Robot semantic place recognition based on deep belief networks and a direct use of tiny images

Ahmad Hasasneh

▶ **To cite this version:**

Ahmad Hasasneh. Robot semantic place recognition based on deep belief networks and a direct use of tiny images. Other [cs.OH]. Université Paris Sud - Paris XI, 2012. English. NNT : 2012PA112298 . tel-00960289

HAL Id: tel-00960289

https://theses.hal.science/tel-00960289

Submitted on 17 Mar 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE : EDIPS

Laboratoire dInformatique pour la Mécanique et les Sciences de lIngénieur

DISCIPLINE : Informatique

THÈSE DE DOCTORAT

soutenue le ... novembre 2012

par

Ahmad Mohammed Hasasneh

Robot semantic place recognition based on deep belief networks and a direct use of tiny images

Directeur de thèse :	Philippe Tarroux	Professeur, ÉNS
	Emmanuelle Frenoux	Maître de Conférences, Université Paris Sud
Composition du jury :		
Rapporteurs :	Patrick Gallinari	Professeur, Université Pierre et Marie Curie
	David Filliat	Professeur, ENSTA
Examinateurs :	Mathias Quoy	Professeur, Université de Cergy-Pontoise
	Gilles Gasso	Maître de Conférences, INSA Rouen
	Hélène Paugam-Moisy	Professeur, Université de Lyon 2
	Anne Vilnat	Professeur, Université Paris Sud

I would like to dedicate this thesis to my loving, parents, wife, sons,
brothers, and sisters ...

Acknowledgements

First and foremost, God, surely, helped me: otherwise it would not be possible to complete this research. Also, I would like to express my deep gratitude to my supervisors, Prof. Philippe Tarroux and Dr. Emmanuelle Frenoux, for their kind support, guidance, and inspirational ideas throughout this research. I mostly thank them for introducing me to the science of "machine learning, image processing, and robot place recognition". They have been an amazing mentors and supervisors, not only in research, but also other aspects of cooperation. I cannot thank them enough. Also, the financial support of the University of Paris Sud and the French Embassy in Palestine is gratefully acknowledged.

I fully dedicate this research to my parents, Mohammed and Zuhra, for their support and continuous encouragement. I owe my loving thanks to my wife, Nadia, and my sons, Tareq, Mohammed, and Layan for making me happy, even in the difficult times. They have lost a lot in order to complete this research. Thus, without their support and understanding the situation it would have been more complicated to complete this work. Also, my special thanks is due to my brother, Dr. Nabil Hasasneh for his great and continuous support.

I have furthermore special thanks to my relatives: Mohamed and Maryiam who are the parents of Nadia for their kind support. I also would like to thank all my colleagues and friends who shared my life activity, including: Dr. Houda Bouamor, Asma Gharsellaoui, Dr. Mehdi Ammi, Dr. Yacine Bellik, Dr. Chahnez Zakaria, Laurent Pointal, Souhir Gahbiche Braham, Nada Labidi, Mehdi Boukhris, Ignace Kuevakioe, Dr. Mathieu Dubois, Gaëtan Pruvost, Sylvain Chevallier, Adrien Girard, Mohamed Yacine Tsalamlal, Fadi Abdin, Dr. Julien Nauroy, Abdelhak Bourdim,

Mohamed Sakhraoui, Ahmed Mohamed, and Islam Kiswani. I also thank all administrators, technicians, and secretaries in the laboratory of LIMSI-CNRS which is associated to Paris Sud and Sorbonne Universities for their supports and help.

Abstract

Usually, human beings are able to quickly distinguish between different places, solely from their visual appearance. This is due to the fact that they can organize their space as composed of discrete units. These units, called "semantic places", are characterized by their spatial extend and their functional unity. Such a semantic category can thus be used as contextual information which fosters object detection and recognition. Recent works in semantic place recognition seek to endow the robot with similar capabilities. Contrary to classical localization and mapping works, this problem is usually addressed as a supervised learning problem.

The question of semantic places recognition in robotics - the ability to recognize the semantic category of a place to which scene belongs to - is therefore a major requirement for the future of autonomous robotics. It is indeed required for an autonomous service robot to be able to recognize the environment in which it lives and to easily learn the organization of this environment in order to operate and interact successfully. To achieve that goal, different methods have been already proposed, some based on the identification of objects as a prerequisite to the recognition of the scenes, and some based on a direct description of the scene characteristics. If we make the hypothesis that objects are more easily recognized when the scene in which they appear is identified, the second approach seems more suitable. It is however strongly dependent on the nature of the image descriptors used, usually empirically derived from general considerations on image coding.

Compared to these many proposals, another approach of image coding, based on a more theoretical point of view, has emerged the last few years.

Energy-based models of feature extraction based on the principle of minimizing the energy of some function according to the quality of the reconstruction of the image has lead to the Restricted Boltzmann Machines (RBMs) able to code an image as the superposition of a limited number of features taken from a larger alphabet. It has also been shown that this process can be repeated in a deep architecture, leading to a sparse and efficient representation of the initial data in the feature space. A complex problem of classification in the input space is thus transformed into an easier one in the feature space. This approach has been successfully applied to the identification of tiny images from the 80 millions image database of the MIT.

In the present work, we demonstrate that semantic place recognition can be achieved on the basis of tiny images instead of conventional Bag-of-Word (BoW) methods and on the use of Deep Belief Networks (DBNs) for image coding. We show that after appropriate coding a softmax regression in the projection space is sufficient to achieve promising classification results. To our knowledge, this approach has not yet been investigated for scene recognition in autonomous robotics.

We compare our methods with the state-of-the-art algorithms using a standard database of robot localization. We study the influence of system parameters and compare different conditions on the same dataset. These experiments show that our proposed model, while being very simple, leads to state-of-the-art results on a semantic place recognition task.

Keywords: *Semantic Places Recognition, Energy-based models, Restricted Boltzmann Machines, Deep Belief Networks, Bag-of-Words, Softmax Regression.*

Résumé

Il est généralement facile pour les humains de distinguer rapidement différents lieux en se basant uniquement sur leur aspect visuel. Cela est, en effet, du fait qu'ils peuvent organiser leur espace de telle sorte qu'il soit composé d'unités discrètes. Ces unités, appelées lieux sémantiques, se caractérisent par leurs limites spatiales et leur unité fonctionnelle. Cette catégorie sémantique peut donc être utilisée comme information contextuelle favorisant la détection et la reconnaissance d'objets. Des travaux récents en reconnaissance des lieux sémantiques visent à doter les robots de capacités similaires. Contrairement aux travaux classiques, portant sur la localisation et la cartographie, cette tâche est généralement considérée comme un problème d'apprentissage supervisé.

En robotique, la reconnaissance de lieux sémantique - la capacité à reconnaître la catégorie sémantique à laquelle un endroit où une scène appartient - peut être considérée comme une condition essentielle pour l'avenir de la robotique autonome. Il est en effet nécessaire pour un robot autonome de reconnaître l'environnement dans lequel il vit et d'apprendre facilement l'organisation de cet environnement pour pouvoir fonctionner et interagir avec succès. Pour atteindre cet objectif, différentes méthodes ont déjà été proposées. Certaines sont basées sur l'identification des objets comme une condition préalable à la reconnaissance des scènes, et d'autres fondées sur une description directe des caractéristiques de la scène. Si nous faisons l'hypothèse que les objets sont plus faciles à reconnaître quand la scène dans laquelle ils apparaissent est bien identifiée, la deuxième approche semble plus appropriée. Elle est cependant fortement dépendante de la nature des descripteurs d'images utilisées qui sont généralement

dérivés empiriquement a partir des observations générales sur le codage d'images.

En opposition avec ces propositions, une autre approche de codage des images, basée sur un point de vue plus théorique, a émergé ces dernières années. Les modèles d'extraction de caractéristiques fondés sur le principe de la minimisation d'une fonction d'énergie en relation avec un modèle statistique génératif expliquant au mieux les données, ont abouti à l'apparition des Machines de Boltzmann Restreintes (RBMs) capables de coder une image comme la superposition d'un nombre limité de caractéristiques extraites à partir d'un plus grand alphabet. Il a été montré que ce processus peut être répété dans une architecture plus profonde, conduisant à une représentation parcimonieuse et efficace des données initiales dans l'espace des caractéristiques. Le problème complexe de la classification dans l'espace de début est ainsi remplacé plus un problème plus simple dans l'espace des caractéristiques. Cette approche a été appliquée avec succès à l'identification de mini-images à partir d'une base de données du MIT contenant 80 millions d'images.

Dans ce travail, nous démontrons que la reconnaissance sémantique des lieux peut être réalisée en considérant des mini-images au lieu des méthodes classiques exploitant les méthodes de type "sacs-de-mots" (bag-of-words, BoW) et par l'utilisation des Deep Belief Networks (DBNs) pour le codage des images. Nous montrons que, après avoir realisé un codage approprié, une régression softmax dans l'espace de projection est suffisante pour obtenir des résultats de classification prometteurs. A notre connaissance, cette approche n'a pas encore été étudiée pour la reconnaissance de scène en robotique autonome.

Nous avons comparé nos méthodes avec les algorithmes de l'état-de-l'art en utilisant une base de données standard de localisation de robot. Nous avons étudié l'influence des paramètres du système et comparé les différentes conditions sur la même base de données. Les expériences réalisées montrent que le modèle que nous proposons, tout en étant très simple,

conduit à des résultats comparables à l'état-de-l'art sur une tâche de reconnaissance de lieux sémantique.

Mots-clés: *reconnaissance de lieux sémantiques, modéles basés sur l'énergie, machine de Boltzmann restreinte, architecture profonde, sac-de-mots, régression Softmax.*

Introduction

Un robot autonome doit être en mesure de reconnaître l'environnement dans lequel il évolue. Cette caractéristique lui permet d'apprendre l'organisation de son environnement pour un fonctionnement et une interaction optimaux. Pour atteindre cet objectif, différentes solutions ont été proposées. Certaines approches sont basées sur la localisation métrique (c.à.d. la capacité d'un robot mobile à déterminer sa position dans un repère commun), d'autres exploitent la localisation topologique (c.à.d. la capacité de produire une carte de son environnement). Toutefois, dans ces approches, l'information concernant l'emplacement est différente de l'information utilisée pour déterminer la catégorie sémantique du lieu. Ainsi, au-delà d'une localisation métrique précise utilisée dans les méthodes de localisation et de cartographie simultanées (Simultaneous Localization and Mapping: SLAM), la capacité pour un robot mobile de déterminer la nature de son environnement (cuisine, pièce, couloir, *etc.*) reste une tâche difficile.

La connaissance des coordonnées métriques ou même l'information de voisinage qui peut être encodée dans des cartes topologiques n'est, en effet, pas suffisante. L'approche par reconnaissance de lieux sémantiques (Semantic Place Recognition: SPR) est cependant nécessaire pour un grand nombre de tâches. Elle peut par exemple être utilisée comme une information contextuelle qui favorise la détection et la reconnaissance d'objets (donnant a priori l'identité, l'emplacement et l'échelle de l'objet). Ceci peut être utile lorsque la sémantique est obtenue sans aucune référence à des objets présents dans la scène. De plus, la catégorisation sémantique offre une référence absolue pour l'emplacement du robot, fournissant une solution simple pour des problèmes où la localisation ne peut pas être déduite à partir des emplacements voisins. C'est le cas, par exemple, pour

résoudre des problémes tels que celui du robot kidnappé ou de la fermeture de boucle.

Etat de l'art

Les recherches récentes ont proposé d'exploiter les descripteurs visuels pour la reconnaissance sémantique. Les approches les plus fréquentes utilisent les descripteurs basés sur des caractéristiques utilisant des détecteurs globaux, tels que les descripteurs GiST et CENTRIST [Pronobis et al., 2006; Torralba et al., 2003a; Wu et al., 2009], ou les signatures locales calculées autour des points d'intérêt en utilisant des détecteurs locaux, comme par exemple les signaux SIFT et SURF [Filliat, 2008; Ullah et al., 2008]. Cependant, ces représentations ont recours à des méthodes de type sac-de-mots (Bag-of-Words : BoWs), afin de réduire la taille des représentations. Une quantification vectorielle est ensuite appliquée de telle sorte que afin de représenter l'image par un histogramme. Les approches discriminantes peuvent être utilisées pour calculer la probabilité d'être dans un lieu donné en fonction de l'observation courante. Les approches génératives peuvent également être utilisées pour calculer la probabilité d'une observation donnée dans un certain lieu en utilisant le filtrage bayésien. Parmi ces approches, certains travaux [Torralba et al., 2008] omettent l'utilisation de l'étape de quantification et modélisent la densité de probabilité à l'aide d'un mélange de gaussiennes (Gaussian Mixture Model : GMM). Les approches récentes proposent également d'utiliser des classificateurs bayésiens naïfs et l'intégration temporelle qui permettent de combiner les observations successives [Dubois et al., 2011].

La SPR nécessite donc l'utilisation d'un espace de caractéristiques approprié qui permet une classification précise et rapide. Contrairement à ces méthodes empiriques, de nouvelles méthodes d'apprentissage automatique ont récemment émergé. La structure auto-similaire des images naturelles a permis la création de codes optimaux. Ces codes sont basés sur des caractéristiques statistiquement indépendantes. A cet effet, différentes méthodes ont été proposées pour construire ces codes à partir de bases de données des images. Imposer des contraintes de localité et de faible

densité à ces caractéristiques est très important. Ceci est probablement dû au fait que les algorithmes simples basés sur ces contraintes peuvent obtenir des signatures linéaires analogues à la notion de champ récepteur dans les systèmes naturels. Ces dernières années, différent travaux se sont intéressé aux algorithmes de vision par ordinateur reposant sur des représentations locales clairsemé, en particulier pour les problèmes de classification d'images et de reconnaissance d'objets [Boureau et al., 2010; Ranzato et al., 2007b; Wright et al., 2010; Yang et al., 2009]. En outre, d'un point de vue génératif, l'efficacité de codage local clairsemé dense, par exemple pour la reconstruction d'image [Labusch and Martinetz], est justifiée par le fait qu'une image naturelle peut être reconstruite par un plus petit nombre de caractéristiques. Il a été démontré que l'analyse par composantes indépendantes (Independent Component Analysis: ICA) génère des caractéristiques localisées. De plus, cette analyse est efficace pour les distributions présentant un niveau de kurtosis élevé qui représentent des statistiques d'images naturelles dominées par des composants rares comme les contours. Cependant, cette méthode est linéaire et non récursive.

Ces deux limitations n'existent pas dans le cas des approches DBN [Hinton et al., 2006] qui introduisent des non-linéarités dans le système de codage et qui présentent de multiples couches. Chaque couche est constituée d'une RBM, une version simplifiée d'une machine de Boltzmann proposé par Smolensky [Smolensky, 1986] et Hinton [Hinton, 2002]. Chaque RBM est capable de construire un modèle génératif statistique pour ses entrées à l'aide d'un algorithme d'apprentissage relativement rapide (Contrastive Divergence: CD), qui a été introduit la première fois par Hinton [Hinton, 2002]. Une autre caractéristique importante des codes utilisés dans les systèmes naturels, la densité de représentation [Olshausen and Field, 2004], est également réalisée avec l'approche DBN. En outre, il a été montré que ces approches sont robustes pour extraire des caractéristiques locales clairsemé dans de mini-images [Torralba et al., 2008].

Cependant, dans ces recherches, nous supposons que les représentations clairsemé conduisent à des problèmes linéairement séparables. Ce type

de représentations devrait simplifier le problème de classification. Par ailleurs, nous avons étudié l'extraction de caractéristiques à partir de données blanchies et normalisées. Nous avons également étudié l'effet de cette normalisation sur le problème SPR.

Description du modèle

Notre nouvelle approche SPR comporte trois principales étapes: le prétraitement des images, l'élaboration non-supervisée des caractéristiques de l'em-placement, et l'apprentissage supervisé de l'emplacement. Plus précisément, la première étape consiste à convertir la couleur en niveaux de gris, en les réduisant à de petits patches d'images, puis en normaliser le résultat. La deuxième étape consiste à coder les images d'entrée en utilisant les caractéristiques extraites. Elle consiste à extraire à travers plusieurs couches RBM formant un DBN un alphabet de caractéristiques. La méthode DBN est capable de coder de façon optimale les images d'une manière adaptée à leur classification. La phase finale est la classification qui consiste à discriminer entre les différents localisations possibles pour le robot.

Traitement des images

Utilisation des mini-images

La dimension d'entrée typique pour un DBN est d'environ 1000 unités (par exemple 30×30 pixels). L'utilisation de plus petits patches pourraient rendre le modèle incapable d'extraire des caractéristiques intéressantes. L'utilisation de plus grands patches peut conduire à des temps d'exécution importants durant l'apprentissage des caractéristiques. En outre, la multiplication des poids de connexion agit négativement sur la convergence de l'algorithme CD. La question est donc de savoir comment redimensionner la taille des images réalistes (par exemple 300×300 pixels) pour les rendre appropriées pour l'DBN.

Trois solutions peuvent être envisagées. La première consiste à sélectionner les patches aléatoirement à partir de chaque image comme réalisé dans les travaux de [Ranzato et al., 2010]. La seconde approche consiste à utiliser une architecture convolutive, telle que proposée dans [Lee et al.,

2009]. Enfin, la dernière approche consiste à redimensionner la taille de chaque image pour obtenir une image de plus petite taille comme proposé dans [Torralba et al., 2008]. La première solution revient à extraire les caractéristiques locales. La caractérisation d'une image à l'aide de ces caractéristiques peut être réalisée à l'aide de l'approche BoW que nous souhaitons éviter. La deuxième solution présente les mêmes limites et augmente le nombre de calculs qui doivent être traités par le processeur graphique. L'extraction de caractéristiques utilisant les patches aléatoires est indépendante des structures spatiales de chaque image [Norouzi et al., 2009]. Dans le cas de scènes structurées comme celles utilisées avec les SPR, ces structures portent une information intéressante.

En outre, des mini-images ont été utilisées avec succès dans [Torralba et al., 2008] pour classer et extraire des images à partir de la base de données de 80 millions d'images développée au MIT. Torralba et al. ont montré que l'utilisation des mini-images combinées avec une approche DBN conduit à coder chaque image par un petit vecteur binaire. Ce vecteur définit les éléments d'un alphabet caractéristique qui peut être utilisé pour définir de façon optimale l'image originale. Le vecteur binaire agit comme un code-barres tandis que l'alphabet de caractéristiques est calculé une seule fois à partir d'un ensemble représentatif de l'image. L'intérêt de cette approche est démontré par le fait que le petit vecteur binaire (comme ceux que nous utilisons comme sortie de notre structure de DBN) dépasse largement le nombre d'images qui doivent être codées même dans le cas d'une énorme base de données ($2^{256} \sim 10^{75}$). Pour toutes ces raisons nous avons choisi l'approche de réduction de l'image.

Blanchiment des données et normalisation locale

Généralement, les images naturelles sont très structurés et contiennent d'importantes redondances statistiques, c'est à-dire que leurs pixels présentent de fortes corrélations [Attneave, 1954; Barlow, 2001]. Par exemple, il est bien connu que les images naturelles incluent des régularités importantes dans leurs statistiques de premier et second ordre (corrélations

spatiales). Ces statistiques peuvent être mesurées à l'aide d'une fonction d'autocorrélation ou de la densité spectrale de Fourier [Field, 1987]. Ces corrélations sont dues à la nature redondante des images naturelles (les pixels adjacents ont généralement de fortes corrélations, sauf autour des bords). La présence de ces corrélations permet, la reconstruction de l'image, par exemple, en utilisant les champs de Markov. Il a ainsi été montré par [Bell and Sejnowski, 1997; Field, 1987; Olshausen and Field, 1996] que les arêtes sont les principales caractéristiques des images naturelles et qu'elles sont plutt codées par des dépendances statistiques d'ordre supérieur. On peut déduire de cette observation que les statistiques des images naturelles ne sont pas gaussiennes comme démontré précédemment (puisque les moments supérieurs à l'ordre deux sont nuls pour les distributions gaussiennes). Ces statistiques sont dominées par des événements rares comme les contours, conduisant à des kurtosis élevés.

Les prétraitements visant à éliminer ces corrélations d'ordre deux sont connus sous le nom de blanchiment. Il a été montré que le blanchiment est une stratégie de prétraitement utile pour l'ICA [Hyvärinen and Oja, 2000; Soman et al., 2009]. Il est également une étape obligatoire pour l'utilisation de méthodes de classification dans la reconnaissance d'objets [Coates et al., 2011]. Le blanchiment est un processus linéaire. Par ailleurs, il ne supprime pas les statistiques d'ordre supérieur ou encore les régularités présentes dans les données. Théoriquement, le blanchiment est une tâche simple. Après centrage, les vecteurs de données sont projetés sur les axes principaux (calculés comme des vecteurs propres de la matrice de variance-covariance) et ensuite divisés par la variance le long de ces axes. De cette façon, le nuage de données présente une forme sphérique, laissant apparaître uniquement les axes correspondant généralement à ses ordres supérieurs de dépendances statistiques.

Une autre approche pour le prétraitement des données consiste à effectuer une normalisation locale. Dans ce cas, chaque correctif $x^{(i)}$ est normalisé en soustrayant la moyenne et en divisant le résultat par l'écart-type de ses éléments. Pour les données visuelles, cela correspond à la normalisation

locale de la luminosité et du contraste. On peut trouver dans [Coates et al., 2011] une étude sur la normalisation locale et ses effets sur une tâche de classification. Cependant, on peut noter que cette étude a été effectuée en utilisant deux bases de données, NORB et CIFAR, qui ont été spécialement conçues pour la reconnaissance d'objets.

Nous pouvons également noter que dans [Ranzato et al., 2010], les auteurs affirment que le blanchiment accélère le convergence de l'algorithme. Cependant, ce résultat n'a pas été justifié.

Élaboration de caractéristiques spatiales non supervisée

Machine de Boltzmann Restreinte (RBM avec Gaussienne-Bernoulli)

À la différence de la machine de Boltzmann, une RBM est un modèle graphique non orienté bipartite $\theta = \{w_{ij}, b_i, c_j\}$, qui apprend un modèle généré à partir de données observées. Elle consiste en deux couches. La couche cachée, contenant des variables latentes $\mathbf{h}$, est utilisée pour générer la couche visible, contenant les variables observées $\mathbf{v}$. Dès que la génération $P(\mathbf{v}|\mathbf{h})$ a appris, les connexions non orientées peuvent déterminer $P(\mathbf{h}|\mathbf{v})$. Les deux couches sont entièrement connectées par le biais d'un ensemble de poids w_{ij} et les biais $\{b_i, c_j\}$ et il n'y a pas de connexion entre les unités d'une même couche. Dans un RBM classique, la configuration des connexions entre les unités binaires visibles et les unités binaires cachées a une fonction d'énergie $E(\mathbf{v}, \mathbf{h}; \theta)$ donnée par :

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_{i \in \mathbf{v}} b_i v_i - \sum_{j \in \mathbf{h}} c_j h_j \tag{1}$$

La probabilité de l'état d'une unité en une seule couche est basée sur l'état de l'autre couche et peut donc être aisément calculée. Selon la distribution de Gibbs:

$$P(\mathbf{v}, \mathbf{h}; \theta) = -\frac{1}{Z(\theta)} \exp^{-E(\mathbf{v}, \mathbf{h}; \theta)} \tag{2}$$

où $Z(\theta)$ est une constante de normalisation. Ainsi, après la marginalisation, la probabilité d'une configuration cachée de l'état **h** peut être dérivée comme suit :

$$P(\mathbf{h};\theta) = \sum_{\mathbf{v}} P(\mathbf{v},\mathbf{h};\theta) = \frac{\sum_{\mathbf{v}} e^{-E(\mathbf{v},\mathbf{h};\theta)}}{\sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h};\theta)}} \tag{3}$$

Cependant, selon [Krizhevsky, 2009], la probabilité conditionnelle ci-dessus peut être calculée en utilisant la fonction logistique sigmoïde comme suit :

$$P(h_j = 1 \mid \mathbf{v};\theta) = \sigma(c_j + \sum_i w_{ij} v_i) \tag{4}$$

où $\sigma(x) = 1/(1+e^{-x})$ est la fonction logistique. Une fois que les états binaires cachés sont échantillonnés, nous produisons une "reconstruction" de la mini-image d'origine en mettant l'état de chaque unité visible à la valeur 1 avec une probabilité :

$$P(v_i = 1 \mid \mathbf{h};\theta) = \sigma(b_i + \sum_j w_{ij} h_j). \tag{5}$$

Cependant, des unités visibles logistiques ou binaires ne sont pas appropriées pour coder des valeurs multiples en entrées comme les niveaux de gris des pixels, parce que les unités logistiques représentent mal des données telles que les sous-images d'images naturelles. Pour surmonter ce problème, comme l'a suggéré [Hinton, 2010], dans le présent travail, nous remplaçons les unités binaires visibles par un système d'activation gaussienne avec moyenne nulle comme suit :

$$P(v_i = 1 \mid \mathbf{h};\theta) \leftarrow \mathcal{N}(b_i + \sum_j w_{ij} h_j, \sigma^2) \tag{6}$$

où σ^2 désigne la variance du bruit. Dans ce cas, la fonction d'énergie de RBM avec Gaussienne-Bernoulli est donnée par:

$$E(\mathbf{v},\mathbf{h};\theta) = \sum_{i\in\mathbf{v}} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j\in\mathbf{h}} c_j h_j - \sum_i \sum_j \frac{v_i}{\sigma_i} h_j w_{ij} \tag{7}$$

Apprentissage RBM avec une contrainte de parcimonie

Pour connaître les paramètres RBM, il est possible de maximiser la log-vraisemblance dans une procédure de descente de gradient. Ainsi, la dérivée du modèle du logarithme népérien de la vraisemblance sur un ensemble d'apprentissage D est donnée par:

$$\frac{\partial}{\partial\theta}L(\theta) = \left\langle \frac{\partial E(\mathbf{v},\theta)}{\partial\theta} \right\rangle_M - \left\langle \frac{\partial E(\mathbf{v},\theta)}{\partial\theta} \right\rangle_D \tag{8}$$

où le premier terme correspond à la moyenne par rapport au modèle de distribution et le second correspond à l'espérance sur les données. Bien que le second terme soit simple à calculer, le premier est souvent insoluble. Cela est dû au fait que le calcul de la vraisemblance a besoin du calcul de la fonction de partition, $Z(\theta)$, qui est habituellement impossible à calculer. Une méthode de type Markov-Chain Monte Carlo, comme l'échantillonnage de Gibbs, peut être utilisée pour calculer l'espérance. Ces méthodes, cependant, sont très lentes et souffrent d'une forte variance dans leurs estimations.

En 2002, Hinton a proposé une procédure d'apprentissage rapide appelé Divergence Contrastive (Contrastive Divergence : CD) [Hinton, 2002]. Cet algorithme d'apprentissage est basé sur le fait que minimiser l'énergie du réseau revient à minimiser la distance entre les données originales et les données statistiques générées. La comparaison est faite entre les statistiques des données et des statistiques générées par un échantillonnage de Gibbs. Par conséquent, dans l'apprentissage des CD, nous essayons de minimiser la distance de Kullback-Leibler entre la distribution des données, Q^0, et le modèle de distribution, Q^∞, comme suit:

$$CD_n = KL(Q^0||Q^\infty) - KL(Q^1||Q^\infty) \tag{9}$$

Le principal avantage de cet algorithme, est que les termes irréductibles, Q^∞, dans l'équation ci-dessus s'annulent les uns les autres, comme il est expliqué dans [Andrzejewski, 2009; Hinton, 2002]. Cela signifie que,

dans la pratique, nous utilisons habituellement seulement quelques pas de l'échantillonnage de Gibbs (la plupart du temps ré duit à un) pour assurer la convergence. Pour une RBM, les poids du réseau peuvent donc être mis à jour à l'aide de l'équation suivante:

$$-\frac{\partial}{\partial w_{ij}}\left(Q^0\|Q^\infty - Q^1\|Q^\infty\right) \approx \langle v_i^0 h_j^0\rangle_{Q^0} - \langle v_i^n h_j^n\rangle_{Q^1} \tag{10}$$

Cette équation peut être réécrite comme suit :

$$w_{ij} \leftarrow w_{ij} + \eta(\langle v_i^0 h_j^0\rangle_{data} - \langle v_i^n h_j^n\rangle_{recon.}) \tag{11}$$

où η est le taux d'apprentissage, v^0 correspond à la distribution de données initiales, h^0 est calculé en utilisant l'équation 4, v^n est échantillonné à l'aide de la distribution Gaussienne de l'équation 6 et avec n pas d'échantillonnage de Gibbs. h^n est de nouveau calculée à partir de l'équation 4. En outre, les règles de mise à jour des biais des neurones visibles et cachés sont similaires à la règle de mise à jour pour les poids:

$$b_i \leftarrow b_i + \eta[\langle v_i^0\rangle_{data} - \langle v_i^n\rangle_{recon.}] \tag{12}$$

et

$$c_j \leftarrow c_j + \eta[\langle h_j^0\rangle_{data} - \langle h_j^n\rangle_{recon.}] \tag{13}$$

où v_i, h_j, b_i, et c_j désignent le i-ième neurone visible, le j-ième neurone caché, le i-ième biais visible, et le j-ième biais caché respectivement.

En ce qui concerne la contrainte de parcimonie dans les RBMs, nous suivons l'approche développée dans [Lee et al., 2008]. Cette méthode introduit un terme de régularisation qui réduit les activations moyennes des variables cachées sur l'ensemble des exemples de formation. Ainsi, l'activation des neurones du modèle devient également clairsemée. En fait, cette méthode est similaire à celle utilisée dans d'autres modèles Olshausen and Field [1996]. Ainsi, comme illustré dans [Lee et al., 2008], étant donné un ensemble d'apprentissage $\{v^{(1)}, \ldots, v^{(m)}\}$ qui comprend m

exemples, nous posons le problème d'optimisation suivant:

$$minimize_{\{w_{ij},b_i,c_j\}} -\sum_{l=1}^{m} log\left(\sum_h P(\mathbf{v}^{(l)},\mathbf{h}^{(l)})\right) + \lambda \sum_{j=1}^{n} \left| p - \frac{1}{m}\sum_{l=1}^{m} \mathbb{E}[h_j^{(l)}|\mathbf{v}^{(l)}] \right|^2, \tag{14}$$

où $\mathbb{E}[.]$ est l'espérance conditionnelle en fonction des données, p est la cible contrlant de la parcimonie des unités cachées h_j, et λ est le coût de parcimonie. Ainsi, après avoir employé cette régularisation dans l'algorithme d'apprentissage de CD, le gradient du terme de régularisation de parcimonie sur les paramètres (poids w_{ij} et les biais cachés c_j) peut être écrite comme suit:

$$w_{ij} \leftarrow \mu * w_{ij} + \eta * \left[\left(\left(\langle v_i^0 h_j^0\rangle - \langle v_i^n h_j^n\rangle\right)\right) - \lambda * (p - \frac{1}{m}\sum_{l=1}^{m} p_j^{(l)}), \right. \tag{15}$$

$$c_j \leftarrow c_j + \eta[\langle h_j^0\rangle_{data} - \langle h_j^n\rangle_{recon}] - \lambda * (p - \frac{1}{m}\sum_{l=1}^{m} p_j^{(l)}), \tag{16}$$

où m dans ce cas est la taille du mini-batch et $p_j^{(l)} \triangleq \sigma(\sum_i v_i^{(l)} w_{ij} + c_j)$.

Il a été montré que l'algorithme d'apprentissage clairsemé RBM peut capturer d'intéressantes caractéristiques d'ordre supérieur à partir d'images naturelles [Lee et al., 2008]. Nous espérons qu'un tel algorithme d'apprentissage reste capable de capturer des caractéristiques d'ordre supérieur à partir de diverses bases de données, comme par exemple une base de données créée afin de localiser d'un robot.

Apprentissage par couche pour les DBNs

Les RBM peuvent être empilées pour produire une architecture DBN, où les paramètres du modèle θ_i, à la couche i, sont appris en gardant les paramètres du modèle dans la partie inférieure des couches constants. Autrement dit, l'algorithme d'apprentissage DBN forme les couches RBM d'une façon gloutonne par couche. Les paramètres du modèle à la couche $i-1$ sont figés et les probabilités conditionnelles des valeurs unitaires cachées sont utilisées afin de générer les données nécessaires pour entraîner les paramètres du modèle à la couche i. Ce procédé peut être répété

à travers les couches pour obtenir des représentations creuses des données initiales qui seront utilisées comme des vecteurs d'entrée pour effectuer le processus de classification.

Description des bases de données

La base de données d'images naturelles de Van Hateren

Afin d'étudier l'impact de la normalisation des données sur la détection de caractéristiques, nous utilisons une base de données populaire contenant des images naturelles, la base de données de Van Hateren. Il s'agit d'une base de données d'images de haute résolution, calibrées et monochromes prises dans des conditions d'éclairage définies, conçues pour différentes tâches de traitement d'images. Cette base contient environ 4000 images de résolution $1536x1024$ pixels.

Pour cette tâche, nous avons extrait aleatoirement un échantillon de 100000 de parcelles d'images 16×16. Ces parcelles sont ensuite blanchies en utilisant un algorithme de blanchiment et normalisées à l'aide d'une normalisation locale dans deux prétraitement distincts, tel qu'indiqué dans la figure 1.

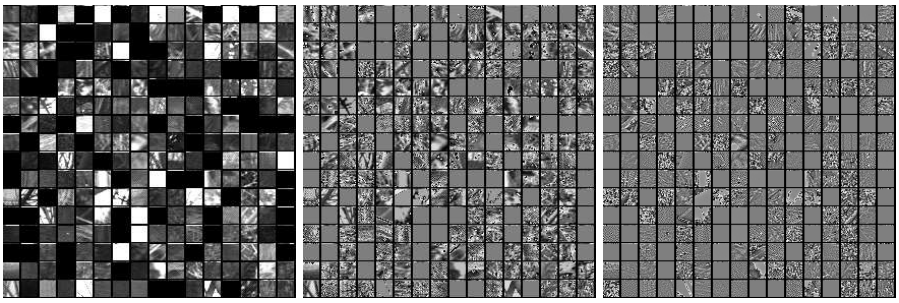

Figure 1: **Première colonne:** 256 patchs choisis au hasard à partir de la base de données de van Hateren. **Deuxième colonne:** Les éléments correspondants normalisés. **Troisième colonne:** Les éléments correspondants blanchis.

La base de données COLD

Cette base de données (base de données de localisation COSY) a été originellement développée par [Ullah et al., 2007] pour la localisation en robotique. Cette base contient une collection d'images étiquetées de resolution 640×480 acquises à cinq images par seconde lors de l'exploration

d'un robot de trois laboratoires differents: Freiburg, Ljubljana, et Saarbruecken. Deux ensembles de chemins (Type A et B) ont été acquis dans des conditions d'éclairage différentes (ensoleillé, nuageux et nuit), et pour chaque condition, un chemin consiste à visiter différentes pièces (couloirs, zones d'impression, *etc*.). Ces promenades à travers les laboratoires sont répétées plusieurs fois. Bien que les images en couleur ont été enregistrées au cours de l'exploration, seules les images en niveaux de gris sont utilisées puisque des travaux antérieurs ont démontré que dans les couleurs de la base de données COLD sont faiblement informatives et rendent le système plus dépendant de l'éclairage [Ullah et al., 2007].

Tel que proposé par [Torralba et al., 2008], la taille de l'image est réduite à 32×24 (voir, par exemple, la figure 2). La dernière série des mini-images (une nouvelle base de données appelée tiny-COLD) est centrée et blanchie/normalisée afin d'éliminer les statistiques de second ordre. Par conséquent, la variance dans l'équation 6 est définie à 1. Contrairement à Torralba, les $32 \times 24 = 768$ pixels des images blanchies ou normalisées sont utilisés directement en tant que vecteur d'entrée du réseau.

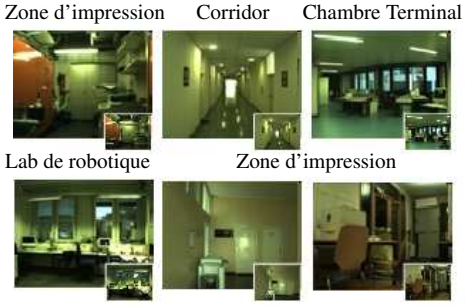

Figure 2: Des échantillons de la base de données initiale COLD. les mini-images correspondantes sont affichées en bas á droite. On peut voir que, malgré la réduction de la taille, ces mini-images restent pleinement reconnaissables.

Les résultats expérimentaux

Effet de la normalisation sur les caractéristiques spatiales

Pour cette tâche, nous avons mené deux expériences en utilisant un ensemble de données de patchs aléatoirement échantillonnés à partir de la base

de données de van Hateren. Après avoir décorrélé (algorithme de blanchiment) et normalisé des patchs en deux pré-processus séparés comme montré précédemment, une structure plus-complète $(256-512)$ de la première couche RBM a été utilisée.

La figure 3.20 (à gauche) montre des caractéristiques extraites en utilisant les données localement normalisées, tandis que la figure 3.20 (à droite) montre des caractéristiques extraites en utilisant les données blanchies. Il est évident que les caractéristiques extraites à partir des données blanchies sont plus localisées. Les données blanchies modifient clairement les caractéristiques apprises. Le lien entre les corrélations du second ordre et la présence de basses fréquences dans les images pourrait expliquer l'effet de blanchiment. Si l'algorithme de blanchiment enlève ces corrélations dans l'ensemble des données d'origine, cela produit des données ne couvrant que les fréquences spatiales élevées. Dans ce cas L'algorithme de RBM ne trouve que des caractéristiques de haute fréquence.

Toutefois, les caractéristiques apprises à partir des données de normalisation sont totalement différentes de celles apprises avec les données blanchies. Ces caractéristiques restent clairsemées, mais couvrent un large spectre de fréquences spatiales. Il est intéressant de noter que ces caractéristiques ont l'air plus proches de celles obtenues avec les réseaux à convolution Lee et al. [2009] pour lesquels aucun blanchiment n'est appliqué aux données initiales. Nous pouvons remarquer que ces différences entre les données normalisées et blanchies ont déjà été observées dans Krizhevsky [2009]. Il a obtenu de meilleures performances en utilisant des caractéristiques tirées des données normalisée sur CIFAR-10 dans une tâche de reconnaissance d'objets.

Pour essayer de comprendre plus profondément pourquoi les caractéristiques obtenues à partir de patchs blanchis ou normalisés sont différentes, nous avons calculé la densité spectrale moyenne de Fourier des patchs dans les deux conditions, et nous l'avons comparée à la même fonction pour les patchs originaux. Nous avons tracé la moyenne du logarithme de la densité de puissance spectrale de la transformée de Fourier de tous les patchs

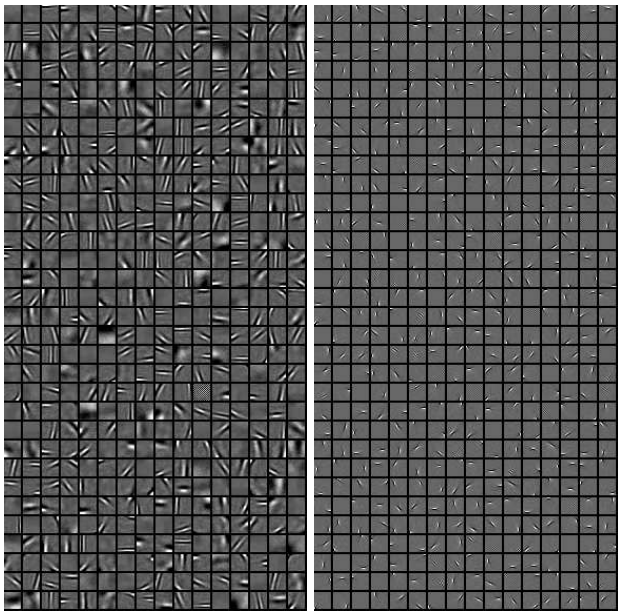

Figure 3: Bases sur-complète extraites d'images naturelles. **A gauche :** 512 les caractéristiques appris par l'apprentissage de la couche RBM première en utilisant de patchs normalisée (16×16) échantillonnées à partir de van Hateren base de données. **A droite:** Caractéristiques correspondantes acquises par l'apprentissage de la première couche RBM en utilisant des patchs blanchis (16×16) échantillonnés à partir de la même base de données. Pour les deux expériences. Le protocole d'apprentissage est similaire à celui proposée dans Lee et al. [2008] (300 époques, taille de mini-batch 200, taux d'apprentissage $0,02$, moment initial $0,5$, moment final $0,9$, décroissance des poids $0,0002$, un paramètre de parcimonie de $0,02$ et un coût de parcimonie de $0,02$).

selon les fréquences comme indiqué dans la figure 4. La loi d'échelle en $1/f^{\alpha}$ caractéristique des images naturelles est approximativement vérifiée comme prévu pour les patchs initiaux. Pour la normalisation locale, la loi d'échelle est aussi conservée (le décalage entre les deux courbes est uniquement du à une différence de multiplication de l'amplitude du signal entre l'original et les patchs localement normalisés). Cela signifie que la composition de fréquence des images localement normalisés ne diffère de la première que par un facteur constant. La composition de fréquence relative est la même que dans les images initiales.

Au contraire, le blanchiment supprime complètement la dépendance entre l'énergie du signal et la fréquence. Cela signifie que le blanchiment égalise le rle de chaque fréquence dans la composition des images. Ceci

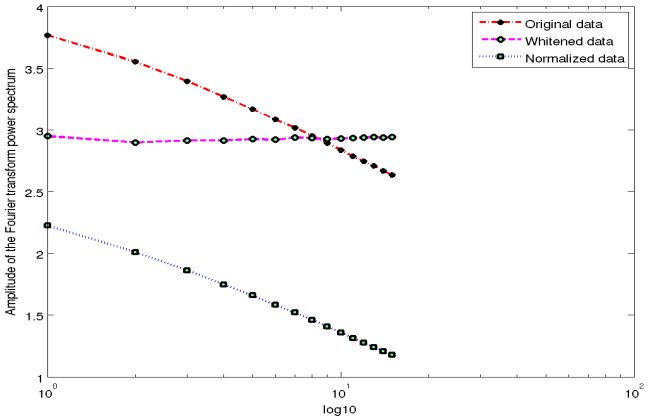

Figure 4: La représentation Log-Log du spectre de Fourier puissance moyenne pour les patchs d'image avec et sans normalisation. 256 de 16×16 patchs ont été extraites de la base de données van Hateren et puis normalisées. Le Log de la transformée de Fourier de chacun de ces patchs a été calculé et tracé selon le Log de la fréquence spatiale.

suggère une relation entre la loi d'échelle des images naturelles et les deux premiers moments de la statistique de ces images. Il est nécessaire de souligner que nous avons une manifestation du lien entre les propriétés statistiques d'une image et ses propriétés structurelles (en termes de fréquences spatiales). Ce lien est bien illustré à travers le théorème de Wiener-Khintchine et la relation entre la fonction d'auto-corrélation de l'image et sa densité spectrale de puissance. En ce qui concerne les caractéristiques extraites, les remarques citées ci-dessus permettent de déduire que la représentation similaire (en termes d'amplitude) de toutes les fréquences dans le signal initial donne lieu à une sur-représentation des hautes fréquences dans les caractéristiques obtenues. Cela peut être dû au fait que, dans les données blanchies, l'énergie contenue dans chaque bande de fréquence augmente avec la fréquence pendant qu'elle est constante dans les images initiales ou normalisées.

Toutefois, le résultat dépend de la base de données utilisée et par conséquent des fréquences spatiales contenues dans les patchs initiaux. Le fait que la normalisation locale conserve (à une constante près) la même composition de fréquence que dans données initiales. Cela prouve que la normalisation ne supprime pas entièrement les corrélations du second or-

dre. Olshausen [Olshausen and Field, 1997] a montré que, en utilisant le blanchiment, L'analyse en composantes indépendantes (Independent Component Analysis : ICA) conserve principalement des filtres dans une gamme étroite de fréquences spatiales. Les basses fréquences spatiales sont sous-représentées dans le résultat obtenu. Ces remarques concernent les résultats obtenus en utilisant les données de blanchiment. Cependant, dans le cas des données de normalisation, les caractéristiques enregistrent une plus large gamme de fréquences spatiales.

Les dépendances entre les basses fréquences sont liées à la corrélation statistique entre les pixels voisins. Ainsi, la suppression de ces corrélations du second ordre supprimerait ces basses fréquences dans les patchs blanchis. Nous observons que les caractéristiques, qui sont moins localisées, ont plus de chances de contenir un plus grand nombre de basses fréquences.

Dans la section suivante nous présentons comment nous avons utilisé la base de données COLD pour tester notre modèle SPR selon ces deux méthodes de normalisation. Nous présentons également comment ces changements dans la composition de fréquence spatiale affectent les performances de classification.

Extraction des caractéristiques: l'alphabet

Des essais préliminaires ont montré que la structure optimale du DBN en termes de score final de classification est $768 - 256 - 128$. Les caractéristiques indiquées sur la figure 5 (à gauche) ont été extraites par apprentissage de la couche RBM sur 137.069 patchs blanchis ($32x24$ pixels) échantillonnés à partir de la base de données COLD. Certains d'entre eux représentent des parties du couloir, qui est sur-représenté dans la base de données. Il correspond à de longues séquences d'images très similaires lors de l'exploration du robot. D'autres sont localisées et correspondent à de petites parties des vues initiales, comme les bords et les coins, qui peuvent être identifiés comme éléments de pièce, c'est à-dire qu'ils ne sont pas spécifiques á pièce donnée). Les caractéristiques indiquées sur la figure 5 (à droite) ont été obtenues en utilisant les données normalisées. Comme nous l'avons observé précédemment pour la base de

données de van Hateren, les caractéristiques obtenues sont très différentes. Les parties de pièces sont beaucoup plus représentés que dans la base de données blanchie. Nous remarquons que la gamme de fréquences spatiales couverte par les caractéristiques est beaucoup plus large. Dans les deux cas, les combinaisons de ces caractéristiques initiales dans les couches supérieures correspondent aux structures les plus caractéristiques des différentes pièces.

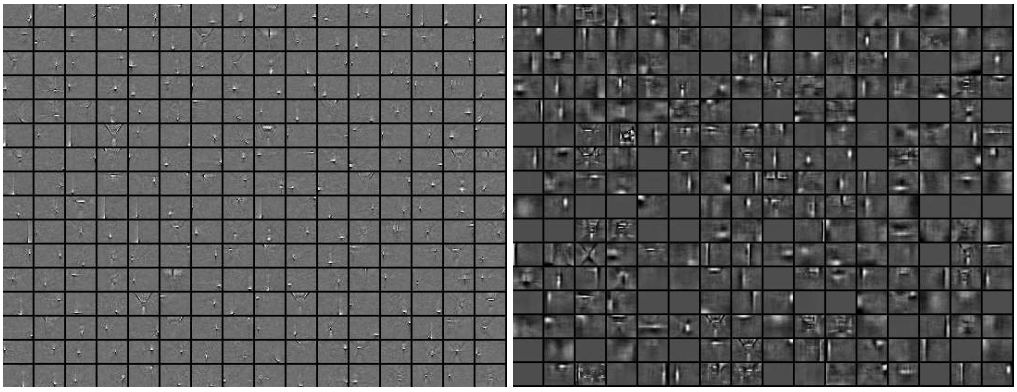

Figure 5: **A gauche:** Les 256 filtres obtenus par l'apprentissage d'une première couche de RBM 32×24 avec des patchs blanchis échantillonnés à partir de la base de données COLD. **A droite:** les 256 filtres obtenus par l'apprentissage d'une première couche de RBM $32x24$ avec des patchs normalisée échantillonnés à partir de la base de données COLD. Le protocole de l'apprentissage est similaire à celles proposée dans Krizhevsky [2010]; Lee et al. [2008] (300 époques, taille de mini-batch 100, taux d'apprentissage $0,002$, décroissance des poids $0,0002$, moment initial $0,5$, moment final de $0,9$, paramètre de parcimonie $0,02$, coût de parcimonie $0,02$).

Apprentissage supervisé des lieux

Après la réalisation de la représentation appropriée en fonction des DBNs, une classification a été effectuée dans l'espace des caractéristiques comme le montre le tableau 1 (la deuxième ligne). En supposant que la transformation non linéaire exploitée par les DBN améliore la séparabilité linéaire des données, une méthode de régression simple a été utilisée pour effectuer le processus de classification dans le cas initial. Pour exprimer le résultat final comme une probabilité qu'une vue donnée appartienne à une seule pièce, nous normalisons le résultat en utilisant la méthode de régression

softmax. Nous avons également étudié la phase de classification en utilisant un classifieur non-linéaire, comme Support Vector Machine (SVM). Nous avons utilisé ce classifieur non-linéaire pour démontrer que le DBN calcule une signature séparable linéairement et donc il ne devrait pas affecter les résultats de la classification finale.

Laboratory name	Saarbruecken			Freiburg			Ljubljana		
Condition / Training	Cloudy	Night	Sunny	Cloudy	Night	Sunny	Cloudy	Night	Sunny
Ullah	84.20%	86.52%	**87.53%**	79.57%	75.58%	77.85%	84.45%	87.54%	**85.77%**
No thr.	70.21%	70.80%	70.59%	70.43%	70.26%	67.89%	72.64%	72.70%	74.69%
SVM	69.92%	71.21%	70.70%	70.88%	70.46%	67.40%	72.20%	72.57%	74.93%
0.55 thr.	**84.73%**	**87.44%**	87.32%	**85.85%**	**83.49%**	**86.96%**	**84.99%**	**89.64%**	85.26%

Table 1: Résultats de la classification moyenne pour les trois laboratoires différents et les trois conditions de l'apprentissage. **Première ligne:** le travail de Ullah; **deuxième ligne:** résultats bruts sans seuil; **troisième ligne:** taux de classification en utilisant SVM classifieur; **quatrième ligne:** taux de classification avec seuil, comme indiqué dans le texte. Nos résultats ont été obtenus sur la base des caractéristiques apprises á partir des données blanchies.

Les échantillons prélevés dans chaque laboratoire et chaque état d'éclairage ont subi un apprentissage séparément, comme dans [Lee et al., 2008]. Pour chaque image, le résultat du réseau softmax a donné la probabilité d'être dans chacune des pièce visitées. Selon les principes du maximum de vraisemblance, la plus grande valeur de probabilité détermine la décision du système. Lorsque nous utilisons les caractéristiques extraites des données blanchies, on obtient une moyenne de bonnes réponses allant de 65% à 80% selon les différentes conditions et les laboratoires comme le montre le tableau 1 (la deuxième ligne). Plus précisément, on obtient $73,4\%$, $69,5\%$ et 71% pour les laboratoires COLD-Ljubljana, COLD-Fribourg et COLD-Sarrebruck respectivement, et avec une moyenne globale de réponses correctes de $71,3\%$. En revanche, lorsque nous utilisons les caractéristiques extraites des données normalisées, on obtient une moyenne de bonnes réponses allant de 71% à 90% selon les différentes conditions et les laboratoires comme le montre le tableau 2 (la deuxième ligne). Plus précisément, on obtient $83,13\%$, $80,515\%$ et $81,5\%$ pour les laboratoires

présentés ci-dessus, et avec une moyenne globale de réponses correctes de $81,375\%$. Les derniers résultats sont comparables aux meilleurs résultats donnés dans [Lee et al., 2008]. Les résultats restent robustes aux variations d'illumination comme dans [Lee et al., 2008].

Laboratory name	Saarbruecken			Freiburg			Ljubljana		
Condition / Training	Cloudy	Night	Sunny	Cloudy	Night	Sunny	Cloudy	Night	Sunny
Ullah	84.20%	86.52%	**87.53%**	79.57%	75.58%	77.85%	84.45%	87.54%	85.77%
No thr.	80.41%	81.29%	83.66%	**81.65%**	**80.08%**	**79.64%**	83.14%	82.38%	83.87%
0.55 thr.	**86.00%**	**88.35%**	87.36%	**88.15%**	**85.00%**	**87.98%**	**85.95%**	**90.63%**	**86.86%**

Table 2: Résultats de la classification moyenne pour les trois laboratoires différents et les trois conditions de l'apprentissage. **Première ligne:** le travail de Ullah; **deuxième ligne:** résultats bruts sans seuil; **troisième ligne:** taux de classification avec un seuil, comme indiqué dans le texte. Nos résultats ont été obtenus sur la base des caractéristiques tirées des données normalisée.

Ces résultats démontrent qu'un RBM calculé à partir de données normalisées est plus performant qu'un RBM provenant de données blanchis. Ceci illustre le fait que le processus de normalisation conserve plus d'informations ou de structures provenant des images initiales. En effet, ces structures sont très importantes pour le processus de classification. D'autre part, le blanchiment enlève complètement les statistiques d'ordre un et deux à partir de la donnée initiale. Cette dé-corrélation permet au DBN d'extraire des caractéristiques d'ordre supérieur. Cela démontre que les données de blanchiment pourraient être utiles pour le codage d'images. Cependant, ce n'est pas la méthode de pré-traitement optimale dans le cas de la classification d'images.

Toutefois, il existe deux stratégies différentes pour améliorer ces résultats. La première est d'utiliser l'intégration temporelle tel que proposé dans [Guillaume et al., 2011]. La seconde stratégie s'appuie sur la théorie de la décision. Le taux de détection présenté dans le tableau 1 (deuxième ligne) a été calculé à partir des classes ayant les plus grandes probabilités, quelles que soient les valeurs relatives de ces probabilités. Certaines de

ces probabilités sont proches de la chance (dans notre cas $0,20$ ou $0,25$, selon le nombre de catégories à reconnaître) et il est évident que dans de tels cas, la confiance dans la décision rendue est faible. Ainsi, en dessous d'un seuil donné, lorsque la distribution de la probabilité tend à devenir uniforme, on pourrait considérer que la réponse donnée par le système n'a pas de signification. Cela pourrait être dû au fait que l'image donnée contient des caractéristiques communes ou des structures qui peuvent être trouvées dans deux ou plusieurs classes. L'effet du seuil est alors d'éliminer les résultats les plus incertains. Le tableau 1 (troisième ligne) montre les résultats de la classification moyenne pour un seuil de $0,55$ (seuls les résultats où $\max_X p(X = c_k|I) \geq 0.55$, où $p(X = c_k$ est la probabilité que le point de vue actuel I appartient à c_k, sont conservés). Ces résultats ont été obtenus en utilisant les caractéristiques extraites à partir des données blanchies. Dans ce cas, le taux d'acceptation moyen (le pourcentage d'exemples pris en compte) varie de 75% à 85%, selon le laboratoire. Les résultats obtenus ici sont meilleurs que ceux publiés dans [Ullah et al., 2008]. Lorsque l'on considère l'ensemble des résultats obtenus par apprentissage et par tests avec des conditions de luminosité semblables, nous avons obtenu un taux de classification moyen de $90,68\%$ pour COLD-Saarbrucken laboratoire, $89,88\%$ pour COLD-Freiburg laboratoire et $90,66\%$ pour COLD-Ljubljana laboratoire.

Comme les résultats présentés dans [Ullah et al., 2008] la performance a diminué pour les expériences menées dans des conditions de luminosité différentes. Dans ce cas, nous avons obtenu des taux de classification de $83,683\%$ pour COLD-Saarbrucken laboratoire, $83,14\%$ pour COLD-Freiburg laboratoire et $84,62\%$ pour COLD-Ljubljana laboratoire.

Nous avons également appliqué la méthode du seuil sur les résultats obtenus avec les données normalisées localement. Le tableau 2 (deuxième ligne) montre les résultats de la classification moyenne en utilisant un seuil similaire $(0,55)$. On remarque que le taux moyen des images acceptées a augmenté pour se situer entre 86% à 90%, selon le laboratoire. ces résultat

démontrent qu'un nombre plus élevé d'images a été utilisé dans la classification que dans l'expérience précédente. En outre, les résultats moyens sont largement meilleurs que ceux publiés dans [Ullah et al., 2008]. Ceci indique que la séparabilité linéaire des données a été significativement améliorée dans le cas de l'utilisation des données normalisées pour l'extraction de caractéristiques.

En ce qui concerne la sensibilité à la luminosité. Dans les deux cas, nos résultats semblent être moins sensibles aux conditions d'illumination par rapport aux résultats obtenus dans [Ullah et al., 2008]. Comme dans les expériences précédentes, nous avons constaté une faible performance sur les données COLD-Freiburg données, ce qui confirme que cette collection est la plus difficile de toute la base COLD comme indiqué dans [Ullah et al., 2008]. Toutefois, dans le cas de l'utilisation des fonctions apprises à partir des données non blanchies, avec et sans seuillage, nos résultats de classification pour le laboratoire Freiburg dépassent les meilleurs obtenus par [Ullah et al., 2008].

En règle générale, les tableaux 1 et 2 montrent une comparaison globale de nos résultats avec ceux de [Ullah et al., 2008] pour les trois conditions d'apprentissage. Ils montrent également les résultats obtenus en utilisant une classification SVM au lieu d'une régression softmax. Les résultats obtenus sont tout à fait comparables à softmax montrant que le DBN calcule une signature linéairement séparable. Ils soulignent le fait que les éléments appris par l'approche DBNs sont plus robustes pour une tâche de reconnaissance de lieu sémantique que l'extraction des caractéristiques *ad-hoc* basée sur les descripteurs (GiST, CENTRIST, SURF, et SIFT).

Conclusion et perspectives

Le but de cette thèse était d'étudier l'utilisation de DBNs dans une tâche de reconnaissance d'image difficile, la reconnaissance sémantique de lieux. Nos résultats montrent qu'une approche fondée sur des images miniature suivie d'une projection sur un espace de caractéristiques approprié peut obtenir des résultats intéressants dans la classification d'une tâche de reconnaissance de lieux sémantiques. Ils ont dépassé les performances

des meilleures publications [Ullah et al., 2008] basés sur des techniques plus complexes (utilisation de détecteurs SIFT suivie d'une classification SVM). Comme attendu, les résultats de classification ont été significativement meilleurs quand nous avons utilisé les caractéristiques tirées d'un ensemble de données normalisées localement. On peut dire que les caractéristiques extraites par les statistiques de premier et second ordre sont nettement meilleures que les caractéristiques extraites par les statistiques d'ordre supérieur en termes de classification comme déjà indiqué par Aggarwal and Agrawal [2012]. Toutefois, afin de reconnaître un lieu, il ne semble pas nécessaire de classer correctement chaque image du lieu. En ce qui concerne la reconnaissance de lieu, toutes les images ne sont pas instructives: certaines d'entre elles sont floues quand le robot tourne ou se déplace trop rapidement d'un endroit à un autre, d'autres ne montrent pas de détails informatifs (par exemple lorsque le robot est face à un mur). Comme le système proposé calcule la probabilité de la pièce la plus probable parmi toutes les pièces possibles, il offre la possibilité de pondérer chaque conclusion d'un facteur de confiance associé à la distribution de probabilité sur toutes les classes. On peut alors éliminer les images les plus incertaines, augmentant ainsi le score de reconnaissance. Il offre une alternative plus simple à la méthode proposée dans [Pronobis et al., 2006] basée sur l'intégration d'indices et le calcul d'un critère de confiance dans une approche de classification SVM.

L'apport fondamental de cette thèse est donc la démonstration que les DBNs couplés avec des mini-images peuvent être utilisées avec succès dans le cadre de la SPR. Ces considérations ont grandement contribué à la simplification de l'algorithme de classification global. En effet, ils apportent des vecteurs de codage qui peuvent être utilisés directement dans une méthode discriminante. À notre connaissance, c'est la première démonstration que l'extraction de caractéristiques à partir de mini-images normalisées en utilisant les DBNs est une approche discriminante alternative pour la SPR qui mérite d'être pris en considération.

Ainsi, la présente approche obtient des scores comparables aux approches basées sur des signatures obtenues manuellement (comme les détecteurs de GiST ou SIFT) et des techniques de classification plus sophistiqués comme SVM. Comme l'ont souligné [Hinton et al., 2011], les caractéristiques extraites par les DBNs sont plus prometteuses pour la classification d'images que les caractéristiques obtenues manuellement.

Différentes voies peuvent être utilisées dans des prochaines études pour étendre cette recherche. Une dernière étape d'ajustement fin peut être introduite à l'aide de rétro-propagation au lieu d'utiliser des caractéristiques grossières, comme illustré dans [Krizhevsky and Hinton]. Cependant, l'utilisation de caractéristiques grossières rend l'algorithme entièrement incrémentiel évitant l'adaptation à un domaine spécifique. La séparation stricte entre la construction de l'espace des caractéristiques et la classification permet d'étudier les problèmes de classification qui partagent les mêmes caractéristiques d'espace. L'indépendance de la construction des caractéristiques d'espace a un autre avantage dans le contexte de la robotique autonome: cela peut être considéré comme une maturation de développement acquise en ligne par le robot, une seule fois, au cours d'une phase d'exploration de son environnement. Une autre question n'a pas été étudiée dans ce travail et reste ouverte malgré quelques tentatives intéressantes [Guillaume et al., 2011; Ullah et al., 2008] il s'agit de la catégorisation de lieux basée sur la vision. La catégorisation est la façon de reconnaître le caractère fonctionnel d'une pièce, par exemple avec la base de données COLD la reconnaissance d'un bureau ou d'un couloir dans différents laboratoires. Ainsi, il pourrait être intéressant de voir si une approche basée sur les DBNs est capable d'améliorer les performances de catégorisation. En outre, il pourrait être également intéressant d'évaluer la performance de DBN sur les tâches de reconnaissance d'objets.

Declaration

I declare that this thesis is a presentation of my original research work and it was composed by myself. All contributions of other authors are involved in this thesis have been acknowledged with due reference to the literature. The whole research work was done under the supervision of Professor Philippe Tarroux and Dr. Emmanuelle Frenoux, at LIMSI-CNRS Laboratory, Paris Sud University, Paris.

Contents

List of Figures

List of Tables

List of Algorithms

List of acronyms

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
BMs	Boltzmann Machines
BoWs	Bag-of-Words
BP	Back-Propagation
CD	Contrastive Divergence
CENTRIST	CENsus TRansform hISTogram
COLD	COsy Localization Database
CRBM	Convolutional Restricted Boltzmann Machines
CT	Census Transform
DB	Database
DBM	Deep Boltzmann Machine
DBNs	Deep Belief Networks
DNNs	Deep Neural Networks
DPS	Direct Policy Search
EBMs	Energy-Based Models

EVD	Eigen-Value Decomposition
GBRBM	Gaussian-Bernoulli Restricted Boltzmann Machine
GiST	Generalized Search Tree
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HLDs	Harries Laplace Detectors
HMM	Hidden Markov Model
ICA	Independent Component Analysis
IDOL	Image Database for rObot Localization
KLD	Kullback-Leibler Divergence
KMC	k-Means Clustering
KNN	K-Nearest Neighbor Rule
$LBP_{8,1}$	Local Binary Pattern Code
LFW	Labeled Faces in the Wild database
MAP	Maximum A Posterior
MCL	Monte Carlo Localization
MCMC	Markov-Chain Monte Carlo
MCMs	Monte Carlo Methods
MIT	Massachusetts Institute of Technology
ML	Maximum Likelihood
ML	Machine Learning
MLP	Multi-layer Perceptron

MMCL	Mixture Monte Carlo Localization
MMH	Maximum Margin Hyperplane
NBC	Naive Bayes Classifier
NNs	Neural Networks
PCA	Principal Component Analysis
PDF	Probability Density Function
PoE	Product of Experts
RBMs	Restricted Boltzmann Machines
RMS	Root Mean Square
ROI	Region Of Interest
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SPR	Semantic Place Recognition
SURF	Speeded Up Robust Features
SVC	Support Vector Classifier
SVM-C	Support Vector Machine Classifier
SVMs	Support Vector Machines
TDMs	Temporal Difference Methods
XOR	Exclusive OR

Chapter 1

Introduction

1.1 Introduction

Today, mobile robotic systems are widely used in the industry to perform different tasks such as packaging, painting, welding, *etc*. However, robotics research develops more sophisticated applications that need the robot to be autonomous. Autonomy means that the robot will be able to decide by itself what behavior to adopt in unknown and uncertain environments. One type of tasks that require autonomy are the one involving people since people behavior is often unpredictable. Situations in which robots work in interaction with people are numerous. However, they often require the knowledge of the human environments. Personal assistance for example needs that the robots has a knowledge of the organization of the environment. In order to achieve that, the robot needs to locate itself. The answer of questions like "Where am I?", "How do I get there?", and "Where am I going to?" will allow the robot to behave and interact successfully and freely. Localization, mapping, and semantic place recognition seem to be required to solve these questions. Most of works have focused on the first two tracks. On the contrary, the last point has been addressed only recently and is currently always under investigations.

Whenever and wherever the robots are designed to behave and interact with the users, semantic information concerning places can be interesting and important. If the robot location is correctly recognized, its behavior will be improved for a lot of different applications and tasks. This is one of the main reasons why, in this research, we address the problem of robot localization based on semantic cues.

In this introduction, we start by identifying the research problem statement and the substantial motivations to propose a new approach to this problem. We then describe the main objectives and advantages to develop this work. We end this chapter by presenting an outline of the remaining chapters of this thesis.

1.2 Problem definition and overview

Robot localization is one of the major problems for the future of autonomous systems. If the robot does not know where it is, it will be very difficult to do further processes or tasks. The robot will indeed need to have at least some information about where it is to be able to operate and interact successfully [Kor, 1998; Borenstein et al., 1997]. The question of semantic place recognition poses immediately one important problem : what are the categories a robot should be able to recognize and what are the mechanisms by which human and animals recognize the categories of their environments. It seems obvious that the categories that divide the world of a fly, a dog or a human are deeply different. This seems to be due to the fact that at least some of these categories involve specific functionalities that each of these animals must use for recognition. The fly is attracted by soft meals, the dog recognizes its kennel to specific smells, the human forms categories with chairs since it can use them to sit. This ecological way to consider categories is surely the most adapted to future autonomous robotics. It would involve reinforcement and unsupervised learning but it seems for the moment largely out of reach. This is why the large majority of researches on place recognition have focused on recognition categories that humans make in their environment. Note that if the problem is less ecological than the recognition of genuine categories, it is helpful for interaction with humans. Besides, the problem is simpler than the previous one and can be solved using supervised learning methods.

Probabilistic approaches [Thrun et al., 2005] have given rise to Simultaneous Location and Mapping (SLAM) techniques. However, the place information in this case is different from the information used for the determination of the semantic categories of places. Beyond the precise metric localization given by SLAM, the ability for a mobile robot to determine the nature of its environment (kitchen, room, corridor, *etc*.) remains a challenging task. The knowledge of its metric coordinates or even the neighborhood information that can be encoded into topological maps is indeed not sufficient.

The semantic place recognition (SPR) is however required for a large set of tasks. It can for example be used as contextual information which fosters object detection and recognition (giving priors on object identity, location and scale) when it is achieved without any reference to the objects present in the scene. Moreover, it is able to build an absolute reference to the robot location, providing a simple solution for problems where the localization cannot be deduced from neighboring locations, such as in the kidnapped robot or the loop closure problems.

Most of the proposed approaches for place recognition or place categorization have used vision alone or combined with several types of telemeters such as laser or sonar (see [Pronobis et al., 2010]). Only few approaches have addressed this problem by integrating semantic information (see [Guillaume et al., 2011]). It is well known that vision provides richer information than telemeters which is an important advantage for fine discrimination. Vision also offers more portable and cost effective solutions. It also can provide information unavailable for other sensors, for instance, it can provide semantic information on a scene through the understanding of its visual appearance, not just the geometric aspect of it. This is why we are interested in visual semantic place recognition. This would improve the performance in terms of flexibility and complexity for the classification process.

In spite of recent works, current approaches to vision-based semantic place recognition are still facing several challenges. The complexity and adaptability are probably the most important ones. In this thesis, we focus on developing a system that should overcome some of these challenges. We tried an alternative approach than the ones proposed recently in the literature. All of the approaches require a description of the scene in terms of a signature. Two different approaches are possible, the one more traditional based on hand-crafted methods based on empirical descriptors (like SURF (Speeded Up Robust Features) or SIFT (Scale Invariant Feature transform) detectors), the second based on the use of an alphabet of features designed from theoretical considerations and able to create an appropriate representation of the initial images. Recently the use of Restricted Boltzmann Machines (RBMs) has been shown able to derive such an alphabet. The method we propose in this work is based on such approaches.

1.3 Research objectives

This thesis thus presents a novel approach for SPR based on RBMs and a direct use of tiny images. We will see that the major advantage of this model is that it provides a simple alternative to the existing approaches of SPR like [Pronobis and Caputo, 2007; Ullah et al., 2008]. In particular, the objectives of this thesis are itemized in the following points:

- The model must provide mobile robotic systems an ability to determine the current location based on semantic information.
- Ideally, the performance of the system should be directly proportional to the recognition accuracy, *i.e.* this system must be able to provide an accurate classification process.
- Since most of the current approaches are very complex, the designed system should simplify the overall classification process.
- The system should also demonstrate that Deep Belief Networks (DBNs) coupled with tiny images followed by a simple classifier, can be used as an alternative approach to achieve vision-based place recognition for autonomous agents.

The main objective of this work is thus to define a feasible simple algorithm for scalable semantic place recognition. The first two goals have already been achieved in most of the current approaches such as [Pronobis and Caputo, 2007; Ullah et al., 2008; Wu and Rehg, 2011] but irrespective to the complexity of the whole process. We want to show here that these goals can be achieved using DBNs coupled with tiny images, that would facilitate the classification process.

1.4 Thesis organization

The rest of the thesis is organized as follows. In chapter 2, we first introduce the problem of SPR in detail and then present background information about the existing SPR approaches. Energy-based models, such as Boltzmann Machines (BMs), RBMs, and DBNs, are also described in this chapter. The chapter also presents the Contrastive

Divergence (CD) learning algorithm for training RBM models. Then, it presents a theoretical background for SVM and softmax classifiers. Finally, it concludes towards the choices of the proposed model for a SPR task.

In chapter 3, using two standard databases of natural images, we experimentally investigate several parameters and factors that play important roles in having an optimal generative model. We also study the role of normalization on the selection of spatial ferquencies in the initial image set. The chapter concludes that DBNs can capture a set of high-level interesting features, and thus their use in image coding could simplify the problem of SPR.

In chapter 4, we describe the different phases of the model including, image preprocessing, image coding, and image classification, are explained. Then, we study what kind of features are extracted from datasets that are directly used as the input of the network. To do that we reduce the images to very small images ("tiny" images). We show that the loss of information is limited and that such tiny images can be used for SPR. We also study the effect of the normalization of the images on the extracted features to conclude that normalization extracts higher semantic level features than whitening.

In chapter 5, we use the COLD database of robot localization to present the final performances of the proposed model. Features extracted by training two RBM layers are then used to create a new representation of the initial data that is used as input to the classification. Several classification results using a linear and a nonlinear classifiers are presented.

In chapter 6, we present our conclusions and suggestions for future research. A number of directions for future development of a SPR using DBNs are given. Finally, several proofs are presented in the appendices.

Chapter 2

Background and related work

2.1 Introduction to semantic place recognition

The first part of the present chapter introduces the problem of semantic place recognition (SPR) for autonomous robotic systems in more details and the tracks followed to solve it. The SPR problem refers to distinguishing differences between different environmental locations (*e.g.* distinguishing a kitchen from an office). Usually, coding is the first step before recognizing the robot place. Thus, this chapter provides a detailed discussion of coding and learning methods that have been used to achieve SPR. The coding methods can be classified into object-based and view-based methods [Torralba et al., 2003a]. Object-based recognition methods are used to identify a large number of objects as an information to recognize the robot place, while view-based recognition approaches are used to compute directly a set of signatures or features, that exploit visual context, without having to identify specific regions or objects. These features can be used to generate a new representation of the initial data and then allow performing classification in the feature space. According to [Hsu and Griffiths, 2010], learning methods can also be categorized into generative and discriminative approaches. Generative approaches are used within the framework of naive Bayes classifier (NBC) and Bayesian filtering [Torralba et al., 2003b; Wu and Rehg, 2011] to learn a model fitting the original data. While discriminative approaches are used to directly discriminate data within the framework of Neural Networks (NNs) and support vector machines (SVMs) [Ullah et al., 2008]. A detailed description of these methods will be presented later in this chapter.

The second part of the chapter presents deep architecture models as alternative techniques to SVMs for the construction of the feature space. In this context, we first illustrate the main mathematical concepts of Energy-Based Models (EBMs), general Boltzmann Machines (BM), and Restricted Boltzmann Machines (RBMs). We then explain how to stack RBM models to generate Deep Belief Networks (DBNs). The chapter ends by introducing the different learning techniques such as maximizing the log-likelihood, Markov-Chain Monte-Carlo (MCMC) sampling methods like Gibbs sampling, and Contrastive Divergence (CD) that can be used to train Product of Experts (PoEs) models.

In general, knowing "where am I?" is a challenging question for mobile robotic platforms. Different researches have answered this question either by the use of metric localization or by the use of topological localization (mapping). Metric localization is the ability of a mobile robot to determine its position in a common coordinate frame, while topological localization is the ability for a mobile robot to produce a map of its environment.

For simple environments, it is possible to provide a map to the robot, for instance from building plans. However, these plans are not always accurate and do not consider the various objects inside each building which can impede the progress of the robot. Moreover, these plans are not always available (consider for example robots that must operate after a natural disaster). It is therefore necessary for the robot to map its environment while it explores. This problem is known as Simultaneous Localization and Mapping (SLAM). In fact, this problem is difficult because the construction of a map requires locating the different objects. However, the authors in [Thrun, 2001; Thrun et al., 2005] have proposed several probabilistic approaches for this problem. They are based on statistical models of sensors. Thus, in a navigation task, instead of giving a single estimation of the robot location, they proposed to use probabilistic algorithms which are based on the probability distribution of all locations. However, the ability for a mobile robot to determine the nature of its environment (the ability to distinguish different rooms using semantic cues) remains a challenging task in these approaches. This task is known as semantic place recognition.

The semantic category of a place gives priors on objects and defines what to do; for instance, the probability to find a television is higher in the living room than in the bathroom. So, if we first predict the robot place, then it will be easier to use this

place to recognize local objects. Besides, SPR build an absolute reference to the robot location, providing a simple solution for problems in which the localization cannot be deduced from neighboring locations, such as in the kidnapped robot or the loop closure problems. Furthermore, semantic place recognition will facilitate the human-robot interaction (with a topological map), *e.g.* the human can give the following command to the robot "go to the bedroom".

However, as mentioned earlier, performing semantic place recognition seems to be a challenging task for the following reasons:

- Firstly, the appearance of a room is not always stable due to the dynamical variations in time, like illumination condition changes (day or night), presence or absence of people, or even changing furnitures. For example, see the changes in figure 2.1. All these samples are selected from the COLD database [Ullah et al., 2007] [1] .

Figure 2.1: Dynamical variations due to the: (a) Influence of illumination or (b) Influence of human activity.

- Secondly, some classes share common visual features, *e.g.*, the offices in the COLD database as shown in figure 2.2 or even when the robot turns from one room to another one. It means that the variance between these classes is very small.

[1] The COLD database (COsy Localization database) is a collection of labeled images created for the purpose of robot localization. See Chapter 4 for a more in depth description.

Figure 2.2: Two different classes of 1-person and 2-person offices, but they have strong common visual features. All these samples are also selected from the COLD database [Ullah et al., 2007].

- Finally, the image annotation is usually based on the position of the robot during acquisition rather than the contents of the images. As a result, the labels might not be consistent with the visual information when the robot was positioned in a transition region between two rooms. This difficulty is illustrated in figure 2.3, which shows sample images of the interior of each room, captured with both perspective and omni-directional cameras [1] [Ullah et al., 2007],.

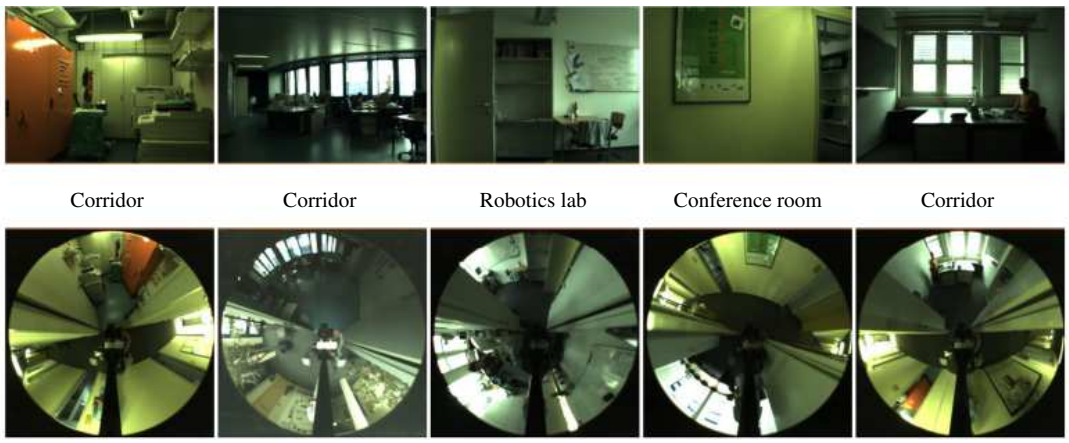

Figure 2.3: Exemplary images selected from the COLD-Saarbruecken dataset illustrating the limitations of the labeling technique. The figure shows images acquired with perspective camera with labels assigned on the basis of the location of the robot. The labels do not correspond to the visual information in the images due to the relatively narrow field of view of the cameras. The corresponding images are acquired using the omni-directional camera [Ullah et al., 2007].

In this work, we need to design an approach to perform the classification process taking into account illumination variations. We want to design a supervised machine

[1] Omnidirectional camera shows the interior of each room by a set of images rather than an image by itself as in the perspective camera.

learning approach by letting the closed-world assumption hold true, that is, the robot does not have to decide what are the places or even the extent of these different places. The robot indeed needs to recognize its current location based on semantic categories which are defined by the human. A detailed description of this novel approach will be drawn in Chapter 4 after having a thorough review of the relevant literature.

Several recent methods have been used to address the problem of robot localization based on semantic information (for instance see [Dubois et al., 2011; Guillaume et al., 2011; Torralba et al., 2003a; Ullah et al., 2008; Wu et al., 2009]). These approaches and others are illustrated in the next sections.

2.2 Current approaches for semantic place recognition

Although most of the proposed approaches to the problem of robot localization have given rise to SLAM techniques, some other approaches have addressed this problem as a SPR task. This task usually requires first to produce an appropriate code for the initial data and then use this code to learn the robot places. In the following sections we make a survey of the different coding and learning approaches that can be used in the context of SPR.

2.2.1 Coding methods

Most of the researchers have shown that before preforming a SPR task, it is necessary to create an appropriate code of the initial data [Oliva and Torralba, 2006; Torralba et al., 2003b; Ullah et al., 2008; Wu et al., 2009]. To achieve that, different coding methods have already been proposed within the framework of object-based and view-based methods. We introduce both approaches in order to understand the differences between them and we also describe the state-of-the-art of these approaches in the context of place recognition.

2.2.1.1 Object-based semantic place recognition methods

Object-based place recognition can be used for mobile robotics, to determine the robot place. Traditionally, this kind of approach is first based on learning and detecting a set of interesting objects in the images and then use them to determine the robot location.

In other words, object-based methods seek to learn the objects in the scene to perceive and understand the environment of the robot [Chang et al., 2008, 2009].

Recently, visual object recognition has been widely investigated, with the development of different methods. Most of these approaches have first focused on extracting the local image features at a variety of positions and scales, and then comparing the extracted features of an object with a set of well-known objects. This allows classifying each image patch independently from the others (see [Murase and Nayar, 1995; Papageorgiou and Poggio, 2000; Schneiderman and Kanade, 2000; Viola and Jones, 2002]).

Several other approaches based on image segmentation have investigated both problems of object and place recognition. In these approaches, the authors first segment the image into objects, using a segmentation algorithm, and then use them to recognize the robot location. Among of these approaches, some authors [Chang et al., 2008] propose to use an image segmentation algorithm, called "jigsaw puzzle", to segment the input scene image into regions that may correspond to objects or parts of objects. Based on these image regions, they detect a set of salient objects to represent a place and the SIFT descriptors contained in these salient objects are kept in the database. A similar approach, in [Chang et al., 2009], proposes to use a more sophisticated segmentation algorithm. This algorithm first segments the salient objects appearing in the scene using "Gestalt laws" to detect the boundaries of the major object classes like vehicles, buildings, pedestrians, *etc.* [1]. They then represent each prominent object with a list of SIFT descriptors. Both approaches recognize the place by recalling if they have seen some of the salient objects appearing in the scene before.

However, although it is possible to use object-based place recognition approaches, Fergus stated that successful approaches to object recognition must address a variety of problems [Fergus, 2005]:

- Changes of aspect. Different views of an object can be very different, as shown in figure 2.4 (a).

[1] Gestalt principles, or gestalt laws, are rules of the organization of perceptual scenes. They aim to formulate the regularities according to which the perceptual input is organized into unitary forms, also referred to as (sub)wholes, groups, groupings. These principles mainly apply to vision.

- Changes of viewpoint. Objects can also be subject to in-plane transformations (translation, rotation, scaling, skews) and out-of-plane transformations (foreshortenings) that change their appearance. However, some viewpoints may be more likely than others (*i.e.* motorbikes are rarely vertically oriented) and this prior knowledge may be exploited.

- Illumination differences. A change in the lighting of the object will change the pixel values in the image. The change could be a shift or scaling of the pixel values or, if the light source changes position, a non-linear transformation, complicated by shadows falling on the object. The images in figure 2.4 (b) illustrate examples of drastic changes in lighting.

- Background clutter. In the majority of images it is rare for the object to be cleanly segmented from the background. More typically, the background of the image contains many other objects (other than the one of interest), which distract from the object itself. The images in figures 2.4 (a and b) have cluttered backgrounds.

- Occlusion. Some parts of the object may be obscured by another object, as illustrated by the monkeys in figure 2.4 (c). Additionally, as the aspect changes, one part of the object may hide another. This is known as self-occlusion.

- Intra-class variation. As in the car example of figure 2.5, the category itself can have a large degree of visual variability. The variability can take various forms: in the geometry, appearance, texture and so on. Also, one instance of an object may have features which are missing on another (*e.g.* the radiator grille on the cars of figure 2.5).

Globally, the methods based on objects require complex models and heavy learning procedures. Furthermore, the use of object recognition in classification is not trivial. Therefore, recognizing the objects before recognizing the place itself seems to be a difficult task. In contrast, recognizing the room would provide strong priors, simplifying the process of object recognition, which is not possible in object-based methods but possible in view-based methods.

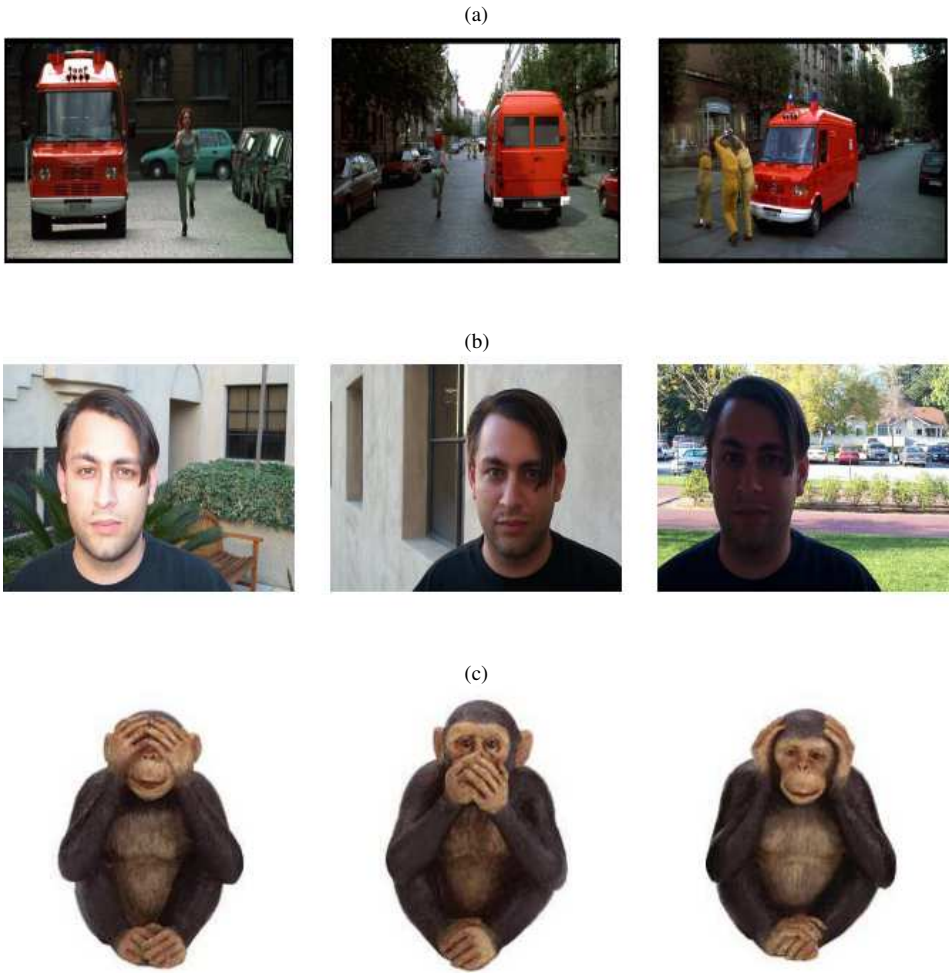

Figure 2.4: (a) Variation due to changes in aspect, (b) Variation in appearance due to a change in illumination, and (c) Some examples of occlusion [Fergus, 2005].

2.2.1.2 View-based semantic place recognition methods

View-based methods are used to predict the robot place from images of the scene. They seem to be a more powerful technique than object-based methods for both place and object recognition problems [Torralba et al., 2003b]. Vision-based place recognition have two major advantages:

- Firstly, vision can guide the action selection by the system. It can determine what are the most significant actions (features) in the place so that have to be extracted.

  Generally, the best methods reach a high level classification rate, sometimes exceeding 90%. However, most of these methods do not test the classifica-

Figure 2.5: Some examples from a visual category [Fergus, 2005].

tion robustness with respect to the illumination conditions. Some others do so, for instance, the authors in [Pronobis and Caputo, 2007] proposed to use the IDOL (Image Database for rObot Localization) database which allows this kind of tests. More information about this database can be found in [Luo et al., 2006]. The recognition rate reaches 95% when the illumination conditions are the same for learning and recognition, but drop to about 75% when it is not the case. The authors in [Ullah et al., 2008] also proposed to use the COLD database to test more precisely the robustness to dynamic changes and environmental variabilities. They proposed to use the following protocol: training and testing were always done on different sequences acquired in the same laboratory. They trained on one illumination condition, and tested on sequences acquired under various illumination conditions, and after some time. With these experiments they were able to address at the same time the robustness with respect to dynamic and geographical changes.

- Secondly, several works (like [Torralba, 2003; Torralba et al., 2003b]) have shown that recognizing a place would facilitate the recognition of its local objects. For instance, the recognition of an object like a coffee machine would be easier if we first know that the robot is located in the cafeteria and own the detailed local properties of this room, like its materials components, its typical shape, *ect*. This is called contextual priming: being in a place (*e.g.* a kitchen), you can expect to find specific objects like pan, but not others like television.

Besides, semantic cues (color, shape, and texture) in the retinal image produced by an object provide enough information to unambiguously determine the object category [Torralba et al., 2003a].

The place usually appears as a two-dimensional image, and each image is represented by a raster-scan of pixels, *i.e.* a vector of intensity values. More formally, view-based systems exploit visual context (contextual information) to have a low-dimensional representation of the image (for instance the Generalized Search Tree "GiST" of the scene) [Oliva and Torralba, 2001]. Such a representation can be simply computed without the necessity to identify specific regions or objects within the scene. Having identified the overall type of the scene, one can then proceed to identify specific objects within the scene.

Recently, significant works have been developed for place recognition based on visual descriptors. These descriptors are used to extract signatures, by extracting the regions of interest (ROI), from the images. These signatures are used recognize the robot place. They consist in a constellation of descriptors, computed over different kinds of local or global covariant regions [Ramisa et al., 2009]. In other words, these signatures are a vocabulary computed either using global descriptors (Generalized Search Tree (GiST) [Oliva and Torralba, 2001] and CENsus TRansform hISTogram (CENTRIST) [Wu et al., 2009]) or using local descriptors (Speeded Up Robust Features (SURF) [Bay et al., 2006], and Scale Invariant Feature Transform (SIFT) [Lowe, 1999, 2004]).

Local descriptors are used to extract the most interesting pixels in the image in order to compute its signature. In particular, SIFT descriptors are used to detect and describe local features in images, like elements of scenery, objects, people, *etc.* [Lowe, 1999, 2004]. They are digital information derived from local analysis of an image and they characterize the visual content of the image as independently as possible from the rotation, translation, and scale invariance. SURF is another robust image local detector and descriptor, first introduced by [Bay et al., 2006]. It is partly inspired by the SIFT descriptors, where the extracted features based on the sum of two-dimensional Haar wavelet responses with the aid of integral images to reduce the computation load.

Global descriptors have also been used to detect the features (for instance see [Torralba et al., 2003a; Wu et al., 2009]). These descriptors use the whole image pixels

to compute the signature regardless of "interesting points". Similarly to local descriptors, global descriptors capture the visual context of the image. In particular, GiST descriptor was introduced in the context of scene recognition by [Oliva and Torralba, 2001]. It is a vector of principal components of outputs of a Gabor-like filter bank applied to the image. This descriptor describes the spatial layout by capturing features such as naturalness, openness, expansion, depth, roughness, complexity, ruggedness and symmetry [Coelho and Ribeiro, 2010]. CENTRIST is another global descriptor presented by [Wu et al., 2009]. It aims at capturing the local intensity pattern in the image, based on the Census Transform (CT) of the edges [Zabih and Woodfill, 1994] [1]. Note that the CT is equivalent to the local binary pattern code ($LBP_{8,1}$) [Ojala et al., 2002]. It has been shown that this descriptor is robust to illumination changes and other minor variations (*e.g.* moving persons, moved objects in an image, *etc.*) because the transformation in the CT is robust to these changes [Wu and Rehg, 2011]. This point is interesting, because the robot can recognize its current location under different illumination conditions, *i.e.* the place recognition can be achieved insensitive to the lighting conditions.

Several vision-based place recognition approaches have already been proposed based on these coding methods. For instance, the authors in [Oliva and Torralba, 2006; Torralba et al., 2003b] used the GiST descriptor, while the authors in [Andreasson et al., 2005; Se et al., 2001; Ullah et al., 2008] used the SIFT descriptor, and finally the authors in [Wu and Rehg, 2011] used the CENTRIST one. These representations usually give signatures that are continuous vectors which are not suitable to compute probabilities and they are not constant in size (*i.e.* the number of interest points varies from one image to another). To reduce the size of these representations, most of the authors use Bag-of-Words (BoWs) approaches, which consider only a small set of interest points in the image [Filliat, 2008; Gokalp and Aksoy, 2007; Lazebnik et al., 2006; Wu et al., 2009]. This step is usually followed by a vector quantization process, such that the image can be represented as an histogram.

Several other approaches have been suggested to extract color histograms using panoramic images [Blaer and Allen, 2002; Ulrich and Nourbakhsh, 2000], or extract

[1] CT compares the intensity value of a pixel with its eight neighboring pixels, if the intensities of the neighboring pixels are equal or less than the intensity of the center pixel, then a bit is set to 1 at the corresponding location, otherwise it is set to 0.

the Fourier coefficients of low frequency image components [Menegatti et al., 2004], or use eigen-space representation of images [Gaspar et al., 2000] to recognize the different places. Some other works [Pronobis, 2005; Pronobis et al., 2006] use composed receptive field histograms which were introduced by [Linde and Lindeberg, 2004]. These histograms have shown to be able to cope with small illumination and pose variations.

Generally, it is difficult to conclude whether global or local descriptors are more beneficial. However, global encodings seem to have good performance in terms of classification.

After image coding, the next step is to perform the recognition process itself (learning the robot places) using classification methods. In the next section we will introduce the existing classification methods that can be used to recognize the robot place.

2.3 Classification methods

Before introducing the main concepts of the existing classification methods and their use in the context of SPR, it is important to note that we are interested in the recognition of instances, as presented in [Ullah et al., 2008], and not in the recognition of concepts "categorization" which has also been investigated in [Ullah et al., 2008]. In categorization, observers make decisions about whether distinct objects belong to the same class or not (*i.e.* they measure the similarity between objects or between groups of objects) (*e.g.* discriminate the offices in different buildings). On the contrary in recognition or classification, observers judge whether each test object exactly matches a study object or not [Nosofsky et al., 2011] (*e.g.* distinguish between office and corridor). Many terms related to the word "recognition" are used in a somewhat loose manner in the literature. So, to avoid confusion between these terms, here are some definitions for them:

- Category (class): set of entities grouped together under one or more common characteristics. For example, the books are categorized into beginner and advanced.

- Instance recognition (identification): the process by which an entity is identified in an image, with respect to the objects, the viewing angle, brightness, *etc*.

Concerning the scene, the task is to recognize the same scene from a different angle.

- Categorization (category recognition): the process of deciding what class the entity belongs to, out of many possible classes. In the literature, this term is sometimes used in the sense of classification. However, in this work we assume that the classification process which associates each object to a category is different from the categorization process which creates a category in order to associate an object to it.

- Classification: the process by which an object is recognized as belonging to a class or category. For instance, the books in the library are classified according to the subject.

- Localization (detection): the process of specifying the location within the image of all instances of an object.

- Recognition: this term is used generically to refer to the problem as a whole (*i.e.* any of the set of classification, identification or localization.

Therefore, in this work we use the terms "recognition" and "classification" to speak about the process of "instance recognition" or "identification". Instance recognition usually involves linking each new instance to a particular class. Of course, it could be possible to use rules characterizing a room. However, they are difficult to find for the human. This is why it seems to be more appropriate to use numerical methods to solve this machine learning (ML) problem.

The goals of machine learning are first to develop algorithms that could learn (i.e. re-cognize patterns) from an initial set of known data and then make accurate predictions for previously unseen data. Methods derived from machine learning approaches have been applied to various questions like autonomous car driving, optical character recognition, face detection, and speech recognition, *etc.* [Abdel-Rahman et al., 2011; Hinton et al., 2006; Sarikaya et al., 2011].

ML is a branch of Artificial Intelligence (AI) and it focuses on the statistical nature of learning. Arthur Samuel has defined ML as a field of study that gives computers the ability to learn without being explicitly programmed [Samuel, 1959]. Thomas Mitchell

has also defined ML saying: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [Mitchell, 1997]. The main goal of ML is to develop algorithms which learn directly from the empirical data by exploiting the statistical relationships present in the image. These algorithms can be organized into supervised, unsupervised, semi-supervised, or reinforcement learning based on the desired outcome of the algorithm. We provide here some definitions for these algorithms, in order to understand the differences between them.

- Supervised learning algorithms generate a function that labels the inputs to desired output, where the labeling "right answers" are provided by human experts. The learning in this case is thus performed with the presence of an "expert", teacher, or knowledge of output. Some examples of supervised learning algorithms are: Neural Network [Rumelhart and McClelland, 1986], Support Vector Machines [Vapnik, 1998], Decision Trees [Quinlan, 1986], Bayesian Classifiers [Pazzani and Domingos, 1997], *etc*.

- Unsupervised learning algorithms model a set of inputs by themselves, so that the learning procedure does not include any knowledge about output class or value (data is unlabeled or value is unknown). Some self-guided learning algorithms are: k-Nearest Neighbor algorithm (k-NN) [Beyer et al., 1999], genetic algorithms [Deb et al., 1999], clustering approaches [Steinbach et al., 2000], *etc*.

- Semi-supervised learning algorithms combine both labeled and unlabeled examples for training to generate an appropriate function. Typically, semi-supervised learning techniques learn from a combination of a limited set of labeled data with a large amount of unlabeled data which can be inexpensive to generate. For instance, semi-supervised and transductive SVM [Bennett and Demiriz, 1998; Joachims, 1999] and co-training [Blum and Mitchel, 1998] are two examples of semi-supervised learning algorithms.

- Reinforcement learning algorithms study how artificial systems and animals can learn to improve and optimize their actions in a complex environment. The most reinforcement learning algorithms are: Monte Carlo Methods (MCMs)

[Doucet, 1998], Temporal Difference Methods (TDMs) like Q-learning [Christopher, 1989], and Direct Policy Search (DPS) [Peters et al., 2003].

The problem of semantic place recognition can therefore be solved using machine learning methods (for instance see the discriminative approaches based on SVMs in [Pronobis and Caputo, 2007; Ullah et al., 2008] or see the generative approaches based on Bayesian filtering techniques in [Torralba et al., 2003b; Wu and Rehg, 2011]). In addition to that, contrarily to the classical localization and mapping works, this work addresses the problem of robot localization as a supervised machine learning problem.

To introduce the notations of supervised discriminative approaches, let us consider an example of supervised learning which is represented by a pair (y,c), where $y \in Y$ is the representation of the place in a space Y of dimension t and $c \in C$ is the class membership of y from a finite set of C classes. A training set $S = \{(y_1,c_1),\ldots,(y_N,c_N)\}$ consisting in N examples can then be generated. The idea is to discriminate between N different classes in the representation data space Y, *i.e.* find and learn the "decision frontier" that will assign each class to each point of the data space. To achieve that, generative or discriminative machine learning methods can be used. Discriminative approaches aim at directly finding the best way to separate the classes (they directly search for the decision frontier), while the generative ones aim at first finding the optimal model that explains and fits the original data and then, using the generated model, find the frontier between the data. In the next sections, we will explain in more detail how to use both approaches in the context of semantic place recognition.

2.3.1 Generative approaches

Generative approaches can be used to compute the likelihood of an observation given a certain place within the framework of Bayesian filtering. These approaches are used in machine learning for data modeling using the probability density function (PDF) as shown in figure 2.7 (left) and through the use of Bayes' rule. They describe the data using structured probabilistic models.

The problem of semantic place recognition can then be expressed as a conditional probability problem using Bayes theorem as follows: $P(x_t = c|y^t)$. More precisely, we have a set of observations, $y^t : y_1,\ldots,y_t$, from which we need to deduce the place or class "c" the robot is in.

2.3.1.1 Naive Bayes classifiers

Naive Bayes Classifiers (NBCs) are probabilistic methods based on Bayes theorem, under the assumption of the independency between the model features. NBCs assume that the features of one class are statistically independent from the features of other classes. This can be explained mathematically using Bayes theorem as follows:

$$P(x_t = c|y^t) = \frac{P(x_t = c)P(y^t|x_t = c)}{P(y^t)} \tag{2.1}$$

where

- $P(x_t)$ represents the prior probabilities assigned to each class independently from the observations.
- $P(y^t|x_t)$ is the probability density of an observation y^t given a certain place x_t. It represents the probability of observing different data y^t for each class belongs to x_t. Note that y^t is the observation sequence: $y_{1:t}$.
- $P(y^t)$ is the marginal density (or the probability density) of the data y^t, *i.e.* it represents the probability of observing the different examples. However, this denominator has no influence on the classification process. Because it does not depend on c and the values of the features y^t are given, so that it is effectively constant and it plays a normalization role so that $\sum_i P(c_i|y^t) = 1$.
- $P(x_t = c|y^t)$ is the posterior probability, *i.e.* the conditional probability of x_t given y^t. In other words, the probability to be in a given place x_t according to a given observation y^t.

Bayes rule can therefore express the posterior probability $P(x_t = c|y^t)$ in terms of prior probability $P(x_t)$ and conditional probability density $P(y^t|x_t)$. Once the model is learned, the distribution $P(X|Y)$ models the phenomena of the original training data and allows generating new samples. That's why learning in this model is called generative.

The most important characteristic of NBC is that the data attributes are assumed to be independent, $\big(i.e.\ P(y^t) = \prod_{i=1}^{k} P(y_i)\big)$, and thus this term has no influence on equation 2.1 as earlier said. However, the question is how to compute the likelihood

of an observation $\big($the probability density term $P(y^t|x_t)\big)$ using Bayes theorem. There are a couple of solutions to this problem: one is to use the Maximum Likelihood (ML) technique; another one is to use *Maximum A Posterior (MAP)* technique, which learns the data maximizing the likelihood as follows:

$$\begin{aligned} \mathrm{x}_{MAP} &= \arg\max_{t\in T} P(x_t = c|y^t) \\ &= \arg\max_{t\in T} \frac{P(x_t)P(y^t|x_t = c)}{P(y^t)} \\ &= \arg\max_{t\in T} P(x_t)P(y^t|x_t = c) \end{aligned} \tag{2.2}$$

where T is a set of observations. Once again, we dropped the factor $p(y^t)$ because the probability of the data is constant, based on the fact that independent features are assumed in NBC. Besides, this gives the rule of NBC and equation 2.2 can be re-written as follows:

$$\mathrm{x}_{MAP} = \arg\max_{t\in T} P(x_t) \prod_i P(y_i|x_t = c) \tag{2.3}$$

Hence, this equation computes the individual measurements which are independent given the robot position. Finally, if we assume that all the classes are equally probable, the previous equation can be re-written as follows:

$$\mathrm{x}_{MAP} = \arg\max_{t\in T} P(y|x_t = c) \tag{2.4}$$

NBC therefore provides a decision theory for data classification.

Among those approaches, recent researches have been published in [Dubois et al., 2011]. In this work, the authors propose to use NBCs and temporal integration that combines successive observations. This model obtained interesting classification results on the COLD database. The overall performance is very close to the state-of-the-art results [Guillaume et al., 2011; Ullah et al., 2008].

2.3.1.2 Bayesian filtering techniques

Another probabilistic generative approach based on Bayesian filtering techniques has been developed to achieve visual place recognition [Torralba et al., 2003b; Wu et al.,

2009]. These techniques provide a powerful statistical tool to filter the classification results. More formally, Bayes filters address the problem of recognizing the place, c, the robot is in, using sensory information. Given a stream of observations y^t and controls u^t, which describe the dynamics of the system, Bayes filter recursively computes the posterior distribution according to the following equation:

$$Bel(x_t) = P(x_t = c|u_1, y_2, \ldots, u_{t-1}, y_t) \tag{2.5}$$

However, Bayes filters make the assumption that the dynamic system is modeled as Markov Chain, *i.e.* as shown in figure 2.6, the observation y_t and the control measurement u_t are conditionally independent from the previous measurements given the state x_t. More precisely, this probabilistic approach is based on two assumptions:

1. The Markov assumption of the state evolution says that the knowledge of the state at time t depends only on the previous state at time $t-1$

$$P(x_t|x_{0:t-1}, y_{1:t-1}, u_{1:t}) = P(x_t|x_{t-1}, u_t) \tag{2.6}$$

   This probability distribution is called the transition model and represents the changing state of the world based on the actions of the robot. It also depends on the dynamics of the system.

2. The assumption of completeness says that the current state completely explains the current observation: $P(y_t|x_{0:t}, y_{1:t-1}, u_{1:t}) = P(y_t|x_t)$. Moreover, this probability distribution is often independent of t: $P(y_t|x_t) = P(y|x)$. This is known as sensors model although the model response depends on the state of the world.

Based on these assumptions, the posterior distribution of equation 2.5 can be recomputed efficiently using the following update rule:

$$Bel(x_t) = P(x_t = c|u_1, y_2, \ldots, u_{t-1}, y_t) \sim P(y_t|x_t = c) \sum_{t-1} P(x_t|x_{t-1}) P(x_{t-1}|y^{t-1}) \tag{2.7}$$

However, there are still two problems in this update rule. The first one is how to compute or model the likelihood part, $P(y_t|x_t = c)$, *i.e.* how to compute the conditional probability of an observation according to the considered class. Two possible ways

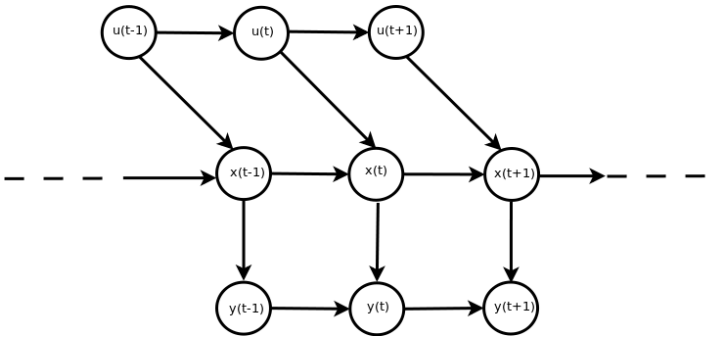

Figure 2.6: Markov assumption: $P(y_t|x_{0:t}, y_{1:t-1}, u_{1:t}) = P(y_t|x_t)$ and $P(x_t|x_{0:t-1}, y_{1:t-1}, u_{1:t}) = P(x_t|x_{t-1}, u_t)$.

have been proposed to solve this problem: the first one, proposed in [Torralba et al., 2003b], omits the quantization step and models the likelihood as a Gaussian Mixture Model (GMM); another possible solution is proposed in [Wu and Rehg, 2011], where the authors discretized the likelihood using BoW methods.

The second problem is how to compute the place transition distribution, $P(x_t|x_{t-1})$. As illustrated in [Wu et al., 2009], this distribution can be defined using a transition matrix as: $P(x_t|x_{t-1}) = p_e$ if x_t equals x_{t-1}, where the value of p_e can be in the interval from 0.9 to 0.99. The rest of the probability mass is shared uniformly among all other transitions. A frame t is then classified as the category which index is $arg\ max\ P(x_t|y^t)$ in the Bayesian filtering framework.

Markov-Chain Monte Carlo (MCMC) approaches can also be used to determine the robot location given a map of its environment and using Markov localization. Although these methods have been discovered since 1960's by [Handschin, 1970; Handschin and Mayne, 1969], they became increasingly popular in robotics over the last few years. This is due to the fact that they need powerful computer machines. A detailed discussion around Monte Carlo methods can be found in [Doucet, 1998].

MCMC methods were therefore used in the field of mobile robot localization. For example, in [Dellaert et al., 1999], the authors proposed a Monte Carlo Localization (MCL) method to determine the robot location. This method uses uncertainty representation, *i.e.* it represents the PDF by maintaining a set of samples that are randomly drawn from it. This step is followed by the use of Monte Carlo methods to update this density representation over time. Another probabilistic localization algorithm is proposed in [Thrun et al., 2000], where the authors develop an algorithm called

Mixture-MCL, which integrates two complementary ways to generate samples in the estimation. To apply this algorithm for mobile robots localization, a kernel density tree is learned that permits fast sampling. Another work has been developed in [Torralba et al., 2003a], where the authors use Hidden Markov Model (HMM) to represent each robot place as a hidden state of the HMM and the feature vector stands for the observation. The recognition is therefore achieved using standard Bayesian techniques.

Furthermore, another approach proposed in [Torralba et al., 2003b] which is the first works that addressed the problem of semantic place recognition and categorization. This approach uses the global descriptor GIST together with a temporal integration. The temporal integration is based on a HMM to mix the information over time and space. This system was tested in recognition and categorization of instances, on indoor and outdoor places. However, the estimation of the transition between places is not straightforward. Besides, the overall algorithm needs high computational cost to perform the whole classification process. The authors in [Wu et al., 2009] also proposed to use another global descriptor "CENTRIST" for feature extraction. This step is followed by the use of K-means clustering algorithm, NBC and BoW methods to perform the classification process. The authors also used the same Bayesian filtering to improve the results. Besides, the overall algorithm is very complex.

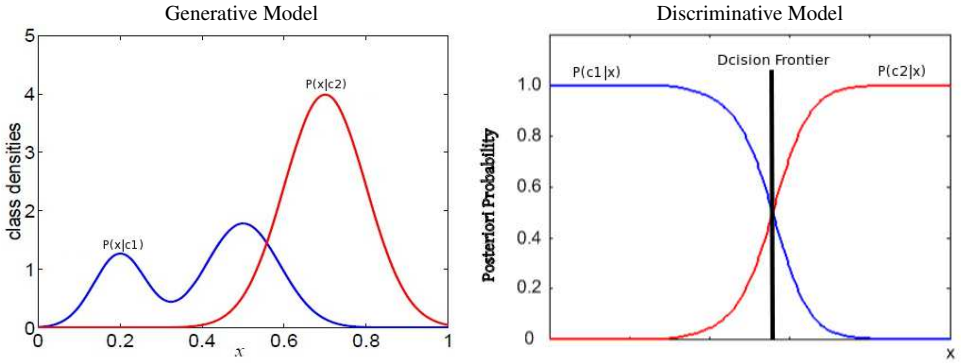

Figure 2.7: Discriminative *versus generative models. Discriminative models try to directly perform the classification, which can be expressed as* $P(c|X)$*. Generative approaches first learn the conditional probability* $P(X|c)$ *of each class of the data representation space and then compute the likelihood,* $P(c|X)$*, using the Bayesian theory.*

If the observed data are truly sampled from the generative model, then fitting the generative model parameters to maximize the data likelihood is a common method.

However, since most statistical models are only approximations to the true distribution, then it can be argued that the approximation makes more assumptions than are necessary to solve the problem by hand. In such cases, it can be more accurate to model the conditional density functions directly using a discriminative model, although each application-specific details will ultimately dictate which approach is most suitable.

2.3.2 Discriminative approaches

Although it was possible to use generative approaches using Bayes rules to compute the posterior probability $P(x_t = c|y^t)$, there is a direct way to estimate this probability using discriminative approaches. Discriminative approaches can be used to compute the probability to be in a given place, c, according to the current observation, y_t, as $P(x_t = c|y_t)$. More precisely, these approaches directly learn the decision frontier in the data space representation as shown in figure 2.7 (right). The decision frontier in the binary case, for instance, means that some examples belong to class 0 and other examples belong to class 1. The principle of classification in these methods is based on learning the separating surfaces between the data. This learning is particularly based on modeling the relationships between input and output data. These relationships can be obtained using a minimum number of assumptions about the structure of input data to minimize the cost function of the classification.

Most of the discriminative approaches have focused on NNs [Rumelhart and McClelland, 1986] and SVMs models [Vapnik, 1998]. These two approaches are based on a scalar product. More formally, NNs are based on the scalar product in the data representation space, x_i, *i.e.* a measure of the projection of one vector onto another. To illustrate that, let's consider the example shown in figure 2.8:

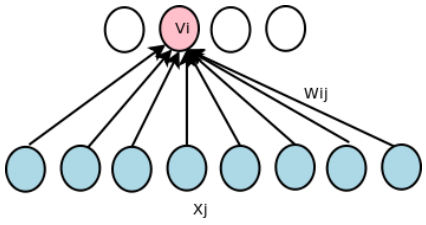

Figure 2.8: Scalar product example of neural networks.

The unit v_i has an activation function, a_i, given by the following equation:

$$a_i = \sum_{j=1}^{n} w_{ij} x_j =< w, x > \tag{2.8}$$

where this activation function can be represented as a scalar product of the input vector, x_j, and the weights, w_{ij}. Besides, this function defines the decision frontier to separate the data into different classes. Similarly, SVMs provide an efficient way to map the data into a high-dimensional feature space, *i.e.* $x \rightarrow \varphi(x)$, using the following kernel function:

$$K(w,x) =< \varphi(w)\varphi(x) > \tag{2.9}$$

NNs and SVMs approaches can therefore be used to predict the current robot place. In order to illustrate these models in a more precise way, let's explain each of them separately.

2.3.2.1 Neural networks classifiers

Conventional feed-forward networks are non-recurrent neural networks in which the connections between the layers are not directed cycles. As shown in figure 2.9, the data are feed-forward from the input layer, through the hidden layer(s) and then to the output layer. This kind of network is called Multi-Layer Perceptrons (MLPs). Usually, the training process of MLPs for pattern classification problems includes two different tasks, the first one is to choose an appropriate network for the problem, and the second is the adjustment of the network connection weights. If the problem is linearly separable, then it is easy to find the appropriate decision frontier. Otherwise, this task seems to be a big challenge since we don't know how many hidden units or even the number of hidden layers are required to compute the optimal decision frontier. Finding the appropriate network indeed varies from one problem to another and can be achieved by preliminary trials specific of the problem at hand. For the learning of weights, Rumelhart *et al.* [Rumelhart et al., 1986] introduced back-propagation as an alternative learning algorithm to Boltzmann Machine for multi-layer neural networks training. Back-propagation, or propagation of error, is a supervised learning mech-

anism to teach neural networks to perform a given task for networks which have no feedback.

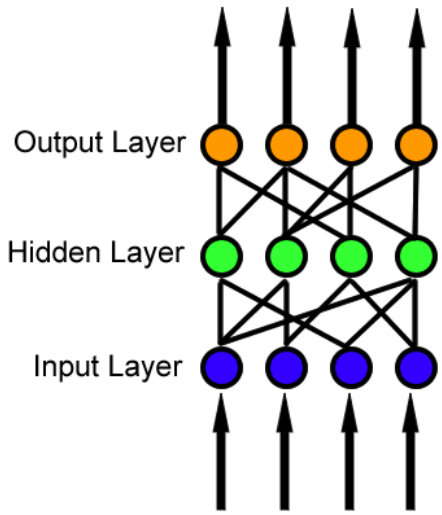

Figure 2.9: Partly connected feed-forward neural network with one hidden layer and one output layer. Energies can be contributed by output variables in both hidden and output layers, where the number of output variables need not correspond to the number of input variables.

The outputs of MLPs, which are based on the weights w, are usually used to perform the classification process, *i.e.* we use the extracted weights features to separate the different classes within the framework of Bayesian theorem, $P(x_t = c|y_t;w)$. However, these output units do not sum to 1, *i.e.* $\sum_i P(c_i|y_t;w) \neq 1$, and thus cannot be used directly to perform the classification process. To overcome this problem, one way is to use a softmax regression to transform these units into real probability values so that the summation of these values will be equal to 1 for a given observation y_t. This transformation can then be achieved using the following softmax formula:

$$P(x_t = c_i|y_t;w) = \frac{e^{w_i^T y_t}}{\sum_{l=1}^{k} e^{w_l^T y_t}} \tag{2.10}$$

Note that a detailed description to softmax regression can be found in section 2.3.2.3.

NNs models can therefore be used to find the decision frontier which separates the data into different classes. However, Vapnik [Vapnik, 1995] has demonstrated that the

decision frontier in NNs cannot be assumed to be the optimal one. This result has lead to the development of a new classification method, the Support Vector Machines.

2.3.2.2 Support vector machines approaches

Support Vector Machine (SVM) is a discriminative model able to construct a hyperplane or a set of hyperplanes in a high-dimensional space and can be used for further tasks such as classification. SVMs methods were first proposed by [Vapnik, 1998] to find the decision rules based on the feature space to discriminate the different classes. Contrarily to NNs, SVMs do not need to choose the number of hidden units or layers, but their feature space depends on the use of the different kernels (polynomial, linear, sigmoid, *etc.*).

SVM is a supervised statistical learning algorithm [1]. It can be used to train the data and predict the patterns of the data to create a "decision-maker". Figure 2.10 illustrates the high-level view of how we perform these tasks using an SVM approach, where the input corresponds to the dataset and the output results correspond to the classification of the data which is used for testing. More precisely, using this algorithm we perform two different tasks:

- **Learning**: by training the input examples (data) using SVM-train function.
- **Prediction**: new examples are used for testing. Usually, if the problem task is to classify the observations in a set of finite labels, the task is said to be a classification task.

It has been shown that this approach has a good performance on character recognition, text classification (see for instance [Joachims, 1999; Leopold and Kindermann, 2002]). Promising results are also reported for place recognition in [Pronobis et al.] using this approach. Moreover, SVMs approach has a major advantage over many other techniques, like Artificial Neural Networks (ANNs): it's solution is global and unique, and thus avoids local minima, (for further details, see for instance [Bishop, 1995]). Another important characteristic in this approach is the ability to use different kernels,

[1] Sometimes, the authors use SVC or SVM-C to show that the SVM approach is used for a classification purpose.

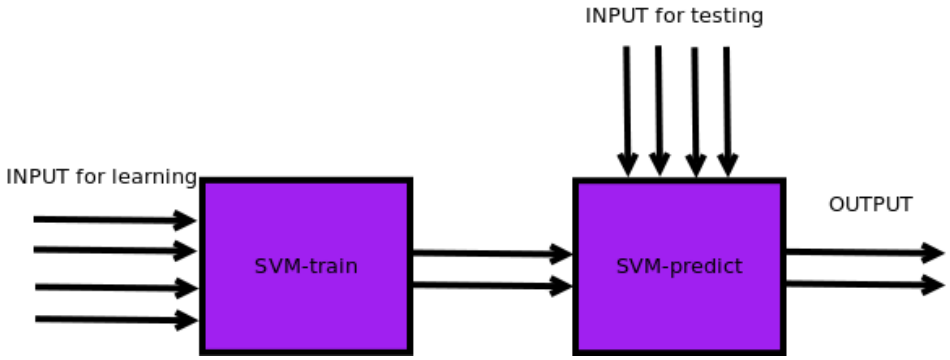

Figure 2.10: High-level conceptual point of view of an SVM approach.

of different degrees, such as linear, polynomial, radial, and sigmoid. It means that it is possible to identify the appropriate kernel for a particular classification problem.

Data classification is a common task in machine learning. If the data is linearly separable, then a linear SVM would be sufficient to perform the classification process. Otherwise, a nonlinear SVM is required. The only difference between the nonlinear SVM and their linear counterparts is the use of kernel function instead of the inner product. In the next sections, we introduce both linear and nonlinear SVMs.

Linear support vector machines

Linear SVM classifier is used to learn linear separators in a high dimensional space with a maximum margin as shown in figure 2.11. In particular, in the case of a linear classification, a data can be viewed as a n-dimensional vector, and the idea behind is to know whether we can separate such points with a $(n-1)$-dimensional hyperplane. Usually, there are many hyperplanes that could separate the data. However, it is possible to find the optimal hyperplane which represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. This is called *maximum-margin hyperplane (MMH)*.

To better understand how linear SVMs classify the data, let us consider the following example shown in figure 2.12 (a). In this example there are two different classes of positive and negative nodes. So, the question is how could we separate (*i.e.* classify) these data examples into two different regions.

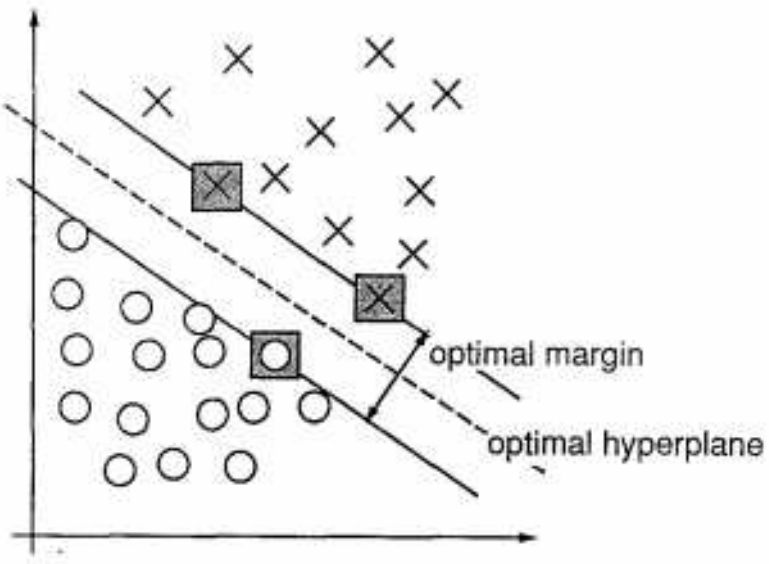

Figure 2.11: An example of a separable problem in a 2 dimensional space. The support vectors, marked with gray squares, define the margin of largest separation between the two classes [Cortes and Vapnik, 1995].

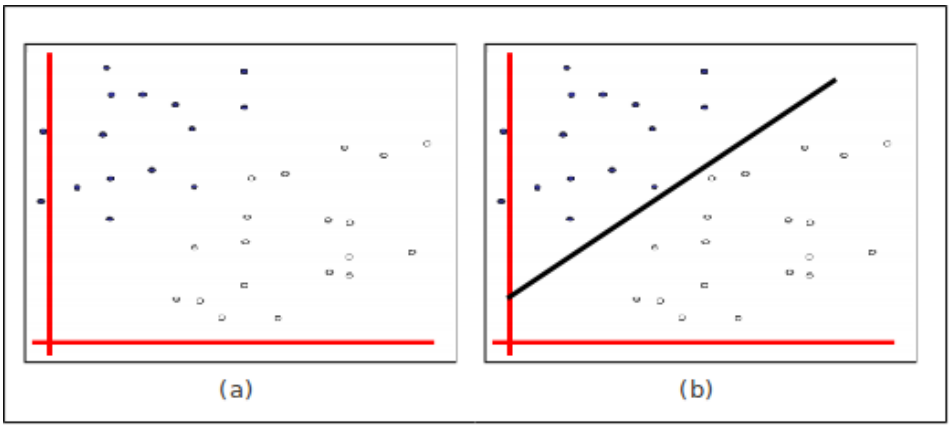

Figure 2.12: (a) Simple example of two different sets of nodes which are linearly separable. (b) The results obtained using a linear SVM classifier, the black line perfectly separates positive and negative nodes.

It is obvious that the two different nodes shown in this figure are linearly separable. It is possible therefore to use a linear SVM to separate the two different categories, as shown in figure 2.12 (b). However, finding the optimal separation between these data examples is a challenging task. Therefore, we need to find the separator margin which

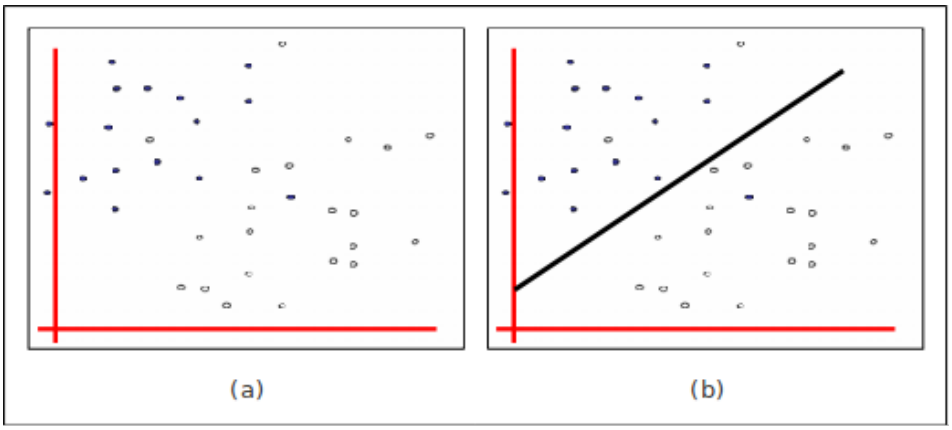

Figure 2.13: (a) Another example of two different sets of nodes which are not linearly separable. (b) The results obtained using a linear SVM, the black line does not perfectly separate positive and negative nodes.

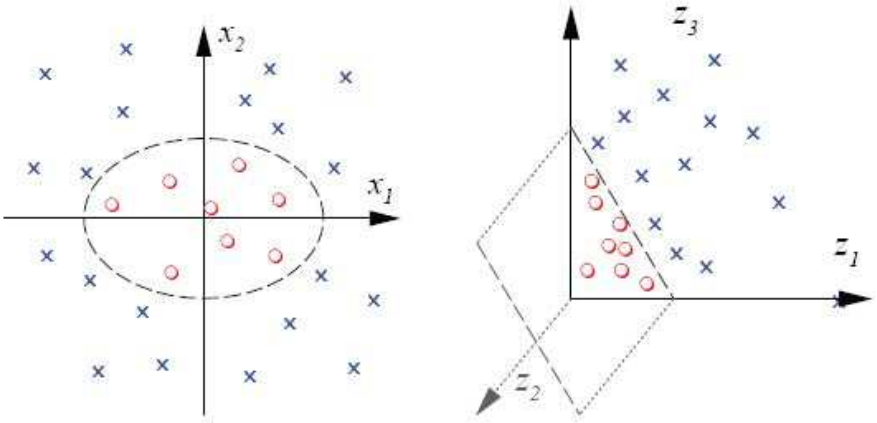

Figure 2.14: A nonlinear SVM process presentation. It shows that after the data is transformed from two-dimensional space to three-dimensional space, the data becomes linearly separable.

is determined by just a few examples called "*support vectors*" [1]. In other words, as shown in figure 2.11 this separator can be defined in terms of support vectors s_i and classifier examples x as follows:

$$f(\vec{x}) \leftarrow (\sum_{s_i} w_i \vec{s_i} \cdot \vec{x} + b) \tag{2.11}$$

[1] Support vectors are a subset of training instances that define the decision boundary between classes as shown in figure 2.11.

Non-Linear Support Vector Machines

In the former section, we described how to find the optimal separating hyperplane in the linearly separable case using a linear SVM. However, if the data is not linearly separable as shown in figures 2.13 (a) and 2.14 (left), then linear SVMs are not sufficient to classify the data into different regions. To illustrate that, let us consider the *Exclusive OR (XOR)* classification problem, it has 4 different samples, $\{(0,0),(0,1),(1,0),(1,1)\}$, located on the corners of a rectangle. Finding a decision frontier that separates these samples in a two-dimensional space seems to be impossible since the data is not linearly separable. However, if we add a new dimension, the problem becomes linearly separable and the samples can thus be separated by a hyperplane in the three dimensions space. This indicates that it is possible to transform non-linearly separable problems into linearly separable ones by projecting the data into a higher dimensional space, as illustrated in figure 2.14.

SVMs provide an easy and efficient way of doing this mapping to a higher dimensional space, which is referred to the use of the kernel function, and then they construct the decision frontier in that space. The linear SVM relies on a dot product between data point vectors, as illustrated in equation 2.11. However, the nonlinear SVM classifier relies on a dot product between feature vectors which will be illustrated mathematically later. It means that it is possible to increase the separability of the data $\mathbb{R}^N$ by mapping it to a high-dimensional space H using a non-linear kernel basis function φ which can be defined as follows:

$$\varphi : \mathbb{R}^N \rightarrow H \tag{2.12}$$

This kernel function can be used to define discriminant function of the SVM classifier in the feature space H as follows:

$$f(x) = \sum_{i=1}^{m} w_i y_i \varphi(x_i)^T \varphi(x) + b \tag{2.13}$$

where $x_1, x_2, \ldots, x_m$ are the support vectors, w_i are *Lagrange multipliers* [1], y_i are the

[1] Lagrange multipliers provide a strategy for finding the local maxima and minima of a function subject to equality constraints. For instance, consider the following optimization problem: $f(x_1,\ldots,x_n)$: $\mathbb{R}^n \rightarrow \mathbb{R}$ subject to $g(x_1,\ldots,x_n) = 0$. In other words, you need to find the minimum and maximum

corresponding labels of the input samples x_i, and b is a bias. It can be seen that the computations in equation 2.13 depend on the inner product of the vectors in the feature space. Unfortunately, performing these computations in a high dimensional space can be extremely costly. However, this problem can be solved exploiting the idea of kernel function. In other words, the training depends only on the inner products of the form $\varphi(x_i)^T\varphi(x)$. Consequently, we can overcome determining the feature space representation of the vectors by introducing the kernel function, which is defined as follows:

$$K(x,y) = \varphi(x)^T\varphi(y) \tag{2.14}$$

Thus after the substitution, equation 2.13 becomes:

$$f(x) = \sum_{i=1}^{m} w_i y_i K(x,y) + b \tag{2.15}$$

Note that the kernel function $K(x,y)$ is a similarity measure between the two vectors x and y which is sometimes mentioned as *Mercer kernels theorem*. In other words, the kernel function must satisfy the *Mercer's theorem*, that is, the *kernel matrix K* given by:

$$K = \begin{bmatrix} K(x_1,x_1) & \dots & K(x_1,x_n) \\ \vdots & \ddots & \vdots \\ K(x_n,x_1) & \dots & K(x_n,x_n) \end{bmatrix} \tag{2.16}$$

must have only non-negative eigenvalues. A thorough explanation of this theorem can be found in [Cristianini and Shawe-Taylor, 2000].

This similarity measure is obtained using different kernels in order to classify various kinds of data (see for instance [Chapelle et al.; Wallraven et al., 2003]). The more important kernel functions are:

- Linear kernel:

$$K(x,y) = x^T y \tag{2.17}$$

- Polynomial kernel:

$$K(x,y) = (x^T y + \theta)^d \tag{2.18}$$

extremes with respect to the constraint g.

where θ denotes the first coefficient of this kernel function (default is 0) and d denotes the degree of the kernel function.

- Gaussian kernel:
$$K(x,y) = e^{-\frac{||x-y||^2}{2\sigma^2}} = e^{-\gamma||x-y||^2} \tag{2.19}$$
where γ is a coefficient in the kernel function (default is $1/features\_number$).

- Sigmoid kernel:
$$K(x,y) = tanh(\gamma x^T y + \theta) \tag{2.20}$$

- Exponential kernel:
$$K(x,y) = e^{-\frac{||x-y||}{2\sigma^2}} = e^{-\gamma||x-y||} \tag{2.21}$$

The parameters of these kernel functions are specified by the user, usually experimentally. For instance, in the sigmoid kernel we keep changing in γ and θ until the *Mercer's theorem* is satisfied. In chapter 5, we will employ some of these kernels and study their impact on the final classification results.

SVM approaches have been widely used in the literature. For instance, [Pronobis and Caputo, 2007] proposed to use cue integration followed by a SVM classification technique. Also, in [Ullah et al., 2008] the authors used SIFT descriptor combined with Harries Laplace Detectors (HLDs) for feature extraction, followed by the use of SVM to perform the classification process. These algorithms lead to a highly accurate result in recognition and is robust to the noise. However, they are based on sophisticated classifiers.

2.3.2.3 Softmax regression

In statistics, this model is a probabilistic, linear classifier (for example see figure 2.15). It is a supervised learning algorithm which can be used to predict of the probability of occurrence of an event based on the input data. More precisely, this technique can also be used in recognition for robotic systems to compute the probability to be in a given place according to the input image.

Binary or binomial classification is the task of classifying the members of a given set of objects into two groups on the basis of whether they have some property or

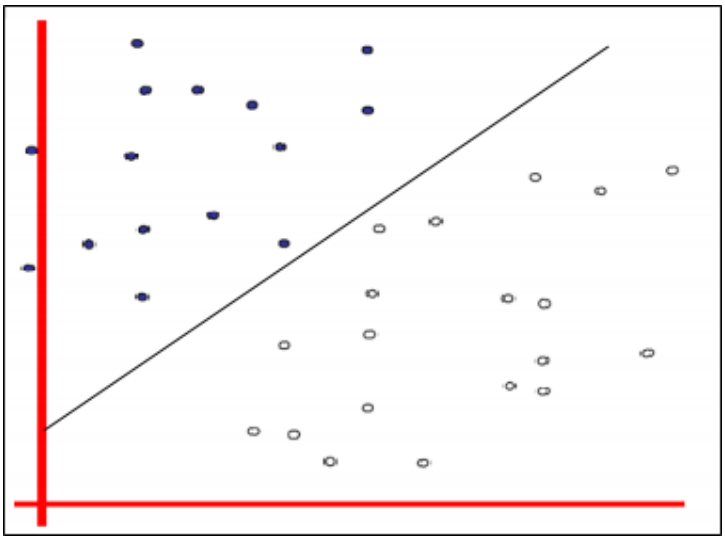

Figure 2.15: Simple example of linear classification for two different classes.

not. For instance, distinguishing e-mails into two different classes of spam and not-spam. While, multiclass or multinomial classification is the problem of classifying instances into more than two classes. For instance classifying the e-mails into three different classes; spam, not-spam, and personnel e-mail or performing the recognition task on MNIST handwritten digits to classify them into 10 different classes. This can be modeled as a distributed system according to a multinomial distribution. However, before explaining the softmax regression model, we want to give a brief description of the logistic function which is a special case of the softmax regression.

Given a training set: $\left\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\ldots,(x^{(m)},y^{(m)})\right\}$ where $y^{(i)}$ denotes the labeled samples of the input features $x^{(i)} \in \mathbb{R}^{n+1}$. If the labels are taking the following form: $y^{(i)} \in \{0,1\}$, a logistic regression is then enough to classify the data. While if $y^{(i)} \in \{1,2,3,\ldots,k\}$, a softmax regression is required to distinguish between k classes. For a logistic regression, the hypothesis is given by the following formula:

$$h_\theta(x) = \sigma(\theta^T x) = \begin{bmatrix} P(y^{(i)}=1|x^{(i)};\theta) \\ P(y^{(i)}=2|x^{(i)};\theta) \end{bmatrix} = \frac{1}{1+e^{-\beta\theta^T x}} = \frac{1}{\sum_{j=1}^{2} e^{\beta\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\beta\theta_1^T x^{(i)}} \\ e^{\beta\theta_2^T x^{(i)}} \end{bmatrix} \tag{2.22}$$

where

$$\sigma(z) = \frac{1}{1+e^{-\beta z}} \tag{2.23}$$

is called the logistic function. β represents the inverse temperature which determines

the slope of the sigmoid function. Figure 2.16 shows this function, where, in this case, the inverse temperature is assumed to be 1. Note that this function tends towards 1 as $z \to \infty$, and it tends towards 0 as $z \to -\infty$. Therefore, the logistic function and hence $h(x)$, are always bounded between 0 and 1. The model parameters θ_1 and θ_2, which are also known as "regression coefficients", can be learned minimizing the following cost function:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log h_\theta(x^{(i)}) + (1 - y^{(i)}) log\left(1 - h_\theta(x^{(i)})\right)\right] \tag{2.24}$$

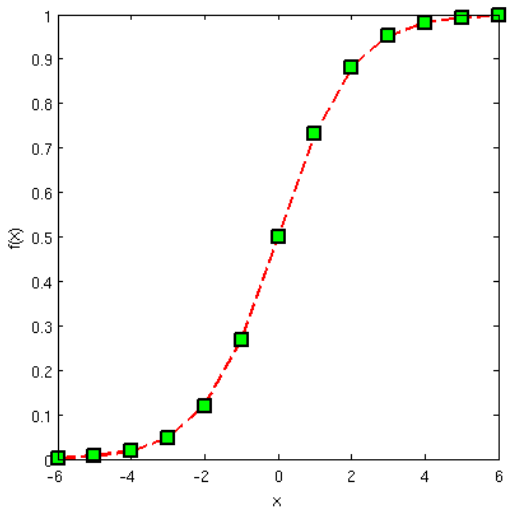

Figure 2.16: Logistic sigmoid function: $f(x) = 1/(1 + Exp(-\beta x))$ with $\beta = 1$. This function can be used as an "activation function" for a mathematical model of a neuron.

Softmax and logistic regressions are discriminative approaches since they try to approximate decision frontiers between the data. If the classes are mutually exclusive, so a softmax regression classifier would be appropriate [1]. However, if they are not mutually exclusive it would be more appropriate to build a set of separate logistic regression classifiers. It means that the non-exclusive case for multiple classes is just an extension of the binary case. A data belongs or not (with a given probability) to each of the considered classes but these probabilities do not have to sum to 1. On contrary, all the probabilities of the classes in the exclusive case have to sum to 1 and thus

[1] Two events are said to be mutually exclusive if $P(A \cap B) = 0$. Note that if all events in a sample space are mutually exclusive, then all the probabilities must sum to 1 and thus $P(A \cup B) = 1$

we need to use a softmax regression instead of the logistic one for our classification problem. It results that in our classification problem using a non-exclusive approach leads to a scene classification (a scene can belong to several classes because it can include parts of different rooms or locations). While using an exclusive approach leads to a real place recognition since the robot cannot be in two different places at the same time. So, logically, a softmax regression classifier would be appropriate for our classification problem because it ensures that probabilities of the different classes are mutually exclusive.

The hypothesis of a softmax regression for a multinomial distribution takes the following form:

$$h_\theta(x) = \begin{bmatrix} P(y^{(i)} = 1|x^{(i)};\theta) \\ P(y^{(i)} = 2|x^{(i)};\theta) \\ \vdots \\ P(y^{(i)} = k|x^{(i)};\theta) \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} e^{\beta\theta_1^T x^{(i)}} \\ e^{\beta\theta_2^T x^{(i)}} \\ \vdots \\ e^{\beta\theta_k^T x^{(i)}} \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\beta\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\beta\theta_1^T x^{(i)}} \\ e^{\beta\theta_2^T x^{(i)}} \\ \vdots \\ e^{\beta\theta_k^T x^{(i)}} \end{bmatrix} \tag{2.25}$$

where Z represents the "partition function" which normalizes the data distribution, so that it sums to one. Also, it is important to note that the model parameters, $\theta_1, \theta_2, \ldots, \theta_k \in \mathbb{R}^{n+1}$, are stacking up in rows and they are given by the following matrix:

$$\theta = \begin{bmatrix} -\theta_1^T - \\ -\theta_2^T - \\ \vdots \\ -\theta_k^T - \end{bmatrix} \tag{2.26}$$

If these regression coefficients are positive, the probability of the softmax function outcome will be increased, while negative regression coefficients mean that the probability of that outcome will be decreased. Large coefficients of the regression will strongly influence the probability, while very small regression coefficients will have a small influence on the probability of that outcome. To overcome these challenges, we need to regularize the model coefficients during the learning phase. This can be achieved using a regularized term or a weight decay, which penalizes the large parameters during the learning process.

Like in the logistic regression, the parameters of a softmax regression can be learned minimizing the following cost function:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{e^{\beta\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\beta\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2 \tag{2.27}$$

where the first term on the left-hand-side of the previous equation represents the regular cost function of a softmax regression and the second term on the right-hand-side of the same equation represents the weight decay term. Also, in the first term there is an indicator function, $\{\cdot\}$, which takes a value of 1 if its argument is true and 0 otherwise. So, the idea behind this model is to minimize this cost function, $J(\theta)$, by changing the model parameters, θ. This can be achieved using a gradient descent procedure as follows:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta),\ for\ all\ j \tag{2.28}$$

where α is the learning rate of the model. The partial derivative of the cost function, $\frac{\partial}{\partial\theta_j}J(\theta)$, can be derived as follows:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \nabla_{\theta_j}J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left(1\{y^{(i)} = j\} - P(y^{(i)} = j|x^{(i)};\theta)\right)\right] + \lambda\theta_j \tag{2.29}$$

where $P(y^{(i)} = j|x^{(i)};\theta)$ represents the conditional probability of a given class with respect to the model parameters. This probability can be rewritten as follows:

$$P(y^{(i)} = j|x^{(i)};\theta) = \frac{e^{\beta\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\beta\theta_l^T x^{(i)}}} \tag{2.30}$$

The demonstration of this equation can be found in Appendix B and in [Ng, 2011], and a pseudo code of the update rule for a softmax regression is proposed in Algorithm 1.

It was stated that the use of a softmax regression depends on the assumption of obtaining the linear separation of the data using DBNs. However, if this assumption is false, a nonlinear classifier, like SVM, is required to perform the classification process.

The question of whether discriminative or generative approaches are more efficient to solve robot localization problem seems to be interesting to explore in this context.

input : the model parameters, θ, which are the weights matrix, W, and the biases, b. y represents the top hidden layer, the matrix z represents the pre-defined labels of the classes ($1\{y^{(i)} = j\}$), $\lambda = 0.008$ is a regularization rate used to avoid over-parameterization problem, $\alpha = 0.1$ is a learning rate of the gradient descent, and finally, the number of required epochs to ensure the convergence of the model parameters, set to 10000.

output: recognize the robot places such as corridor, toilet, office, ... *etc*.

for $e = 1$ **to** *epochs* **do**

  compute the conditional probability using:

  $P(y^{(i)} = j|x^{(i)};\theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}$

  compute the partial derivative of the cost function using:

  $\frac{\partial}{\partial \theta_j} J(\theta) = \nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[x^{(i)} \left(z - P(y^{(i)} = j|x^{(i)};\theta) \right) \right] + \lambda \theta_j$

  use the gradient descent method which is given in equation 2.28 to update the model parameters as follows:

  $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta),\ for\ all\ j$

end

Algorithm 1: This algorithm shows the learning update procedure of a soft-max regression.

This question has been investigated in the literature, for example, in [Ng and Jordan, 2002] the authors have introduced an interesting comparison between the two different approaches considering two particular classification algorithms. The first one is the logistic regression, which is a discriminative approach, and the second one is NBC, which is a generative one. Together they form "discriminative-generative approach" pair. It means that while naive Bayes aims at maximizing the joint likelihood, $P(x,y)$, of the inputs x and the labels y, the logistic regression aims at maximizing the conditional likelihood on the training set, $P(y|x)$. In this case, the authors have empirically shown that the performance of naive Bayes is better than the logistic regression for less data, but the asymptotic error is higher for the former. They have also shown that the error in the generative model may converge asymptotically much faster than the discriminative approach, where the number of training examples is only logarithmic in the generative model and linear in the discriminative one. However, these hypotheses are not true for all pairs of discriminative and generative models, *i.e.* we can not generalize them. Hence, the conclusion of Ng and Jordan is that discriminative learning can sometimes be more efficient than generative learning algorithms for some problems. On the other hand, generative learning models might be advantageous for other problems at least when the model considered fits well the data. Another interesting work is presented in [Ulusoy and Bishop, 2005] where the authors have introduced and compared the two approaches for object recognition based on local invariant features. They have shown that a discriminative model is capable of very fast inference, and is able to focus on highly informative features. By contrast, the generative model gives high classification accuracy, and also has some ability to localize the objects within the image.

As a conclusion, after presenting the different existing approaches that have been used to achieve SPR, we have noted that these approaches generally include two main phases of coding and classification. We have also seen that most of the coding methods are based on hand-crafted feature extractors like SIFT, SURF, CENTRIST, and GIST detectors. These detectors are empirical and they often use BoWs approaches [Chum et al., 2009; Philbin et al., 2007]. Some of them use the local feature matching [Lowe, 2004] and they often need to reduce the size of their representations [Torralba et al., 2008]. They are often followed by vector quantization such that the image can be represented as a histogram.

SPR thus requires the use of an appropriate feature space. In the next section we will see that DNNs offer an interesting alternative to these empirical methods.

2.4 Deep architecture methods

Semantic place recognition therefore requires projecting images onto an appropriate feature space that allows an accurate and rapid classification. Although in the previous approaches, the feature space was build in an empirical way, we are going to see that a set of recent methods based on deep architectures of neural networks give the ability to build it from theoretical considerations.

Concerning features extraction, the last two decades have seen the emergence of new approaches strongly related to the way natural systems code images [Olshausen and Field, 2004]. One of the roots of these methods is the analogy with the visual system, the first layers of which seem to correspond to a coding step and to the extraction by more and more specialized cells of universal elements of the images, *i.e.* elements that are present in almost all natural images. It has been shown that these elements are parts of contours and their combinations [Field, 1994; Olshausen and Field, 2004]. These approaches are based on the consideration that natural image statistics are not Gaussian as it would be if they had a completely random structure [Field, 1994]. The auto-similar structure of natural images allows the evolution to build "optimal codes". These codes are made of statistically independent features and many different methods have been proposed to construct them from image datasets. One characteristic of these features is their locality, that can be related to the notion of receptive field in natural systems.

It has been shown that Independent Component Analysis (ICA) [Bell and Sejnowski, 1997] produces localized features. Besides, it is efficient for distributions with high kurtosis well representative of natural image statistics, dominated by rare events like contours; however the method is linear and not recursive. These two constraints are released by Deep Belief Networks (DBNs) [Hinton et al., 2006] that introduce non-linearities in the coding scheme and exhibit multiple layers.

Each layer in DBNs is made of a Restricted Boltzmann Machine (RBM), a simplified version of a Boltzmann Machine proposed by Smolensky [Smolensky, 1986] and Hinton [Hinton, 2002]. Each RBM is able to build a generative statistical model of

its inputs using a relatively fast learning algorithm, Contrastive Divergence (CD), first introduced by Hinton [Hinton, 2002]. Another important characteristic of the codes used in natural systems, the sparsity of the representation [Olshausen and Field, 2004] is also achieved in DBNs.

Deep architecture learning has indeed recently become popular as a powerful way to code data using a set of independent features [Bengio, 2009]. In particular, deep neural networks (DNNs) like DBNs and deep Boltzmann machines (DBMs) have been applied to different machine learning tasks with impressive improvements over conventional approaches ([Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2009]). They have recently been used in different applications such as phone recognition in [Abdel-Rahman et al., 2011], natural language processing in [Sarikaya et al., 2011], and audio processing in [Abdel-Rahman et al., 2012]. They have also been used for hand-written character recognition [Hinton, 2002; Hinton et al., 2006], object recognition [Nair and Hinton, 2009], collaborative filtering [Salakhutdinov et al., 2007] and document retrieval. Based on previous observations, RBMs can be used to model high-dimensional, sequential data and they have proved to be very successful for motion capture data modeling [Taylor et al., 2006]. They have shown to be efficient and powerful for image coding. [Hinton et al., 2006; Torralba et al., 2008].

In [Torralba et al., 2008] the authors have shown that DBNs can be successfully used to code huge amounts of images in an efficient way. Each image in a very large database is first reduced to a small size patch (*e.g.* 32x32) to be used as an input vector for a DBN network. A set of predefined features (the alphabet) is computed, only once, from a set of representative images and each image is represented by a unique weighted combination of features taken from the alphabet. With the appropriate parameters the CD algorithm converges towards a sparse representation of the images, which means that an image is coded by the smallest possible number of features. A simple distance measurement between the image codes allows comparing them. To better understand how DBNs approaches can be used for image coding, we give a detailed description of them in the next sections.

DNNs are characterized by a large number of layers of neurons and by the use of layer-wise unsupervised pre-training to learn a probabilistic model for the data. A DBN is typically constructed by multiple layers of RBMs stacking so that the hidden

layer of one RBM becomes the visible layer of another higher RBM layer. Layer-wise pre-training of RBMs then facilitates finding a more accurate model for the data. Many researchers have empirically shown that such multi-stage learning works better than conventional learning methods, such as the back-propagation with random initialization [Hinton and Salakhutdinov, 2006; Ranzato et al., 2010; Salakhutdinov and Hinton, 2009]. It is thus important to have an efficient method for RBM training.

More precisely, deep architectures are used to find a high-level representation of the initial data (extract the most interesting features of the input image and use them to create a new representation of the initial data) for instance, see figure 2.17.

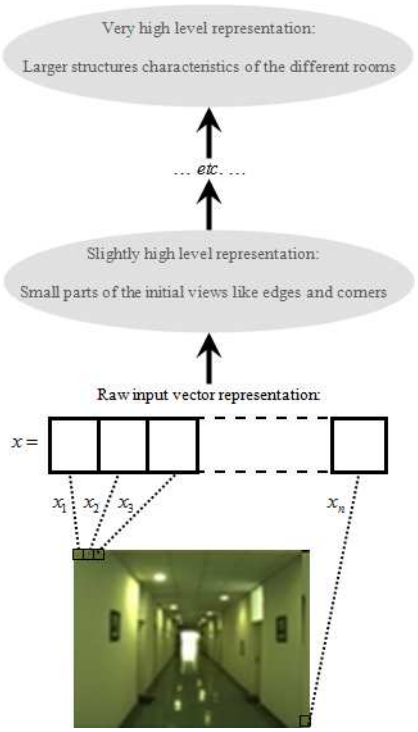

Figure 2.17: An explanation of how to transform the input image into higher levels of representation, which includes the most interesting information (characteristics) such as: edges, corners, object parts, *etc.*

Because DBNs are based on RBMs, which are particular type of Energy-Based Models (EBMs), we first introduce the main mathematical concepts of EBMs (A more detailed description can be found in [LeCun et al., 2006]), a detailed discussion on Boltzmann Machines (BMs) and its simpler variant, RBMs, that could be helpful to understand the main concepts of DBNs models. Then, we focus on describing the

mathematical concepts of Contrastive Divergence (CD) as a powerful learning algorithm that can be used to train DBNs models.

2.4.1 Energy-based models

Several methods have been proposed to achieve image coding [Bell and Sejnowski, 1997; Hinton et al., 2006; Olshausen and Field, 1996, 1997]. Some of them are Non-Energy-Based Models (NEBMs) such as Independent Component Analysis (ICA). It has been shown that ICA produces localized features and is efficient for distributions with high kurtosis well representative of natural image distributions; however this method is linear and non-recursive as previously said [Bell and Sejnowski, 1997]. These restrictions are released by DBNs [Hinton et al., 2006] which are a particular type of EBMs based on RBMs.

The energy-based approach is interesting because it suggests an ICA extension to overcomplete [Olshausen and Field, 1997] and multi-layer models [Teh et al., 2003]. It has also been shown that the features of an EBM exhibit marginal dependencies [Teh et al., 2003]. Allowing these dependencies can strongly contribute in speeding up the inference process for the model. While in causal generative models, like ICA, the assumption of marginal independence often leads to intractable inference which needs to be approximated using some iterative, data dependent scheme. The role of these iterations can be understood as suppressing the "activity" of less relevant feature, thus producing a sparse code. However, EBMs can be enriched with inhibitory lateral connections to suppress less relevant features in order to produce a sparser representation.

Another powerful generalization of EBMs is a hierarchical non-linear architecture in which the output activities are computed with a feed-forward neural network (see figure 2.9), where each layer may contribute to the total energy (for related work see [Hyvärinen et al., 2001]). To fit this model to data, back-propagation or CD techniques can be used to compute the energy gradients with respect to both data vector and weights. Finally, the authors in [Teh et al., 2003] have concluded that the EBMs provide a flexible modeling tool which can be trained efficiently to uncover useful structures in the data.

Usually, the main purpose of statistical modeling and machine learning is to encode dependencies between variables [LeCun et al., 2006]. By capturing those dependen-

cies, a model can be used to answer questions about the values of unknown variables given the values of known variables. Recognition systems capture the dependencies between a set of observed variables $\mathbf{x}$, for example the pixels of an image, and a set of answer variables $\mathbf{y}$ to be predicted (*e.g.* the robot places of natural images). An EBM takes all the variables (observed and unobserved) as inputs, and produces a scalar energy $(\mathbf{y},\mathbf{x})$ which measures the "compatibility" between the values of the variables. More precisely, in an EBM the inference process is done by choosing a value $\mathbf{y}^*$, from the set of all possible values of the unobserved variables $\mathbf{y}$, for which the energy function $E(\mathbf{y},\mathbf{x})$ is the smallest:

$$\mathbf{y}^* = argmin_{y\in\mathcal{Y}} E(y,\mathbf{x}) \tag{2.31}$$

where $\mathcal{Y}$ is a suitably defined domain for $\mathbf{y}$ [1]. Therefore, EBMs associate to each configuration a global energy, E, that is the sum of a number of local contributions which define the probability for an image to be proportional to $exp(-E)$ [Welling et al., 2004], *i.e.* EBM is one of the exponential family forms. In particular, one can transform an EBM into a probabilistic model through the Gibbs distribution:

$$P(\mathbf{y}|\mathbf{x}) = \frac{e^{-\beta E(\mathbf{y},\mathbf{x})}}{Z} \tag{2.32}$$

where Z is a normalization factor or a "partition function". It represents the sum of the numerator over all possible observation vectors of the input space and it is given by:

$$Z = \sum_{y\in\mathcal{Y}} e^{-\beta E(y,\mathbf{x})} \tag{2.33}$$

where the parameter β is an arbitrary positive constant, the "inverse temperature $1/T$", which determines the slope of the energy function.

The EBM inference through the energy minimization can therefore be seen as a *Maximum A Posteriori* (*MAP*) estimation of $\mathbf{y}$. In general, we would like to learn

[1] inference process is the task of finding the best answer for a given input. For example, $\mathbf{y}$ could take six possible values: animal, human figure, airplane, truck, car, and "none of the above". Given a fixed input $\mathbf{x}$, which is observed from the world, the process of inference involves, asking the model to produce a value of the unobserved variable $\mathbf{y}$ that is most compatible with the observed variable $\mathbf{x}$ [LeCun et al., 2006].

a set of features based on the principle of having low energy between the different configurations of a model. In the following sections, we describe how it is possible to compute the conditional probability of one layer given the other one for BM and RBM models, which are particular examples of EBMs, using the exponential family distributions.

2.4.2 Classical Boltzmann machines

The general Boltzmann Machine (BM) learning algorithm is a kind of probabilistic generative models which was originally introduced by Hinton and Sejnowski [Hinton et al., 1984]. As shown in figure 2.18 (left), the classical BMs can be viewed as a network of binary probabilistic units, which interact through weighted undirected connections. In this model, the network is fully connected, *i.e.* a BM consists in one layer of visible units, **v**, and one layer of hidden units, **h**. The units in each layer are fully connected and are also connected to all other units in other layers. The visible units are usually clamped by the observed data and the hidden units can be computed using equation 2.32 by letting the network run freely and sampling the activities of all units.

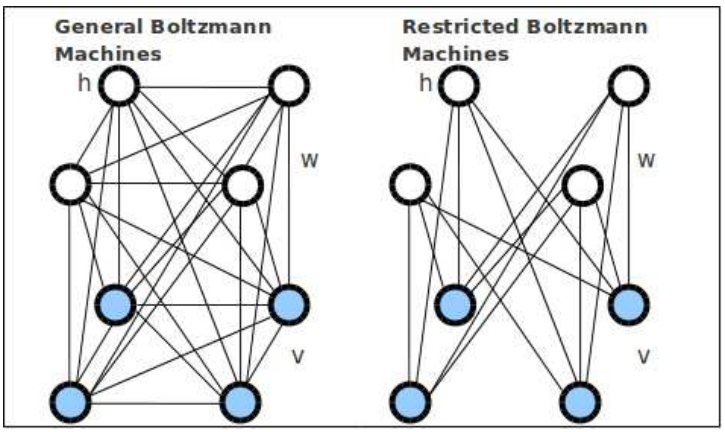

Figure 2.18: **Left**: a general Boltzmann Machine. The top layer represents a vector of hidden features, **h**, the bottom layer represents a vector of visible units, **v**, and **w** represents the symmetric interactions between **v** and **h** layers. **Right**: A restricted Boltzmann machine with no hidden-to-hidden and no visible-to-visible connections. Since there are no direct connections within the same layer, the activation function can update all units simultaneously.

Given the units of BM, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, the energy function of BM is postulated

by the following equation:

$$E(\mathbf{x};\theta) = -\sum_i \sum_j w_{ij} x_i x_j - \sum_i b_i x_i, \; for \; i < j \tag{2.34}$$

where $\theta = \{\mathbf{W}, \mathbf{b}\}$ denotes the model parameters consisting of a weight matrix $\mathbf{W} = [w_{ij}]$ and a bias vector $\mathbf{b} = [b_i]$. w_{ij} is the weight of the synaptic connections (symmetric connections) between neurons i and j. As in equation 2.32, the probability of a particular state $\mathbf{x}$ is then given through the Gibbs distribution as follows:

$$P(\mathbf{x};\theta) = \frac{e^{-\beta E(\mathbf{x};\theta)}}{Z(\theta)} = \frac{e^{-\beta E(\mathbf{x};\theta)}}{\sum_{\mathbf{u}} e^{-E(\mathbf{u};\theta)}} \tag{2.35}$$

For the binary case, the above conditional probability equation of a single unit x_i, given the states of all other units can be driven as follows:

$$P(x_i = 1|\mathbf{x};\theta) = \sigma(b_i + \sum_{i \neq j} w_{ij} x_j) \tag{2.36}$$

where the sigmoid function is given by: $\sigma(x) = 1/(1+e^{-\beta x})$. The derivation of equation 2.36 is explained in Appendix A. This derivation is universal for RBM and general BM [Krizhevsky, 2009].

The neurons of BM are usually divided into visible and hidden units $\mathbf{x} = [\mathbf{v}, \mathbf{h}]$, where the states $\mathbf{v}$ of the visible neurons are clamped to observed data, and the states $\mathbf{h}$ of the hidden neurons can change freely as previously said. In this case, the probability of a specific configuration of the visible neurons can be computed by marginalizing out the hidden neurons.

Although general BMs are theoretically easy to understand, they have not proven to be useful for practical problems in machine learning or inference [Hinton, 2002]. This is due to the fact that the learning is impractical in general BMs, *i.e.* the convergence process needs long time to be achieved since we have to learn $P(\mathbf{x})$ as it was illustrated in equation 2.36. In fact, the unconstrained connectivity between the units (see figure 2.18 (left)) is the main problem in those approaches. However, this connectivity problem has been restricted by introducing RBM models, which can be useful in practical problems.

2.4.3 Gaussian-Bernoulli restricted Boltzmann machines

In 1986, Smolensky introduced Restricted Boltzmann Machines (RBMs) [Smolensky, 1986] as a powerful learning algorithm which trains deep networks in a greedy layer-wise fashion. In other words, RBMs train one hidden layer of DBNs at a time by minimizing the energy function which is given in equation 2.37. Contrarily to feed-forward architectures, which support only bottom-up inference, RBMs are generative approaches for image coding, which support both bottom-up and top-down inference processes [1].

Unlike a classical Boltzmann machine, a RBM is a bipartite undirected graphical model $\theta = \{w_{ij}, b_i, c_j\}$, that learns a generative model of the observed data. It consists in two layers. The hidden layer, containing latent variables $\mathbf{h}$, is used to generate the visual layer, containing observed variables $\mathbf{v}$. While generation $P(\mathbf{v}|\mathbf{h})$ is learned, the undirected connections also allow recognition $P(\mathbf{h}|\mathbf{v})$. The two layers are fully connected through a set of weights w_{ij} and biases $\{b_i, c_j\}$, and there are no connections between units of the same layer, as shown in figure 2.18 (right). As illustrated in [Hopfield, 1982], a joint configuration, ($\mathbf{v}$,$\mathbf{h}$) of the visible and hidden units has an energy function, $E(\mathbf{v}, \mathbf{h}; \theta)$, given by:

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_i \sum_j v_i h_j w_{ij} - \sum_{i \in \mathbf{v}} b_i v_i - \sum_{j \in \mathbf{h}} c_j h_j \tag{2.37}$$

This energy function corresponds to the binary states of visible units $\mathbf{v}$ and hidden units $\mathbf{h}$. The probabilities of the state for a unit in one layer conditional to the state of the other layer can therefore be easily computed. According to Gibbs equation:

$$P(\mathbf{v}, \mathbf{h}; \theta) = -\frac{1}{Z(\theta)} \exp^{-\beta E(\mathbf{v}, \mathbf{h}; \theta)} \tag{2.38}$$

where $\theta = \{w, b, c\}$ represents the model parameters, and $Z(\theta)$ is again the "partition function". Intuitively, configurations with low energy are assigned high probability, while configurations with high energy are assigned low probability.

[1] Bottom-up inference is the synthesis of new information from the old one, while top-down inference is the analysis of goals into subgoals. The lower RBM layers could support object detection by spotting low-level features indicative of object parts. Conversely, information about objects in the higher RBM layers could resolve lower-level ambiguities in the image or infer the locations of hidden object parts.

Thus after marginalization, the probability of a particular hidden state configuration $\mathbf{h}$ can be derived as follows:

$$\begin{aligned} P(\mathbf{h};\theta) &= \sum_{\mathbf{v}} P(\mathbf{v},\mathbf{h};\theta) \\ &= \frac{\sum_{\mathbf{v}} e^{-\beta E(\mathbf{v},\mathbf{h};\theta)}}{\sum_{\mathbf{v}}\sum_{\mathbf{h}} e^{-\beta E(\mathbf{v},\mathbf{h};\theta)}} \end{aligned} \tag{2.39}$$

The sum in the denominator is over all possible visible and hidden configurations, and is thus extremely hard to compute when the number of units is large. However, as previously said, in RBMs there are no direct connections between the visible neurons or the hidden neurons. It can thus be easy to write down the conditional probability of a single unit being either 0 or 1 given the states of the other units as follows:

$$P(h_j = 1 \mid \mathbf{v};\theta) = \frac{P(h_j = 1, \mathbf{v};\theta)}{P(\mathbf{v};\theta)} \tag{2.40}$$

Given the energy function, $E(v,h)$, of the visible and hidden units, we can rewrite the above conditional probability equation as follows:

$$P(h_j = 1 \mid \mathbf{v};\theta) = \frac{e^{-\beta E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{h}} e^{-\beta E(\mathbf{v},\mathbf{h})}} \tag{2.41}$$

where β represents the inverse temperature, $1/T$, which determines the slope of the sigmoid function. However, for the binary case where $h_j \in \{0,1\}$, the above probability equation of turning on can be derived using the logistic sigmoid function as demonstrated in [Krizhevsky, 2009] and Appendix A, according to its energy function, as follows:

$$P(h_j = 1 \mid \mathbf{v};\theta) = \sigma(c_j + \sum_i w_{ij} v_i) \tag{2.42}$$

Once the hidden binary states are computed, we produce a "reconstruction" of the original patch by setting the state of each visible unit to be 1 with probability:

$$P(v_i = 1 \mid \mathbf{h};\theta) = \sigma(b_i + \sum_j w_{ij} h_j) \tag{2.43}$$

However, logistic or binary units are not appropriate for multi-valued inputs like pixel levels, because logistic units are a very poor representation for data such as patches of natural images [Hinton, 2010]. To overcome this problem, as suggested by Hinton [Hinton, 2010], in the present work we replace the binary visible units by a zero-mean Gaussian activation scheme as follows:

$$P(v_i = 1 \mid \mathbf{h}; \theta) \leftarrow \mathcal{N}(b_i + \sum_j w_{ij} h_j, \sigma^2) \tag{2.44}$$

Concerning the variance of the noise, σ^2, it is possible to learn it for each visible unit, but this is difficult using CD as it is time-consuming. It is more appropriate to first normalize the data components to have zero-mean and unit variance and then use a unit variance and zero-mean for the Gaussian noise. After this modification, Gaussian visible units and binary hidden units correspond to the following energy function model:

$$E(\mathbf{v}, \mathbf{h}; \theta) = \sum_{i \in \mathbf{v}} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j \in \mathbf{h}} c_j h_j - \sum_i \sum_j \frac{v_i}{\sigma_i} h_j w_{ij} \tag{2.45}$$

As a conclusion, contrarily to auto-encoder models that are non-statistical but "deterministic models" and aim at finding the best reconstruction of the data or reproduce the images. RBMs are generative models used to learn the optimal statistical model which explains the original data or images. Moreover, there are other good reasons to use RBMs:

First, these approaches are able to produce sparse efficient features extracted from a larger alphabet. A sparse representation means that the linear separability of the initial data should be gained in the feature space. Secondly, RBM can handle multi-modal stimuli. Units are independent of each other, since there are no connections between units of the same layer. This allows the units to encode different modalities without problems. Moreover, RBMs are just basic building blocks. They can be stacked on top of each other to create Deep Belief Networks. They can be extended to model time-series [Sutskever et al., 2008] or to feature three-way interactions ([Taylor and Hinton, 2009]. Finally, RBMs fall under Bayesian models, which have been used extensively to model the brain [Vilares and Kording, 2011]. However, full Bayesian models are computationally intractable and therefore biologically implausible. By using only an

approximation, we have not only fast inference and learning, but also a more plausible algorithm.

The parameters of RBM can be learned from the data using different training techniques. Some of them are: maximizing the log-likelihood, Markov-Chain Monte-Carlo (MCMC) sampling techniques like Gibbs sampling, and Contrastive Divergence (CD) learning algorithm. We will see in the next section how we could use these mechanisms for Product of Experts (PoEs) models training and we will concentrate on the use of CD as a faster and powerful learning algorithm to maximize the log-likelihood gradient.

2.4.4 Learning products of experts by minimizing contrastive divergence

It has been mentioned that Back-Propagation (BP) learning technique works well in a network where just a single layer is sufficient. However, it does not work well in networks with many hidden layers, like DBNs, because it requires to train the whole layers of the model [Bengio and LeCun, 2007; Hecht-Nielsen, 1995; Larochelle et al., 2009; Tesauro, 1992]. In other words, minimizing the error for each layer requires back-propagating the error on all model layers, updating the weights and biases. Therefore, the convergence of BP algorithm becomes more difficult in networks with multiple hidden layers, *i.e.* it is time-consuming to assure the convergence. Furthermore, BP algorithm is not very accurate in the case of DBNs because the gradient of error becomes flat for the different layers and weights. However, several alternative learning techniques have been proposed to train DBNs models. But before introducing them we need to understand the term "Products of Experts" (PoEs).

General BMs or RBMs are particular examples of PoEs model. PoE combines n individual models by taking the product of their conditional probabilities and normalizing the result using the partition function $Z(\theta)$ as follows [Hinton, 2002]:

$$P(\mathbf{d} \mid \theta_1, \ldots, \theta_n) = \frac{\prod_m P_m(\mathbf{d} \mid \theta_m)}{Z(\theta_m)} = \frac{\prod_m P_m(\mathbf{d} \mid \theta_m)}{\sum_{\mathbf{c}} \prod_m P_m(\mathbf{c} \mid \theta_m)} \tag{2.46}$$

where $\mathbf{d}$ is a discrete input vector, $\mathbf{c}$ represents the indexes for all possible vectors in the input space, θ_m represents the parameters of a particular model m, and $P_m(\mathbf{d} \mid \theta_m)$ is the

probability of $\mathbf{d}$ under the model m. A common way to learn the PoE parameters is to maximize the data log-likelihood. Given a training dataset $\{d_i\}_{i=1}^{N}$, the log-likelihood of PoE is given by:

$$log\ P(\mathbf{d} \mid \theta_1, \ldots, \theta_n) = \sum_{i=1}^{N} \sum_{j=1}^{M} log\ P(d_i \mid \{\theta_j\}) \tag{2.47}$$

By taking the first derivative of the log-likelihood function, $log\ P(\mathbf{d} \mid \theta_1, \ldots, \theta_n)$, with respect to the model parameters, θ_j, we can then drive the gradient descent as follows:

$$\frac{\partial log\ P(\mathbf{d} \mid \theta_1, \ldots, \theta_n)}{\partial \theta_m} = \left(\frac{\partial log\ P_m(\mathbf{d} \mid \theta_m)}{\partial \theta_m} - \sum_{\mathbf{c}} P(\mathbf{c} \mid \theta_1, \ldots, \theta_n) \frac{\partial log\ P_m(\mathbf{c} \mid \theta_m)}{\partial \theta_m} \right) \tag{2.48}$$

A thorough demonstration for this derivative can be found in Appendix C and in [Wood and Hinton, 2012]. In equation 2.48, the first part represents the data distribution, while the second one represents the expected derivative of the log-likelihood of an expert on fantasy data, $\mathbf{c}$, generated from the PoE [1]. The average over the data and model distributions in equation 2.48 can thus be written as follows:

$$\left\langle \frac{\partial log\ P(\mathbf{d} \mid \theta_1, \ldots, \theta_n)}{\partial \theta_m} \right\rangle_{Q^0} \propto \left(\left\langle \frac{\partial log\ P_m(\mathbf{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^0} - \left\langle \frac{\partial log\ P_m(\mathbf{c} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^\infty} \right) \tag{2.49}$$

However, the computation of Q^∞ seems to be difficult to obtain, especially in an inner gradient ascent loop. In other words, computing the partition function, $Z(\theta)$, requires to compute the summation over all possible configurations of BM, and it is simply impossible for large BMs. Fortunately, this problem can be tackled in various ways. One obvious approach is to use Markov-Chain Monte-Carlo (MCMC) sampling techniques to compute the stochastic gradient to maximize the log-likelihood. Due to the simplicity of the activation rule for a single neuron given the states of other neurons, a simple Gibbs sampling is enough to get stochastic gradients. A detailed description of how to use Gibbs sampling method for traditional BMs training can be found in [Geman and Geman, 1984].

[1] Fantasy data (or confabulation data) is the reconstruction data produced by training an observed data-vector using the CD algorithm.

However, there are also other kinds of limitations in Gibbs sampling use for BM training. The biggest problem is due to the full-connectivity of BM: since each neuron is connected to and influenced by all the other neurons, it takes as many steps as the number of neurons to get one sample of the BM state. Even when the visible neurons are clamped to the training data, the number of required steps for a single fresh sample is still at least the number of hidden neurons. This makes the successive samples in the chain highly correlated with each other, and this poor mixing affects the performance of learning. Another limitation of this approach is that multi-modal distributions are problematic for Gibbs sampling due to the nature of component-wise sampling, the samples might miss some modes of the distribution [Salakhutdinov, 2009].

MCMC learning techniques are therefore not useful for classical BMs training where the network is fully-connected. They also remain slow in training RBMs even if we restrict the connectivity of the visible and hidden units and thus cannot be used to train RBMs.

Another way based on Kullback-Leibler Divergence (KLD), first introduced by [Kullback and Leibler, 1951], can be used to maximize the log-likelihood. It has been formally demonstrated that maximizing the log-likelihood of the data (averaged over the data distribution) is equivalent to minimizing the KLD between the data distribution, Q^0, and the equilibrium distribution over the visible variables, Q^∞, that is produced by prolonged Gibbs sampling from the generative model [Hinton, 2002] [1]. The KLD between the data and model distributions can then be written as follows:

$$\begin{aligned} D_{KL}\left(Q^0 \| Q^\infty\right) &= \sum_{\mathbf{d}} Q^0_{\mathbf{d}} ln \frac{Q^0_{\mathbf{d}}}{Q^\infty_{\mathbf{d}}} \\ &= \sum_{\mathbf{d}} Q^0_{\mathbf{d}} log Q^0_{\mathbf{d}} - \sum_{\mathbf{d}} Q^0_{\mathbf{d}} log Q^\infty_{\mathbf{d}} \\ &= -H\left(Q^0\right) - \langle log Q^\infty_{\mathbf{d}} \rangle_{Q^0} \end{aligned} \tag{2.50}$$

where $\|$ represents the KLD operator, $\langle \cdot \rangle$ denotes the cross entropy of Q^0 and $Q^\infty_{\mathbf{d}}$ (expectations over the distribution), $H\left(Q^0\right)$ represents the entropy of the data distribution which can be ignored during learning because Q^0 does not depend on the model parameters, and $Q^\infty_{\mathbf{d}} = P(\mathbf{d} \mid \theta_1, \ldots, \theta_n)$.

[1] Q^0 is a natural way to denote the data distribution if we imagine starting a Markov chain at the data distribution at time 0 [Hinton, 2002].

However, as illustrated in [Hinton, 2002], instead of minimizing the KLD between Q^0 (initial derivative) and Q^∞ (final derivative), it is possible to minimize the divergence between $(Q^0 \| Q^\infty)$ and $(Q^1 \| Q^\infty)$ where Q^1 can be computed by performing one step of reconstruction of the data generated by one full step of Gibbs sampling. In fact, this minimization represents the definition of "Contrastive Divergence" (CD) which was proposed by Hinton [Hinton, 2002] as an approximate learning method for PoE models training. Instead of running the chain to equilibrium and comparing the initial and final derivatives, we can simply run the chain for one full step and then update the parameters to reduce the tendency of the chain to wander away from the initial distribution on the first step. A comparison is thus made between the statistics of the data and the statistics of its representation generated by Gibbs sampling. Therefore, in contrastive divergence learning, we try to minimize the following related objective:

$$CD_n = KL(Q^0 || Q^\infty) - KL(Q^1 || Q^\infty) \tag{2.51}$$

The key benefit for the contrastive divergence is that the intractable expectation over Q^∞ on the right-hand-side of equation 2.50 cancels out as cited in [Hinton, 2002], *i.e.* the Q^∞ term of equation 2.51 cancels each other out, as explained in [Andrzejewski, 2009; Hinton, 2002]. Consequently, equation 2.49 can be re-written as follows:

$$-\frac{\partial}{\partial \theta_m}\left(Q^0 \| Q^\infty - Q^1 \| Q^\infty\right) = \left(\left\langle \frac{\partial log\ P_m(\mathbf{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^0} - \left\langle \frac{\partial log\ P_m(\hat{\mathbf{d}} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^1}\right) + \frac{\partial Q^1}{\partial \theta_m} \frac{\partial (Q^1 \| Q^\infty)}{\partial Q^1} \tag{2.52}$$

The first two terms of equation 2.52 are tractable, because we can compute the derivative of the initial data, d, and the derivative of the reconstruction data, $\hat{d}$. In other words, it is straightforward to sample from Q^0 and Q^1, while the third term is problematic to compute. However, extensive simulations have shown that this term can safely be neglected because it has a small effect on the final result compared with the other two terms [Hinton, 2002]. These extensive simulations were performed using RBMs with small numbers of visible and hidden units. By performing computations that are exponential in the number of hidden units and exponential in the number of

visible units, it is possible to compute the exact values of $\langle v_i^0 h_j^0 \rangle_{Q^0}$ and $\langle v_i^n h_j^n \rangle_{Q^1}$. It is also possible to measure what happens to $Q^0 \| Q^\infty - Q^1 \| Q^\infty$ when the approximation in equation 2.54 is used to update the weights by an amount that is large compared with the numerical precision of the machine but small compared with the curvature of the CD. After performing that, two histograms of the improvements in the CD and in data log likelihood have been presented in [Hinton, 2002]. The main conclusion is that the learning procedure does not always improve the log likelihood of training data, though it has a strong tendency to do so. However, when we ignore the third term in equation 2.52, the learning procedure in the case of CD becomes better. Consequently, by ignoring the third term in equation 2.52, the model parameters can be adjusted using the following update rule:

$$\Delta\theta_m \propto \left(\left\langle \frac{\partial log\ p_m(\mathbf{d} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^0} - \left\langle \frac{\partial log\ p_m(\hat{\mathbf{d}} \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^1} \right) \tag{2.53}$$

As shown in figure 2.19, CD learning starts by setting the states of the visible units to a training vector. Then the binary states of the hidden units are all computed in parallel using equation 2.42. Once binary states have been sampled for the hidden units, a "reconstruction" is produced by setting each v_i to 1 with a probability given by the equation 2.44. The overall update formula in the weights w_{ij} is therefore given by:

$$-\frac{\partial}{\partial w_{ij}} \left(Q^0 \| Q^\infty - Q^1 \| Q^\infty \right) \approx \langle v_i^0 h_j^0 \rangle_{Q^0} - \langle v_i^n h_j^n \rangle_{Q^1} \tag{2.54}$$

This equation can be rewritten as:

$$w_{ij} \leftarrow w_{ij} + \eta [\langle v_i^0 h_j^0 \rangle_{data} - \langle v_i^n h_j^n \rangle_{recon}] \tag{2.55}$$

where η denotes the learning rate and $\langle v_i^0 h_j^0 \rangle$ and $\langle v_i^n h_j^n \rangle$ are the cross product of the visible and hidden units with respect to the data and the model (reconstruction) distributions. v^0 corresponds to the initial data distributions, h^0 is computed using equation 2.42, v^n is sampled using the Gaussian distribution in equation 2.44 and with n full steps of Gibbs sampling, and h^n is again computed from equation 2.42. Then, for separate biases of visible and hidden neurons, the update rules are, in analogy to the

update rule for the weights:

$$b_i \leftarrow b_i + \eta[\langle v_i^0 \rangle_{data} - \langle v_i^n \rangle_{recon}] \tag{2.56}$$

and

$$c_j \leftarrow c_j + \eta[\langle h_j^0 \rangle_{data} - \langle h_j^n \rangle_{recon}] \tag{2.57}$$

where v_i, h_j, b_i, and c_j denote the i-th visible neuron, the j-th hidden neuron, the i-th visible bias, and the j-th hidden bias respectively.

As it can be anticipated from the fact that the direction of the gradient is not identical to the exact gradient, CD learning is known to be biased [Bengio, 2009; Carreira-Perpinan and Hinton, 2005]. Nevertheless, CD learning has been shown to work well in practice. A good property of CD is that if the data distribution is multi-modal, running the chains starting from each data sample guarantees that the samples approximating the negative phase are representative from different modes. Therefore, it has been formally demonstrated that the minimization of the CD is an approximation of the maximization of the data log-likelihood [Carreira-Perpinan and Hinton, 2005; Hinton, 2002].

A clear pseudo-code of the Contrastive Divergence learning algorithm is proposed in Algorithm 2. This pseudo-code is valid to train Gaussian-Bernoulli-RBM model, *i.e.* to train a RBM with Gaussian visible units and binary hidden units.

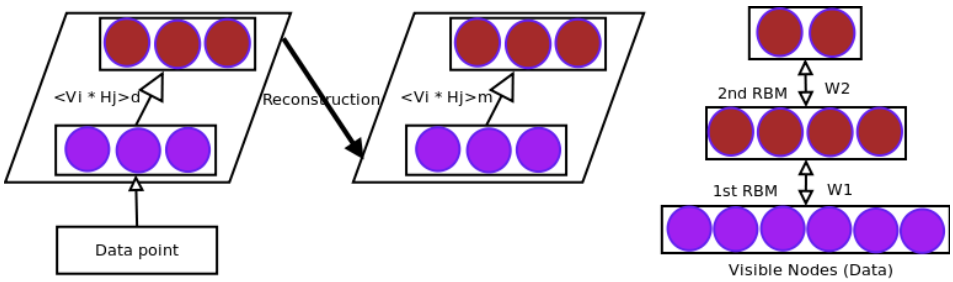

Figure 2.19: **Left**: Layer-wise training for a RBM with visible and hidden layers using contrastive divergence learning algorithm. **Right**: a deep belief network with two hidden layers.

2.4.5 Deep belief networks

DBNs are probabilistic generative models composed of multiple RBMs layers of latent stochastic variables. The latent variables typically have binary values. They corre-

input : random patch, v_0, selected from the database, learning rate, η the weights matrix, w_{ij}, of dimension (number of visible units, number of hidden units) and it is initialized with Gaussian distribution. b_i represents the visible biases, c_j represents the hidden biases, λ is the weight decay, μ is the momentum, σ is a unit variance set to 1, the training set is divided into small "minibatch-size (γ)" of 100, the weights increments, $w_{inc_{ij}}$, are set to zero, num_h represents the number of hidden units, num_v represents the number of visible units, and *epochs* represents the number of epochs need to ensure the convergence.

output: a set of features which are stored in (wij, b_i, c_j).

for $e = 1$ **to** *epochs* **do**
  for $j = 1$ **to** num_h **do**
    compute the binary hidden units, using: $h_0 = \sigma\left(c_j + \sum_i w_{ij} v_{i0}\right)$;
    sample the hidden states, using: $ph_0 \approx Bernoulli(h_0)$;
  end
  for $i = 1$ **to** num_v **do**
    compute the visible activation units, using: $v_1 = \sigma\left(b_i + \sum_i w_{ij} ph_{0j}\right)$;
    sample the visible states using Gaussian distribution as: $pv_1 \approx v_1 + \mathcal{N}(0, \sigma^2)$;
  end
  for $j = 1$ **to** num_h **do**
    compute the binary hidden units, using: $h_1 = \sigma\left(c_j + \sum_i w_{ij} pv_{i1}\right)$;
  end
  for $i = 1$ **to** num_v **do**
    for $j = 1$ **to** num_h **do**
      $w_{inc_{ij}} \leftarrow \mu * w_{inc_{ij}} + \eta * \left(\left((v_0 * h_0) - (pv_1 * h_1)\right)/\gamma\right) - \lambda * w_{ij}$
      $w_{ij} \leftarrow w_{ij} + w_{inc_{ij}}$
    end
    $b_i \leftarrow b_i + \eta * (v_0 - pv_1)$
  end
  for $j = 1$ **to** num_h **do**
    $c_j \leftarrow c_j + \eta * (h_0 - h_1)$
  end
end

Algorithm 2: Training update procedure for a RBM over Gaussian visible units and binomial hidden units using Contrastive Divergence.

spond to hidden units or feature detectors. The input variables are zero-mean Gaussian activation units and are often used to reconstruct the visible units. As shown in figure 2.20, the top two layers have undirected, symmetric connections between them and they form the weights or the features. These features are extracted using the principle of energy function minimization according to the quality of the image reconstruction.

DBNs are powerful unsupervised machine learning models for several reasons, including:

- There is an efficient, layer-by-layer procedure for learning the top-down, generative weights that determine how the variables in one layer depend on the variables in the layer above [Hinton, 2009].
- After learning, the values of the latent variables in every layer can be inferred by a single, bottom-up pass that starts with an observed data vector in the bottom layer and uses the generative weights in the reverse direction [Hinton, 2009].
- DBNs are able to extract sparse efficient features from a larger alphabet. These features can be successfully used to code huge amounts of images in an efficient way [Torralba et al., 2008].

Since DBNs are composed of RBMs layers (Figure 2.20), they can be trained in a greedy layer-wise way. We describe this training methodology in more detail in the next section.

2.4.6 Deep belief networks layer-wise training

Extensive works have empirically shown that DNNs training is a challenging task [Bengio et al., 2007; Erhan et al., 2009]. The authors of those works have suggested to use gradient-based methods for supervised DNNs training starting from a random initialization. However, this approach gets stuck in "apparent local minima" and the problem becomes much more complex with deep architectures [Bengio, 2009]. In 2006, Hinton [Hinton et al., 2006] has suggested that it would be more efficient if we train deep neural networks in a greedy layer-wise unsupervised learning way. By using this model we try to learn a hierarchical feature representation of which high level features are composed of simpler low level features.

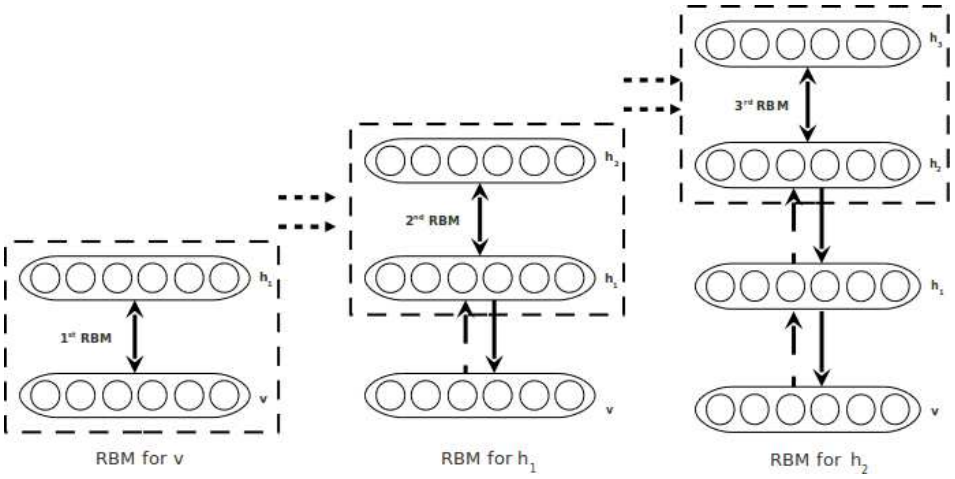

Figure 2.20: Stacking Restricted Boltzmann Machines (RBM) to achieve Deep Belief Network. This figure also illustrates the layer-wise training of a DBN.

As shown in figure 2.21, DBNs can thus be learned one layer at a time, by considering the values of the latent variables in one layer, after they have been inferred from the previous layer, as the data for training the next layer.

This efficient, greedy learning can be followed by, or combined with, other learning procedures that fine-tune all of the weights to improve the discriminative performance of the whole network.

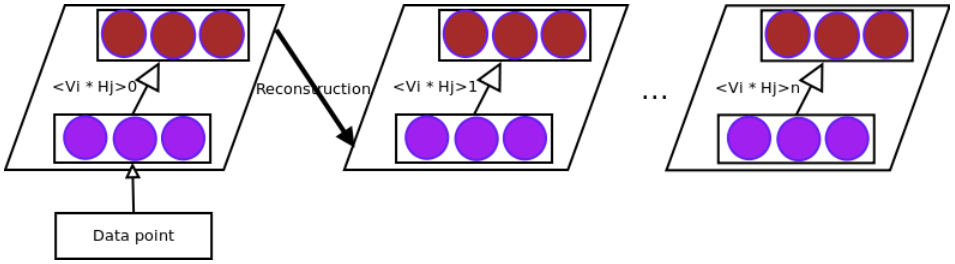

Figure 2.21: DBNs Layer-wise training with n hidden layers using CD and RBM.

More precisely, the first DBN model parameters $\theta = \{w_{ij}, b_i, c_j\}$, are learned by training the first RBM layer between the visible and hidden layers. Then, the model parameters are frozen and the conditional probabilities of the first hidden unit values are used to generate the data to train the higher RBM layer in the network. New layers can be stacked and trained using the same scenario. The process is repeated across the layers to obtain a sparse representation of the initial data that will be used as the final output. In sparse methods, the code is forced to have only few non-zero units while

most code units are zero most of the time. Eventually, sparse representations present several potential advantages, as demonstrated in a number of recent studies [Doi et al., 2006; Olshausen and Field, 1997; Ranzato et al., 2006]. They, in particular, have good robustness to noise, and provide a simple interpretation of the input data in terms of a small number of "parts" by extracting the structure hidden in the data. Furthermore, using high-dimensional representations increases the likelihood that image categories will be easily (possibly linearly) separable. Therefore, in this context, we assume that a sparse feature code increases the linear separability between the data in the feature space, which would facilitate the classification process.

2.5 Summary

In the present chapter we have discussed the problem of SPR and outlined the most significant difficulties to develop solutions for this problem. We started by introducing the differences between "metric localization", "topological localization", and "semantic place recognition". We also studied the SPR issues in terms of place recognition system designing and testing. We therefore reviewed different approaches for this problem, including in particular [Oliva and Torralba, 2006; Pronobis and Caputo, 2007; Torralba et al., 2003b; Ullah et al., 2008; Wu and Rehg, 2011; Wu et al., 2009]. These approaches have led to notable successes to achieve vision-based place recognition. However, they are based on complex or sophisticated techniques in order to achieve robust place recognition. In other words, performing the task of SPR using a simple classification method is still an open question. Therefore, SPR requires an appropriate code that allows fast and robust classification. That's why we have presented in this chapter a recent machine learning method, DBNs, as an alternative to hand-designed feature coding approaches. We hope that such approach is suitable for crating an appropriate representation for our classification problem.

Chapter 3

Feature space construction : a parameter study

3.1 Introduction

We have seen that most of the methods used for SPR solving are based on the use of hand-engineered features (GiST, CENTRIST, SURF, or SIFT descriptors). In order to cope with the continuous nature of the data representation, a discretization step using BoWs approaches and vector quantization is often applied. The used descriptors are low level and don't capture the structural organization of the scene. It has been shown that despite this lack of information, BoWs methods have given interesting results in SPR. However these methods are complex and depend on the quantization step. Their use is often followed by a complex phase of learning with sophisticated methods like SVMs.

Concerning the feature extraction, RBMs seem more appropriate since they take their root in theoretically grounded statistical methods (PCA and ICA) and they have shown to be efficient for image coding [Abdel-Rahman et al., 2011; Hinton, 2002; Hinton et al., 2006; Nair and Hinton, 2009; Salakhutdinov et al., 2007; Taylor et al., 2006; Torralba et al., 2008] in many applications. They also learn sparse edge filters, which are more suitable for classification and they are based on models of natural vision [Serre et al., 2007] which have impressive performances in object and scene recognition. Another attractive characteristic of this approach is that RBMs can be stacked to

form deep networks, the output of which could provide a high level non linear representation of the scene [Hinton et al., 2006] and thus capture spatial relationship lacking in the previous approaches.

We thus propose in the present work to code the images as a set of independent features obtained using DBNs [Hinton and Salakhutdinov, 2006; Torralba et al., 2008]. It has been shown that features extracted by DBNs are more promising for image classification than hand-engineered features [Hinton et al., 2011]. So, we hope that, due to the statistical independence of the features and their sparse nature, learning in the feature space will become linearly independent, greatly simplifying the way we will learn to classify the scenes.

One of the main question with the use of DBNs to classify an image set is to define the conditions required to build an optimal feature space. The general RBM training algorithm is governed by a lot of parameters acting on learning rate, sparsity of the final representation, locality of the obtained features and also speed of convergence. As stated by Hinton [Hinton, 2009] these parameters have to be set up carefully to obtain an appropriate feature extraction. In this chapter we report our own parametric study of the RBMs and DBNs that will be further used in our model. We study the effect of all these parameters on the feature obtained for image coding. In particular, we investigated the effect of the different parameters on the sparsity of the obtained coding and the locality of the features. These properties are indeed a major requirement for achieving good classification results.

3.2 Used databases

In this preliminary study, in order to compare easily our results with the ones published in the literature, we used two popular datasets, the van Hateren and the Berkeley image databases. The first one is a database of high-resolution calibrated monochrome images taken in defined illumination conditions [1], designed for various image processing tasks. It contains approximately 4000 images of $1536x1024$ pixels. The second one, the Berkeley database, is a collection of $481x321$ and $321x481$ natural images [2]. This database has been created to provide an empirical basis for image segmentation and

[1] van Hateren's Natural Image Database is available at: http://www.kyb.tuebingen.mpg.de/?id=227

[2] Berkeley database is available at: http://www.cs.berkeley.edu/projects/vision/grouping/segbench/

boundary detection researches [David et al., 2004]. It contains 300 different color images divided into a training set of 200 images, and a test set of 100 images. A subset of samples from this database is shown in figure 3.1.

Figure 3.1: A subset of 481x321 and 321x481 natural images selected from the Berkeley database [David et al., 2004].

As shown in figure 3.1, the images contain a lot of borders, corners, textures and edges because they are created for the purpose of image segmentation as earlier said. This kind of images would facilitate the process of feature extraction using a RBM. Moreover, they do not contain a lot of flat areas.

Two approaches can be considered for feature extraction. The first one consists in the extraction of small patches that can be used as inputs of a DBN. The second one consists in reducing the size of the whole images to an acceptable value in such a way that they can be used directly as the inputs of the network. This second approach will be considered in the next chapter.

For now, as proposed by [Ranzato et al., 2010], after gray-scale conversion, we have sampled $100,000$ random patches of size $16x16$ pixels from these databases. This extraction step was followed by a normalization that we are going to consider more in depth in the next section. After that, the obtained image patches were used as a training set for an RBM.

3.3 Normalization

3.3.1 Data whitening

Usually, natural images are highly structured and contain significant statistical redundancies, *i.e.* their pixels have strong correlations [Attneave, 1954; Barlow, 2001]. For example, it is well known that natural images bear considerable regularities in their first and second order statistics (spatial correlations), which can be measured using the autocorrelation function or the Fourier power spectral density [Field, 1987]. These correlations are due to the redundant nature of natural images (adjacent pixels usually have strong correlations except around edges). The presence of these correlations allows, for instance, image reconstruction using Markov Random Fields. It has thus been shown [Bell and Sejnowski, 1997; Field, 1987; Olshausen and Field, 1996] that the edges are the main characteristics of the natural images and that they are rather coded by higher order statistical dependencies. It can be deduced from this observation that the statistics of natural images is not Gaussian (since the moments greater than order-two are zero for Gaussian distributions). This statistics is dominated by rare events like contours, leading to high-kurtosis long-tailed distributions.

Pre-processing the images to remove these expected order-two correlations is known as whitening. It has been shown that whitening is a useful pre-processing strategy in Independent Component Analysis (ICA) [Hyvärinen and Oja, 2000; Soman et al., 2009]. It seems also a mandatory step for the use of clustering methods in object recognition [Coates et al., 2011]. Whitening being a linear process, it does not remove the higher order statistics or regularities present in the data. The theoretical grounding of whitening is simple: after centering, the data vectors are projected onto their principal axes (computed as the eigenvectors of the variance-covariance matrix) and then divided by

the variance along these axes. In this way, the data cloud is sphericized, letting appear only the usually non orthogonal axes corresponding to its higher-order statistical dependencies.

More formally, vectors of observations x are linearly transformed to obtain new vectors $(\tilde{x})$, which components are uncorrelated and which variances equals unity. In other words, the covariance matrix of the whitened data is equivalent to the identity matrix as follows:

$$E\{\tilde{x}\tilde{x}^T\} = I \tag{3.1}$$

There are several equivalent ways to perform this whitening transform. One popular way is the use of EigenValue Decomposition (EVD) of the covariance matrix as follows:

$$E\{\tilde{x}\tilde{x}^T\} = EDE^T \tag{3.2}$$

where E is the orthogonal matrix of eigenvectors of $E\{\tilde{x}\tilde{x}^T\}$ and D is the diagonal matrix of its eigenvalues, $D = diag(d_1, \ldots, d_n)$. Whitening can thus be achieved by:

$$x_{whiten} = \tilde{x} = ED^{-1/2}E^T x \tag{3.3}$$

where $D^{-1/2}$ is the inverse square root of the diagonal matrix which can be computed as follows:

$$D^{-1/2} = diag(d_1^{-1/2}, \ldots, d_n^{-1/2}) \tag{3.4}$$

Figure 3.2 (right) shows a subset of $16x16$ random whitened patches from the Berkeley database. Figure 3.2 (left) shows the corresponding original patches. It can be seen that after whitening, a lot of noise has been removed. Besides, the first and second statistical structures have also been removed. Thus, after data whitening, we assume that the final set of tiny-images is centered and whitened. Consequently the variance, σ^2 in equation 2.43 can be set to 1.

It is easy to see on the covariance matrix of a set of original patches from the van Hateren database (Figure 3.3) (first row) that pixels are strongly correlated to nearby pixels and weakly correlated to faraway pixels. These strong correlations can prevent

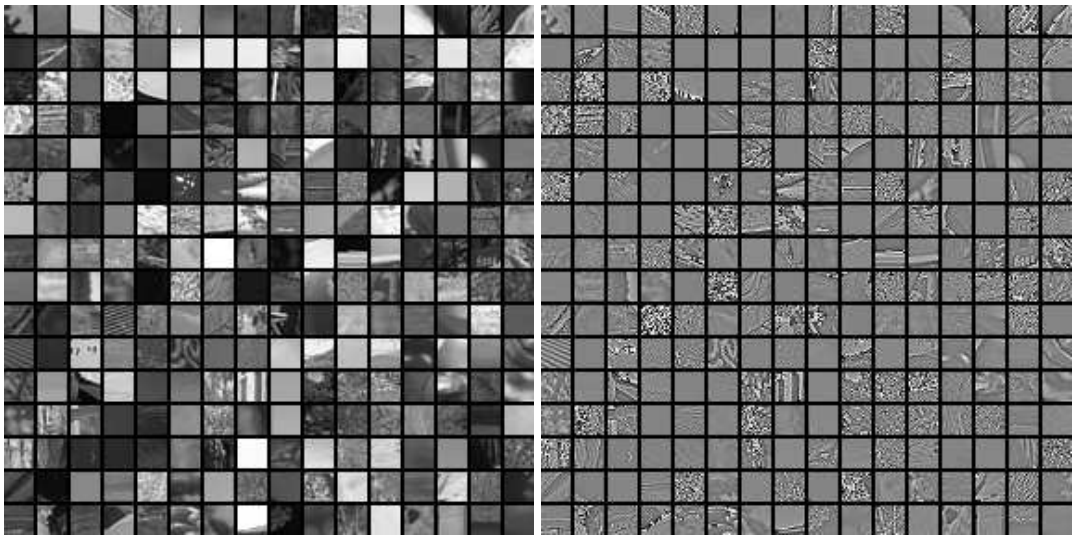

Figure 3.2: **Left:** A subset of 16x16 original random image patches sampled from Berkeley database. **Right:** The corresponding 16x16 whitened image patches are obtained after preprocessing.

the algorithms learning the feature space from rather focus on higher-order correlations. Instead it can force the model to get distracted by modeling order-two correlations. Thus if these correlations need to be eliminated before attempting to extract features, data whitening is required. Figure 3.3 (second row) shows the covariance matrix of the whitened patches for the same database. It shows that the whitened data became uncorrelated. However the higher order statistics corresponding to the difference between correlation and statistical dependence remains preserved. To build a set of statistically independent detectors, it will be required to find "factorial codes" such that the statistical distribution of the transformed data is as close as possible to the product of its components [Bell and Sejnowski, 1997; Olshausen and Field, 1996].

3.3.2 Local normalization

Another way to preprocess data is to perform local normalization. In this case, each patch $x^{(i)}$ is normalized by subtracting the mean and dividing by the standard deviation of its elements. For visual data, this corresponds to local brightness and contrast normalization. One can find in [Coates et al., 2011] a study of whitening and local normalization and their effect on a further classification task. However we can note

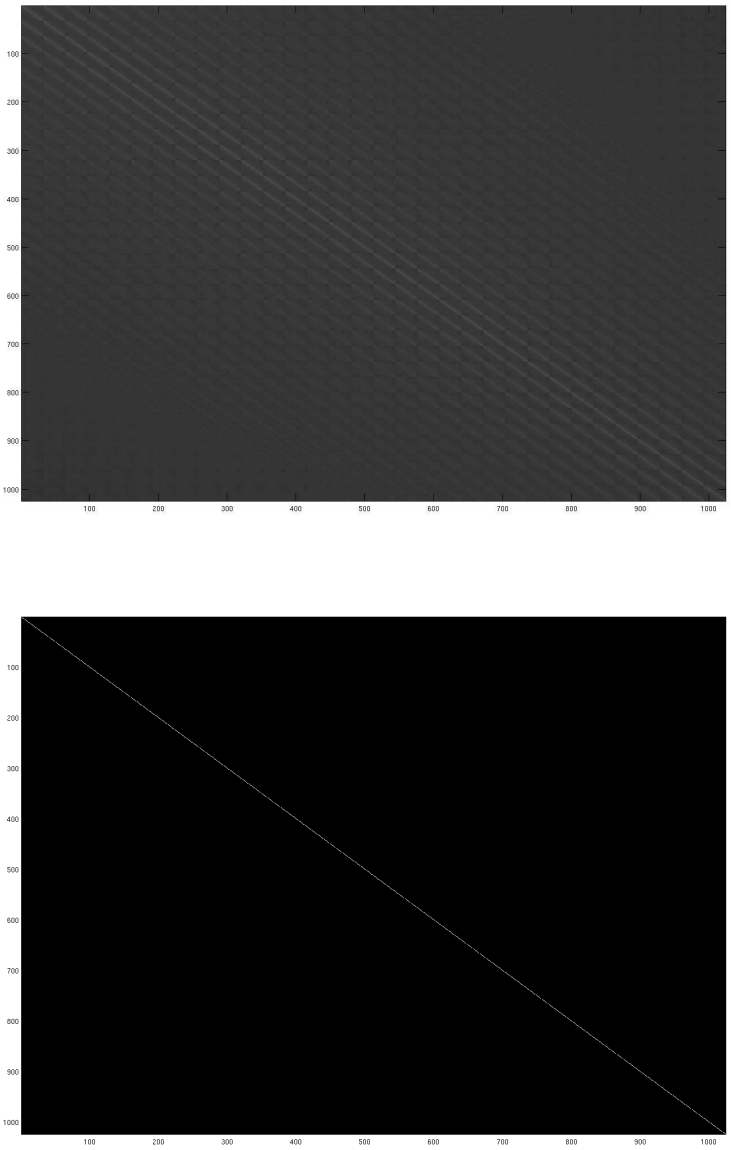

Figure 3.3: **First row:** The covariance matrix of the tiny images for the van Hateren database. White indicates high values, black indicates low values. All values are positive. The size of the tiny images is 32x32. **Second row:** The corresponding covariance matrix of the whitened tiny images from the same database.

that this study has been performed using two databases, NORB [1] and CIFAR [2], that

[1]NORB dataset : www.cs.nyu.edu/ ylclab/data/norb-V1.0/

[2]CIFAR dataset : www.cs.utoronto.ca/ kriz/cifar.html

have been especially designed for object recognition.

Figure 3.4 shows a dataset from the van Hateren database showing the effects on a initial dataset (left) of respectively local normalization (middle) and whitening (right).

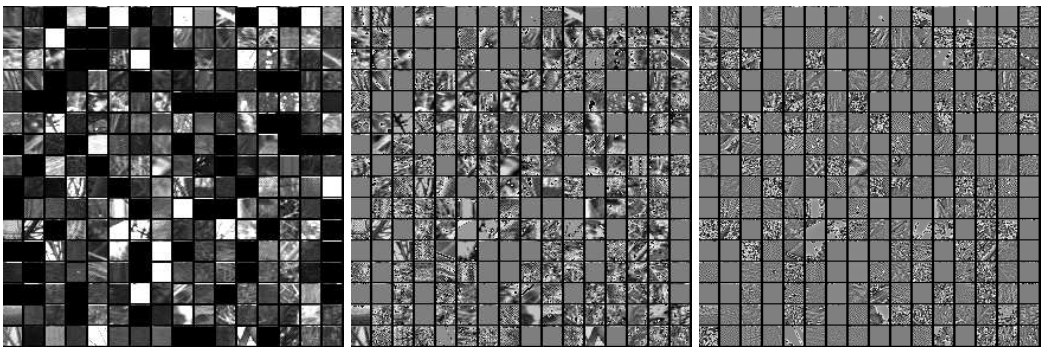

Figure 3.4: **First column:** 256 tiny images randomly sampled from the van Hateren database. **Second column:** The corresponding normalized ones. **Third column:** The corresponding whitened ones.

We can also note that in [Ranzato et al., 2010], the authors argue that whitening speeds-up the convergence of the algorithm without any justification. It could probably due to the fact that all variables have similar variances.

3.4 Unsupervised construction of the feature space

An RBM is usually trained as shown in figure 3.5, using the contrastive divergence learning procedure proposed by [Hinton, 2002].

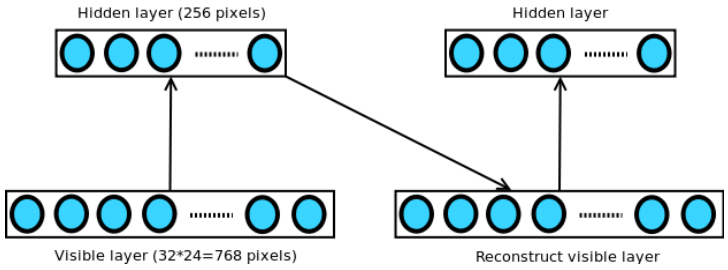

Figure 3.5: Training an RBM layer using contrastive divergence learning..

In order to present the general setup of the RBM training algorithm we will refer first on figure 3.5. A first set of weights[1] linking visible and hidden layers is taken

[1]Weights represent the symmetric interactions between the visible and hidden units which are known as features.

at random. Thus from one image it is possible to compute a first configuration of the hidden layer units probabilities. From configurations of the hidden layer drawn from the probabilities, the visible layer is reconstructed. We obtain a so-called "confabulation" of the input image. The contrastive divergence (CD) is then computed for this particular image for weights and biases. However, in a practical implementation, this CD is not directly used for weights and biases updating. The results for a set of images taken at random and called a mini-batch are pooled together and used to update the parameters. This process is repeated for a specific number of epochs or until convergence.

Unfortunately, training a RBM is known to be difficult. Recent researches have shown that without a careful choice of learning parameters, well suited to specific data sets and RBM structures, learning algorithms can easily fail to model the data distribution correctly [Desjardins et al., 2010; Fischer and Igel, 2010; Schulz et al., 2010]. This problem is often evidenced during learning. As illustrated in [Hinton, 2010], the update procedure of an RBM requires a certain amount of practical experience to decide how to set the values of different parameters, including the learning rate η, the initial and final momentums μ_i and μ_f, the weight-decay λ, the penalty term, the initial values of the weights, the size of the mini-batch γ, the number of epochs ε, the number of the hidden layers and the size of each hidden layer in order to learn the optimal features. In the next sections we will describe the most significant points concerning these parameters, which have been empirically investigated during our research work, to find out the optimal values. Note that a thorough description of these parameters and other questions can be found in [Hinton, 2010].

3.4.1 Overall organization of the network

RBM's were originally developed using binary visible and hidden units. However, other types of units such as Gaussian, binomial, and rectified linear ones can be used, for example see [Hinton, 2010; Krizhevsky, 2009; Lee et al., 2009; Norouzi et al., 2009]. The use of one of these unit types indeed depends on the problem to be solved. For instance, binary units with Bernouilli statistics work well in the case of handwritten digits, but they are not appropriate for multi-valued inputs like pixel levels. To deal with multi-valued data such as the pixel intensities in natural images, Hinton and

Salakhutdinov [Hinton and Salakhutdinov, 2006] replaced the binary visible units by linear units with independent Gaussian noise as first suggested by [Freund and Haussler, 1994]. This model has been successfully used in several other works such as [Lee et al., 2009; Norouzi et al., 2009; Torralba et al., 2008].

Within the framework of stochastic approaches, the first question raised by the model organization concerns the type of unit to use. We have shown that Bernouilli units are not well suited for image coding. We thus used Gaussian units for the input layer. However, for the upper layers of the network, binary Bernoulli units can be used. In this case, the outputs act as an indicator function indicating that a feature is selected or not for the construction of the internal representation of the image. The neural analogy corresponds to a neuron switched on in the presence of a specific feature in the visual field and off when the feature is absent. The feature is coded by the weights of the previous layer. It has been shown that Gaussian-Bernoulli RBMs are efficient for gray-scale images modeling, such as speech waves in [Jaitly and Hinton, 2011] and faces images in [Hinton and Salakhutdinov, 2006].

After preparing and pre-processing the different databases, we used a RBM with Gaussian visible and Bernoulli hidden units. However, training a Gaussian-Bernoulli RBM can be expensive, because we need a much greater number of weight updates than for an equivalent binary RBM. This problem becomes more difficult when the dimensionality of the input image is too large. To tackle this problem, in the present work we have followed the approach of [Nair and Hinton, 2008] by first training the first layer of the DBN as a Gaussian-Bernoulli RBM and then uses its hidden units as input to higher RBM layers. In other words, the higher layers of the DBNs use RBMs with Bernoulli visible and hidden units, as pereviously said. Note that the first model parameters are frozen and the conditional probabilities of the first hidden unit values are used to generate the data to train the higher RBM layers. This process is repeated several times across the RBM layers in order to obtain a sparse representation of the initial data which will be used as the final output.

As previously mentioned, we first tested the RBM algorithm on general purpose natural images and then apply it on images created for the purpose of robot localization. In this experiment, the structure of the first RBM layer is $256-256$. Figure 3.6 (left) shows 256 global filters of size $16x16$ pixels learned by training an RBM on $100,000$ whitened patches that are randomly sampled from Berkeley database. As we expected,

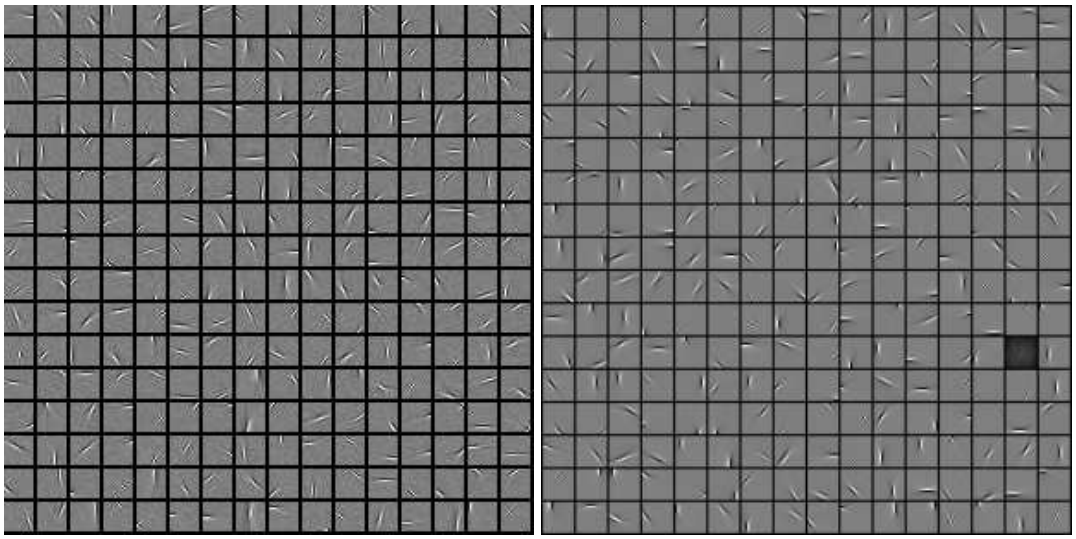

Figure 3.6: **Left:** 256 features learned by training the first RBM layer on whitened patches sampled from Berkeley dataset, these features are globally normalized. The training protocol is similar to the one proposed in [Ranzato et al., 2010] ($\varepsilon = 300$, $\gamma = 100$, $\eta = 0.02$, $\mu_i = 0.5$, $\mu_f = 0.9$, and $\lambda = 0.0002$). **Right:** 256 features obtained from the same database by [Ranzato et al., 2010].

these features look like band-pass oriented and localized edge detectors. The extracted features are quite similar to those obtained by [Ranzato et al., 2010] shown in the same figure (right), or to those extracted by Independent Component Analysis (ICA) models in [Hyvärinen et al., 2001], and to sparse coding algorithms in [Olshausen and Field, 1997; Teh et al., 2003].

A similar experiment was applied to the van Hateren database. In this experiment, the structure and the training protocol were similar to the Berkeley experiment [Ranzato et al., 2010]. Figure 3.7 shows 256 global filters of size $16x16$ pixels learned by training an RBM on $100,000$ whitened patches that are randomly sampled from the van Hateren database. These features are very close to the ones obtained from the Berkeley database.

3.4.2 The learning rate

The learning rate parameter, η, determines to what extent the newly acquired information will override the old information. Therefore, this parameter plays an important role for features extraction. In the gradient descent procedure, the learning rate is im-

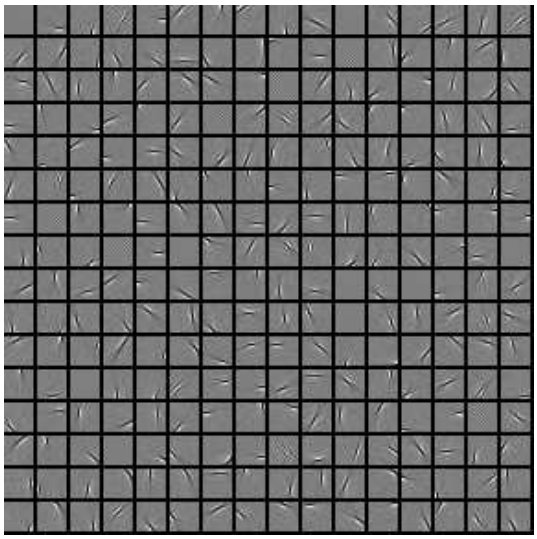

Figure 3.7: 256 features learned by training the first RBM layer on whitened patches sampled from the van Hateren's dataset, these features are globally normalized and obtained using the same training protocol as in the experiment of Berkeley database (see figure 3.6).

portant because some high and low values of η may suppress some of RBM's feature maps to become always inactive, and in fact dismiss some of features [Norouzi et al., 2009]. Thus, in this work we have tested a set of different values for η and selected the one giving the higher number of active features.

Practically, using large values of η (*e.g.* 0.05) yielded to increase the reconstruction error and thus the weights (features extracted from the Berkeley database) have completely exploded within the first few epochs, *i.e.* the RBM did not learn anything. Thus, using large values of η increase the oscillation problem during the adjustments of the weights [1]. However, when the value of η is reduced to 0.02, for instance, the network does not converge fast within the first few epochs as shown in figure 3.8 where some features are going to be extracted and appeared with more epochs [2].

On the other hand, using very small values of η will slow down the convergence process and a lot of epochs are thus required to reach the equilibrium. For example if we reduced the learning rate from 0.02 to 0.002, the extracted features for both cases was quite different, as shown in figure 3.9 after 200 epochs. This figure shows that

[1] Oscillation: the repetitive variation, typically in time, of some measure about a central value (often a point of equilibrium) or between two or more different states as in our case.

[2] One epoch means a complete iteration through all images in the dataset. Within an epoch we update the weights after presenting a mini-batch of size 100 selected at random from the training data.

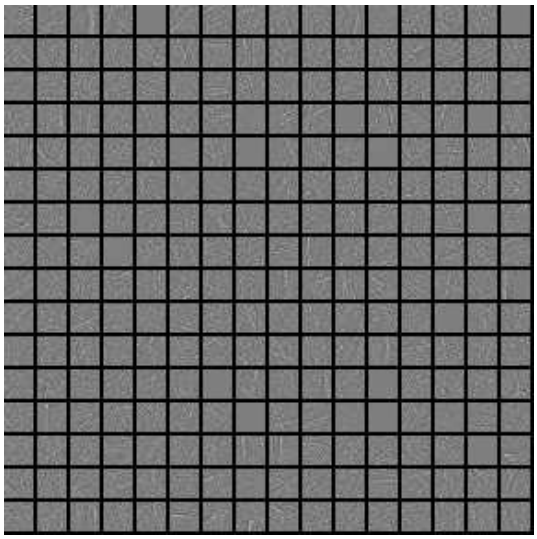

Figure 3.8: 256 features learned by training the first RBM layer on whitened patches from Berkeley dataset. These features are obtained using: $\eta = 0.02$, $\lambda = 0.0002$, $\gamma = 100$, $\varepsilon = 5$, and momentums.

the convergence has been almost achieved when the learning rate was 0.02, while a lot of additional epochs were still required to achieve the convergence for the other case when the learning rate was 0.002. Another important fact is that, once the convergence was almost achieved for both cases, the number of extracted features with a learning rate of 0.02 was greater than the number of extracted features when the learning rate was 0.002 (see figure 3.10), where the first case has converged after 300 epochs while the second case has converged after 1000 epochs.

A good way to set the learning rate parameter is to look at a histogram of weight updates and a histogram of the weights as illustrated in [Hinton, 2010]. The updates should be about 10^{-3} times the weights. This way, we can reset (increase or decrease) the value of the learning rate.

3.4.3 The size of the mini-batch

Although it is possible to update the weights after estimating the gradient on a single training case, Hinton [Hinton, 2010] and other researchers [Lee et al., 2009; Norouzi et al., 2009] have experimentally demonstrated that it is more efficient to divide the training set into small "mini-batches" of 10 to 100 images [1]. In this case, we adjust the

[1] A mini-batch is a collection of patches. It usually means the entire training set.

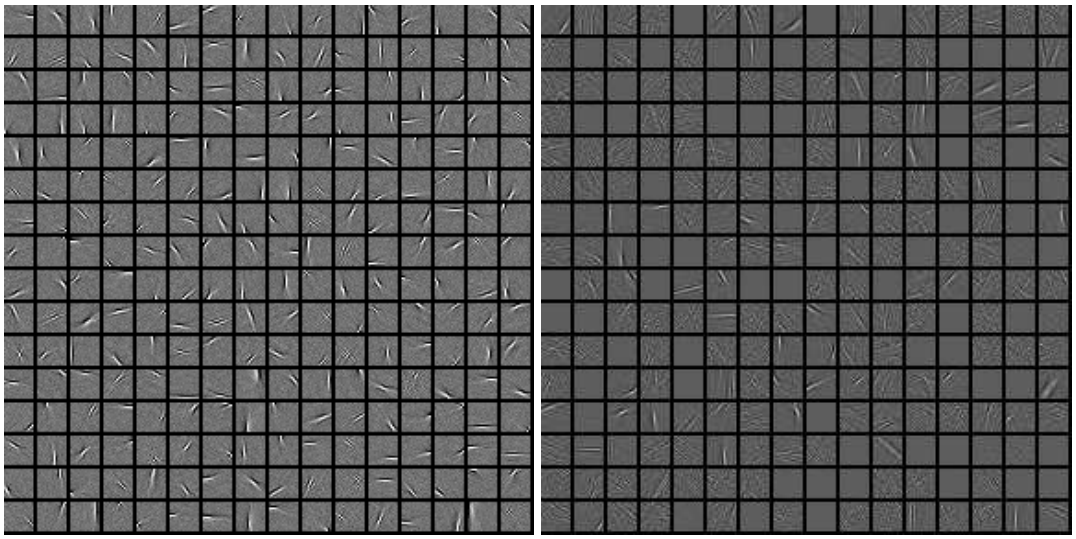

Figure 3.9: **Left:** 256 features learned by training the first RBM layer on 16x16 whitened patches sampled from Berkeley dataset. Same conditions as in figure 3.8, except $\varepsilon = 200$. **Right:** 256 features learned using the same conditions, except $\eta = 0.002$.

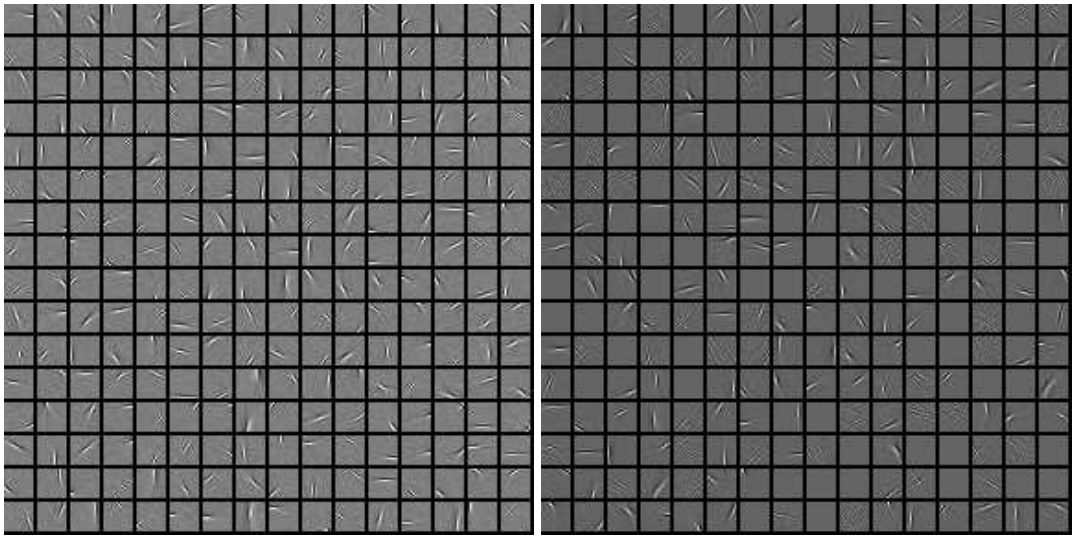

Figure 3.10: **Left:** 256 features learned by training the first RBM layer on 16x16 whitened patches sampled from Berkeley dataset. These features are also obtained using the same conditions as in figure 3.8, except $\varepsilon = 300$. **Right:** A similar subset of filters leaned using the same conditions, except $\eta = 0.002$ and $\varepsilon = 1000$.

weights after estimating the total gradient of the mini-batch, which means that we need to divide the total gradient computed on a mini-batch by the size of the mini-batch (γ),

and thus the weights update procedure in equation 2.54 can be re-written as follows:

$$w_{ij} \leftarrow w_{ij} + \eta * (\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle)/\gamma \tag{3.5}$$

Usually, the ideal mini-batch size is equal to the number of different classes. Each mini-batch should contain one example of each class to reduce the sampling error while estimating the gradient for the whole training set from a single mini-batch [Hinton, 2010]. In this work, we have tried many different values of γ to examine its final impact on the feature extraction. It is noticed that when we use a large mini-batch size, for instance 200, the extracted features are not so different from those obtained using a mini-batch of 100, as shown in both experiments, figure 3.11. The total number of features is slightly higher when $\gamma = 100$, while when we use a small mini-batch, for instance 10, the RBM algorithm does not learn interesting features after hundred of epochs as shown in figure 3.12. Moreover, the convergence was not improved by an increased number of epochs (*e.g.* from 200 to 400 epochs) as illustrated in the same figure.

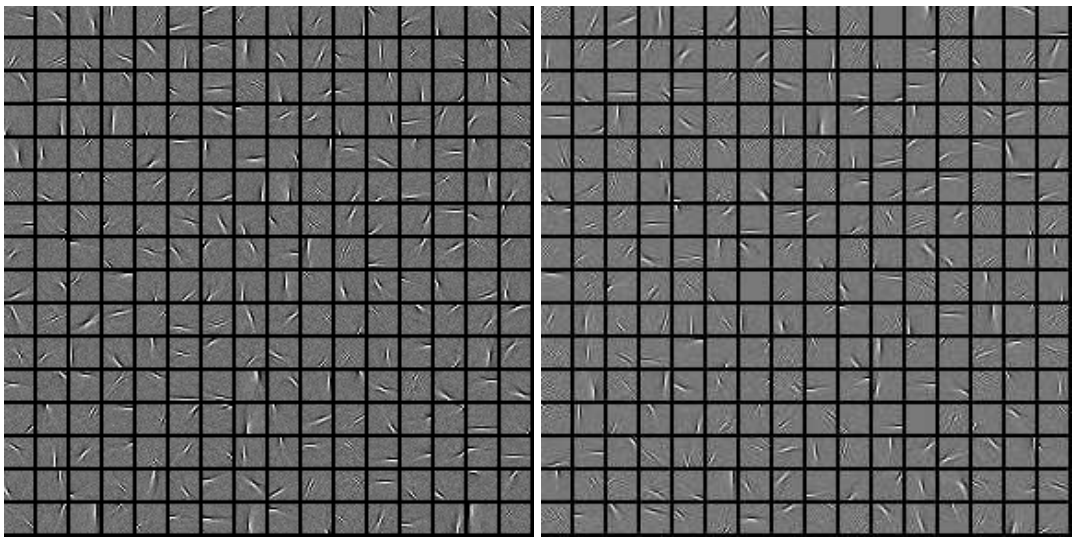

Figure 3.11: **Left:** 256 features learned by training the first RBM layer on 16x16 whitened patches sampled from Berkeley dataset. Similar conditions as in figure 3.9 (left) are used for this experiment. **Right:** 256 filters extracted using the same conditions, except $\gamma = 200$.

Therefore, the size of the mini-batch also plays an important role in the extraction of features. Furthermore, it has been shown that this process speeds-up the training process [Hinton, 2010; Salakhutdinov et al., 2007].

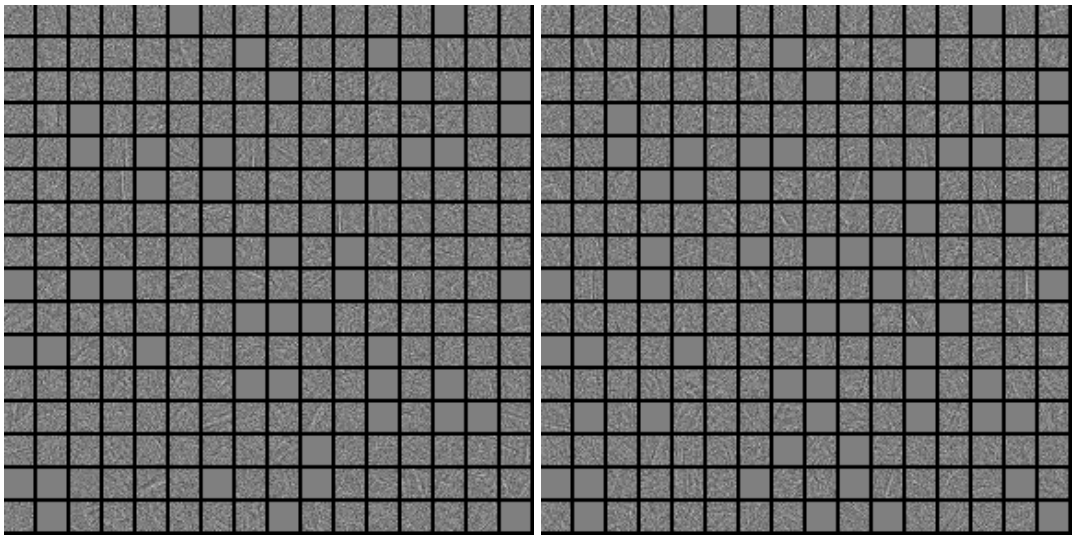

Figure 3.12: **Left:** 256 features learned by training the first RBM layer on $16x16$ whitened patches sampled from Berkeley dataset. Also, the same conditions as in figure 3.9 (left) are used for this experiment, except $\gamma = 10$. **Right:** The same filters, but with $\varepsilon = 400$ epochs.

3.4.4 The initial values of weights and biases

Several ways can be used to initialize the weights. A possible one is to use small random values chosen from a zero-mean Gaussian distribution with a standard deviation, σ, of 0.01. Another way is to use small random values chosen from a uniform distribution. In fact, using larger random values will force the network to converge faster, which might lead to a worse final model.

Since we have proposed to use a Gaussian-Bernoulli RBM model, we have initialized the weights matrix using a Gaussian distribution as follows:

$$w_{ij} \sim \mathcal{N}(\mu, \sigma^2) \tag{3.6}$$

where μ represents the mean and σ^2 represents the variance of the Gaussian distribution.

In this work, we have tested different values of the standard deviation, σ (for instance see figure 3.13). We can see that the convergence with high values of σ is slightly faster than with smaller values of σ. If we compare the extracted features shown in figure 3.13 with each other or even with the features shown in figure 3.10 (left)(obtained using $\sigma = 0.01$, they seem quite similar. Thus, in order to be sure that

the network leads to a good final model, it is more appropriate to use a small value of σ like 0.01, as recommended in [Hinton, 2010].

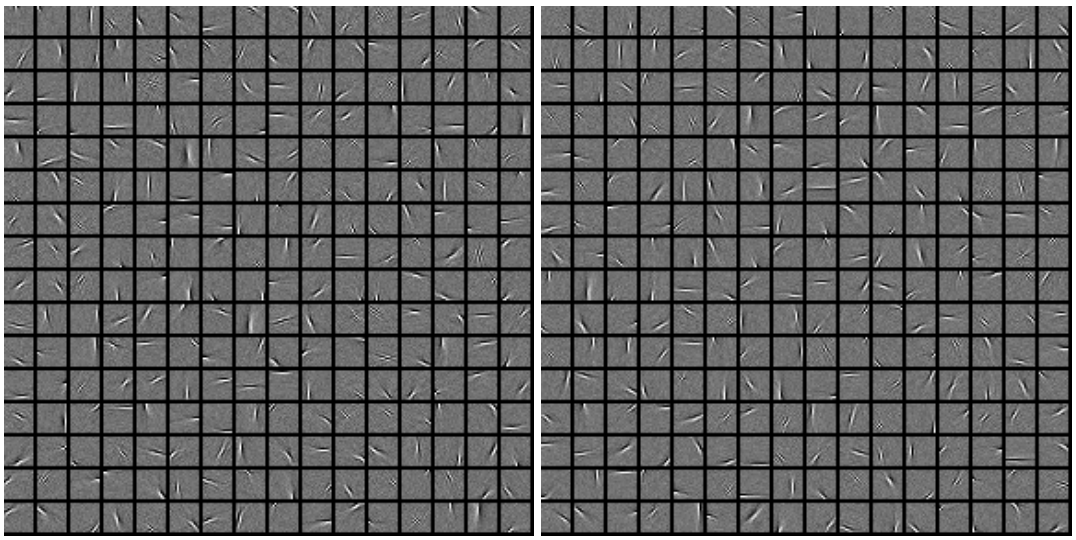

Figure 3.13: **Left:** 256 features learned by training the first RBM layer on $16x16$ whitened patches sampled from Berkeley dataset. Similar conditions as in figure 3.10 (left) are used, however, $\sigma = 0.5$. **Right:** 256 features learned using the same conditions, except $\sigma = 0.1$.

Concerning the initial values of visible and hidden biases, one possible way is to initialize them all to zero. However, to encourage the sparsity, Hinton [Hinton, 2010] recommended to start the hidden biases with very large negative values of about -4 as we will see later (section 3.4.9).

3.4.5 Momentum

Momentum (μ) is a standard parameter used in many neural network applications. It is added to speed-up the learning process of a RBM, smoothen the gradients and avoid getting stuck in local minima. So that, this term, together with the learning rate, controls the weights update and yield a modified update rule:

$$w_{ij} \leftarrow \mu * w_{ij} + \eta * (\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle)/\gamma \tag{3.7}$$

where the momentum term must be between 0 and 1. More precisely, Hinton and other researchers have recommended to start with a momentum of 0.5 for the first several

epochs (for example, the first five epochs) and a momentum of 0.9 for the rest of epochs [Hinton, 2010; Lee et al., 2008; Norouzi et al., 2009].

In fact, we have also tested different values of this factor. For instance, if we ignore the effect of this term by using a momentum of 1, the network converges towards a wrong solution as shown in figure 3.14. Because when μ is equal to 1, the effect of the previous weights (the Left-Hand Side of equation 3.7) will be much higher than the effect of the gradient (the Right-Hand Side of equation 3.7) since we use a small learning rate. It means that using high values of momentum also lead to increase the oscillation in the weights adjustments as in the case of learning rate. Therefore, using a smaller momentum, like 0.5, for the first several epochs, decreases the effect of the initial weights and let the gradient make more influence on the network to converge to the right solution. After 5 epochs the initial weights become more compatible with the model (they fit the model) and, in this case, it is possible to increase the momentum to 0.9, for example, in order to increase the network convergence speed. This way to proceed ensures the stability of the learning process.

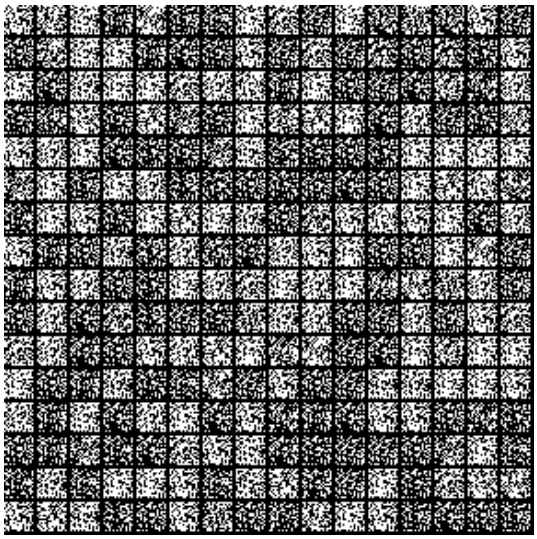

Figure 3.14: 256 features learned by the first RBM layer on 16x16 whitened patches sampled from Berkeley dataset. Similar conditions were used in this experiment as in figure 3.10 (left), except $\mu_i = 1$ and $\varepsilon = 10$.

3.4.6 Weight decay

Weight-decay (or weight-cost (λ)) is another important factor added to the normal gradient as shown in equation 3.8. This regularization term penalizes large parameter values, such as weights, which sometimes happens during the learning process. This yields the following update rule:

$$w_{ij} \leftarrow \mu * w_{ij} + \eta * \left[\left(\left(\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle \right) / \gamma \right) - \lambda * w_{ij} \right] \tag{3.8}$$

In general, the values of λ should be typically ranged from 0.01 to 0.00001. Similarly to previous factors, different values of (λ) have been investigated in this work. For instance, see the extracted features shown in figure 3.15 using a weight-decay of 0.02 and 0.002 respectively. This figure obviously demonstrates two facts: firstly, when we use a large value of λ the RBM learns flat features. This is due to the fact that the impact of the weight-cost becomes much higher than the impact of the gradient descent and thus all weights reach very large negative values. Secondly, when we use a smaller value, a set of features have emerged. Therefore, a large value of λ forces the network to have less effect of the gradient descent. In other words, we need to select a value of λ, so that we must make a balance between the two expressions (*i.e.* $(\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle)$ and $(\lambda * w_{ij})$) of equation 3.8.

It is important to multiply the derivative of the weight-decay by the learning rate. Otherwise, changes in learning rate change the function that is being optimized rather than just changing the optimization procedure [Hinton, 2010]. Weight-decay is typically not applied to the hidden and visible biases because they are less likely to cause overfitting [1]. Also, the biases sometimes need to be quite large.

There are other forms of the weight-decay that can be used to achieve sparsity and locality in the features. One of them called "$L2_{norm}$" which is half of the sum of the squared weights times a coefficient which is called the weight-decay. Another form called "$L1_{norm}$" which is the use of the derivative of the sum of the absolute values of the weights. $L1_{norm}$ weight-decay often leads to strongly localized receptive fields, because it causes many of the weights to become exactly zero whilst allowing few of

[1] Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship. Overfitting generally occurs when a model is excessively complex, *e.g.* a model has too many parameters with respect to the number of observations.

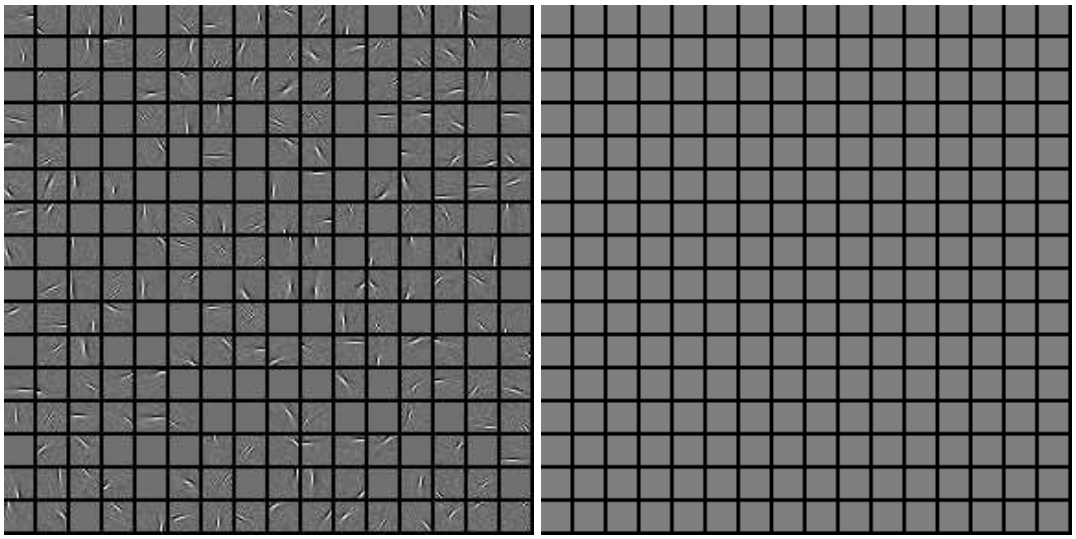

Figure 3.15: **Left:** 256 features learned by training the first RBM layer on $16x16$ whitened patches sampled from Berkeley dataset. Similar conditions as in figure 3.10 (left) are used, except $\lambda = 0.002$. **Right:** 256 filters learned using the same conditions, except $\lambda = 0.02$.

them to increase significantly. Moreover, Hinton has mentioned four different reasons to use these weight-cost terms in an RBM [Hinton, 2010]:

1. It improves the generalization to new data by reducing overfitting to the training data.

2. It makes the receptive fields of the hidden units smoother and more interpretable by shrinking useless weights.

3. It increases the sparsity for the extracted features.

4. It improves the mixing rate of the alternating Gibbs Markov chain. So, with small weights, the Markov chain mixes more rapidly.

3.4.7 Penalty term

A popular way to minimize the information content in the code is to make it sparse or low-dimensional. In this context, a vector is called sparse if it contains only a minimum number of active (non-zero) units. Sparsity estimation (or sparse recovery) is playing an increasingly important role in statistics and machine learning communities, because

it simplifies further tasks like the classification process and it is a useful way to reduce the dimensionality of data with neural networks [Hinton and Salakhutdinov, 2006].

Usually the model parameters are trained through the maximization of the log-likelihood of the reconstructed data. This maximization problem corresponds to learning w_{ij}, b_i, and c_j to minimize the energy of states drawn from the data distribution. In the meantime, the hidden unit activations have to be sparse. To achieve that goal, it is possible to apply "$L1_{norm}$" regularization, however, it is expensive because the Gaussian-Bernoulli RBM representation uses stochastic binary variables. Hence, several other methods have recently been developed [Lee et al., 2008, 2009; Mairal et al., 2008; Olshausen and Field, 1996, 1997] to achieve the sparsity goal for RBMs. Their methods rely on adding a penalty or regularization term to improve the sparsity of the data representation. For instance, [Lee et al., 2008] proposed to couple the maximum likelihood of contrastive divergence (CD) with a regularization term that penalizes non-selective units. Similarly, [Nair and Hinton, 2009] used the cross-entropy measure between the actual and desired distributions to compute the penalty. In more details, in the latter method [Nair and Hinton, 2009], the additional update is a penalty proportional to $q-p$, where p is the "sparsity target", which represents the desired probability that a unit is active, and q is the penalty term which encourages the actual probability of being active. As illustrated in [Hinton, 2010], the sparsity of a hidden unit is therefore computed by a process of averaging its activation across training as follows:

$$q_t = \lambda * q_{t-1} + (1-\lambda) * q_{current} \tag{3.9}$$

where λ represents the decay-rate which can be between 0.9 and 0.99, $q_{current}$ is the average hidden activation probability that a unit is active for the current mini-batch.

The cross entropy between the desired and actual activation distributions is used as a penalty measure as follows:

$$cost = -plogq - (1-p)log(1-q) \tag{3.10}$$

where the sparsity target, p, can be between 0.01 and 0.1.

This has the derivative of $q-p$ and is scaled by a meta-parameter called "sparsity-cost". So, sparsity adds three meta-parameters to the model which are: the sparsity

target p, the decay rate λ, and the sparsity cost, $cost$. When we add the sparsity penalty to the learning rule, the new weight update formula becomes:

$$w_{ij} \leftarrow \mu * w_{ij} + \eta * \left[\left((\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle)/\gamma \right) - cost * (p_t - q) \right] \tag{3.11}$$

Using this regularization term, we have obtained the features shown in figure 3.16. They are more localized than the features shown in figure 3.10 (left). It turns out that adding the penalty term has successfully encouraged to achieve sparse activities for the hidden units. Figure 3.17 shows the 128 unit outputs of the corresponding network for 300 samples of the Berkeley database. It shows graphically that most of the input images are represented by a few active units (spatial sparsity) and that each unit is rarely active over samples (temporal sparsity).

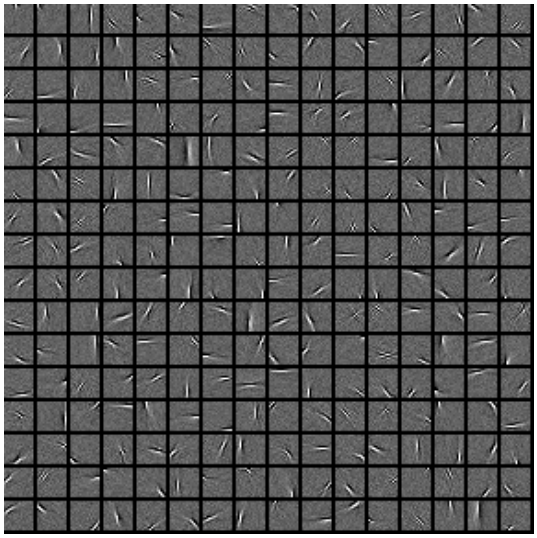

Figure 3.16: 256 features learned by training the first RBM layer on $16x16$ whitened patches sampled from Berkeley dataset. Similar conditions as in figure 3.10 (left) are also used, except the penalty term was included. Thus, we used $p = 0.02$ and $\lambda = 0.99$.

The other method proposed in [Lee et al., 2008] and mentioned above introduces a regularizer term that makes the average hidden variable activation low over the entire training examples. Thus the activations of the model neurons become also sparse. In fact, this method is similar to the one used in other models [Olshausen and Field, 1996]. Thus, as illustrated in [Lee et al., 2008], given a training set $\{v^{(1)}, \ldots, v^{(m)}\}$

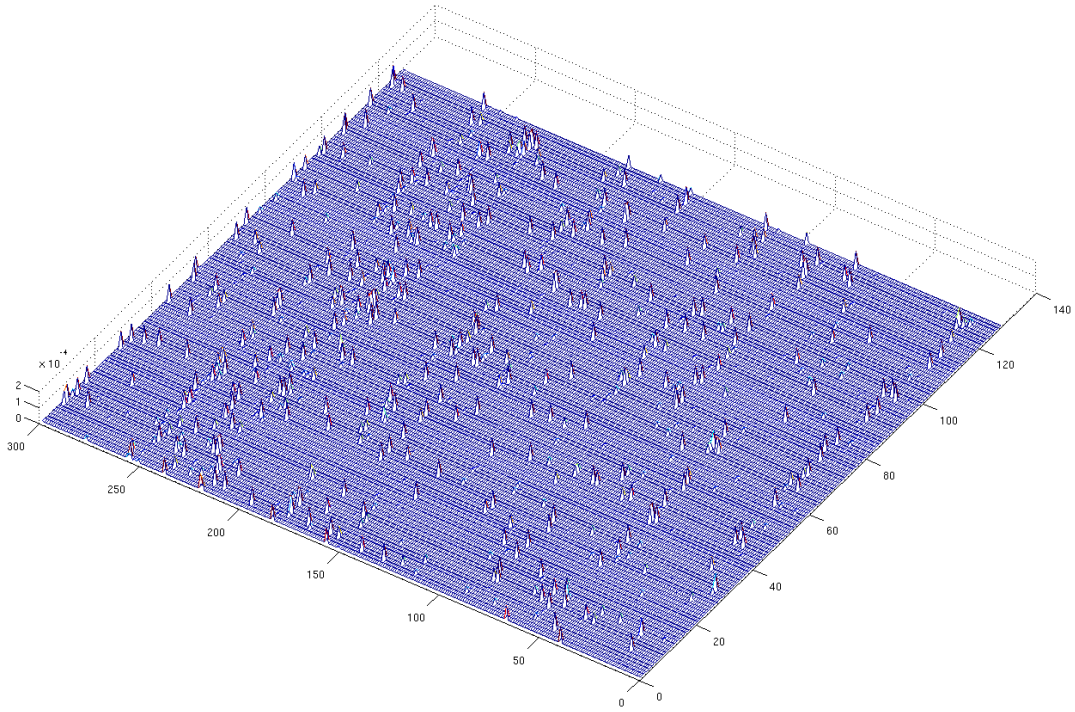

Figure 3.17: An exemple of the spatial and temporal sparsities achieved using a sparse network obtained from the Berkeley database. Same parameter settings as in figure 3.16. λ for both weight and hidden biases was set to 0.015.

including m examples, we pose the following optimization problem:

$$minimize_{\{w_{ij},b_i,c_j\}} - \sum_{l=1}^{m} log\left(\sum_{h} P(\mathbf{v}^{(l)},\mathbf{h}^{(l)})\right) + \lambda \sum_{j=1}^{n} \left| p - \frac{1}{m}\sum_{l=1}^{m} \mathbb{E}[h_j^{(l)}|\mathbf{v}^{(l)}] \right|^2 \quad (3.12)$$

where $\mathbb{E}[.]$ is the conditional expectation given the data, once again p is the sparsity target controlling the sparseness of the hidden units h_j, and λ is the sparsity cost. Thus, after involving this regularization in the CD learning algorithm, the gradient of the sparsity regularization term over the parameters (weights w_{ij} and the hidden biases

c_j) can be written as follows:

$$w_{ij} \leftarrow \mu * w_{ij} + \eta * \left[\left(\left(\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle \right) / \gamma \right) - \lambda * (p - \frac{1}{m} \sum_{l=1}^{m} p_j^{(l)} \right) \tag{3.13}$$

$$c_j \leftarrow c_j + \eta [\langle h_j^0 \rangle_{data} - \langle h_j^n \rangle_{recon}] - \lambda * (p - \frac{1}{m} \sum_{l=1}^{m} p_j^{(l)}) \tag{3.14}$$

where m, in this case,represents the size of the mini-batch and $p_j^{(l)} \triangleq \sigma(\sum_i v_i^{(l)} w_{ij} + c_j)$.

It has been shown that the sparse RBM learning algorithm can capture interesting high-order features from natural image statistics [Lee et al., 2008]. The hope is that such a learning algorithm remains capable to capture higher-order features from various databases, such as a database created for the purpose of robot localization.

The previous experiments have been achieved using the conventional Gaussian-Bernoulli RBM learning algorithm, where the question of sparsity has been included in this algorithm using a penalty term as illustrated earlier. We have also implemented the sparse RBM which is developed in [Lee et al., 2008]. However, we have not observed a big difference in features extraction using both learning algorithms (see for instance figure 3.18). This similarity of features for locality and sparsity confirms that the penalty term in the regular RBM and the sparsity target in the sparse RBM can capture sparse codes from the initial images. Therefore, using any of them leads to extract sparse features.

3.4.8 The number of hidden units

Usually, three different situations are distinguished in the literature for the size of the hidden layer : less than the size of the input layer (under-completeness), equal (completeness) or greater (over-completeness). These situations can be related to the feature set of ICA that is, by construction, always complete. The question of using over-complete feature sets has been first raised in [Olshausen and Field, 1997]. It seems that in the visual system, working with an over-complete feature alphabet makes the coding more adaptive to natural images variability. Thus, for a particular image, the final coding is more precise, even if the over-completeness reintroduces a kind of redundancy.

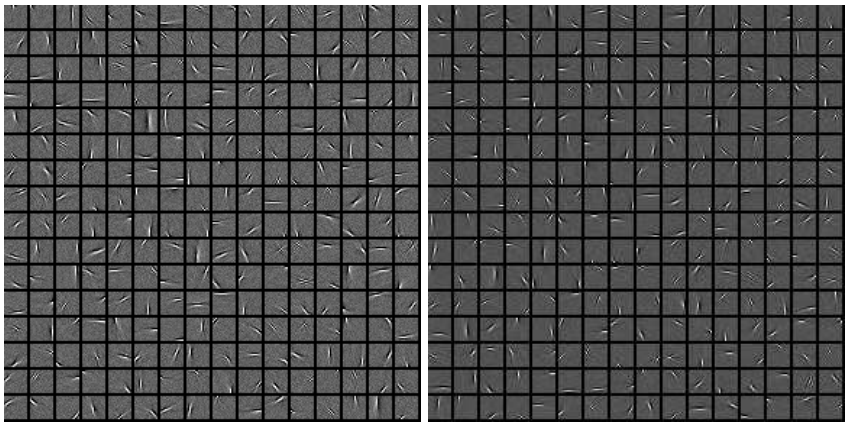

Figure 3.18: **Left:** 256 features learned by training the regular RBM on whitened image patches ($16x16$) sampled from Berkeley dataset, these filters are obtained using the same conditions as in figure 3.16. **Right:** The corresponding features learned by training the sparse RBM on the same whitened image patches. In this case, we also used the same conditions, except $p = 0.02$ and $\lambda = 0.02$.

In fact, the optimal size of the hidden layer depends on several things: the size of the visible layer, the target to achieve, and the redundancy of the patches in the training set. More precisely, the size of the hidden layer should be compatible with the size of the visible layer. However, if the sparsity target is very small, more hidden units could be used. Contrarily, if the training cases are highly redundant, as they typically will be for very big training sets, fewer units are necessary [Hinton, 2010] for two reasons: first, it may be quite reasonable to use less output units if in $100,000$ training images each image is repeated for instance $1,000$ times. Otherwise it's time-consuming. Second, when you feed similar images to the network, the use of many parameters would not change the results as shown in figure 3.19.

In this work, since our training set is highly redundant, the size of the first hidden layer could either be equivalent to the size of the visible layer or reduced to half (more or less) of the visible layer size. The same scenario can be used for higher hidden layers. This proposition has been investigated using different sizes of the hidden layer. For instance, the features shown in figure 3.9 (left) were obtained using an equivalent visible and hidden layers size ($256-256$ units), while figure 3.19 (top) shows features obtained using an under-complete hidden layer ($256-128$ units) and (bottom) shows features obtained using an over-complete hidden layer ($256-512$ units). These experiments show that when we reduce the size of the hidden layer, the network extracts

similar features, but the matrix is fully used, while when we increase the size of the hidden layer, a lot of the features were flat. However, these facts cannot be generalized to other databases.

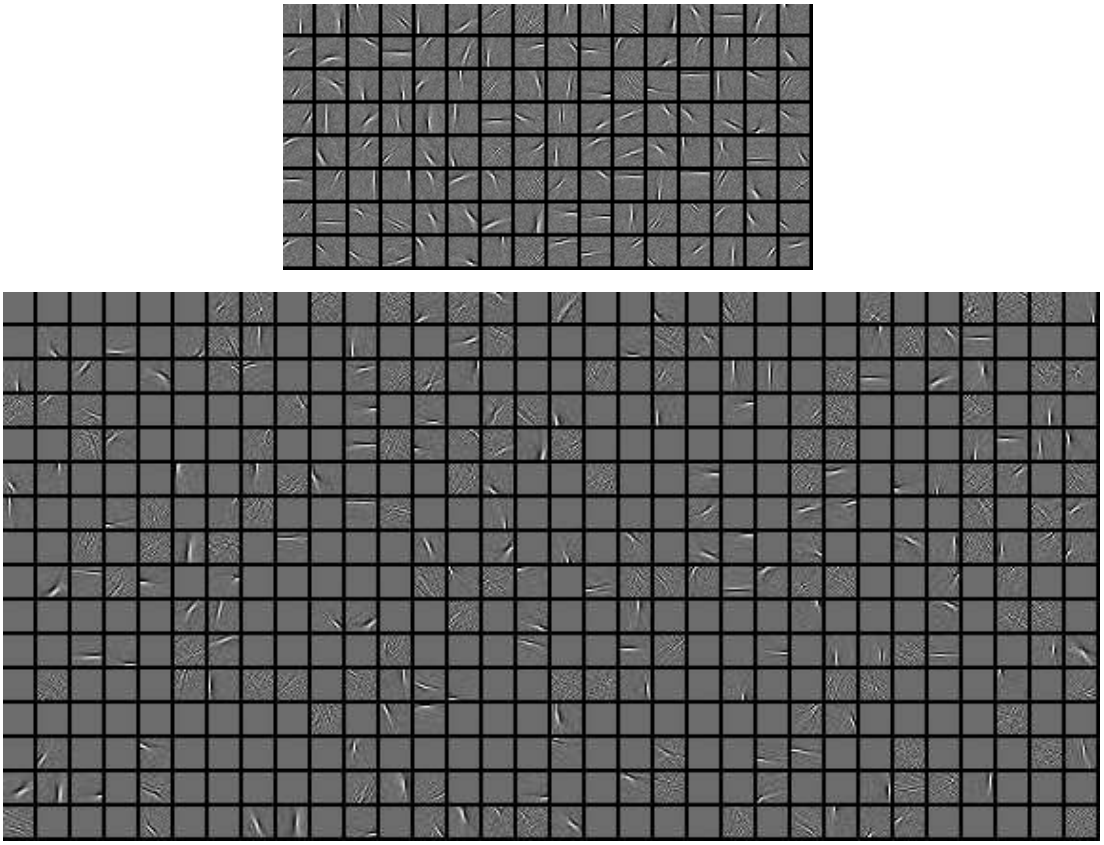

Figure 3.19: **Top:** 128 features learned by training the first RBM layer on 16x16 whitened patches sampled from Berkeley dataset. Same conditions were used as in figure 3.10 (left). **Bottom:** 512 features learned using the same conditions.

3.4.9 Effect of normalization on the feature space

In the previous sections, we have seen that several parameters (for example the sparsity term, the number of hidden units, the mini-batch size, *etc.*) play an important role in obtaining interesting features. In this section we investigate the effect of whitening and normalization on the detection of features using a RBM learning algorithm. These factors, orthogonal to the learning algorithm itself, can have a large impact on

performances.

For this task, we have conducted extensive experiments using datasets of random patches sampled from van Hateren and Berkeley natural image databases respectively. The random patches extracted from these databases were normalized or whitened in two separate pre-processes. In both experiments, an over-complete structure ($256-512$) of the first RBM layer was used.

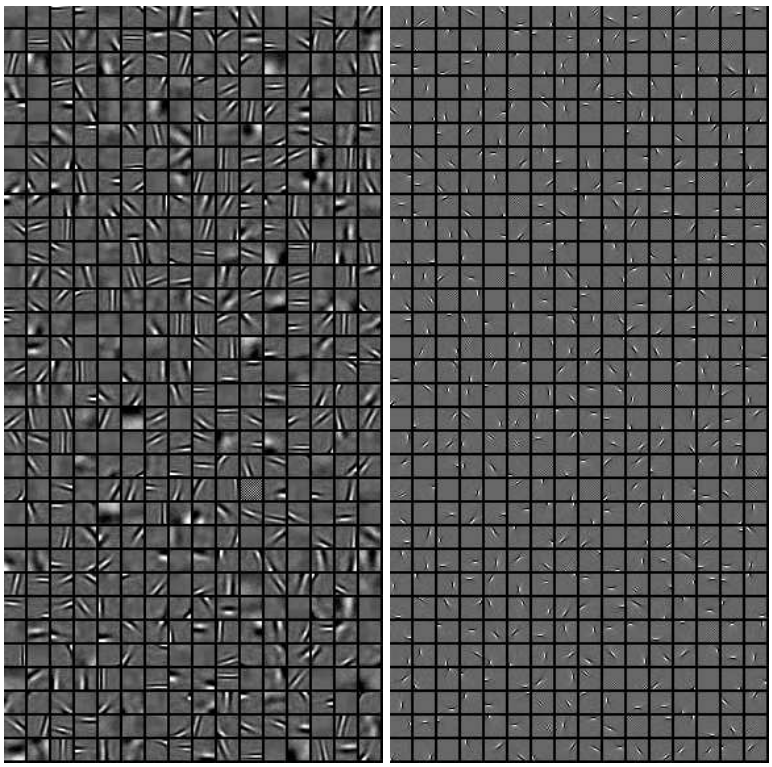

Figure 3.20: Learned over-complete natural image bases. **Left:** 512 features learned by training the first RBM layer on normalized image patches ($16x16$) sampled from van Hateren's dataset. **Right:** The corresponding features learned by training the first RBM layer on whitened image patches ($16x16$) sampled from the same database. For both experiments, the training protocol is similar to the one proposed in [Lee et al., 2008] ($\varepsilon = 300$, $\gamma = 200$, $\eta = 0.02$, $\mu_i = 0.5$, $\mu_f = 0.9$, $p = 0.02$, $\lambda = 0.02$).

Figure 3.20 (left) shows features extracted using the locally normalized data, while figure 3.20 (right) shows features extracted using the whitened one. It is obvious that the features extracted from the whitened data are more localized. Data whitening clearly changes the characteristics of the learned bases. One explanation could be that the second order correlations are linked to the presence of low frequencies in the

images. If the whitening algorithm removes these correlations in the original data set, it leads to whitened data covering only high spatial frequencies. The RBM algorithm in this case finds only high frequency features.

It is also obvious that the features obtained from the whitened dataset are more localized compared with the previous experiments that were also conducted using the whitened data. This is due to the fact that the hidden biases, in this case, are initialized with very large negative values of -4.

We have also applied our learning algorithm on the Berkeley database. Similarly, we have created two datasets (normalized and whitened data) from this database. Each dataset contains $100,000$ of $16x16$ random patches. The features shown in figure 3.21 are quite similar to those obtained from the van Hateren database. However, the number of features extracted from van Hateren database is more than the number of features learned from the Berkeley one. It could mean that the natural images, in the case of van Hateren database, contain more interesting structures, edges, and orientation contours.

Several other experiments have been conducted using both complete and under-complete RBM structures. The main goal for doing these experiments is to demonstrate that an under-complete RBM structure can also capture interesting high-order features using the locally normalized data.

Two experiments with a complete structure $(256 - 256)$ have been conducted for van Hateren and Berkeley databases. As shown in figure 3.22, the RBM learning algorithm extracts features similar to those obtained using an over-complete structure. They still cover a larger range of spatial frequencies. There seems to be no difference in the shape of the features from the two databases. However the number of almost flat features (features that have not converged or that will take a very long time to appear) is greater for the Berkeley database.

Similarly, with an under-complete structure, as shown in figure 3.23, the learning algorithm remains able to extract interesting features covering a large range of spatial frequencies as in the cases of complete and over-complete RBM structures. Once again, we observe that the number of features extracted from the van Hateren database is greater.

In general, the features learned from the normalization data are totally different from the ones learned with whitened data. They remain sparse but cover a broader spectrum of spatial frequencies. An interesting observation is that they look closer to

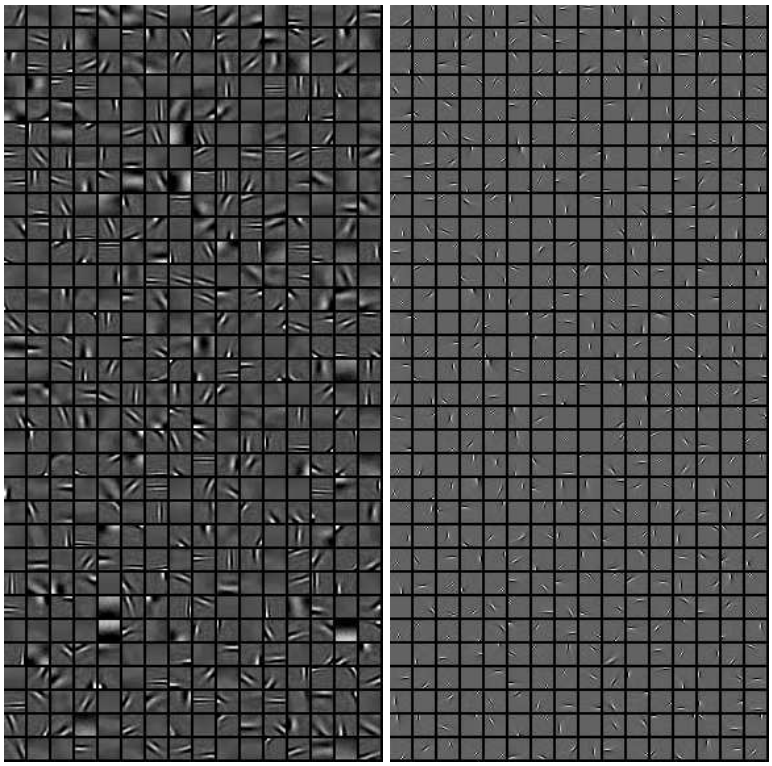

Figure 3.21: Learned over-complete natural image bases. **Left:** 512 features learned by training the first RBM layer on normalized image patches (16x16) sampled from the Berkeley dataset. **Right:** The corresponding features learned by training the first RBM layer on whitened image patches (16x16) sampled from the same database. For both experiments, the same conditions as in figure 3.20 were used.

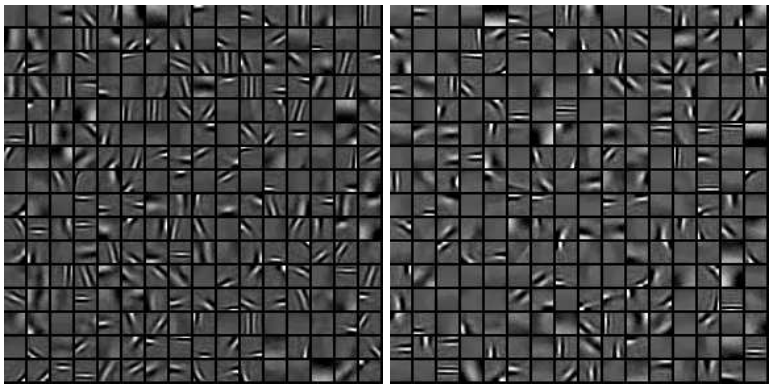

Figure 3.22: Learned complete natural image bases. **Left:** 256 features learned by training the first RBM layer on normalized image patches (16x16) sampled from van Hateren dataset. **Right:** The corresponding features learned from Berkeley database. For both experiments, similar conditions as in figure 3.20 were also used.

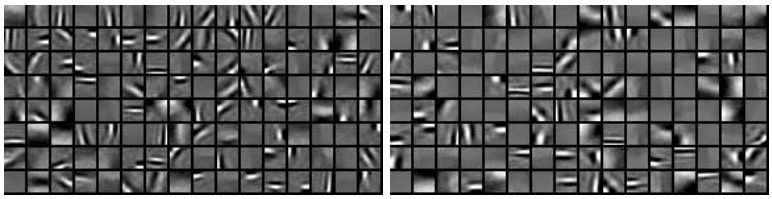

Figure 3.23: Learned under-complete natural image bases. **Left:** 128 filters learned by training the first RBM layer on normalized image patches (16x16) sampled from van Hateren dataset, these features are globally normalized. **Right:** The corresponding features learned from Berkeley database. For both experiments, similar conditions as in figure 3.20 were also used.

the ones obtained with convolutional networks [Lee et al., 2009] for which no whitening is applied to the initial dataset. We can mention that these differences between normalized and whitened data have already been observed in [Krizhevsky, 2009] and related to better performances for normalized data on CIFAR-10 in an object recognition task.

To try to understand more deeply why features obtained from whitened or normalized patches are different, we computed the mean Fourier spectral density of the patches in the two conditions and we compared them to the same function for the original patches. We plotted the mean of the Log Fourier power spectral density of all the patches according to the Log of the frequencies shown in figure 3.24. The scale law in $1/f^{\alpha}$ characteristic of natural images is approximatively verified as expected for the initial patches. For the local normalization it is also conserved (the shift between the two curves is only due to a multiplicative difference in the signal amplitude between the original and the locally normalized patches). It means that the frequency composition of the locally normalized images differs from the initial one only by a constant factor. The relative frequency composition is the same as in initial images.

On the contrary, whitening completely abolishes this dependency of the signal energy with frequency. This means that whitening equalizes the role of each frequency in the composition of the images[1]. This suggests a relationship between the scale law of natural images and the first two moments of the statistics of these images. It is interesting to underline that we have here a manifestation of the link between the statistical properties of an image and its structural properties (in terms of spatial frequencies). This link is well illustrated by the Wiener-Khintchine theorem and the

[1]That is an expected effect since whitening can be related to white noise, a noise in which all the frequencies are equally represented

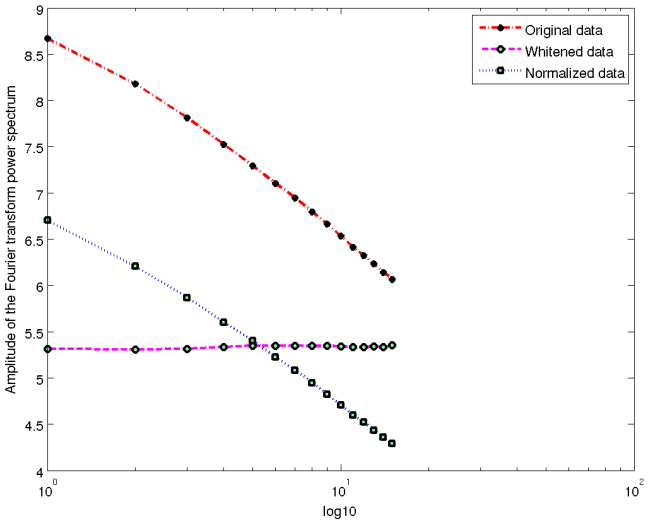

Figure 3.24: The Log-Log representation of the mean Fourier power spectrum for image patches with and without normalization. 10000 16x16 patches have been extracted from the Berkeley database and then normalized. The mean of the Log of the Fourier transform of each of these patches has been computed and plotted according to the Log of the spatial frequency.

relationship between the autocorrelation function of the image and its power spectral density. Concerning the extracted features, these observations allow to deduce that an equal representation (in terms of amplitude) of all the frequencies in the initial signal gives rise to an over-representation of high frequencies in the obtained features. This could be due to the fact that, in whitened data, the energy contained in each frequency band increases with the frequency while it is constant in initial or normalized images.

However the result depends on the database used and consequently on the spatial frequencies contained in the initial patches. The fact that local normalization preserves (to a constant value) the same frequency composition as in initial data tends to prove that normalization does not entirely remove second-order correlations. Olshausen [Olshausen and Field, 1997] showed that, with whitening, ICA mainly retains filters in a narrow range of spatial frequencies. Low spatial frequencies are under-represented in the obtained result. This is clearly what we obtain here with whitening but not with normalization, which tends to save a broader range of spatial frequencies.

We are going to see in the next two chapters how the final database used to test our SPR model behaves according to these two normalization methods and how these changes in spatial frequency composition affect classification performances.

We can argue that low frequency dependencies are related to statistical correlation between neighbor pixels. Thus, the suppression of these second order correlations would suppress these low frequencies in the whitened patches. The resulting features set is expected to contain a larger number of low frequency less localized features, what is actually observed.

3.5 Summary

In this chapter, we have first conducted experiments on different datasets and we have shown that our RBM algorithm captures high-order interesting features similar to those obtained by the state-of-the-art. This underlines that the algorithm has been successfully implemented. We have seen that the appropriate values of these parameters are typically found by trial and error. This task is very tricky and confirmed that the stochastic gradient learning of an RBM can easily diverge, if the associated parameters are not chosen carefully [Fischer and Igel, 2010; Schulz et al., 2010]. Our findings are roughly in accordance with those proposed in the most recent literature [Hinton, 2010]. Finally, we have seen that data normalization significantly affects the detection of features by extracting higher semantic level features than whitening.

Chapter 4

Model presentation and properties

4.1 Introduction

In the preceding chapters we have first seen the main approaches to SPR and we have focused on a new way of building a feature space appropriate for SPR, the use of Deep Belief Networks (DBNs). In the previous chapter we have presented a thorough set of experiments designed to precise the conditions for obtaining this appropriate feature space.

In this chapter we are going to put at work all of these observation in a model able to perform SPR. We will first present the overall organization of the proposed model and then the different conditions of use of this model in the context of SPR, including image pre-processing and the use of tiny images. We will study what kind of features we obtain in these specific conditions and how they can be used for image classification. Thus, two main questions will be raised here : what kind of features will be extracted from tiny images and how the feature space is affected by the normalization procedure in this specific case? The present chapter focuses on describing these approaches and their characteristics in the context of image coding. We assume that the use of this coding method (based on the assumption that sparse features are extracted) will increase the linear separability of the data representation, so that a simple classifier like softmax regression in the feature space will suffice to determine the robot place according to a given image.

4.2 Databases description

However, before describing the organization and study the properties of the model, we are going to present the databases used for the experiments of this chapter. In particular we will here discuss the way these databases are transformed in order to make them suitable for image classification and SPR.

4.2.1 The WILD database

The RBM algorithms seen in the previous chapter used random patches sampled from natural images. We are interested to investigate our RBM model using small tiny images. In this context, there is a difference between object and scene recognition. One major question is, for object recognition, the role of focusing. It seems obvious that when the objects are aligned (focused at the center of the images) the feature set build by a DBN is different from the one with unaligned objects. To investigate the effects of the alignment technique on the detection of features we used a new database called "Labeled Faces in the Wild database" (LFW). Contrarily to previous ones, this database is designed for classification but it is simpler than NORB or CIFAR-10. It has been already used in the literature in such a way that we can compare our results with previous ones and is suitable, as we have said, for studying the effect of object alignment or focusing.

The LFW dataset is a collection of $250x250$ color faces designed for studying the problem of unconstrained face recognition [Gary et al., 2007]. This database contains $13,233$ images of $5,749$ people collected from the web [Gary et al., 2007] [1]. An alignment technique is used to normalize the faces. As a result, the noses, eyes, eyebrows, mouths, and head limits become aligned.

For the centered faces, background surrounding each face is not useful. Thus, as proposed in [Nair and Hinton, 2010], we have eliminated the background information using a $144x144$ window from the center of each face, as if the face would have been at the center of an attentional window.

[1] LFW dataset is available at: http://vis-www.cs.umass.edu/lfw/

4.2.2 The COLD database

The COLD database (COsy Localization Database) was originally developed by [Ullah et al., 2007] for the purpose of robot localization [1]. It is the main database on which we have chosen to validate our model. The main reason is that it has been used to produce what can be considered as the state-of-the-art results in SPR. This is naturally to these results we want to compare the results obtained by our model.

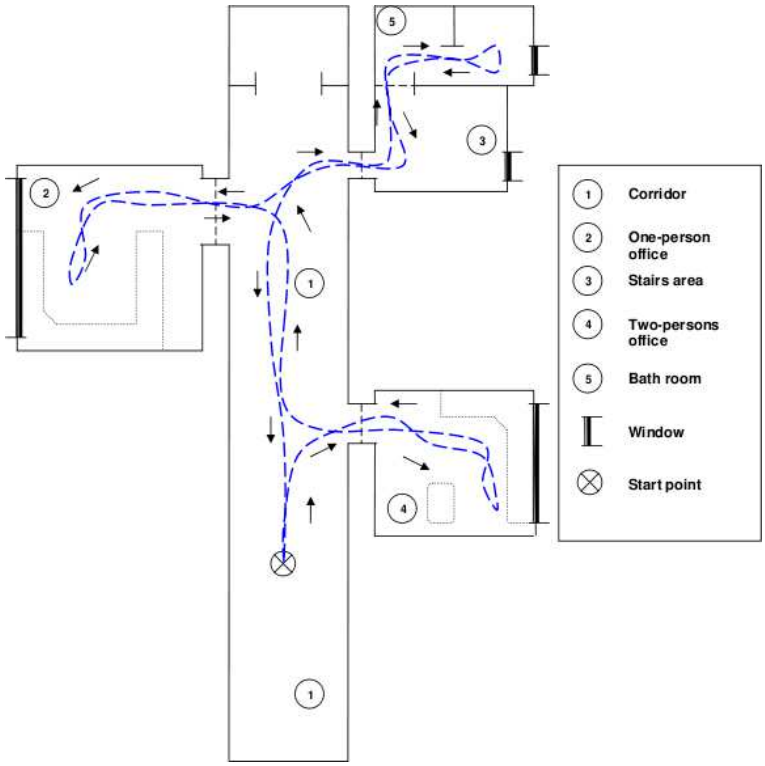

Figure 4.1: Map of Freiburg laboratory, portion B.

This database is a collection of labeled $640x480$ images ($137,069$ different images) acquired at 5 frames/sec during a robot exploration of three different laboratories in Freiburg, Ljubljana, and Saarbruecken. Two sets of paths called standard A and B have been acquired under different illumination conditions (sunny, cloudy and night), and for each condition, one path consists in the visit of different rooms (corridors, printer areas, one person office, two persons office, toilets, *etc.*), for detail see for instance

[1] COLD Database is available at: http://cogvis.nada.kth.se/COLD/

the map of Freiburg laboratory for portion B in figure 4.1. These walks across the laboratories are repeated several times.

Although color images have been recorded during the exploration, only gray-level images were used since previous works have demonstrated that, in the case of the COLD database, colors are weakly informative and made the system more illumination dependent [Ullah et al., 2007]. This fact is illustrated in figure 4.2.

Figure 4.2: **First row:** color images acquired under three different illumination conditions (sunny, cloudy, and night) respectively. **Second row:** the corresponding gray-scale images. It is obvious that the illumination variations reduced in the case of gray images.

4.3 Model initial steps - Unsupervised feature learning

Figure 4.3 shows the general scheme of the proposed approach. It involves three main phases : i/ image coding, ii/ unsupervised feature space elaboration, and iii/ supervised places learning. We hope that the properties of separability of the feature space will allow good classification performances.

Figure 4.4 illustrates in detail the phases of the proposed model. The first phase consists in the conversion of color to gray-scale images, reducing them to small image patches, and then normalizing them. The second phase is the coding of the input images using features. It consists in the extraction through several layers of RBMs forming a DBN of an alphabet of features able to optimally code the images and suitable for their classification. The third phase is the classification itself, which discriminates between the robot possible places.

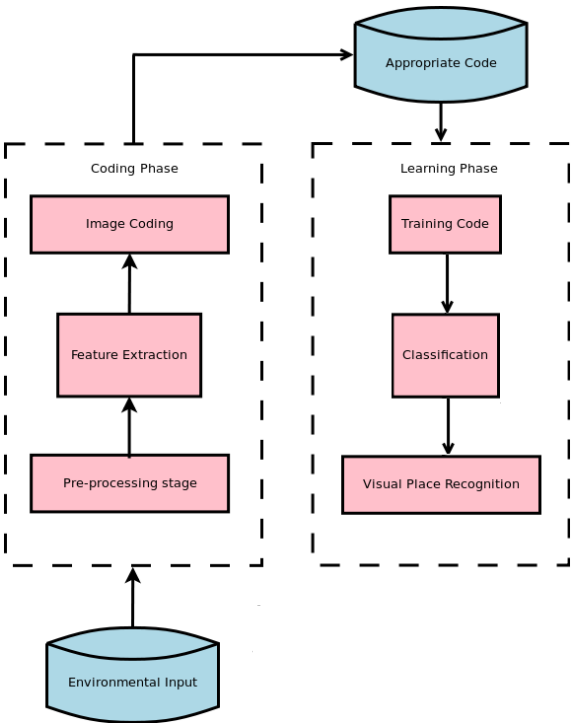

Figure 4.3: General framework of the proposed visual place recognition system. The arrows show the direction of the data flow between the different phases.

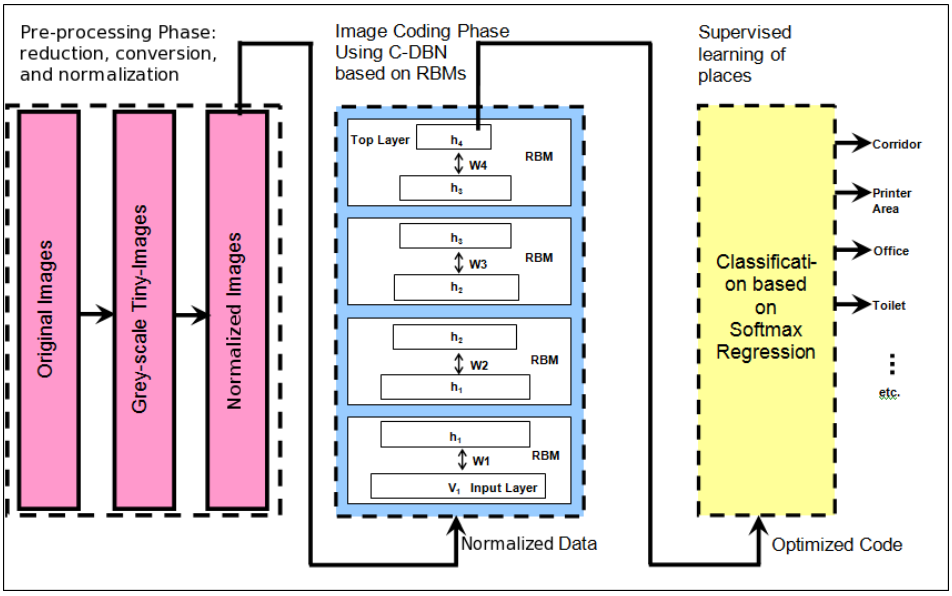

Figure 4.4: The different phases of the proposed model to achieve SPR for autonomous systems.

4.3.1 Pre-processing

As shown in figure 4.4, data pre-processing is the first phase of our algorithm. It involves three inner functions (or steps): image color conversion, image reduction, and image normalization. These three stages are illustrated in the following sections.

4.3.1.1 Use of tiny images?

The typical input dimension for a DBN is approximately 1000 units (*e.g.* 30*x*30 pixels). Dealing with smaller patches could make the model unable to extract interesting features. Using larger patches can be extremely time-consuming during feature learning. Additionally the multiplication of the connexion weights acts negatively on the convergence of the CD algorithm. The question is therefore how to scale the size of realistic images (*e.g.* 300*x*300 pixels) to make them appropriate for DBNs?

Three solutions can be envisioned. The first one is to select random patches from each image as done in [Ranzato et al., 2010], the second is the use of convolutional architectures, as proposed in [Lee et al., 2009], and the last one is to reduce the size of each image to a tiny image as proposed in [Torralba et al., 2008]. The first solution extracts local features and the characterization of an image using these features can only be made using BoWs approaches we wanted to avoid. The second solution shows the same limitations as the first one and additionally gives raise to extensive computations that are only tractable on GPU architectures. Features extraction using random patches is irrespective of the spatial structures of each image [Norouzi et al., 2009]. In the case of structured scenes like the ones used in SPR these structures bear an interesting information.

Besides, tiny images have been successfully used ([Torralba et al., 2008]) for classifying and retrieving images from the 80-million images database developed at MIT [1]. Torralba showed that the use of tiny images combined with a DBN approach led to code each image by a small binary vector defining the elements of a feature alphabet that can be used to optimally define the considered image. The binary vector acts as a bar-code while the alphabet of features is computed only once from a representative set of images. The power of this approach is well illustrated by the fact that a relatively small binary vector (like the ones we use as the output of our DBN structure)

[1] The 80-million database is available at : http://groups.csail.mit.edu/vision/TinyImages/

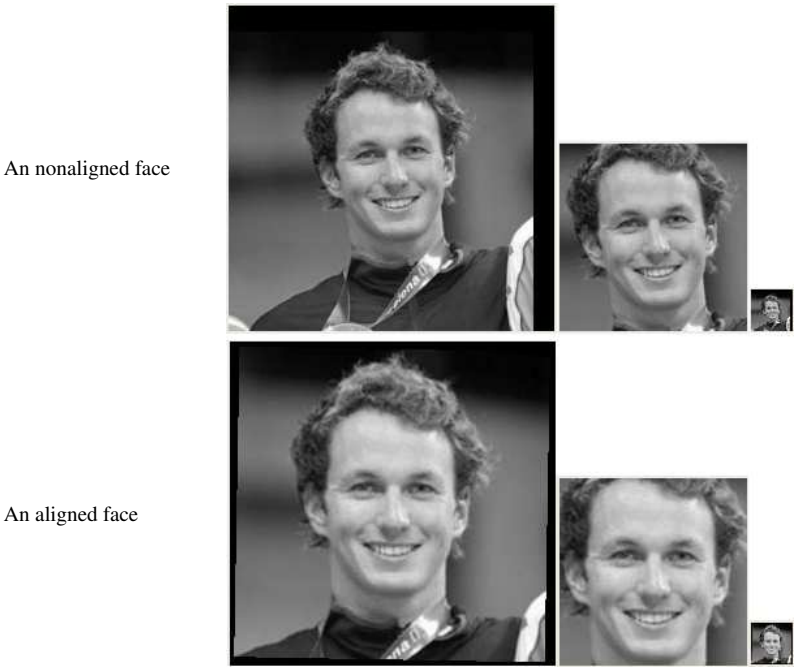

Figure 4.5: An example of non-aligned and aligned faces: the first row represents an unaligned face, where the first image is the original face of size of 255x255, the second one is the centered face of size of 144x144, and the final one is the reduced face of size of 32x32. The second row represents the corresponding aligned face [Gary et al., 2007].

largely exceeds the number of images that have to be coded even in a huge database ($2^{256} \approx 10^{75}$). So, for all these reasons we have chosen image reduction.

Thus for the WILD database, the images were reduced to 32x32 pixels for the aligned and non-aligned (original faces) images as shown in figure 4.5.

For the COLD database they were reduced to 32x24 pixels (figure 4.6) to approximately save the aspect ratio of the initial images (640x480). As for the WILD database, after this reduction, the tiny images are still fully recognizable as shown in the same figure. The final set of tiny gray images (a new database called tiny-gray-COLD) will therefore be used as input for the normalization algorithms.

4.3.1.2 Image conversion

Although color images have been recorded during the exploration, only gray images are used according to [Ullah et al., 2007]. Therefore, the whole images are converted

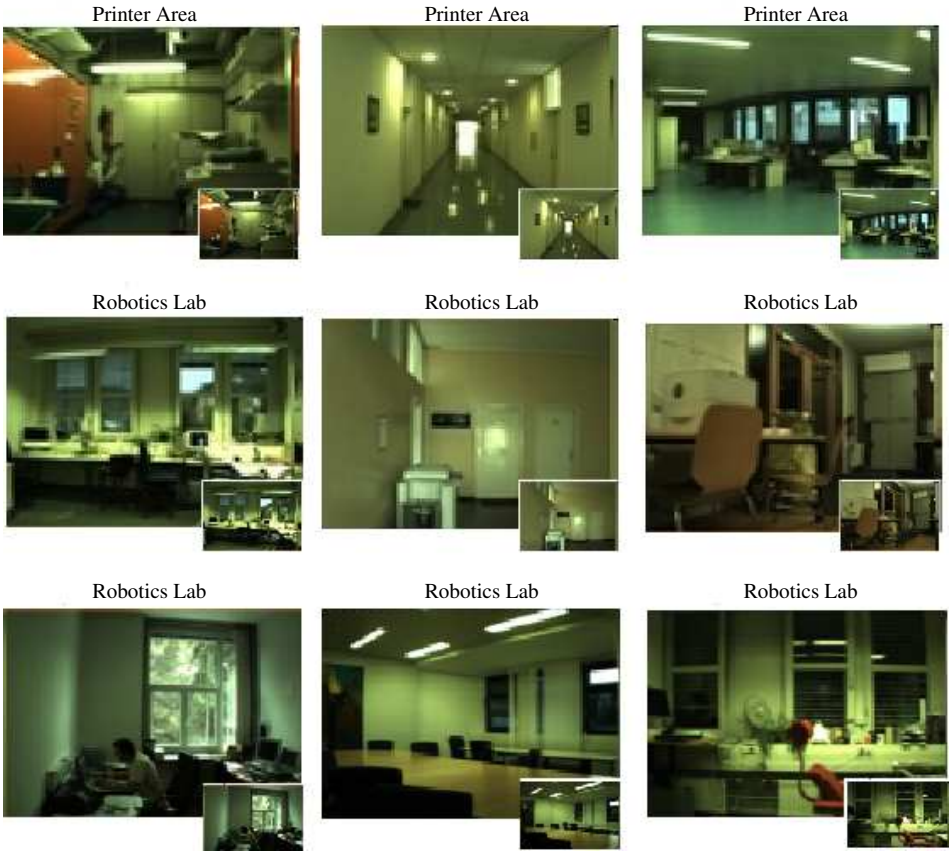

Figure 4.6: Samples of the initial COLD DB. The corresponding 32x24 tiny images are displayed bottom right. One can see that, despite the size reduction, the small images remain fully recognizable.

from colors to gray-scale using the following linear transformation function:

$$Y = (0.299 * R) + (0.587 * G) + (0.114 * B) \tag{4.1}$$

Thus, the first two functions of the pre-processing phase, gray-scale conversion and size reduction, are applied to the whole images. A proposed flowchart is shown in figure 4.7 to perform these functions.

As explained in the previous chapter, data can be whitened or locally normalized. However, in the previous chapter we investigated the effect of the two methods on small random patches extracted from large images. Here we will consider the effects of these methods on reduced images. Before seeing these effects on the classification results in the next chapter, we will study them here on the feature extraction.

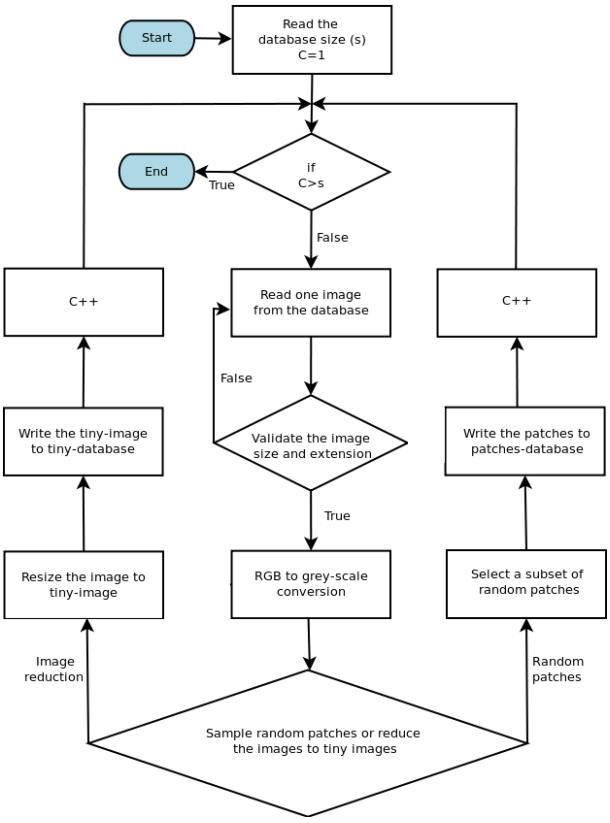

Figure 4.7: A flowchart of the first step of pre-processing phase: Image conversion and reduction. Note that the parameters s and C denote the size of the database and the image counter respectively.

4.3.2 Unsupervised features extraction

4.3.2.1 WILD database

Focusing We have tested our model on aligned and non-aligned faces databases. After gray-scale conversion and image reduction, whitened tiny images from the WILD database have been used as input for training the RBM algorithm. We have tested first a complete structure. Thus the first RBM layer was $1024 - 1024$ as the images have been reduced to $32x32$ pixels.

Figure 4.8 shows the most interesting features extracted from respectively the non-aligned faces database (left) and the aligned one (middle). These features have been selected as the ones having the lowest entropy among all the 1024 features obtained in both experiments. These results are quite similar to the ones published by [Nair

and Hinton, 2010] as shown on the same figure 4.8 (right). The only difference is that the authors in [Nair and Hinton, 2010] used color images as input to the RBM. The aligned database leads to features representing faces or parts of faces (like eyes, mouth or noses) as well in our case as in [Nair and Hinton, 2010].

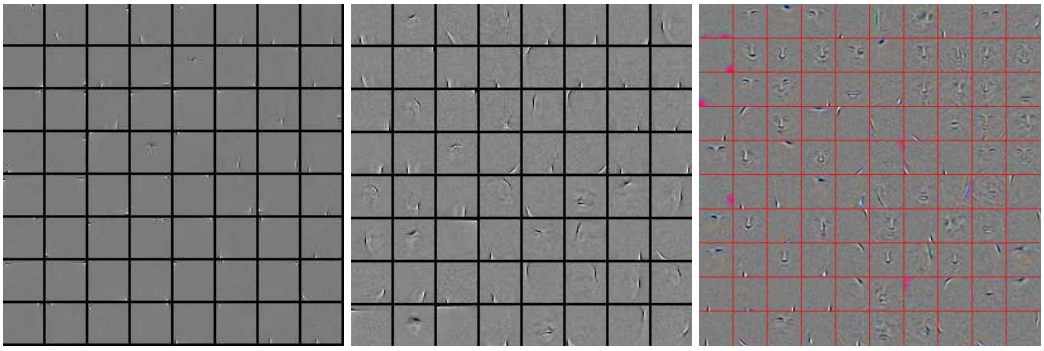

Figure 4.8: **Left:** 64 selected filters among 1024 learned by training a RBM layer on $32x32$ whitened image patches sampled from the non aligned face database. **Middle:** 64 selected filters learned by training a RBM layer on $32x32$ whitened image patches sampled from aligned faces. **Right:** A subset of features extracted from the same database by [Nair and Hinton, 2010] for comparison.

The features extracted from the non-aligned database are more localized. They are very peaked and include no semantic details about the nature of the objects although the database is only made of faces.

Completeness Another important point is that a large number of features did not converge to a significant pattern even after extensive computation. Contrarily to what happened with the van Hateren and Berkeley databases for which the number of extracted features increased with the number of hidden units, it was not here possible to obtain the same result. We have also tried other experiments using over-complete structures ($1024 - 4096$), but it does not show any improvement in the extracted features with a much greater computational load.

Thus, it could be interesting to investigate the RBM algorithm ability to extract similar features, reducing the number of hidden units. To do so, an under-complete structure of $1024 - 256$ has been investigated using the aligned database, as shown in figure 4.9.

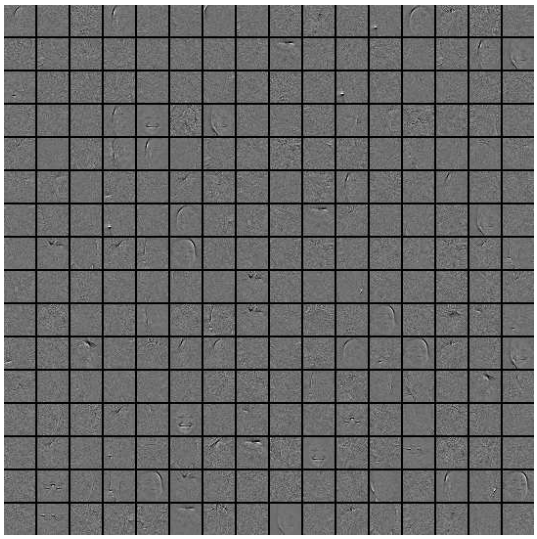

Figure 4.9: 256 features learned by an RBM layer on $32x32$ whitened image patches sampled from the aligned face database. The same training protocol as in previous experiments has been used.

We can observe that the features extracted using an under-complete structure are similar to the most significant of those extracted using a complete structure. The lack of improvement using a more important number of hidden units suggests to use such under-complete structure for the final classification. We will see later if this result is confirmed with the COLD database.

Normalization The question of whitening impact has also been investigated on the LFW database as we have done on the van Hateren and Berkeley databases. Two experiments with aligned and non-aligned LFW normalized image patches gave the features shown in figures 4.10 and 4.11 respectively. Note that, in this case, a complete RBM structure was used. Two observations can be made : first, all the features gave significant patterns and second, the obtained features seem to cover a broader range of spatial frequencies especially in the case of the non aligned database. Thus the result seems to be different from the one obtained with whitened data and shown before (figure 4.9).

Such features seem to be more promising for reconstructing original or new faces. We can close up this result with the one published years ago [Turk and Pentland, 1991] showing that with a face database, the principal axes of a PCA correspond to ”eigen-

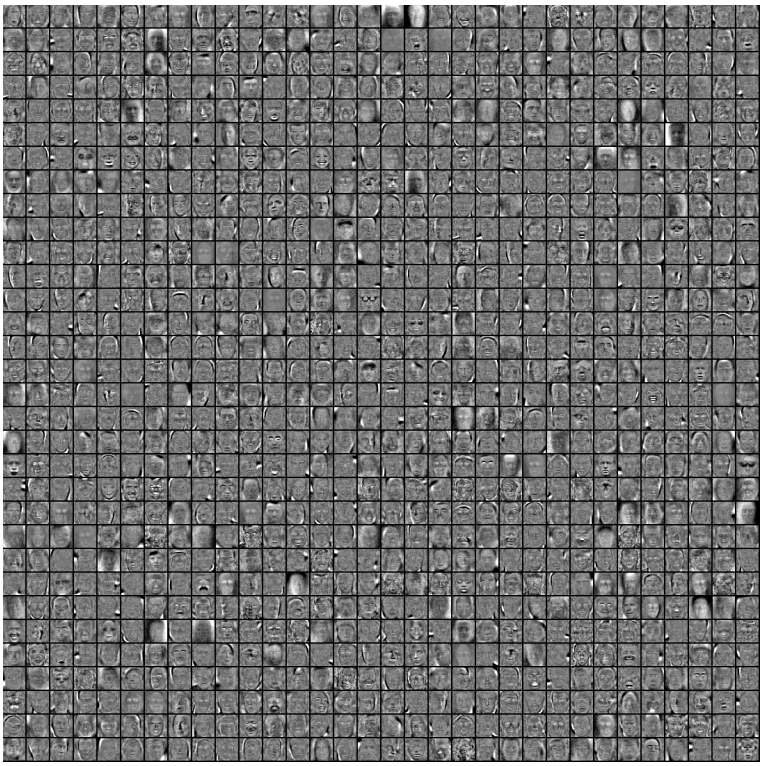

Figure 4.10: 1024 filters learned by training the first RBM layer on $32x32$ normalized image patches sampled from the aligned face database. The training protocol is similar to the one proposed in [Nair and Hinton, 2010] ($\varepsilon = 300$, $\gamma = 100$, $\eta = 0.02$, $\mu_i = 0.5$, $\mu_f = 0.9$, and $\lambda = 0.0002$).

faces". Otherwise stated, the first eigen-vectors of the PCA correspond to the main shapes of faces from which any particular face can be reconstructed. We have here an interesting parallel with PCA and linear methods. Instead of extracting the decorrelated characteristics of an input signal or even the independent components on the basis of linear transforms, RBMs extract much richer and numerous statistically independent components from a similar database.

However, these results do not show the difference we have observed with whitened data between aligned and non aligned face images. Although, in whitened data, the features obtained with the aligned faces consist in parts of faces and are rarely "eigen-faces", with normalized data, the features seem to code for larger structures like the structures of the whole faces. To summarize, the normalized data produces more features as well in the case of non-aligned as aligned faces. These features seem to represent high-level semantic parts of the images and more frequently "eigen-faces" [Turk

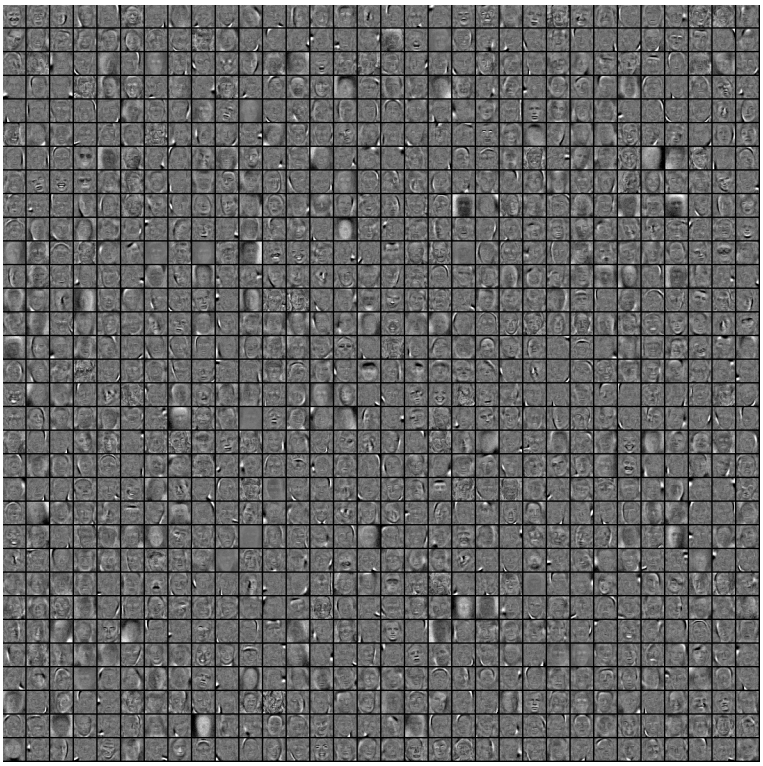

Figure 4.11: 1024 filters learned by training the first RBM layer on 32x32 normalized image patches sampled from the non-aligned face database. These features have been obtained using the same training protocol as in the previous experiment.

and Pentland, 1991] in aligned images than in non-aligned images.

Additionally, an under-complete RBM extracts similar features as shown in figure 4.12. However in this case the variability of the features is less than for complete structures.

4.3.2.2 COLD database - Final validation

Low level features and Normalization We finally performed the same experiments on the tiny-COLD database which is our reference database. In these experiments, with a learning rate of 0.02 as in previous experiments, the network converged very fast but the obtained solutions are not optimal. Consequently, we have reduced the learning rate to 0.002. One explanation could be that, in the case of the COLD database, the images are highly redundant and neighbor images in time are similar. The network could need more time to focus on the small details distinguishing one image to the

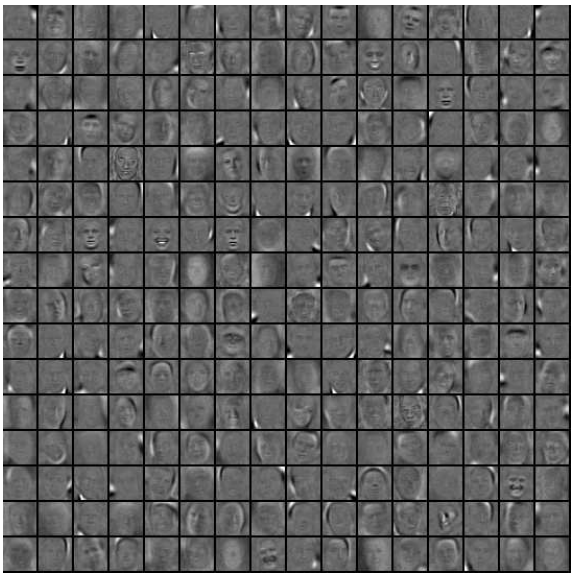

Figure 4.12: Learned under-complete natural faces bases. 256 features learned by an RBM layer on $32x32$ normalized image patches sampled from the aligned face database. In this experiment, same protocol used as previous ones.

others. Using a smaller learning rate will slow down the convergence process but it allows the network to converge to a better solution (*i.e.*, the network extracts more interesting features). It has also been shown that when modeling real-valued Gaussian visible units, training the first RBM layer of features typically requires a much smaller learning rate to avoid oscillations [Salakhutdinov and Hinton, 2009]. But we observed this effect only on the COLD database.

The features shown in figure 4.13 have been extracted by training the first RBM layer on $137,069$ whitened image patches ($32x24$ pixels) sampled from the COLD database. Some of them represent parts of the corridor, which is over-represented in the database. They correspond to long sequences of images quite similar during the robot exploration. Some others are localized and correspond to small parts of the initial views, like edges and corners, that can be identified as room elements (*i.e.* they are not specific of a given room).

The features shown in figure 4.14 have been obtained using the normalized data. As previously observed for the other databases, the obtained features look very different. Parts of rooms are much more represented than for the whitened database and it seems that the range of spatial frequencies covered by the features is much broader confirming

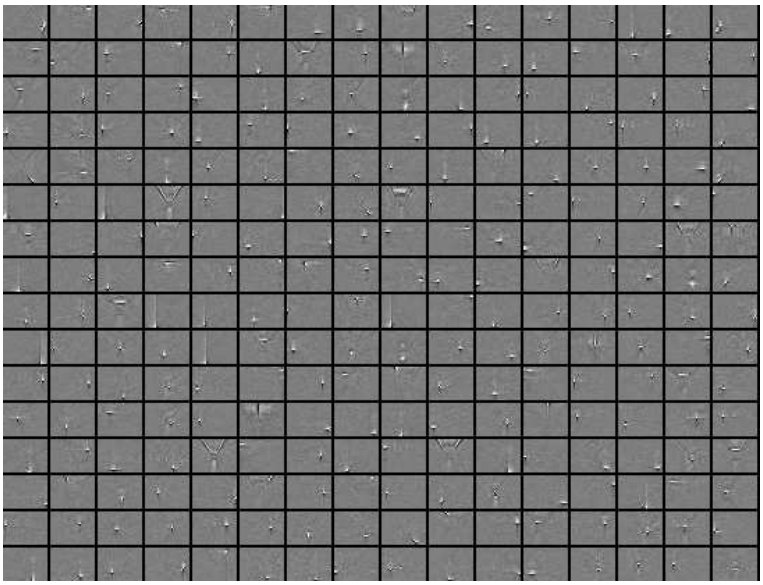

Figure 4.13: 256 filters obtained by training a first RBM layer on 32x24 whitened image patches sampled from the COLD database. The training protocol is similar to the one proposed in [Krizhevsky, 2010] ($\varepsilon = 100$, $\gamma = 100$, $\eta = 0.002$, $\lambda = 0.0002$, $\mu_i = 0.5$, and $\mu_f = 0.9$).

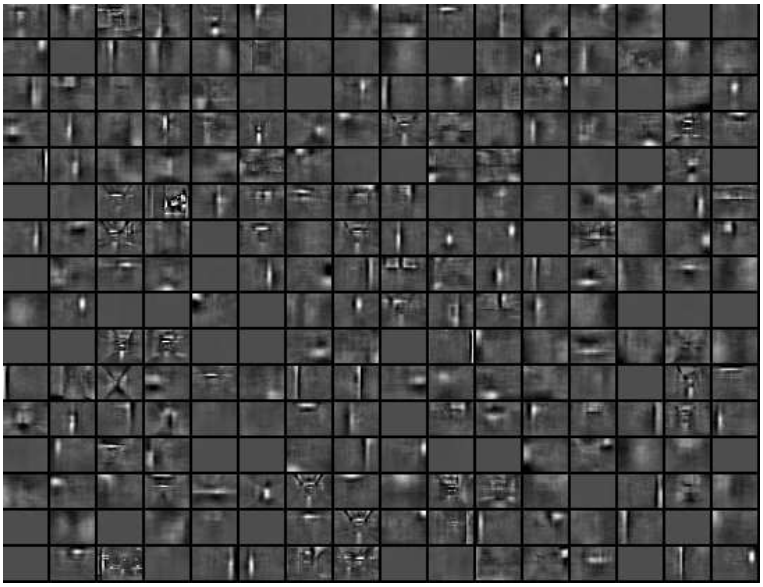

Figure 4.14: The 256 filters obtained by training the first RBM layer on 32x24 normalized image patches sampled from the COLD database. The training protocol is similar to the previous experiment.

what has been already observed in other parts of this work.

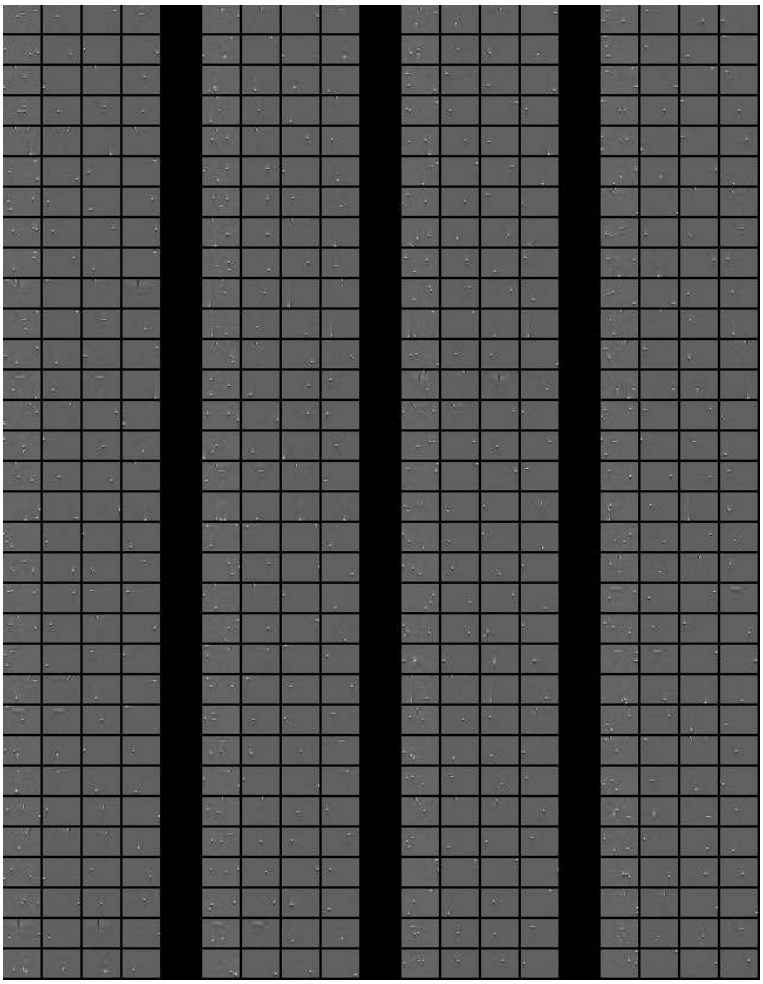

Figure 4.15: The second-level features extracted from the whitened tiny-COLD data-base. The figure shows the three most prominent first level features used in the construction of each high-level feature. The pattern for this second level feature is a linear combination of the first one weighted by the first layer connection weights.

Upper layers In order to understand how these low level features are used to form higher level representations in the upper layer of the DBN we have performed a similar computation as in [Lee et al., 2008]. figures 4.15 and 4.16 show the 128 high level features (first left patch in each column) formed with the linear combination of features in the preceding layer. The combination of these initial features in higher RBM layers correspond or partially correspond to larger structures, more characteristics of the different rooms (figures 4.15 and 4.16).

However, the high level feature space obtained with locally normalized data shows

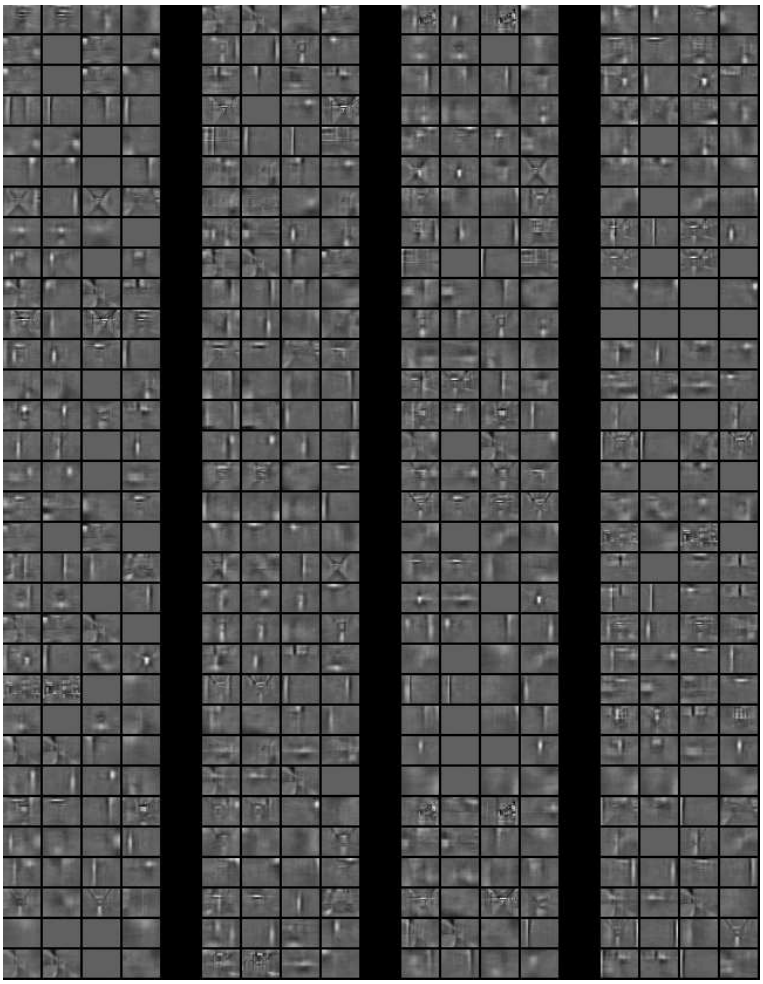

Figure 4.16: The second-level features extracted from the normalized tiny-COLD database. The figure shows the three most prominent first level features used in the construction of each high-level feature. The pattern for this second level feature is a linear combination of the first one weighted by the first layer connection weights.

patterns that can be more easily related to room structures than those from whitened data.

The obtained codes in the different conditions of normalization were used directly as the final input vector of the classification process that we will study in the next chapter.

4.4 Summary

After the relevant works have already been presented and discussed in the previous chapter, in this chapter we have focused on some specific characteristics of our model for SPR with an emphasis to the following points:

- The use of tiny images.
- The differences in the obtained features between whitening and local normalization.

Image conversion and normalization This chapter has provided a detailed description of the proposed model stages. In particular, we first presented the different functions (image color conversion, image reduction, and image normalization) required to pre-process images. On the other hand, we have noted that without any initial correction (without whitening and without local normalization of brightness and illumination) of the patches, the RBM algorithm converges to a wrong solution. A normalization of any kind of the initial patches is necessary for a RBM to get localized features.

Then, the second part of these experiments have focused on studying the final impact of whitening and normalization on features extraction. We have seen that data whitening encourages the RBM algorithm to learn very localized features, while data normalization forces the RBM algorithm to converge toward features that cover a larger spatial frequency spectrum and especially the low frequencies able to capture the overall organization of the scene.

Use of tiny images We have proposed to use tiny images instead of image patches that would have required to use BoWs approaches and we have shown that DBNs can be successfully used in this case to extract sparse efficient features. Working with size-reduced images seems indeed simpler than BoWs approaches.

Sparse code A sparse code makes the data linearly separable in the feature space and thus allows to use a simple linear classifier, like softmax regression, to achieve an accurate and rapid SPR. As a consequence, we proposed to use RBM approaches to extract a set of independent features that can be used in image coding. We assumed

that these features must be sparsely represented and thus they can be used to create a linear separable code in the feature space. This process can be repeated several times until the CD learning algorithm converges towards a sparse representation of the initial images.

Linear separability If this assumption is true, a simple classifier is then sufficient to determine the robot place according to the given input image. To investigate this hypothesis, we proposed to test the classification phase first using a linear classifier, like softmax regression, and then using a nonlinear classifier, like SVM. The hope is that DBN computes a linear separable signature for the initial data. These questions will be studied in the next chapter.

Completeness and network structure Finally, our RBM learning algorithm extracts interesting features in the cases of over-complete, complete, and under-complete structures. This allows us to use an under-complete structures of DBNs to speed-up the learning process and thus simplifies the classification process. Besides, DBNs perform non-linear dimensionality reduction and they can learn short binary codes that allow very fast retrieval of documents or images [Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2007]. More precisely, the high-dimensional data can be converted to low-dimensional codes by training a deep belief network. This reduction speeds up the classification process. It has been formally demonstrated that this reduction works much better than principal components analysis (PCA) as a tool to reduce the dimensionality of data [Salakhutdinov and Hinton, 2007].

Alignment We have also considered the question of the effect of focusing the gaze of the system towards a region of interest through a study of the feature extracted from an aligned and a non aligned face database. As expected, focus the image on a specific class of objects allows the system to extract more semantically relevant features than when the images are not centered on the object to be considered. However, it could seem contradictory with the view-based scene recognition approach we have adopted since focusing on objects could require to recognize them before. We just want to mention the numerous recent works on attentional focalization on proto-objects or region of interest [Walther and Koch, 2006] that open the vicious circle : to recognize an

object, it is required to center it in the visual field and, in the meantime, to center it, it must have been recognized before.

The COLD database is obviously uncentered. So we would wait for unspecific features with a low semantic level. This is not exactly the case due to the way the robots acquire the images : for example in corridors, the images are more centered due to the high level of constraints in the movements of the robots and the repetitive nature of the images. In this case, the extracted features will be more semantically significant. If a robotic system is endowed with a mechanism that attracts its gaze to specific views, it will have an over-representation of these view in its dataset and these views will be learned as independent features as in natural systems (*e.g.* the existence of specific cells in the infero-temporal cortex of mammals).

Chapter 5

Vision-based classification: Supervised learning of robot places

5.1 Introduction

In the present chapter, we want to gather all the observations we have made in the previous chapters and the organization we have developed to process the COLD database. We will study here the classification abilities of the high level representation that can be obtained from tiny images. Therefore, this chapter focuses on performing extensive classification experiments in order to evaluate the performance of our approach. To investigate whether the linear separability in the feature space is achieved, we propose to compare the classification performances using a linear or a nonlinear classifier. All the classification results presented here will be interpreted and compared with the most recent approaches of SPR [Guillaume et al., 2011; Ullah et al., 2008].

5.2 Vision-based classification results: Supervised learning of robot places

5.2.1 Recall of previous results

In this section, we will first present the best performance for instances recognition that have been achieved in the literature [Pronobis and Caputo, 2007; Ullah et al., 2008] for

easy comparison with our own results. The authors in [Ullah et al., 2008] used the descriptor Scale-Invariant Feature Transform (SIFT) to describe the images. Each image is then classified independently through the use of Support Vector Machines (SVMs). The results obtained for the three laboratories (in particular for standard sequences) are presented in figure 5.1. For each training illumination condition (indicated on top of the charts), the bars present the average classification rates over the corresponding testing sequences under the illumination condition marked on the bottom axis.

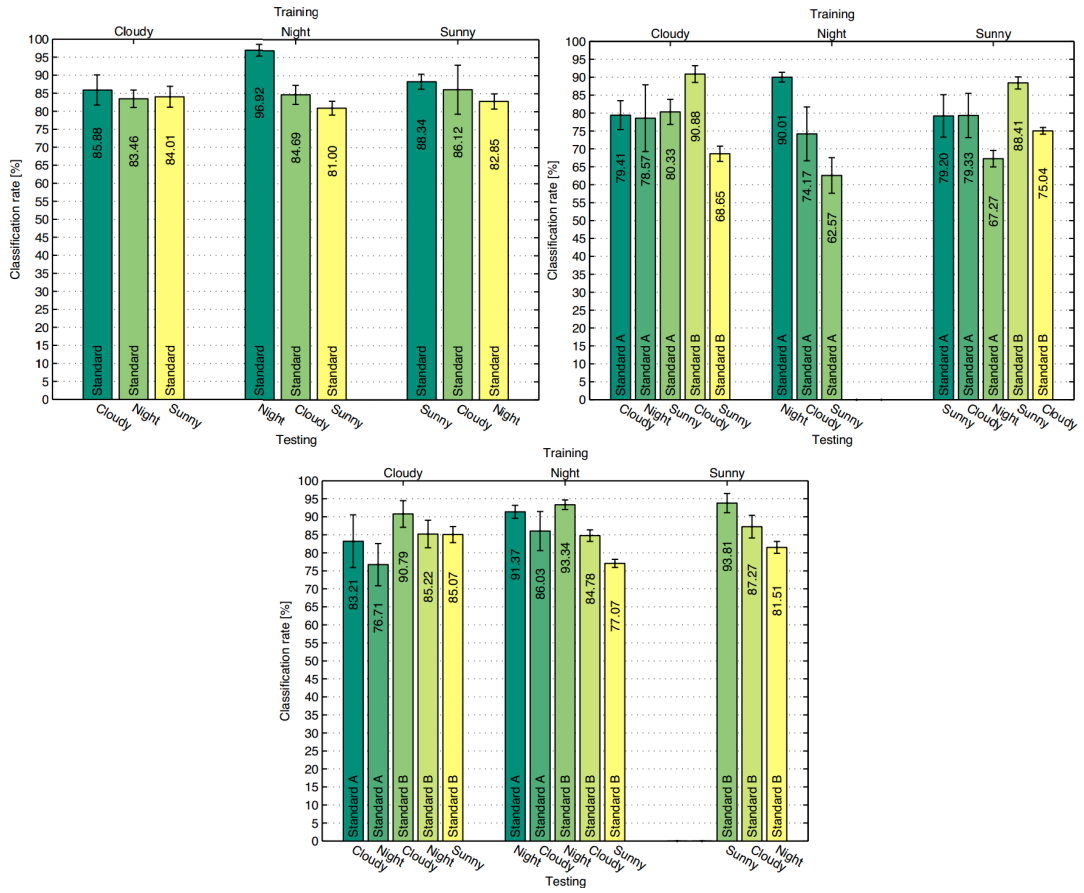

Figure 5.1: Average classification rates from the three different laboratories obtained by [Ullah et al., 2008]. They are grouped according to the illumination conditions under which the training sequences were acquired. Thus, the training conditions are on top of each set of bar-charts while the bottom axes indicate the illumination conditions used for testing. The uncertainty bars represent the standard deviation. Results corresponding to the two different portions of the laboratories which are indicated by A and B. We can see from these results that the system is quite robust to the changes of illumination conditions and the overall performance is almost identical. These graphs have been taken from [Ullah et al., 2008].

Based on their method, they have got an average of correct classification of 90.5% for Saarbrucken laboratory, 85.58% for Freiburg laboratory and 90.38% for Ljubljana laboratory. These average results have been obtained when the illumination conditions were similar for the training and testing. However, the performances decrease for experiments were conducted under various illumination conditions. In this case they have reached classification rates of 82.96% for Saarbrucken laboratory, 73.24% for Freiburg laboratory and 83.69% for Ljubljana laboratory. Finally, they have observed that there is a decrease in performance for the Freiburg laboratory. This can be caused by the glass walls in Freiburg laboratory and the fact that the cameras were mounted significantly lower than for the other laboratories, resulting in less diagnostic information in some of the images [Ullah et al., 2008].

As a conclusion, these methods lead to notable classification rates, however, they are based on sophisticated classification techniques like SVM. Also, the classification results decreased in the case of testing under different illumination conditions, which indicates that these models remain sensitive to these changes.

5.2.2 Recognition of places based on DBNs and tiny images

In this section, we present the classification result of our model, based on DBNs and a direct use of tiny images. We follow the same training and testing protocols as in [Ullah et al., 2008]. Also, we interpret and compare our results with the work of [Ullah et al., 2008]. Finally, we investigate the robustness of our results to the illumination conditions.

It was stated at the beginning of this chapter that after image coding, the final step is to use a classification algorithm to the actual recognition on the basis of the features extracted from the input data, *i.e.* using the feature space. In the present section, we will show the classification results using a simple linear classifier like softmax regression [Ng, 2011] and a nonlinear classifier like Support Vector Machines (SVMs) [Cristianini and Shawe-Taylor, 2000; Vapnik, 1995] respectively. Assuming that the non-linear transform operated by DBN improves the linear separability of the data, a simple regression method would suffice to perform the classification process. We have seen that DBNs are able to extract sparse features from a large amount of images and, using these features, we can create sparse representations of the initial images.

However, if we assume that the problem has not been made linearly separable by DBNs, a simple linear classifier is not sufficient and therefore a nonlinear classifier, like SVMs, will be required to perform the classification process. Studying the classification process using a linear and a nonlinear classifiers would definitely demonstrate whether the linear separability of the data has been obtained by DBNs or not.

The different experiments presented in the preceding chapters have suggested to use an under-complete structure for SPR. We have tested different size of the DBN and concluded that the optimal structure of the DBN for SPR using the COLD database is $768 - 256 - 128$. We have used this network structure in all the results presented here unless otherwise stated.

5.2.2.1 Classification results using a softmax regression

The samples have been taken from each laboratory and each illumination condition were trained separately, as in [Ullah et al., 2008]. For each image, the softmax network output gives the probability of being in each room. According to the maximum likelihood principle, the largest probability value gives the decision of the system. In this case, we obtain an average of correct answers ranging from 50% to 76% according to the different conditions and laboratories as shown in figure 5.2.

Note that the classification results shown in figure 5.2 have been obtained using the code generated by the features extracted without including the regularization term which plays a key element in improving the sparsity property for the features.

5.2.2.2 Classification results using support vector machines

Figure 5.3 shows the classification results using a SVM classifier [1] with a polynomial kernel of degree 2. In this experiment, we have also used the same protocol for both training and testing as in the previous experiment. This figure shows that the average of correct answers is still ranging from 50% to 75% for the different conditions and laboratories. For verification, we have also tried to use this nonlinear classifier with different kernels (linear, radial basis function, sigmoid function, and precomputed kernel). All of them gave very close results to the polynomial one. Furthermore, we have

[1] We have used teh SVM package developed at http://www.csie.ntu.edu.tw/ cjlin/libsvm/

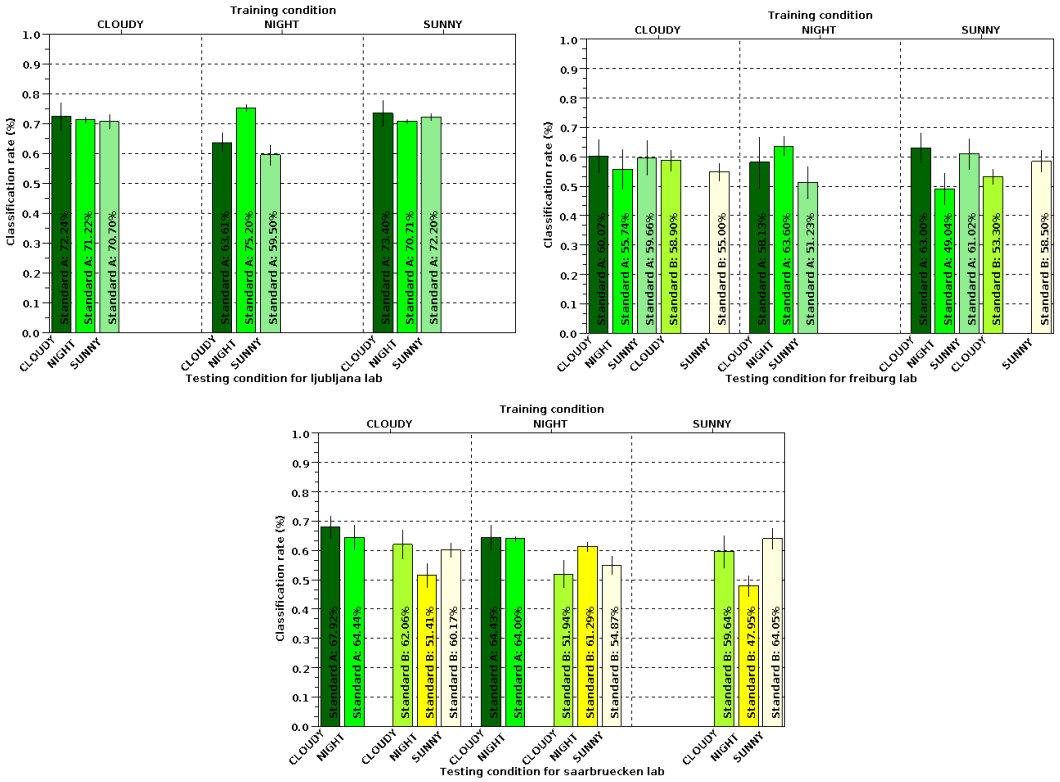

Figure 5.2: Average classification rates from the three different laboratories using a softmax regression. Training conditions are on top of each set of bar-charts. Each bar corresponds to a testing condition. The extracted features in this case are obtained using a learning rate of 0.002 and a weight decay of 0.0002.

investigated different degrees (2 "default", 3, and 4) of the polynomial kernel. They did not notably change or improve the final classification results.

As we have previously said, we can thus conclude that the feature space we have obtained with DBN is linearly separable for the current classification problem.

Concerning the classification results achieved by the linear regression, they seem to be worse compared with the state-of-the-art results [Guillaume et al., 2011; Ullah et al., 2008]. More precisely, the results obtained by [Guillaume et al., 2011] have an average of correct recognition of 80% based on GIST descriptor and 81.24% based on CENTRIST descriptor for the three different laboratories of the COLD database. Also, the results in [Ullah et al., 2008] have an average of correct answers of 83% based on more sophisticated techniques (use of SIFT detectors followed by a SVM classification) for the same three laboratories. While in our case, the results have

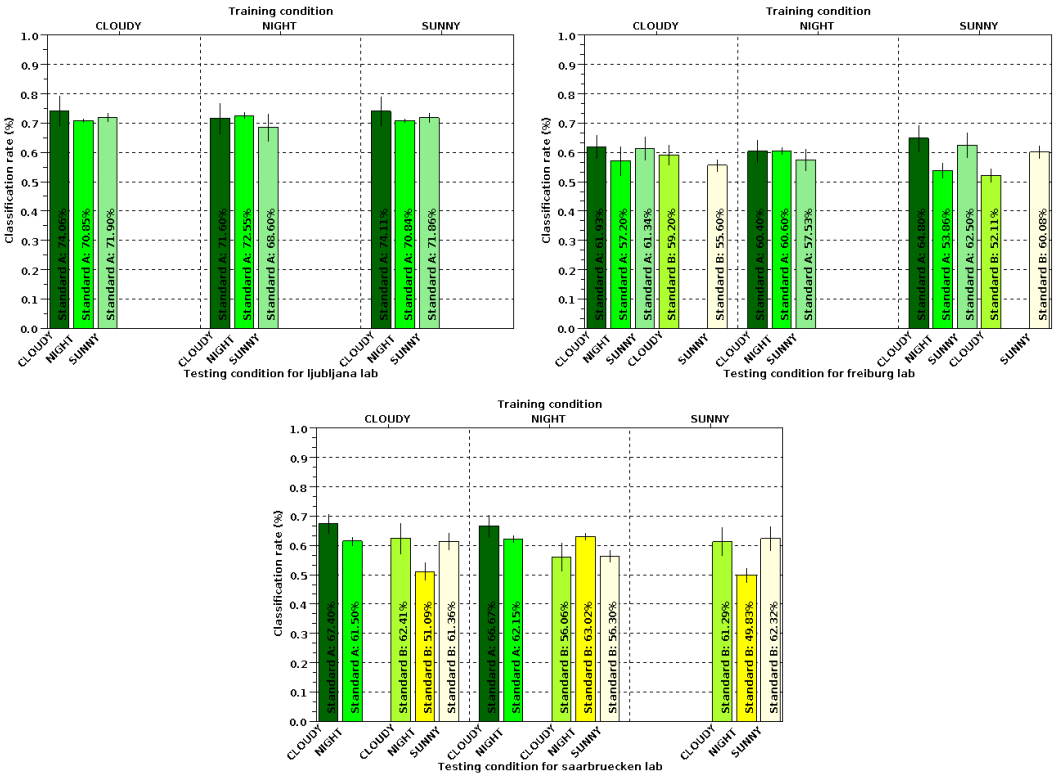

Figure 5.3: Average classification rates from the three different laboratories using a nonlinear SVM classifier. The extracted features in this case are obtained using a learning rate of 0.002 and a weight decay of 0.0002.

an average of correct answers of 61.5%. Perhaps the strong size reduction of the initial images from $640x480$ to $32x24$ pixels has strongly affected in losing a lot of interesting information. However, there are still several open ways for improving these classification results to reach the state-of-the-art results. The first possible way is to study different factors acting on sparsity.

5.3 Encouraging sparse hidden activities

Since the use of the nonlinear classifier does not change or improve the classification results, thus we should think about another way to improve them. A possible way relies on encouraging the sparsity property for the hidden activities. Improving this property should allow final recognition results improvement through linear separability data increasing. To achieve that, several factors, including the learning rate, the weight

decay, and the penalty term could have some impact. In particular, adding a penalty term to encourage hidden activation units to be sparse would significantly change the overall classification results.

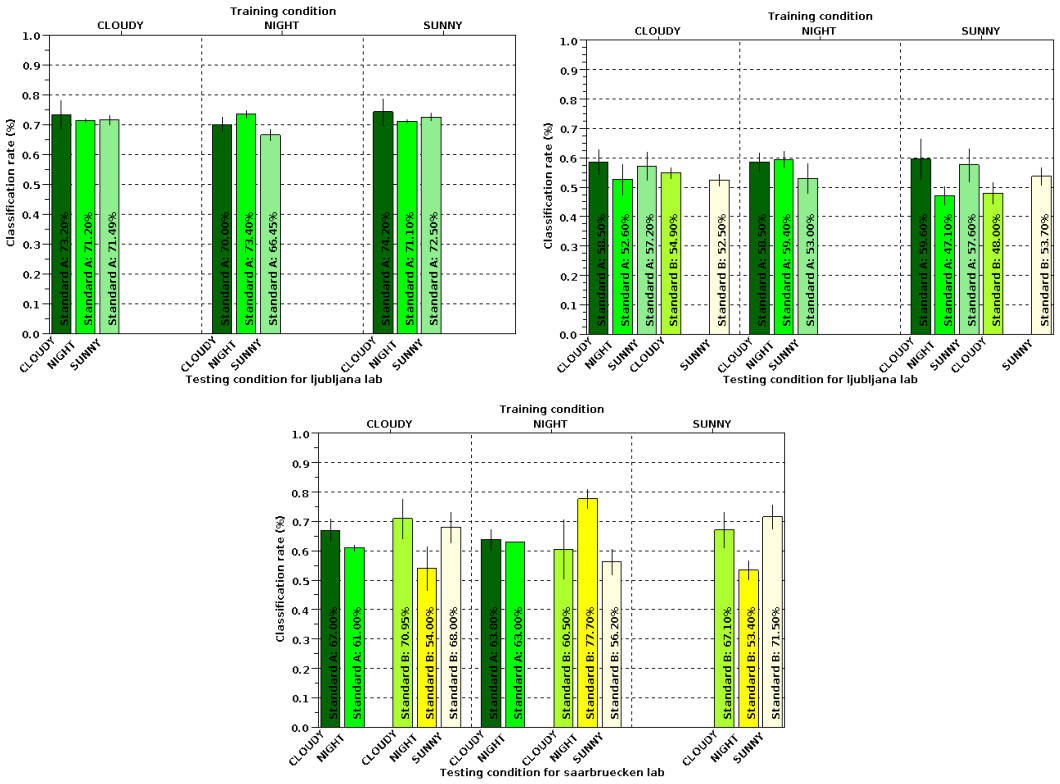

Figure 5.4: Average classification rates from the three different laboratories. The extracted features in this case are obtained using a learning rate of 0.001 and a weight decay of 0.0001.

We have started by decreasing the learning rate and the weight cost to 0.001 and 0.0001 respectively. After obtaining the code, the classification process was performed in the feature space as shown in figure 5.4. Unfortunately, the results seem to be similar for COLD-Ljubljana laboratory and worse for COLD-Freiburg and COLD-Saarbruecken laboratories compared with the experiment shown in figure 5.2. We thought that we have obtained these worse results because of decreasing the weight decay, thus we kept the learning rate unchanged and we increased the weight cost to 0.0008. However, the results shown in figure 5.5 are still very close to the results obtained with a weight cost of 0.0001. This underlines that these factors do not improve the sparsity and thus they did not have any real impact on changing the final classification results.

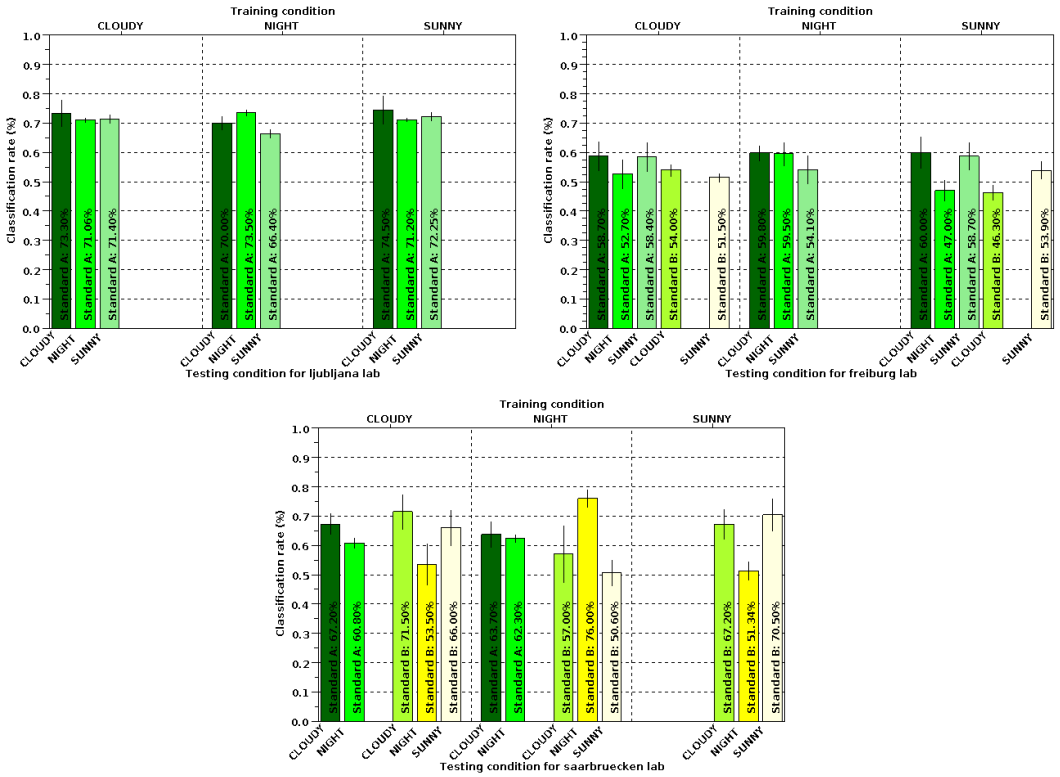

Figure 5.5: Average classification rates from the three different laboratories. The extracted features in this case are obtained using a learning rate of 0.001 and a weight decay of 0.0008.

Based on these experiments, we can observe two facts: First, we noted that in the case of the COLD database, using a learning rate ranging from 0.001 to 0.005 does not affect or change the features too much. The number and types of features were quite similar. The only difference is that the network will converge faster if we use a larger learning rate, however, after a lot of epochs, any learning rate value in the above range will let the network converge towards similar features. Secondly, We have seen that using a weight decay ranging from 0.0001 to 0.0008 does not change the results. This indicates that any value in this range is sufficient to penalize large values that could happen during the learning process.

However, by adding a penalty term which particularly aims at encouraging the sparsity on hidden activation units, the final average classification scores are significantly increased, as shown in figure 5.6. In this experiment, we obtained an average of correct answers ranging from 65% to 80% according to the different conditions and laboratories as shown in figure 5.6. It has been shown that adding this term effectively improves

the sparsity of the data representation through encouraging hidden unit activations to be sparse [Hinton, 2010; Lee et al., 2008, 2009; Mairal et al., 2008; Olshausen and Field, 1996, 1997]. More explanation of this term can be found in chapter 3.

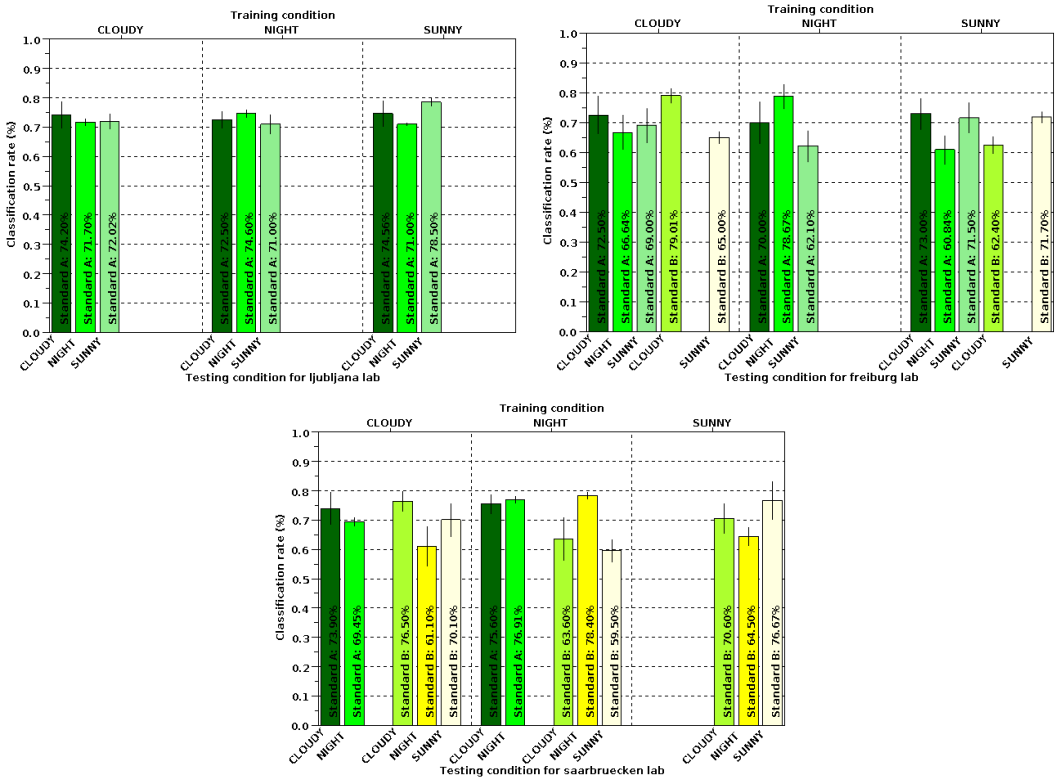

Figure 5.6: Average classification rates from the three different laboratories. The extracted features in this case are obtained using the same CD parameters but with a penalty term.

Compared to the previous experiments shown in figure 5.2, the average results of correct classification were: 69.9%, 57.5%, and 60% for COLD-Ljubljana, COLD-Freiburg, and COLD-Saarbruecken laboratories respectively and with an overall average of correct answers of 61.5% for the three laboratories. However, the average results shown in figure 5.6 are: 73.4%, 69.5%, and 71% for the same three laboratories and with an overall average of correct answers of 71.3%. This means that we have managed to successfully raise the classification results by 10% which become closer to the state-of-the-art results. This underlines an important fact that adding the penalty term has effectively improved the quality of the image coding.

Another possible way is to investigate the effect of normalization on the final classification results. We have seen in the previous chapter using data whitening or local

normalization had some influences on features extraction. Thus, investigating this factor would also have some impact on the classification process. In the next section, we study the classification process using features extracted from the normalized data.

5.4 Role of normalization on the classification

All previous experiments have been conducted using the features learned by training two RBM layers on the whitened data. However, during our work we have observed that learning features from a non-whitened data would play an important role in improving the classification results. Thus, after using the features shown in figure 4.14 to train the second RBM layer, the real-valued output of the second RBM units is used to perform the classification as shown in figure 5.7 using a softmax regression.

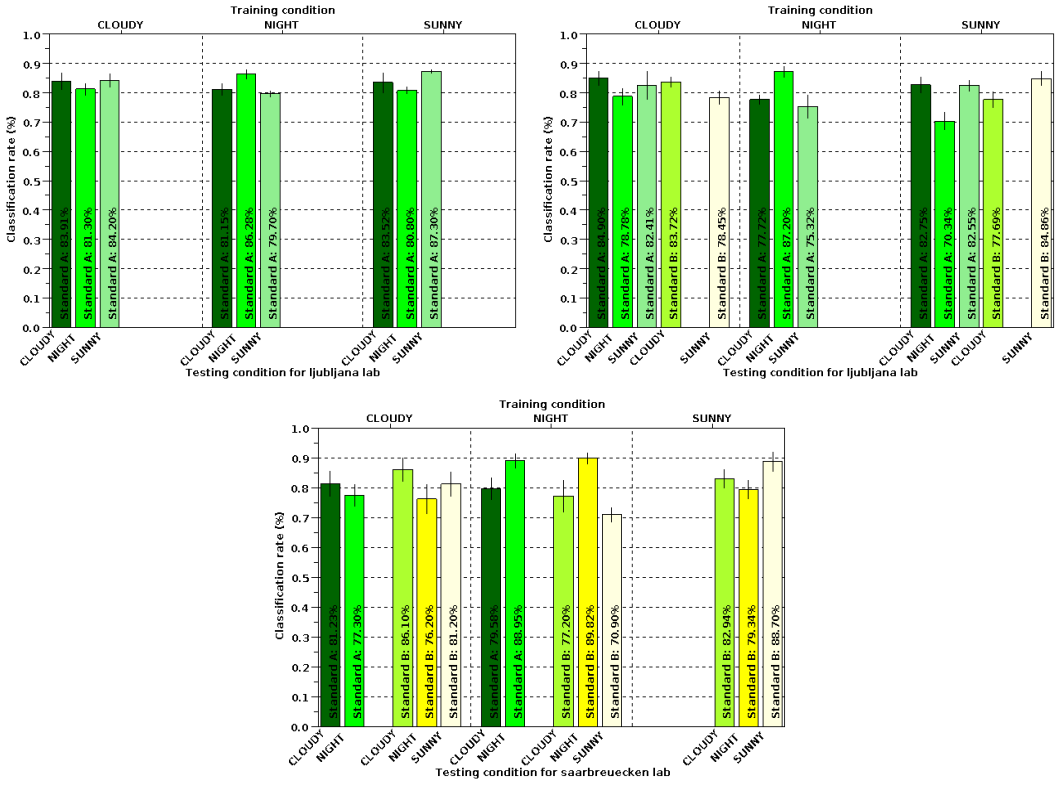

Figure 5.7: Average classification rates from the three different laboratories. These results have been achieved by training two RBM layers on the normalized COLD data.

In this case, we obtain an average of correct answers ranging from 71% to 90%

according to the different conditions and laboratories. Compared to our previous results shown in figure 5.6, these classification results seem to be more competitive to the state-of-the-art. More precisely, the average results of correct classification are: 83.13%, 80.515%, and 81.50% for COLD-Ljubljana, COLD-Freiburg, and COLD-Saarbruecken laboratories respectively and with an overall average of correct answers of 81.375% for the three laboratories. This indicates that we have improved the results by 10% to 11% for the different laboratories. They are then at the level of the best published ones [Ullah et al., 2008]. The results remain robust to illumination variations as in [Ullah et al., 2008]. We underline once again the lower performance on the COLD-Freiburg dataset. However, in this case, our results outperform the results obtained in [Ullah et al., 2008].

These results demonstrate that the features learned from a data normalization are more beneficial for our classification problem. It illustrates the fact that the normalization process keeps much more information or structures of the initial views which are very important for the classification process. On the other hand, data whitening completely removes the first and second order statistics from the initial data which allows DBNs to extract higher-order features. This demonstrates that data whitening could be useful for image coding. However, it is not the optimal pre-processing method in the case of image classification. This is in accordance with the results in the literature showing that first and second order statistics based features are significantly better than higher order statistics in terms of classification [Aggarwal and Agrawal, 2012; Krizhevsky, 2010].

However, two different ways are still open to improve these results. The first one is to use temporal integration, as proposed in [Guillaume et al., 2011]. The second one is presented in the next section and relies on decision theory.

5.5 Image rejection

Usually, in any video sequence taken during robot exploration, some of the images are non informative especially when the robot faces a wall or when it turns or moves too fast as noted in the case of the Freiburg laboratory. The main justification of using a rejection mechanism is therefore to discard these blurred images from the classification process.

The detection rate presented in figure 5.6 has been computed from the classes with the highest probabilities, irrespective of the relative values of these probabilities. Some of them are close to the chance (in our case 0.20 or 0.25 depending on the number of categories to recognize) and it is obvious that, in such cases, the confidence in the decision made is weak. Thus below a given threshold, when the probability distribution tends to become uniform, one could consider that the answer given by the system is meaningless. This could be due to the fact that the given image contains common characteristics or structures that can be found in two or more classes. As shown in figure 5.8, the effect of the threshold is then to discard the most uncertain results. Figure 5.9 (first column) shows the average classification results for a threshold of 0.55 (only the results where $\max_X p(X = c_k|I) \geq 0.55$, where $p(X = c_k$ is the probability that the current view I belongs to c_k, are retained). These results have been achieved using the features extracted from the whitened data.

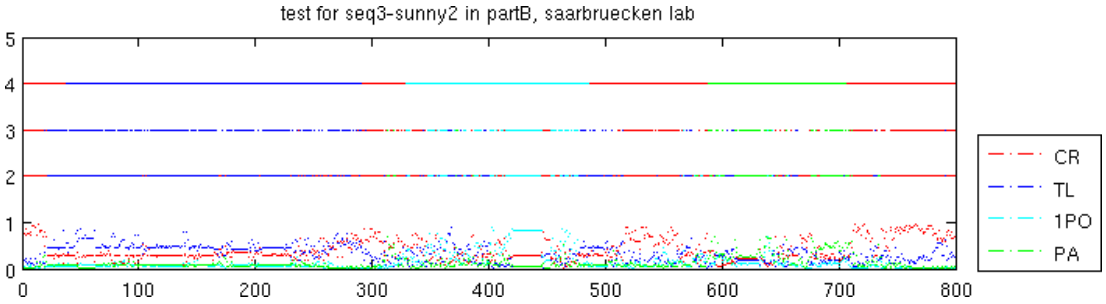

Figure 5.8: A comparison between the classification results with and without a threshold. This test has been done for a subset of images selected from Saarbruecken laboratory, partB. First level represents the actual probabilities of the four different classes (corridor (CR), toilet (TL), one person office (1PO), and printer area (PA)). Second and third levels represent classifications based on a softmax regression without and with a threshold respectively. Finally, the forth level represents the original correct classification.

In this case, the average acceptance rate (the percentage of considered examples) ranges from 75% to 85%, depending on the laboratory, and the average results show values that outperform the best published ones [Ullah et al., 2008]. When considering all the results obtained by training and testing on similar illumination conditions, we got an average classification rate of 90.68% for COLD-Saarbrucken laboratory, 89.88% for COLD-Freiburg laboratory and 90.66% for COLD-Ljubljana laboratory. Similarly to [Ullah et al., 2008] results, the performance has also decreased in case of the experiments under varying illumination conditions. In this case we have achieved

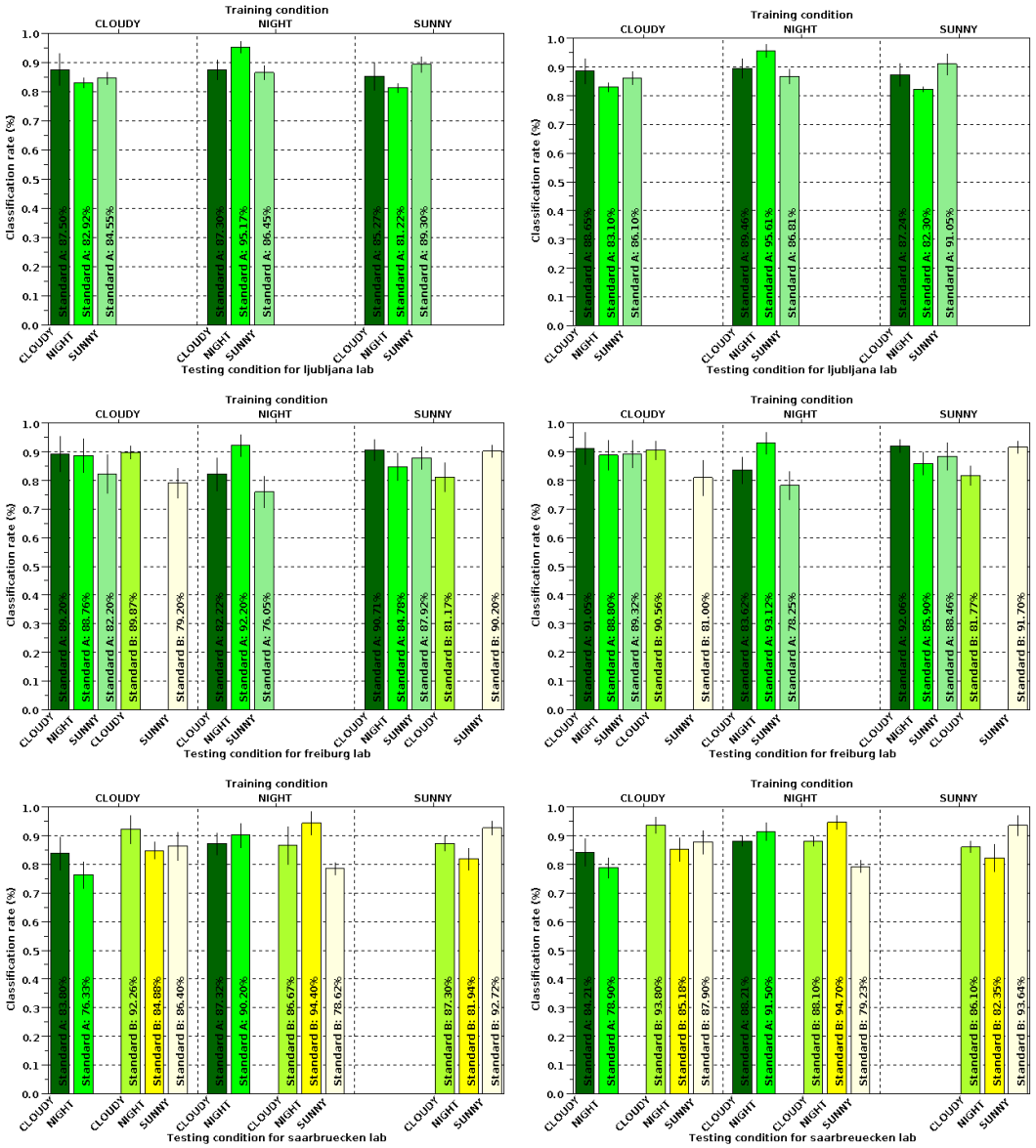

Figure 5.9: Average classification rates from the three different laboratories with a threshold of 0.55. **First column:** classification rates that have been obtained based on the features extracted from the whitened data. **Second column:** classification rates that have been obtained based on the features extracted from the normalized data.

classification rates of 83.683% for COLD-Saarbrucken laboratory, 83.14% for COLD-Freiburg laboratory and 84.62% for COLD-Ljubljana laboratory. However, our results are less sensitive to the illumination conditions than the results obtained in [Ullah et al., 2008]. As in previous experiments, we noted the weaker performance on the COLD-

Freiburg data, which confirms that this collection is the most challenging of the whole COLD database as indicated in [Ullah et al., 2008]. However, with and without threshold, our classification results for this laboratory outperforms the best ones obtained by [Ullah et al., 2008].

Similarly, we have also applied the threshold method on the results obtained in figure 5.7 with locally normalized data. Figure 5.9 (second column) shows the average classification results using a similar threshold (0.55). In this case, the average rate of acceptance examples increases to be between 86% to 90%, depending on the laboratory, showing that more examples are used in the classification than the former experiment. Also, the average results, in this case, show scores that strongly outperform the best published one [Ullah et al., 2008]. This indicates that the linear separability of the data was significantly improved in the case of using the normalized data for features extraction.

Concerning the sensitivity to illumination for both cases, our results seem to be less sensitive to the illumination conditions compared to the results obtained in [Ullah et al., 2008]. As in previous experiments, we noted the lower performance on the COLD-Freiburg data, which confirms that this collection is the most challenging of the whole COLD database as indicated in [Ullah et al., 2008]. However, in case of using features learned from the un-whitened data, with and without threshold our classification results for this laboratory outperforms the best ones obtained by [Ullah et al., 2008].

Tables 1 and 2 show an overall comparison of our results with those from [Ullah et al., 2008] for the three training conditions in a more synthetic view. It also shows the results obtained using a SVM classification instead of a softmax regression. The results are quite comparable to softmax showing that the DBN computes a linearly separable signature. They underline the fact that features learned by DBNs approach are more robustness for a semantic place recognition task than the extraction of *ad hoc* features based on (GiST, CENTRIST, SURF, and SIFT detectors).

Laboratory name	Saarbruecken			Freiburg			Ljubljana		
Condition / Training	Cloudy	Night	Sunny	Cloudy	Night	Sunny	Cloudy	Night	Sunny
Ullah	84.20%	86.52%	**87.53%**	79.57%	75.58%	77.85%	84.45%	87.54%	**85.77%**
No thr.	70.21%	70.80%	70.59%	70.43%	70.26%	67.89%	72.64%	72.70%	74.69%
SVM	69.92%	71.21%	70.70%	70.88%	70.46%	67.40%	72.20%	72.57%	74.93%
0.55 thr.	**84.73%**	**87.44%**	87.32%	**85.85%**	**83.49%**	**86.96%**	**84.99%**	**89.64%**	85.26%

Table 5.1: Average classification results for the three different laboratories and the three training conditions. **First row:** Ullah's work; **second row:** rough results without threshold; **third row:** classification rates using SVM classifier; **fourth row:** classification rates with threshold as indicated in text. Our results have been obtained based on the features learned from the whitened data.

Laboratory name	Saarbruecken			Freiburg			Ljubljana		
Condition / Training	Cloudy	Night	Sunny	Cloudy	Night	Sunny	Cloudy	Night	Sunny
Ullah	84.20%	86.52%	**87.53%**	79.57%	75.58%	77.85%	84.45%	87.54%	85.77%
No thr.	80.41%	81.29%	83.66%	**81.65%**	**80.08%**	**79.64%**	83.14%	82.38%	83.87%
0.55 thr.	**86.00%**	**88.35%**	87.36%	**88.15%**	**85.00%**	**87.98%**	**85.95%**	**90.63%**	**86.86%**

Table 5.2: Average classification results for the three different laboratories and the three training conditions. **First row:** Ullah's work; **second row:** rough results without threshold; **third row:** classification rates with threshold as indicated in text. Our results have been obtained based on the features learned from the normalized data.

5.6 Summary

In the present chapter we have presented our experiments on SPR and image classification using the DBN we have designed in the previous chapters. Concerning the classification, our system was tested using two different classification methods (linear with softmax and nonlinear with SVM). We observed that the results of the nonlinear classifier are quite comparable to the softmax regression results, suggesting that the DBN computes a linearly separable signature. We have also observed that adding a penalty term improved the quality of image coding and thus increased the classification scores.

We also investigated the effect of normalization on the classification process. We saw that extracting features from a locally normalized database that covers a larger range of spatial frequencies gave significantly better classification results. Compared to the state-of-the-art [Guillaume et al., 2011; Ullah et al., 2008], they are in the same range for COLD-Saarbrucken and COLD-Ljubljana laboratories and they outperformed the results for COLD-Freiburg laboratory.

Finally, we introduced a method to discard the most uncertain images and we show that even with a small rejection rate the classification results are significantly improved and largely outperformed the state-of-the-art. This last way to perform SPR relates to the observation that to recognize a specific location it could not be necessary to recognize all the views within a data flow but only the most statistically significant ones. The reached confidence level in this case could be very high. We can push forward this hypothesis saying that recognizing a unique but specific detail characterizing a place could be sufficient to recognize it. Future approaches could be based on such considerations.

Chapter 6

Conclusions and future works

6.1 Summary of contributions

The aim of this thesis was to study the use of Deep Belief Networks in a challenging image recognition task, View-based Semantic Place Recognition. DBNs have been widely used to learn high-level feature representations that can be successfully applied in a wide spectrum of application domains, including in particular image retrieval, classification and regression tasks, as well as nonlinear dimensionality reduction. We proposed here to use them as a novel approach to achieve robot SPR. The most significant characteristics behind learning deep generative models are as follows :

- Multiple layers of representation that can be trained in a greedy layer-wise by training one layer at a time.
- The greedy learning carried out in a completely unsupervised way.
- Their ability to learn sparse efficient features and perform non-linear dimensionality reduction that simplify the classification.
- The theoretical grouding of the feature space construction that outperforms the empirical building of sets of descriptors.

The first part of the thesis focused on introducing the problem of SPR in robotics systems. Thus, in chapter 2 we provided a detailed overview of the different coding

and classification methods that have been used to solve the problem of SPR. In particular, we showed that, in the framework of the view-based approaches, the problem of SPR first requires an appropriate code of the initial data. Such a code can be provided by DBNs with the advantage to be problem independent and to be theoretically grounded. We then provided a detailed description of DBNs and their building modules, Restricted Boltzmann Machines (RBMs), along with its most popular learning algorithm, Contrastive Divergence (CD). We have seen that although Boltzmann Machines (BMs) and RBMs have been introduced as early as in the 80's, the wide use of them had to wait until [Hinton, 2002] who introduced CD learning. The main barrier in the acceptance of RBMs was the difficulty in computing the stochastic gradient for training the model. Thanks to CD learning, the popularity of RBM and its variants grew rapidly, and a whole field opened [Bengio, 2009] in the early 2000s.

Since different parameter settings strongly influence the quality and the nature of the obtained feature space, we have developed an extensive parameter study presented in chapter 3. This study allowed us to precise the role of the network structure and to discuss the question of over-completeness. We also focused on the parameters influencing the locality of the obtained features. This locality can increases the sparsity of the network, its ability to activate only a few units to code for an image (spatial sparsity) and to activate a given unit only rarely over time (time sparsity).

The effect of whitening compared to local normalization was also studied in this chapter. Our studies allow to draw an interesting conclusion about the spatial frequency representation with the two modes of normalization. While whitening equalizes the Fourier power spectral density and thus the autocorrelation of the signal, the local normalization equalizes the energy included in each frequency band (each octave). The obtained features in this case cover a broader range of spatial frequencies suggesting that the energy of the signal plays the most important role in the emergence of the features during RBMs learning.

Chapter 4 studies the use of tiny images for classification through the dimensionality reduction ability of the DBNs. The impact of image centering was studied with aligned and non aligned images and it is shown that the plasticity of the RBM learning algorithm allows to build different feature spaces in these case with a higher semantic level for the aligned dataset. This shows that the composition of the database plays the

most prominent role in the nature of the features that will be obtained. Obtaining localized low-level features focused on non specific edges is not a property of the RBM algorithm but depends on the used databases. To shed light on this point we can argue that the statistically independent components of an image could be higher level details if these details are aligned in the initial dataset. This can be related to the role of attentional mechanisms in the acquisition of an optimal image coding. Without attention, looking at random to the scene, the obtained feature space is made of the localized low-level edges of the images. With the attention to very frequent objects or image characteristics (like faces or familiar objects) a DBN network can easily mimick what happens in the primate visual cortex, the selection of detectors specific of the structural details of these frequent characteristics (*e.g.* parts of faces).

Chapter 5 focused on performing extensive classification experiments to show the performance of the proposed model. An approach based on tiny images followed by a projection onto an appropriate feature space can achieve good classification results in a semantic place recognition task. They outperformed the best published ones [Ullah et al., 2008] based on more complex techniques (use of Scale Invariant Feature Transform (SIFT) detectors followed by a Support Vector Machine (SVM) classification). As we expected, the classification results were significantly better when we used the features learned from a locally normalized dataset. It can be argued that first and second order statistics based features are significantly better than higher order statistics in terms of classification as already stated by [Aggarwal and Agrawal, 2012]. However, to recognize a place it seems not necessary to correctly classify every image of the place. With respect to place recognition not all the images are informative: some of them are blurred when the robots turns or moves too fast from one place to another, some others show no informative details (*e.g.* when the robot is facing a wall). As the proposed system computes the probability of the most likely room among all the possible rooms, it offers the way to weight each conclusion by a confidence factor associated with the probability distribution over all classes. We can then discard the most uncertain views thus increasing the recognition score. It offers a simpler alternative to the method proposed in [Pronobis and Caputo, 2007] based on cue integration and the computation of a confidence criterion in a SVM classification approach.

The fundamental contribution of this work is therefore the demonstration that DBNs coupled with tiny images can be successfully used in the context of semantic place

recognition. These considerations have greatly contributed in simplifying the overall classification algorithm. They indeed provide coding vectors that can be used directly in a discriminative method. To our knowledge this is the first demonstration that tiny images feature extraction using DBN is an alternative approach for SPR that deserves to be considered.

6.2 Future works and open questions

There are several potential extensions and applications of the ideas presented in this thesis, particularly related to learning DBNs.

Convolutional deep belief networks. As we have stated before, scaling such models to full-sized, high-dimensional images remains a difficult problem for DBNs. However, very recent works have addressed this problem (see for instance, [Lee et al., 2009; Norouzi et al., 2009]) through the use of convolutional operator. In particular, they proposed to use a probabilistic max-pooling, a technique which shrinks the representations of higher layers in a probabilistically sound way. They have shown that the algorithm learns useful high-level visual feature and led to excellent performance on visual recognition tasks. Therefore, we plan to apply this model on the COLD database and investigate their features extraction. However, for the classification process, we need to use the small tiny images, otherwise the classification becomes very expensive.

Sparse-overcomplete representations. This idea has already been investigated in this thesis in terms of features extraction. We have seen that the use of overcomplete structures for DBNs did not improve the features themselves for some databases. However, the number of extracted features was different as in the case of LFW database experiment. Sparse-overcomplete representations have a number of theoretical and practical advantages, as demonstrated in a number of recent studies [Doi et al., 2006; Olshausen and Field, 1997; Ranzato et al., 2007a]. In particular, they have good robustness to noise, and provide a good tiling of the joint space of location and frequency. These representations can be

advantageous for problems of classifications, such as the problem SPR, because they will allow to have more features of the different places.

Place categorization. A question that has not been investigated in this work and that remain open despite some interesting attempts [Guillaume et al., 2011; Ullah et al., 2008] is the view-based categorization of places. The work presented in this thesis concerns instance classification. Categorization is the way to recognize the functional nature of a room, for example with the COLD database the recognition of an office or a corridor from the different labs. The view-based approaches usually give very poor results (usually around 25% of recognition). Although we think that this problem is fundamentally ill-posed and that categorization of functional classes like printer area or kitchen must be rather made on the basis of the recognition of their functions, it could be interesting to see if an approach based on DBNs is able to improve these results.

Object recognition. As previously said, DBNs have the ability to learn layers of feature detectors that become progressively more complex, which is thought to be a promising model to address the problem of object recognition. However, currently, most of the existing object recognition systems that achieve state-of-the-art results are based on hand coded methods like GiST, CENTRIST, SURF, and SIFT detectors (see for instance [Guillaume et al., 2011; Ullah et al., 2008] and include many hand-crafted features. On the light of what was done in this thesis for SPR, it could be interesting to evaluate the performance of DBNs on these object recognition tasks.

We have outlined several potential research works for the future. However, research on deep learning is still new and there are a lot of open questions that have not been considered yet [Yu et al., 2009]. Some of them are: Can we develop better optimization or approximation techniques that would allow us to learn deep models more efficiently without significant human intervention? Can we develop algorithms that are capable of extracting high-level feature representations that can be transferred to unknown future tasks? Under what conditions does the feature hierarchy achieve a better regularization or statistical efficiency? How can we make deep models more robust

to deal with highly ambiguous or missing sensory inputs? We believe that answering these questions will allow and facilitate the emergence of more intelligent machines.

Finally, the system presented in this thesis could also be successfully applied to mobile robot platform with limited memory and processing resources. In particular, semantic place recognition can be used to guide the robot navigation. It seems not too difficult to use the classification tool after off-line learning of both the feature space and the classification space. The fact that these learning steps are not well-suited for on-line learning is one of the major drawback. However, it could be very interesting to study the features that emerge from on-line DBN learning when the images are provided during the exploration of its environment by the robot. In particular, the impact of two different situations, the free random acquisition of images and the acquisition of images driven by an attentional mechanism, will be interesting to study.

6.3 Publications

6.3.1 Posters and oral presentations

Part of the work presented in this thesis has been also presented in the following events:

- Ahmad Hasasneh, Emmanuelle Frenoux, and Philippe Tarroux (2010, July). Medical Image Segmentation Using New Machine Learning Methods: a Prospective Study. Poster session presented at the BMVA Summer School on Computer Vision, 2010, Kingston University, London, UK.

- Ahmad Hasasneh, Emmanuelle Frenoux, and Philippe Tarroux (2012, February). Semantic place recognition using tiny images and deep belief networks. Oral presentation in SIG-TAO meetings, 2012, LRI Lab, Paris SUD University, Paris, France.

6.3.2 International conferences

Part of the work presented in this thesis has been published in the following international conference:

- Ahmad Hasasneh, Emmanuelle Frenoux, and Philippe Tarroux (2012, July). Semantic Place Recognition Based on Deep Belief Networks and Tiny Images. In 9^{th} *International Conference on Informatics in Control, Automation and Robotics ICINCO*, 2012, Rome, Italy.

6.4 Closing remarks

This thesis has explored the problem of semantic place recognition for autonomous systems. As one solution for this problem, I proposed to use DBNs approaches that exploit sparsity and locality, while demonstrating good performance in many AI problems. Given that the quality of features significantly affects the performance of image classification. We have seen that our approach obtains scores comparable to approaches based on computer vision methods (like the use of SIFT detectors) and more sophisticated classification techniques like SVM. As emphasized by [Hinton et al., 2011], it illustrates the fact that features extracted by DBN algorithms are more promising for image classification than hand-engineered features. I believe that such algorithms will allow machine learning systems to be much more easily applied to problems in vision, text understanding, audio understanding, and other problems, and to achieve superior performance without the manual feature engineering while using significantly less labeled data.

Appendices

This section demonstrates the derivation of the most important equations.

Appendix A This appendix presents the derivation of the sigmoid function for an RBM and general BM [Krizhevsky, 2009]. An RBM with **V** visible units and **H** hidden units is governed by the following energy function:

$$E(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} c_j h_j,$$

where Agence franaise pour la promotion de lenseignement suprieur

- **v** is the binary state vector of the visible units,
- **h** h is the binary state vector of the hidden units,
- v_i is the state of visible unit i,
- h_j is the state of hidden unit j,
- w_{ij} is the real-valued weight between visible unit i and hidden unit j,
- b_i is the real-valued bias into visible unit i,
- c_j is the real-valued bias into hidden unit j.

According to Gibbs distribution, a probability is associated with configuration (**v**,**h**) is given as follows:

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

where Z is a normalizing constant. Thus after marginalization:

$$P(\mathbf{v}) = \sum_{\mathbf{g}} P(\mathbf{v}, \mathbf{g})$$

We can also derive some simple conditional expressions:

$$P(\mathbf{h}|\mathbf{v}) = \frac{P(\mathbf{v}, \mathbf{h})}{P(\mathbf{v})} = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{\sum_g e^{-E(\mathbf{v},\mathbf{g})}}$$

As illustrated in [Krizhevsky, 2009], it can also drive closed-form expression for $P(h_k = 1|\mathbf{v})$, the probability of a particular hidden unit being on given a visible configuration.. To do this, they introduced the notation,

$$P(h_k = 1, \mathbf{h}_{j \neq k}, \mathbf{v})$$

to denote the probability of the configuration in which hidden unit k has state 1, the rest of the hidden units have state $\mathbf{h}_{j \neq k}$, and the visible units have state $\mathbf{v}$. Given this,

we have:

$$
\begin{aligned}
P(h_k=1|\mathbf{v}) &= \frac{P(h_k=1,\mathbf{v})}{P(\mathbf{v})} \\
&= \frac{\sum_{\mathbf{h}_{j\neq k}} P(h_k=1,\mathbf{h}_{j\neq k},\mathbf{v})}{P(\mathbf{v})} \\
&= \frac{\frac{\sum_{\mathbf{h}_{j\neq k}} e^{-E(h_k=1,\mathbf{h}_{j\neq k},\mathbf{v})}}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{v})}}}{\frac{\sum_{\mathbf{g}} e^{-E(\mathbf{v},\mathbf{g})}}{\sum_{\mathbf{u}}\sum_{\mathbf{g}} e^{-E(\mathbf{u},\mathbf{v})}}} \\
&= \frac{\sum_{\mathbf{h}_{j\neq k}} e^{-E(h_k=1,\mathbf{h}_{j\neq k},\mathbf{v})}}{\sum_{\mathbf{g}} e^{-E(\mathbf{v},\mathbf{g})}} \\
&= \frac{\sum_{\mathbf{h}_{j\neq k}} e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)+\left(\sum_{j\neq k}^{H}\sum_{i=1}^{V}(v_i h_j w_{ij}+c_k)+\sum_{i=1}^{V} v_i b_i+\sum_{j\neq k}^{H} h_j c_j\right)}}{\sum_{\mathbf{g}} e^{-E(\mathbf{v},\mathbf{g})}} \\
&= \frac{\left[e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right]\sum_{\mathbf{h}_{j\neq k}} e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)+\left(\sum_{j\neq k}^{H}\sum_{i=1}^{V}(v_i h_j w_{ij}+c_k)+\sum_{i=1}^{V} v_i b_i+\sum_{j\neq k}^{H} h_j c_j\right)}}{\sum_{\mathbf{g}} e^{-E(\mathbf{v},\mathbf{g})}} \\
&= \frac{\left[e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right]\sum_{\mathbf{h}_{j\neq k}} e^{-E(h_k=0,\mathbf{h}_{j\neq k},\mathbf{v})}}{\sum_{\mathbf{g}} e^{-E(\mathbf{v},\mathbf{g})}} \\
&= \frac{\left[e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right]\sum_{\mathbf{h}_{j\neq k}} e^{-E(h_k=0,\mathbf{h}_{j\neq k},\mathbf{v})}}{\sum_{\mathbf{g}_{j\neq k}} e^{-E(g_k=1,g_{j\neq k},\mathbf{v})}+\sum_{\mathbf{g}_{j\neq k}} e^{-E(g_k=0,g_{j\neq k},\mathbf{v})}} \\
&= \frac{\left[e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right]\sum_{\mathbf{h}_{j\neq k}} e^{-E(h_k=0,\mathbf{h}_{j\neq k},\mathbf{v})}}{\left[e^{\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right]\sum_{\mathbf{g}_{j\neq k}} e^{-E(g_k=1,g_{j\neq k},\mathbf{v})}+\sum_{\mathbf{g}_{j\neq k}} e^{-E(g_k=0,g_{j\neq k},\mathbf{v})}} \\
&= \frac{1}{1+e^{-\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}} \\
&= \left(1+e^{-\left(\sum_{i=1}^{V} v_i w_{ik}+c_k\right)}\right)^{-1} \\
\implies \quad & P(h_k=1|\mathbf{v}) = \sigma(c_k+\sum_i v_i w_{ik})
\end{aligned}
$$

Appendix B The derivation of the softmax regression for multinomial classification problem [Ng, 2011].

Unlike logistic regression, in softmax regression we have a multi-nomial classification problem, so first let us define $y \in \{1,2,\ldots,k\}$ of k different classes. We also need to define the model parameters over k possible outcomes as follow: $\phi_1, \phi_2, \ldots, \phi_k$. In other words, these parameters specify the probability of each outcome as follows:

$$\phi_i = P(y = i)$$

and

$$\phi_k = 1 - \left(\phi_1 + \phi_2 + \cdots + \phi_{k-1}\right)$$

it means that we need to process $k-1$ parameters. To express the multinomial as an exponential family distribution, we need to define $T(y) \in \mathbb{R}^{k-1}$ as follows:

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, T(2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, T(3) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, T(k-1) = \begin{bmatrix} 0 \\ p0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, T(k) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

We will therefore write $\left(T(y)_i\right)$ to denote the i^{th} elements of the vector $T(y)$. For notational convenience, the relationship between $T(y)$ and y can be expressed as:

$$\left(T(y)_i\right) = 1\{y = i\}$$

where $1\{\cdot\}$ is the usual definition of the indicator function which takes a value of 1 if its argument is true, and 0 otherwise. The expectation of $T(y)$ can be defined as:

$$E\left(T(y)_i\right) = P(y = i) = \phi_i$$

Using these definitions, it is possible now to define the multinomial as exponential distribution as follows:

$$\begin{aligned}
P(y=i) &= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\ldots\phi_k^{1\{y=k\}} \\
&= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\ldots\phi_k^{1-\sum_{i=1}^{k-1}1\{y=k\}} \\
&= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}\ldots\phi_k^{1-\sum_{i=1}^{k-1}\left(T(y)\right)_i} \\
&= e^{\left(\left(T(y)\right)_1 log(\phi_1)+\left(T(y)\right)_2 log(\phi_2)+\cdots+\left(1-\sum_{i=1}^{k-1}\left(T(y)\right)_i\right)log(\phi_k)\right)} \\
&= e^{\left(\left(T(y)\right)_1 log(\phi_1/\phi_k)+\left(T(y)\right)_2 log(\phi_2/\phi_k)+\cdots+\left(T(y)\right)_{k-1} log(\phi_{k-1}/\phi_k)+log(\phi_k)\right)} \\
&= b(y)e^{\left(\eta^T T(y)-a(\eta)\right)}
\end{aligned}$$

where

$$\eta = \begin{bmatrix} log(\phi_1/\phi_k) \\ log(\phi_2/\phi_k) \\ \vdots \\ log(\phi_{k-1}/\phi_k) \end{bmatrix},\ a(\eta) = -log(\phi_k),\ b(y) = 1.$$

The above formulations confirm that the multinomial can be expressed as an exponential family distribution. For more convenience, the link function is given (for $i = 1,\ldots,k$) by

$$\eta_i = log\frac{\phi_i}{\phi_k}$$

by taking the exponential for both sides, we can then get

$$\begin{aligned}
e^{\eta_i} &= \frac{\phi_i}{\phi_k} \\
\implies \phi_k e^{\eta_i} &= \phi_i \\
\implies \phi_k \sum_{i=1}^{k} e^{\eta_i} &= \sum_{i=1}^{k}\phi_i = 1
\end{aligned}$$

This implies that $\phi_k = 1/\sum_{i=1}^{k} e^{\eta_i}$, and it can be substituted in the above equation to give:

$$\phi_i = \phi_k e^{\eta_i} = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}$$

but

$$\begin{aligned} \phi_i &= P(y = i|x;\theta) \\ &= \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} \\ &= \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}} \end{aligned}$$

This model is called softmax regression which can be used to perform multi-class classification problems, *i.e.*, $y \in \{1,2,\ldots,k\}$. This model is a generalization of the logistic regression.

Appendix C This appendix demonstrates the partial derivative of the log-likelihood function for Product of Experts (PoEs) models [Wood and Hinton, 2012].

Usually, PoEs combines n individual models by taking the product of their conditional probabilities and normalizing the result using $Z(\theta)$, as follows [Hinton, 2002]:

$$\begin{aligned} P(X \mid \theta_1, \ldots, \theta_n) &= \prod_{x \in X} \frac{\prod_m P_m(x \mid \theta_m)}{Z(\theta_m)} \\ &= \prod_{x \in X} \frac{\prod_m P_m(x \mid \theta_m)}{\sum_y \prod_m P_m(y \mid \theta_m)} \end{aligned}$$

The log-likelihood of the previous equation can be written as follows:

$$logP(X \mid \theta_1, \ldots, \theta_n) = log \prod_{x \in X} \frac{\prod_m P_m(x \mid \theta_m)}{\sum_y \prod_m P_m(y \mid \theta_m)}$$

also, the gradient of the likelihood can be defined with respect to the model parameters θ_m as follows:

$$\partial \frac{logP(X \mid \theta_1, \ldots, \theta_n)}{\partial \theta_m} = \frac{\partial}{\partial \theta_m} log \prod_{x \in X} \frac{\prod_m P_m(x \mid \theta_m)}{\sum_y \prod_m P_m(y \mid \theta_m)}$$

now, it is possible to multiply both sides of the previous equation by $1/N$ and we then get:

$$\begin{aligned} \frac{1}{N} \partial \frac{logP(X \mid \theta_1, \ldots, \theta_n)}{\partial \theta_m} &= \frac{1}{N} \frac{\partial}{\partial \theta_m} log \prod_{x \in X} \frac{\prod_m P_m(x \mid \theta_m)}{\sum_y \prod_m P_m(y \mid \theta_m)} \\ &= \frac{1}{N} \sum_{x \in X} \frac{\partial logP_m(x \mid \theta_m)}{\partial \theta_m} - \frac{1}{N} \sum_{x \in X} \frac{\partial log \sum_y \prod_m P_m(y \mid \theta_m)}{\partial \theta_m} \\ &= \frac{1}{N} \sum_{x \in X} \frac{\partial logP_m(x \mid \theta_m)}{\partial \theta_m} - \frac{\partial log \sum_y \prod_m P_m(y \mid \theta_m)}{\partial \theta_m} \\ &= \left\langle \frac{\partial logP_m(x \mid \theta_m)}{\partial \theta_m} \right\rangle_{Q^0} - \frac{\partial log \sum_y \prod_m P_m(y \mid \theta_m)}{\partial \theta_m} \end{aligned}$$

however, we know that $log(x)' = x'/x$. It means that $\partial log \sum_y \prod_m P_m(y \mid \theta_m)/\partial\theta_m = \left(\partial \sum_y \prod_m P_m(y \mid \theta_m)/\partial\theta_m\right) * \left(1/\sum_y \prod_m P_m(y \mid \theta_m)\right)$ and thus the previous equation can be rewritten as follows:

$$\begin{aligned}\frac{1}{N}\partial\frac{logP(X \mid \theta_1,\ldots,\theta_n)}{\partial\theta_m} &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \frac{1}{\sum_y \prod_m P_m(y \mid \theta_m)} * \frac{\partial \sum_y \prod_m P_m(y \mid \theta_m)}{\partial\theta_m} \\ &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \frac{1}{\sum_y \prod_m P_m(y \mid \theta_m)} * \frac{\sum_y \prod_m \partial P_m(y \mid \theta_m)}{\partial\theta_m}\end{aligned}$$

remember, $\partial P_m(y \mid \theta_m)/\partial\theta_m = P_m(y \mid \theta_m) * \partial logP_m(y \mid \theta_m)/\partial\theta_m$, so that:

$$\begin{aligned}\frac{1}{N}\partial\frac{logP(X \mid \theta_1,\ldots,\theta_n)}{\partial\theta_m} &= \left\langle\frac{\partial logP(X \mid \theta_1,\ldots,\theta_n)}{\partial\theta_m}\right\rangle \\ &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \frac{1}{\sum_y \prod_m P_m(y \mid \theta_m)} * \frac{\sum_y \prod_m P_m(y \mid \theta_m)\partial logP_m(y \mid \theta_m)}{\partial\theta_m} \\ &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \sum_y \frac{\prod_m P_m(y \mid \theta_m)}{\sum_y \prod_m P_m(y \mid \theta_m)} * \frac{\partial logP_m(y \mid \theta_m)}{\partial\theta_m} \\ &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \sum_y P(y \mid \theta_1,\ldots,\theta_n) * \frac{\partial logP_m(y \mid \theta_m)}{\partial\theta_m} \\ &= \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \left\langle\frac{\partial logP_m(y \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^\infty}\end{aligned}$$

Therefore, the gradient of the log-likelihood is proportional to the following equation:

$$\left\langle\frac{\partial logP(X \mid \theta_1,\ldots,\theta_n)}{\partial\theta_m}\right\rangle \propto \left\langle\frac{\partial logP_m(x \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^0} - \left\langle\frac{\partial logP_m(y \mid \theta_m)}{\partial\theta_m}\right\rangle_{Q^\infty}$$

Bibliography

Case Studies of Successful Robot Systems. MIT Press, Cambridge, MA, 1998. 2

M. Abdel-Rahman, T. N. Sainath, G. E. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny. Deep belief networks using discriminative features for phone recognition. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2011), pages 5060–5063, Prague Congress Center, Prague Czech Republic, May 22-27 2011. IEEE. 18, 43, 62

M. Abdel-Rahman, G. E. Dahl, and G. E. Hinton. Acoustic modeling using deep belief networks. Journal of IEEE Transactions on Audio, Speech and Language Processing, 20(1):14–22, 2012. 43

N. Aggarwal and R. K. Agrawal. First and second order statistics features for classification of magnetic resonance brain images. Journal of Signal and Information Processing, 3(2):146–153, 2012. xxx, 124, 132

H. Andreasson, A. Treptow, and T. Duckett. Localization for mobile robots using panoramic vision, local features and particle filter. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2005), pages 3348–3353, Barcelona, Spain, April 18-22 2005. IEEE. 16

D. Andrzejewski. Training binary restricted boltzmann machines with contrastive divergence. Technical report, Department of Computer Science, University of Wisconsih-Madison, Wisconsih, Madison, USA, 2009. xvi, 55

F. Attneave. Some informational aspects of visual perception. Journal of Psychological Review, 61(3):183–193, 1954. xii, 65

H. Barlow. Redundancy reduction revisited. Journal of Network: Computations in Neural Systems, 12:241–325, 2001. xii, 65

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In Proceedings of the 9^{th} European conference on Computer Vision (ECCV 2006), volume 3951, pages 404–417, Graz, Austria, May 7-13 2006. Springer. 15

A. J. Bell and T. J. Sejnowski. The 'independent components' of natural scenes are edge filters. Journal of Vision Research, 37(23):3327–3338, 1997. xiii, 42, 45, 65, 67

Y. Bengio. Learning deep architectures for ai. Journal of Foundations And Trends In Machine Learning, 2(1):1–127, 2009. 43, 57, 59, 131

Y. Bengio and Y. LeCun. Scaling learning algorithms towards ai. In L. Bottou, C. Olivier, D. DeCoste, and J. Weston, editors, Large Scale Kernel Machines. MIT Press, New York University, New York, USA, 2007. 52

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2007), volume 19, pages 153–160, Hyatt Regency Vancouver, Vancouver, B.C., Canada, December 3-8 2007. MIT Press. 59

K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 1998), pages 368–374, Denver, Colorado, USA, November 29- December 4 1998. 19

K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In Proceedings of the International Conferecne on Database Theory (ICDT 1999), volume 1540, pages 217–235, Jerusalem, Israel, January 10-12 1999. Springer. 19

C. M. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK, 1995. 29

P. Blaer and P. Allen. Topological mobile robot localization using fast vision techniques. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002), pages 1031–1036, Washington, DC, USA, May 11-15 2002. 16

A. Blum and T. M. Mitchel. Combining labeled and unlabeled data with cotraining. In Proceedings of the 11^{th} Annual Conference on Learning Theory, pages 92–100, University of Wisconsin, Madison, July 24-26 1998. 19

J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. Journal of Robotic Systems, 14(4):231–249, 1997. 2

Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010), pages 2559–2566, San Francisco, Canada, June 13 - 18 2010. IEEE. x

M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning, January 06-08 2005. 57

C. Chang, D.L. Page, and M. A. Abidi. Object-based place recognition and loop closing with jigsaw puzzle image segmentation algorithm. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008), pages 557–562, Pasadena, California, USA, May 19-23 2008. IEEE. 11

C. Chang, A. Koschan, and M. A. Abidi. Object-based place recognition and scene change detection for perimeter patrol. Journal of Transactions of the American Nuclear Society, 101:823–824, 2009. 11

O. Chapelle, P. Haffner, and V. Vapnik. Journal of IEEE Transactions on Neural Networks. 34

W. J.C.H. Christopher. Learning from delayed rewards. Phd thesis, King's College, Oxford, Cambridge, UK, 1989. 20

O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), pages 17–24, Fontainebleau Resort, Miami Beach, Florida, USA, June 20-26 2009. IEEE. 41

A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. Journal of Machine Learning Research (JMLR) - Proceedings Track, 15:215–223, 2011. xiii, xiv, 65, 67

F. Coelho and C. Ribeiro. Evaluation of global descriptors for multimedia retrieval in medical applications. In A. M. Tjoa and R. Wagner, editors, DEXA Workshops, pages 127–131, University of Deusto, Bilbao, Spain, August 30 - September 3 2010. IEEE Computer Society. 16

C. Cortes and V. Vapnik. Support-vector networks. Journal of Machine Learning, 20 (3):273–297, 1995. xxxix, 31

N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge, UK, first edition, March 2000. 34, 116

R. M. David, C. F. Charless, and M. Jitendra. Learning to detect natural image boundaries using local brightness, color, and texture cues. Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 26(5):530–549, 2004. xl, 64

K. Deb, S. Karthik, and T. Okabe. An introduction to genetic algorithms. In Sadhana, pages 293–315. John Wiley and Sons, Inc, 1999. 19

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1999), pages 1322–1328, Detroit, Michigan, USA, May 10-15 1999. 24

G. Desjardins, A. Courville, Y. Bengio, P. Vincent, and O. Delalleau. Parallel tempering for training of restricted boltzmann machines. In Proceedings of the

International Conference on Artificial Intelligence and Statistics (ICAIS 2010), pages 145–152, Sardinia, Italy, May 13-15 2010. 70

E. Doi, D. C. Balcan, and M. S. Lewicki. A theoretical analysis of robust coding over noisy overcomplete channels. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2006), volume 19, pages 307–314, Hyatt Regency Vancouver, Vancouver, B.C., Canada, December 4-7 2006. MIT Press. 61, 133

A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Department of Engineering, University of Cambridge, Cambridge, UK, 1998. 20, 24

M. Dubois, H. Guillaume, E. Frenoux, and P. Tarroux. Visual place recognition using bayesian filtering with markov chains. In Proceedings of the 19^{th} European Symposium on Artificial Neural Networks (ESANN 2011), pages 435–440, Bruges, Belgium, April 27-29 2011. ix, 10, 22

D. Erhan, P. A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. Journal of Machine Learning Research - Proceedings Track, 5:153–160, 2009. 59

R. Fergus. Visual object category recognition. Ph.d. thesis, University of Oxford, 2005. xxxviii, 11, 13, 14

D. J. Field. Relations between the statistics of natural images and the response properties of cortical cells. Journal of Optical Society of America, A, 4(12):2379–2394, 1987. xiii, 65

D. J. Field. What is the goal of sensory coding? Journal of Neural Computation, 6(4): 559–601, 1994. 42

D. Filliat. Interactive learning of visual topological navigation. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2008), pages 248–254, Nice, France, September 22-26 2008. ix, 16

A. Fischer and C. Igel. Empirical analysis of the divergence of gibbs sampling based learning algorithms for restricted boltzmann machines. In Proceedings of the 20^{th} international conference on Artificial neural networks (ICANN 2010), volume 6354, pages 208–217, Thessaloniki, Greece, September 15-18 2010. Springer. 70, 93

Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California, Santa Cruz Santa Cruz, CA, USA, 1994. 71

B. H. Gary, R. Manu, B. Tamara, and L. M. Erik. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. xliii, 95, 100

J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omni-directional camera. Journal of IEEE Transactions on Robotics and Automation, 16(6):890–898, 2000. 17

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. Journal of IEEE Transactions Pattern Analysis and Machine Intelligence, 6(6):721–741, 1984. 53

D. Gokalp and S. Aksoy. Scene classification using bag-of-regions representations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007), pages 1–8, Minneapolis, Minnesota, USA, June 18-23 2007. IEEE Computer Society. 16

H. Guillaume, M. Dubois, E. Frenoux, and P. Tarroux. Temporal bag-of-words - a generative model for visual place recognition using temporal integration. In Proceedings of the 6^{th} International Conference on Computer Vision Theory and Applications (VISAPP 2011), pages 286–295, Vilamoura, Algarve, Portugal, March 05-07 2011. SciTePress. xxvii, xxxi, 3, 10, 22, 114, 118, 124, 128, 134

J. Handschin. Monte carlo techniques for prediction and filtering of non-linear stochastic processes. Journal of Automatica, 6(4):555–563, 1970. 24

J. Handschin and D. Mayne. Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. International Journal on Control, 5 (5):547–559, 1969. 24

R. Hecht-Nielsen. Replicator neural netowrks for universal optimal source coding. Journal of Science, 269:1860–1863, 1995. 52

G. E. Hinton. Training products of experts by minimizing contrastive divergence. Journal of Neural Computation, 14(8):1771–1800, 2002. x, xvi, 42, 43, 48, 52, 54, 55, 56, 57, 62, 69, 131, 143

G. E. Hinton. Deep belief networks. Journal of Scholarpedia, 4(5):47–59, 2009. 59, 63

G. E. Hinton. A practical guide to training restricted boltzmann machines - version 1. Technical report, Department of Computer Science, University of Toronto, Toronto, Canada, 2010. xv, 51, 70, 74, 76, 78, 79, 80, 81, 82, 86, 93, 122

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Journal of Science, 313(5786):504–507, 2006. 43, 44, 63, 71, 82, 112

G. E. Hinton, T.J. Sejnowski, and D.H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. Technical report, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 1984. 47

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Journal of Neural Computation, 18(7):1527–1554, 2006. x, 18, 42, 43, 45, 59, 62, 63

G. E. Hinton, A. Krizhevsky, and S.D. Wang. Transforming auto-encoders. In Proceedings of the International Conference on Artificial Neural Networks (ICANN 2011), pages 44–51, Espoo, Finland, June 14-17 2011. xxxi, 63, 136

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States, 79(8):2554–2558, 1982. 49

A. S. Hsu and T. L. Griffiths. Effects of generative and discriminative learning on use of category variability. In Proceedings of the 32^{nd} Annual Conference of the Cognitive Science Society, Portland, Oregon, August 11-14 2010. 6

A. Hyvärinen and E. Oja. Independent component analysis: Algorithms and applications. Journal of Neural Networks, 13(4-5):411–430, 2000. xiii, 65

A. Hyvärinen, J. Karhunen, and E. Oja. Independent Component Analysis. Wiley, New York, USA, 2001. 45, 72

N. Jaitly and G. E. Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2011), pages 5884–5887, Prague Congress Center, Prague Czech Republic, May 22-27 2011. IEEE. 71

T. Joachims. Transductive inference for text classification using support vector machines. In Proceedings of the 1999 International Conference on Machine Learning (ICML 1999), pages 200–209, Bled, Slovenia, June 27-30 1999. Morgan Kaufmann. 19, 29

A. Krizhevsky. Learning multiple layers of features from tiny images. Master science thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 2009. xv, xxi, 48, 50, 70, 91, 137, 138

A. Krizhevsky. Convolutional deep belief networks on cifar-10. Technical report, Department of Computer Science, University of Toronto, Toronto, Canada, 2010. xxv, xxxviii, xliv, 108, 124

A. Krizhevsky and G. E. Hinton. Using very deep autoencoders for content-based image retrieval. In Proceedings of the 19^{th} European Symposium on Artificial Neural Networks. xxxi

S. Kullback and R. A. Leibler. On information and sufficiency. Journal of the Annals of Mathematical Statistics, 22(1):79–86, 1951. 54

K. Labusch and T. Martinetz. Learning sparse codes for image reconstruction. In Proceedings of the 18^{th} European Symposium on Artificial Neural networks, Computational Intelligence and Machine Learning. x

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. Journal of Machine Learning Research, 1:1–40, 2009. 52

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2006), pages 2169–2178, New York, NY, USA, June 17-22 2006. IEEE Computer Society. 16

Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. Neural Networks: Tricks of the trade. Springer, 1998.

Y. LeCun, S. Chopra, R. Hadsell, M. A. Ranzato, and F. J. Huang. A tutorial on energy-based learning. In Predicting Structured Data. MIT Press, 2006. 44, 45, 46

H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area v2. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2008), volume 20, pages 873–880, Vancouver, British Columbia, Canada, December 8-11 2008. MIT Press. xvii, xviii, xxii, xxv, xxvi, xxvii, xxxvii, xxxviii, xlii, 79, 82, 83, 85, 88, 109, 122

H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26^{th} International Conference on Machine Learning (ICML 2009), pages 609–616, Montreal, Canada, June 14-18 2009. Computer Science Department, Stanford University, Stanford, CA 94305, USA. xi, xxi, 70, 71, 74, 82, 91, 99, 122, 133

E. Leopold and J. Kindermann. Text categorization with support vector machines. how to represent texts in input space? Journal of Machine Learning, 46(1-3):423–444, 2002. 29

O. Linde and T. Lindeberg. Object recognition using composed receptive field histograms of higher dimensionality. In Proceedings of the 17^{th} International Conference on Pattern Recognition (ICPR 2004), volume 2, pages 1–6, Cambridge, England, UK, August 23-26 2004. 17

D. G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the International Conference on Computer Vision (ICCV 1999), volume 2, pages 1150–1157, 1999. 15

D. G. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(10):91–110, 2004. 15, 41

J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. The kth-idol2 database. Technical Report CVAP-Report 304, Computational Vision and Active Perception Laboratory, School of Computer Science and Communications, Royal Insitute of Technology, Available at: http://cogvis.nada.kth.se/IDOL/, 2006. 14

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2008), volume 20, pages 1033–1040, Vancouver, British Columbia, Canada, December 8-11 2008. Curran Associates, Inc. 82, 122

E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Image-based monte-carlo localisation with omnidirectional images. Journal of Robotics and Autonomous Systems, 48(1):17–30, 2004. 17

T. Mitchell. Machine Learning. McGraw Hill Series in Computer Science, Pittsburgh, Pennsylvania, USA, 1997. 19

H. Murase and S.K. Nayar. Visual learning and recognition of 3-d objects from appearance. International Journal of Computer Vision, 14(1):5–24, 1995. 11

V. Nair and G. E. Hinton. Implicit mixtures of restricted boltzmann machines. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Proceedings of the Advances in Neural Information Processing Systems (NIPS 2008), pages 1145–1152, Vancouver, British Columbia, Canada, December 8-11 2008. Curran Associates, Inc. 71

V. Nair and G. E. Hinton. 3-d object recognition with deep belief nets. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2009), volume 22, pages 1339–1347, Hyatt Regency Vancouver, Vancouver, B.C., Canada., December 6-11 2009. 43, 62, 82

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz and T. Joachims, editors, Proceedings of the 27^{th} International Conference on Machine Learning (ICML 2010), pages 807–814, Haifa, Israel, June 21-24 2010. Omnipress. xliv, 95, 102, 103, 105

A. Y. Ng. Cs229 lecture notes on machine learning. Technical report, Stanford University, Department of Computer Science, Stanford, USA, 2011. 39, 116, 140

A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2002), pages 841–848, Vancouver, British Columbia, Canada, December 9-14 2002. 41

M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), pages 2735–2742, Fontainebleau Resort, Miami Beach, Florida, USA, June 20-26 2009. xii, 70, 71, 73, 74, 79, 99, 133

R. M. Nosofsky, D. R. Little, and T. W. James. Activation in the neural network responsible for categorization and recognition reects parameter changes. Proceedings of the National Academy of Sciences, 109(20):91–110, 2011. 17

T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 24(7):971–987, 2002. 16

A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision, 42(3):145–175, 2001. 15, 16

A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. Journal of Progress in Brain Research, 155:23–36, 2006. 10, 16, 61

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Journal of Nature, 381(6583):607–609, 1996. xiii, xvii, 45, 65, 67, 82, 83, 122

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? Journal of Vision Research, 37(23):3311–3325, 1997. xxiv, 45, 61, 72, 82, 85, 92, 122, 133

B. A. Olshausen and D. J. Field. Sparse coding of sensory inputs. Journal of Current Opinion in Neurobiology, 14(4):481–487, 2004. x, 42, 43

C. Papageorgiou and T. Poggio. A trainable system for object detection. International Journal of Computer Vision, 38(1):15–33, 2000. 11

M. Pazzani and P. Domingos. On the optimality of the simple bayesian classifier under zero-one loss. Journal of Machine learning, 29(2-3):103–130, 1997. 19

J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pages 1–20, Karlsruhe and Munich, Germany, October 1-3 2003. 20

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007), pages 1–8, Minneapolis, Minnesota, USA, June 18-23 2007. IEEE Computer Society. 41

A. Pronobis. Indoor place recognition using support vector machines. Master's thesis, Stockholm, Sweden, November 2005. 17

A. Pronobis and B. Caputo. Confidence-base cue integration for visual place recognition. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS 2007), San Diego, California, USA, October 29 - November 2 2007. 4, 14, 20, 35, 61, 114, 132

A. Pronobis, O. Martínez Mozos, and B. Caputo. 29

A. Pronobis, B. Caputo, P. Jensfelt, and H. I. Christensen. A discriminative approach to robust visual place recognition. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), pages 3829–3836, Beijing, China, October 9-15 2006. ix, xxx, 17

A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt. Multi-modal semantic place classification. The International Journal of Robotics Research (IJRR), 29(2-3): 298–320, February 2010. 3

J. R. Quinlan. Induction of decision trees. Journal of Machine Learning, 1(1):81–106, 1986. 19

A. Ramisa, T. Adriana, A. David, T. Ricardo, and L. M. Ramon. Robust vision-based robot localization using combinations of local feature region detectors. Journal of Autonomous Robots archive, 27(4):373–385, 2009. 15

M. Ranzato, Y.-L. Boureau, and Y. Lecun. Sparse feature learning for deep belief networks. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2007), volume 19, Cambridge, MA, 2007a. MIT Press. 133

M. A. Ranzato, C. Poultney, S. Chopra, and Y. Lecun. Efficient learning of sparse representations with an energy-based model. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2006), volume 19, pages 1137–1144, Hyatt Regency Vancouver, Vancouver, B.C., Canada, December 4-7 2006. MIT Press. 61

M. A. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007), pages 1–8, New York University, New York, USA, June 17 - 22 2007b. IEEE. x

M. A. Ranzato, A. Krizhevsky, and G. E. Hinton. Factored 3-way restricted boltzmann machines for modeling natural images. Journal of Machine Learning Research (JMLR) - Proceedings Track, 9:621–628, 2010. xi, xiv, xl, 44, 65, 69, 72, 99

D. E. Rumelhart and J. L. McClelland. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge, MA, UK, 1986. 19, 26

D. E. Rumelhart, G. E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing: explorations in the microstructure of cognition, volume 2, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. 27

R. Salakhutdinov. Learning in markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, Proceedings of the Advances in Neural Information Processing Systems (NIPS 2009), volume 22, pages 1598–1606, Hyatt Regency Vancouver, Vancouver, B.C., Canada., December 6-11 2009. Curran Associates, Inc. 54

R. Salakhutdinov and G. E. Hinton. Semantic hashing. In Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models, Amsterdam, July 23-27 2007. Elsevier. 112

R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In Proceedings of the International Conference on Artificial Intelligence and Statistics (ICAIS 2009), volume 5, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida, USA, April 16-18 2009. 43, 44, 107

R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted boltzmann machines for collaborative filtering. In Proceedings of the 24^{th} international conference on Machine learning (ICML 2007), pages 791–798, Oregon State University in Corvallis, Oregon, USA, June 20-24 2007. ACM Press. 43, 62, 76

A. L. Samuel. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3):210–229, 1959. 18

R. Sarikaya, G. E. Hinton, and B. Ramabhadran. Deep belief nets for natural language call-routing. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2011), pages 5680–5683, Prague Congress Center, Prague Czech Republic, May 22-27 2011. IEEE. 18, 43

H. Schneiderman and T. Kanade. A statistical model for 3d object detection applied to faces and cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000), Hilton Head, SC, USA, June 13-15 2000. IEEE. 11

H. Schulz, A. Müller, and S. Behnke. Investigating convergence of restricted boltzmann machine learning. In Proceedings of NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, Whistler, Canada, 2010. 70, 93

S. Se, D. G. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001), volume 21, pages 2051–2058. IEEE, 2001. 16

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 29(3):411–426, 2007. 62

P. Smolensky. Information processing in dynamical systems: foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing Explorations in the Micorstructure of Cognition, volume 1. MIT Press, Cambridge, MA, USA, 1986. x, 42, 49

K. P. Soman, R. Loganathan, and V. Ajay. machine learning with SVM and other kernel methods. PHI Learning Private Limited, M-97, New Delhi-110015, India, second edition, 2009. xiii, 65

M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical report, University of Minnesota, 2000. 19

I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Proceedings of the Advances in Neural Information Processing Systems (NIPS 2008), pages 1601–1608, Vancouver, British Columbia, Canada, December 8-11 2008. MIT Press. 51

G. W. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman, editors, Proceedings of the 26^{th} International Conference on Machine Learning (ICML 2009), volume 382 of ACM International Conference Proceeding Series, page 129. ACM, June 14-18 2009. 51

G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2006), volume 19, pages 1345–1352, Hyatt Regency Vancouver, Vancouver, B.C., Canada, December 4-7 2006. MIT Press. 43, 62

Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. Energy-based models for sparse overcomplete representations. Journal of Machine Learning Research (JMLR) - Proceedings Track, 4(7-8):1235–1260, 2003. 45, 72

G. Tesauro. Practical issues in temporal difference learning. Journal of Machine Learning, 8:257–277, 1992. 52

S. Thrun. Is robotics going statistics? the field of probabilistic robotics. Journal of Communications of the ACM, March 2001. 7

S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. Journal of Artificial Intelligence, 128(1-2):99–141, 2000. 24

S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). MIT Press, Cambridge, MA, first edition, 2005. 2, 7

A. Torralba. Contextual priming for object detection. International Journal of Computer Vision (IJCV), 53(2):169–191, 2003. 14

A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), pages 273–280, Nice, France, October 14-17 2003a. ix, 6, 10, 15, 25

A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), volume 1, pages 273–280. IEEE Computer Society, October 14-17 2003b. 6, 10, 13, 14, 16, 20, 22, 24, 25, 61

A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008), pages 1–8, Anchorage, Alaska, USA, June 24-26 2008. ix, x, xii, xx, 41, 43, 59, 62, 63, 71, 99

M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71–86, 1991. 104, 105

M. M. Ullah, A. Pronobis, B. Caputo, J. Luo, and P. Jensfelt. The cold database. Technical report, CAS - Centre for Autonomous Systems. School of Computer Science and Communication. KTH Royal Institute of Technology, Stockholm, Sweden, 2007. xix, xx, xxxviii, 8, 9, 96, 97, 100

M. M. Ullah, A. Pronobis, B. Caputo, P. Jensfelt, and H. Christensen. Towards robust place recognition for robot localization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2008), pages 3829–3836, Pasadena, California, USA, May 19-23 2008. ix, xxviii, xxix, xxx, xxxi, xlv, 4, 6, 10, 14, 16, 17, 20, 22, 35, 61, 114, 115, 116, 117, 118, 124, 125, 126, 127, 128, 132, 134

I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2000), volume 2, pages 1023–1029, San Francisco, CA, USA, April 24-28 2000. 16

I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005), volume 2, pages 258–265, June 20-26 2005. 41

V. N. Vapnik. The nature of statistical learning theory. Springer-Verlag, New York, Inc., New York, USA, 1995. 28, 116

V. N. Vapnik. Statistical Learning Theory. Wiley-Interscience, Ney York, USA, 1998. 19, 26, 29

I. Vilares and K. Kording. Bayesian models: the structure of the world, uncertainty, behavior, and the brain. Journal of Annals of the New York Academy of Sciences, 1224, 2011. 51

P. Viola and M. Jones. Robust real-time object detection. International Journal of Computer Vision, 57(2):137–154, 2002. 11

C. Wallraven, B. Caputo, and A. B. A. Graf. Recognition with local features: the kernel recipe. In Proceedings of the IEEE International Conference on Computer Vision (ICCV 2003), volume 1, pages 257–264, Nice, France, October 14-17 2003. IEEE Computer Society. 34

D. Walther and C. Koch. Modeling attention to salient proto-objects. Journal of Neural Networks, 19(9):1395–1407, 2006. 112

M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2004), pages 1481–1488, Cambridge, MA, USA, 2004. MIT Press. 46

F. Wood and G. E. Hinton. Training products of experts by minimizing contrastive divergence. Technical report, Brown University, 2012. 53, 143

J. Wright, Y. Ma, J. Mairal, G. Spairo, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE, 98(6): 1031–1044, 2010. x

J. Wu and J. M. Rehg. Centrist: A visual descriptor for scene categorization. Journal of IEEE Transaction on Pattern Analysis and Machine Intelligence, 33(8):1489–1501, 2011. 4, 6, 16, 20, 24, 61

J. Wu, H. I. Christensen, and J. M. Rehg. Visual place categorization: Problem, dataset, and algorithm. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), pages 4763–4770, St. Louis, USA, 2009. IEEE. ix, 10, 15, 16, 22, 24, 25, 61

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009), pages 1794–1801, Fontainebleau Resort, Miami Beach, Florida, USA, June 20 25 2009. IEEE. x

K. Yu, R. Salakhutdinov, Y. LeCun, G. E. Hinton, and Y. Bengio. Icml 2009 workshop on learning feature hierarchies, 2009. URL `http://www.cs.toronto.edu/~rsalakhu/deeplearning/cfa.html`. 134

R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In Proceedings of the third European conference on Computer Vision (ECCV 1994), volume 801, pages 151–158, New York, Inc. Secaucus, NJ, USA, 1994. Springer-Verlag. 16