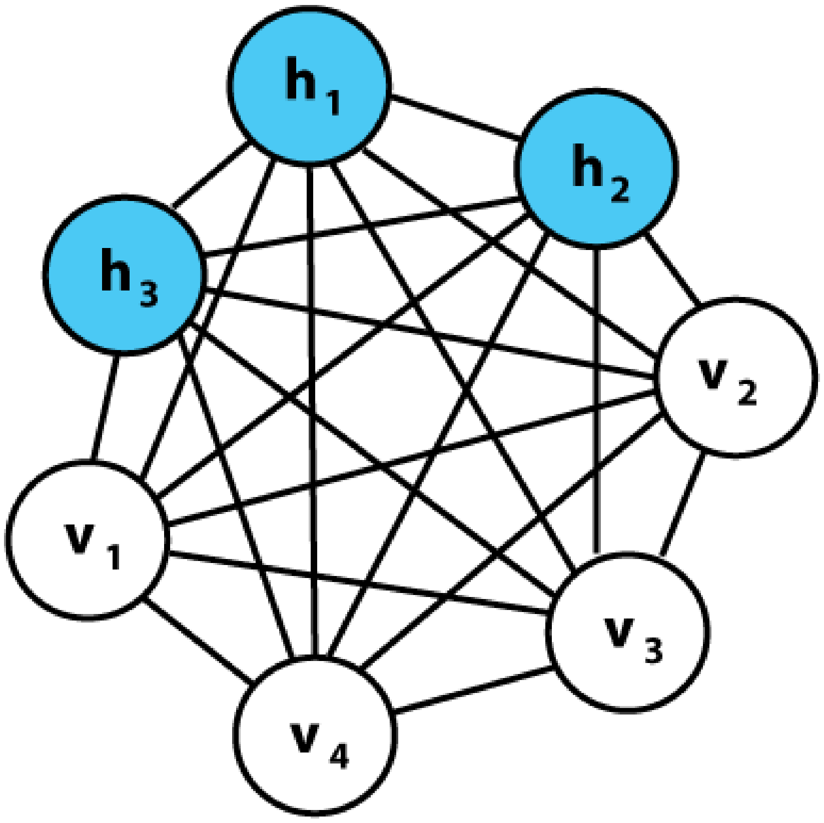


Figure 1.1: Boltzmann machine [5].

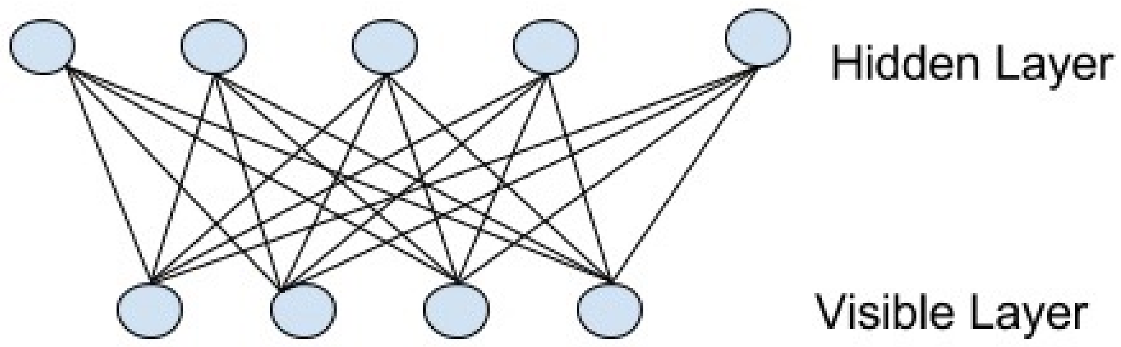


Figure 1.2: restricted Boltzmann machine.

Since the 1980s, RBM design has come a long way and now they are used in more interesting and complex problems due to an increase in computational power and the development of new learning strategies. The learning in the case of RBM is tractable and easily approximated by sampling in comparison to other neural network models. Currently, RBMs are popular as building blocks for deep neural networks, where trained/learned features of the hidden layer of one RBM behave as input to another RBM or another model for detecting more complex features. By stacking RBMs in this way, one can learn features to get more detailed high-level representations. The same deep representation concept makes the RBMs useful in classification and regression using supervised learning algorithms.

These Boltzmann machines can be regarded as probabilistic graphical models, namely undirected graphical models known as Markov random fields. The theoretical journey of graphical models, especially Markov random fields, results in the development of various algorithms. Therefore, in the next chapter we are going to give examples of MRF and graphical models in detail. Training of an RBM is usually based on gradient-based maximization of the likelihood of the RBM. Solving the maximization of likelihood of an undirected graphical model or its gradient is computationally expensive. Thus, sampling-based methods are employed to approximate the gradient of the likelihood. Generally, sampling is not straightforward in the case of undirected graphical models. However, for RBM, the Markov Chain Monte Carlo (MCMC) method of Gibbs sampling is an efficient method for sampling the target distribution. These methods, along with general MCMC theory, will also be discussed in the next chapter. RBMs are energy-based models and share the idea of solving the likelihood maximization. Therefore, it becomes necessary to understand the idea of energy-based models (EBM).

The purpose of this thesis is to apply spatial Markov random field priors during training of RBMs in order to get a better fit for the data. The idea of using an MRF prior comes from the correlation of digital image pixels, which share their properties with their neighbouring pixels. Therefore, we present experiments to compare this prior with other possible priors on RBMs during training with various model settings. The idea of an MRF prior on an RBM is discussed in Chapter 3 and the experimental setup and results are mentioned in Chapter 4 of this thesis.

CHAPTER 2

BACKGROUND

Boltzmann machines are probabilistic graphical models with undirected graphs, also known as Markov random fields. This chapter will give all the required background to understand energy-based models and specifically restricted Boltzmann machines (RBM) in particular. Understanding the concept of a restricted Boltzmann machine and its training requires knowledge of multiple areas. Here, we will describe the required background to understand RBMs, their training and learning, along with general likelihood representations for the model.

2.1 Graphical Models

A probabilistic graphical model (PGM) [22] is a probabilistic model in which a graph represents the conditional dependent or independent structure between random variables, which are represented as nodes of a graph. They encode conditional independence assumptions or factorization of joint probabilities. A directed graphical model is known as a Bayesian network, and an undirected graphical model is called a Markov random field (MRF). Both families encompass the properties of factorization and independence, but they differ in the set of conditional independences they can encode and the factorizations of the distribution that they induce.

2.1.1 Undirected Graphs

An *undirected graph* is an ordered pair $G = (V, E)$, where V represents a finite set of nodes and E represents a set of undirected edges. An edge is a connecting line of a pair of nodes for V. If there is an edge between nodes u and $v; \{u, v\} \in E$, then v belongs to the neighbourhood of u and vice versa. The neighbourhood $\mathcal{N}_u = \{v \in V : \{u, v\} \in E\}$ of u is defined by the set of nodes directly connecting to u. An example of an undirected graph is shown in Figure 2.1, where $\{v_3, v_4, v_6\}$ are the neighbours of node v_5. These connecting edges define dependencies between the random variables represented as nodes in the graph.

A clique is a subset of V in which all nodes are pairwise connected. A clique is maximal if no node can be added such that the resulting set will still be a clique. A set of random variables X is called a Markov random field if the joint probability distribution P fulfills the Markov property with respect to the graph. The property will be fulfilled if a node (representing a random variable) is conditionally independent of all other variables given its neighbours. The probability of x could be written in terms of the nonnegative functions $\{\psi_c\}_{c\in\mathcal{C}}$ where $\mathcal{C}$ is set of all possible maximal cliques, then $P(x)$ will be $P(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}} \psi_c(x)$. The normalization constant $Z = \sum_x \prod_{c\in\mathcal{C}} \psi_c(x)$ is called the partition function. The same becomes $P(x) = \frac{1}{Z}e^{-E(x)}$ with $E = \sum_{c\in\mathcal{C}} \ln \psi_c(x_c)$. This distribution with all nonnegative factors is called a Gibbs distribution.

2.2 Markov Chain and Markov Chain Monte Carlo Techniques

When generating samples from a distribution $P(x)$, sometimes the target distribution does not yield samples. One solution is to construct a Markov chain where the stationary distribution converges to P.

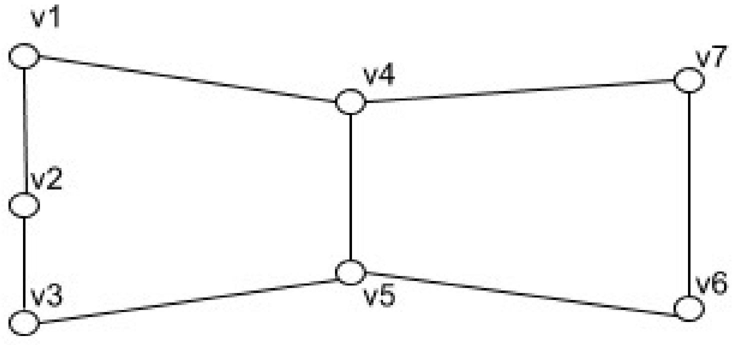


Figure 2.1: Undirected graphical model, with G = (V,E), $V \in \{v_1, \ldots, v_7\}$ where subset $\{v_4, v_5, v_6, v_7\}$ forms a maximal clique and $(v_4 \perp v_2 | v_1, v_5, v_7)$ shows conditional independence.

2.2.1 Markov Chain

A *Markov chain* is a time-discrete stochastic process, where the next state of the system depends only on the current state and not the sequence of events which occurred earlier. Markov chains consist of discrete random variables with a finite set of possible values. Formally, a first order Markov chain is defined as a series of random variables $x^{(1)}, \ldots, x^{(M)}$ such that the following conditional independence property holds for $m \in \{1, \ldots, M-1\}$.

$$P(x^{(m+1)}|x^{(1)}, \ldots, x^{(m)}) = P(x^{(m+1)}|x^{(m)}). \tag{2.1}$$

Equation 2.1 is referred to as a Markov property. We can then specify the Markov chain by giving the probability distribution for the initial variable $P(x^{(0)})$ together with the conditional probabilities for subsequent variables in the form of transition probabilities $T_m(x^{(m)}, x^{(m+1)}) \equiv P(x^{(m+1)}|x^{(m)})$. The Markov chain is called *homogeneous* if the transition probability is the same for all m. If $P(x^{(m+1)}|x^{(m)}) = P$ in the case of a homogeneous Markov chain, and if the starting distribution is $\mu^{(0)}$, the distribution $\mu^{(k)}$ after k discrete times will be given by $\mu^{(k)T} = \mu^{(0)T}P^{(k)}$.

A distribution is said to be invariant or stationary with respect to a Markov chain if each step in the chain leaves that distribution invariant, that is, a distribution π for which $\pi^T = \pi^T P$.

A sufficient condition for ensuring that the required distribution $P(x)$ is invariant is to choose the transition probability that satisfies the condition $\pi(i)P_{ij} = \pi(j)P_{ji}$. This is called the "detailed balance condition." Our goal is to use a Markov chain to sample from a given distribution. We can achieve this if we set up a Markov chain such that the desired distribution is invariant.

2.2.2 Gibbs Sampling

Gibbs Sampling [17] is a simple and widely applicable Markov chain Monte Carlo algorithm that can be seen as a special case of Metropolis-Hastings algorithm [1]. It is a simple MCMC algorithm for producing samples from the joint probability distribution of multiple random variables. The basic idea is to construct a chain by updating each variable based on its conditional distribution given the state of the other random variables.

Consider a Markov random field $x = (x_1, x_2, \ldots, x_N)$ with respect to an undirected graph $G = (V, E)$, where $V \in \{1, 2, \ldots, N)$. As each step of the Gibbs sampling procedure involves replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values from the remaining variables. Thus we replace value x_i by a value drawn from the distribution $P(x_i|x_{-i})$, where x_i denotes the i^{th} component of X and x_{-i} denotes $x_1, \ldots, x_N$ but with x_i omitted. This procedure is followed for each variable to be updated randomly or sequentially from some distribution.

Algorithm 1 Gibbs Sampling

```
1. Initialize $\{x_i : i = 1, \ldots, N\}$.
2. For $\tau = 1, \ldots, T$:
    • sample $x_1^{(\tau+1)}$ from $\tilde{P}(x_1|x_2^{\tau}, x_3^{\tau}, \ldots, x_N^{\tau})$. ‘
    • sample $x_2^{(\tau+1)}$ from $\tilde{P}(x_2|x_1^{(\tau+1)}, x_3^{\tau}, \ldots, x_N^{\tau})$.
    ⋮
    • sample $x_j^{(\tau+1)}$ from $\tilde{P}(x_j|x_1^{(\tau+1)}, \ldots, x_{(j-1)}^{(\tau+1)}, x_{(j+1)}^{\tau}, \ldots, x_N^{\tau})$.
    ⋮
    • sample $x_N^{(\tau+1)}$ from $\tilde{P}(x_N|x_1^{(\tau+1)}, x_2^{(\tau+1)}, \ldots, x_{(N-1)}^{(\tau+1)})$.
```

2.2.3 Metropolis-Hastings Algorithm

Gibbs sampling belongs to the broader class of Metropolis-Hastings algorithms [1]. All MCMC algorithms of this class generate the transition of Markov chain in two steps. In the first step, a candidate state is picked at random from the proposed distribution. In the second step, the candidate state which is the new state of the Markov chain is accepted with an acceptance probability ensuring that a detailed balance holds.

2.3 Energy-Based Models (EBM)

Energy based models [6] capture dependencies between variables by associating a scalar energy to each configuration of the variables. Most probabilistic models can be viewed as special types of energy-based models in which the energy function satisfies certain normalization conditions, and the loss function is the negative log-likelihood that is minimized by learning. Many existing models could be expressed simply through the

Algorithm 2 Metropolis-Hastings Algorithm [1]

```
1. Initialization: pick an initial state x at random;
2. Randomly pick a state $x'$ according to $g(x'|x)$;
3. Accept the state according to $\alpha(x'|x)$, where $\alpha(x'|x) = \min\left(1, \frac{P(x')}{P(x)}\frac{g(x|x')}{g(x'|x)}\right)$;
   If not accepted, transition doesn't take place, and so there is no need to update anything. Else, the system transits to x'
4. Go to 2 until T states were generated;
5. Save the state x, go to 2.
```

framework of energy based models.

According to EBM, there is energy associated with each random variable, higher energy means lower probability. Energy-based probabilistic models define the probabilistic distribution as :

$$P(X) = \frac{e^{-E(X)}}{Z}. \tag{2.2}$$

The normalization factor Z is known as the Partition function by the analogy from the physical systems, with respect to energy-based models.

$$Z = \sum_X e^{-E(X)}.$$

Energy based models can be learned by performing a stochastic gradient descent on the experimental negative log-likelihood of the training data. Optimizing the loss-function with stochastic gradient methods is often more efficient than black box convex optimization methods. The log-likelihood of the data and loss function is defined as:

$$\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x_i \in \mathcal{D}} \log p(x_i). \tag{2.3}$$

$$\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D}). \tag{2.4}$$

In most of the cases, we do not observe the x fully, therefore, to increase the expressiveness of the model, we introduce nonobserved variables (or latent variables) h. Then, we can write EBM with hidden units as,

$$P(x) = \sum_h P(x,h) = \sum_h \frac{e^{-E(x,h)}}{Z}. \tag{2.5}$$

To map this formulation similar to Equation 2.2, we introduce another notation called **Free Energy**, which is defined as follows:

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}. \tag{2.6}$$

Therefore, we can write,

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \quad , \textit{with } Z = \sum_x e^{-\mathcal{F}(x)}. \tag{2.7}$$

The gradient of Equation 2.7 with respect to all the parameters θ of the model can be calculated as described below:

$$\begin{aligned}
\frac{\partial \log p(x)}{\partial \theta} &= -\frac{\partial}{\partial \theta} \log \frac{e^{-\mathcal{F}(x)}}{Z}, \\
&= -\frac{\partial}{\partial \theta} \left(-\mathcal{F}(x) - \log Z\right), \\
&= -\frac{\partial}{\partial \theta} \left(-\mathcal{F}(x) - \log \sum_x e^{-\mathcal{F}(x)}\right), \\
&= \frac{\partial}{\partial \theta}\mathcal{F}(x) - \sum_x \mathcal{F}(\widetilde{x}), \\
&= \frac{\partial}{\partial \theta}\mathcal{F}(x) - \sum_{\widetilde{x}} p(\widetilde{x}) \frac{\partial}{\partial \theta}\mathcal{F}(\widetilde{x}), \\
&= \frac{\partial}{\partial \theta}\mathcal{F}(x) - \sum_{\widetilde{x}} E_p \left[\frac{\partial}{\partial \theta}\mathcal{F}(x)\right], \\
\frac{\partial \log p(x)}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathcal{F}(x) - \frac{1}{|\mathcal{N}|} \sum_{\widetilde{x} \in \mathcal{N}} \frac{\partial}{\partial \theta}\mathcal{F}(\widetilde{x}).
\end{aligned} \tag{2.8}$$

Note that the above gradient contains two terms, which are referred to as positive and negative phases in Equation 2.8 [8]. The first term increases the probability of training data while the second term decreases the probability of samples generated by the model. This is why they are named positive and negative phases –not because of their sign in the above Equation 2.8.

Samples used to estimate the negative phase gradient are referred as negative particles and here referred as $\mathcal{N}$. The samples make the expectation over all possible configurations of input x tractable. These samples are sampled according to P (i.e., using MCMC algorithm). With the above equation we have a stochastic algorithm for learning an EBM. Markov chain Monte Carlo methods for sampling are well suited for models such as RBM.

2.4 Restricted Boltzmann Machine

An RBM is a Markov random field associated with a bipartite undirected graph as shown in Figure 1.2. It consists of s visible units $V = (v_1, v_2, \ldots, v_s)$, which represent the input observed data, and t hidden units $H = (h_1, h_2, \ldots, h_t)$ to capture the dependencies between the observed data, that is, the combination of features learned by the model. Binary RBMs (V, H) take values $(v, h) \in \{0, 1\}^{s+t}$, and the joint probability distribution under the RBM model is given by

$$P(v,h) = \frac{e^{-E(v,h)}}{Z},$$

with energy function defined as:

$$E(v,h) = -b'v - c'h - h'Wv, \tag{2.9}$$

where W is a matrix of real valued weights of size $s \times t$, and b and c are the biases of the visible and hidden units respectively. To map this formula similar to $P(x) = \frac{e^{-E(x)}}{Z}$, we define free energy of RBM and it is given as:

$$\mathcal{F}(v) = \sum_h E(v,h),$$

$$= -b'v - \sum_i log \sum_{h_i} e^{h_i(c_i + W_i v)}.$$

$$P(v,h) = \frac{e^{-\mathcal{F}(v)}}{Z}.$$

As we have already stated in Chapter 1, an RBM has connections only between the layer of visible variables and the layer of hidden variables, but not between the nodes of the same layer. In the probabilistic context, this means that the variables at the hidden layer are independent given the state of the visible units, and vice versa.

$$P(h|v) = \prod_{i=1}^{t} P(h_i|v) \tag{2.10}$$

$$P(v|h) = \prod_{j=1}^{s} P(v_j|h) \tag{2.11}$$

The conditional independence between the variables in the same layer makes the Gibbs sampling easy. Instead of sampling new values for all variables one by one, the state of all variables in one layer can be sampled jointly. Thus, we can perform Gibbs sampling in 2 steps: first, sample the new state of hidden variables given observed data $P(h|v)$ as described in Equation 2.10 and second, sample the new state of visible variables given the new state of hidden variables $P(v|h)$ as in Equation 2.11. This is referred to as block Gibbs sampling [14].

2.4.1 Neural Network With Binary Units

The conditional probability that a single unit equals one can be interpreted as a sigmoid activation function $sig(x) = \frac{1}{1-e^{-x}}$ because

$$P(h_i = 1|v) = sig(c_i + W_i v) \tag{2.12}$$

and

$$P(v_j = 1|h) = sig(b_j + W_j' h). \tag{2.13}$$

Therefore, the RBM is viewed as a deterministic function $\{0,1\}^s \to \mathbb{R}^t$ that maps an input $v \in \{0,1\}^s$ to $y \in \mathbb{R}^t$ with $y_i = P(H_i = 1|v)$. That is, an observation is mapped to the expected value of the hidden neuron given the observation.

2.4.2 Log-Likelihood

As mentioned in the paper [14], the log-likelihood gradient of an MRF can be written in terms of the sum of two expectations as described in Equation 2.14.

$$\frac{\partial log\mathcal{L}(\theta|v)}{\partial\theta} = -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial\theta} + \sum_{v,h} p(v,h)\frac{\partial E(v,h)}{\partial\theta}. \tag{2.14}$$

The first term in Equation 2.14 is the expectation of an energy gradient under the conditional distribution of the hidden variables given a training example v. It can be computed efficiently because it factorizes nicely. For example, with respect to the parameter w_{ij} we get:

$$\sum_h p(h|v)\frac{\partial E(v,h)}{\partial w_{ij}} = \sum_h p(h|v)h_i v_j$$

$$= \sum_h \prod_{r=1}^{t} p(h_r|v)h_i v_j = \sum_{h_i}\sum_{h_{-i}} p(h_i|v)p(h_{-i}|v)h_i v_j.$$

After rearranging the equation:

$$= \sum_{h_i} p(h_i|v)h_i v_j \sum_{h_{-i}} p(h_{-i}|v) = p(H_i = 1|v)v_j.$$

$$\sum_h p(h|v)\frac{\partial E(v,h)}{\partial w_{ij}} = sig\left(\sum_{j=1}^{s} w_{ij}v_j + c_i\right) v_j \, . \tag{2.15}$$

As we know, the joint probability could be written in terms of conditional probability, $p(v,h) = p(v)p(h|v) = p(h)p(v|h)$. Therefore, the second term in Equation 2.14 can be written as $\sum_v p(v)\sum_h p(h|v)\frac{\partial E(v,h)}{\partial\theta}$ or $\sum_h p(h)\sum_v p(v|h)\frac{\partial E(v,h)}{\partial\theta}$. However, the computation remains intractable for regular sized RBMs because its complexity is still exponential in the size of the smallest layer (the outer sum still runs over either 2^s or 2^t states).

Using the simplified derivation of the first term obtained above in the log-likelihood equation with respect to the weight w_{ij} from Equation 2.15, the log-likelihood becomes:

$$\begin{aligned}\frac{\partial log\mathcal{L}(\theta|v)}{\partial w_{ij}} &= p(H_i = 1|v)v_j - \sum_v p(v)\sum_h p(h|v)h_i v_j, \\ &= p(H_i = 1|v)v_j - \sum_v p(v)p(H_i = 1|v)v_j.\end{aligned}$$

This gives the often stated rule:

$$\sum_{v\in S}\frac{\partial log\mathcal{L}(\theta|v)}{\partial w_{ij}} \propto \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}. \tag{2.16}$$

In the same way, we can obtain the derivatives with respect to the bias parameter b_j of the j^{th} visible variable:

$$\frac{\partial log\mathcal{L}(\theta|v)}{\partial b_j} = v_j - \sum_v p(v)v_j. \tag{2.17}$$

And with respect to the bias parameter c_i of the i^{th} hidden variable:

$$\frac{\partial log\mathcal{L}(\theta|v)}{\partial c_i} = p(H_i = 1|v) - \sum_v p(v)p(H_i = 1|v). \tag{2.18}$$

The second term in the log-likelihood gradient with respect to model parameters is approximated by calculating the expectation of samples using the model distribution. These samples are generated from the model using the Gibbs sampling as described in Figure 2.2, which requires running the Markov chain long enough to ensure convergence. Therefore, we need a method to reduce the computational cost of the Monte Carlo Markov chain using other efficient algorithms.

2.4.3 Algorithms to Approximate Log-Likelihood Gradients

For training RBM, we need to have some method for approximating the estimation of likelihood with model distribution. All training algorithms for RBMs approximate

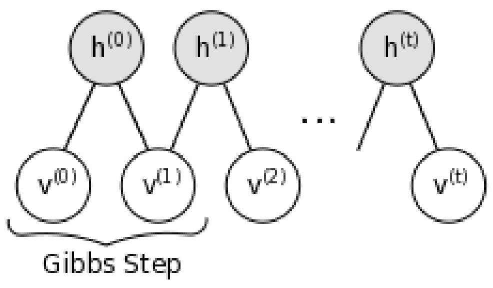


Figure 2.2: Gibbs sampling in RBM (referenced from [8]).

the log-likelihood gradient given some data and perform gradient descent on these approximations. Some of those algorithms for approximating log-likelihood gradients are described here, starting with contrastive divergence learning.

2.4.3.1 Contrastive Divergence

To calculate expectation of the samples obtained from the model distribution, first we need to generate samples. To obtain an unbiased estimate of a log-likelihood gradient using MCMC typically requires many sampling steps. However, it has been shown that this estimate can be obtained after running the chain for just a few steps of the model training. This learning algorithm is known as contrastive divergence (CD) [18]. This algorithm is the standard way of training RBMs.

The idea of CD is simple, instead of approximating the model term in a log-likelihood gradient with a sample from the distribution (which is obtained by running a Markov chain and sampling until the convergence condition is not met), we run a Gibbs chain for k-steps (usually k = 1 works [18]).

Initialize the chain with sample $v^{(0)}$ and run the chain for k-steps and obtain sample $v^{(k)}$; each step consists of obtaining the h using $p(h|v^{(t)})$ and then $v^{(r+1)}$ from $p(v|h^{(r)})$ as defined in Equation 2.12 and 2.13. Then the gradient of log-likelihood with respect to all the parameters θ for one training example $v^{(0)}$ is then approximated using Equation 2.16, 2.17 and 2.18.

This operation for each training sample is costly in terms of computational burden. Therefore, the training is done in batches of a fixed size, each batch is called mini-batch, which is the subset of the overall training data. This process is efficient because it reduces the computational burden between parameter updates.

Algorithm 3 k-step contrastive divergence

```
1: for $v \in S'$ do
2:     $v^{(0)} \leftarrow v$
3:     for $r = 0, \dots, k-1$ do
4:         for $i = 1, \dots, t$ do
5:             sample $h_i^{(r)} \sim p(h_i|v^{(r)})$
6:             ;
7:         end for
8:         for $j = 1, \dots, s$ do
9:             sample $v_j^{(r+1)} \sim p(v_j|h^{(r)})$
10:            ;
11:        end for
12:    end for
13:    for $i = 1, \dots, t, j = 1, \dots, s$ do
14:        $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1|v^{(0)}).v_j^{(0)} - p(H_i = 1|v^{(k)}).v_j^{(k)}$
15:    end for
16:    for $i = 1, \dots, t$ do
17:        $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1|v^{(0)} - p(H_i = 1|v^{(k)})$
18:    end for
19:    for $j = 1, \dots, s$ do
20:        $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
21:    end for
22: end for
```

Usually, the stationary distribution is not always reached after k-sampling steps. This $v^{(k)}$ is not a sample from the model distribution, and the equation of the gradient update step in CD is biased, but the bias vanishes as $k \to \infty$. CD-1 is at present the most commonly used algorithm for RBM training.

2.4.3.2 Persistent Contrastive Divergence (PCD)

Generally CD-n is preferred over CD-1, if enough running time is available. Our goal here is to generate samples from model distribution to approximate the expectation. The standard way is to run a Markov chain, but running a Markov chain for many steps is time consuming. However, in between the model changes slightly. We can take this idea and initialize the Markov chain at the stage in which it ended in the previous step. This initialization will be very close to the model distribution, even though the model has changed slightly in the parameter update. In the case of RBMs, there is only one distribution from which we need samples. Thus, we can use this algorithm for training mini-batches. Using only a few data points from training samples and generating few samples from model distribution to calculate the negative samples. That is, one keeps a "persistent" chain which we run for k-Gibbs steps. After parameter update the initial state of the current Gibbs chain is equal to $v^{(k)}$ from the previous update step.

The PCD algorithm [33] is further studied for potential refinement and the variant is called fast-persistent contrastive divergence (FPCD) [34]. FPCD tries fast mixing of the Gibbs chain by introducing additional parameters w_{ij}^{f}, b_j^{f}, c_i^{f} referred to as fast parameters. This set of parameters is only used in sampling not in the model itself. They place a role when calculating the parameter update during Gibbs sampling. The general parameters now become the sum of the regular parameters and the fast parameter, that is, the Gibbs sampling is based on probabilities:

$$\widetilde{p}(H_i = 1|\mathbf{v}) = sig\left(\sum_{j=1}^{s}(w_{ij} + w_{ij}^{f})v_j + (c_j + c_j^{f})\right),$$

and

$$\widetilde{p}(V_i = 1|\mathbf{h}) = sig\left(\sum_{i=1}^{t}(w_{ij} + w_{ij}^{f})h_i + (b_i + b_i^{f})\right).$$

There is no change in the learning update rule.

There is another promising sampling technique for RBM known as parallel tempering [15] which introduces a supplementary Gibbs chain that samples from increasingly smooth replicas of the original distribution.

CHAPTER 3

MARKOV RANDOM FIELD PRIOR ON RBM

This chapter will formulate the idea of a Markov random field prior on RBM. We will define what an MRF prior is in the context of RBMs and how it will affect the log-likelihood calculation of RBM with a spatial MRF prior.

3.1 Markov Random Field Prior

When we train the RBM model using no prior and plot the filters that are obtained after training on MNIST dataset, we see that they look very noisy, which can be seen in Figure 3.1. That's where the proposed idea in this thesis of a spatial MRF prior comes in. The idea comes from the fact that image pixels share correlation with immediate neighbors, but they are conditionally independent from the pixels which are farther away. RBM model weights can be visualized as a stack of images size $28x28$ as shown in Figure 3.2. Although before diving into MRF prior we have evaluated our RBM model with respect to L_1 and L_2 regularization on weights for smoothing of images filters.

The distribution of $p(W)$ is a Gaussian distribution, where mean of the distribution is the average of the value learned from its neighbors.

$$p(W) \propto \exp\left(-\lambda \sum_{i,j}\left(w_{i,j} - \frac{1}{4}\left(w_{i-1,j} + w_{i+1,j} + w_{i,j-1} + w_{i,j+1}\right)\right)^2\right).$$

The effect mentioned above can be obtained by applying the convolution operator on weight filter and Laplacian kernel. The Laplacian $L(x, y)$ of an image with pixel intensity values $I(x, y)$ is given by:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}.$$

Since the input weight filter is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian, and that kernel is:

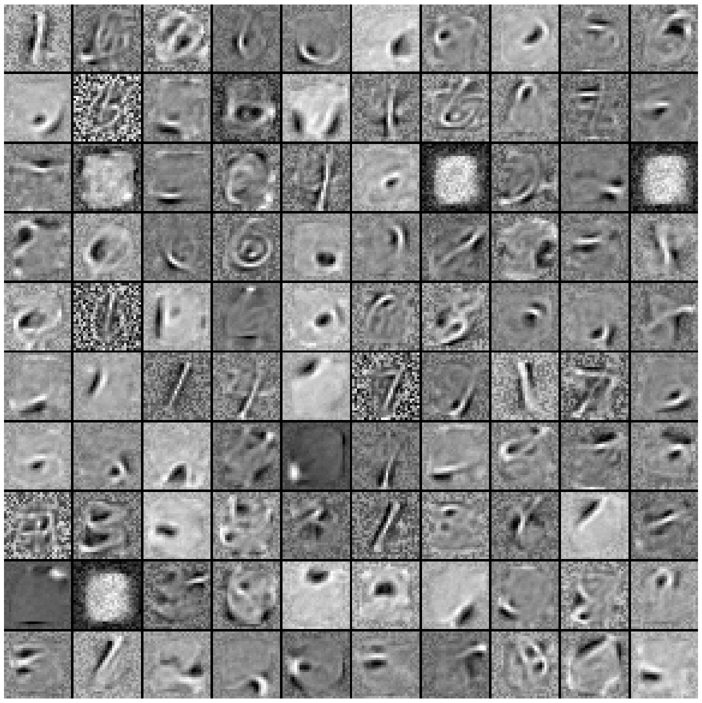


Figure 3.1: RBM weights leaned without any regularization with $h = 500$ after 15 epochs of the training set.

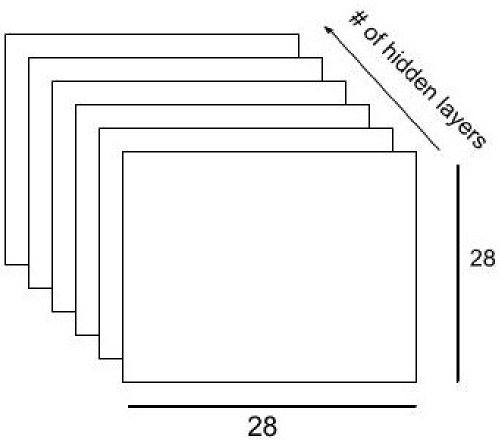


Figure 3.2: RBM weights.

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3.1.1 Evaluating Gradient of Log-Likelihood

The log-likelihood of RBM will incorporate one more term corresponding to a spatial MRF prior. Therefore, the log-likelihood could be written as:

$$\frac{\partial}{\partial\theta}\log\mathcal{L}(\theta|v) = -\sum_h p(h|v)\frac{\partial}{\partial\theta}E(v.h) + \sum_{v,h} p(v,h)\frac{\partial}{\partial\theta}E(v,h) - \frac{\partial}{\partial\theta}\lambda||\nabla W||^2. \tag{3.1}$$

Here, $\theta = \{W, b, c\}$; the parameters of the RBM model. Where the Equation 3.1 will have a last term 0 when the gradient will be taken with respect to b & c, that is, the parameter update for b & c will remain unchanged from general RBM with no prior to the RBM with spatial MRF prior. The gradient update change will occur only when trying to find the optimized W.

Equation 3.1, after writing in terms of convolution with Laplacian kernel results in:

$$\frac{\partial}{\partial\theta}\log\mathcal{L}(\theta|v) = -\sum_h p(h|v)\frac{\partial}{\partial\theta}E(v.h) + \sum_{v,h} p(v,h)\frac{\partial}{\partial\theta}E(v,h) - \frac{\partial}{\partial\theta}\lambda||W * K||^2. \tag{3.2}$$

Gradient of the log-likelihood with respect to W mentioned in Equation 3.2 will become:

$$\frac{\partial}{\partial W}\log\mathcal{L}(\theta|v) = -\sum_h p(h|v)\frac{\partial}{\partial W}E(v.h) + \sum_{v,h} p(v,h)\frac{\partial}{\partial W}E(v,h) - 2\lambda W * K * K. \tag{3.3}$$

The above Equation 3.3 in gradient of spatial MRF prior will become biharmonic with discrete random variables. The classical 13-point stencil for the biharmonic operator is easily derived by applying the standard 5-point Laplacian operator twice, as in Equation 3.4 [7]. Therefore, the second order gradient of spatial MRF prior with respect to W will become equivalent to:

$$\begin{aligned} \nabla||\nabla w_{ij}||^2 \propto 2[&20w_{ij} - 8(w_{i+1,j} + w_{i-1,j} + w_{i,j+1} + w_{i,j-1}) \\ &+ 2(w_{i+1,j+1} + w_{i+1,j-1} + w_{i-1,j+1} + w_{i-1,j-1}) \\ &+ (w_{i+2,j} + w_{i-2,j} + w_{i,j+2} + w_{i,j-2})]. \end{aligned} \tag{3.4}$$

Equation 3.4 could be simplified and optimized by applying a convolution of weight filter with biharmonic discrete kernel.

$$K' = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & -8 & 2 & 0 \\ 1 & -8 & 20 & -8 & 1 \\ 0 & 2 & -8 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Therefore the resultant gradient of log-likelihood with respect to W, will become :

$$\frac{\partial}{\partial W} \log \mathcal{L}(\theta|v) = -\sum_{h} p(h|v) \frac{\partial}{\partial W} E(v.h) + \sum_{v,h} p(v,h) \frac{\partial}{\partial W} E(v,h) - \lambda'(W * K'). \tag{3.5}$$

3.2 Learning Algorithm

Here, we will be merging all the pieces together to show one single pseudo learning algorithm which we have used for RBM training. Algorithm 4 provides top level steps for the RBM learning procedure, with costs calculated by optimizing the log-likelihood gradient to get the optimized parameters and finally get the cost as defined in Equation 3.6:

$$\log PL(x) \sim N * \log[sig(Free\_energy(\widetilde{x}) - Free\_energy(x))]. \tag{3.6}$$

Algorithm 4 Restricted Boltzmann Machine Learning Algorithm [8]

```
1: Load dataset, divide into Training, Test and Development set.
2: Initialize a persistent chain with 0's of size ($\#batch\_size, \#hidden$);
3: Initialize model parameters with small uniform values for weights $W \in (-4 * \frac{6}{\#hidden+\#visible}, 4 * \frac{6}{\#hidden+\#visible})$ and biases with 0 vector of size # of hidden and # of visible units ;
4: for $t = 1, \ldots, T$ : do
5:     for $u = 1, \ldots,$ n_batches : do
6:         Get mean cost by running Contrastive Divergence (CD) or Persistent Contrastive Divergence (PCD) mentioned earlier in Section 2.4.3.1 and 2.4.3.2 respectively until convergence;
7:         ;
8:     end for
9: end for
10: Represent weight filters learned after each epoch;
```

The detailed description of the parameters and other experiments are provided in Chapter 4 with various results and model settings.

CHAPTER 4

EXPERIMENTS

This section will present the experiments illustrating the RBM in practice when various priors are applied during training. After the experimental setup, we will show the learned weights by RBM model after a certain number of epochs with respect to regularization and their variance. All the experiments in this thesis are implemented using the open-source library Theano [2] and over the top of already available code for RBM training with binary units [8].

4.1 Experimental Setup

The experiments are performed on a publicly available MNIST digital hand written dataset [25], where each image is a $28x28$ gray pixel image, binarized with a threshold value of 127. The training set of MNIST consists of 60,000 samples of digital images, out of which 10,000 were kept as a development set. The test set consists of separate 10,000 digital images which are never used or seen during training.

For training, the RBM is initialized with small uniform random weights [10] and zero bias parameters for both hidden and visible layers. The models were trained using $CD-15$ or $PCD-15$, with various sizes of model varying the number of hidden units h; $h \in \{100, 200, 500, 1000, 2000\}$. Different priors are applied to verify the behaviour of RBM such as, L_1 regularization, L_2 regularization, and spatial MRF prior for which we have tried different weight decay parameters λ; $\lambda \in \{0.00001,\ 0.0001,\ 0.001,\ 0.01,\ 0.1\}$. The learning rate was kept fixed $\eta = 0.1$ for the training model to compare the results on common ground. To keep the number of hyper-parameters low, we did not use any momentum term.

In Figure 4.1, we can see the effect of increasing the size of RBM along with the misclassification error rate on the test dataset. It shows that after a while the increase in size won't

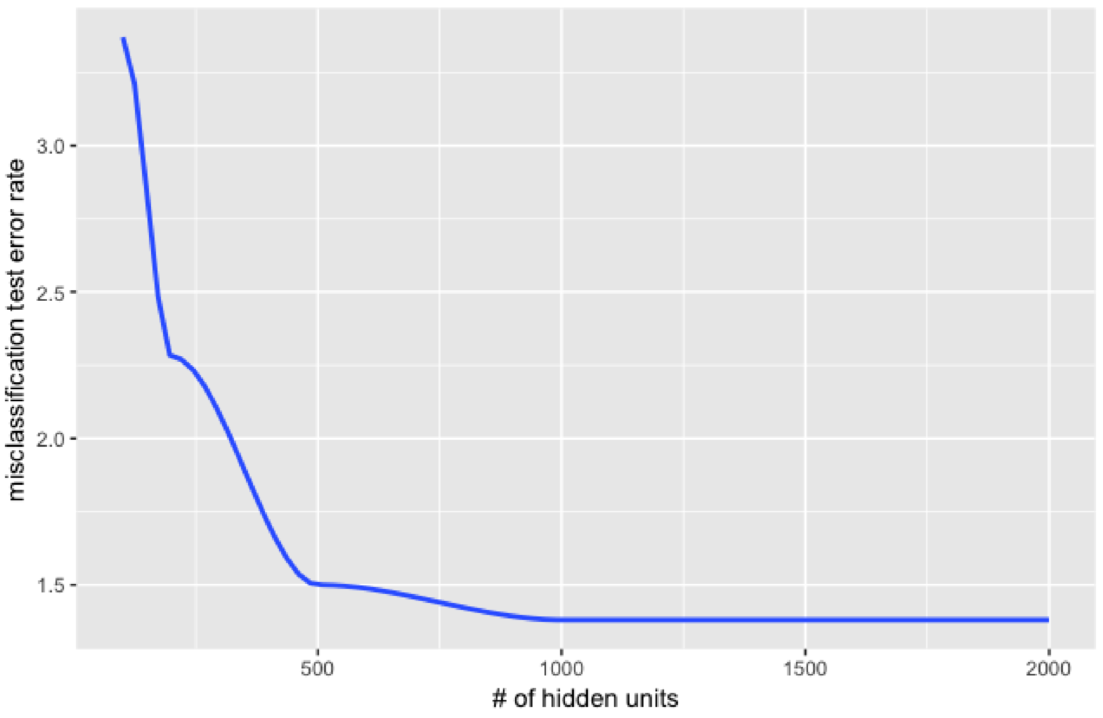


Figure 4.1: Classification error on test dataset in traditional RBM with respect to increase in hidden units.

change the classification much and model starts to overfit the MNIST data. Figure 4.2 shows the effect of various priors on RBM. The lowest misclassification error was achieved using RBM of size $h = 1000$ with spatial prior. The reason for this is shown later in terms of features learned during training. We can see the features in Figure 4.3.

4.2 Classification of MNIST Dataset

One way to classify the MNIST dataset is to use the RBM weights to initialize a feed-forward neural network augmented with an output layer corresponding to the 10 possible label classes from $(0 - 9)$, which can then be fine-tuned in a supervised fashion for classification. Another way could be to use the trained state of the hidden unit as an input layer to the logistic regression layer on top of it with its weights learned during training; given class probability it can further be fine-tuned using back-propagation, as in the supervised manner.

Here, we have used the second approach and are able to report some cool testing evaluations for the model using the logistic layer on top of the RBM with the learned

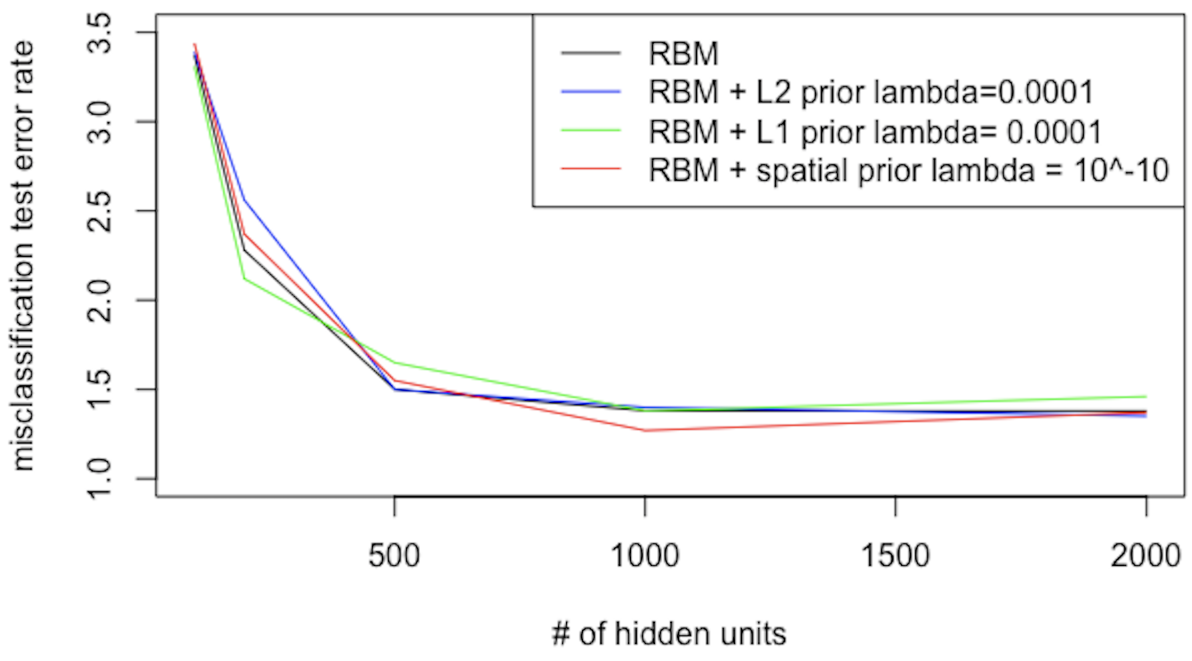
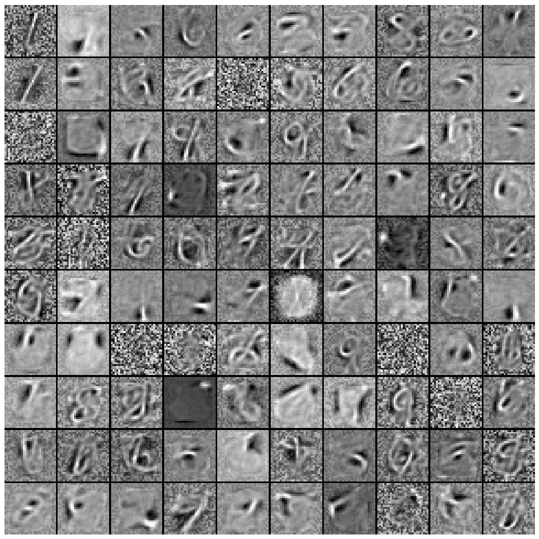


Figure 4.2: Classification error on test dataset.

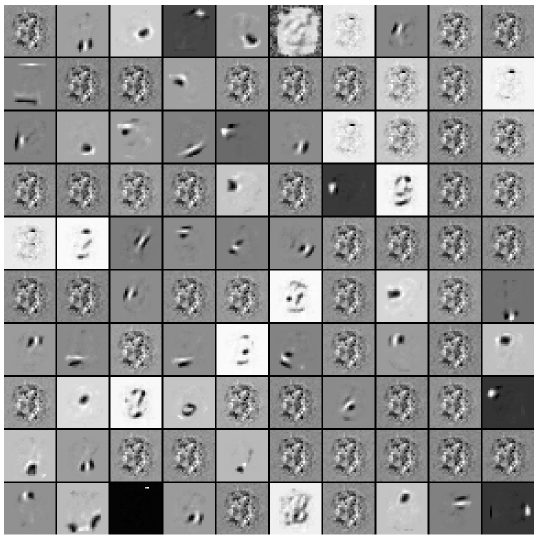
hidden layer and reporting various test set errors on MNIST dataset using the model. The classification model is shown in Figure 4.4. All the results are posted in Appendix A.

The results from Table 4.1 show that the RBM trained with spatial MRF prior can give the best classification model on the MNIST dataset with a gain of almost 1% over traditional RBM (with no prior) or RBM trained using L_1 and L_2 prior, but in all of these cases the filters learned were not that sharp and clear. The results from the spatial MRF prior are more reliable as they can learn small changes in the spatial layout of the image and use that information in training. The images in the MNIST dataset are very tiny ($28x28$) and also very noisy. Therefore, we are assuming the model described in this thesis didn't perform as close to the 1% misclassification error rate on the testing dataset.

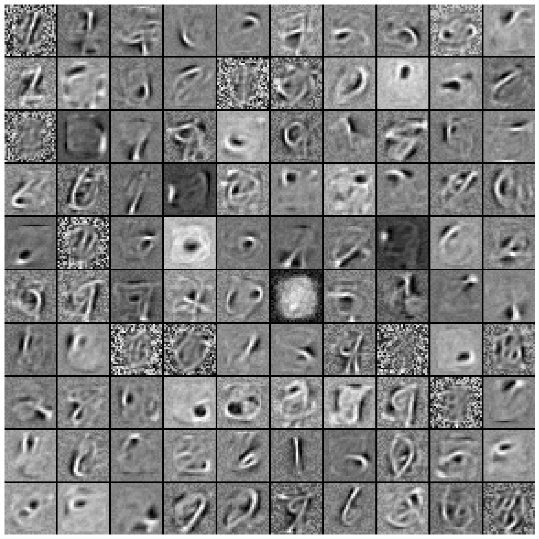
The learned weights for the spatial MRF prior are shown in Figure 4.3, where the filters appear to contain the structure of the digits but the remaining pixels don't have the perfect smooth area surrounding the digit structure. As you can see in Table 4.1, the weight decay parameter for the case of spatial MRF prior is very small as compared to the other regularizer, that is because the gradient of the spatial MRF prior consists of a convolution of weights with biharmonic kernels, which have large coefficient values. Therefore, the



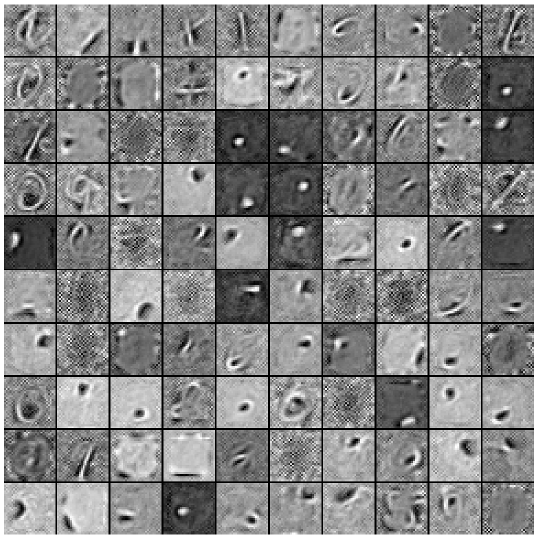
(a) RBM with no prior



(b) RBM with L_1 regularization



(c) RBM with L_2 regularization



(d) RBM with spatial MRF prior

Figure 4.3: Result of 10*10 matrix of filters learned with various models during their best results with $h = 1000$.

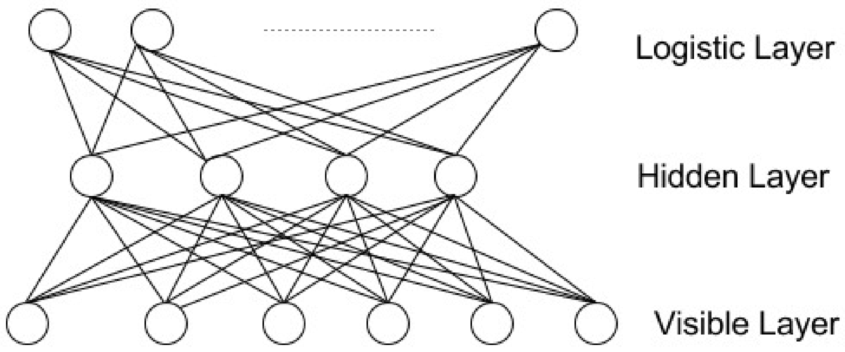


Figure 4.4: Learned hidden layer behaves as input to the logistic layer for classification of MNIST handwritten digits with different set of weights and biases for connection between the hidden layer of RBM and the logistic layer for classification.

Table 4.1: MNIST test classification error.

Number of Hidden units $\rightarrow$ Models $\downarrow$	100	200	500	1000	2000
No prior	3.37	2.28	**1.50**	1.38	1.38
L_2 regularization and $\lambda = .0001$	3.39	2.56	**1.50**	1.40	**1.35**
L_2 regularization and $\lambda = .001$	**2.97**	2.27	1.51	1.50	1.44
L_1 regularization and $\lambda = .0001$	3.31	**2.12**	1.65	1.38	1.46
L_1 regularization and $\lambda = .001$	3.21	2.34	1.89	1.84	1.90
Spatial MRF prior and $\lambda = 10^{-7}$	3.54	2.67	2.75	2.41	2.31
Spatial MRF prior and $\lambda = 10^{-8}$	3.23	2.46	1.79	1.30	1.42
Spatial MRF prior and $\lambda = 10^{-9}$	3.04	2.35	1.57	1.39	1.42
Spatial MRF prior and $\lambda = 10^{-10}$	3.44	2.37	1.55	**1.27**	1.37

weight decay parameter has to be small enough to compensate for the effect of convolution with a high valued kernel.

Table 4.2 describes the effect of a random seed in a hyper parameter initialization. We ran some experiments with a random seed defined as a factor of current time converted into seconds and ran the experiments 10 times for all 4 RBM models with regularization variation, that is, with no prior, with L_1 regularization, L_2 regularization, and spatial MRF prior described in this thesis. From the results it is clear that mean of the model described in this paper provides the lowest mean with significant difference compared to the next best model, which is RBM with no prior. Although there is a difference, which we can see in the standard deviation of these models that they have difference at second position after decimal (except RBM with L_2 regularization). The test classification error mean is still

Table 4.2: Experimental analysis of test classification error with respect to random initialization (seed) factor. Experiments are initialized randomly with $h = 1000$, $\eta = 0.1$ and $\lambda = 0.0001$ for L_1 and L_2 regularization, and $\lambda = 10^{-10}$ for spatial MRF prior.

Model	**Mean**	**Std. deviation**
No prior	1.345	$\pm$ 0.053
L_1 regularization	1.416	$\pm$ 0.057
L_2 regularization	1.463	$\pm$ 0.105
Spatial MRF prior	1.284	$\pm$ 0.069

least in the case of RBM with spatial prior. These results are important to note the effect of randomness in the model. For almost all of our model, we have chosen 123 as random seed arbitrarily.

According to Figure 4.3, we have compared various models described and observed during this thesis. The first image contains filters learned from the training of the RBM with no prior. The results have noisy filters with some of the filters being complete blobs of minimum pixel values shown in dark in the image and some contain weird shapes which do not resembles digits. As we move along to the second image in the figure, that is, filters learned from training with L_1 prior, we see that prior tends to smooth out the weight values for the whole filter and gives some kind of sparse effect to them. Therefore, the filters learned from the L_1 prior are not considered as best representing this dataset. The third image in the figure is of weight filters learned after applying the L_2 prior, which, when compared with the weights learned from RBM with no prior, seems to smooth some of the filters but still remains noisy and shows some scope for further improvement. Finally, we see the last image of Figure 4.3. This is filters learned from RBM with spatial MRF prior, the noise is reduced and the filters are smoothed. But as we can see, the weight filters learned using spatial MRF prior tend to describe some checker-board patterns in the filters. These could be reduced by updating the gradient and zeroing out randomly selected pixels or do alternate even/odd position pixel gradient updates only in each iteration. Those ideas will be tried in the future to see whether the model with spatial MRF prior with less of a checker-board effect does a better job classifying the MNIST dataset. Also, we will evolve this model and try some of the deep networks on the same lines as deep Boltzmann machine (DBM), with spatial MRF prior on the bottom most layer of the DBM with remaining layers as RBM with no prior or with L_2

prior. The deep network on the same idea could be interesting to work with and might provide better classification because as a model gets deep, it tends to learn more complex representations of the input data. MNIST dataset might benefit from the deep architecture of the DBM. Another experiment could involve using some other dataset with real world examples to evaluate the network described here. It would be helpful to use data where the image includes thicker objects which tend to show more properties in the example. The digits dataset has very thin digits which might be affecting the performance of the system measured.

APPENDIX A

TEST CLASSIFICATION ERROR RESULTS

In this appendix, we have tried to state the results of test classification errors obtained after training the RBM model with various settings. Various settings such as initialization of model parameters with different seed, initialization of parameter from uniform and random distribution, various priors on the traditional RBM model described in this document and reporting the results based upon all of the above settings along with varied weight decay parameters.

Table A.1: MNIST test dataset classification error in RBM with no prior.

Number of hidden units $\rightarrow$ Weight initialization $\downarrow$	100	200	500	1000
Uniform initialization with seed = 123	3.37	2.28	1.50	1.38
Uniform initialization with seed = 1234	3.39	2.23	1.77	1.47
Gaussian initialization with std = .1	3.51	2.31	1.50	1.50
Gaussian initialization with std = .01	3.34	2.12	1.50	1.43
Gaussian initialization with std = .001	3.36	2.45	1.59	1.41

Table A.2: MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.00001$.

Number of hidden units $\rightarrow$ Weight initialization $\downarrow$	100	200	500	1000
Uniform initialization with seed = 123	3.40	2.32	1.53	1.44
Uniform initialization with seed = 1234	3.40	2.39	1.68	1.60
Gaussian initialization with std = .1	3.41	2.16	1.52	1.39
Gaussian initialization with std = .01	3.32	2.12	1.57	1.47
Gaussian initialization with std = .001	3.54	2.35	1.55	1.36

Table A.3: MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.0001$.

Number of hidden units → Weight initialization ↓	100	200	500	1000
Uniform initialization with seed = 123	3.39	2.56	1.50	1.40
Uniform initialization with seed = 1234	3.51	2.23	1.61	1.58
Gaussian initialization with std = .1	3.48	2.11	1.56	1.45
Gaussian initialization with std = .01	3.40	2.22	1.65	1.55
Gaussian initialization with std = .001	3.59	2.37	1.48	1.32

Table A.4: MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.001$.

Number of hidden units → Weight initialization ↓	100	200	500	1000
Uniform initialization with seed = 123	2.97	2.27	1.51	1.50
Uniform initialization with seed = 1234	3.19	2.11	1.60	1.47
Gaussian initialization with std = .1	3.00	2.41	1.51	1.40
Gaussian initialization with std = .01	2.82	2.24	1.56	1.45
Gaussian initialization with std = .001	3.05	2.07	1.41	1.47

Table A.5: MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.01$.

Number of hidden units → Weight initialization ↓	100	200	500	1000
Uniform initialization with seed = 123	2.41	1.96	1.44	1.63
Uniform initialization with seed = 1234	2.58	1.87	1.48	1.57
Gaussian initialization with std = .1	2.44	1.81	1.66	1.60
Gaussian initialization with std = .01	2.65	1.74	1.64	1.62
Gaussian initialization with std = .001	2.73	1.79	1.65	1.56

Table A.6: MNIST test dataset classification error in RBM with L_2 regularization with $\lambda = 0.1$.

Number of hidden units → Weight initialization ↓	100	200	500	1000
Uniform initialization with seed = 123	2.47	2.00	2.00	1.88
Uniform initialization with seed = 1234	2.34	2.08	1.95	2.15
Gaussian initialization with std = .1	2.37	2.21	2.05	2.04
Gaussian initialization with std = .01	2.53	2.17	1.97	2.05
Gaussian initialization with std = .001	2.38	2.13	1.96	1.92

Table A.7: MNIST test dataset classification error in RBM with spatial MRF prior.

Number of hidden units → Weight decay parameter ↓	100	200	500	1000
$\lambda = 10^{-6}$	8.64	8.14	8.04	8.14
$\lambda = 10^{-7}$	3.54	2.67	2.75	2.41
$\lambda = 10^{-8}$	3.23	2.46	1.79	**1.30**
$\lambda = 10^{-9}$	3.04	2.35	1.57	1.39
$\lambda = 10^{-10}$	3.44	2.37	1.55	**1.27**

Table A.8: MNIST test dataset classification error in RBM with L_1 regularization.

Number of hidden units → Weight decay parameter ↓	100	200	500	1000
$\lambda = 0.0001$	3.31	2.12	1.65	1.38
$\lambda = 0.001$	3.21	2.34	1.89	1.84
$\lambda = 0.01$	2.92	2.17	1.90	2.02
$\lambda = 0.1$	2.24	2.13	2.05	1.84
$\lambda = 1$	2.27	1.86	1.77	1.88
$\lambda = 10$	3.61	2.89	2.76	2.72

Table A.9: MNIST test dataset classification error in RBM with L_2 regularization.

Number of hidden units → Weight decay parameter ↓	100	200	500	1000
$\lambda = 0.0001$	3.39	2.56	1.50	1.40
$\lambda = 0.001$	2.97	2.27	1.51	1.50
$\lambda = 0.01$	2.41	1.96	1.44	1.63
$\lambda = 0.1$	2.47	2.00	2.00	1.88
$\lambda = 1$	2.85	2.86	2.58	2.55
$\lambda = 10$	2.62	2.58	2.43	2.60

APPENDIX B

WEIGHT FILTERS OBTAINED

In this appendix, we have tried to state the results obtained after the training of the RBM model with different settings.

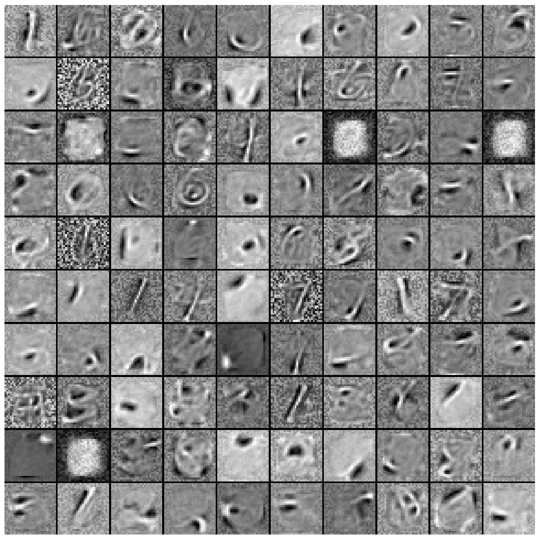
- First set of weight filters are initialized from uniform distribution, which is defined as: $W = U\left(-4 * \frac{6}{\#hidden+\#visible}, 4 * \frac{6}{\#hidden+\#visible}\right)$ and the results are achieved after 15 epochs on training examples (Figure B.1).

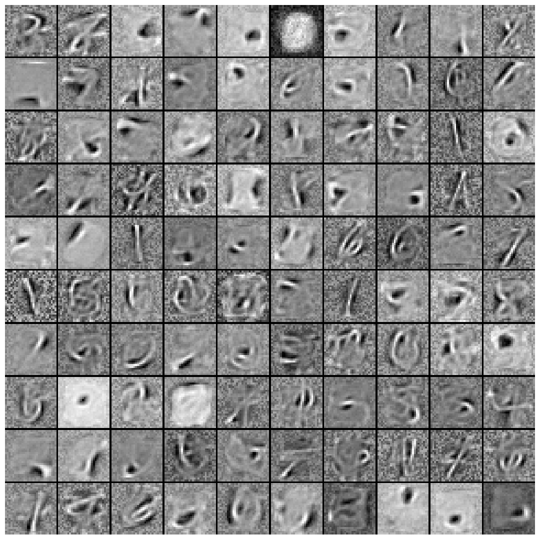
- Second set of weight filters are obtained after applying L_1 regularization on traditional RBM and weights are initialized as previous setting and results are achieved after 15 epochs (Figure B.2).

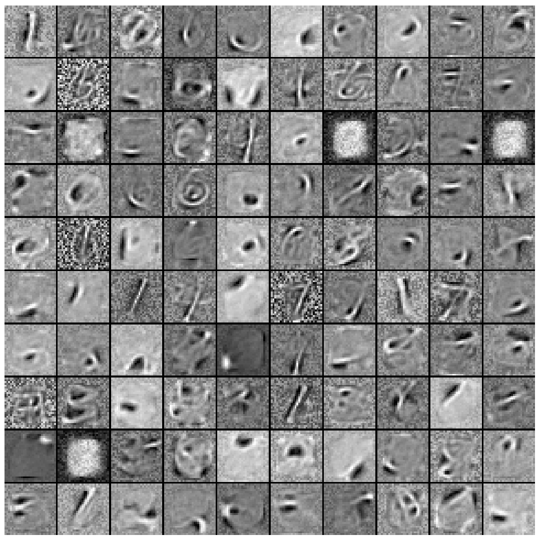
- Third set of weight filters are obtained after applying L_2 regularization on traditional RBM and weights are initialized as previous setting and results are achieved after 15 epochs (Figure B.3).

- Fourth set of weight filters are obtained after applying L_2 prior on traditional RBM with weights initialized by from Gaussian distribution $\mathcal{N}(0, 0.01)$ and with $\lambda = 0.0001$ and the results are achieved after 15 epochs (Figure B.4).

- Final set of weight filters are obtained after applying spatial MRF prior on traditional RBM and weights are initialized as previous setting and results are achieved after 15 epochs (Figure B.5).


(a) # of hidden units =100


(b) # of hidden units =200


(c) # of hidden units =500

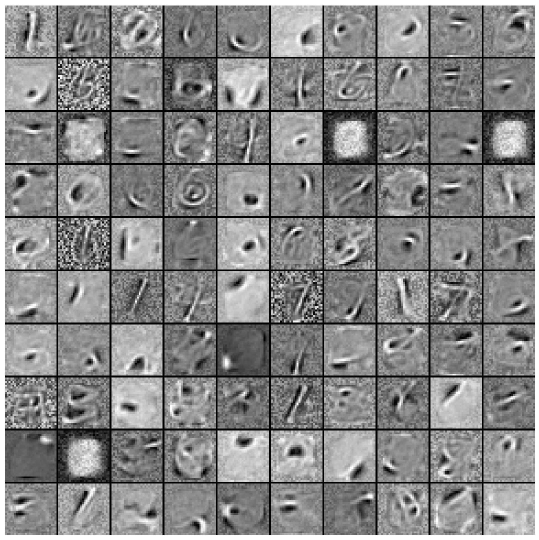
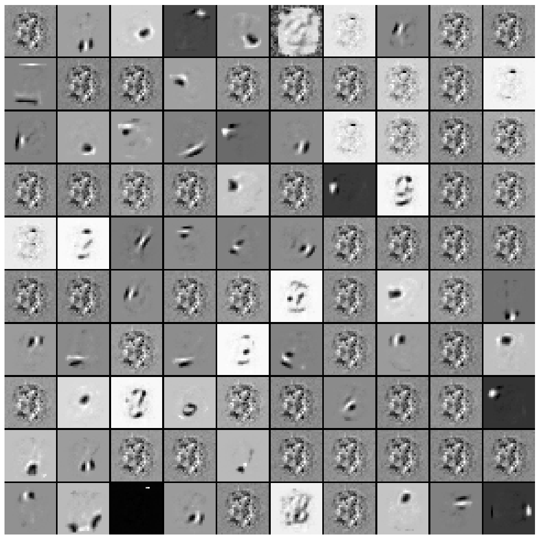
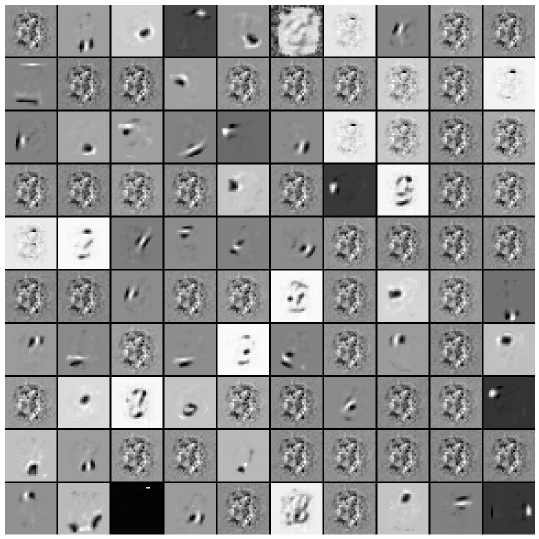
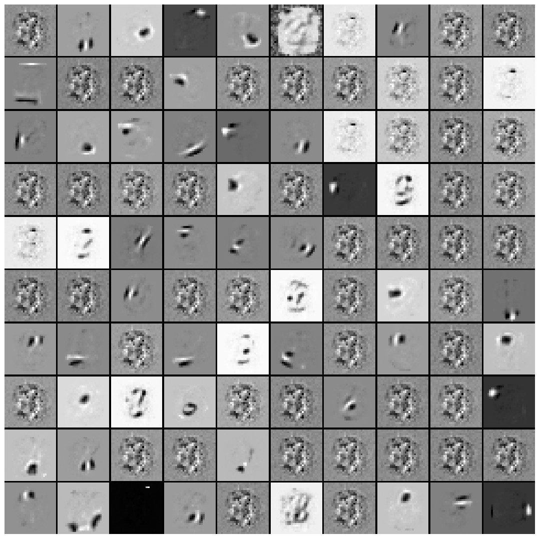

(d) # of hidden units =1000

Figure B.1: Result of 10*10 matrix of filters learned with traditional RBM using various number of hidden units after 15 epochs.


(a) # of hidden units =100


(b) # of hidden units =200


(c) # of hidden units =500

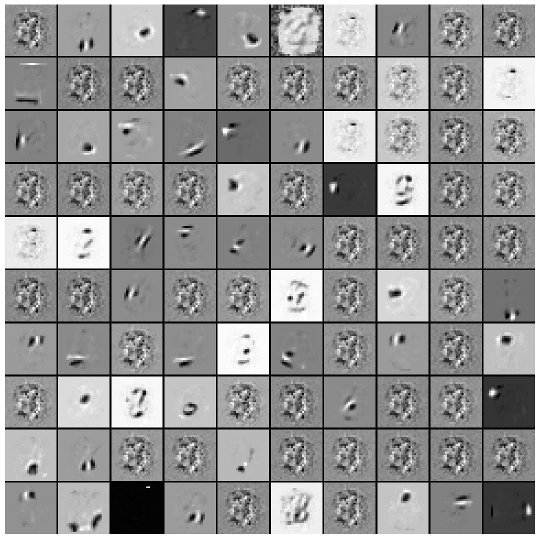
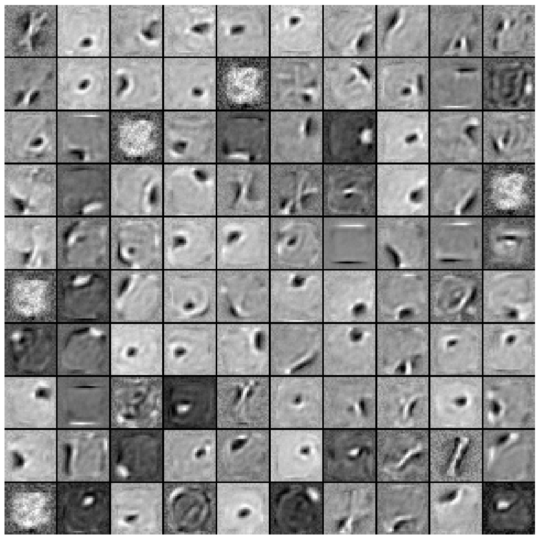
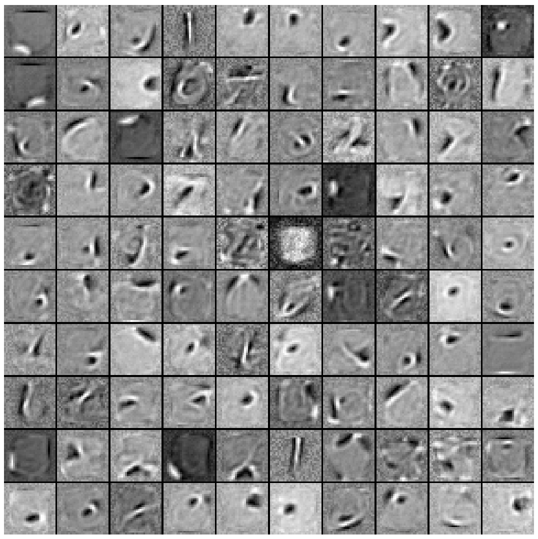
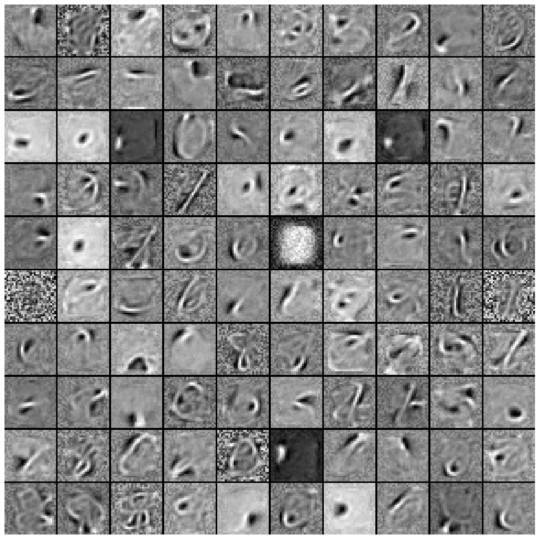

(d) # of hidden units =1000

Figure B.2: Result of 10*10 matrix of filters learned with RBM with L_1 regularization after 15 epochs and $\lambda = 0.001$.


(a) # of hidden units =100


(b) # of hidden units =200


(c) # of hidden units =500

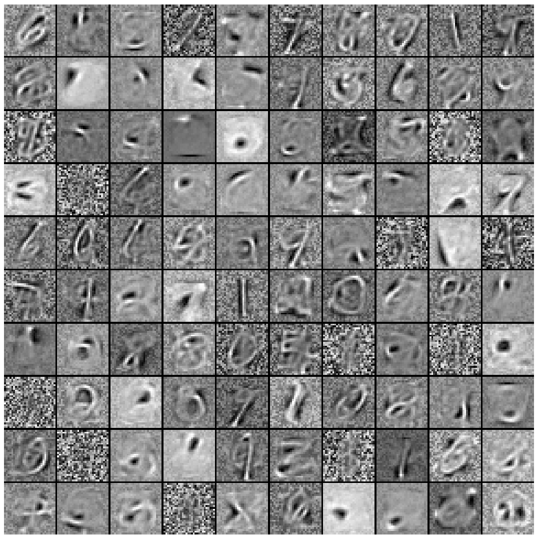
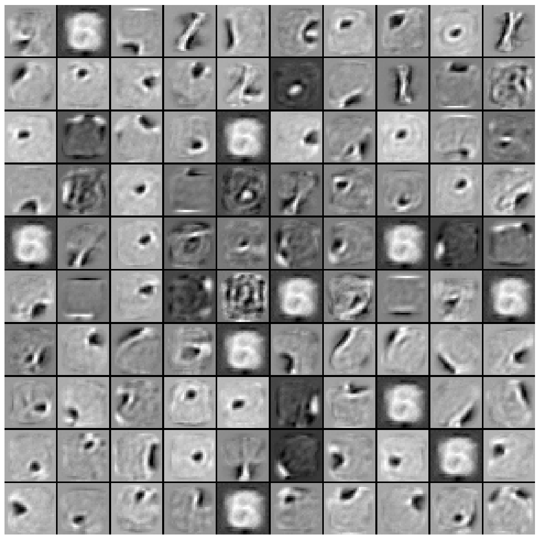
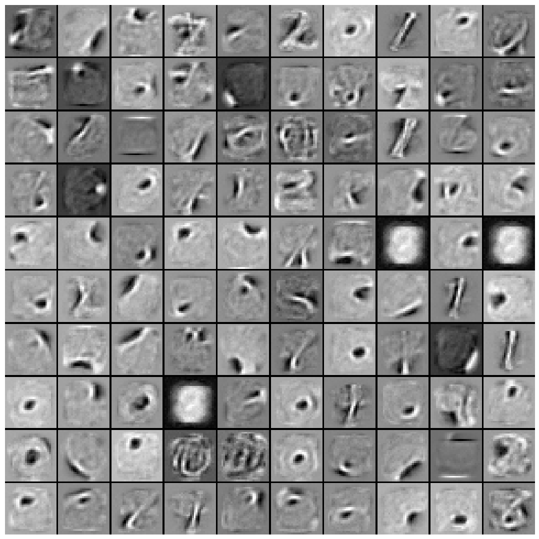
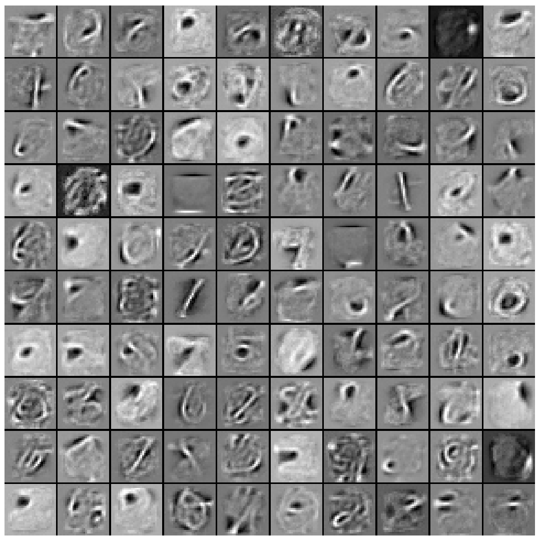

(d) # of hidden units =1000

Figure B.3: Result of 10*10 matrix of filters learned with RBM with L_2 regularization after 15 epochs and $\lambda = 0.00001$, also uniform weight initialization with seed = 1234.


(a) # of hidden units =100


(b) # of hidden units =200


(c) # of hidden units =500

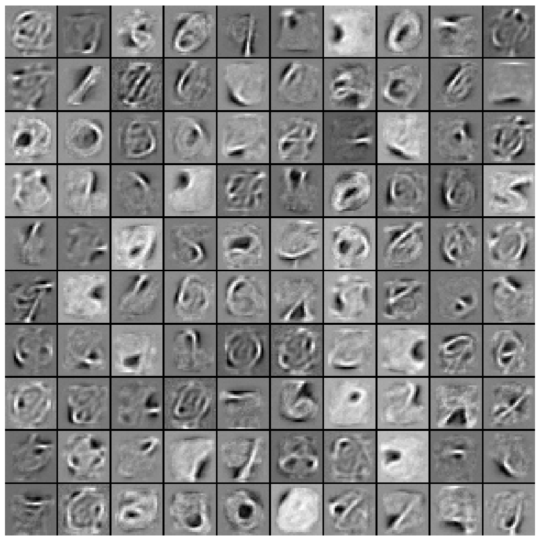
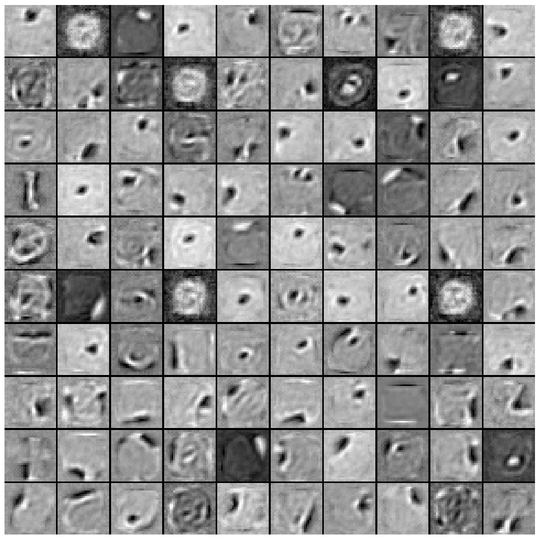
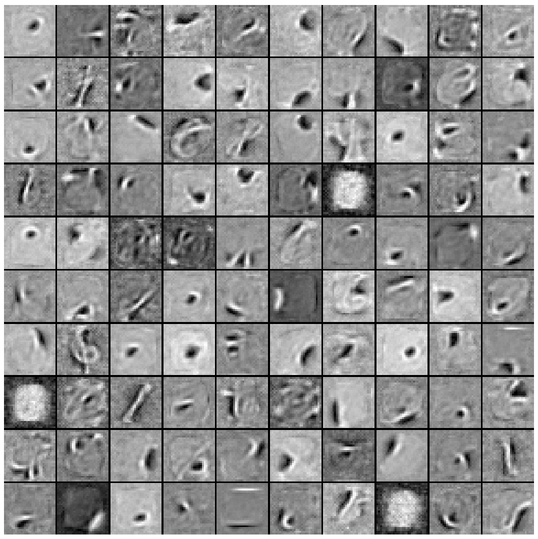
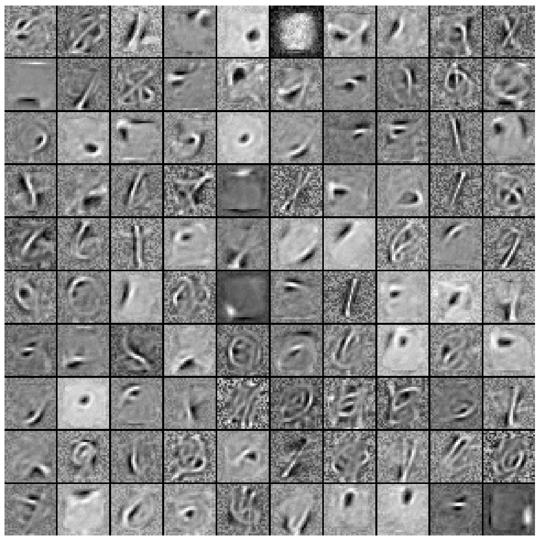

(d) # of hidden units =1000

Figure B.4: Result of 10*10 matrix of filters learned with RBM with L2 prior and Gaussian initialization with std = 0.01 with $\lambda = 0.0001$ and obtained after 15 epochs.


(a) # of hidden units =100


(b) # of hidden units =200


(c) # of hidden units =500

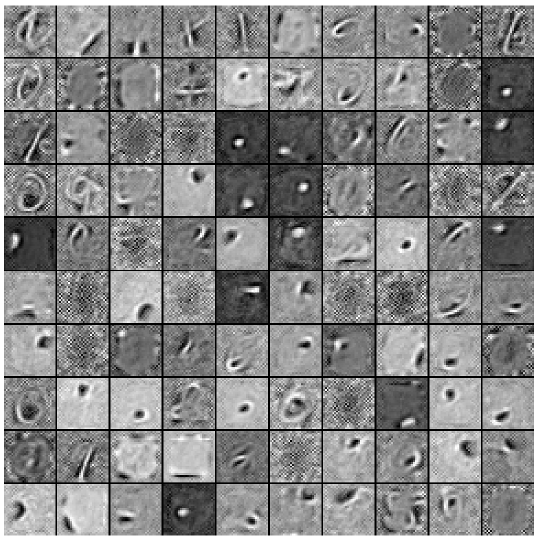

(d) # of hidden units =1000

Figure B.5: Result of 10*10 matrix of filters learned with RBM with spatial MRF prior after 15 epochs and $\lambda = 10^{-10}$ and uniform weight initialization.

REFERENCES

[1] https://en.wikipedia.org/wiki/MetropolisHastings_algorithm.

[2] http://deeplearning.net/software/theano/library/tensor/signal/conv.html.

[3] *Activity recognition using convolutional rbm*. https://s3.amazonaws.com/recpass-production-project_files/project_docs/media/000/002/395/original/project_machine_learning.pdf?1391013472.

[4] *Boltzmann machines*. http://www.scholarpedia.org/article/Boltzmann_machine.

[5] *Boltzmann machines wikipedia*. https://en.wikipedia.org/wiki/Boltzmann_machine.

[6] *Energy based model for bm*. http://www.iro.umontreal.ca/ bengioy/ift6266/H14/ftml-sec5.pdf.

[7] *A fast finite difference method for biharmonic equations on irregular domains*. https://www.ncsu.edu/crsc/reports/ftp/pdf/crsc-tr04-09.pdf.

[8] *Restricted Boltzmann machines*. http://deeplearning.net/tutorial/rbm.html.

[9] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *Connectionist models and their implications: Readings from cognitive science*, Ablex Publishing Corp., Norwood, NJ, USA, 1988, ch. A Learning Algorithm for Boltzmann Machines, pp. 285–307.

[10] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures*, CoRR, abs/1206.5533 (2012).

[11] Y. Bengio, A. Courville, and P. Vincent, *Representation learning: A review and new perspectives*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 1798–1828.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag, Secaucus, NJ, 2006.

[13] S. M. A. Eslami, N. Heess, and J. Winn, *The shape Boltzmann machine: A strong model of object shape*, in IEEE Conference on Computer Vision and Pattern Recognition 2012, 2012.

[14] A. Fischer and C. Igel, *Training restricted Boltzmann machines*, Pattern Recogn., 47 (2014), pp. 25–39.

[15] ——, *A bound for the convergence rate of parallel tempering for sampling restricted Boltzmann machines*, Theor. Comput. Sci., 598 (2015), pp. 102–117.

[16] P. V. Gehler, A. D. Holub, and M. Welling, *The rate adapting Poisson model for information retrieval and object recognition*, in Proceedings of the 23rd International Conference on Machine Learning, ICML '06, New York, NY, USA, 2006, ACM, pp. 337–344.

[17] S. Geman and D. Geman, *Stochastic relaxation, gibbs distributions, and the bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intell., 6 (1984), pp. 721–741.

[18] G. E. Hinton, *A practical guide to training restricted Boltzmann machines*, in Proceedings of Neural Networks: Tricks of the Trade (2nd ed.), 2012, pp. 599–619.

[19] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science, 313 (2006), pp. 504–507.

[20] G. E. Hinton and R. R. Salakhutdinov, *Replicated softmax: An undirected topic model*, in Advances in Neural Information Processing Systems 22, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, eds., Curran Associates Inc., Red Hook, NY, USA, 2009, pp. 1607–1614.

[21] J. Kivinen and C. Williams, *Multiple texture Boltzmann machines*, in Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, N. D. Lawrence and M. Girolami, eds., vol. 22 of Proceedings of Machine Learning Research, La Palma, Canary Islands, 21–23 Apr 2012, PMLR, pp. 638–646.

[22] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*, The MIT Press, Cambridge, MA, USA, 2009.

[23] H. Larochelle and Y. Bengio, *Classification using discriminative restricted Boltzmann machines*, in Proceedings of the 25th International Conference on Machine Learning, ICML '08, New York, NY, USA, 2008, ACM, pp. 536–543.

[24] N. Le Roux, N. Heess, J. Shotton, and J. Winn, *Learning a generative model of images by factoring appearance and shape*, tech. rep., January 2010.

[25] Y. LeCun and C. Cortes, *MNIST handwritten digit database*, (2010).

[26] M. Norouzi, M. Ranjbar, and G. Mori, *Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning.*, in CVPR, IEEE Computer Society, 2009, pp. 2735–2742.

[27] N. Roux, J. Winn, J. Shotton, and N. Heess, *Image processing using masked restricted Boltzmann machines*, Feb. 10 2011. US Patent App. 12/535,178.

[28] R. Salakhutdinov, A. Mnih, and G. Hinton, *Restricted Boltzmann machines for collaborative filtering*, in Proceedings of the 24th International Conference on Machine Learning, ICML '07, New York, NY, USA, 2007, ACM, pp. 791–798.

[29] T. Schmah, G. E. Hinton, S. L. Small, S. Strother, and R. S. Zemel, *Generative versus discriminative training of rbms for classification of fmri images*, in Advances in Neural Information Processing Systems 21, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds., Curran Associates Inc., Red Hook, NY, USA, 2009, pp. 1409–1416.

[30] Y. Tang, R. Salakhutdinov, and G. Hinton, *Robust Boltzmann machines for recognition and denoising*, in 2012 IEEE Conference on Computer Vision and Pattern Recognition, June 2012, pp. 2264–2271.

[31] G. W. Taylor and G. E. Hinton, *Factored conditional restricted Boltzmann machines for modeling motion style*, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, ACM, pp. 1025–1032.

[32] G. W. Taylor, G. E. Hinton, and S. T. Roweis, *Modeling human motion using binary latent variables*, in Advances in Neural Information Processing Systems 19, P. B. Schölkopf, J. C. Platt, and T. Hoffman, eds., MIT Press, Cambridge, MA, USA, 2007, pp. 1345–1352.

[33] T. Tieleman, *Training restricted Boltzmann machines using approximations to the likelihood gradient*, in Proceedings of the 25th International Conference on Machine Learning, ICML '08, New York, NY, USA, 2008, ACM, pp. 1064–1071.

[34] T. Tieleman and G. Hinton, *Using fast weights to improve persistent contrastive divergence*, in Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, New York, NY, USA, 2009, ACM, pp. 1033–1040.

[35] J. Wang, W. Wang, R. Wang, and W. Gao, *Image classification using rbm to encode local descriptors with group sparse learning*, in 2015 IEEE International Conference on Image Processing (ICIP), Sept 2015, pp. 912–916.

[36] E. P. Xing, R. Yan, and A. G. Hauptmann, *Mining associated text and images with dual-wing harmoniums*, CoRR, abs/1207.1423 (2012).