Aalto University
School of Science
Department of Information and Computer Science
Master's Programme in Machine Learning and Data Mining

Tele Hao

Gated Boltzmann Machine in Texture Modeling

Master's thesis

Espoo, 15th October 2012

Supervisor: Prof. Juha Karhunen

Instructor: Tapani Raiko, D.Sc.(Tech.)

Aalto University
School of Science
and Technology

Aalto University School of Science Faculty of Information and Natural Sciences Degree Programme of Computer Science and Engineering Master's Programme in Machine Learning and Data Mining	ABSTRACT OF MASTER'S THESIS

Author:	Tele Hao				
Title:	Gated Boltzmann Machine in Texture Modeling				
Number of pages:	xi + 73	Date:	15th October 2012	Language:	English
Professorship:	Computer and Information Science			Code:	T-61
Supervisor:	Prof. Juha Karhunen				
Instructor:	Tapani Raiko, D.Sc.(Tech.)				

Abstract:

Recently, a new type of neural network method, namely deep learning, was discovered, which yielded excellent results in various tasks such as handwritten digit recognition, speech recognition and facial expression recognition. Different from traditional multi-layer perceptron learning methods (MLP), an unsupervised pre-training step before supervised learning is of huge importance in learning successful features. Also, it is argued that the deep architecture has more expressive power comparing to the conventional shallow networks, such as support vector machine or multi-layer perceptrons.

Even though deep learning has yielded a large amount of world-class records in different tasks, there is little research on how the deep network can be used in texture analysis.

For this particular problem, we consider the problem of modeling complex texture information using undirected probabilistic graphical models. Texture is a special type of data that one can better understand by considering its local structure. For that purpose, we propose a convolutional variant of the Gaussian gated Boltzmann machine (GGBM), inspired by the co-occurrence matrix in traditional texture analysis. We also link the proposed model to a much simpler Gaussian restricted Boltzmann machine where convolutional features are computed as a preprocessing step. The usefulness of the model is illustrated in texture classification and reconstruction experiments.

Keywords: Machine Lerarning, Unsupervised Learning, Deep Learning, Boltzmann Machine, Texture Analysis, Pattern Recognition

Acknowledgement

26th August 2010 - 1st November 2012; over two years' time, has been totally unforgettable for me in the path of chasing my dream. Since the time when I was admitted to MACADAMIA program, the course of my future career changed completely. It's the passion for making the data to speak for themselves drew me to this dream land, and everything happening has been more than amazing.

After two years' extensive studies and research in the department of information and computer science, the MACADAMIA program offered us an excellent platform to attend interesting research and work. I was lucky to work with researchers from Bayes group about the advanced neural network and unsupervised learning algorithms. These research and studies will never be possible without the support and love from my supervisors, instructors, and colleagues.

Prof. Juha Karhunen has not only been a wonderful supervisor for my studies, research and thesis, but also a great mentor for my future career. His guidance enlightens my path to the data analytic communities, and provides me with excellent research resources. Prof. Erkki Oja has been super supportive for my studies and research, and made also my research possible by providing a nice working environment. Dr Tapani Raiko and Dr Alexander Ilin have been more than my instructors. I have always been fascinated by their detailed and excellent mentorships, and nothing will be possible without their patient explanations on those tiny itchy details of the research and studies. Furthermore, I would like to thank Dr. Zhirong Yang for his research collaborations, guidance and friendship. Your vision on the unsupervised learning just blows my mind away.

I would like to particularly thank all my friends encouraging, supporting me and sharing the happiness and sorrows with me. You guys rock! Finally, I can't thank my lovely wife Cana enough. Your love and care just made my world go round.

⋆⋆⋆⋆⋆ Happy data hacking, and happy researching! ⋆⋆⋆⋆⋆

Contents

List of Abbreviation

RBM	Restricted Boltzmann Machine
GRBM	Gaussian Restricted Boltzmann Machine
GGBM	Gated Gaussian Boltzmann Machine
GBM	Gated Boltzmann Machine
CRBM	Conditional Restricted Boltzmann Machine
ICA	Independent Component Analysis
NMF	Non-Negative Matrix Factorization
LBP	Local Binary Pattern
BP	Back propagation
ANN	Artificial Neural Network
MLP	Multi-Layer Perceptron
SIFT	Scale-Invariant Feature Transform
MFCC	Mel-Frequency Cepstral Coefficients
BM	Boltzmann Machine
RP	Random Projection
PT	Parallel Tempering
CD	Contrastive Divergence
GGGBM	General Gated Gaussian Boltzmann machine
CGRBM	Convolutional Gaussian Restricted Boltzmann machine

List of Symbols

$\langle \cdot \rangle_d$	The expectation over the data distribution
$\langle \cdot \rangle_m$	The expectation over the model distribution
$\mathbf{H}$	All hidden neurons obtained from the entire data matrix $\mathbf{X}$
$\mathcal{L}(\mathbf{w})$	The objective function over the parameter vector $\mathbf{w}$
$\text{Cov}_P(\mathbf{x}, \mathbf{h})$	The covariance of the input vector and hidden vector
∇b_i	The gradient information for learning b_i
∇c_j	The gradient information for learning c_j
∇w_{ij}	The gradient information for learning w_{ij}
σ_i	The standard deviation of the real valued input x_i
$\mathbf{h}$	Hidden neurons in Boltzmann machine
$\mathbf{m}$	Neurons in the general Boltzmann machine
$\mathbf{x}$	Input Neurons in Boltzmann machine
$\mathbf{x}, \mathbf{y}$	Neurons in the gated Boltzmann machine
A_{ij}	The directed weight between x_i and y_j
b_i	The bias term for x_i
b_i^*	The dynamic bias terms for x_i in CRBM
$b_j^\star$	The weight for the data y_j
c_k	The bias term for h_k
c_k^*	The dynamic bias terms for h_k in CRBM
C_{ik}	The directed weight between x_i and h_k
$E(\mathbf{x}, \mathbf{h})$	The energy function of Boltzmann machine

$F(\cdot)$ The free energy in an energy model

$P(\mathbf{X})$ The entire data set or data batch for training the Boltzmann machine

$P(\mathbf{X}, \mathbf{H})$ The joint probability distribution over the input vector $\mathbf{x}$ and the hidden variables $\mathbf{h}$

u_d Convolutional bias for the convolutional gated Boltzmann machine

u_{ij} Weight matrix for the interaction between input vectors in GGBM

v_{ik} Weight matrix for the interaction between input and hidden neurons in GGBM

w_{dk} Convolutional weight matrix for the convolutional gated Boltzmann machine

w_{ijk} The interaction weight tensor in GBM

w_{ijk} Weight tensor in the gated Boltzmann Machine

w_{ij} The weight used for characterizing the connection between x_i and h_k

y_j The additional data vector for GBM and CRBM

Z Normalization constant in energy models

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Background and Related Theory

Deep Learning [31] [3] has made a renaissance in the field of neural network research since 2006. It is a new learning paradigm which has resulted in state-of-the-art performances in various learning problems, such as handwritten digit recognition [31], speech recognition [18], facial expression recognition [56], document classification [30], natural image modeling [16], etc.

Prior to deep networks, conventional neural networks were intensively studied since 1980s after the invention of the back-propagation algorithm [57]. Artificial neural network stemmed from information passing paradigm of the biological brains where the information is fed to a sensory input and a series of possible actions are expected as outputs. In order to learn the mapping, a hidden layer resembling the function of brain is defined such that the hidden patterns or information can be extracted from the sensory inputs.

Conventional artificial neural networks [28] originated from a simple network called perceptron, and the stacked layer structure is also called multi-layer perceptron. The connections between layers are directed and the information typically goes from the input to output layer through the hidden layers. Different from other machine learning algorithms such as linear regression, the multi-layer perceptron can process data in a nonlinear distributed way. It was proved in [32] that a multi-layer perceptron network with only one hidden layer can approximate any possible smooth enough mapping function from the input to output with a sufficient amount of hidden units.

Different from the deep network, the conventional multilayer perceptron is regarded as shallow network as the typical number of hidden layers in multilayer perceptron is essentially no more than one or two. Actually, increasing the number of the layers of hidden structure in shallow network hardly improves any performance of the network but introduces more risk of over-fitting and learning effort [4]. A graphical illustration of multi-layer perceptron can be found in Figure 1.1[1]. It shows that the input is fed to a one hidden layer network and optimally the hidden neurons will capture the patterns of the input-output pairs, and

[1]Original version appeared at http://www.texample.net/tikz/examples/neural-network/

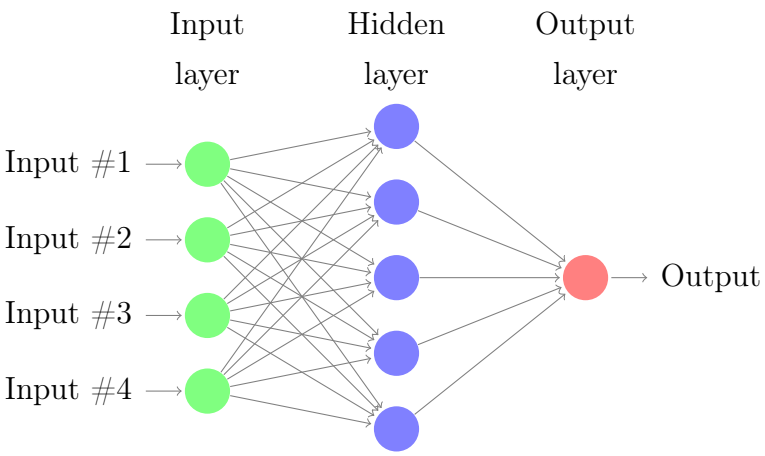


Figure 1.1: The semantic illustration of multi-layer perceptron

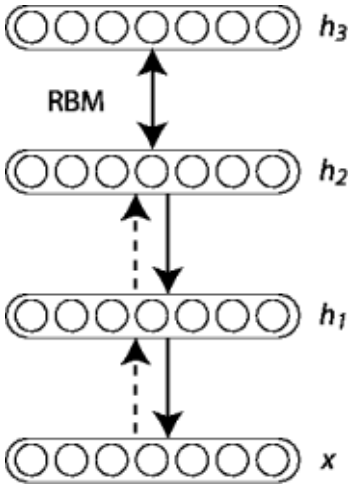


Figure 1.2: The semantic illustration of deep network

also generalize to un-seen new data.

However when people tried to mimic a large-scale deep network using multi-layer perceptron, the traditional back-propagation algorithm failed to learn anything meaningful in a reasonable time. For instance, in a multi-layer perceptron with more than 5 hidden layers, the random initialization of weight matrices and back-propagation is typically not significantly better than a multi-layer perceptron with a single hidden layer. The invention of deep network, utilizing the Boltzmann machine, tackled this problem, and opened up a promising direction in the research of artificial neural networks. For comparison, a graphical illustration of deep network is shown in Figure 1.2. Different from MLP, the deep network has a hierarchical layered structure with multiple intermediate hidden layers. Addition to that, between the layers of hidden layer, bi-directional inferences are present.

The reason that deep learning has attracted a lot of attentions in the field of the neural network and machine learning is that it not only generally outperforms the conventional multilayer perceptron in different tasks, but also surpasses the performance on certain recog-

nition tasks where people had to design various complicated human engineered feature extraction methods, such as speech recognition [18]. It also unified differences between feature extraction methods in different fields [53]. In previous machine learning tasks, people in different fields usually designed different feature extraction methods for their own tasks. For instance, SIFT [43], HOG [19] in computer vision, MFCC [21] [49] in speech recognition and textons [68] in texture analysis. In addition to that, the deep network is argued to have more expressive power comparing to the conventional shallow network [4].

The ideology of deep architecture goes beyond simple recognition tasks. It is also applied to model the image transformation between a pair of images, or a pair of objects from different modalities [47], the internal structure of single image [55], and structured output prediction problem [50]. These problems are extremely hard if one tries to explicitly model the statistical property of the target data manually as there is simply no way of modeling some non-rigid transformation using traditional simple machine learning methods.

1.2 Statistical Approach to Texture Modeling

Texture is a common characteristic of different objects. Texture is often defined as a fixed type of patterns on the object surfaces or contours that resembles distinguishable patterns from other objects. For instance, the bark patterns from trees, the red block patterns from brick walls, etc. The texture information is extremely important because textural information is one of the only patterns that stays constant while the shape, illumination, perspectives of objects changes. For instance, in a construction site, there are hundred of millions of broken pieces of garbage that have to be sorted and utilized. In order to automatize the process, one has to design such a system that the texture information is widely studied as the shape and color of the objects might be corrupted.

From this perspective, a texture is a discriminative feature that can be utilized to perform a range of automatic tasks. To study the benefit of texture, like most of other machine learning problems, the problem is considered using two different approaches: discriminative learning and generative learning. In discriminative texture modeling, the task is to learn the different distinct properties of texture such that the learned model can classify or identify different textures without explicitly having the knowledge about how the texture is formed. On the contrary, the generative learning typically assumes that there is a specific generative pattern for each type of textures, and the learned model is capable of doing more than classification tasks. The generative models can be used to reconstruct damaged textures, and also to generate textures under some circumstances.

Texture modeling has been an important topic for decades in different subfields of machine learning, data mining and computer vision. One of the earliest work concerning texture modeling using machine learning algorithms dates back to 1962 in [33] and was further discussed in [34]. Some actual research interest in texture modeling started with binary image patterns, e.g., see [5] [35]. Soon after the basic research with binary textures, research in the field of texture modeling moved to grey-level pixels texture images. For instances, [26] [25] and [60]. Some further advances have been made in 3D texture model-

ing [17] [37] [41] [59] [66] [67] with the capability of categorizing textures even when there are massive camera pose, illumination, affine transformation changes.

Recent papers regarding texture modeling are published in [42] and [36]. In these two papers, two rather new statistical approaches have been explored in the field of texture modeling, namely the compressed sensing [9] [10] [22] and the deep learning [31][36]. The texture classification using random features shows superior performances comparing to other state-of-the-art methods yet preserving simple computational models. On the other hand, texture modeling using deep network gives possibilities of learning a comprehensive list of different generative models for different texture such that a list of possible actions can be performed if necessary, such as the texture reconstruction, synthesis and recognition.

1.3 Contribution of This Thesis

In this thesis, the possibility of modeling the statistical distribution of texture information using different kinds of Boltzmann machines is explored. Boltzmann machines [1] are typically regarded as the building blocks for the deep network, and different variants of them lead to completely different modeling capabilities. There are enormous amount of publications on deep learning in recent years, but there is little research in the field of texture modeling.

In this thesis, the author considers exploring various texture structures using a convolutional higher order Boltzmann machine. The network tries to consider the pixel interactions in a local area in a way that the network can capture the local interactions within the different texture information. As the proposed network is rather complicated at the first place, an approximated network and its learning rule are also proposed.

1.4 Structure of the Thesis

This thesis consists of several chapters. The background knowledge on conventional neural networks and deep networks is presented in Chapter 2. A extensive review on texture modeling is addressed in Chapter 3. The proposed convolutional higher order Boltzmann machine is discussed in Chapters 4 and 5, which is followed by the conclusion of the work in Chapter 6.

Chapter 2

Background on Deep Learning

2.1 Introduction to Deep Architecture

Deep network [31] is a newly developed hierarchical learning paradigm which tries to extract a set of hierarchical meaningful binary features from an either binary or real-valued input data vector. It usually consists of stacks of same simple building block, and the output of one building block behaves as the input of the next building block.

For different tasks, there exist several common building blocks for the deep network, for instance, the binary restricted Boltzmann machine and Gaussian Bernoulli restricted Boltzmann machine for modeling a single data vector, the conditional restricted Boltzmann machine for modeling a conditional probability distribution, and the gated Boltzmann machine for modeling the higher-order relationship between two sets of possible data vectors. Also, another type of building block is called auto-encoder where the model is to learn a compressed representation of the data. As the autoencoder is not discussed in this thesis, we left this particular building block for future research questions. For more details, please see [2].

In this chapter, we will review the structural differences of different Boltzmann machines, and also their specific learning algorithms.

2.2 Boltzmann Machine

Boltzmann machine is a stochastic recurrent neural network which consists of binary neurons [28]. The states for the neurons $\mathbf{m}$ are either "on" or "off", which often denotes as 1 and 0 for computational simplicity. The connections within neurons $\mathbf{m}$ are un-directed, and often the self-connection is always set to 0. In the definition of Boltzmann machine, and we generally denote all the neurons together as $\mathbf{m}$.

The primary goal of the Boltzmann machine is to learn to model the properties of input data vector correctly by a Boltzmann distribution. In order to do that, the set of neurons are always divided to two parts: visible neurons and hidden neurons. For the consistency of

the thesis, we denote the visible neurons as $\mathbf{x}$ and the hidden neurons as $\mathbf{h}$. The target input data vectors will be fed to the predefined visible neurons $\mathbf{x}$, and the hidden neurons will try to capture the correct data distribution of input data vectors by the Boltzmann distribution.

We assume the data distribution of input vector is denoted as $P(\mathbf{X})$, where $\mathbf{X}$ is the whole training samples resembling $\mathbf{x}$. To compute the probability, we often define $P(\mathbf{X})$ as

$$P(\mathbf{X}) = \sum_{\mathbf{H}} P(\mathbf{X}, \mathbf{H}) = \sum_{\mathbf{H}} \frac{1}{Z} \exp\left(-E(\mathbf{X}, \mathbf{H})\right) \tag{2.1}$$

where $\mathbf{H}$ is the corresponding hidden state of the Boltzmann machine for $\mathbf{X}$. Z is the normalization constant for the energy model. $E(\mathbf{x}, \mathbf{h})$ denotes as the energy of the neurons, where

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{ij, i \neq j} w_{ij} \mathbf{m}_i \mathbf{m}_j \tag{2.2}$$

where w_{ij} is the weight between two different neurons $\mathbf{m}_i$ and $\mathbf{m}_j$.

Training a general Boltzmann machine is considered to be very difficult due to its complex dependencies, and the computationally intractable normalization term Z. But this does not mean that the training cannot be done. To learn the data distribution $P(\mathbf{X})$, one usually tries to maximize the log-likelihood of the marginal distribution of $P(\mathbf{X}, \mathbf{H})$ with respect to $\mathbf{X}$. To write it down,

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \log P(\mathbf{x}) = \log \sum_{\mathbf{h}} \frac{1}{Z} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \\ &= \log \frac{1}{Z} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \\ &= \log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) - \log \sum_{\mathbf{x}, \mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \end{aligned} \tag{2.3}$$

Taking the derivate with respect to the parameter w_{ij}, we can have

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left(\log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) - \log \sum_{\mathbf{x}, \mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \right) \\ &= \frac{\partial}{\partial w_{ij}} \left(\log \sum_{\mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right) - \log \sum_{\mathbf{x}, \mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right) \right) \\ &= \left\langle \frac{\sum_{\mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right) m_i m_j}{\sum_{\mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right)} \right\rangle_d - \left\langle \frac{\sum_{\mathbf{x}, \mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right) m_i m_j}{\sum_{\mathbf{x}, \mathbf{h}} \exp\left(\sum_{ij, i \neq j} w_{ij} m_i m_j \right)} \right\rangle_m \\ &= \langle m_i m_j \rangle_m - \langle m_i m_j \rangle_d \end{aligned}$$

where $\langle \cdot \rangle_d$ represents the expectation over the data distribution where input data vector is $\mathbf{x}$, and $\mathbf{h}$ follows the conditional distribution of the model given the input data $P(\mathbf{h}|\mathbf{x})$. Similarly, $\langle \cdot \rangle_m$ represents the expectation over the model distribution $P(\mathbf{x}, \mathbf{h})$. After obtaining the gradient, the weights can be updated using simply the gradient ascent by

$$w_{ij} = w_{ij} + \sigma \nabla w_{ij}$$

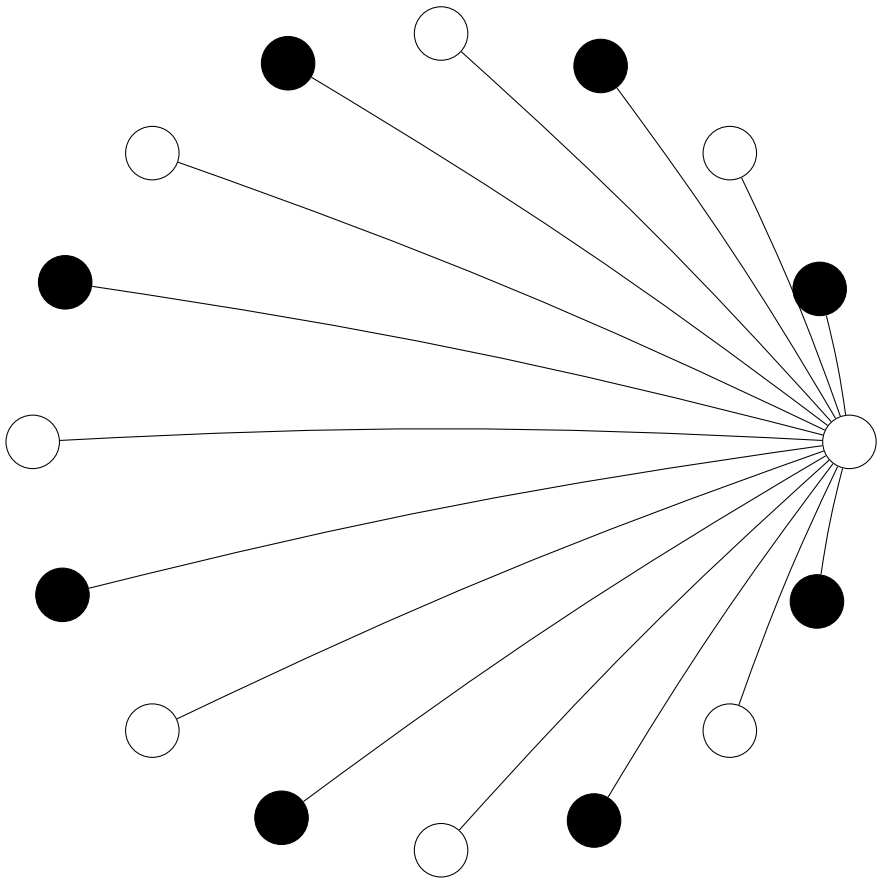


Figure 2.1: The graphical illustration of Boltzmann machine

where σ is the learning rate, which is generally chosen by the cross-validation or heuristically.

From the derivation above, in order to the get the gradient information about w_{ij}, we have to compute the expectation of $m_i m_j$ over model distribution. This is in general intractable as there are simply exponentially many possible combinations in the network[1]. Additionally, due to the complex connections, one cannot efficiently compute the gradient information block by block comparing to other simple Boltzmann machines. A graphical illustration of the general Boltzmann machine can be found in Figure 2.1[2]. From the figure, the black nodes represents the hidden neurons, while the visible neurons are represented by the white node. An arbitrary node is connected to all the other nodes, regardless of visible or hidden variables. This structure gives the Boltzmann machine huge advantages of modeling the structural information, yet the extremely difficult learning algorithms made the general Boltzmann machine hardly applicable to an actual application.

[1]Of course, if there is a few neurons in both input and hidden neurons, the brute force approach definitely works. Whereas the brute force method only work when the network is extremely small, and this is basically useless in real-world data and problems.

[2]Modified from http://www.texample.net/tikz/examples/complete-graph/

2.3 Restricted Boltzmann Machine

General Boltzmann machine is far from the real use due to its difficult learning process and complicated network structure. However, there is an amazing simplification which will make general Boltzmann machine useful, namely, removing the connections within the visible neurons $\mathbf{x}$ and hidden neurons $\mathbf{h}$ such that the network is bi-partite. This simplification resembles a very important building block for deep network, namely the restricted Boltzmann machine. The word "restricted" means that the network is restricted in such a way that there is only undirected connection from the visible neurons to the hidden neurons or vice versa, but not within the visible neurons or hidden neurons.

Given the hidden neurons $\mathbf{h}$ and visible neurons $\mathbf{x}$, the energy function of the model is defined as

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{ik} x_i h_k w_{ik} - \sum_i x_i b_i - \sum_k h_k c_k \tag{2.4}$$

where w_{ik} represent the weight between the neuron x_i and h_k, and b_i and c_k denotes the bias terms for neuron x_i and h_k respectively. The restricted Boltzmann machine similarly tries to model the input data distribution $P(\mathbf{x})$ by a set of hidden neurons $\mathbf{h}$. The marginal distribution $P(\mathbf{x})$ can be computed as

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h}) = \sum_{\mathbf{h}} \frac{1}{Z} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \tag{2.5}$$

To learn the model, the maximum log-marginal likelihood is sought, which is

$$\begin{aligned}
\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}_{ik}} &= \frac{\partial}{\partial \mathbf{w}_{ik}} \log P(\mathbf{x}) = \frac{\partial}{\partial \mathbf{w}_{ik}} \left(\log \sum_{\mathbf{h}} \frac{1}{Z} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \right) \\
&= \frac{\partial}{\partial \mathbf{w}_{ik}} \left(\log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) - \log \sum_{\mathbf{x}, \mathbf{h}} \exp\left(-E(\mathbf{x}, \mathbf{h})\right) \right) \\
&= \langle \mathbf{x}_i \mathbf{h}_k \rangle_m - \langle \mathbf{x}_i \mathbf{h}_k \rangle_d
\end{aligned}$$

Due to the removal of the additional connections, $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$ can be written down in analytical form. This is crucial in training the restricted Boltzmann machine as one has to perform an efficient Gibbs sampling over the model distribution. The exact conditional forms of both of $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$ make it possible to dramatically speed up training a restricted Boltzmann machine. The detailed learning algorithm called contrastive divergence will be discussed in later sections. For the completeness of the discussion, the exact conditional forms are written here:

$$\begin{aligned}
P(\mathbf{x}|\mathbf{h}) &= \prod_i \frac{1}{1 + \exp\left(-\sum_k h_k w_{ik} - b_k\right)} \\
P(\mathbf{h}|\mathbf{x}) &= \prod_k \frac{1}{1 + \exp\left(-\sum_i x_i w_{ik} - c_k\right)}
\end{aligned} \tag{2.6}$$

While the pre-constrained simplification gives excellent statistical properties to the restricted Boltzmann machine, the graphical illustration is also changed dramatically. The

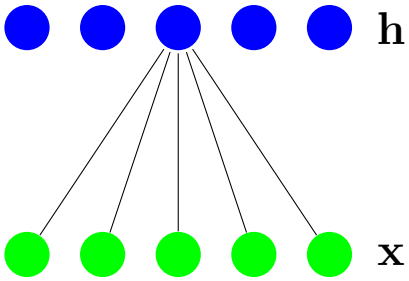


Figure 2.2: The schematic illustration of restricted Boltzmann machine

semantic illustration of the restricted Boltzmann machine is shown in Figure 2.2. In the figure, a visible neuron is only fully connected to the hidden neurons, and never connected to the visible neurons.

2.4 Gaussian Restricted Boltzmann Machine

Restricted Boltzmann machine is capable of modeling only binary input vectors due to its structural assumptions. This actually severely constrains the usability of the network as most data set in the real world applications are real-valued, non-binary: image pixel intensities, sea temperatures, object coordinates, signal strength, etc. Therefore, the extensions of restricted Boltzmann machine for real-valued input data vectors are of huge importance. This resulted in the development of Gaussian Bernoulli restricted Boltzmann machine [31][52].

Gaussian Bernoulli restricted Boltzmann machine is a variant of restricted Boltzmann machine which has the ability to learn a set of binary hidden feature vectors from the continuous valued input data vectors. The data vectors are considered to follow Gaussian distribution. Shortly put, the new energy function is defined as

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{ik} \frac{x_i}{\sigma_i} h_k w_{ik} + \sum_i \frac{(x_i - b_i)^2}{\sigma_i^2} - \sum_k h_k c_k \tag{2.7}$$

where all the parameters have the same meaning as those in the ordinary restricted Boltzmann machine except that b_i and σ_i are new parameters that model the mean and the variance of the continuous valued input neuron x_i. The modification is done by keeping the efficient Gibbs sampling possible while estimating the model distribution. The conditional distribution form for $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$ can be analytically computed as

$$\begin{aligned} P(\mathbf{x}|\mathbf{h}) &= \prod_i \mathcal{N}\left(x_i; \sigma_i \sum_k h_k w_{ik} + b_i, \sigma_i^2\right) \\ P(\mathbf{h}|\mathbf{x}) &= \prod_k \frac{1}{1 + \exp\left(-\sum_i x_i w_{ik} - c_k\right)} \end{aligned} \tag{2.8}$$

The semantic illustration of the Gaussian restricted Boltzmann machine is the same as the structure in Figure 2.2, while the visible neurons are assumed to be Gaussian variables.

2.4.1 Enhanced Gaussian Restricted Boltzmann Machine

From the conditional forms of the Gaussian restricted Boltzmann machine, one can observe that the mean value of $P(\mathbf{x}|\mathbf{h})$ is controlled by the standard deviations σ_i. It it argued in [14] that the distraction from the standard deviations in the sampling procedures decreases the performance of the Gaussian restricted Boltzmann machine. Consequently, a new energy form is proposed to remove the unsatisfactory distraction from the standard deviation. The modified energy function is defined as

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{ik} \frac{x_i}{\sigma_i^2} h_k w_{ik} + \sum_i \frac{(x_i - b_i)^2}{\sigma_i^2} - \sum_k h_k c_k \tag{2.9}$$

where each sample x_i is divided by the variance instead of the standard deviation. Consequently, the conditional forms that are used in learning are then

$$\begin{aligned} P(\mathbf{x}|\mathbf{h}) &= \prod_i \mathcal{N}\left(x_i; \sum_k h_k w_{ik} + b_i, \sigma_i^2\right) \\ P(\mathbf{h}|\mathbf{x}) &= \prod_k \frac{1}{1 + \exp\left(-\sum_i x_i w_{ik} - c_k\right)} \end{aligned} \tag{2.10}$$

The authors proved that the modified energy form is excellent in terms of learning the generative models of images comparing to the old forms. Therefore, in my thesis, when I need to train a Gaussian restricted Boltzmann machine, the modified version is always used.

2.5 Conditional Restricted Boltzmann Machine

Restricted Boltzmann machine and Gaussian Bernoulli restricted Boltzmann machine are developed to model a data distribution $P(\mathbf{x})$. However, in a real-world problem, we might want to know the following information — the probability of event A given the event B. In such circumstances, there are two set of data vectors $\mathbf{x}$ and $\mathbf{y}$, and $P(\mathbf{x})$ only is insufficient for modeling the conditional probabilities. Therefore, in order to model conditional distribution $P(\mathbf{x}|\mathbf{y})$, the traditional restricted Boltzmann machine is extended to handle the conditional forms by introducing a set of dynamic weights of binary or real-valued input vector $\mathbf{x}$ such that they are determined by the presence of different information from $\mathbf{y}$. In general, the newly proposed form is considered as the conditional restricted Boltzmann machine [62] [63] [50]. The new energy function for modeling $P(\mathbf{x}|\mathbf{y})$ is defined as

$$\begin{aligned} E(\mathbf{x}, \mathbf{h}; \mathbf{y}) &= -\sum_{ik} x_i h_k w_{ik} - \sum_i x_i b_i^* - \sum_k h_k c_i^* \\ b_i^* &= b_i + \sum_j y_j A_{ij} \\ c_k^* &= c_k + \sum_k y_j C_{ik} \end{aligned} \tag{2.11}$$

where b_i^* and c_k^* are the dynamic parameters for determining the weights and biases that are determined by $\mathbf{y}$. If we want to model the real-valued data vector in $\mathbf{x}$, the energy model

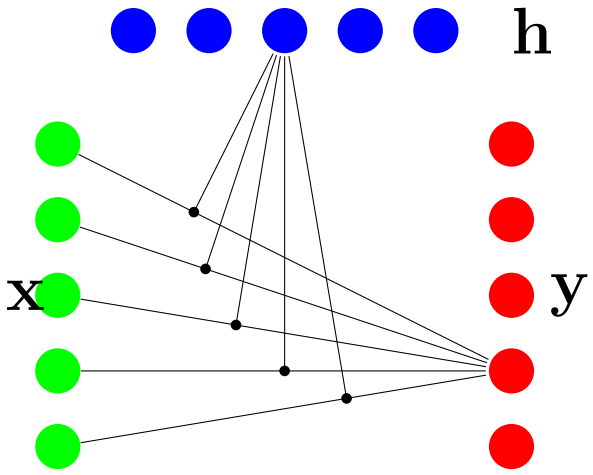


Figure 2.3: The schematic illustration of the gated Boltzmann machine

can accordingly change to

$$E(\mathbf{x}, \mathbf{h}) = -\sum_{ik} \frac{x_i}{\sigma_i} h_k w_{ik} + \sum_i \frac{(x_i - b_i^*)^2}{\sigma_i^2} - \sum_k h_k c_k^*$$

where only the normal bias terms are changed to the dynamic weights determined by the presence of the additional data vector $\mathbf{y}$.

Traditional Restricted Boltzmann machine is designed to model the model distribution $P(\mathbf{x})$, which only tries to model the data distribution correctly. In most problems, the patterns modeled in the hidden neurons are considered essentially to be a better feature set, where the hidden features in the data are better scattered. If one wants to build a classifier for the data and label, the common way is to build a logistic regression model on top of the learned feature. On the contrary, from the perspective of classification task, conditional restricted Boltzmann machine is more natural as it directly tries to model the conditional distribution $P(\mathbf{x}|\mathbf{y})$, which can be considered as the probability with which the data is classified into the class $\mathbf{x}$ given the data vector $\mathbf{y}$.

In many circumstances such as regression and classification, only the conditional distribution of the model is relevant. Also, as the conditional restricted Boltzmann machine is trying to model $P(\mathbf{x}|\mathbf{y})$, the size of data vector and the label vector can be arbitrary. Therefore, the conditional restricted Boltzmann machine has different applications: modeling human motions [63], multi-label classification [50], and image denoising [50], etc. Another relevant Boltzmann machine is called the Classification restricted Boltzmann machine [38] where one tries to build a classifier directly using the restricted Boltzmann machine.

2.6 Gated Boltzmann Machine

Traditional restricted Boltzmann machine and its Gaussian variant both try to model the patterns from a single data vector $\mathbf{x}$. Moreover, the conditional restricted Boltzmann ma-

chine steps further to model more complex patterns by considering the pattern within the data vector $\mathbf{x}$ given the information about the prior data vector $\mathbf{y}$.

One of the similarities of the aforementioned Boltzmann machines is that they only consider the interactions between one visible neuron and one hidden neuron. This somehow restricts the possibility of considering the higher order interactions between two or more visible neurons and the hidden neurons. To this end, a higher-order Boltzmann machine called Gated Boltzmann machine is proposed [46] [47]. Different from the other Boltzmann machines in which there is a interaction term up to first order with respect to $\mathbf{x}$, the gated Boltzmann machine considers the interactions up to two input neurons and one hidden neuron.

Similar to the conditional restricted Boltzmann machine where two data vectors $\mathbf{x}$ and $\mathbf{y}$ are considered, gated Boltzmann machine tries to model the interactions between two data vectors directly in a single interaction term. Therefore, a joint data distribution of $(\mathbf{x}, \mathbf{y})$ is sought, where $\mathbf{x}$ and $\mathbf{y}$ can be defined as a pair of images [47] or data from different modalities [48]. The gated Boltzmann machine is thus capable of modeling the highly complex relationship between two similar structured data or the highly complex inner structures of single structured data. The energy model of the gated Boltzmann machine is defined as

$$E(\mathbf{x}, \mathbf{y}, \mathbf{h}) = -\sum_{ijk} x_i y_j h_k w_{ijk} - \sum_i x_i b_i - \sum_j y_j b_j^\star - \sum_k h_k c_k \tag{2.12}$$

where w_{ijk} defines a element wise interaction weight for a data triplet $\{x_i, y_j, h_k\}$. $b_j^\star$ is the bias term for the input neuron y_j. The tensor term w_{ijk} gives the model the ability to model the interaction between the input vectors $\mathbf{x}$ and $\mathbf{y}$ in a set of hidden neurons $\mathbf{h}$. For modeling real-valued input vectors, a Gaussian bias term can be defined for both $\mathbf{x}$ and $\mathbf{y}$.

The number of free parameters in the tensor weight w_{ijk} can be dramatically large. Given the size of two input vector $\mathbf{x}$ and $\mathbf{y}$ as 500, and $\mathbf{h}$ as 1000, the number of parameters in w_{ijk} can go up to $500 \times 500 \times 1000$. The increase of the parameters makes the learning problem too hard to be efficient. Therefore, a low rank approximation form

$$w_{ijk} \to \sum_f w_{if}^x w_{jf}^y w_{kf}^k$$

has been proposed [47][55]. This approximation leads to an easier learning formulation, and yet gives a better performance in terms of modeling the higher order relationship between neurons.

A semantic illustration of the gated Boltzmann machine is shown in Figure 2.3. From the figure, it is identified that the hidden neurons are used to model the interactions between the other two visible input neurons by controlling the weight of the connections between the input neurons. Please note that there is no connection within each set of input neurons, which preserves the nice properties of training in restricted Boltzmann machine.

2.7 Convolutional Boltzmann Machine

An interesting observation about the image recognition task is that images are often divided into small patches of $n \times n$, where n is typically 8 to 32. Also, when we input the data, the

image patches are flattened to a vector form, and thus the input vector size is often from 64 to 1024. Therefore, image features are usually high-dimensional. It's often considered that high-dimensional features are harder to capture, and so special treatments have to be applied. From the context of deep learning, the input image sizes can be too large to process. For instance, even a small enough image, with size 32×32, is already over 1000 dimensions. Therefore, in the field of deep network, a model called convolutional deep belief network was proposed to extend the traditional deep network so that it is capable of processing arbitrary size of images [40].

The building block of the convolutional deep belief network is still a variant of restricted Boltzmann machine. Differently, the proposed restricted Boltzmann machine is designed to cope with arbitrary size of input vectors by performing a convolutional operations on hidden feature vectors. Therefore, this particular restricted Boltzmann machine is also called convolutional restricted Boltzmann machine. To define a convolutional restricted Boltzmann machine, the weights are shared between several hidden neurons and input vectors.

In convolutional restricted Boltzmann machine, there are also only hidden layer $\mathbf{H}$ with size $k \times k$ and one input layer $\mathbf{V}$ with size $N \times N$ (N is considered to too large for processing in a normal RBM). Differently, in convolutional restricted Boltzmann machines, there are K sets of hidden vectors for the convolutional restricted Boltzmann machine. Each hidden neuron has $k \times k$ neurons, and accompanying weight matrix $\mathbf{W}$ has the size $k \times N$. In order to restrict the computational cost, the weight matrix $\mathbf{W}$ is shared between different hidden vectors. Therefore, the energy function is defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} \sum_{i,j=1}^{N} \sum_{r,s=1}^{K} h_{ij}^{k} W_{rs}^{k} v_{i+r-1,j+s-1} - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N} h_{ij}^{k} - c \sum_{i,j=1}^{N} v_{ij} \tag{2.13}$$

where $\mathbf{b}$ is the shared bias for the set of hidden vectors. To simplify the notations a bit, the Eq. 2.13 can be re-formulated as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} \mathbf{h}^{k} \odot \left(W^{k} \mathbf{v}\right) - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{K} \mathbf{h}_{ij}^{k} - c \sum_{i,j=1}^{N} v_{ij} \tag{2.14}$$

where $\odot$ represents the summation of the dot product of two matrices. By sharing the weight and restricting the dimensions of the hidden neurons, the dimensions of the input vectors are greatly reduced while the majority of the hidden features are learned.

However, only by creating convolutional hidden layers does not allow one to capture comprehensive yet simple enough high level feature representations like the normal deep network. To do that, one has to perform an additional step called probabilistic max pooling where the objective is to provide a concise yet informative layer for recognition. However, different from the traditional max-pooling operations in feed-forward neural network, the probabilistic max pooling operates in a manner where bi-directional inferences are still possible. More importantly, the introduction of the probabilistic max pooling layer can reduce the size of the hidden layers by a significant order, which potentially makes the learning scalable in terms of the size of the input images.

To define the probabilistic max pooling, an additional pooling layer P is defined. As there are in total K different hidden layers, there are also P number of pooling layers. For

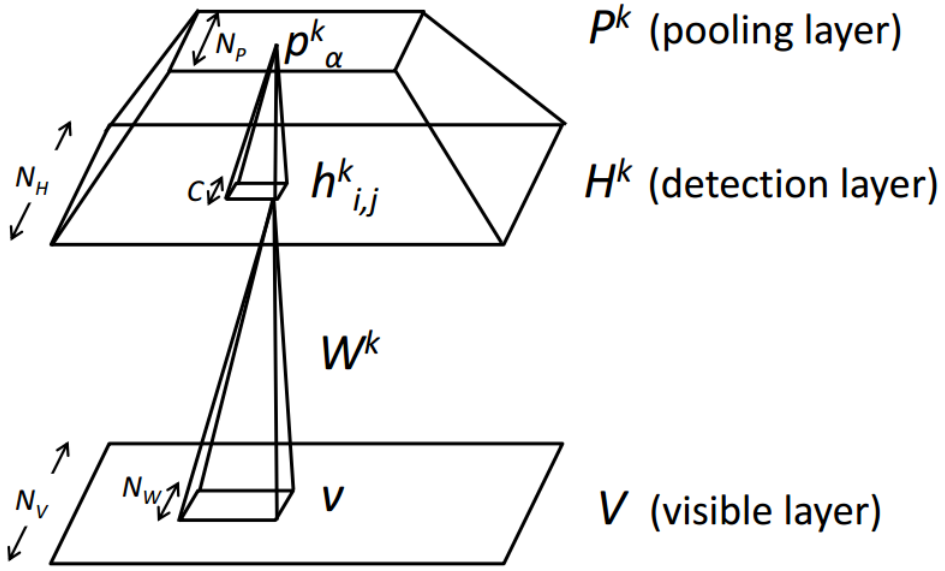


Figure 2.4: The convolutional Gaussian restricted Boltzmann machine [40]

each pooling layer, the number of elements in pooling layer is shrunk by a constant factor C by grouping one neuron p in the pooling layer P with a small block of neighboring hidden neurons B_α . Therefore, for a certain detection block B_α in hidden layer $\mathbf{H}^k$, the connected neuron p is activated in the pooling layer P. The neuron p is active if and only if one hidden neuron is active within the detection block B_α. The semantic illustration of the convolutional restricted Boltzmann machine is shown in Figure 2.4.

By defining the probabilistic max pooling, the formal definition of the convolutional restricted Boltzmann machine can be formulated as

$$\begin{aligned} E(\mathbf{v},\mathbf{h}) &= -\sum_k \sum_{i,j} \left(h^k_{i,j} \left(W^k v \right)_{ij} + b_k h^k_{ij} \right) - c \sum_{ij} v_{ij} \\ \text{subject to } & \sum_{(i,j)\in B_\alpha} h^k_{ij} \leq 1, \forall k, \alpha \end{aligned} \tag{2.15}$$

To create a deep network, the convolutional restricted Boltzmann machines are stacked together, and then pre-trained layer-wise to obtain a stable result.

Convolutional deep belief network is interesting because it has the ability to process full-size images, and can produce truly high level features in their top layers. One set of filter examples are shown in Fig 2.5.In the first row, the filters learned by the probabilistic max-pooling operations are shown, while the bottom figure shows the features learned from the original convolutional hidden layer.

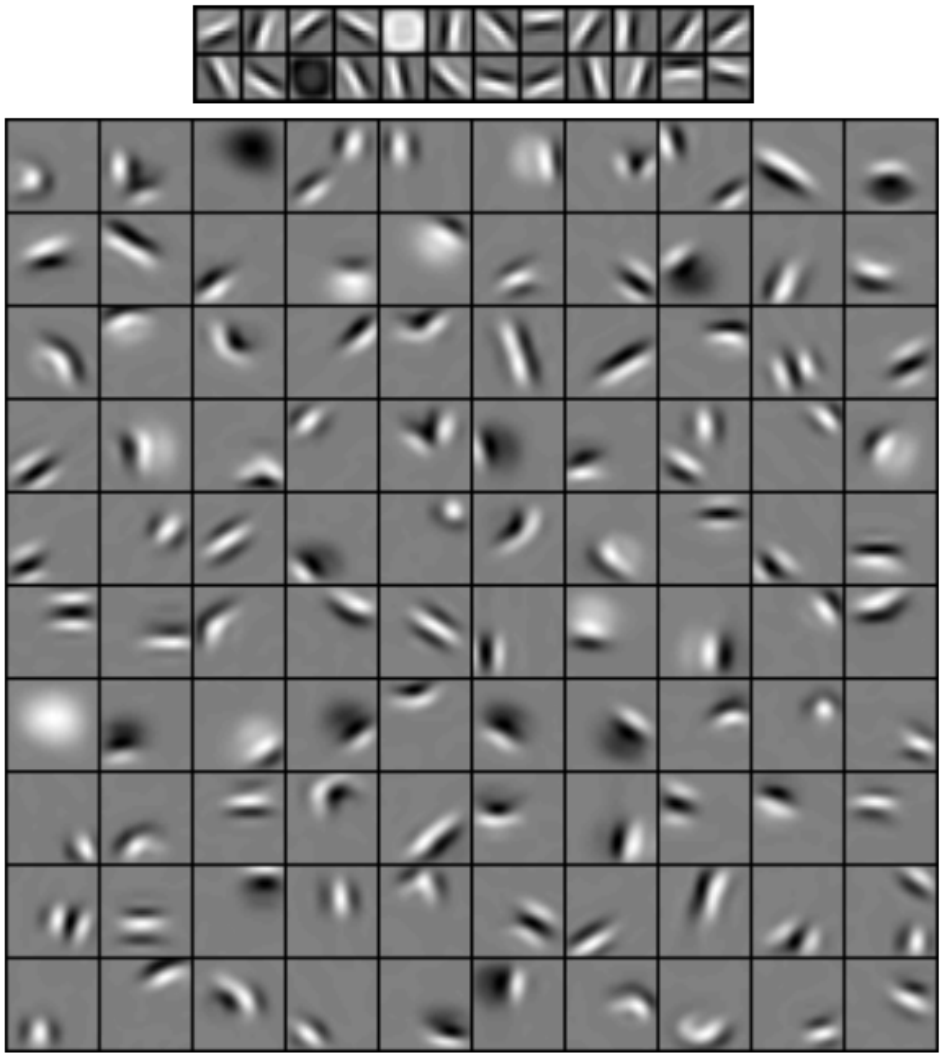


Figure 2.5: A middle level feature learned by the convolutional Gaussian restricted Boltzmann machine [40]

2.8 Learning Algorithms

In the pervious sections, a few representative formulations for Boltzmann machine are reviewed. Different formulation is used when the problem is different. Even though the energy function formulation shows difference, the underlying learning algorithm is the same.

Recall that training a Boltzmann machine needs to maximize the log-marginal likelihood of data distribution. As all the other proposed Boltzmann machines are special versions of the general Boltzmann machine, the same optimization strategy should apply to all of them. No matter if the Boltzmann machine has to learn $P(\mathbf{x})$, $P(\mathbf{x}|\mathbf{y})$ or $P(\mathbf{x}, \mathbf{y})$, the optimization strategy is to maximize the log-marginal distribution of the corresponding data distribution.

Without losing the generality, we define the energy function of Boltzmann machine as $E(\mathbf{m}, \mathbf{h})$ where $\mathbf{m}$ can be a general term that resembles the possible number of data vectors, for instance, $\mathbf{x}$ in the restricted Boltzmann machine and $(\mathbf{x}, \mathbf{y})$ in the conditional restricted Boltzmann machine and the gated Boltzmann machine. The marginal distribution of $P(\mathbf{m})$ is defined as

$$P(\mathbf{m}|\theta) = \sum_{\mathbf{h}} \frac{1}{Z} \exp\left(-E(\mathbf{m}, \mathbf{h}|\theta)\right) = \frac{1}{Z} \exp(-F(t)) \tag{2.16}$$

where Z is the normalization factor, and $Z = \sum_{\mathbf{m},\mathbf{h}} \exp\left(-E(\mathbf{m}, \mathbf{h})\right)$. θ is a general representation of the parameters in different models. $F(\mathbf{m})$ is called the free energy of the model, which can be calculated in a linear time with respect to the number of hidden neurons. To model a particular parameter θ, we need to maximize the log-marginal maximum likelihood by the gradient ascent algorithm, which is followed by

$$\begin{aligned}
\frac{\partial \log P(\mathbf{m})}{\partial \theta} &= \frac{\partial}{\partial \theta}\left(\log \frac{1}{Z} \exp\left(-F(\mathbf{m}|\theta)\right)\right) \\
&= \frac{\partial}{\partial \theta}\left(\log \exp\left(-F(\mathbf{m}|\theta)\right) - \log \sum_{\mathbf{t}} \exp\left(-F(\mathbf{m}|\theta)\right)\right) \\
&= \frac{\partial - F(\mathbf{m}|\theta)}{\partial \theta} - \frac{\partial \log \sum_{\mathbf{m}} \exp\left(-F(\mathbf{m}|\theta)\right)}{\partial \theta}
\end{aligned}$$

The first term of the gradient can be efficiently computed and it is always called the positive gradient. The training difficulties of the energy model arise from that the second term of the gradient, often called as a negative gradient, cannot be efficiently computed with respect to the number of input neurons and hidden neurons, and thus an efficient approximation algorithm for the negative gradient is required.

In this section, a set of approximation algorithms are reviewed, and the advantages and disadvantages of different methods are also discussed.

2.8.1 Gibbs Sampling

Gibbs sampling [13] is a Markov chain Monte Carlo sampling algorithm for obtaining a sequence of random samples from a multivariate stochastic distribution. It is an alternative approach for drawing samples from a computationally intractable joint distribution such as $P(\mathbf{t}, \mathbf{h})$.

In quite a few probabilistic settings, where the joint distribution of the variables is intractable, the conditional distributions of the variables can be computed efficiently. Gibbs sampling stemmed from this fact, and enables to obtain samples from the true joint distribution efficiently. In order to sample from the joint distribution, the Gibbs sampler first initialize a set of random values for the set of variables, and then obtains a sample for a variable from the conditional probability distribution of that variable given the rest of samples. Afterwards, another sample is drawn for another variable from its conditional form given the previous sampled variable and other random variables. This process has to proceed until the the sampled distribution reach the equilibrium distribution.

In Boltzmann machine, due to the partition function in the model distribution, drawing samples from the joint probability distribution is too expensive to do in general cases. However, thanks to the bipartite structure in the simplified Boltzmann machine, it is however possible to compute the conditional forms of the model exactly and efficiently.

In the simplest form, we introduce the example for sampling from the model distribution in the restricted Boltzmann machine. Similar sampling rules can be applied to other Boltzmann machines as long as the Gibbs sampling is possible in those models.

In restricted Boltzmann machine, the model tries to learn the pattern from the data distribution $P(\mathbf{x})$ in a set of hidden neurons. Maximizing the log-marginal-likelihood of model distribution is identical to minimizing the inverse version of that. Therefore,

$$\mathcal{L}(\theta) = -\log P(\mathbf{x}) = -\log \sum_{h} P(\mathbf{x}, \mathbf{h})$$

To minimize, take the derivative of negative-log-likelihood with respect to parameter θ,

$$\begin{aligned}
\frac{\partial \mathcal{L}(\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\left(-\log P(\mathbf{x})\right) = \frac{\partial}{\partial \theta}\left(-\log \sum_{\mathbf{h}} P(\mathbf{x}, \mathbf{h})\right) \\
&= \frac{\partial}{\partial \theta}\left(-\log \frac{1}{Z}\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))\right) \\
&= \frac{\partial}{\partial \theta}\left(-\log \frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}^0, \mathbf{h}))}{\sum_{\mathbf{x},\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}\right) \\
&= \frac{\partial}{\partial \theta}\left(-\log \sum_{h} \exp(-E(\mathbf{x}, \mathbf{h})) + \log \sum_{\mathbf{x}^0,\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))\right) \\
&= \left(\frac{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))\frac{\partial -E(\mathbf{x},\mathbf{h})}{\partial \theta}}{\sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))}\right) - \left(\frac{\sum_{\mathbf{x}^0,\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h}))\frac{\partial -E(\mathbf{x},\mathbf{h})}{\partial \theta}}{\sum_{\mathbf{x},\mathbf{h}} \exp(E(\mathbf{x}, \mathbf{h}))}\right) \\
&= \left\langle \frac{\partial - E(\mathbf{x}, \mathbf{h})}{\partial \theta} \right\rangle_{P(\mathbf{h}|\mathbf{x})} - \left\langle \frac{\partial - E(\mathbf{x}, \mathbf{h})}{\partial \theta} \right\rangle_{P(\mathbf{x},\mathbf{h})}
\end{aligned}$$

where we can see that one has to compute the expectation over the model distribution $P(\mathbf{x}, \mathbf{h})$ to calculate the gradient information. This computation is intractable but the block Gibbs sampling can be done extremely fast due to the factorial nature of the Boltzmann machine. To sample from the model distribution, Gibbs sampler alternatively samples from

Algorithm 1 Gibbs sampling algorithm

Input: $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$

Output: Samples from the model distribution $P(\mathbf{x}, \mathbf{h})$

```
Initialize the input neurons x with data vector x^0
t = 0
while True do
    Sample from P(h^{t+1}|x^t)
    Sample from P(x^{t+2}|h^{t+1}) , and denote the sampled visible vector as x^t
    if the distribution reach equilibrium then
        break
    end if
end while
Choose the x^tth sample as the sample from the model distribution
```

the conditional form $P(\mathbf{x}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{x})$ until reaching the equilibrium distribution. The detailed Gibbs sampling algorithm is shown in algorithm 1.

Note that, similar gradient information can also be obtained similarly when the other Boltzmann machine is modeled. For instance, for conditional restricted Boltzmann machine, the gradient information for parameters θ is

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \left\langle \frac{\partial - E(\mathbf{x}, \mathbf{h}; \mathbf{y})}{\partial \theta} \right\rangle_{P(\mathbf{h}|\mathbf{x},\mathbf{y})} - \left\langle \frac{\partial - E(\mathbf{x}, \mathbf{h}; \mathbf{y})}{\partial \theta} \right\rangle_{P(\mathbf{x},\mathbf{h}|\mathbf{y})}$$

where we can still do the Gibbs sampling on the model distribution $P(\mathbf{x}|\mathbf{y})$ for the approximation of the gradient information.

Even though the Gibbs sampler can reach equilibrium distribution in a finite number of iterations, the sample obtained from the Gibbs chain is far from ideal [29]. The samples from the equilibrium distribution generally have high variances, and the high variance usually swamps the model's distribution. Therefore, the actual learning process for Boltzmann machine using Gibbs sampler cannot be carried out efficiently due to this poor nature.

2.8.2 Contrastive Divergence

Due to the limited sampling results of Gibbs sampling in training the Boltzmann machine, an efficient and feasible learning algorithm called contrastive divergence is proposed by Hinton [29].

Contrastive divergence is a general learning algorithm for energy-based models, including Boltzmann machines. The basic idea is to use the samples from the k finite step Gibbs sampling to approximate the target distribution.

The main problem in the pure Gibbs sampling is that the samples from the equilibrium distribution in the Gibbs sampler have too high variances, overwhelming the true target

Algorithm 2 Contrastive divergence algorithm

Input: $P(\mathbf{x}|\mathbf{h})$, $P(\mathbf{h}|\mathbf{x})$ and K

Output: Samples from the model distribution $P(\mathbf{x})$

Initialize the input neurons $\mathbf{x}$ with data vector $\mathbf{x}^0$

$t = 0$

while $t <= K$ **do**

Sample from $P(\mathbf{h}^{t+1}|\mathbf{x}^t)$

Sample from $P(\mathbf{x}^{t+2}|\mathbf{h}^{t+1})$, and denote the visible vector as $\mathbf{x}^t$

end while

Choose the $\mathbf{x}^t$th sample as the sample from the model distribution

distribution. However, if one initializes the Gibbs chain with the data vectors in the distribution, the samples from the kth step of the Gibbs sampler have already made some changes from original data points, which is the error information that is needed for the learning the target model. In practice, one can perform the contrastive divergence according to the algorithm 2. It is also experimentally confirmed that even 1 step of Gibbs sampling is enough for learning a successful target distribution using RBM. Moreover, while doing the actual sampling, the sampling approach can be replaced by the mean-field approximation [58] [45].

The contrastive divergence was proposed to mainly cope with the learning of product of expert models, and then mainly used in learning variants of Boltzmann machines (e. g. see [31] [61] [55] [30]). Although there are lots of large scale applications of the learning algorithm in various fields, there is still room for improvement in terms of learning a good generative models. The main flaw of the contrastive divergence is the initialization of a new Gibbs chain. Every single update based on the learning instances can be too arbitrary and ignore the learned information from the previous updates. For overcoming this theoretical weak point, several notable algorithms are proposed: Persistent Contrastive Divergence which is aimed to tackle the arbitrary initialization in contrastive divergence for every update, and parallel tempering which is aimed to tackle the arbitrary initialization problem by the multiple Gibbs sampling chains.

2.8.3 Persistent Contrastive Divergence

Due to the success of the contrastive divergence and some theoretical weak points of the algorithm, a better algorithm called persistent contrastive divergence [64] is proposed to extend the excellent learning properties of contrastive divergence. A further extension and speed-up of the algorithm was proposed later in [65].

The model is proposed because the changes in the model between each update are small and reseting the Gibbs chain will harm the actual learning. Therefore, persistent contrastive divergence is proposed to use a single Gibbs chain throughout the experiment and to use the

estimated value of previous update as the current initial point of the Gibbs chain, and then further do the training in each step.

A series of experiments confirmed that the persistent contrastive divergence can learn a better model given the same computation power and time. A speed-up version of persistent contrastive divergence was proposed later [65]. This algorithm is omitted here because the persistent contrastive divergence is not studied in this thesis.

2.8.4 Enhanced Gradient Learning

One big drawback of the contrastive divergence is that it is not guaranteed to converge to the true distribution, but rather it converges to a biased model distribution [12]. Even though the bias is rather small in practice, it is however good to have a better solution. Therefore, Cho et al. [15] proposed an enhanced gradient algorithm that has better generative modeling learning capabilities. In addition to that, the enhanced gradient learning algorithm also simplifies the choice of the learning parameters by introducing the adaptive learning rate for the RBM training.

Cho et al. [15] argued that the traditional restricted Boltzmann machine learning algorithm is biased in terms of data representations, and has difficulties in focusing on learning meaningful feature sets due to the biases. By defining the covariance between the visible neurons and hidden neurons under the arbitrary distribution P,

$$\mathrm{Cov}_P(\mathbf{x}, \mathbf{h}) = \langle \mathbf{x}, \mathbf{h} \rangle_P - \langle \mathbf{v} \rangle_P \langle \mathbf{h} \rangle_P \tag{2.17}$$

the traditional gradient information of $\mathbf{w}_{ij}$ is written as

$$\nabla w_{ij} = \mathrm{Cov}_d(x_i, h_k) - \mathrm{Cov}_m(x_i, h_j) + \langle x_i \rangle_{dm} \nabla c_j + \langle h_j \rangle_{dm} \nabla b_i \tag{2.18}$$

where $\langle \cdot \rangle_{dm}$ is the average of the expectation of the variables over the data distribution and the model distribution. From the previous formulation, it is clear that the bias terms are involved in updating the feature sets. The effects can be dominant when large amount of hidden neurons are active, neglecting the information from other sources. To cope with this problematic fact, a new learning rule called the enhanced gradient learning algorithm for Boltzmann machine is proposed. The effects from the bias terms can be neglected if the expectation of input neurons $\mathbf{x}$ over the distribution P is close to zero. From this perspective, the new learning rule of the enhanced gradient learning rule combines all the possible combinations of bit-flipping transformations so as to produce a new less-biased learning rule, which is defined by the gradients,

$$\begin{aligned} \nabla w_{ij} &= \mathrm{Cov}_d(x_i, h_k) - \mathrm{Cov}_m(x_i, h_j) \\ \nabla b_i &= \langle x_i \rangle_d - \langle x_i \rangle_m - \sum_j \langle h_j \rangle_{dm} \nabla w_{ij} \\ \nabla c_j &= \langle h_j \rangle_d - \langle h_j \rangle_m - \sum_i \langle v_i \rangle_{dm} \nabla w_{ij} \end{aligned} \tag{2.19}$$

These update rules are more elegant in terms of removing the biases from the input data to the gradient information of feature sets. For the detailed derivation, please see Cho's thesis[3].

[3] http://lib.tkk.fi/Dipl/2011/urn100427.pdf

In addition to the aforementioned enhanced gradient learning rules, a new adaptive learning rate selection scheme is also proposed to ease the learning of the deep models.

We assume θ and θ' are the parameters for the deep model, and θ' denotes the parameters after one update of θ. If the change between the two updates is minimal, we can assume the following equality,

$$P_{\theta'}(\mathbf{x}) = \frac{P_\theta(\mathbf{x})}{Z_\theta}\frac{Z_\theta}{Z_{\theta'}} \tag{2.20}$$

where $P_{\theta'}(\mathbf{x})$ is the model distribution given the parameter θ', and $P_\theta(\mathbf{x})$ is the model distribution given the parameter θ. $Z_{\theta'}$ and Z_θ are the corresponding normalization constants. According to the definitions of normalization constants in deep model, the previous equality can be further expanded to

$$P_{\theta'}(\mathbf{x}) = \frac{P_\theta(\mathbf{x})}{Z_\theta}\frac{Z_\theta}{Z_{\theta'}} = \frac{P_\theta(\mathbf{x})}{Z_\theta}\left\langle \frac{P_{\theta'}(\mathbf{x})}{P_\theta(\mathbf{x})} \right\rangle_{P_\theta}^{-1} \tag{2.21}$$

From the new formulation, we can actually compute the local likelihood by literally utilizing the positive and negative gradient obtained by the learning procedure. To select the most appropriate learning rate, the one that gives the best log-likelihood values is selected. In order to minimize the additional computational time because of selecting the better learning rate, and also to eliminate the potential fluctuation of learning rate throughout the learning, a potential set of learning rate is $\{(1-\epsilon)^2\eta, (1-\epsilon)\eta, \eta(1+\epsilon)\eta, (1+\epsilon)^2\eta\}$, where ϵ is a small value, often selected $\epsilon = 0.01$.

2.9 Conclusions

In this chapter, we introduced and reviewed a series of different Boltzmann machines and their learning algorithms. Different Boltzmann machines are developed to perform different tasks, for instance, restricted Boltzmann machines for modeling binary input vectors in hidden binary vectors, and gated Boltzmann machines for modeling the interactions between two sets of input vectors in single hidden neurons. These differences make the applications of deep networks fairly broad, and a number of applications have been developed and discussed in the past years.

To overcome the difficulties of learning the energy model like that in Boltzmann machines, a milestone was achieved when Hinton proposed the initial contrastive divergence algorithms. Afterwards, various authors have been trying to improve the learning algorithms, resulting in several enhanced learning schemes, such as the persistent contrastive divergence, fast persistent contrastive divergence, and the enhanced gradient learning.

In spite of the vast development of this sub-field of machine learning, there are still lots of barriers in exploring full potentials of deep networks. A series of possible future directions are possible.

Chapter 3

Background on Texture Modeling

3.1 Texture Modeling

Texture information modeling has been studied for decades, see, e.g., [27]. It can be understood by considering combinations of several repetitive local features. In this manner, various authors proposed hand-tuned feature extractors. Instead of understanding the generative models for textures, those extractors try to consider the problem discriminatingly. An old model called co-occurrence matrix was proposed in [27], where it was used to measure how often a pair of pixels with a certain offset gets particular values, thus tackling the structure of the textures. Despite the good performances of these extractors, they suffer from the fact that they contain only little information about the generative model for textures. Also, these extractors can only be applied to certain type of data, and it is fairly hard to adopt them to other tasks if needed. Conversely, generative models of textures can be applied to various texture modeling applications. In this direction, some statistical approaches for modeling textures have been introduced in [66] and [42]. A pioneering work of texture modeling using deep network is proposed in [36].

Texture modeling is a very important task in real-world computer vision applications. An object can have any shape, size, and illumination condition. However, the texture pattern within the objects can be rather consistent. By understanding that, one can improve the understanding of objects in complex real-world recognition tasks.

3.2 Texture Classification

Texture can in general be treated as repetitive patterns where different ones of them resemble certain of types of textural information. Textural information is included as one of the basic properties of objects. Other properties include shapes, size, color, and reflections. Sometimes, while the other properties of the objects vary dramatically, the textures of objects can remain rather consistent. For instance, in a construction site, the remains of different materials can be found in any shape, and it is also likely that the objects are occluded

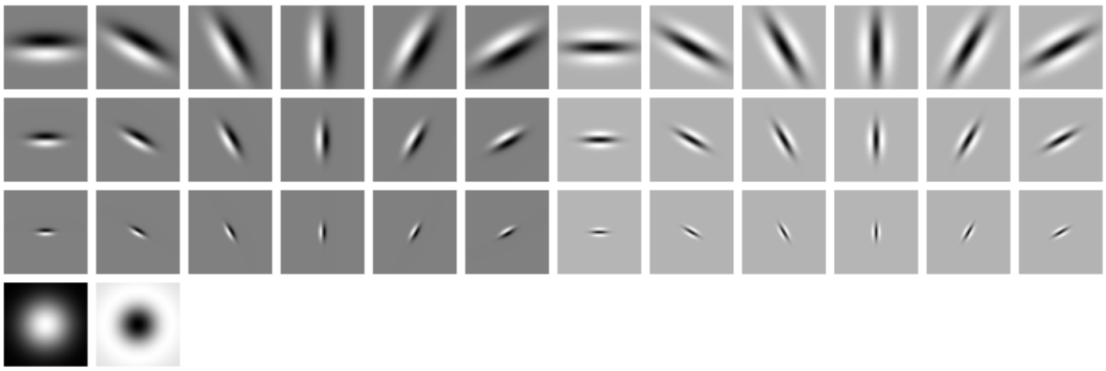


Figure 3.1: An illustrative example of the texton used in texture classification [67].

by other objects. For a successful recognition application, it might be crucial that the recognition system is capable of capturing the object by the textural information of the different objects.

One of the most widely studied subject for texture modeling is its classification problem. In real life, people mostly need to categorize different types of textures, such as trees, grass, etc. Texture classification studies can essentially be divided into the following periods:

Early Research Period Early research on visual patterns are conducted in [34] [33]. In these papers, the main research goal was to investigate how textural patterns are of importance in visual systems. For instance, Julesz [33] discussed in which ways different textures are discriminant to the human vision. In the same publication, the same author proposed that texture cannot be discriminated if the second-order statistics are identical in two different textures, and then proved the statement false in his later publication in [34]. More importantly, a very important local feature extractor called texton is proposed to tackle the textural information discrimination tasks. The texton have been a main stream for most of the texture classification algorithms until some other statistical methods were proposed in recent years. An example of textons is shown in Figure 3.1.

Binary Texture Classification The actual texture classification research started with binary textures. This is one of the simplest form of texture comparing to other complex textures with different perspectives. Binary textures are binary visual patterns resembling different visual perceptions. Clearly, in binary texture images. each pixel has typically only two states: black and white. Béla Julesz developed a series of theorems on how textural information can be discriminated using different statistical information in [34] [33]. Some further theoretical textural discrimination algorithms were studied in [44]. Binary texture classification serves as a theoretical foundation and starting point of the real world textural information research.

2D Texture Classification Soon after the introduction of the digital scanner and the digital cameras, there exists a massive amount of grey-level textures images that cannot

be studied using the simple binary texture discrimination methods. Some of the representative work includes [26] [25] and [60]. In these works, all the methods strived for coping with the texture with different rotations, scaling transformation. As all the possible transformations are in two dimensional plane, this category is often called 2D texture classification.

3D Texture Classification To advance the texture recognition and classification in computer vision, it is crucial to understand how the 3D transformation can be compensated in the field of texture classification. In real world problems, the form of textures appearing in the real world data set are so arbitrary that those simple 2D texture classification methods cannot perform well enough. From this perspective, a few representative work including [6] [17] [37] [41] [59] [66] [67] are proposed to tackle the problem. One of the most recent advances was to use the random projection method [42] to produce a random low dimensional representation of the texture. This method is much simpler than the previous methods but yet outperform the previous methods.

3.3 Texture Reconstruction

Texture synthesis is a different application where the objective is not to differentiate between the textures, but to generate textures based on some criteria. Texture synthesis is an interesting problem as it involves a true understanding of the texture, and then re-generation of similar or identical textures based on the learned model. Typical discrimination methods don't work in this application as they barely understand the structures of the textures, but only know the discriminative features from one another. On the contrary, in general, statistical approaches to texture modeling can have texture synthesis properties as the methods are developed to understand the nature of the different textures. For instance, in [66], the author experimented with the texture synthesis with their texture modeling methods by learning explicitly the probability distribution of the textures.

Another similar texture synthesis example comes from [23] where the authors proposed to model the conditional distribution of a pixel given the other nearby pixels. This method is different from the aforementioned method because it didn't learn explicitly the probability distribution of the texture, but learned a conditional Markov random field and then tried to minimize the corresponding energy model.

The latter approach is similar to deep network in terms of utilizing the energy model. Deep network often uses a building block called restricted Boltzmann machine or its variants such that it can learn a highly informative feature representation of the input vectors. Due to the restricted structures of the different proposed Boltzmann machines, we can re-generate textures by sampling from the hidden neurons. Additionally, the highly informative textural feature sets are highly likely to provide better classification results comparing to raw pixel classification using the conventional classifier such as k-nearest neighborhood classifier.

In this thesis, a new possible way of modeling texture is proposed such that the model can have a good texture classification model yet preserving good generative models of the

textures. Before going to the detailed explanation of the proposed approaches, some basic texture classification methods are introduced in the next section.

3.4 Common Texture Modeling Methods

In this section, some typical texture modeling methods are reviewed. The algorithms are reviewed because of their important contributions to the texture modeling and computer vision, and also due to the fact that the reader can have a better perspective about the texture modeling field if the following methods are understood. The previous methods are divided into two sub-sections according to the nature of methods: if they utilize the statistical properties of the textures, the methods are counted as statistical approaches, and if not, then the method is classified into non-statistical approaches. Typically, in non-statistical approaches, methods like gray level co-occurrence matrices are included. On the other hands, methods like texture classification using random projections are included in statistical approach as they all utilize the statistical properties of textural information.

3.4.1 Early Stage Approaches

Early stage approaches generally utilize the early conventional feature extraction methods such as the gray level co-occurrence matrices.

Co-occurrence Matrices in Texture Modeling

Co-occurrence matrix [27] measures the frequencies of a pair of pixels which with a certain offset get particular values. Modeling co-occurrence matrices instead of pixels brings the analysis to a more abstract level immediately, and it has therefore been used in texture modeling.

The co-occurrence matrix $\mathbf{C}$ is defined over $\{m \times n\}$ size image I, where $\{1 \dots N_g\}$ levels of gray scales are used to model pixel intensities. Under this assumption, the size of $\mathbf{C}$ is $\{N_g \times N_g\}$. Each entry in $\mathbf{C}$ is defined by

$$c_{ij} = \sum_{m=1}^{M} \sum_{n=1}^{N} \begin{cases} 1 \text{ if } I(m,n) = i \ \& \ I(m+\delta_x, n+\delta_y) = j \\ 0 \text{ otherwise} \end{cases} \tag{3.1}$$

Different offset schemes for $\{\delta_x, \delta_y\}$ result in different co-occurrence matrices. For instance, one can look for textural pattern over an image with offset $\{-1, 0\}$ or $\{0, 1\}$. These different co-occurrence matrices typically have information about the texture from different orientations. Therefore, a set of invariant features can be obtained by having several different co-occurrence matrices concatenated together.

Feature extractors related to the basic co-occurrence matrix include gray level difference histogram, Markov random field, gray level run length histogram, etc. The aforementioned methods all try to extract a set of compact yet highly meaningful feature (histogram) to distinguish between different textures.

Multi-Dimensional Histograms in Texture Modeling

The introduction of Gabor filters, and wavelets stimulated the further development of texture modeling using more complex image features. The methods can be summarized into multi-dimensional histograms as the images are often scaled to different level, and features are concatenated from multiple scaled images.

One of the most influential feature extraction method is called the local binary pattern, proposed in [54]. Its invention made a wide range of success in textural analysis due to the following two reasons:

- The method is computationally extremely light and simple, which shows the possibility of performing the calculations in real-time applications.
- The extracted binary features from the gray level texture images are extremely expressive in a compact vector, and thus made the method achieve high performances in texture analysis.

The local binary pattern was first developed to deal with textural images, and then expanded into various fields such as face recognition, etc. The original local binary pattern is defined over 3×3 pixel regions where the surrounding pixels over the center pixels are compared. First, the center pixel values are subtracted from the nearby pixels. Second, the surrounding pixels are binarized by setting the value of surrounding pixels to zero if the remaining values in that pixel is negative, one otherwise. Third, the binarized pixels are encoded by the multiplication of the power of 2 sequentially and then consequent values are added together. Eventually, the histogram of the calculated local binary patterns in the images are used as the features of the certain textures.

Furthermore, after the introduction of the original local binary patterns, various variants and extensions are developed to further explore the characteristics of the local binary patterns. One of the extensions is to compute the local binary patterns in an arbitrary distance; another extension is to compute the spatial-temporal local binary patterns in 3-D dimensional space.

3.4.2 State-of-the-Art Approaches

In recent years, there have been lots of advancements in the statistical modeling of conventional machine learning problems and methods. For instance, instead of hand-crafting features in computer vision, the deep network has been used to learn the natural feature representations of the object, and a series of state-of-the-art performances have been achieved in various complicated machine learning problems. This trend has also largely influenced texture modeling where the local information in the textures are vastly investigated. The statistical textural analysis is more robust in terms of the arbitrary viewpoint, and illumination changes. When the changes in the textural information provide 3-dimensional information, this type of textural classification is often called 3-D texture classification methods.

Filter Based Texture Classification

Given a class of textures, the task is generally to categorize the new textures into a pre-trained texture category. Therefore, in [67], the author proposed a series of standard texture classification methods based on the frequency responses of the textural information. To compute the feature sets from the texture, in training step, each texture is convoluted with a pre-defined set of filters, and then several clusters are formed using the k-means algorithms over all the filtered textures. The cluster centers are recorded and called textons accordingly.

In the testing step, the new texture images are then also filtered by the same set of filters, and categorized into the class using K-nearest neighbor classification.

In the whole process, one of the most important matters is to select the correct filters. The filter should be able to extract invariant features in terms of illumination, rotation, view-point changes, etc. Therefore, in [67], a rather successful texture filter bank called the maximum response filter sets is proposed. In the set, there are in total 38 different filters but only 8 filter responses. The texture filter bank consists of a Gaussian filter, a Laplacian of Gaussian filters, and 12 different edge filters with 3 different magnitudes. The visualization of the filters is shown in Figure 3.1.

Many different textures filters have been introduced in the literature. For more details, please see [67].

Filter Free Texture Classification

Filter bank methods are excellent in terms of texture classification. However, due to the complex procedure of the selection of texture filter banks, it should be ideal not to choose any filter bank but directly operate on the raw pixels. Under this perspective, a new texture classification method purely based on the texture patches is proposed in [66] where the filter banks are replaced by the clustered samples of patches of texture images.

While leaving most of the procedures of texture classification the same as that in the previous methods, the texture processing procedure is different: instead of filtering the texture images, a large set of texture patches are extracted from the images, and then some cluster centers are recorded for the large collections of texture patches. Consequently these cluster centers are treated as the new textons used for the texture classification.

The new approach is better in terms of the texture classification performance, while making the feature set smaller.

Random Projections in Texture Classification

In recent years, texture classification methods are further developed to analyze textural information using the state-of-the-art statistical approaches: Compressed sensing [22] and deep network [51].

In [36], a pioneering work on analyzing the textural information using the implicit mixture of Boltzmann machine is proposed. This method is great in terms of modeling complex generative models of different textures. Accordingly, the proposed method is extremely expressive in terms of generation of different texture images from scratch.

In [42], recently developed random projection methods are utilized in texture modeling. The proposed methods are capable of achieving a compatible state-of-the-art classification performance while preserving a small sized feature representation.

3.5 Texture Data Sets

Similar to the other computer vision research fields, there are also standardized data sets that are utilized in the field of texture modeling. During the different stages of textural information research, the following data sets in the literature have been widely used: the Brodatz texture data set, the KTH-TIPS (2) textural data set, the UIUC texture dataset, and the Columbia-Utrecht texture dataset. These data sets all resembles the rich diversity nature of texture by including varieties of different textures. Different textures also differ from the each other by the emphasis each data set presents.

Brodatz texture data set is the one of the oldest texture datasets that has been used in all the major texture modeling research. The Brodatz texture collection was first published in a graphical collection in [7] and then re-published in [8]. It contains 111 different texture images, and there is only one image for each texture class. For each texture image, there are no changes in illumination, view point, and affine transformations. Therefore, the Brodatz texture dataset is one of the simplest texture datasets. As there is only one texture image for each class, the trained texture classifier might not really generalize to other similar texture images in the same class. In a typical texture classification experiment, each image is divided into identical 9×9 subimages, and 5 of them are used to generate training patches, and the rest is used to generate testing patches. The aforementioned experiment is also used in this thesis.

One of the example patches for Brodatz texture images is to use a sub-collection of 24 images. These images are collected to prevent the some ambiguities in the original collections. Brodatz24 data sets are shown in Figure 3.2. There is only one image in each class, and thus in total there are 24 images.

KTH-TIPS (2) texture data set contains more variations. In each texture class, there are samples representing different illumination, pose, and scaling changes. This texture data set is more realistic in terms of considering the actual environmental changes in texture modeling. There are two different versions of KTH-TIPS dataset: one is the original data set, and other one is the superset of the original KTH-TIPS2 texture dataset which added some missing figures in the original texture dataset. Therefore, in real use, the KTH-TIPS2 data set is always used for its completeness. For clarifications, TIPS itself represents *Textures with varying Illuminations, Poses and Scales*.

In the KTH-TIPS2 data set, 11 different texture materials are sampled from different illuminations, poses and scales. For each texture, there are in total 108 different images, and each sample image is cropped to 200×200 size. Differently, in my experiments, half of the images are used to generate training samples, and the other half is used to generate the testing samples. A recent citation of the dataset appears for instance in [11]. A sample collection of KTH-TIPS2 is shown in Figure 3.3.

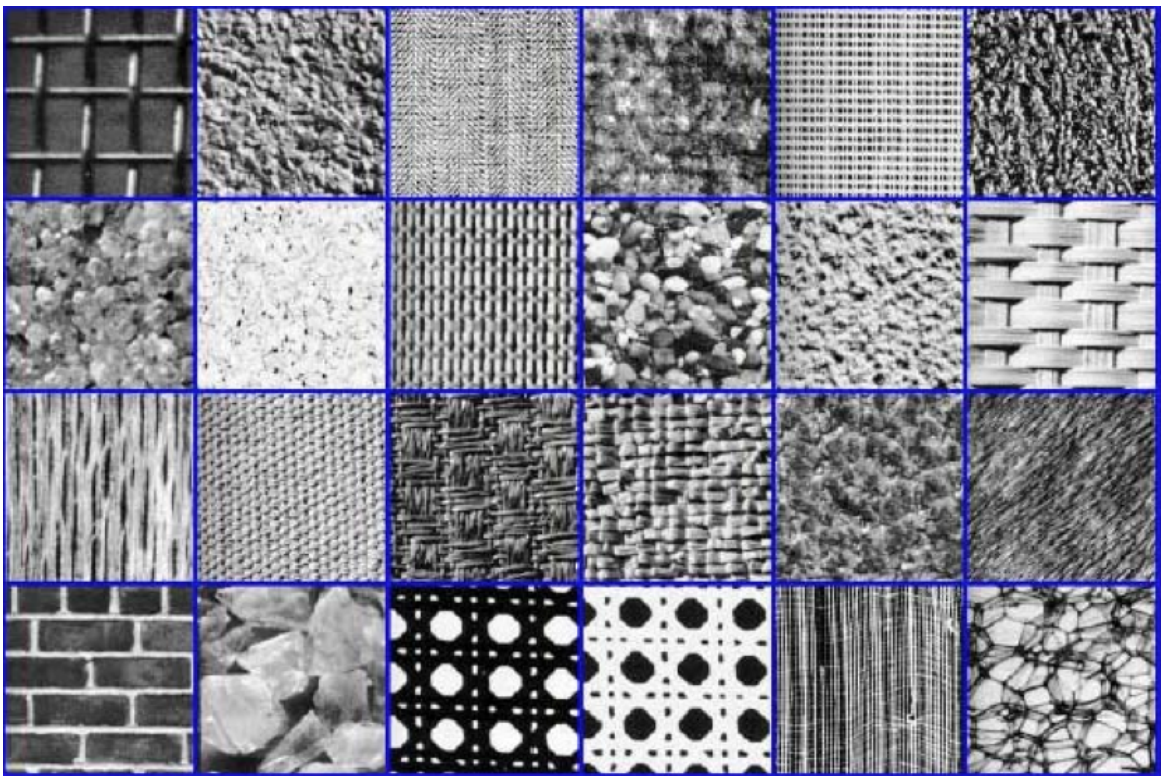


Figure 3.2: A subset collection of 24 images from Brodatz database [42].

UIUC texture database is yet another complicated texture dataset widely accepted as a standard texture modeling benchmarking data set. The texture dataset was introduced and first used in [39]. There are in total 25 different classes, and 40 images for each classes. Every image is grey-scaled with size 640×480. The significant changes in view-point and scales made the data set much more complicated than the Brodatz texture data set. UIUC texture is complicated in terms of the viewing angles, illumination changes. A sample collection of data set is shown in Figure 3.4

Last but not least, CUReT texture data set, or the Columbia-Utrecht data set, is one of the most comprehensive data sets for texture modeling. It was first introduced by a group of researchers in Columbia university and Utrecht university and published in [20]. There are in total 61 different surfaces being recorded in 92 different sample images. There are wide ranges of changes in illuminations, poses, etc. The CUReT data set is only used in validating the learning algorithms. A sample collection of several different textures in CUReT data set is shown in Figure 3.5.

3.6 Conclusion

Texture is a natural information on how the objects are presented in the world. In most cases, despite the change of the shape, illuminations, orientations, the texture information is consistently present among various similar objects. Given these facts, there has been

Figure 3.3: A subset collection of KTH-TIPS2 database.

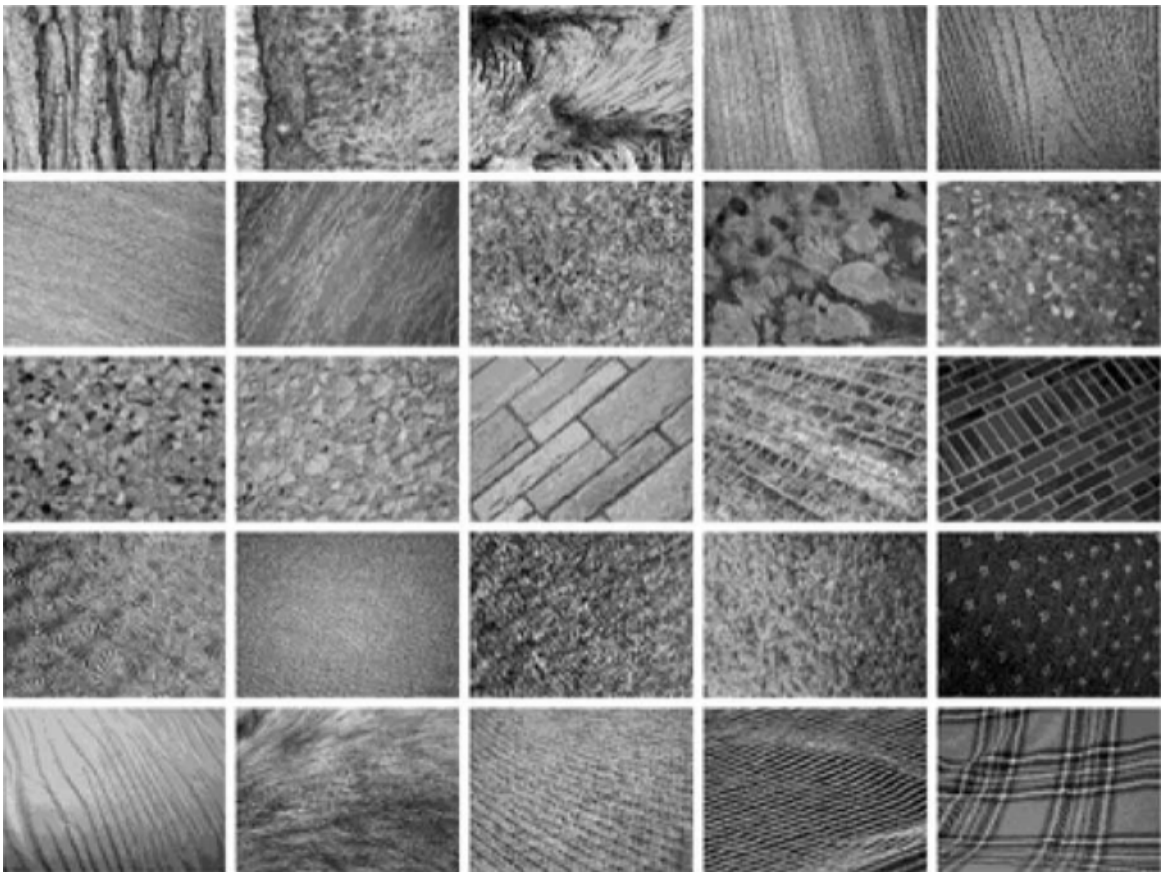


Figure 3.4: A sample collection of UIUC database [66].

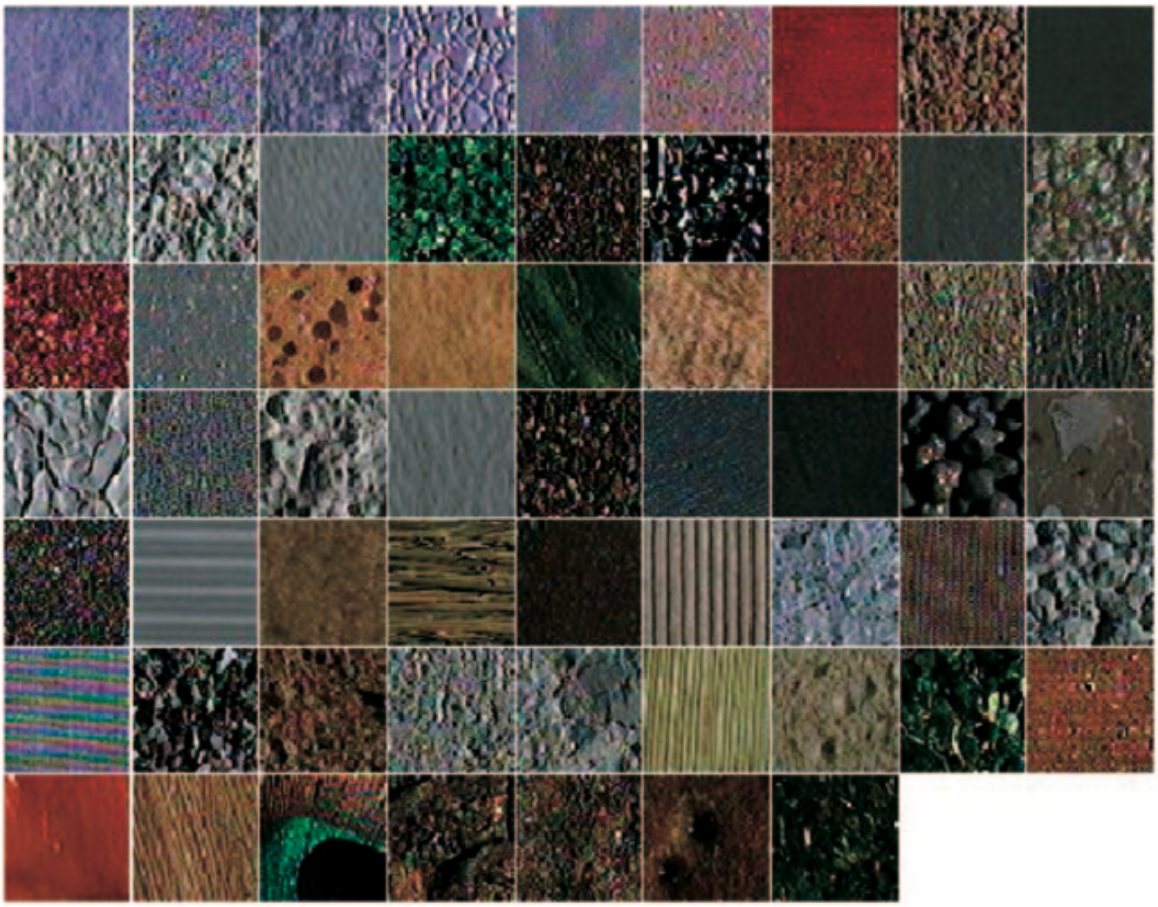


Figure 3.5: A sample collection of CUReT database [42].

enormous amount of research on how to represent, illustrate, recognize, and reconstruct the textural information for different objects. Among various texture learning algorithms, some of the most successful approaches are the statistical approaches. Simple statistics like co-occurrence matrices are already informative enough for recognizing the different textures in 2D settings. However, in order to increase the recognition capabilities, more advanced methods such as random projection, and recently restricted Boltzmann machine are applied. These methods are much more promising as they are more tailored to understand the complicated relationships between local textural information by exploring the interactions within the local patch.

This work is a new attempt to understand how the textural information can be understood using the local features of the texture by applying a convolutional higher order Boltzmann machine. Different from the previous approaches, we have a strong concentration on how the different pixels interacts, and only consider local interactions.

Chapter 4

Convolutional Gated Boltzmann Machine in Texture Modeling

4.1 Statistical Modeling in Computer Vision

Statistical modeling provides advantages in many machine learning problems. Comparing to the explicit modeling of the data, in recent years, statistical modeling provides flexibilities in modeling different types of data in a unified framework. In the conventional computer vision approaches, a series of hand-crafted features are designed for the extraction of the invariant features. Even though some satisfactory results have been achieved, these results are easily beaten by the natural features extracted from the objects using deep networks, for instance, in recognition of hand-written digits.

Therefore, in this work, following the recent work of modeling the texture using the implicit mixture of restricted Boltzmann machine [36], we propose to model the textural information using a convolutional gated Boltzmann machine, where features from different types of textures are controlled by the hidden variables. Different from the previous work, we don't need to learn class-specific features before mixing all the features. Instead, all the raw samples from different textures are fed to the learning machine, and a set of meaningful features will be presented in a reasonable learning time.

In the following chapters, we review the connection between our model and other higher-order Boltzmann machines, and then explain the detailed procedures designed for the proposed model.

4.2 Multi-view Feature Learning in Texture Modeling

Our model is based on the higher-order Boltzmann machine and the convolutional Gaussian restricted Boltzmann machine, both of which are explained in Chapter 2. The higher-order

Boltzmann machine is a new type of Boltzmann machine which tries to model the higher-order features in the interactions between two data vector $\mathbf{x}$ and $\mathbf{y}$. In the typical gated Boltzmann machine, the interactions terms $\sum_{ijk} x_i y_j h_k w_{ijk}$ are the key to learn the features. The tensor w_{ijk} connects the interacting features between x_i and y_j of data vectors $\mathbf{x}$ and $\mathbf{y}$ in hidden variables h_k. As the analytical conditional forms are still preserved, the inference in the model can be extremely fast. However, learning becomes extremely hard because the number of free parameters in the tensor w_{ijk} increased dramatically w.r.t. the size of inputs. To cope with this, a factorization scheme is proposed

$$w_{ijk} \to \sum_f w_{if}^x w_{jf}^y w_{kf}^h$$

In this way, the number of free parameters is reduced dramatically. However, this factorization actually gives rise to another problem: while feeding the same patches to $\mathbf{x}$ and $\mathbf{y}$ in some learning problems, as we have different sets of weight matrices for $\mathbf{x}$ and $\mathbf{y}$, the samples from the model distribution can hardly reflect this situation.

4.3 General Gaussian Gated Boltzmann Machine: Revisited

Our approach stems from a simple intuition: the local interactions between nearby pixels are far more important and dominant so that one can basically ignore the far away pixels to some extent. In order to dive into this, a bottom-up approach is taken to build an appropriate energy function to describe the interactions. Before that, a more detailed explanation of general Gaussian gated Boltzmann machine is presented.

Figure 4.1 shows a simple example of undirected graph about the relationship between pairs of input vectors and the hidden vectors. In addition to the Gaussian gated Boltzmann machine, there are terms to specify the relationship between the input vectors. In other words, every node is inter-connected by an undirected graph.

However, this general model is too general to be used for modeling texture. Given two image patches, not all the interactions even make sense. For instance, it doesn't seem to be useful to consider the interactions between pixels that are far away enough from each other, e.g., pixels from two corners.

A standard general Gaussian gated Boltzmann machine can be defined as

$$\begin{aligned} E(\mathbf{x}, \mathbf{y}, \mathbf{h}) = &- \sum_{ijk} \frac{x_i}{\sigma_i} \frac{y_j}{\sigma_j} h_k w_{ijk} - \sum_{ij} \frac{x_i}{\sigma_i} \frac{y_j}{\sigma_j} u_{ij} + \sum_i \frac{(x_i - b_i^x)^2}{2\sigma_i^2} \\ &+ \sum_j \frac{(y_j - b_j^y)^2}{2\sigma_j^2} - \sum_k h_k c_k - \sum_{ik} \frac{x_i}{2\sigma_i^2} h_k v_{ik}^{(1)} - \sum_{jk} \frac{y_j}{2\sigma_j^2} h_k v_{jk}^{(2)} \end{aligned} \tag{4.1}$$

where u_{ij}, $v_{ik}^{(1)}$ and $v_{ik}^{(2)}$ are additional parameters to model the pair-wise connections between two sets of visible neurons $\{\mathbf{x}, \mathbf{y}\}$ and hidden neurons $\mathbf{h}$. Both $\mathbf{x}$ and $\mathbf{y}$ are considered to be Gaussian variables. The formulation is actually identical to the Boltzmann machine where

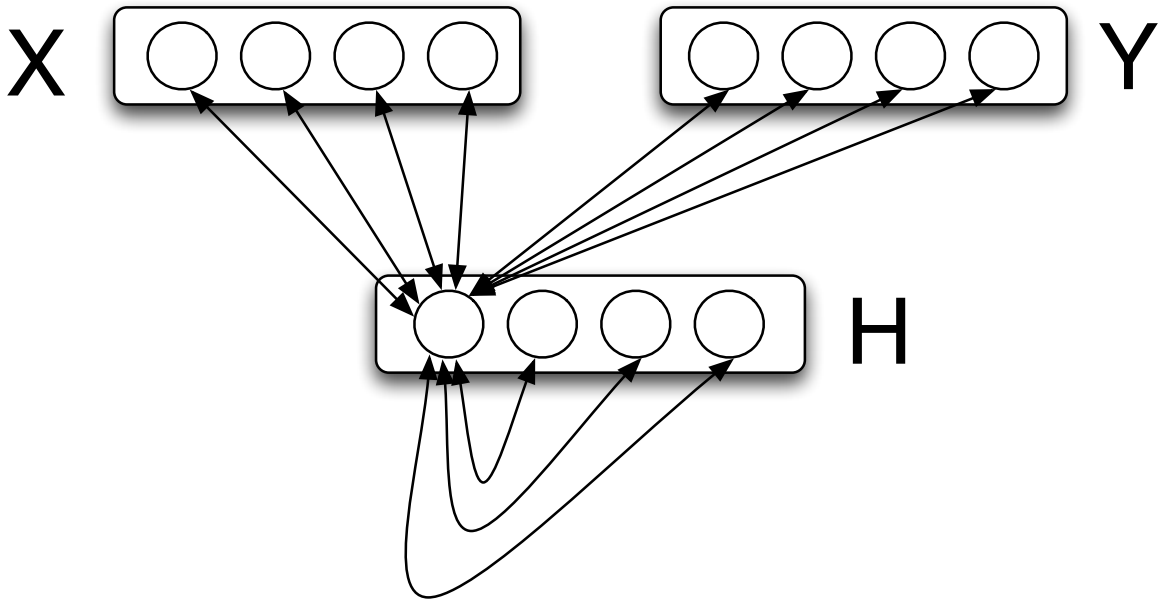


Figure 4.1: The graphical illustration of general gated Boltzmann machine

there is no distinction between visible neurons and hidden neurons. What's different here, is that we explicitly wrote out three sets of neurons, each having a special meaning.

This explicit formulation inspired us to dive into the formulation of the local interactions of the neurons by considering a convolutional model within the pairs of visual neurons.

4.4 Convolutional Gated Boltzmann Machine

Combining the nature of texture information and Gaussian gated Boltzmann Machine, a modified Gaussian gated Boltzmann machine especially suitable for texture modeling is proposed. The proposed method is particularly suitable for capturing the local relations within image patches. One can argue that modeling explicitly the relationship between pairs of pixels can be potentially beneficial in texture modeling.

To start with, we consider a slightly modified general gated Boltzmann machine where there are pairwise connections between all sets of nodes. This model has the most comprehensive information about the input vectors. Accordingly, the energy function of the model is

$$\begin{aligned} E(\mathbf{x},\mathbf{y},\mathbf{h}) = &- \sum_{ijk} \frac{x_i}{\sigma_i}\frac{y_j}{\sigma_j} h_k w_{ijk} - \sum_{ij} \frac{x_i}{\sigma_i}\frac{y_j}{\sigma_j} u_{ij} + \sum_i \frac{(x_i - b_i^x)^2}{4\sigma_i^2} \\ &+ \sum_j \frac{(y_j - b_j^y)^2}{4\sigma_j^2} - \sum_k h_k c_k - \sum_{ik} \frac{x_i}{2\sigma_i^2} h_k v_{ik}^{(1)} - \sum_{jk} \frac{y_j}{2\sigma_j^2} h_k v_{jk}^{(2)} \end{aligned} \tag{4.2}$$

Similarly, where u_{ij}, $v_{ik}^{(1)}$ and $v_{ik}^{(2)}$ are additional parameters to model the pair-wise connections between two sets of visible neurons $\{\mathbf{x},\mathbf{y}\}$ and hidden neurons $\mathbf{h}$.

It is clear that, the modified general gated Boltzmann machine has similar structure as the original one, but with a few modifications. These changes are made to ease the derivation in the later section only, and the properties of the Boltzmann machine should remain the same. From the above energy model, it is shown that the interactions between two input vectors are counted. Additionally, each input vector and the hidden vector should have bias terms. As before, the general gated Boltzmann machine is too general for representing textural information given the difficulties arising in for learning such models.

Instead, given our assumptions about the data and problem, there is actually a lot of room for streamlining the model. In our context, we are aiming to look for the inner relationship within the images, and this results that the input vectors $\mathbf{x}$ and $\mathbf{y}$ are actually getting the same input. Besides that, given the fact that $\mathbf{x}$ and $\mathbf{y}$ are actually requiring the model to feed same input, the characteristics of the two input vectors should also follow some conventions: they should share the same variance; they should have the same mean value.

From this direction, several crucial simplifications and omissions have been done to make the general gated Boltzmann machine meaningful and useful yet preserving the abilities to model the relationships by this particular model. In the following paragraphs, the detailed modifications and reasons are addressed.

Instead of looking for the image transformation, we seek for the internal structure of texture information. Therefore, the same patch of image is fed to the two sets of visible neurons, that is $\mathbf{x} = \mathbf{y}$. Accordingly, the weights $\mathbf{v}$ and bias $\mathbf{b}$ for the two sets of visible neurons are tied, which is

$$\mathbf{V} = \mathbf{V}^{(1)} = \mathbf{V}^{(2)} \ ; \mathbf{b} = \mathbf{b}^x = \mathbf{b}^y$$

Also, a unified variance

$$\sigma^2 = \sigma_i^2 = \sigma_j^2$$

is learned to reduce the complexity of the model further. This simplifications reduced the energy function 4.2 to

$$E(\mathbf{x}, \mathbf{y}, \mathbf{h}) = -\sum_{ijk} \frac{x_i}{\sigma} \frac{y_j}{\sigma} h_k w_{ijk} - \sum_{ij} \frac{x_i}{\sigma} \frac{y_j}{\sigma} u_{ij} - \sum_k h_k b_k^h + \sum_i \frac{(x_i - b_i^x)^2}{2\sigma^2} - \sum_{ik} \frac{x_i}{\sigma} h_k b_{ik}^{xh}$$

where the formulations are much clearer and representative. We can come to the conclusion that the formulation is actually a combination of two different models: a normal Gaussian restricted Boltzmann machine and a higher order Boltzmann machine characterized by the triplet terms. This actually gave us quite a lot of inspirations about how we should continue our optimization and simplification. In a recognition task, it doesn't really matter if one actually learns the exact model, as long as the approximated model is capable of producing a set of meaningful features that distinguishes different objects. Therefore, in the following sections, there is a further discussion on how the whole modified general gated Boltzmann machine is useful in terms of modeling the relationship within the images.

4.4.1 Convolutional Transformation in GGBM

The complexity of the model remains as the weight tensor w_{ijk} still needs huge learning efforts. As $\mathbf{x} = \mathbf{y}$, the inputs x_i and y_j can be considered a pair of pixels, and the hidden neurons h_k are learned to model this interaction. Given an image patch, the traditional Gaussian gated Boltzmann machine will go through all the combinations of such pairs. This is highly redundant as the texture is repetitive within a very small region. Recalling that co-occurrence matrix tries to summarize the interaction of pairs of pixels over a certain area, this structure can be introduced to Gaussian gated Boltzmann machine. d represents the max offset distance vector from the center x_i to a nearby pixel in a grid of $\{d, d\}$. Please note that we only include terms where pixel y_{i+d} stays within the image patch.

The distance constant d is defined particularly to force the modeling of the interaction between different pixels in a tiny local area. In quite a few experiments, d is defined to be not larger than 20, while the image size can actually be arbitrarily big. This idea follows seamlessly with the previously proposed convolutional restricted Boltzmann machine, where the number of the hidden neurons is rather limited to constrain the learning efforts on the actual local area. In convolutional restricted Boltzmann machine, there are K different hidden neuron sets with size smaller than the actual image patch sizes. Given our model and the local concentration, a set of hidden vectors are also present in our model to capture the multiple features in a single setting. Similar to the convolutional restricted Boltzmann machine where the weight matrix $\mathbf{W}$ is shared between different hidden neurons, we also achieved this characteristics, which is derived in detail in the following chapters. This formulation has multiple benefits regarding the learning of the model and interpretation. Given a smaller weight matrix, the number of the free parameters shrank to a small amount, and such that the learning is easier, and less dangerous to over-fit. Additionally, as most of the data we are now facing has a strong manifold structure, it is essentially important to understand the structure of the data. The way the local concentrations are considered in our model forces the model to only consider the local distance. This follows the manifold properties where only the local distance is valid. Different from the convolutional restricted Boltzmann machine, the dimensionality of the weight tensor in our model is also decreased to 2, which made the whole implementation much easier.

In the following paragraph, the simplifications and derivations are addressed. We will assume

$$w_{ijk} \rightarrow w_{dk}$$

such that the weight w_{ijk} depends only on the displacement d and the hidden neuron h_k. d represents the offset from i to j. Similarly,

$$u_{ij} \rightarrow u_d$$

One can think of w_{dk} and u_d as a convolutional model only over the local regions in image patches. Convolutional approximation has been argued to be rather successful in other applications such as image recognition tasks [40]. It is further assumed that $w_{dk} = 0$ for large displacement d. As the size of the image patches is often larger than the size of the

$d \times d$, the weight vector $\mathbf{W}_{dk}$ is shared between different location of the image patches, and thus forcing the number of free parameters to a normal level.

After these simplifications, the energy function (4.2) becomes

$$\begin{aligned} E(\mathbf{x}, \mathbf{y}, \mathbf{h}) = &- \frac{1}{\sigma^2} \sum_{ijk} x_i y_j h_k w_{d_{ij}k} - \frac{1}{\sigma^2} \sum_{d} x_i y_j u_{d_{ij}} + \frac{1}{2\sigma^2} \sum_{i} (x_i - b_i)^2 \\ &- \frac{1}{\sigma^2} \sum_{ik} x_i h_k v_{ik} - \sum_{k} h_k c_k \end{aligned} \tag{4.3}$$

Learning and inference of Gaussian gated Boltzmann machine can be based on sequentially sampling from the conditional distributions $p(\mathbf{x}|\mathbf{y}, \mathbf{h})$, $p(\mathbf{y}|\mathbf{x}, \mathbf{h})$ and $p(\mathbf{h}|\mathbf{x}, \mathbf{y})$. As $\mathbf{x}$ is assumed to have the same image patches as $\mathbf{y}$, $p(\mathbf{x}|\mathbf{y}, \mathbf{h}) = p(\mathbf{y}|\mathbf{x}, \mathbf{h})$ should be enforced.

These conditional forms can all be written in closed forms as

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}, \mathbf{h}) &= \prod_{j} \mathcal{N}\left(b_j + \sum_{ik} x_i h_k w_{ijk} + \sum_{i} x_i u_{ij} + \sum_{k} h_k v_{jk}, \sigma^2 \right) \\ p(\mathbf{x}|\mathbf{y}, \mathbf{h}) &= \prod_{i} \mathcal{N}\left(b_i + \sum_{jk} y_j h_k w_{ijk} + \sum_{j} y_j u_{ij} + \sum_{k} h_k v_{ik}, \sigma^2 \right) \\ p(\mathbf{h}|\mathbf{x}, \mathbf{y}) &= \prod_{k} \frac{1}{1 + \exp\left(-\frac{1}{\sigma^2} \sum_i x_i v_{ik} - \frac{1}{\sigma^2} \sum_j y_j v_{jk} - \frac{1}{\sigma^2} \sum_{ij} x_i y_j w_{ijk} - c_k^h \right)}. \end{aligned} \tag{4.4}$$

However, simply alternatively sampling from the conditional probability distribution can hardly make learning to progress further. The reason is that the prerequisite that $\mathbf{x}$ should have the same fantasy particles $\mathbf{y}$, can be be satisfied on every iteration as there is no enforcement about this constraint in the conditional forms. Therefore, to make the learning simple yet effective enough, some approximation is made based on the similarities between this model and other existing models. Such an approximation is made that, the learned features from the approximated model can actually represent the original model well enough, therefore the learned features can directly be used on the original model, and this is partially proved by the texture reconstruction experiment.

In the next subsection, the approximation and derivations are introduced. Note that all the approximations and simplifications are empirically shown to be correct. The more detailed experiments are present in next chapter.

4.4.2 GRBM with Preprocessing Units

By considering the simplified model of the combinations of the normal Gaussian restricted Boltzmann machine and some other additional terms, our model can actually be simplified to be learned by a Gaussian restricted Boltzmann machine.

Essentially, we are now modeling the textural information and its local interactions by the proposed convolutional gated Boltzmann machines. Therefore, we can actually consider preprocessing the inner relationship terms in a sensible way such that the proposed model can

be modeled by a Gaussian restricted Boltzmann machine. To this end, several modifications, addressed below are made.

Firstly, we define auxiliary variables

$$t_d = \sum_i x_i y_{i+d}$$

where d is the offset between pixels i and j as before. This formulation stems from the principle of the co-occurrence matrix where each feature is only related to particular pairs of pixels in the image. These computations can be done as a preprocessing step. Secondly, we learn a GRBM using the concatenation of vectors $[\mathbf{x}, \mathbf{t}]$ as data. We call this model the GRBM(X,T) and illustrate it in Figure 4.2. In the figure, the dashed line represents $\mathbf{t}$ being computed from $\mathbf{x}$.

When we write the energy function of GRBM(X,T)

$$\begin{aligned}\mathbf{E}(\mathbf{x}, \mathbf{t}, \mathbf{h}) = &-\frac{1}{\sigma^2}\left(\sum_{ik} x_i h_k v_{ik} + \sum_{dk} t_d h_k w_{dk}\right) - \sum_k h_k c_k \\ &+ \frac{1}{2\sigma^2}\left(\sum_i (x_i - b_i)^2 + \sum_d (t_d - u_d)^2\right),\end{aligned} \tag{4.5}$$

we notice the similarities to the GGBM energy function in Equation (4.3). Each parameter has its corresponding counterpart. The only remaining difference is

$$\mathbf{E}(\mathbf{x}, \mathbf{t}, \mathbf{h}) - \mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \frac{1}{2\sigma^2}\sum_d t_d^2 + \text{const} \tag{4.6}$$

t_d^2 is constant in terms of the learning parameters and such that during the learning time, the difference can be omitted. These facts give us the opportunities to approximate a complex learning algorithm well enough by considering a much simpler and well-established learning problem.

It turns out $p(\mathbf{h}|\mathbf{x}, \mathbf{y})$ can be written in the exact same form as in Equation (4.4). This approximation makes the learning of the model extremely easy. After the simplifications, the proposed energy function (4.3) can be written as,

$$\begin{aligned}\mathbf{E}(\mathbf{m}, \mathbf{h}) = &-\frac{1}{\sigma^2}\sum_{ik} m_i h_k v_{ik} - \sum_k h_k c_k \\ &+ \frac{1}{2\sigma^2}\sum_i (m_i - b_i)^2\end{aligned} \tag{4.7}$$

where $\mathbf{m}$ represents the concatenation of the $\mathbf{x}$ and $\mathbf{t}$. The only difference between equations (4.3) and (4.7) is that $\frac{1}{\sigma^2}\sum_d t_d b_d$ is replaced by $\frac{1}{2\sigma^2}\sum_{i,p_i=t_i}(m_i - b_i)^2$. As t_d is constant *w.r.t.* to the learning parameters and b_d can be removed by normalization constant $\mathbf{Z}$, in this particular case, our proposed model can be approximated by a much simpler model.

Since learning higher order Boltzmann machines is known to be quite difficult, we propose to use this approximated model as a way for learning them. So in practice we first train a GRBM(X,T), and then convert the parameters

$$w_{dk} \rightarrow w_{ijk}$$

and

$$u_d \rightarrow u_{ij}$$

to the convolutional gated Boltzmann machine model.

Actually, in texture classification, the converted model produces exactly the same hidden activations $\mathbf{h}$ and thus the same classification results should be expected. On the other hand, in the texture reconstruction problem, the GRBM(X,T) model cannot be used directly, since $\mathbf{t}$ cannot be computed from partial observations.

We noticed experimentally that the converted GGBM model needs to be further regularized, since the regularizing terms t_d^2 in the energy function of GRBM(X,T) are dropped off as seen in Equation (4.6). We simply converted w_{dk} and u_d by scaling them with a constant factor smaller than 1, and chose that constant by the smallest validation reconstruction error. The constant here work as a "normalization" factor in a sense that, we are trying to minimize the difference between the exact model and the approximated model.

The introduction of the $\mathbf{t}$ in the data representation essentially increased our feature abstraction level while decreasing the complexity of the model. According to the definition of the $\mathbf{t}$, it represents the local convolutional features in the image patches, and potentially has a higher representational power than the normal $\mathbf{y}$ image patches. While at the same time, the introduction of $\mathbf{t}$ decreased the dimensionality of the weight matrix from a three dimensional tensor to a two dimensional matrix, which obviously decreases the difficulties of the learning.

The proposed learning algorithm is explained in summarized in Algorighm 3. In general, the structure of the proposed model is shown in Figure 4.2. Given a set of input vectors $\mathbf{x}$, a new view of input vector is obtained in $\mathbf{t}$. Afterwards, Equation (4.7) is trained jointly using $\mathbf{x}$ and $\mathbf{t}$. Eventually, the weight matrix is converted back to original form in Equation (4.3).

Algorithm 3 Texture learning algorithm

Input: input image $\mathbf{x}_0$, $\mathbf{y}_0$ where $\mathbf{x}_0 = \mathbf{y}_0$. Proposed textural model $GGBM(\mathbf{x}, \mathbf{y}, \mathbf{h})$, number of iteration k

Output: The shared weight matrix and vector $\mathbf{W}$ and $\mathbf{b}$

feed to $GGBM(\mathbf{x}, \mathbf{y}, \mathbf{h})$ with $\mathbf{x}_0$, $\mathbf{y}_0$

compute convolutional feature $\mathbf{t}$ from $\mathbf{x}$ and $\mathbf{y}$

while iteration smaller than k **do**

compute $P(\mathbf{h}|\mathbf{x}, \mathbf{t})$

compute $P(\mathbf{x}|\mathbf{h}, \mathbf{t})$ and $P(\mathbf{t}|\mathbf{h}, \mathbf{x})$

Update $\mathbf{W}, \mathbf{b}, \mathbf{c}, \sigma$ by gradient ascent $\theta \leftarrow \theta + \eta\nabla\theta$ according to update rules in [14]

continue iteration

end while

4.4.3 Texture Reconstruction

Generative model is capable of understanding the structure of the target objects. This differs fundamentally from the discriminative models, because the discriminative models consider learning discriminative representations of object such that one can easily distinguish them based on those features.

Therefore, in this section, we introduce an approximate way of reconstructing the textural information. The idea is that, given a corrupted texture image, the model can generate a reconstructed texture image based on the corrupted image.

In the proposed model, the image patches are fed to and represented in the model by $\mathbf{x}$ and $\mathbf{y}$. The learned feature representations are stored in $\mathbf{h}$. The feature vector $\mathbf{h}$ restores the generative models of the images along with the weight matrices $\mathbf{W}$ and biases $\mathbf{b}, \mathbf{c}$ in the proposed model.

Remember that the corrupted image is fed to $\mathbf{x}$ and $\mathbf{y}$, Texture reconstruction can be done by alternatively sampling from $P(\mathbf{x}|\mathbf{y}, \mathbf{h})$, $P(\mathbf{y}|\mathbf{x}, \mathbf{h})$ and $P(\mathbf{h}|\mathbf{x}, \mathbf{y})$. These conditional forms can all be written in a closed form. The conditional forms are then calculated as

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x}, \mathbf{h}) &= \prod_j \mathcal{N}\left(b_j + \sum_{ik} x_i h_k w_{ijk} + \sum_i x_i u_{ij} + \sum_k h_k v_{jk}, \sigma^2\right) \\
p(\mathbf{x}|\mathbf{y}, \mathbf{h}) &= \prod_i \mathcal{N}\left(b_i + \sum_{jk} y_j h_k w_{ijk} + \sum_j y_j u_{ij} + \sum_k h_k v_{ik}, \sigma^2\right) \\
p(\mathbf{h}|\mathbf{x}, \mathbf{y}) &= \prod_k \frac{1}{1 + \exp\left(-\frac{1}{\sigma^2}\sum_i x_i v_{ik} - \frac{1}{\sigma^2}\sum_j y_j v_{jk} - \frac{1}{\sigma^2}\sum_{ij} x_i y_j w_{ijk} - c_k^h\right)}
\end{aligned}
$$

The detailed algorithm for texture reconstruction is shown in Algorithm 4. The basic idea is to only update the corrupted pixels in the inference steps while keeping other pixels the same as before. To obtain better reconstruction, it might be a good idea to use the average of 100 reconstructed images.

4.5 Conclusions

A new type of convolutional gated Boltzmann machine is introduced. Different from the previous convolutional gated Boltzmann machine, this particular type of Boltzmann machine is aimed to modeling the local relationship within image patches, while other models in the literature purely concentrated on the modeling of the image pairs or the full patch.

Our model is designed in the context of texture analysis. A similar model in texture analysis, separate Boltzmann machine is learned for each class of textures, and then a mixture of Boltzmann machine is presented. Clearly, the way they treat the texture problem is that, different textures have distinctly different texture information. However, due to the fact that, in the real world, textural information is mixed, and can hardly separate them into classes before training. Therefore, our method tries to tackle this problem by considering the

Algorithm 4 Texture reconstruction algorithm

Input: corrupted image $\mathbf{x}_0$, $\mathbf{y}_0$ where $\mathbf{x}_0 = \mathbf{y}_0$. Proposed textural model $GGBM(\mathbf{x}, \mathbf{y}, \mathbf{h})$, number of iteration k

Output: reconstructed image $\mathbf{x}_\infty$ and $\mathbf{y}_\infty$

feed to $GGBM(\mathbf{x}, \mathbf{y}, \mathbf{h})$ with $\mathbf{x}_0$, $\mathbf{y}_0$

while iteration smaller than k **do**

compute $P(\mathbf{h}|\mathbf{x}, \mathbf{y})$

compute $P(\mathbf{x}|\mathbf{h}, \mathbf{y})$

obtain $\mathbf{x}_k, \mathbf{y}_k$ by setting the corrupted pixels by the value of $P(\mathbf{x}|\mathbf{h}, \mathbf{y})$ and the rest remains the same.

continue iteration

end while

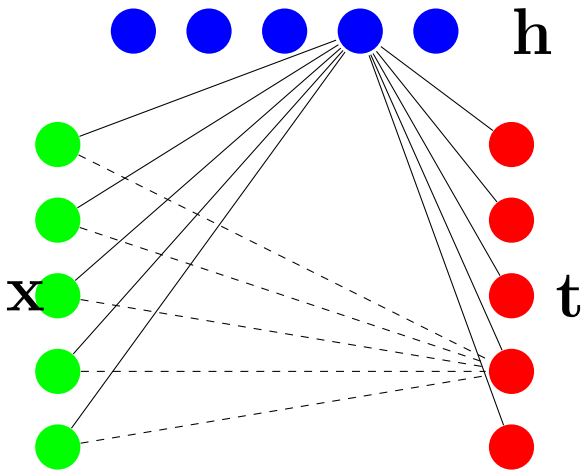


Figure 4.2: The schematic illustration of the convolutional gated Boltzmann machine

texture information as a whole, and let the model to differentiate the difference between the textures. To do this, a hidden layer **h** is defined to capture the different local interactions between pairs of local image patches.

In this chapter, the theoretical analysis and derivation of the model is addressed. In the next chapters, a few proof-of-concept experiment are carried out on several standard texture datasets.

Chapter 5

Experiments using Convolutional Gated Boltzmann Machine

5.1 Experiment Settings

In this chapter, experiments and their detailed explanations and interpretations are addressed. The data used in the experiments have been largely used in the previous scientific literature in textural modeling and deep learning.

Based on the nature of the proposed model, it is trying to learn the generative model based on the given data. Therefore, the proposed methods are tested on two different tasks. One is texture recognition. A texture recognizer can be built based on features learned from the data. For this particular task, the better recognition capability the model produces, the better the learned model is. Secondly, a texture reconstruction task will be performed to understand the capabilities of learned generative models in the proposed methods. For this particular task, one can compare the performance of the model based on the visual outcome of the model.

In the following sections, the learned feature representations of the textures are compared. Also, the recognition results and reconstruction results are shown in texture reconstruction experiments.

5.2 Texture Features

It was mentioned in the previous sections that the model we proposed is capable of capturing the textural informations in a hidden feature $\mathbf{h}$. In general, one of the most convenient ways of identifying the learned results of the hidden feature sets $\mathbf{h}$ is to visually inspect the learned feature weight matrix $\mathbf{W}$. By definition, given the length of the input vector as $N \times N$, and the length of the hidden vector as $K \times K$, the hidden feature weight vector can be visualized using a grid of $K \times K$ weight patches with size $N \times N$.

In our experiment, the size of the image varied, but the size of the hidden features are

fixed with size 1000. This number has no particular meaning in terms of the selection. In practice, after the learning, the model is capable of determining the size of the hidden neurons automatically by omitting the hidden vectors with small enough L2 norm.

In our experiment, the following data sets are used:

- Brodatz24 data set. One of the most standard texture data used in the field of texture analysis. Each texture only contains one image, and thus the problem is rather simple.
- KTH-TIPS2-A data set. The data set I used here is a subset of the standard KTH-TIPS2 data, where only the collection A is used in training. The problem is much harder here as the texture images have rotations, and illumination changes.
- CUReT data set. This is an advanced data set for texture modeling, and can be treated as one of the most advanced texture data set in the field for now.
- UIUC data set. UIUC data is yet another standard data set used in the texture analysis.

In this section, the different learned visual filters from different data sets are shown. Given $P(\mathbf{h}|\mathbf{x}, \mathbf{y})$ is characterized by the weight matrix $\mathbf{W}$ and accompanying $\mathbf{b}$. The performances of different methods largely differ between the learned hidden features.

In all our experiments, the size of the image patches is chosen to be 20×20. Also, the displacement distance t is chosen to be 5, such that the size of the local patch is set to 11×11. Also, the size of the hidden layer is chosen to be 1000. Brodatz 24 data set is the simplest texture data sets in this experiment. We tried to extract a set of 1000 meaningful hidden features. As we discussed in the previous chapter, our proposed model is approximated by a modified Gaussian restricted Boltzmann machine, where only the inputs of the models are different, namely the raw image patches, the transformed convolutional images, and the concatenations of the previous two images per sample. The features shown in this section represents essentially the weight matrix in different models. The weight matrix is used to characterize the conditional probability $P(\mathbf{h}|\mathbf{x}, \mathbf{y})$. In Fig 5.1, 1000 weight matrices are shown for the raw texture image patches. For comparison, Fig 5.2 shows the weight matrix characterizing for the convolutional image patches. In Fig 5.3 and 5.4, the weight matrix for characterizing the approximation of the proposed model is compared. It is clearly seen that, the proposed method is capable of producing a clearer feature representation.

Similarly, feature representation for the KTH data set and the CUReT data set are shown in Fig 5.5, 5.6, 5.7 and 5.8, as well as Fig 5.9, 5.10, 5.11 and 5.12 respectively. In all these cases, introducing the convolutional feature representation of the image, gave the possibility of making the learned feature more distinct, and visible, which potentially made the recognition ability of the whole system better.

5.3 Texture Classification

One of the main goal of our model is to use it for recognizing unseen texture images as accurately as possible. Therefore, in this section, a few texture classification comparison

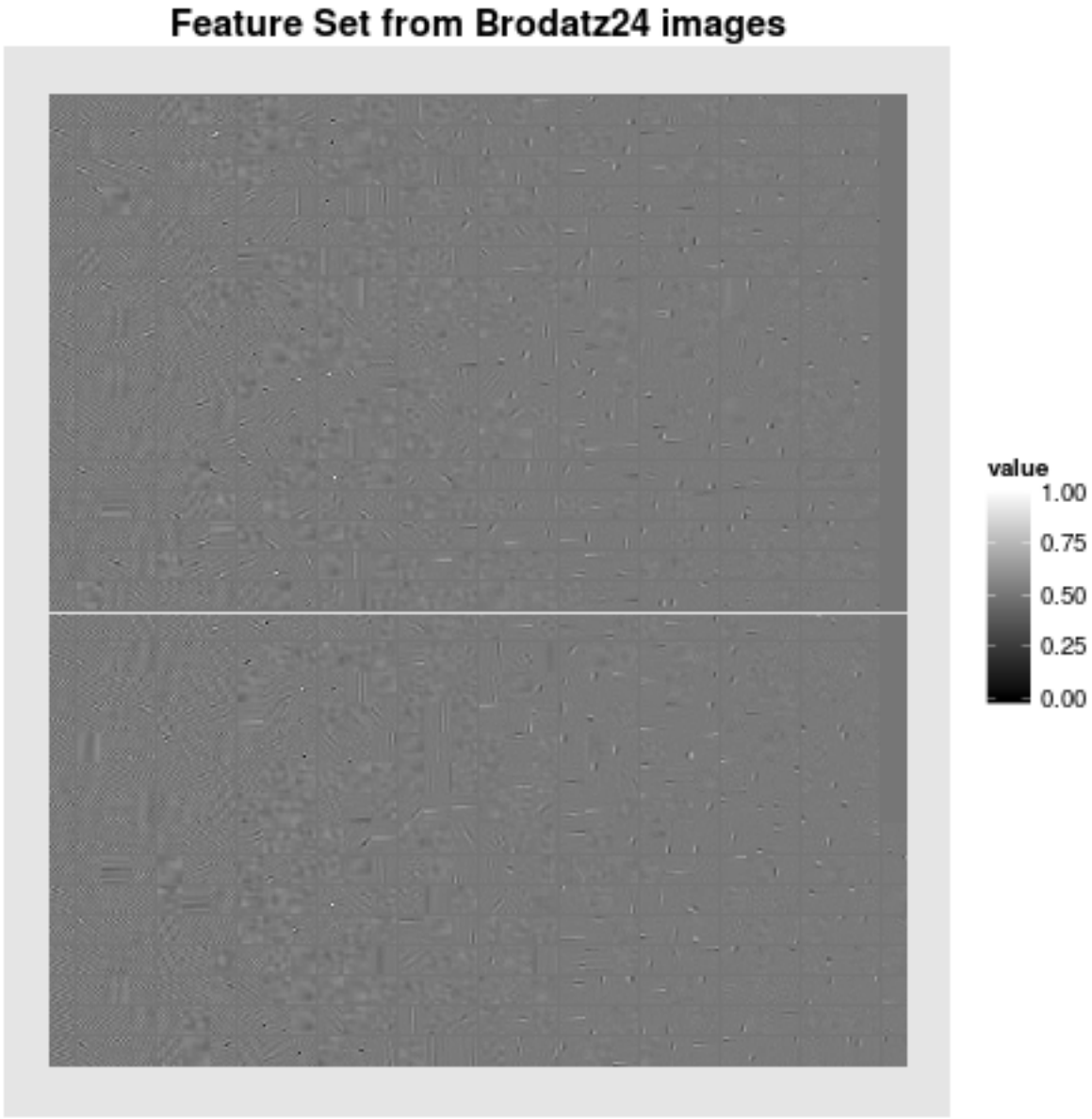


Figure 5.1: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine for Brodatz24 data set

Figure 5.2: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine on the convolutional feature $\mathbf{t}$ for Brodatz24 data set

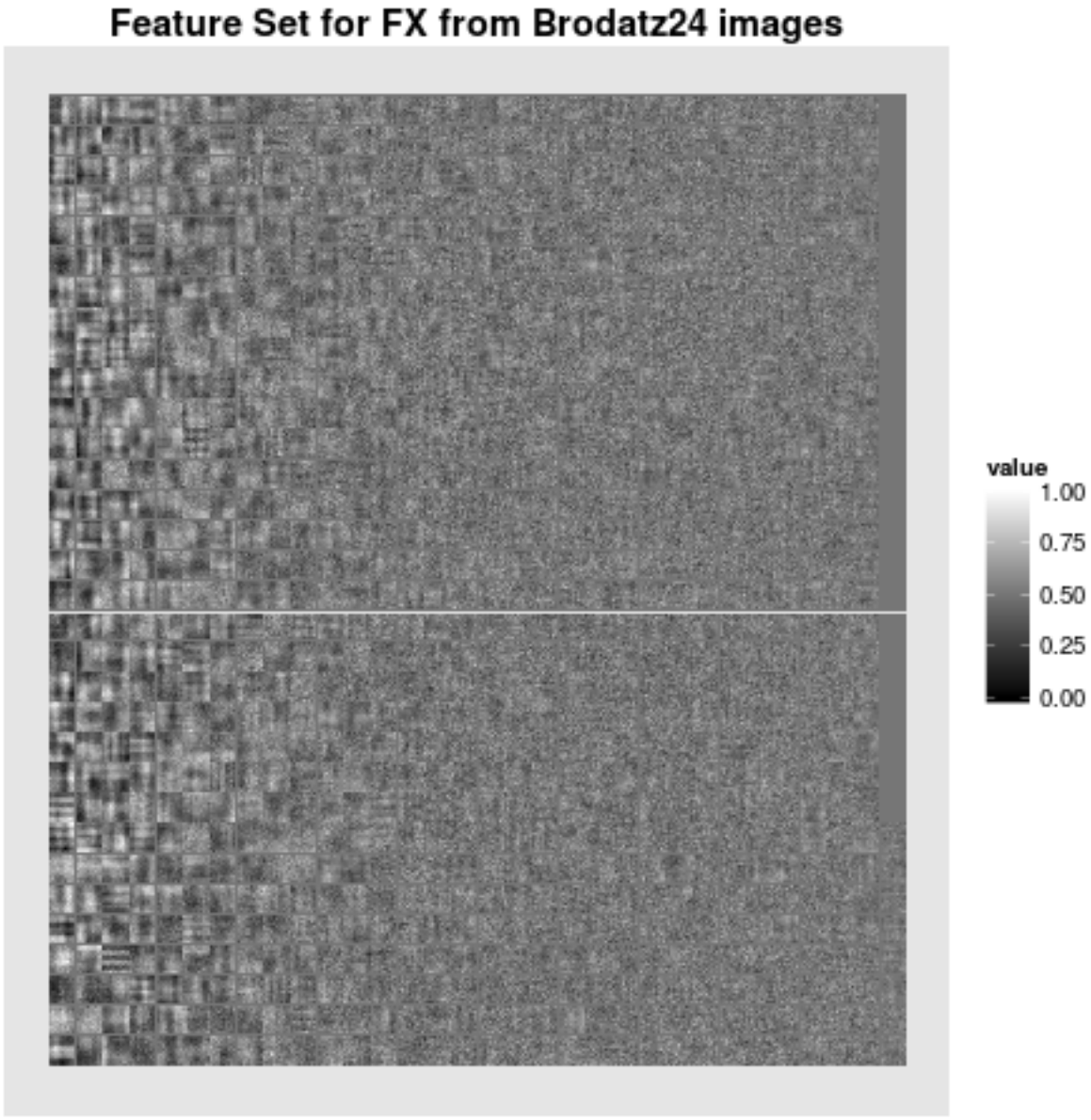


Figure 5.3: The visual filter learned for the normal image patches $\mathbf{x}$ by the proposed model for Brodatz24 data set

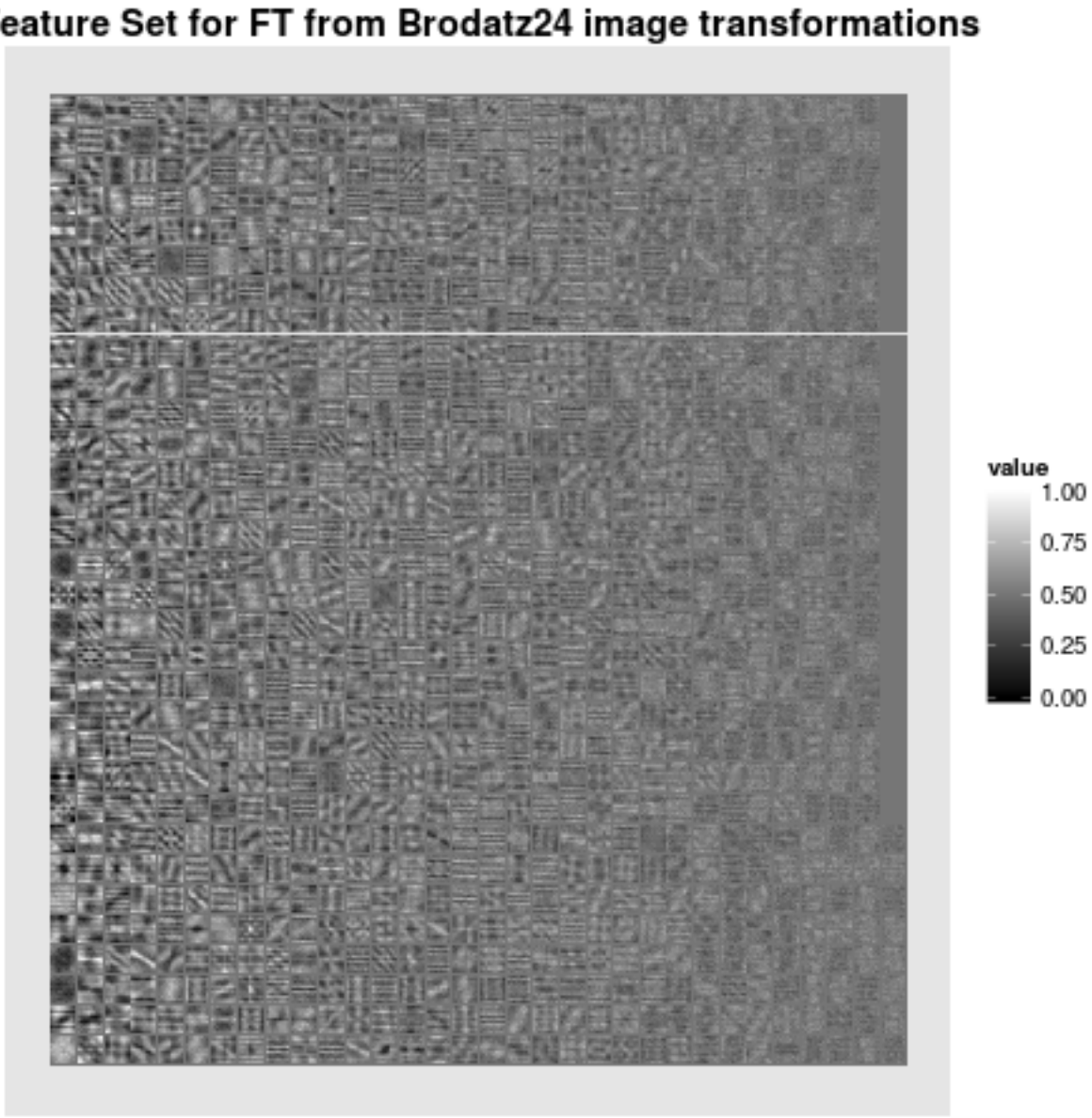


Figure 5.4: The visual filter learned for the convolutional feature $\mathbf{t}$ by the prosed model for Brodatz24 data set

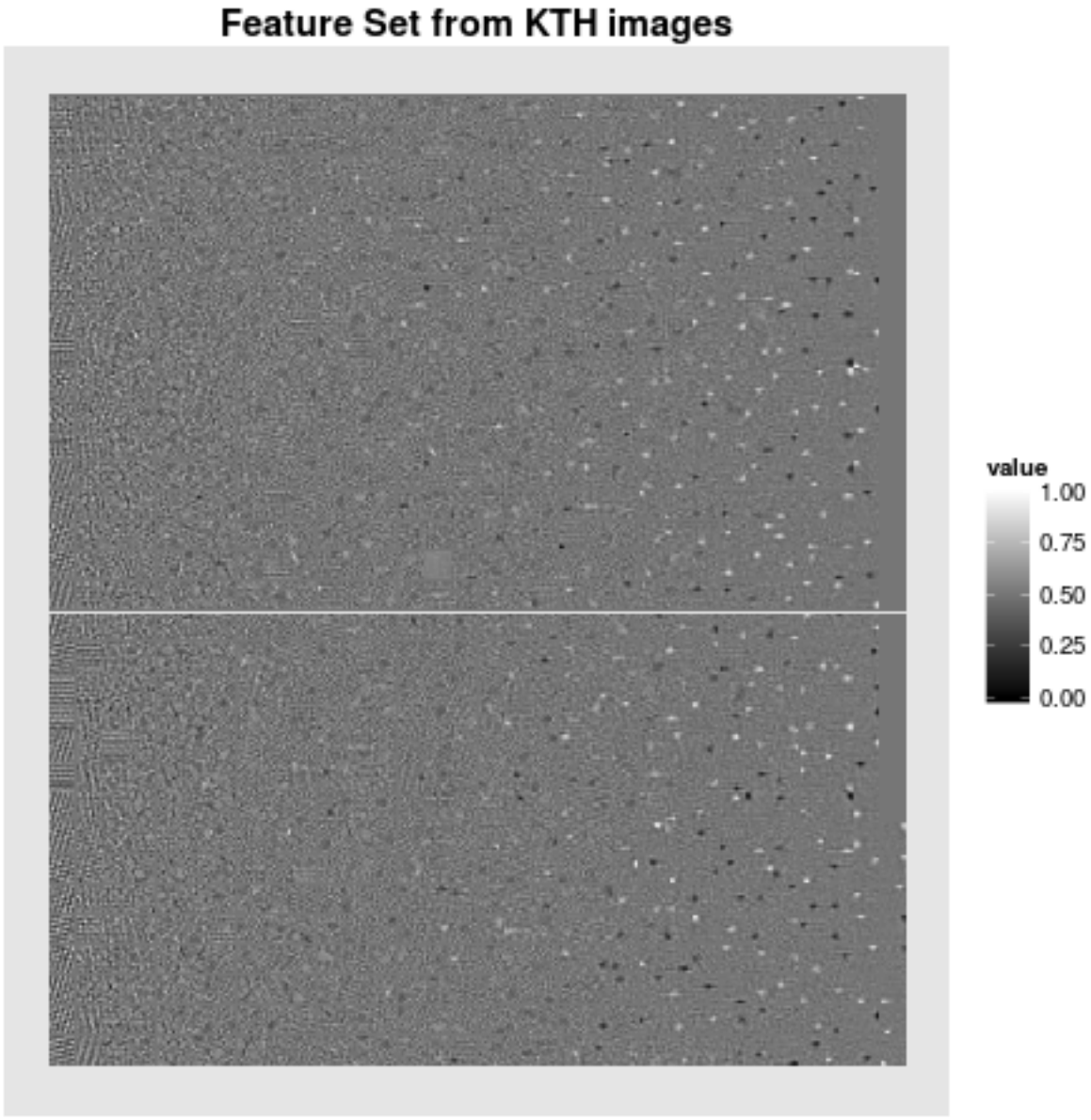


Figure 5.5: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine for KTH data set

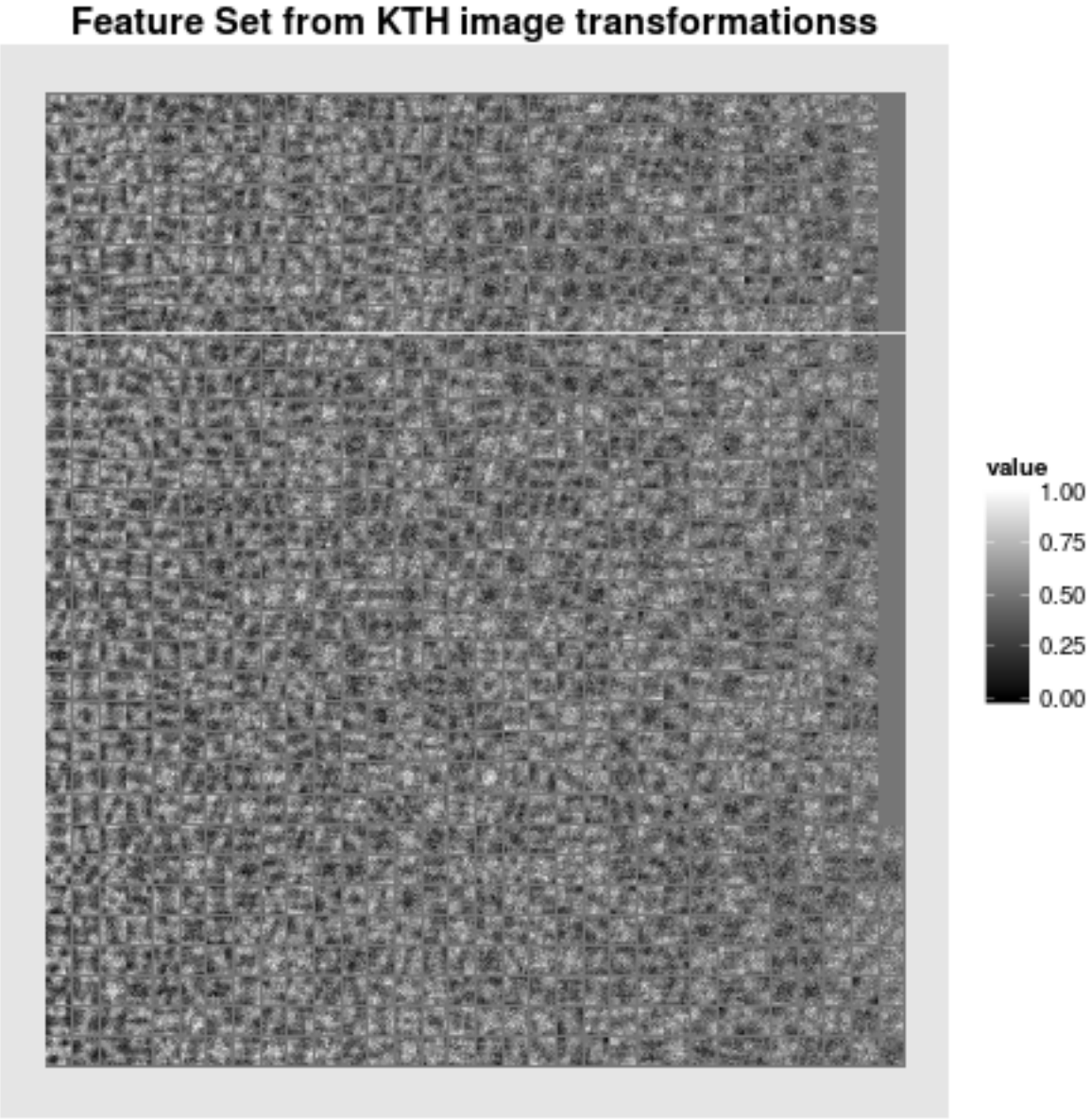


Figure 5.6: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine on the convolutional feature $\mathbf{t}$ for KTH data set

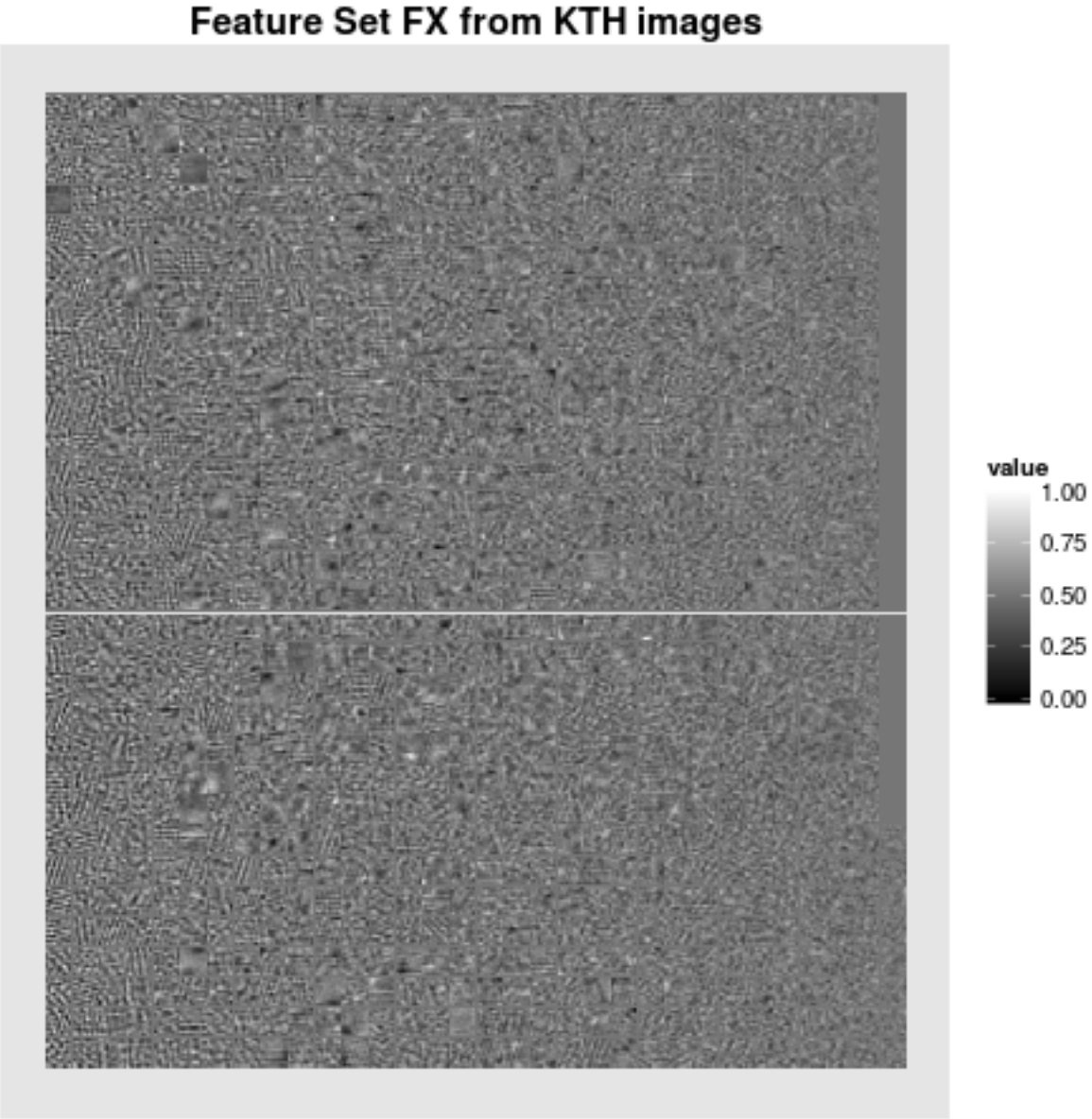


Figure 5.7: The visual filter learned for the normal image patches $\mathbf{x}$ by the proposed model for KTH data set

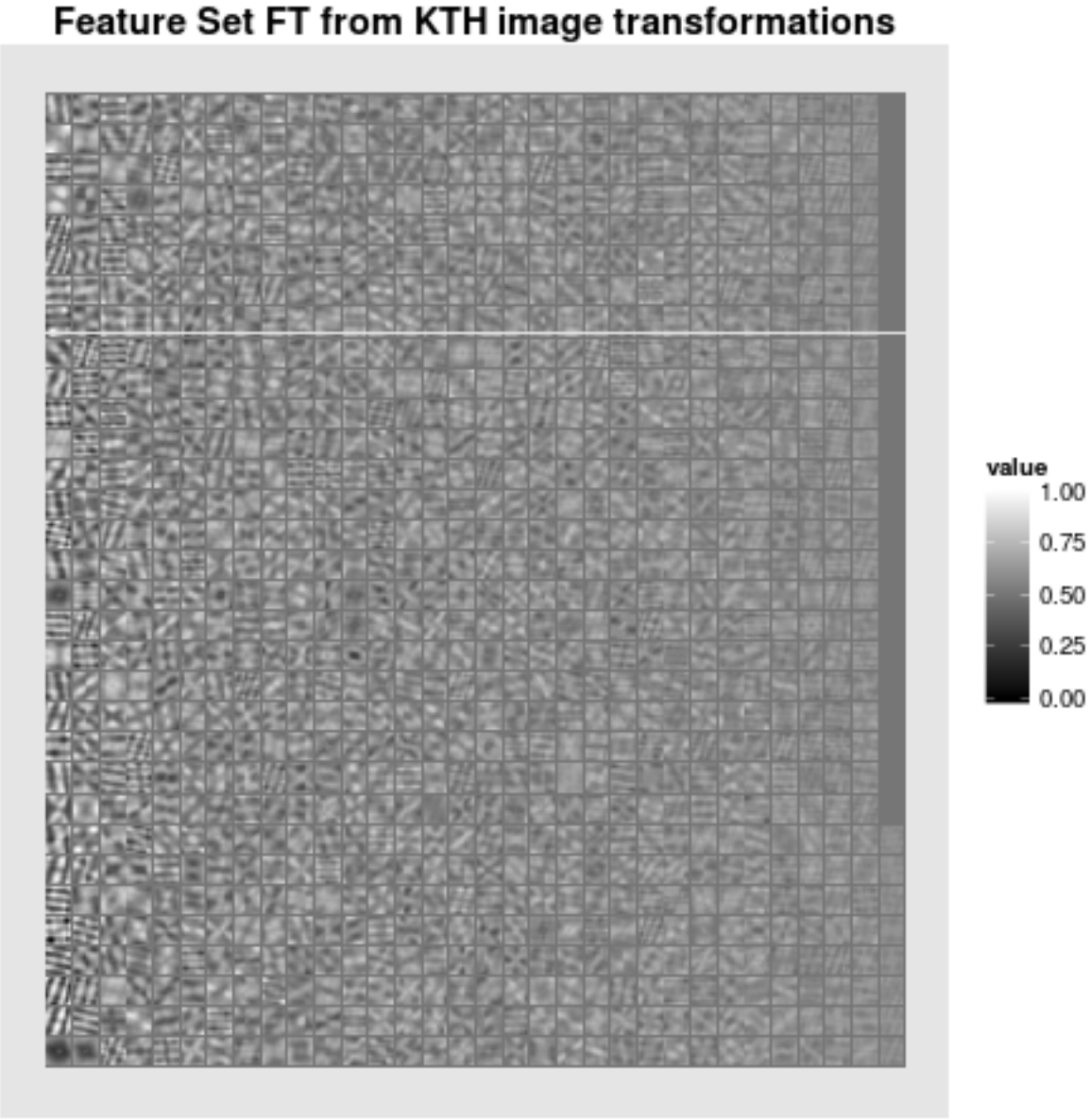


Figure 5.8: The visual filter learned for the convolutional feature $\mathbf{t}$ by the prosed model for KTH data set

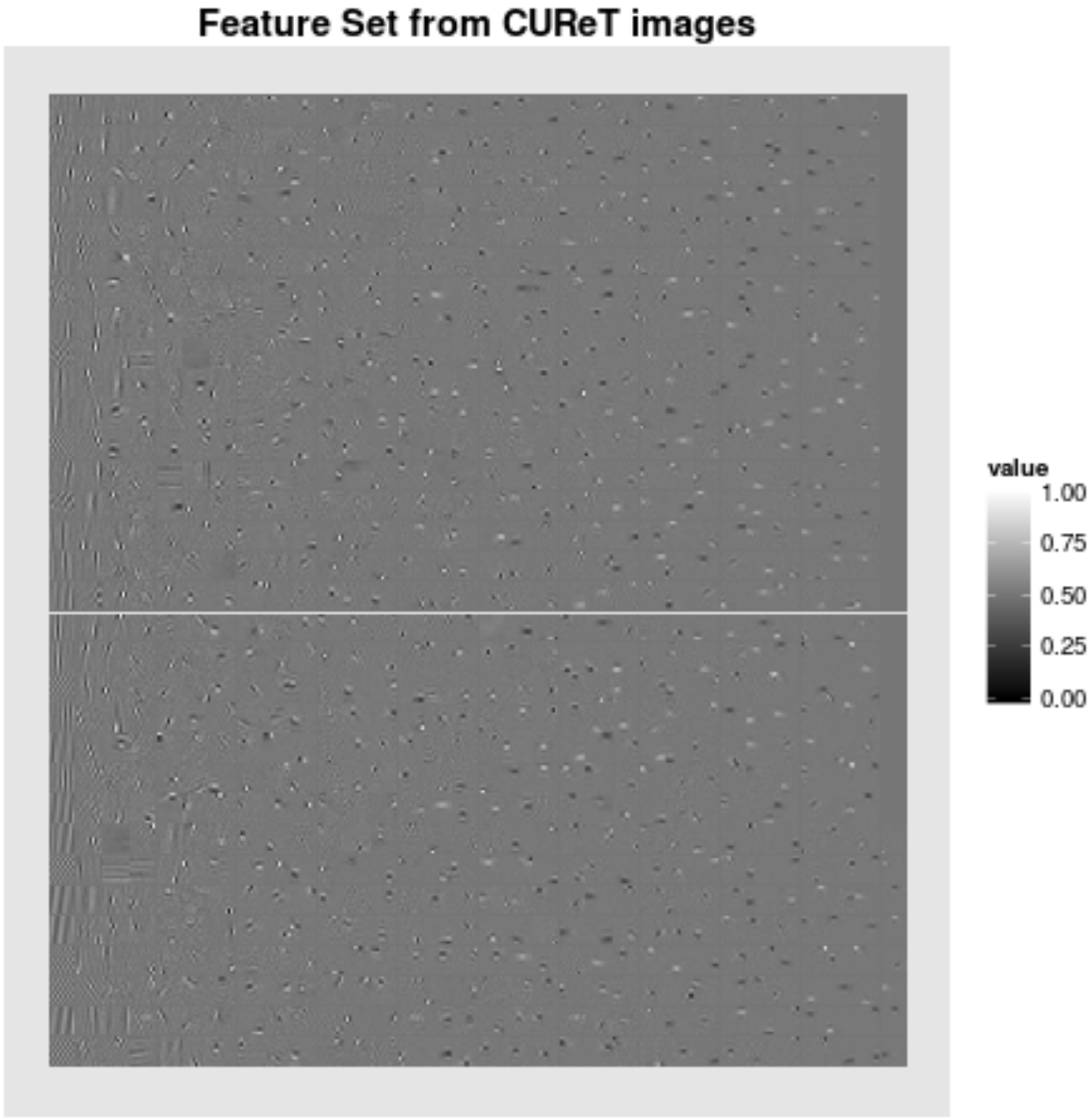


Figure 5.9: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine for CUReT data set

Figure 5.10: The visual filter learned by a set of normal enhanced Gaussian Boltzmann machine on the convolutional feature $\mathbf{t}$ for CUReT data set

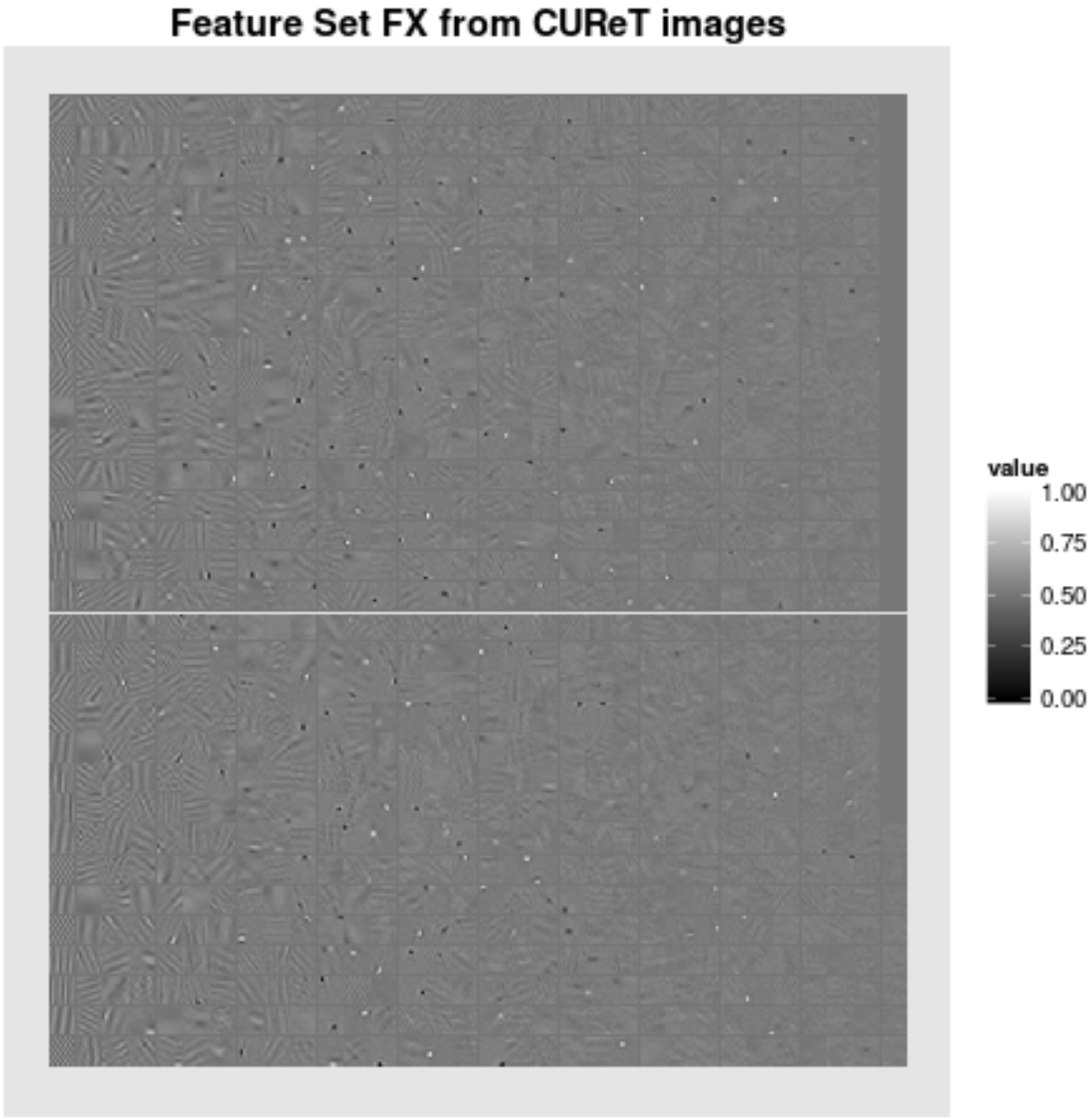


Figure 5.11: The visual filter learned for the normal image patches $\mathbf{x}$ by the proposed model for CUReT data set

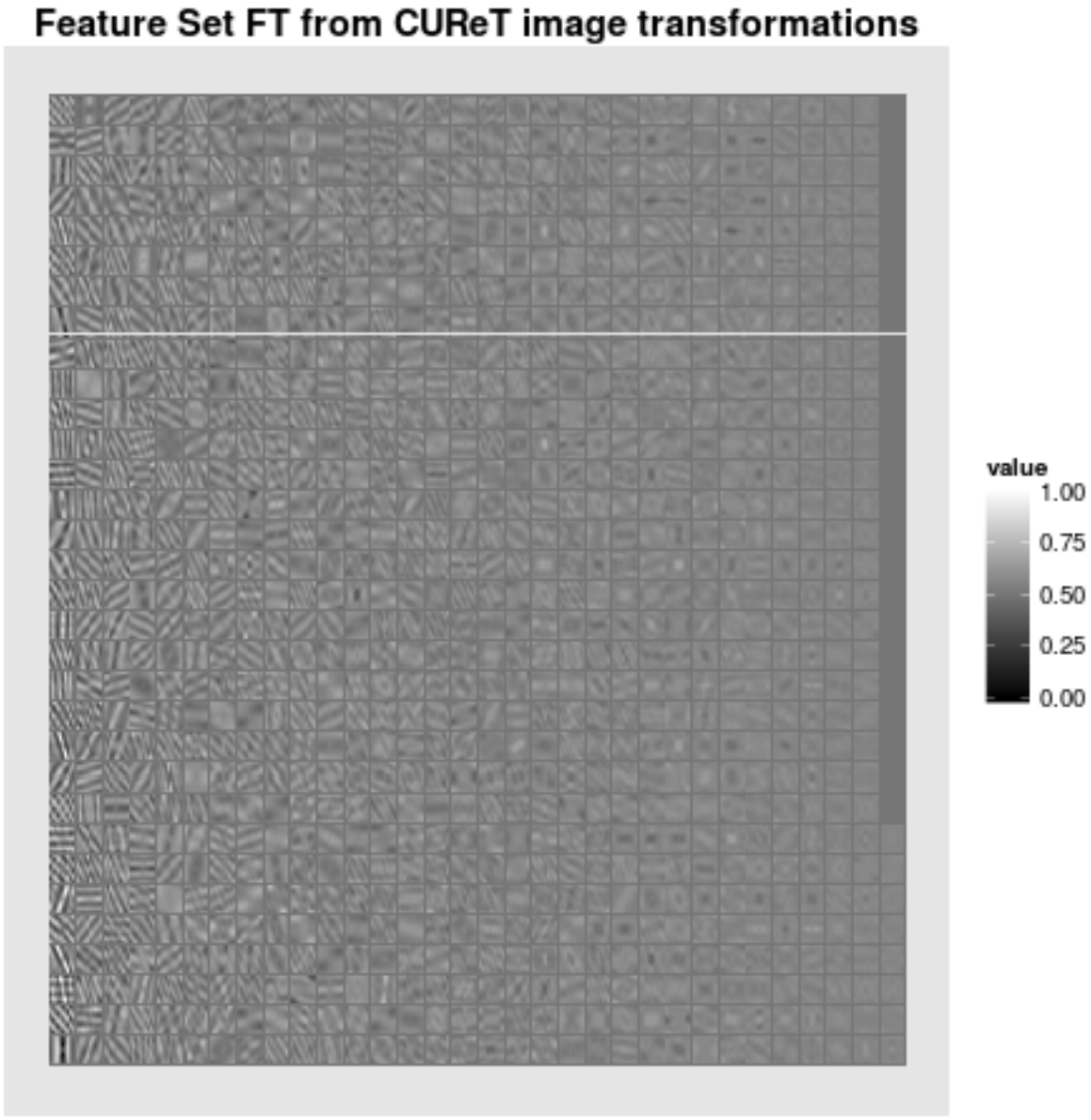


Figure 5.12: The visual filter learned for the convolutional feature $\mathbf{t}$ by the prosed model for CUReT data set

Data Source	Experiment Settings
Raw image patches $\mathbf{X}$	Perform a L1LR on $\mathbf{X}$
Transformation images $\mathbf{T}$ from $\mathbf{X}$	Perform a L1LR on $\mathbf{T}$
Combination of $\mathbf{X}$ and $\mathbf{T}$	Perform a L1LR on $[\mathbf{X}, \mathbf{T}]$
Feature $F(\mathbf{X})$ from $GRBM(\mathbf{X})$	Perform a L1LR on $F(\mathbf{X})$
Feature $F(\mathbf{T})$ from $GRBM(\mathbf{T})$	Perform a L1LR on $F(\mathbf{T})$
Feature $F(\mathbf{X}, \mathbf{T})$ from $CGRBM(\mathbf{X}, \mathbf{T})$	Perform a L1LR on $F(\mathbf{X}, \mathbf{T})$

Table 5.1: Texture Classification Settings

experiments are conducted to prove the usefulness of the model. To perform the classification, our model uses a logistic regression model using the feature extracted from the unseen texture features which generated from our learned model.

Three publicly available texture data sets are tested. The LIBLINEAR library [24] is used to build a classifier. In all classification experiments, a L1-regularized logistic regression classifier (L1LR) is trained. For the feature extraction experiments, one step contrastive divergence and some regularization parameters[1] are used. In all the experiments, 1000 hidden neurons are chosen, and $w_{dk} = 0$ for all $||d||_\infty > 5$. For comparison, we conducted six different classification experiments which are specified in Table 5.1.

The classification results in our experiments cannot be directly compared to other texture classification experiments as they typical extract a highly complex feature set from the whole image, while we directly extract features from small patches of textures. In other words, our model is capable of performing classification even though there is only little information about the texture, while it is typically hard to extract features if the images are too small in other conventional texture classification experiments.

Brodatz 24 Data Set

A subset of 24 different textures is manually selected from a large collection of 112 different textures. Only one large image is available for each class [42]. Each image in each class is divided into 25 $\{128 \times 128\}$ small images, 13 of them are used to generate the training patches, and rest of them are used to generate the testing patches. The patch size in the learning and testing is manually selected as $\{20 \times 20\}$. 240000 image patches are used in extracting the features. 24000 samples are used for training a classifier and 2400 samples are used for testing. The classification results are shown in Table 5.2a. Among all the experiments, the proposed method performs the best.

[1] weight decay = 0.0002, momentum = 0.2

Settings	Training	Testing
X	25.0%	16.2%
T	54.2%	50.4%
XT	61.8%	52.8%
FX	87.6%	63.0%
FT	91.7%	65.3%
FXT	**94.8%**	**67.0%**

(a) Brodatz 24 data set

Settings	Training	Testing
X	29.2%	19.0%
T	46.7%	43.8%
XT	57.3%	49.2%
FX	68.2%	60.4%
FT	72.0%	62.2%
FXT	**77.4%**	**66.2%**

(b) KTH data set

Settings	Training	Testing
RI	17.20%	10.16%
TI	24.67%	17.88%
RI + TI	30.88%	20.52%
FRI	44.82%	32.8%
FTI	48.62%	35.8%
FRI + FTI	**54.676%**	**38.76%**

(c) UIUC Dataset

Table 5.2: The texture classification result on various benchmark data sets.

KTH texture dataset

This dataset [11] has 11 different textures, 4 different samples for each texture, and 108 different images are available for each sample. Each image is of size $\{200 \times 200\}$, and the patch size is still selected as $\{20 \times 20\}$. Only the 108 images from sample a[2] in each texture are used: 54 for generating training samples and 54 for generating testing samples. 118800 patches are used for extracting the features. 11000 patches are used for training a classifier and 1100 sample are used for testing. The best result is obtained with the proposed method. Please note a poorer overall performance is expected as the variations within the training samples make the problem harder. The detailed results are shown in Table 5.2b.

UIUC Dataset

The UIUC texture dataset has 25 different texture and 40 different images in each class. Each image has the resolution of $\{640 \times 480\}$. Each image is downsampled to $\{160 \times 120\}$, the patch size for training is still $\{20 \times 20\}$. The data set is first used in [39]. In each class, 20 random images are used for generating training samples, and other ones are used

[2]Available at http://www.nada.kth.se/cvap/databases/kth-tips/

for generating test samples.A total number of 250000 samples are used as training. 25000 sample are used for training multi-class classifier and 2500 sample are used for testing.The same comparison results are shown in table 5.2c, and yet our proposed method works the best. Similarly, a poorer classification result can be expected as the complex variation in training samples will decrease the performance of the model.

5.4 Texture Reconstruction

As the proposed model is capable of capturing the internal structures of the trained objects, it's possible to reproduce, reconstruct and synthesize the trained data using an appropriate initialization.

However, following the previous chapter, our proposed model is successfully trained using a close enough approximation. This did not change its capabilities of producing useful reconstruction.

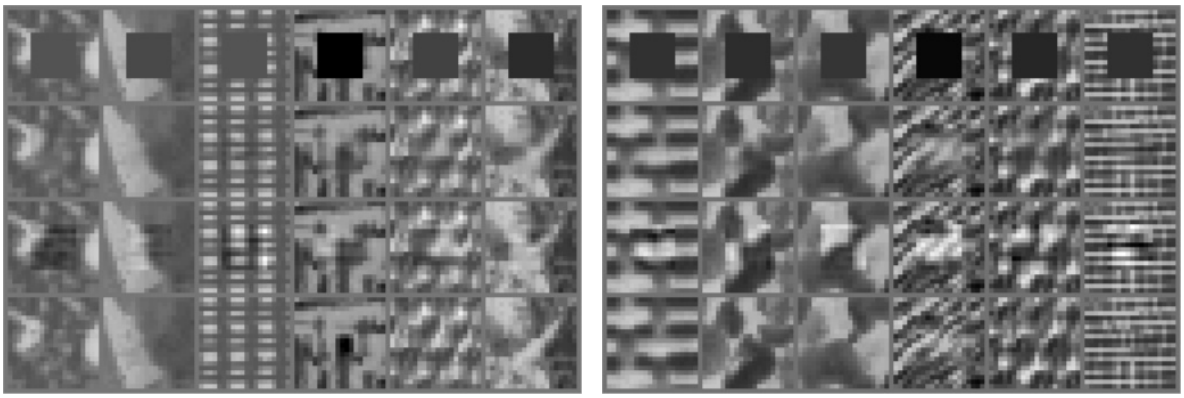
To this purpose, we made a demonstration of texture reconstruction for showing the connections between the proposed model and its approximation. In this experiment, 6 random image patches are chosen from the Brodatz24 dataset testing samples, and a $\{10\times10\}$ square center of the patches are removed for reconstruction. The reconstruction result can be seen in Figure 5.13. In the figure, we conducted two sets of different reconstruction experiments: one is on reconstructing seen training examples on the left; another is on reconstructing unseen testing examples on the right. Here, seen and unseen means if data is present for training the classifier. For both sides, the first row shows the random samples with missing centers. The second row shows the reconstruction from GRBM model, and the reconstruction from the proposed model GGBM is shown in the third row. The original samples are shown at the last row.

For comparison, the reconstruction result from GRBM(X) model is provided on the second row of the images. From this experiment, we can see that the learned model is capable of learning a generative model for the texture successfully. Despite the regularization, the reconstructions still seem to have too high contrast in our reconstruction on the third row. One way to improve the result would be to use the GRBM(X,T) as an initialization for the GGBM, and train it further.

The issues on how to realize the exact transformation from the approximated model to the real model remains an open research topic.

5.5 Conclusion

In this thesis, we have tackled the problem of modeling texture information. We proposed a modified version of GGBM and a simpler learning algorithm for that. From the experimental results, we can argue that the proposed model is beneficial in terms of modeling the structured information such as textures. Among all the results, the highest accuracies are obtained by the features learned from the proposed model. Although these accuracies



(a) Training Samples (b) Testing Samples

Figure 5.13: The texture reconstruction experiment

are not the state-of-the-art, the proposed model opened up a possibility where the texture information can be successfully modeled using the higher order Boltzmann machine.

Clearly, a few improvements are expected in our approach. As the true power of the deep network is the modeling the hierarchies of information, the texture information is expected to be modeled by a deep network where the proposed method works as a building block. Also, a more comprehensive test on the qualities of the generative model learned from the proposed model is expected.

Chapter 6

Conclusion

In this thesis, a series of studies combining deep networks and texture modeling are conducted.

Texture modeling is an old yet important subject in the field of machine vision. The texture information is generally considered as an internal property or characteristic of an object, and it is often one of the most distinguishable feature from other objects. However, due to the complex changes of environment illumination, affine transformation, scaling, rotation, etc., the detection and identification of texture can be a rather hard problem.

Since the earliest paper in [33], texture modeling has arrived in the era of computational modeling. Since then, various human-engineered feature extraction methods such as textons and co-occurrence matrices have been studied and adopted. Even though these methods work perfectly in most of the occasions, there is yet no understanding about how the texture is formed. To that end, generative models are applied to the texture information to understand the structural information about the textural information. In this thesis, an attempt has been made to understand the local textural information by applying the latest achievements in the advanced neural network research, namely deep learning.

Deep learning, based on deep networks, has been experimentally shown to have a huge capability of understanding the generative model of objects. Since the introduction of deep neural networks, they have been producing world-record level performances on various different machine learning tasks, such as hand-written digit classification, document hashing, speech recognition, etc. However, there is little research on applying deep learning on textural information until now. From this thesis, we can claim that textural information is greatly captured by the higher order Boltzmann machine, and the learned models are capable of generating good enough textures and excellent classification results on various publicly available texture databases.

The future research can be extended into several directions. First of all, an exact learning algorithm shall be proposed for the convolutional higher order Boltzmann machine. One of the reasons why we need to propose an approximate learning scheme is that the learning of the proposed model is too sensitive to the initial parameters, and can hardly produce consistent learning results on the different datasets. Additionally, based on the fact that our model is actually closely related to the approximated objective function, we can actually

utilize the well defined learning algorithm for the Gaussian restricted Boltzmann machine. However, in order to popularize the proposed method, a similar attempt on proposing an efficient learning algorithm is needed for making a consistent learning results on the various widely available datasets.

Secondly, a deep network based on the proposed method is expected to be introduced. In this work, an early attempt is made to learn the low level feature sets from the texture. It is expected that the deeper the deep network is, the more abstract texture features there will be present. Therefore, a systematic attempt to build a comprehensive deep network based on the proposed convolutional higher order Boltzmann machine will be made.

Thirdly, in order to produce consistent learning results on deep models, a new enhanced learning algorithm shall be studied to some extent. One of the problems that we faced is the difficulties of learning in the proposed model. One of the references can be the enhanced gradient learning algorithm proposed by Cho, et al in [15].

Bibliography

[1] D H Ackley, G.E Hinton, and T J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] Y. Bengio, A. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *arXiv preprint arXiv:1206.5538*, 2012.

[3] Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[4] Yoshua Bengio and Olivier Delalleau. On the Expressive Power of Deep Architectures. *Algorithmic Learning Theory*, pages 18–36, 2011.

[5] James R Bergen and Edward H Adelson. Early vision and texture perception. *Nature*, 333(6171):363–364, May 1988.

[6] RE Broadhurst. Statistical estimation of histogram variation for texture classification. *Proceedings of International Workshop on Texture Analysis and Synthesis*, pages 25–30, 2005.

[7] Phil Brodatz. *Textures: A Photographic Album for Artists and Designers 112 Plates.* Dover Publications, Inc., 1966.

[8] Phil Brodatz. *Textures: A Photographic Album for Artists and Designers.* Dover Publications, August 1999.

[9] E.J Candes and T Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[10] E.J Candes and T Tao. Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[11] Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-Specific Material Categorisation. *International Conference on Computer Vision*, pages 1597–1604, 2005.

[12] Miguel 'A Carreira-Perpiñ and Geoffrey Hinton. On Contrastive Divergence Learning. In Robert G Cowell and Zoubin Ghahramani, editors, *Proceeding of Artificial Intelligence and Statistics*, pages 33–40. Society for Artificial Intelligence and Statistics, 2005.

[13] George Casella and Edward I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[14] KyungHyun Cho, Alexander Ilin, and Tapani Raiko. Improved Learning of Gaussian-Bernoulli Restricted Boltzmann Machines. *International Conference on Artificial Neural Networks*, pages 10–17, 2011.

[15] KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Enhanced Gradient and Adaptive Learning Rate for Training Restricted Boltzmann Machines. *International Conference on Machine Learning*, pages 105–112, 2011.

[16] Aaron C Courville, James Bergstra, and Yoshua Bengio. Unsupervised Models of Images by Spikeand-Slab RBMs. *International Conference on Machine Learning*, pages 1145–1152, 2011.

[17] Oana G Cula and Kristin J Dana. 3D Texture Recognition Using Bidirectional Feature Histograms. *International Journal of Computer Vision*, 59(1):33–60, 2004.

[18] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[19] N Dalal and B Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[20] Kristin J Dana, Bram van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[21] S Davis and P Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[22] D.L Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[23] A.A Efros and T.K Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, 1999.

[24] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal Of Machine Learning Research*, pages 1871–1874, 2008.

[25] H Greenspan, S Belongie, R Goodman, and P Perona. Rotation Invariant Texture Recognition Using a Steerable Pyramid. In *12th International Conference on Pattern Recognition*, pages 162–167. IEEE Comput. Soc. Press, 1994.

[26] G M Haley and B S Manjunath. Rotation-Invariant Texture Classification Using Modified Gabor Filters. In *International Conference on Image Processing*, pages 262–265. IEEE Comput. Soc. Press, 1995.

[27] Robert M Haralick, K Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man and Cybernetics*, (6):610–621, 1973.

[28] Simon O Haykin. *Neural Networks and Learning Machines (3rd Edition)*. Prentice Hall, 3 edition, November 2008.

[29] GE Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

[30] Geoffrey Hinton and Ruslan Salakhutdinov. Discovering Binary Codes for Documents by Learning Deep Generative Models. *Topics in Cognitive Science*, 3(1):74–91, August 2010.

[31] Geoffrey E Hinton, Simon Osindero, and Yee Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18(7):1527–1554, July 2006.

[32] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[33] B Julesz. Visual Pattern Discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.

[34] B Julesz, E N Gilbert, L A Shepp, and H L Frisch. Inability of humans to discriminate between visual textures that agree in second-order statistics – revisited. *Perception*, 2(4):391–405, 1973.

[35] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, March 1981.

[36] Jyri Kivinen and Christopher Williams. Multiple Texture Boltzmann Machines. *Int. Conf. Artificial Intelligence and Statistics*, 22:638–646, 2012.

[37] S Konishi and A L Yuille. Statistical Cues for Domain Specific Image Segmentation with Performance Analysis. In *IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, pages 125–132. IEEE Comput. Soc, 2000.

[38] Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. Learning Algorithms for the Classification Restricted Boltzmann Machine. *The Journal of Machine Learning Research*, 13:643–669, March 2012.

[39] S Lazebnik, C Schmid, and J Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[40] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *International Conference on Machine Learning*, page 77, 2009.

[41] Thomas K Leung and Jitendra Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[42] Li Liu and P Fieguth. Texture Classification from Random Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):574–586, 2012.

[43] D.G Lowe. Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[44] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America A*, 7(5):923–932, 1990.

[45] Michael I Mandel and Daniel P W Ellis. A Web-Based Game for Collecting Music Metadata. *Journal of New Music Research*, 37(2):151–165, June 2008.

[46] R Memisevic and G Hinton. Unsupervised Learning of Image Transformations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[47] Roland Memisevic and Geoffrey E Hinton. Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines. *Neural computation*, 22(6):1473–1492, 2010.

[48] Roland Memisevic, Christopher Zach, Geoffrey E Hinton, and Marc Pollefeys. Gated Softmax Classification. *Advances in Neural Information Processing Systems*, pages 1603–1611, 2010.

[49] P Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 1976.

[50] Volodymyr Mnih, Hugo Larochelle, and Geoffrey E Hinton. Conditional Restricted Boltzmann Machines for Structured Output Prediction. *Uncertainty in Artificial Intelligence*, pages 514–522, 2011.

[51] Vinod Nair and Geoffrey E Hinton. Implicit Mixtures of Restricted Boltzmann Machines. *Advances in Neural Information Processing Systems*, pages 1145–1152, 2008.

[52] Vinod Nair and Geoffrey E Hinton. 3D Object Recognition with Deep Belief Nets. *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.

[53] Andrew Ng. Unsupervised feature learning and deep learning. Technical report, Standford University, 2011.

[54] T Ojala, M Pietikäinen, and T Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[55] Marc'Aurelio Ranzato, Alex Krizhevsky, and Geoffrey E Hinton. Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images. *Int. Conf. Artificial Intelligence and Statistics*, 9:621–628, 2010.

[56] Marc'Aurelio Ranzato, Joshua Susskind, Volodymyr Mnih, and Geoffrey Hinton. On Deep Generative Models with Applications to Recognition. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2857–2864. University of Toronto, IEEE, 2011.

[57] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323:pp 533–536, October 1986.

[58] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E Hinton. Restricted Boltzmann machines for collaborative filtering. *International Conference on Machine Learning*, pages 791–798, 2007.

[59] Cordelia Schmid. Weakly Supervised Learning of Visual Models and Its Application to Content-Based Retrieval. *International Journal of Computer Vision*, 56(1/2):7–16, 2004.

[60] J R Smith and Shih-Fu Chang. Transform Features for Texture Classification and Discrimination in Large Image Databases. In *1st International Conference on Image Processing*, pages 407–411. IEEE Comput. Soc. Press, 1994.

[61] Joshua Susskind, Geoffrey E Hinton, Roland Memisevic, and Marc Pollefeys. Modeling the joint density of two images under a variety of transformations. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2793–2800, 2011.

[62] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted Boltzmann Machines for modeling motion style. *International Conference on Machine Learning*, page 129, 2009.

[63] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling Human Motion Using Binary Latent Variables. *Advances in Neural Information Processing Systems*, pages 1345–1352, 2006.

[64] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. *International Conference on Machine Learning*, pages 1064–1071, 2008.

[65] Tijmen Tieleman and Geoffrey E Hinton. Using fast weights to improve persistent contrastive divergence. *International Conference on Machine Learning*, page 130, 2009.

[66] M Varma and A Zisserman. A Statistical Approach to Material Classification Using Image Patch Exemplars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):2032–2047, 2009.

[67] Manik Varma and Andrew Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62(1/2):61–81, 2005.

[68] Song Chun Zhu, Cheng-en Guo, Yizhou Wang, and Zijian Xu. What are Textons? *International Journal of Computer Vision*, pages 121–143, 2005.

Appendix A

Update rule for Convolutional Gated Boltzmann Machine

Gated Boltzmann Machine in Texture Modeling

X refers to the whole data, and one sample is denoted as **x**. The whole hidden activation is denoted as **H**, and the hidden activation for one sample is **h**.

The energy function for Gated Restricted Boltzmann machine is

$$
\begin{aligned}
&E(\mathbf{x}, \mathbf{y}, \mathbf{h}) \\
&= -\sum_{ijk} x_i y_j h_k w_{ijk} - \sum_k b_k^h h_k - \sum_{ij} x_i y_j b_j^{xy} - \sum_i x_i b_i^x - \sum_j y_j b_j^y - \sum_{ik} x_i h_k b_{ik}^{xh} - \sum_{jk} y_j h_k b_{jk}^{yh} \\
&= -\sum_{ijk} t_d h_k w_{ijk} - \sum_k b_k^h h_k - \sum_d t_d b_d^t - \sum_i x_i b_i^x - \sum_j y_j b_j^y - \sum_{ik} x_i h_k b_{ik}^{xh} - \sum_{jk} y_j h_k b_{jk}^{yh} \\
&= -\sum_{dk} t_d h_k w_{dk} - \sum_k b_k^h h_k - \sum_j t_j b_j^t - \sum_i x_i b_i^x - \sum_j y_j b_j^y - \sum_{ik} x_i h_k b_{ik}^{xh} - \sum_{jk} y_j h_k b_{jk}^{yh}
\end{aligned}
$$

where $d = i - j$, which represents the nearby neighborhood of pixel i. Also, $t_d = \sum_{id} x_i y_d$ and $\mathbf{W}_{ijk} = \mathbf{W}_{dk}$. Therefore,

$$
\begin{aligned}
&p(\mathbf{x}, \mathbf{h}) \\
&= \frac{1}{Z} \exp\left(-E(x, h)\right) \\
&= \frac{1}{Z} \exp\left(\sum_{dk} t_d h_k w_{dk} + \sum_k b_k^h h_k + \sum_d t_d b_d^t + \sum_i x_i b_i^x + \sum_j y_i b_j^y + \sum_{ik} x_i h_k b_{ik}^{xh} + \sum_{jk} y_j h_k b_{jk}^{yh}\right)
\end{aligned}
$$

Update rules

Learning this proposed model can be extremely complicated due to the enormous number of parameters. However, through a simplification, the model can be learned efficiently. Instead of feeding normal **X**, $[\mathbf{X}, \mathbf{T}]$ is fed to the normal RBM learning algorithms.

If we denote $[\mathbf{X}, \mathbf{T}]$ as $\mathbf{Q}$, one can find the joint distribution of samples and hidden neurons:

$$p(\mathbf{q}, \mathbf{h}) = \frac{1}{\mathbf{Z}} \exp\left(\sum_{ik} q_i h_k w_{ik} + \sum_i q_i b_i^p + \sum_k h_k b_k^h\right)$$

This equation can be transformed to

$$\begin{aligned}
&p(\mathbf{q}, \mathbf{h}) \\
&= \frac{1}{\mathbf{Z}} \exp\left(\sum_{ik} q_i h_k w_{ik} + \sum_i q_i b_i^p + \sum_k h_k b_k^h\right) \\
&= \frac{1}{\mathbf{Z}} \exp\left(\sum_{ik} [x_i, t_i] h_k w_{ik} + \sum_i [x_i, t_i] b_i^p + \sum_k h_k b_k^h\right) \\
&= \frac{1}{\mathbf{Z}} \exp\left(\sum_{ik} x_i h_k w_{ik}^x + \sum_{ik} t_i h_k w_{ik}^t + \sum_i x_i b_i^x + \sum_i t_i b_i^t + \sum_k h_k b_k^h\right) \\
&= \frac{1}{\mathbf{Z}} \exp\left(2\sum_{ik} x_i h_k \frac{w_{ik}^x}{2} + \sum_{ik} t_i h_k w_{ik}^t + 2\sum_i x_i \frac{b_i^x}{2} + \sum_i t_i b_i^t + \sum_k h_k b_k^h\right) \\
&= \frac{1}{\mathbf{Z}} \exp\left(\sum_{ik} x_i h_k \frac{w_{ik}^{xh}}{2} + \sum_{jk} y_j h_k \frac{w_{jk}^{jy}}{2} + \sum_{dk} t_d h_k w_{dk}^t + \sum_i x_i \frac{b_i^x}{2} + \sum_j y_j \frac{b_j^y}{2} + \sum_d t_d b_d^t + \sum_k h_k b_k^h\right)
\end{aligned}$$

From this, we can argue that one can learn a semi-Boltzmann machine by RBM algorithm. The sampling rule for each sample is:

$$\begin{aligned}
p(h_k = 1|\mathbf{x}, \mathbf{y}) &= \frac{1}{1 + \exp\left(-\sum_i x_i \frac{w_{ik}^{xh}}{2} - \sum_j y_j \frac{w_{jk}^{yh}}{2} - \sum_d t_d w_{dk} - b_k^h\right)} \\
p(x_i = 1|\mathbf{x}, \mathbf{h}) &= \frac{1}{1 + \exp\left(-\sum_{ik} h_k \frac{W_{ik}^{xh}}{2} - \sum_{jk} y_j h_k w_{ijk} - b_i^x - \sum_{ij} y_j b_i^t\right)} \\
p(y_j = 1|\mathbf{y}, \mathbf{h}) &= \frac{1}{1 + \exp\left(-\sum_{jk} h_k \frac{W_{jk}^{yh}}{2} - \sum_{ik} x_j h_k w_{ijk} - b_j^y - \sum_{ij} x_j b_i^t\right)}
\end{aligned}$$

Gaussian GBM and its update rule

Similarly, one can train a Gaussian GBM using GRBM. The energy function for GB-semi-GBM is defined as

$$
\begin{aligned}
&E(\mathbf{x},\mathbf{y},\mathbf{h}) \\
&= -\sum_{ijk}\frac{x_i}{\sigma_i}\frac{y_j}{\sigma_j}h_k w_{ijk} - \sum_{ij}\frac{x_i}{\sigma_i}\frac{y_j}{\sigma_j}w_{ij} - \sum_k h_k b_k^h + \sum_i \frac{(x_i - b_i^x)^2}{2\sigma_i^2} \\
&+ \sum_j \frac{(y_j - b_j^y)^2}{2\sigma_j^2} - \sum_{ik}\frac{x_i}{\sigma_i}h_k b_{ik}^{xh} - \sum_{jk}\frac{y_j}{\sigma_j}h_k b_{jk}^{yh} \\
&= -\sum_{dk}\frac{t_d}{\sigma_d^2}h_k w_{dk} - \sum_d \frac{t_d}{\sigma_d^2}b_d^t - \sum_k h_k b_k^h + \sum_i \frac{(x_i - b_i^x)^2}{2\sigma_i^2} \\
&+ \sum_j \frac{(y_j - b_j^y)^2}{2\sigma_j^2} - \sum_{ik}\frac{x_i}{\sigma_i^2}h_k b_{ik}^{xh} - \sum_{jk}\frac{y_j}{\sigma_j^2}h_k b_{jk}^{yh} \\
&= -\frac{1}{\sigma^2}\sum_{dk} t_d h_k w_{dk} - \frac{1}{\sigma^2}\sum_d t_d b_d^t + \frac{1}{\sigma^2}\sum_i (x_i - b_i^x)^2 \\
&+ \frac{1}{\sigma^2}\sum_j (y_j - b_j^y)^2 - \frac{1}{\sigma^2}\sum_{ik} x_i h_k b_{ik}^{xh} - \frac{1}{\sigma^2}\sum_{jk} y_j h_k b_{jk}^{yh} - \sum_k h_k b_k^h
\end{aligned}
$$

Therefore, the joint probability of data and hidden neurons are:

$$
p(\mathbf{x},\mathbf{y},\mathbf{h}) = \frac{1}{\mathbf{Z}}\exp\left(-E(\mathbf{x},\mathbf{y},\mathbf{h})\right)
$$

Then, the update rules are defined as

$$
\begin{aligned}
p(\mathbf{x}|\mathbf{y},\mathbf{h}) &= \prod_i \frac{1}{\sigma_i\sqrt{2\pi}}\exp\left(-\frac{1}{2\sigma_i^2}(x_i - b_i^x - \sum_{jk} y_j h_k w_{ijk} - \sum_j y_j w_{ij} - \sum_k h_k w_{ik})^2\right) \\
p(\mathbf{y}|\mathbf{x},\mathbf{h}) &= \prod_j \frac{1}{\sigma_j\sqrt{2\pi}}\exp\left(-\frac{1}{2\sigma_j^2}(y_j - b_j^y - \sum_{ik} x_i h_k w_{ijk} - \sum_i x_i w_{ij} - \sum_k h_k w_{jk})^2\right) \\
p(h=1|\mathbf{x},\mathbf{y}) &= \frac{1}{1+\exp\left(-\frac{1}{\sigma^2}\sum_i x_i w_{ik}^x - \frac{1}{\sigma^2}\sum_j y_j w_{jk}^y - \frac{1}{\sigma^2}\sum_{ijk} x_i y_j w_{ij} - b_k^h\right)}
\end{aligned}
$$