Information Based Universal Feature Extraction and Object Recognition

DISSERTATION
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von

Mohammad Amiri
aus Sary, Iran

Frankfurt, September 2016
(D 30)

vom Fachbereich Informatik und Mathematik der Johann Wolfgang Goethe - Universität als Dissertation angenommen.

Dekan: Prof. Dr. Uwe Brinkschulte

1. Gutachter: Prof. Dr. Rüdiger Brause

2. Gutachter: Prof. Dr. Matthias Kaschube

Datum der Disputation:

"There are no incurable diseases - only the lack of will. There are no worthless herbs - only the lack of knowledge"

Avicenna

"Logic will get you from A to B. Imagination will take you everywhere."

Albert Einstein

Abstract

One of the main things that we as humans do in our lifetime is the recognition and/or classification of all kind of visual objects. It is known that about fifty percentage of the neocortex is responsible for visual processing. This fact tells us that object recognition (OR) is a complex task in our and in the animal brain, but we do it in a fraction of a second .

The main question is: How does the brain exactly do it? Does the brain use some feature extraction algorithm for OR tasks? The hierarchical structure of the visual cortex and studies on a part of the visual cortex called V1 tell us that our brain uses feature extraction for OR tasks by Gabor filters. We also use our previous knowledge in object recognition to detect and recognize the objects which we never saw before. Also, as we grow up we learn new objects faster than before.

These facts imply that the visual cortex of human and other animals uses some common (universal) features at least in the first stages to distinguish between different objects. In this context, we might ask: Do universal features in images exist, such that by using them we are able to efficiently recognize any unknown object? Is it necessary to extract new special features for any new object? How about using existing features from other tasks for this? Is it possible to efficiently use extracted feature of a specific task for other tasks? Are there some general features in natural and non-natural images which can also be used for specific object recognition? For example, can we use extracted features of natural images also for handwritten digit classification?

In this context, our work proposes a new information-based approach and tries to give some answers to the questions above. As a result, in our case we found that we could indeed extract unique features which are valid in all three different kinds of tasks. They give classification results that are about as good as the results reported by the corresponding literature for the specialized systems, or even better ones.

Another problem of the OR task is the recognition of objects, independently of any perception changes. We as humans or also animals can recognize objects in spite of many deformations (e.g. changes in illumination, rotation in any direction or angles, distortion and scaling up or down) in a fraction of a second. When observing an object which we never saw, we can imagine the rotated or scaled up object

in our mind. Here, also the question arises: How does the brain solve this problem? To do this, does the brain learn some mapping algorithm (transformation), independent of the objects or their features ?

There are many approaches to model the mapping task. One of the most versatile ones is the idea of dynamically changing mappings, the dynamic link mapping (DLM). Although the dynamic link mapping systems show interesting results, the DLM system has the problem of a high computational complexity. In addition, because it uses the least mean squared error as risk function, the performance for classification is also not optimal. For random values where outliers are present, this system may not work well because outliers influence the mean squared error classification much more than probability-based systems. Therefore, we would like to complete the DLM system by a modified approach.

In our contribution, we will introduce a new system which employs the information criteria (i.e. probabilities) to overcome the outlier problem of the DLM systems and has a smaller computational complexity. The new information based self-organised system can solve the problem of invariant object recognition, especially in the task of rotation in depth, and does not have the disadvantage of current DLM systems.

Übersicht und Zusammenfassung

Menschen nutzen ihr bestehendes Wissen um Neues zu lernen: Sie nutzen Informationen über grundlegende mathematische Tatsachen um komplexere mathematische Probleme zu lösen, oder sie nutzen Wissen über das Fahren eines Motorrads um schneller Auto zu fahren. Das Übertragen von Wissen ist eine der Strategien unseres Gehirns, um Objekte und Konzepte schneller zu erlernen.

Eines der vielen Dinge, die wir als Menschen in unserem Leben tun ist das Erkennen und/oder Kategorisieren aller Arten von visuellen Objekten ("OE-Aufgabe"). In [1] wird erwähnt, dass rund fünfzig Prozent des Neocortex für das Verarbeiten visueller Reize genutzt wird. Aus dieser Tatsache können wir schließen, dass Objekterkennung eine komplizierte Aufgabe in unserem Gehirn und in den Gehirnen von Tieren ist, und trotzdem schaffen wir dies in Sekundenbruchteilen [2].

Die zentrale Frage dabei ist: Wie schafft das Gehirn das? Nutzt das Gehirn eine Art Merkmalsextraktionsalgorithmus für OE-Aufgaben? Der hierarchische Aufbau des visuellen Cortex und Studien eines Teils der Sehrinde V1 zeigen, dass unser Gehirn Merkmalsextraktion mit Gabor-Filtern ausführt [3, 4].

Wir nutzen außerdem bereits erworbenes Wissen über Objekterkennung um Objekte zu bemerken und zu erkennen, die wir nie zuvor gesehen haben. Außerdem lernen wir, neue Objekte schneller zu erkennen wenn wir älter werden. Die Frage ist wieder, wie machen wir das? Gibt es Merkmale, die verschiedene Objekte miteinander gemeinsam haben, die uns helfen verschiedenste Objekte mit unterschiedlichsten Wahrscheinlichkeiten und Eigenschaften zu erkennen?

Die Extraktion von Merkmalen ist ein wichtiger Schritt in der Erkennung von Mustern und zielt darauf ab, die relevanten Informationen, die eine Objektklassifizierung ermöglichen, zu erhalten. Es ist die Basis für jede Erkennung von Objekten, sowohl im menschlichen als auch beim maschinellen Sehen. Die Extraktion und Wiederverwendung von Information impliziert, dass die Sehrinde von Menschen und anderen Tieren gemeinsame (universelle) Merkmale zumindest in den tieferen Ebenen nutzt, um zwischen verschiedenen Objekten zu unterscheiden. Es ist immer noch ein schwieriges Problem im maschinellen Sehen, Merkmale zu extrahieren, die die fundamentale Substanz von Bildern so komplett wie möglich abbilden.

In diesem Zusammenhang stellt sich die Frage: Gibt es universelle Merkmale in Bildern, so dass unter Verwendung dieser Merkmale ein unbekanntes Objekt effizient erkannt werden kann? Ist es nötig neue, spezielle Merkmale für jedes neue Objekt zu finden? Was geschieht mit den bereits gelernten Merkmalen früherer Objekterkennungen? Ist es ohne großen Aufwand möglich, extrahierte Merkmale aus einer Aufgabe für eine andere Aufgabe zu nutzen? Gibt es einige allgemeine Merkmale in natürlichen und nicht-natürlichen Bildern, die auch für spezielle Objekterkennungsaufgaben erfolgreich verwendet werden können? Können wir beispielsweise Merkmale natürlicher Bilder für das Erkennen handgeschriebener Ziffern verwenden?

In den letzten Jahrzehnten wurden diese Fragen kaum erforscht. Manchmal wird das Konzept von *transfer learning* verwendet, um Wissen aus einem Klassifizierungsproblem für ein anderes Problem wiederzuverwenden. In diesem Kontext schlagen wir einen neuen, informationsbasierten Ansatz vor und versuchen, Antworten auf die oben gestellten Fragen zu finden.

Ein weiteres Problem der Objekterkennung ist das Erkennen von Objekten, unabhängig von jeglichen Änderungen, die durch den Kontext verursacht werden. Wir als Menschen und auch viele Tiere sind in der Lage, Objekte trotz vieler Deformationen (z.B. Änderung der Lichtverhältnisse, Drehung um beliebige Achsen und Winkel, Verzerrung sowie Vergrößerung und Verkleinerung) in Sekundenbruchteilen zu erkennen. Beim Beobachten eines Objektes, das wir nie zuvor gesehen haben, können wir uns trotzdem eine gedrehte oder vergrößerte Version des Objektes vorstellen. Damit stellt sich auch hier die Frage, wie wir das tun. Lernt das Gehirn eine Abbildungstransformation, unabhängig von Objekten oder deren Merkmalen?

Zu dieser Frage wurde in den vergangenen Jahrzehnten viel geforscht, aber es gibt immer noch viele ungelöste Probleme aufgrund der beschriebenen Schwierigkeiten bei Objekterkennungsaufgaben. Zum Beispiel gibt es noch kein künstliches Objekterkennungssystem, das eine Objekterkennung auf menschlichem Level ausführen kann, unabhängig von bestehenden Objektdeformationen.
Eines der flexibelsten Systeme, Abbildungen von visuellen Objekten zu gespeicherten, bekannten Objekten zu finden, ist die Gruppe der dynamischen Abbildungen (Dynamic Link Mapping DLM). Obwohl diese Gruppe vobn Systemen interessante Resultate aufweist, haben sie auch Probleme: zum einen sind sie durch die Verwendung des mittleren quadratischen Fehlers empfindlich gegenüber

Ausreißern, und zum anderen haben sie eine hohe Rechenkomplexität.
Aus diesem Grund beschäftigen wir uns in dieser Arbeit näher mit diesen Systemen und entwickeln einen neuen Algorithmus , der auf der Shannon-Information basiert. Wir können zeigen, dass der neue informationsbasierte, selbst-organisierende Algorithmus das Problem der invarianten Objekterkennung lösen kann, insbesondere auch das Problem der 3D-Rotation in der Tiefe.

Kontext der Arbeit
In diesem Zusammenhang ist es sinnvoll, einige allgemein gebräuchliche Begriffe genauer vorzustellen.

Merkmalsextraktion

Bei jeder Klassifizierung von Mustern oder Objekterkennungsaufgabe ist die Extraktion und Verwendung von Merkmalen, die geeignete Informationen zur Charakterisierung der Objekte bereitstellen, essentieller Teil der Diagnose. Es gibt viele Algorithmen um eine Merkmalsextraktion für Muster- und Objekterkennungsaufgaben durchzuführen. Sie lassen sich grob in drei Hauptgruppen unterteilen: Merkmalsextraktion basierend auf statistischer Analyse, Merkmalsextraktion basierend auf Interessensfokus und Merkmalsextraktion mit künstlichen Neuronalen Netzen.

Merkmalsextraktion mithilfe statistischer Analyse

Als Merkmalsextraktion mithilfe statistischer Analyse bezeichnet man eine Merkmalsextraktion, die statistische Werkzeuge wie Varianz, Kovarianz oder Korrelation nutzt um die wertvollste Information aus den Eingabedaten zu extrahieren. Einige dieser Extraktionsalgorithmen sind Hauptkomponentenanalyse (PCA), Lineare Diskriminanzanalyse (LDA), Unabhängigkeitsanalyse (ICA) und Faktoranalyse (FA), die alle Bildpunkte umfassen.

Interessensfokus (POI) und Orientierungspunkt

Im Gegensatz dazu basieren POI Methoden auf interessanten Punkten oder Regionen eines Bildes, die nützlich für Objekterkennungsaufgaben sein können. Diese Punkte sollten gegen Beleuchtungsänderungen, Drehungen und Verschiebungen im Vergleich zu den Originalbildern sich nicht ändern. Die Methoden dafür beinhalten beispielsweise Kantenerkennung [5–7], Eckpunkterkennung [8, 9], Kleckserkennung [10], skaleninvariante Merkmalstransformation (SIFT) [11], beschleunigte, robuste Merkmale (SURF) [12] und Gabor Jets.

Merkmalsextraktion mit künstlichen neuronalen Netzen

Der traditionelle Ansatz zur Objekterkennung besteht darin, den Erkennungsprozess in zwei Teile zu teilen: (1) Merkmalsextration und (2) Klassifikation. Eines der Hauptprobleme bei diesem Ansatz ist, dass die Genauigkeit der Klassifikation davon abhängt, dass vorher die Art der Merkmale per Hand gut entworfen worden sind. Durch ein mehrschichtiges neuronales Netzwerk kann das Problem handgefertigter Merkmale umgangen und nichtlineare Objekterkennungsaufgaben gelöst werden. Merkmale werden nicht entworfen, sondern beim Lösen der Aufgabe selbst trainiert und sind damit nicht willkürlich gewählt. Die zur Optimierung der Netzwerke eingesetzten Verfahren basierend meist auf einer Zielfunktion und dem Gradientenverfahren. Sie werden genutzt, um komplexe nichtlineare Probleme zu lösen. Die am weiten verbreitete Familie von neuronalen Netzen zur Mustererkennung [13] sind *feed-forward*-Netzwerke, zu denen auch mehrlagige Perzeptrons und radiale Basisfunktionsnetzwerke gehören. Eine weitere beliebte Netzwerkart ist die selbstorganisierende Karte (Kohonen-Netzwerk) [14], die hauptsächlich für das Clustern von Daten oder das Abbilden von Merkmalen verwendet wird. In der Literatur der letzten Jahrzehnte bis heute [15–18] taucht eine Vielzahl von Merkmalsextraktionsverfahren, basierend auf neuronalen Netzen, auf. In den letzten Jahren wurden dabei auch sogenannte *deep learning*-Netzwerke aus mehrschichtigen, komplexen Architekturen benutzt, um eine Vielzahl von Problemen in der Mustererkennung und im maschinellen Lernen zu lösen; viele Wissenschaftler arbeiten auf diesem Feld. Außerdem gibt es sog. *tiefe neuronale Netze*. Dies sind vielschichtige neuronale Netze (mindestens 3, aber auch 20 Schichten) mit Fehlerrückführung zur Bild- (Objekt-) oder Mustererkennung. In [19] ist eine Einschätzung von tiefen neuronalen Netzen zu finden.

Partizipation von Wissen

In der Praxis sind die meisten Modelle zum maschinellen Lernen entwickelt worden, um eine einzelne Aufgabe zu erledigen. In diesem Abschnitt fokussieren wir uns mehr auf die Details verschiedener Arten von Wissen bei der Nutzung dieser Methode für die Aufgaben der Klassifizierung. Dazu gehören das Übertragen von Wissen (Transfer learning), eigenständig erworbenes Wissen (self-taught learning), und universelle Merkmalsextraktion. Deswegen bezeichnen wir diese drei Typen des Lernens als Wissenspartizipation (knowledge sharing).

Invariante Objekterkennung

Invariante Objekterkennung bezeichnet die Unterscheidung eines Objektes unabhängig von Variationen im Bild des Objekts, wie zum Beispiel Änderungen des Blickwinkels, Lichts oder Hintergrunds. Wie bereits in **Kapitel 1** erwähnt, haben die schwierigsten Aufgaben der Objekterkennung mit diesen Änderungen von Blickwinkeln, Licht, Translationen und anderen affinen oder nicht-affinen Transformationen zu tun. Für Menschen und manche Tiere sind dies einfache Probleme. Zum Beispiel wird in [20] gezeigt, dass Ratten invariante visuelle Objekterkennung auszuführen. Es wurde außerdem festgestellt, dass Ratten spontan verschiedene Ansichten desselben Objekts als ähnlich wahrnehmen, also als verschiedene, veränderte Instanzen desselben Objekts wiedererkennen. Für die invariante Objekterkennung wurde eine Klasse von Algorithmen erfunden, um dieses Problem zu lösen. Abhängig von der Art des zu lösenden Problems und dem Feingefühl der Autoren wurde diese Problem auch als "Übereinstimmungsregistrierung" oder "Abgleich" bezeichnet. Diese Konzepte sind sehr ähnlich und überschneiden sich.

Dynamische Verbindungsabgleichsmethoden (DLM)

Die bisher erwähnten Graph-basierten Abgleichsmethoden zur Abbildung von visuellen Objekten auf gespeicherte Objekte sind im Wesentlichen statische Methoden, wogegen in der Natur sich der Abgleichsprozess dynamisch verhält. Dies spiegelt sich in Ansätzen mit dynamischen Methoden (DLM) wider, die Verbindungen zwischen den Bildpunkten des gesehenen Objekts und den Punkten des gespeicherten Objekts dynamisch herzustellen. Die DLM-Verfahren basieren auf der biologischen Beobachtung schneller, neuronaler Plastizität. Eine der neuesten Methoden dazu ist ein System, das in [21] vorgeschlagen wurde. In dieser Arbeit versuchen die Autoren, ein neues Modell zu definieren, das auf der Dynamik von sog. Häussler-Systemen basiert, das in [22] vorgeschlagen wurde. Dort wird die Formierung multipler Abbildungen mithilfe sogenannter "Kontrolleinheiten" gelernt. In diesem System ist jede Kontrolleinheit verantwortlich für eine spezielle Abbildung wie Verschiebung oder Deformation. Trotzdem hat auch dieses System einige Nachteile, die wir versuchen, zu vermeiden.

Dazu versuchen wir in unserem Beitrag zuerst, Merkmale zu finden, die für spezielle Klassifikationen optimal sind, und danach Merkmale zu finden, die auch für verschiedene Klassifikationsaufgaben verwendet werden können. Weiterhin werden wir ein System vorstellen, welches das Shannon-Informationskriterium (also Wahrscheinlichkeiten) nutzt um das Ausreißer Problem von DLM Systemen zu bewältigen und

außerdem eine niedrigere Rechenkomplexität aufweist.

Gliederung der Arbeit

In dieser Arbeit stellen wir einen umfangreichen Ansatz vor, bestehend aus universeller Merkmalsextraktion und einem informationsbasierten, selbstorganisierenden System, um die invariante Objekterkennung auszuführen.

In den vorherigen Absätzen haben wir gezeigt, dass es verschiedene Methoden für die Merkmalsauswahl und die Durchführung der Objekterkennung von deformierten invarianten Bildern gibt. Wie wir erwähnt haben, haben die POI-basierten Methoden das Problem, dass sie anwendungsspezifisch sind und manuell angepasste Algorithmen nutzen, die je nach Anwendungsfall unterschiedlich ausfallen und nicht allgemein verwendbar sind. Statistische Methoden wie PCA und ICA nutzen die Annahme, dass die Originaldaten in einem linearen Verfahren separiert werden können, aber diese Annahme ist für die meisten Daten in der realen Welt nicht zutreffend. Ein Ansatz, basierend auf künstlichen neuronalen Netzen wie beispielsweise tiefe Netzwerke, versucht dieses Nichtlinearitätsproblem von PCA und ICA zu lösen, aber diese Methoden können mit Transformationen, Translationen oder Änderungen in der Objektposition nur schwer umgehen. Obwohl die DLM-Systeme interessante Ergebnisse liefern, werden wir später auch im Detail zeigen, das DLM-Systeme ein Problem haben: das Problem einer hohen Rechenkomplexität. Da sie außerdem auf dem mittleren quadratischen Fehler als Risikofunktion basieren, ist die Klassifikationsleistung nicht optimal. Bei zufälligen Werte ergeben sich immer Ausreißer, so dass das System unter Umständen nicht die gewünschten Ergebnisse liefern kann: Ausreißer beeinflussen das Ergebnis des quadratischen Fehlers viel stärker als dies bei wahrscheinlichkeitsbasierten Systemen der Fall ist. Dies ist die Hauptmotivation unseres Ansatzes, einen neuen Ansatz für DLM zu finden. Dazu versuchen wir in unserem Beitrag zuerst, Merkmale zu finden, die für verschiedene Klassifikationen verwendet werden können, und danach Merkmale zu finden, die für verschiedene Klassifikationsaufgaben verwendet werden können. Weiterhin werden wir ein System vorstellen, welches das Shannon-Informationskriterium (also Wahrscheinlichkeiten) nutzt um das Ausreißer Problem von DLM Systemen zu bewältigen und außerdem eine niedrigere Rechenkomplexität aufweist.

In **Kapitel 3** haben wir eine neue Methode für die universelle Merkmalsextraktion vorgeschlagen, die allgemeiner ist, als die Aufgaben, die davor ausgeführt wurden, also Transferlernen, halbüberwachtes Lernen und unüberwachtes Lernen. Um dies zu erreichen nutzten wir einen auf der Informationstheorie basierenden Ansatz, um eine Risikofunktion zu wählen, die die Kreuzentropie minimiert. Als Grundlage für die Extraktion universeller Merkmale haben wir ein feed-forward neuronales Netz entwickelt. Außerdem wird die Anzahl der Parameter, die gelernt werden müssen, durch die Übertragung der Parameter unter allen rezeptiven Feldern reduziert. Zusätzlich zur reduzierten Anzahl von Parametern hat dies den Vorteil, dass alle Neuronen dieselben Merkmale lernen, unabhängig von ihrer Position im Eingabebild. Als Nachteil unseres Systems sollte angeführt werden, das diese Entscheidung nicht von den ursprünglichen Beweisen [23] für die Approximationseigenschaften von zweischichtigen neuronalen Netzen abgedeckt wird, so dass die klassischen Approximationseigenschaften nicht garantiert werden. Außerdem ist die Betrachtung der Filtereigenschaften der Ersten Schicht als *Merkmale* plausibel, aber willkürlich gewählt.

Trotzdem zeigen unsere Ergebnisse, dass diese universell trainierten Merkmale erfolgreich in verschiedenen Bildverarbeitungsaufgaben angewendet werden können, wie zum Beispiel bei der Erkennung handgeschriebener Ziffern sowie der Erkennung natürlicher und künstlicher Objekte, die vor künstlichen oder natürlichen Hintergründen platziert wurden oder der Erkennung verschiedener Texturen.

In **Kapitel 4** haben wir eine zweite und dritte verdeckte Schicht zu dem flachen neuronalen Netz aus dem vorangegangenen Kapitel hinzugefügt und die Leistung dieser Netze für verschiedene Parameter getestet. Interessanterweise zeigt das flache Netzwerk bessere Resultate als das Vielschichtennetzwerk, was auch von anderen Autoren bestätigt wird.

In **Kapitel 5** haben wir eine neue Methode zur Selbstorganisation des Systems vorgestellt. Wir nutzten einen Informationstheoretischen Ansatz um ein neues Ähnlichkeitsmaßzu definieren. Um die Ähnlichkeit zwischen Ein- und Ausgabe der neuronalen Aktivität zu messen, verwendeten wir eine Zielfunktion, die auf lokalisierter Shannon-Entropie basiert und von einer Gaußfunktion gewichtet wird. Als Ergebnis entwickelte sich eine gewünschte neuronale Abbildung zwischen visuellem Objekt und gespeichertem Objekt, sowohl ein- als auch zweidimensional. Um die Qualität des Endergebnisses und die Rechenkomplexität der vorgeschlagenen Methode mit zwei anderen erwähnten Methoden zu vergleichen, können

wir sagen, dass die finale Version unseres Vorschlags besser ist, als ein Kohonen-System. Vergleicht man die Methode, mit dem Häussler System lässt sich sagen, dass die Ergebnisse bei den untersuchten Beispielen zwar dieselben sind, aber unser Vorgehen hat immer eine niedrigere Rechenkomplexität.

In **Kapitel 6** haben wir zunächst die Theorie zu überwachten Systemen von Tomas et. al. für die Aufgabe der deformationsinvarianten Objekterkennung mit unserer Informationsabbildung verglichen. Wir nutzten die verbesserte Architektur des Häussler-Systems, die von Tomas und Malsburg in [21] vorgeschlagen wurde. Sie wendeten diese für die Verschiebung und Drehung im zweidimensionalen Raum an. Die guten Ergebnisse waren eine Motivation, den Fall von dreidimensionalen Transformationen zu testen. Das Ergebnis qualifiziert unser System mindestens so gut wie das Häussler-System, aber es nutzt weniger Rechenleistung und ist weniger beeinflusst von Ausreißern aufgrund des wahrscheinlichkeitsbasierten Ansatzes. Danach haben wir für den Fall des unüberwachten Lernens die Fähigkeiten ihrer Systeme für die drehungs- und tiefeninvariante Objekterkennung untersucht und die Konvergenz gezeigt. Auf jeden Fall ist auch für den unüberwachten Fall ein informationsbasierter Ansatz vorzuziehen. Dafür werden wir unser System in zukünftigen Arbeiten in diese Richtung ausbauen.

Acknowledgements

There are many people, organizations, and entities that I would like to thank for their assistance, services, and support. Without them many accomplishments in this thesis could be much more difficult to achieve, if not impossible.

My first and special thanks and appreciation to my advisor Professor Rüdiger Brause, who gave me energy, hint, and support through this research. He helped me to do my thesis and gave me the chance to manage some courses in the university and got experience for teaching.

Secondly, I want to thank Professor Christoph Von der Malsburg for his constructive advice and comprehensive comments during this research.

Thanks to Professor Jochen Triesch for letting me work in his research group for a month at the beginning of my study.

I would like to thank my friends Hota, Subbu, Tomas, Alex Schmid, Konda, and Dr. Morteza Monemizadeh.

Thanks to all staff at department of computer science who gave me support during this research.

Finally and most importantly, I wish to express my deep gratitude and appreciation to my dear wife and two daughters for their patience, understanding, and support during all these years. An honorable mention also goes to my family, especially my mother and father, for being totally encouraging and supportive, though they have been far away.

Contents

List of Figures

List of Tables

Abbreviations

SONP	**S**elf-**O**rganized **N**ural **P**rojection
SURF	**S**peeded-**U**p **R**obust **F**eatures
SIFT	**S**cale **I**nvariant **F**eatures **T**ransform
PCA	**P**rinciple **C**omponent **A**nalyses
ICA	**I**ndependent **C**omponent **A**nalyses
LDA	**L**inear **D**iscriminant **A**nalyses
KLD	**K**ullback **L**eibler **D**istance
SOM	**S**elf **O**rganizing **M**ap
UFE	**U**niversal **F**eature **E**xtraction
POI	**P**oint **O**f **I**nterest
ANN	**A**rtificial **N**eural **N**etwork
MLP	**M**ulti **L**ayer **P**erseptron
WTA	**W**inner **T**ake **A**ll
DLM	**D**ynamic **L**ink **M**atching
RF	**R**eceptive **F**ield
TL	**T**ransfer **L**earning
FA	**F**actor **A**nalysis
OR	**O**bject **R**ecognition
V1	Cortical Area **1**

To 175 divers of Karbala-4

Chapter 1

Introduction

1.1 Preamble

Humans use their existing knowledge to learn new things. They use the information taken from learning basic math operations to solve more complex mathematic problems or they use knowledge taken from driving a motorcycle to learn car driving faster. Therefore, knowledge sharing is one of our brain strategies to learn objects and also concepts faster.

One of the main things that we as human do in our lifetime is the recognition and/or classification of all kind of visual objects ("OR task"). It is noted in [1] that around fifty percentage of the nonhuman neocortex is responsible for visual processing. This fact tells us that object recognition is a complex task in our and in animal brain, but we do it in a fraction of a second [2]. The main question is: How does the brain exactly do it? Does the brain use some feature extraction algorithm for OR tasks? The hierarchical structure of the visual cortex and studies on a part of visual cortex called V1 tell us that our brain use feature extraction for OR tasks using Gabor filters [3, 4].

We also use our previous knowledge in object recognition to detect and recognize the objects which we never saw before. Also, as we grow up we learn new objects faster than before. The question is again: how do we do that? Are there some

common features among different objects that can be used for classification of many other objects with different probabilities and properties?

Feature extraction is an important step in the construction of any pattern classification and aims at the extraction of the relevant information that characterizes each class. This is basic for any task of object recognition, both in human vision and computer vision. These facts imply that the visual cortex of human and other animals use some common (universal) features at least in lower layers to distinguish between different objects. Additionally, it is still a challenging problem in computer vision how to extract universal features that reflect the fundamental substance of images as complete as possible. In this context, we might ask: Do universal features in images exist, such that by using them we are able to efficiently recognize any unknown object? Is it necessary to extract new special features for any new object? How about using existing features from other tasks? Is it possible to efficiently use extracted feature of a specific task for other tasks? Are there some general features in natural and non-natural images which can also be used for specific object recognition? For example, can we use extracted features of natural images also for handwritten digit classification?

Very little research attention has been paid to these problems in the last decades. Some people used the concept of *transfer learning* to reuse the knowledge taken from one classification problem for similar problems. In this context, our work proposes a new information-based approach and tries to give some answers to the questions above.

Another problem in the OR task is the recognition of objects, independently of any changes. We as human or also animals can recognize objects in spite of many deformations (e.g. changes in illumination, rotation in any direction or angles, distortion and scaling up or down) in a fraction of a second. When observing an object which we never saw, we can imagine the rotated or scaled up object in our mind. Here also the question arises: How does the brain solve this problem? To do this, does the brain learn some mapping algorithm (transformation), independent of the objects or their features ? Also about this question, much work has been

done during the past decades, but there are still many problems existing due to the difficulty of OR tasks which need to be solved in future. For instance, there is still no artificial OR system which could perform deformation independent object recognition as good as humans. In this contribution, also this question is addressed and a new algorithm is devised, based on information.

1.2 Thesis outline

In **chapter 2**, we review some feature extraction algorithms (e.g. SIFT and SURF) and knowledge sharing methods including *transfer learning*, *semi-supervised learning* and *self-taught learning*.

In **chapter 3**, we focus on the extraction of very general features that can be useful for object recognition and classification, implemented by a neural network of only two layers. We also do some experimental tests on different data sets with different classes and distributions in order to show the feasibility of our approach.

In **chapter 4**, a second and third hidden layer will be added to the shallow neural network from chapter 2. The chapter starts with a short introduction into deep architectures and then will look on the equations and updated learning rules for a three and four-layer neural network as well as on the network specifications. Lastly, the performance of the three and four layer networks for different parameters will be tested.

In **chapter 5**, additional to state-of-the-art systems, we introduce a new dynamical self-organized mapping which can be used for dynamical object recognition and classification. Firstly, we explain two well-known dynamical object recognition systems (Kohonen and Häussler systems). Then, we describe our new self-organized neural projection system which is based on information theory.

In **chapter 6**, we apply the new self-organized neural projection method developed in chapter 5 to the task of deformation invariant object recognition task and compare it to a Häussler system.

Finally in **chapter 7**, we describe some important conclusions about our result and discuss future work in this area.

Chapter 2

Feature Extraction and Object Recognition Today

In this chapter we give an overview of the state of the art on feature extraction, knowledge sharing and invariant object recognition algorithms, setting the ground for our newly developed algorithms.

2.1 Feature extraction

In any pattern classification or object recognition (OR) task, the extraction and usage of features which have excellent information to characterizes each class or object are the most crucial part of the diagnosis. There are many algorithms to do feature extraction for pattern and object recognition tasks. We can categorize them into several main groups, e.g., statistical analysis based feature extraction, point of interest (POI) based feature extraction and ANN based feature extraction.

2.1.1 Statistical analyses based feature extraction

Statistical analysis based feature extraction refer to the types of feature extractions which employ statistical tools e.g. variance, covariance or correlation to extract

most valuable information from input data. Some of this classical algorithms are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analyses (ICA) and Factor Analysis (FA).

PCA is type of linear dimension reduction (feature extraction) techniques which aim to transfer data from high dimension to lower dimension such that the variance of transformed data should be maximized. This task is done by using the covariance matrix of the data and the eigenvectors of the covariance matrix. The eigenvectors of the covariance matrix with the biggest eigenvalues are the new base for the linear transform. This reduced space is called *principle components* and is an orthogonal base system.
PCA is concerned with explaining the variance-covariance structure of a set of variables by a few linear combinations of these variables. Its general objectives are (1) data reduction and (2) interpretation [27].

In image classification work, it is common to use PCA as preprocessing step of analysis. We can use it to reduce the number of variables and avoid redundancy. PCA projects the high-dimensional data to a low-dimensional subspace by minimizing the expected reconstruction error. Among linear dimension methods, PCA is one of the most efficient of them but it's effective is based on the assumption that biggest variances have most importance, but this may not be true in the real world. Another assumption is that the data embedded in the low-dimensional space are globally linear or approximate linear [28].

LDA is a traditional statistical technique which is proposed by Fisher in [29]. It reduces dimensionality while the classes are separated as much as possible. To measure the distance among classes, he applied the concept of a between-class and within-class covariance matrix. The conventional form of the LDA assumes that all data are available in advance. The LDA computes its feature space by finding the spectral decomposition of the appropriate matrix. The conventional form of the LDA assumes that all the data are available in advance. The LDA computes its feature space by finding the spectral decomposition of the appropriate matrix. It is a traditional supervised technique for both dimensionality reduction

and classification [30]. In [31], they described that when the training data set is small, PCA can outperform LDA and, also, that PCA is less sensitive to different training data sets.

Independent component analysis (ICA) is a method in which a linear representation of non-Gaussian data has to be found such that the components are statistically independent, or as independent as possible [32]. In comparison to PCA, we can say that PCA is based on the information given by the second order statistics, while ICA goes up to forth order statistics.

Factor analysis (FA) is another types of linear feature extraction which is based on correlated variables and aims to transfer this correlated variables to lower number of unobserved variables called factors which are sufficient for the task of classification [33]. The main applications of FA are to reduce the number of variables and to detect structure in the relationships between variables for classifying variables. Therefore, factor analysis is applied as a data reduction or structure detection method.

2.1.2 Point of interest (POI) and landmarks

POI methods are based on some interesting points or regions of an image which can be useful for object recognition tasks. These points should be resistant due the change in illumination, rotation or shift in the original images. These methods include (but are not limited to) edge detection [5–7], corner detection [8, 9], blob detection [10], scale-invariant feature transform (SIFT) [11], speeded up robust features (SURF) [12] and Gabor jets .

2.1.3 Artificial neural network based feature extraction

The methods we talked in the first section (e.g. PCA or ICA) are linear, but the problems we meet in the real world are nonlinear. Therefore, it is necessary to search for nonlinear feature extraction methods for pattern recognition (PR). In

addition, the disadvantage of methods which are based on landmarks (POI) is that they are very task specific and need some hand-crafted algorithms which change from application to application.

The traditional way for object recognition is to divide the recognition process into two parts: (1) feature extractor and (2) classification. One of the main problems with this approach is that the accuracy rate depends on the ability of the designer to prepare a set of good features. By using a multilayer neural network for pattern recognition, we can overcome the problem of preparing hand-crafted features, and also solve the nonlinear PR problems. Features are learned by training during the task itself and are not subject to arbitrary design.

To learn very complex nonlinear problems, objective functions and gradient-based minimization techniques using the objective functions are used. The most commonly used family of neural networks for pattern classification tasks [13] are the feed-forward networks, which include multilayer perceptrons and radial basis function networks. Another popular network is the Self-Organizing Map or Kohonen-Network [14], which is mainly used for data clustering and feature mapping.

A large variety of feature extraction methods based on neural networks appeared in the literature from last decades up to now [15–18]. In recent years, so-called "deep" artificial neural networks have been used to overcome numerous problems in pattern recognition and machine learning, and many researchers are working in this field. Deep neural networks use artificial neural networks with many layers (from at least three to even 20 layers) and back-propagation technique for image (object) classification or pattern recognition. In [19], a review of deep learning in neural networks is available.

2.2 Knowledge sharing

In practice, most of the time machine learning models are designed to accomplish a single task. In this section, we try to focus on more details of different types of

knowledge using methods for the task of classification, including *transfer learning*, *self-taught learning* and *universal feature extraction*. Therefore, we reference these three types of learning as *knowledge sharing*.

The study of knowledge sharing is motivated by the human brain and the learning process within the human brain. Human learners appear to have natural ways to transfer knowledge between tasks. For instance, suppose that two persons want to learn driving a car. One of these persons has the motorcycle driving license and the other one does not have it. Learning the new task (here, driving a car) is easier for the person who has the driver license of the motorcycle than for the other one, because he or she can reuse some of the already acquired abilities. Depending on the methods which are used for knowledge sharing - including what to transfer (share), how to transfer and when to transfer - we can find different types of knowledge sharing algorithms like transfer learning, multi-task learning, self-taught learning, semi-supervised learning and universal (common) features learning. Let us have a closer look on these methods.

2.2.1 Multi-task learning and transfer learning

Multi-task learning is an approach that learns a problem together with other related problems at the same time. By this way, the major task can be learned better through using the experience gained by other tasks. This approach is effective when the tasks have some similarities. For instance, in [34],they used an algorithm which alternately performs a supervised and an unsupervised step, where, in the supervised step it learns task-specific functions and in the unsupervised step it learns common across tasks representations.

In [35] *transfer learning* is defined as "the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks". By this definition, transfer learning aims to extract the information from one or more source functions and applies the knowledge to a different function. Consider that

the target task should be related to the task of origin. You can find some work on transfer learning in [36] and [37].

In comparison to multi-task learning which tries to learn all of the source and target tasks simultaneously, transfer learning goal is to improve the learning of target task.

2.2.2 Semi-supervised learning

Semi-supervised learning is another type of knowledge sharing algorithm which is used in the literature [38], [39] and [40]. This kind of transfer learning is useful when we want to classify some data, and there are not sufficient labeled data of the domain, or it is difficult to get, but unlabeled data of the same domain is cheap and available. In this case, we try to transfer information from unlabeled data to classify labeled data. In comparison with transfer learning, we can say that in transfer learning we use typically labeled data from a different but related task, and try to transfer knowledge from one supervised learning task to another supervised task.

2.2.3 Self-taught learning

One more general algorithm than semi-supervised learning is *self-taught learning* which was introduced by Rajat Raina et al. in [41]. In this work, they tried to use information (features) extracted from some unlabeled natural images to classify labeled image data (elephants and rhinos). In another test, they used features extracted from a font character recognition task to classify handwritten English characters. Here also, the source and target task should be from the same domain but a different distribution.

2.3 Invariant object recognition

Invariant object recognition refers to distinguish an object regardless of image variations, such as variations in viewpoint, lighting or background. As we mentioned in Chapter 1, one of the hardest tasks in OR problems is related to changes of an object in view angle, illumination, translation and other types of affine or non-affine transformation. This task is done by animals and human easily. For instance, in [20] it is shown that rats are capable of invariant visual object recognition. They also got the result that rats spontaneously perceive different views of a visual object as similar to each other, i.e. as instances of the same object.

Some groups of algorithms are invented to overcome this problem. Depending on the type of problem and also on the tact of authors this problem has also been referred to as *alignment*, *registration*, or *matching*. These concepts are very close to each other and have overlappings. We will explain these methods in the rest of this section.

2.3.1 Image registration

Image registration techniques refer to the task of geometric transformation between input (source) and output (target) images [42, 43]. In this case, they try to align two or more images of the same scene, taken in different conditions like the change in viewpoints or sensors. The authors use some algorithms like correlation-like methods, methods using invariant descriptors and mutual information methods to perform image registration.

2.3.2 Shape correspondence

Shape correspondence methods refer to find important points between the source domain (object) and the target domain (objects) [44]. The correspondence may be done very densely which means trying to find the relevant pixels in the reference image for the corresponding pixels in the target image. Or, a sparse mapping is

done which uses only the important points in the input and output pattern [44]. In figure 2.1 an example of a sparse shape correspondence between two images is illustrated. Certainly, to match different parts of the shape, the corresponding POI has to be identified before

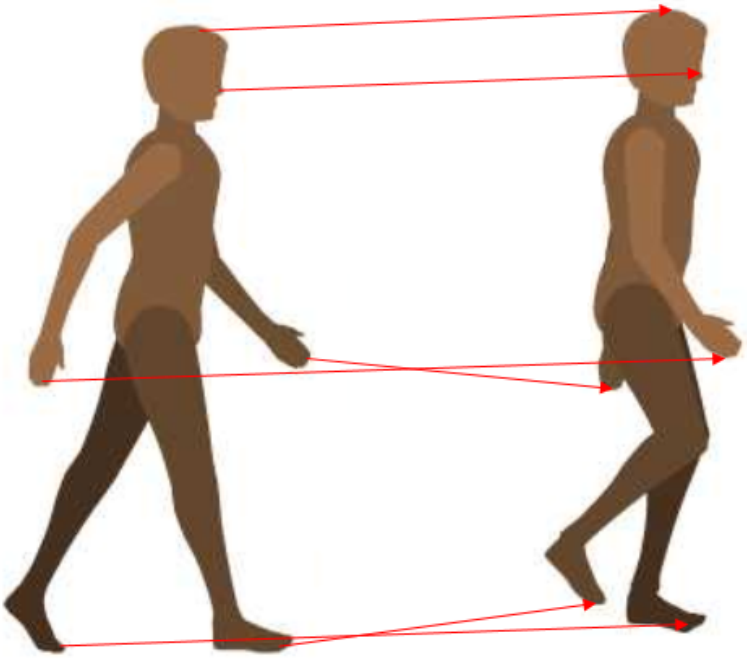

FIGURE 2.1: Sparse corresponding between two images.

2.3.3 Graph matching methods

Some authors (e.g. [45]) tried to see the invariant pattern recognition as a graph matching problem between input and output images. In these methods the graph matching schema offers the ability to match transformed patterns. These methods seem to use the ideas of image registration and shape correspondence and combine them into a formal approach using well-known graph theory.

2.3.4 Dynamic link matching (DLM) methods

The graph matching methods mentioned so far basically are static methods, whereas in nature the matching process is very fast and dynamically changing. This is reflected by the dynamic link methods (DLM). The DLM is based on the biological observation of rapid synaptic plasticity for performing transformations. One of the state-of-the-art methods to overcome the mapping problem is a system proposed in [21]. In this work, they defined a new model which is based on Häussler system which was already proposed in [22] to learn the formation of multiple mappings using the concept of *control units*. In this system, each control unit is responsible for a particular transformation. Nevertheless, also this system has some drawbacks which will be discussed in the next section.

2.4 Conclusion

In the previous section, we have shown that different methods exist for selecting features and performing deformation invariant image recognition. As we mentioned, the methods based on POI have the problem that they are task specific and need some hand-crafted algorithms which change from application to application. Statistical based methods like PCA and ICA have the assumption that the original data should be separated in a linear approach, but this assumption is not valid for most of the data in the real world.

We will propose a new method for universal feature extraction which is more general than the previous tasks which have been done before e.g. transfer learning, semi-supervised learning and self-taught learning. In our contribution, it is not needed for the source task and target task to be related while in knowledge sharing methods it is necessary.
An ANN-based method like deep network tries to solve the nonlinearity problem not solved by PCA and ICA, but they have the problem that these methods are not robust in the case of transformations, translations or change in the position

of the objects.

Although the DLM system shows interesting results for this problem, as we will show in more detail later on, the DLM system has the problem of a high computational complexity. In addition, because it uses the least mean squared error as risk function, the performance for classification is also not optimal. For random values where outliers are present, this system may not work well because outliers influence the mean squared error classification much more than probability-based systems. Therefore, the DLM system should be completed by a modified approach.

In our contribution, first, we try to find features which can be used for multiple classification tasks, and second, we will introduce a new system which employs the information criteria (i.e. probabilities) to overcome the outlier problem of the DLM systems and has a smaller computational complexity.

Chapter 3

Information-Based Universal Feature Extraction

3.1 Introduction

In many real world image based pattern recognition tasks, the extraction and usage of task-relevant features are the most crucial part of the diagnosis[1]. In the standard approach, either the features are given by common sense like edges or corners in image analysis, or they are directly determined by expertise. They mostly remain task-specific, although humans who perform such a task always use the same image features, trained in early childhood. It seems that a *universal feature* set exists, but it is not yet systematically found.

Humans have sought to extract information from imagery ever since the first photographic images were acquired [46]. The most useful basic components are called *features*. Feature extraction and representation are crucial steps for object recognition. One issue is the effective identification of important features in images, and the other one is extracting them. It is a difficult task to obtain a prior knowledge

[1]The results of this chapter published as: (1) Information based universal feature extraction. *In Seventh International Conference on Machine Vision (ICMV2014), volume 9445 of , page 94450D, February 2015* and (2) accepted as :Information Based Universal Feature Extraction in Shallow Networks, *International Journal of Pattern Recognition and Artificial Intelligence*.

of what kind of information is required from the image, even when you know the image domain. Feature extraction is a type of dimension reduction that efficiently represents interesting parts of an image as a compact feature vector with less data. Features are functions of original measurements that are useful for classification and/or pattern recognition. In other words, feature extraction of images is the process of defining a set of image characteristics, which represent most efficiently or significantly the information that is important for analysis and classification; much of the information in the data set may be of little value for discrimination.

There have been many attempts to solve this problem. Dong [47] presents a review on image feature extraction and representation techniques. In his view, there are three feature representations: global, block-based, and region- based features. Chow et al. [48] proposed an image classification approach through a tree-structured feature set. In this approach, they combined both the global and the local features through the root and the child nodes. Tsai and Lin [49] compared global, block-based, and region-based features and their combinations by using a standard classifier over thirty categories. However, it is not clear whether these features are important or not. All those feature definitions seem to arbitrary subjective, not guided by the task specification itself.

However, there is a general agreement that the tools available for analysis of images are not sufficient. Additionally, it is still a challenging problem in computer vision how to extract *universal features* that reflect the fundamental substance of images as complete as possible.

In this context, we might ask: Do *universal features* in images exist such that by using them we are able to efficiently recognize any unknown object? Is it necessary to extract new special features for any new object recognition tasks? How about using existing features from other tasks? Is it possible to use extracted feature of a specific task for other tasks? Are there some general features in natural and non-natural images which can also be used for specific object recognition? For example, can we use extracted features of natural images also for handwritten digit classification? Very little research attention has been paid to these problems

in the last decades. Some people used the concept of *transfer learning* to reuse the knowledge which taken from one classification problem for similar problems. Dan et al. [50] used the knowledge of Latin digits classification for Latin uppercase letters. Raina et al. [41] also used a similar paradigm which is *self-taught learning*, or transfer learning from unlabeled data to use the knowledge of some non-labeled data for supervised classification of groups of animals with limited number of images .

This context proposes an new information based approach and tries to give some answers. Therefore, in this paper we focus on the extraction of very general features that can be useful for object recognition and classification, implemented by a neural network of only two layers. The rest of this chapter is organized as follows. Section 3.2 elaborates the definition of universal features and describes the proposed method for their extraction. In section 3.3, we present the implementation of the method by neural networks, and in section 3.4, some experimental results of this algorithm are shown. Finally, some important conclusions and future potential research directions are shown in Section 3.5.

3.2 Universal feature extraction

In this section, we will develop the notation of **universal features**. How can we show that a feature is universal or not? One criterion is its applicability: an universal feature has to be effective in all applications ever existed and yet to come up. Unfortunately, there is no practical way to prove this. Instead, we will first define the features by theoretical considerations and then show their effectiveness.

Alternatively, we may not need to prove that a certain feature is universal; it rather means that it is not specific to any particular application. For example, textures are common in many computer vision applications which means for many texture features to have a multipurpose nature. For applications of different nature,

texture feature extracting algorithms may determine what morphological particularities are typical in all those applications. The algorithms addressing these types are then also practicable in this scenario.

In this contribution, for task specification we will focus on the question: What kind of features are the best for classifying objects? It is well known that the best strategy for classification is the Bayes decision criterion [51]: given an image x, choose that class ω_k which has the highest conditional probability of occurrence.

$$\omega_k = arg\, max_{\omega_i} P(\omega_i|x) \tag{3.1}$$

Unfortunately, we do not know the conditional probabilities. Instead, we have to estimate them.

Let us assume that we observe pictures x containing an object. Additionally, a teacher will tell us with the decision L if the object is present: $L = 1$ indicates *yes* and $L = 0$ means *no*. Therefore, the observation set consists of pairs (x, L) and the best classification is the one which maximizes the probability $P(L|x)$. Now, instead of using the whole picture, only a small set of features $h_1, \ldots, h_n$ extracted from x by a function $h(x)$ should be sufficient for detecting the object. How can we find it? Let us first consider just one feature h. This means, that the probability of the correct decision for the presence of object $P(L|x)$ should be as close to $P(L|h)$ as possible. Since the probability for correct classification is based on the conditional probabilities, the distance between the two probability distributions can be seen as a measure for the classification quality, implementing the Bayes decision. It is well known that the Kullback-Leibler distance

$$D(P(L|x)||P(L|h)) = \sum_L P(L|x) \log \frac{P(L|x)}{P(L|h)} \tag{3.2}$$

becomes zero if and only if the two probability distributions become equal [52]. It implements the difference

$$\begin{aligned}\sum_L P(L|x)\log\frac{P(L|x)}{P(L|h)} &= \sum_L P(L|x)\log P(L|h) - \sum_L P(L|x)\log P(L|x)\\ &= H_L(h) - H_L(x)\end{aligned} \tag{3.3}$$

between the estimated Shannon information $H_L(h)$ and the observed information $H_L(x)$ of the image pattern x for the teacher classification decision L.

Now, we have a problem: since $h(x)$ is an unknown function, we do not know $P(L|h)$. Instead, we can estimate it by a function $g(L|h)$ which does depend on the decision L, but is indeed a function of h only. Therefore, we note it by $g_L(h)$. Nevertheless, if we maintain $0 < g < 1$ the Kullback-Leibler distance will still become zero if the two probability distributions become equal. Therefore, we might use the expected distance as an objective function R for setting up the unknown function.

$$\begin{aligned}R &= \sum_x P(x)D(P(L|x)||g_L(h))\\ &= \sum_x\sum_L P(x)P(L|x)\log\frac{P(L|x)}{g_L(h)}\\ &= \sum_x\sum_L P(L,x)\log\frac{P(L|x)}{g_L(h)}\\ &= \sum_x\sum_L P(L,x)\log P(L|x) - \sum_x\sum_L P(L,x)\log g_L(h)\end{aligned} \tag{3.4}$$

The objective function is composed by two additive terms. The first term does not depend on the unknown function g, remaining constant when changing g. Therefore, for *minimizing* R it suffices to *maximize* the new risk function

$$R(g,h) = \sum_x\sum_L P(L,x)\log g_L(h) = \langle\log g_L(h(x))\rangle \tag{3.5}$$

The expectation $\langle . \rangle$ is computed over all values of x and L. This is also covered by the uniformly distributed M observations $(x(i), L(i))$ where $i = 1, \ldots, M$ by

$$R(g,h) = \frac{1}{M} \sum_{i=1}^{M} \log g_L(h(x(i))) \tag{3.6}$$

In our observation set, each $x(i)$ is accompanied by the teacher decision $L(i) \in \{0,1\}$. For $L(i) = 1$, the feature should be present to show the presence of the object. Assuming the probability $g_L(h)$ for $L = 1$ is $g(h)$, then for the second case $L = 0$ the probability must be $(1 - g(h))$. Therefore, the term $\log g(h)$ in the objective function can be written as

$$\log g_L(h) = L(i) \log g(h) + (1 - L(i)) \; log(1 - g(h)) \tag{3.7}$$

and the objective function becomes

$$R(g,h) = \frac{1}{M} \sum_{i=1}^{M} L(i) \log g(h) + (1 - L(i)) \log(1 - g(h)) \tag{3.8}$$

This function is well known as *maximum likelihood objective function*[53]. It should be mentioned that for N independent objects, the probabilities of the multiple output $\mathbf{h} = (h_1, \ldots, h_N)$ factorize

$$g_L(\mathbf{h}) = \prod_{k=1}^{N} g_{Lk}(h_k) \tag{3.9}$$

and the log probability becomes by eq.(3.7)

$$\log g_L(\mathbf{h}) = \sum_{k=1}^{N} L_k(i) \log g_k(h_k) + (1 - L_k(i) \log(1 - g_k(h_k))) \tag{3.10}$$

Thus, our objective risk function forms a sum over all single risks

$$R(g,\mathbf{h}) = \frac{1}{M} \sum_{i=1}^{M} \sum_{k=1}^{N} L_k(i) \log g_k(h_k) + (1 - L_k(i) \log(1 - g_k(h_k))) \tag{3.11}$$

Now, how can we obtain the unknown functions g and h? Let us assume that

we use parameterized functions, i.e., the necessary information for extracting and using the features are stored in a finite set of parameters. For m features, we assume m extraction functions $h_i(x)$, each one containing n parameters

$$h_j(x) = h_j(\mathbf{u}, x) \quad with \ \mathbf{u} = (u_1, \ldots, u_n)$$

The object detection function $g(\mathbf{y})$ is determined by s parameters

$$g(\mathbf{h}(\mathbf{u}), \mathbf{w}) \quad with \ \mathbf{h} = (h_1, \ldots, h_m) \quad and \ \mathbf{w} = (w_1, \ldots, w_s)$$

Thus, the task of determining the universal features becomes a task of determining the appropriate parameters of the unknown functions.

3.3 Learning the feature extraction

In this section, we will describe our approach for extracting the universal features by minimizing the objective function. Unfortunately, the desired solution is problem dependent, i.e. it depends on our observation set. One common approach for minimizing an objective function, if there is no analytic solution, is the stepwise iteration of an approximation expression, a so-called *learning algorithm*, using the observations as *training set*.

As learning algorithm for the parameters $\mathbf{w}$ we might use the well-known stochastic gradient ascend for maximizing R,

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \gamma(t) \frac{\partial R(g(\mathbf{w}))}{\partial \mathbf{w}} \tag{3.12}$$

which does not use the expectation value over M samples of the objective function R,

$$R(g, \mathbf{h}) = \frac{1}{M} \sum_{i=1}^{M} L(i) \log g(\mathbf{h}) + (1 - L(i)) \log(1 - g(\mathbf{h})) = \frac{1}{M} \sum_{i=1}^{M} R_i(g, \mathbf{h}) \tag{3.13}$$

but its stochastic version for the $i-th$ sample pairs $(x(i), L(i))$

$$R_i(g, \mathbf{h}) = L(i) \log g(\mathbf{w}, h(\mathbf{u}, i)) + (1 - L(i)) \log(1 - g(\mathbf{w}, h(\mathbf{u}, i))) \tag{3.14}$$

For further computations, let us drop the index i from the notation, since the formulas should be applied to all pairs $(x(i), L(i))$ of the training set. The gradient of the stochastic objective function becomes

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} R(\mathbf{w}, \mathbf{u}) &= \frac{\partial}{\partial \mathbf{w}} L \log g(\mathbf{w}) + (1 - L) \log(1 - g(\mathbf{w})) \\ &= \frac{L}{g(\mathbf{w})} \frac{\partial g(\mathbf{w}, y)}{\partial \mathbf{w}} - \frac{1 - L}{1 - g(\mathbf{w})} \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} \\ &= [\frac{L}{g} - \frac{1 - L}{1 - g}] \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} \\ &= (\frac{L(1 - g) - g(1 - L)}{g(1 - g)}) \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} = (\frac{L - g}{g(1 - g)}) \frac{\partial g(\mathbf{w})}{\partial \mathbf{w}} \end{aligned} \tag{3.15}$$

For the second set of parameters $\mathbf{u}$ we proceed analogously. Here, our learning algorithm is:

$$\mathbf{u}(t + 1) = \mathbf{u}(t) + \gamma(t) \frac{\partial R(g(\mathbf{u}))}{\partial \mathbf{u}} \tag{3.16}$$

and the gradient becomes

$$\frac{\partial}{\partial \mathbf{u}} R(\mathbf{w}, \mathbf{u}) = (\frac{L - g}{g(1 - g)}) \frac{\partial g(\mathbf{w}, h(\mathbf{u}))}{\partial \mathbf{u}} \tag{3.17}$$

For estimating the unknown function g and the parameters $\mathbf{w}$ by we learn the parameters by iteratively analyzing the data.

3.3.1 The neural net for extracting one feature

Now, we have to choose the kind of functions $g(.)$ and $h(.)$ to use here. It is well known that all continuous functions can be approximated sufficiently well by two layers networks using sigma neurons and squashing functions S as output functions [23]. Therefore, choosing our approximation functions like this will not limit our approach in any way. With the image input described by a pixel tuple $\mathbf{x}$, we might

choose as extraction function a squashing function with an affine argument

$$h_j(\mathbf{u}, \mathbf{x}) = S(z) \quad with\ z = \mathbf{u}^T\mathbf{x}$$

and as object detection function for one object we choose the Fermi function

$$g(\mathbf{w}, \mathbf{h}) = S_F(v) \quad with\ \ S_F(v) = \frac{1}{1+\exp(-v)} \quad and\ \ v = \mathbf{w}^T\mathbf{h}$$

This can be interpreted as having a first layer of formal neurons, implementing sigma neurons and squashing function $h(\mathbf{u}, \mathbf{x})$, and a second layer, implementing the object detection function $g(\mathbf{h}, \mathbf{w})$. In figure 3.1 the two layer architecture is shown with N output units, each one detecting a different object.

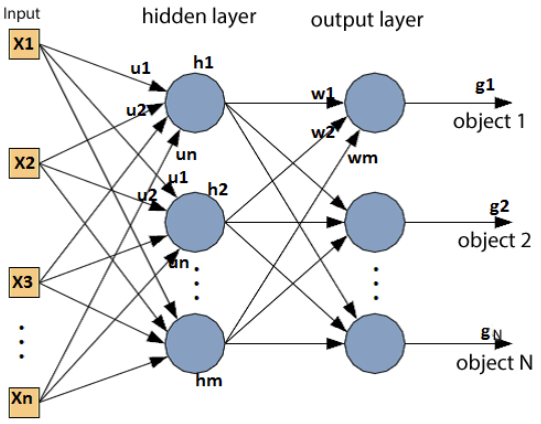

FIGURE 3.1: The network architecture for function approximation

Now, to obtain the desired iteration equations, the learning rules, we use the standard back-propagation approach for our risk function and compute the necessary derivatives.

In our approach and our learning rules we have the properties $0 \leq g \leq 1$ and $0 \leq L \leq 1$. This is covered by the choice of the Fermi function $S_F(v) = \frac{1}{1+\exp(-v)}$ as squashing function of the output layer with $dim(\mathbf{w}) = dim(\mathbf{h}) = m$ and the

derivative

$$\begin{aligned}\frac{\partial g}{\partial v} &= \frac{\partial S_F(v)}{\partial v} = \frac{\partial}{\partial v}(1+e^{-v})^{-1} = (1+e^{-v})^{-2}e^{-v} = \frac{1+e^{-v}-1}{(1+e^{-v})(1+e^{-v})} \\ &= g(1-g)\end{aligned} \tag{3.18}$$

For the first layer, the *hidden layer*, we get

$$h_j(\mathbf{u},\mathbf{x}) = S_t(z) = \ \ e.g.\ \ tanh(z)\ \ with\ \ z = \mathbf{u}^T\mathbf{x}$$

Therefore, the derivatives in equations 3.12 and 3.16 become

$$\begin{aligned}\frac{\partial}{\partial \mathbf{w}}R(\mathbf{w},\mathbf{u}) &= (\frac{L-g}{g(1-g)})\frac{\partial g(\mathbf{w},h(\mathbf{u}))}{\partial \mathbf{w}} \\ &= (\frac{L-g}{g(1-g)})\frac{\partial g}{\partial v}\frac{\partial v}{\partial \mathbf{w}} \\ &= (\frac{L-g}{g(1-g)})g(1-g)\mathbf{h} = -(g-L)\mathbf{h}\end{aligned} \tag{3.19}$$

and

$$\begin{aligned}\frac{\partial}{\partial \mathbf{u}}R(\mathbf{w},\mathbf{u}) &= (\frac{L-g}{g(1-g)})\frac{\partial g(\mathbf{w},h(\mathbf{u}))}{\partial \mathbf{u}} \\ &= (\frac{L-g}{g(1-g)})\frac{\partial g}{\partial v}\frac{\partial v}{\partial \mathbf{u}} \\ &= (\frac{L-g}{g(1-g)})g(1-g)\frac{\partial v}{\partial \mathbf{u}} = -(g-L)\frac{\partial v}{\partial \mathbf{u}}\end{aligned} \tag{3.20}$$

The $s-th$ term of the vector $\frac{\partial v}{\partial \mathbf{u}}$ is

$$\frac{\partial v}{\partial u_s} = \sum_{j=1}^{m} w_j \frac{\partial h_j}{\partial u_s} = \sum_{j=1}^{m} w_j S'(z_j)\frac{\partial z_j}{\partial u_s} = \sum_{j=1}^{m} w_j S'(z_j)x_s \tag{3.21}$$

By this, our learning equations 3.12 and 3.16 become

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \gamma_1(t)(g-L)\mathbf{h} \tag{3.22}$$

$$\mathbf{u}(t+1) = \mathbf{u}(t) - \gamma_2(t)(g-L)\sum_{j=1}^{m} w_j S'(z_j)\mathbf{x} \tag{3.23}$$

with e.g. $S_t'(z_j) = 1 - h_j^2$.

Now, for learning we assume several important restrictions:

- Each extraction function h_j covers a different part of input $\mathbf{x}$, i.e., it has an unique receptive field and is not completely overlapping with other fields, see figure 3.2. This means, that the tuple of input pixels $\mathbf{x}$ is different for each extraction unit j, denoted by $\mathbf{x}_j$.

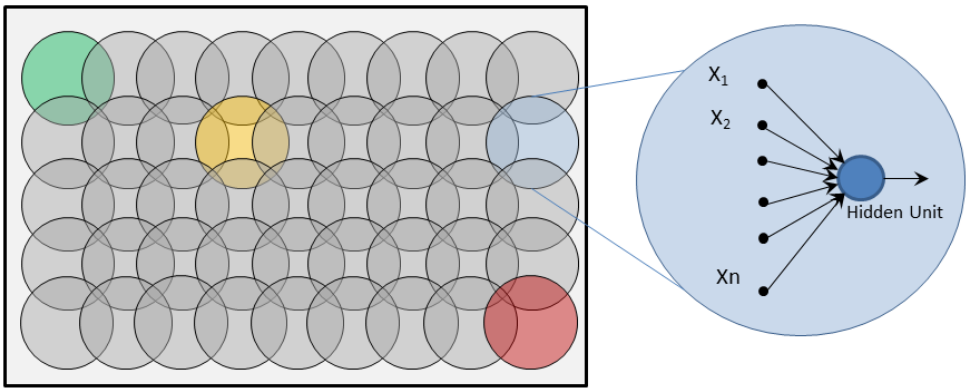

FIGURE 3.2: The receptive field patch extraction from an image

- The object should be recognized everywhere on the image. Therefore, in order to train only the statistics and avoid overfitting, we put the constraint that the parameters $\mathbf{u}$ of each extraction function are the same ones, i.e., all hidden neurons share the same k weights.

- There can be more than one object present, i.e. N ones which should be recognized independently. Therefore, we assume not one, but N functions g_k, i.e., N output units.

- Training a weight will also result in training neighboring weights by a certain degree.

An important decision in this network is that we use the weight sharing idea in the feature extraction layer. Using weight sharing has two advantages: First, it reduces the number of parameters for learning, and second, all neurons learn to detect the same features, although their receptive fields are located at different positions in the input image.

The number of output neurons depends on the number of classes (sets) that we need or how many sets we want for classification. Therefore, the weights $\mathbf{w}$ in the last layer are not shared and specific for the detected classes. In figure 3.3 the overall architecture is shown.

We use the first layer (U) as feature extractor and the second layer (W) as

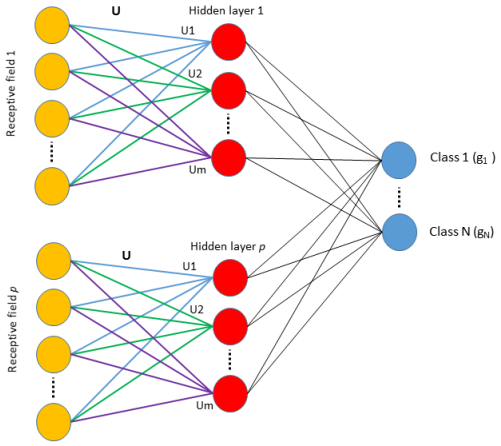

FIGURE 3.3: The architecture of the two layers neural network for universal feature extraction

classifier layer. Since all outputs $g_k(\mathbf{w}_k, \mathbf{h})$ can be computed independently from each other, the stochastic gradient learning rule does not change much.
For the $k-th$ output unit, we get by eq.(3.22)

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) - \gamma_1(t)(g_k - L_k)\mathbf{h} \tag{3.24}$$

and equation (3.23) becomes by all N output units

$$\mathbf{u}(t+1) = \mathbf{u}(t) - \gamma_2(t) \sum_{k=1}^{N} (g_k - L_k) \sum_{j=1}^{m} w_{kj} S'(z_j) \mathbf{x}_j \tag{3.25}$$

and each component of $\mathbf{u}$ (u_{rs}) in two dimensional view of weights has a little effect on its neighbors u_{mn} by

$$u_{mn}(t+1) = u_{mn}(t) + \aleph(m, n, r, s) u_{rs}(t+1) \tag{3.26}$$

where

$$\aleph(m, n, r, s) = e^{-((r-m)^2+(s-n)^2)/2\sigma^2} \tag{3.27}$$

is the neighborhood function and variable σ in this equation is related to neighborhood radius. It is updated for each unit j differently. In figure 3.4 this is shown.

The input samples are no longer treated similarly by the extraction units $h_j(x)$,

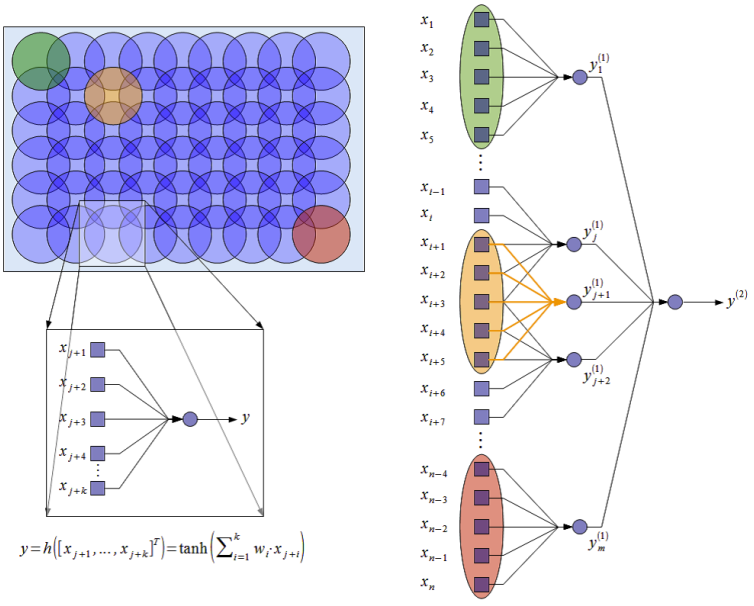

FIGURE 3.4: The input samples covered by the first layer by two-dimensional overlapping receptive fields (left) or one-dimensional overlapping receptive fields (right) (from [24])

but they are grouped into subsets. Each unit j processes only a subset x_j. The input samples can be arranged in different manners. On the left hand side of figure 3.4 the samples are arranged in a two-dimensional manner, e.g. like pixels of an image. The one-dimensional case is shown on the right hand side of figure 3.4 e.g. for a speech signal with $k = 5$.

As you can also see in figure 3.4, we extract several patches from each image and use them as inputs for the network. The number of patches that can be extracted from an image depends on some factors, e.g. the size of a patch, the size of the image and the number of pixels shared between two neighbor patches. For instance, the number of rectangular patches which can be taken from an image with 60×80 pixels and a patch size of 9×9 sharing three pixels is 108. Please note that we

extract square patches instead of circular ones because it is computationally more feasible.

There are still some open questions for this kind of architecture:

- What is the best size of a receptive field (patch)?
- What is the optimum number of hidden units?

We will discuss these questions in later sections presenting some experimental results. It is clear that, by increasing the size of the image, we need more receptive fields and more parameters in the subsequent layer. Instead, it might be better to increase the receptive field size for covering the image by a smaller number of fields.

3.3.2 Extracting several features

Our feature extraction analysis of the previous section only covers just one feature in each receptive field, the most important one. How do we get additional, helpful features? Let us assume that in each receptive field we extract not only one feature, but r ones. Then, each extraction result h_j of receptive field j has several components

$$h_j = (h_1(\mathbf{u}_1, \mathbf{x}_j), \ldots h_r(\mathbf{u}_r, \mathbf{x}_j))^T \quad with \;\; h_i(\mathbf{u}_i, \mathbf{x}_j) = S(z_{ij}), z_{ij} = \mathbf{u}_j^T \mathbf{x}_j \tag{3.28}$$

The corresponding network architecture is shown in figure 3.5. On the left hand side, we see the two-dimensional input sample image covering. On the right hand side, the corresponding one-dimensional receptive fields are shown. The activity of the second layer, the object detection, will not change except of the fact that for each output unit the number of inputs becomes mr instead of m.

$$g(\mathbf{w}, \mathbf{h}) = S_F(v) \;\; and \;\; v = \mathbf{w}^T \mathbf{h} \;\; and \;\; dim(\mathbf{w}) = dim(\mathbf{h}) = mr \tag{3.29}$$

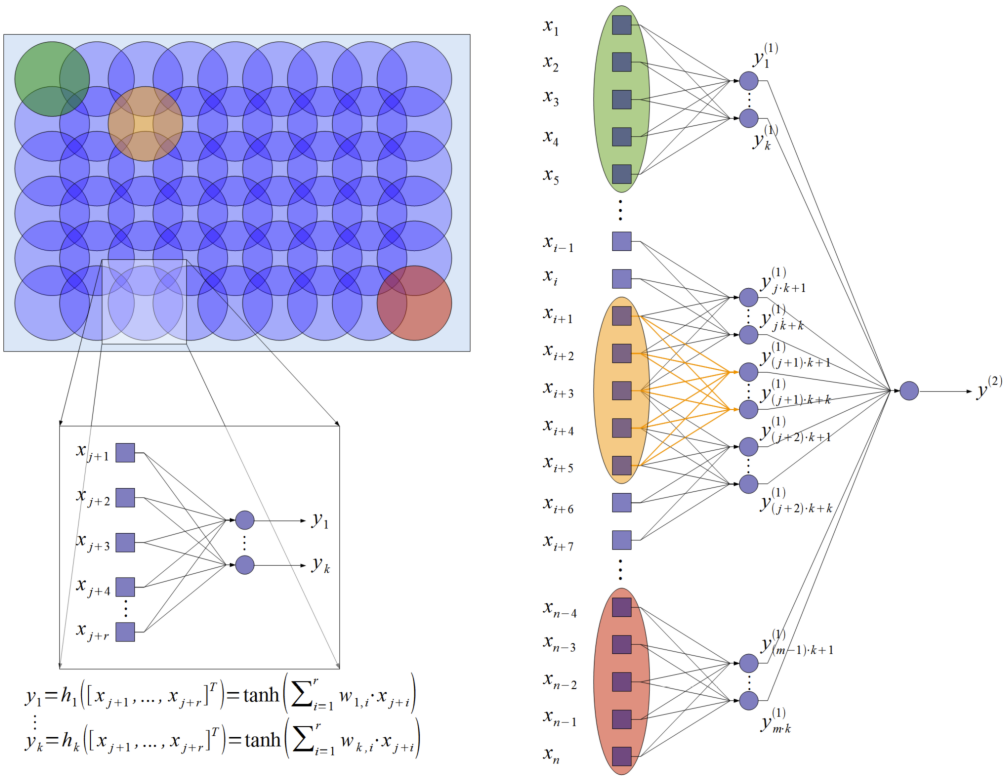

FIGURE 3.5: The extraction of multiple features (from [24])

Certainly, the learning equations change with the additional features. Equation (3.22) has now mr components, and equation (3.23)) becomes for the s-th feature

$$\mathbf{u}_s(t+1) = \mathbf{u}_s(t) - \gamma_2(t)(g-L)\frac{\partial v}{\partial \mathbf{u}_s} \tag{3.30}$$

With the activity of

$$v = \mathbf{w}^T\mathbf{h} = \sum_{p=1}^{rm} w_p h_p = \sum_{i=1}^{r}\sum_{j=1}^{m} w_{ij}h_{ij} + w_0 \tag{3.31}$$

containing the $i-th$ feature of the $j-th$ receptive field, we get the derivative of

$$\frac{\partial v}{\partial \mathbf{u}_s} = \frac{\partial}{\partial \mathbf{u}_s}\sum_{i=1}^{r}\sum_{j=1}^{m} w_{ij}h_{ij} = \sum_{i=1}^{r}\sum_{j=1}^{m} w_{ij}\frac{\partial}{\partial \mathbf{u}_s}h_{ij}(\mathbf{u}_i, \mathbf{x}_j) \tag{3.32}$$

Since the $i-th$ feature extraction function $h_{ij}(\mathbf{u}_i, \mathbf{x}_j)$ does only depend on the $i-th$ parameter vector $\mathbf{u}_i$, we get zero for all terms where $i \neq s$

$$\frac{\partial v}{\partial \mathbf{u}_s} = \sum_{i=1}^{r}\sum_{j=1}^{m} w_{ij}\frac{\partial}{\partial \mathbf{u}_s}h_{ij}(\mathbf{u}_i, \mathbf{x}_j) = \sum_{j=1}^{m} w_{sj}S'(z_{sj})\mathbf{x}_j \tag{3.33}$$

and our learning equation becomes

$$\mathbf{u}_s(t+1) = \mathbf{u}_s(t) - \gamma_2(t)(g-L)\sum_{j=1}^{m} w_{sj}S'(z_{sj})\mathbf{x}_j \tag{3.34}$$

using k inputs x_{kj} at each receptive field j, obtaining the feature $y_{sj} = S_t(z_{sj}) = e.g.\tanh(\mathbf{u}_s^T\mathbf{x}_j)$ with $S'(z_{sj}) = 1 - \tanh^2(z_{sj})$.

For N outputs, equation(3.22) changes to

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) - \gamma_1(t)(g_k - L_k)\mathbf{h} \tag{3.35}$$

and equation(3.34) is determined by all N output units

$$\mathbf{u}_s(t+1) = \mathbf{u}_s(t) - \gamma_2(t)\sum_{k=1}^{N}(g_k - L_k)\sum_{j=1}^{m} w_{ksj}S'(z_{sj})\mathbf{x}_j \tag{3.36}$$

and here also like in 3.26 and 3.27 each component of $\mathbf{u}$ (u_{pq}) in two dimensional view of weights has a little effect on its neighbors u_{mn} by

$$u_{mn}(t+1) = u_{mn}(t) + \aleph(m,n,p,q)u_{pq}(t+1) \tag{3.37}$$

Now, there are r features learned by each of the receptive fields. How can we assume that they will be different although they have the same input and the same learning rules? The answer lies in the fact that all feature vectors $\mathbf{u}_s$ have different feedback from the second layer, depending on their own activity h_s. This leads to different learning behavior and different convergence states.

3.3.3 Training and testing strategies

There are several training strategies which distinguish among different learning modes:

1. Parallel training: All weights w_{kij} and u_{sp} are updated after the whole net has determined its activity. This strategy is preferable, but suffers from strong convergence problems of the network parameter iteration.

2. Sequential training: We first train $\mathbf{u}_1$ and the corresponding w_{k1} until convergence, not using the other $\mathbf{u}_2, \ldots, \mathbf{u}_r$. Then, leaving $\mathbf{u}_1$ constant, we train $\mathbf{u}_2$ and the w_{k1} and w_{k2}, still not using the other $\mathbf{u}_3, \ldots, \mathbf{u}_r$. After this, we train $\mathbf{u}_3$ including $w_{k1}, \ldots, w_{k3}$ and so on, until all the other feature vectors are determined.

3. Batch vs. stochastic training: Parallel or sequential trainings can be used as elements in a more comprehensive strategy, the use of batch offline or stochastic online learning. Let us show this for the proposed back-propagation scheme by the following nested loops of pseudo code for batch training.

In contrast to batch offline learning, stochastic online learning differs slightly. It does not take the average over all corrections, but use them instantly. Thus, each learning step is based on the previous one and not on the average over all patterns. For the training procedure, we might ask the following questions: Does the performance decrease with increasing system size k? Does the performance increase with increasing number f of features? The results of these different training procedures are different. The stochastic training converges faster, but has higher performance variations than the batch procedure.

3.4 Experimental results

In this section we report several results using the ideas and algorithms presented so far. First, we discuss the setup of the training procedure and some of the network

Algorithm 1 Algorithm of Batch Training (*offline learning*)

for all features f **do**
  for all units k, increasing system size **do**
    for all cross-validations p **do**
      reset weights w_{kf} and $\mathbf{u}_f$ for unit k and feature f
      for all iteration steps t **do**
        for all patterns i of the training set **do**
          compute the activity $(\mathbf{w}, \mathbf{u})$ of the current network layers
          compute the corrections Δw_{kf} and $\Delta \mathbf{u}_f$ for the $k-th$ unit and the $f-th$ feature
        end for i
        update the $k-th$ weights w_{kf} and $\mathbf{u}_f$ for feature f
        compute the objective function R(training set), R(test set)
      end for t
      change training and test set
    end for p
    compute the average of $R(.)$ of all training and test sets p for one system size k
  end for k
  compute $R(.)$ of the full system size and one feature f
end for f

Algorithm 2 Algorithm of Stochastic Gradient (*online learning*)

for all features f **do**
  for all units k, increasing the system size **do**
    for all cross-validations p **do**
      reset all weights w_{kf} and $\mathbf{u}_f$ of unit k and feature f
      for all iteration steps t **do**
        for all patterns i of the training set **do**
          compute the activity $(\mathbf{w}, \mathbf{u})$ of the current network layers
          compute the corrections Δw_{kf} and $\Delta \mathbf{u}_f$ for the k-th unit and $f-th$ feature
          update the $k-th$ weights w_{kf} and $\mathbf{u}_f$ for feature$f$
        end for i
        compute objective function R(training set) and R(test set)
      end for t
      change training and test set
    end for p
    compute average of $R(.)$ of all training and test sets p for one system size k
  end for k
  compute $R(.)$ of the full system size for one feature f
end for f

parameters used. Then, the results of training and testing with different kind of images and parameters are reported.

3.4.1 Network parameters

To prepare the network for training, several decisions have to be taken before. First, let us discuss the general decisions which are taken for training and testing.

Activation Functions: There are number of common activation functions in use for artificial neural networks e.g. *tanh*, the step function, Gaussian function for RBF nets, the *logistic sigmoid function* $f(x) = 1/(1 + e^{-x})$ or the *bipolar sigmoid function* $f(x) = (1 - e^{-x})/(1 + e^{-x})$. In some literature, e.g. [54], it is emphasized that, although selection of an activation function for a neural network or its node is an important task, other factors like the training algorithm, the network size or the learning parameters are more vital for a proper training of the network. In [55] it has been shown that for general purpose *bipolar sigmoid*, *unipolar sigmoid* and *tanh* functions are better than others. In our case, we used the bipolar *tanh* activation function for the hidden layer units of the network and the unipolar sigmoid function for the output units of the second layer, because the output should show the amount of probability that an input object may be in a class. Therefore, the output function has to take values between zero and one.

Weight Initialization: There are also many possible algorithms for initializing the weights for feed forward neural networks [26, 56, 57]. One method is the usual weight initialization: an uniform random initialization inside the interval $[-0.05, +0.05]$ or $[-0.01, +0.01]$. For large number of inputs the smaller random interval is preferred to avoid the saturation of the sigmoids. Random weight initialization is still the most popular method because of simplicity and comparable results with other methods [26, 56].

In [57] Kim proposed a minimum bound for the weight initialization. The initialization is still random, but satisfying a minimum value. In the equation, the

minimum is the learning step used in the back-propagation training after initialization. In the reference, the initialization procedure is not clearly specified because there is just a lower bound and not an upper one.

$$\sqrt{(\gamma/n_{input})} < |w_{ij}| \tag{3.38}$$

In [26] there is just a maximum bound for the weights and the initialization is still random, but satisfying the maximum.

$$|w_{ij}| < 2.4/n_{input} \tag{3.39}$$

The variable n_{input} refers to the number of input units, w_{ij} refers to the weight between neuron j and input i and γ refers to the learning rate.

We used the method of random uniform distribution with interval $[-0.01, +0.01]$ for initializing the weights because it is simple and our experimental result showed that it performs better than the methods proposed in [57] and [26].

Learning Rate: In all tests in training and test phase the learning rate is $\gamma = 0.005$.

3.4.2 Input data preparation

Some object images are taken from the Amsterdam library of object images (ALOI) database [58]. ALOI is a color image collection of 1000 small objects, recorded for scientific purposes. In order to capture the sensory variations in object appearance, they systematically varied viewing angle, illumination angle, and illumination color for each object and additionally captured wide baseline stereo images. They recorded over a hundred images of each object, yielding a total of 110,250 images for the whole collection[58]. Objects can be characterized as natural (e.g. an apple or an orange) or artificial (e.g., a hat or a cup), see figures 3.6 and 3.7.

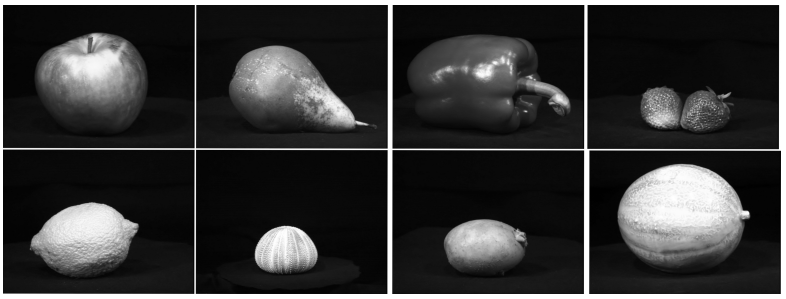

FIGURE 3.6: Some examples of natural objects

FIGURE 3.7: Some examples of artificial objects

We placed the selected objects in the middle of some natural or artificial background images and used shifted variations of three pixels left, right, up or down, maximally. By this, we prepared six sets of data. For preprocessing the input images, we normalized each input pixel set $\mathbf{x}$ to zero mean and unit variance of all pixel values. It is also possible to normalize only the set of extracted patches instead of normalizing the whole image, but the result of image normalization was better. The size of the objects to recognize is 80×60 pixels. Objects in sets 8 to 11 in addition of having different viewing angle, illumination angle and illumination color, have also different scales (1, 2 and 2.5). These sets are multi scale versions of set 1 to 4 and are designed for training and testing scale invariance. These objects are placed before different backgrounds. For example, figures 3.8 and 3.9 show different objects with multiple scales, illumination and viewing angles, placed before different background images.

FIGURE 3.8: Example objects located on multiple **natural** background

FIGURE 3.9: Example objects located on multiple **artificial** background

Additionally, we used different kinds of datasets including MNIST handwritten digits, KTH-TIPS2 and UIUCTEX texture to validate that features are universal.

The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. The digit images have 28×28 pixels [59]. Initially, before use we normalized the size and centered it in a fixed-size image.

The KTH-TIPS (Textures under varying Illumination, Pose and Scale) image database was created to extend the CUReT database in two directions, by providing variations in scale as well as pose and illumination, and by imaging other samples of a subset of its materials in different settings. The KTH-TIPS2 databases took this a step further by imaging 4 different samples of 11 materials, each under varying pose, illumination and scale [60]. The UIUC texture database features 25 texture classes, 40 samples each. All images are in grayscale JPG format, 640×480 pixels [61].

In figures 3.10, 3.11 and 3.12 you can see some examples of these three datasets. These sets were chosen to represent unnatural objects. If features are *universal*, they have to represent efficiently also those objects. Examples of the resulting training and test objects are shown in figure 3.13 .

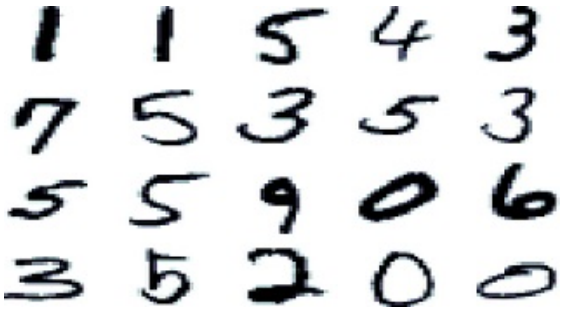

FIGURE 3.10: Some examples of MNIST handwritten digits

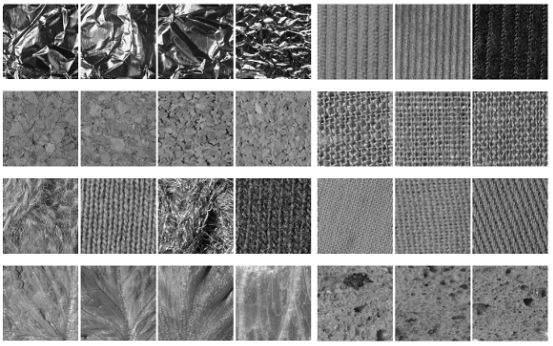

FIGURE 3.11: Some examples of KTH-TIPS2 texture dataset

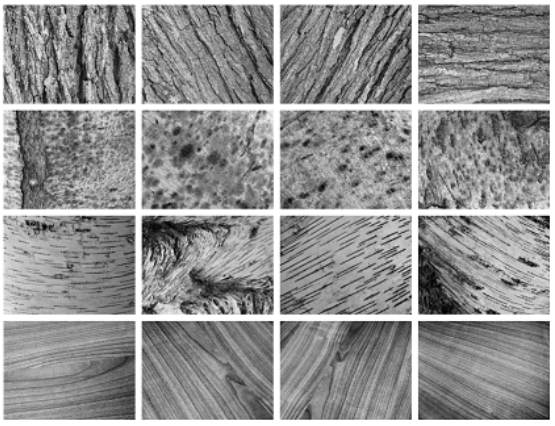

FIGURE 3.12: Some examples of UIUC texture dataset

The following table 3.1 gives an overview of the composition of the different training and test sets. In set 2, objects are shifted a little from the center and they have different view and illuminations with many natural backgrounds. In set 4, the same objects are used as in set 2, but they use non-natural backgrounds. In sets 1 and 3 objects are natural (like apples or potatoes) with the same backgrounds as in sets 2 and 4 respectively. Data sets 8 to 11 have multiple scales in addition to multiple views and illuminations and natural or non-natural background, similar to sets 1 to 4 respectively.

FIGURE 3.13: Image examples of the training and test sets

Set label	Object	Background	Scale
set1	natural	natural	1
set2	artificial	natural	1
set3	natural	artificial	1
set4	artificial	artificial	1
set5	MNIST handwritten digits		1
set6	KTH texture		1
set7	UIUC texture		1
set8	natural	natural	1, 2, 2.5
set9	artificial	natural	1, 2, 2.5
set10	natural	artificial	1, 2, 2.5
set11	artificial	artificial	1, 2, 2.5

TABLE 3.1: The composition of the training and test sets

3.4.3 Does the network learn universal features?

We trained the system with 40,000 input images in 10 different groups with 16 neurons in the hidden layer. Each group included one object with multiple illuminations and view angles, placed in the middle of many background images, with a maximum of three pixel shift in right or left and up or down. The size of a receptive field was set to 9 by 9, and each receptive field shared three pixels with its neighbors. This value was determined experimentally; it gives better result than others. After convergence of the network, we fixed the value of the first layer (U), the features, and used it as feature extractor for further processing.
The weights of the second layer were trained separately to classify multiple objects with multiple backgrounds. After training, it could classify 10 groups (according to the objects in the images) of data set images. For evaluation, we used as classification accuracy

$$Accuracy = 0.5(Prob(PT) + Prob(NF)) \tag{3.40}$$

Please note that, for calculating the rate of accuracy, we had to record the positive (true) PT and negative (false) NF system classification decisions. If we just use the positive input PT rate to compute the accuracy rate, by changing the threshold value we can get better results. Therefore, for a fair comparison, we had to take both rates into account. In general, a ROC analysis have to be computed, but the averaged correct decision is sufficient for this application. For more information about ROC analysis, see [62, 63]. For computing the probabilities, we used the classification output of the neural network units. Because the output of the units is between zero and one, to assign an object to a class we selected the maximum value of the output in accordance to the Bayes classification rule. Thus, the object is the member of a class with the maximum output value [2]

$$choose\ C_i\ if\ g_i = max(g_j)\ j = 1..N \tag{3.41}$$

For instance, after using set 2 as training set to learn the features, the test revealed

that with 82.28% accuracy set 1 was correctly classified, and with 91.97% accuracy set 4. Set 1 includes natural objects and set 4 includes artificial objects, see table 3.2 for more results. After this, we set up the second layer to classify the MNIST handwritten digits by using 60,000 data for training and 10,000 data for test. As result, it could classify ten groups (0-9) of the handwritten digit images of the test set with 90.97% accuracy. It is interesting to know that by using the MNIST exclusively for training the features, the rate of correct accuracy was 89.25%. The small difference between the results shows that both sets had the same statistical proportions, giving rise to quite optimal features used for digit classification.
In comparison to this, the result of the handwritten digit recognition by the LDA classifier implemented in the Matlab software package was only 87.6%. The best result for handwritten digit classification reported in literature is 99.77% for the training error and was obtained using a special 6 layer non-linear neural network, each layer stacked on top of another one (convolutional neural network) [50]. Consider that this result was not obtained by universal features and their test set results should be worse, our results are very good.

If we use the same dataset both for feature learning and classification, the result was close to using a different set for feature learning. This means that, by using universal feature extraction, we loose almost no accuracy and in some cases (like set 5) we got the better result.
In table 3.2 you see more results of this test. All results of this table show the average accuracy of 10 times running for each test to obtain a robust result . In all cases (including the results of this chapter and next chapter) the variance was small (about 0.6) and the confidence interval also was very small. The table displays the rate of accuracy (in percent) with multiple training and test sets for feature extraction and classification. In the four last columns, we used feature weights trained by the union of multiple data sets. We can see that, if the training set material is sufficient rich, adding other sets to this set does not have much effect on the rate of accuracy, though the result is a little bit better. In figure 3.14, we can see the effect of using multiple data sets for training. It shows that,

[2]It is also possible to use a Softmax policy [64] to assign an object to a specific class.

Test Set \ Training Set	set 1	set 2	set 3	set 4	set 5	sets 1,5	sets 1,6	sets 5,6	sets 1,5,6
set 1	85.11	82.28	85.49	84.57	66.30	71.35	83.63	81.63	83.90
set 2	86.32	85.37	87.02	87.30	73.72	77.57	85.22	83.66	84.68
set 3	92.18	90.13	94.47	93.66	73.93	81.93	93.22	88.44	91.17
set 4	91.83	91.97	92.66	92.48	83.41	87.16	91.42	91.50	91.34
set 5	91.58	90.97	91.91	92.10	89.25	89.72	91.24	91.71	91.85
set 6	76.70	80.20	78.00	76.40	65.38	67.32	85.33	88.47	86.32
set 7	98.30	99.20	99.50	99.60	98.00	99.7	100	100	100

TABLE 3.2: Universal feature learning: test results

by using more data sets, the results are about the same or improve also. In the case of handwritten digits the result of using multiple sets of different data for feature extraction is much better than using only handwritten digits both for feature extraction and classification. The digits are graphically very simple; using statistically more diversified images in the training will lead to more complex features and improve the results. The low accuracy of class 6 by a system trained only with the union of sets 1 and 5 may be due to the fact that the statistical diversity within set 6 is extraordinary high. So, the class boundaries can be hardly found to the overlapping material of sets 1 and 5. Here, higher level form features are demanded for recognition.

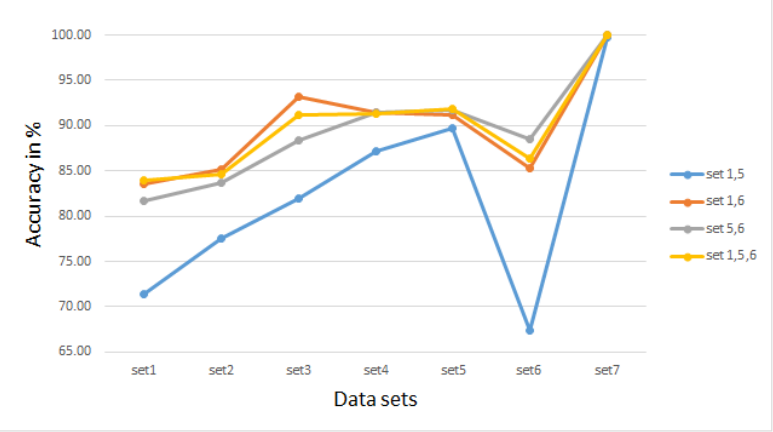

FIGURE 3.14: Effect of using multiple data sets as classifier

3.4.4 Does the network learn also scale invariant features?

In this test we used the data sets with 40,000 objects with three different scales (1, 2 and 2.5) and also with different illumination and view angles to train the system (see figure 3.9). After training, we used the features to classify the data sets which include multiple scales, illuminations and view angles, and we got the accuracy of 82.91% and 82.11% for set 8 and 9 respectively. These results were 82.20% and 81.35% for sets 10 and 11 respectively. For more details see table 3.3. The smaller accuracy reflects the fact that we use a static system which does not adapt to possible changes of input. There are a lot of systems which try to cope with this problem and we try to cope with this in the next chapter.

Set	RF size	RF share	Feature number	**Accuracy %**
set 8	9	3	16	82.91
set 9	9	3	16	82.11
set 10	9	3	16	82.20
set 11	9	3	16	81.35

TABLE 3.3: The accuracy rates for multi scale objects

3.4.5 Changing the size of the receptive field

Changing the size of the receptive field from 9×9 to 19×19 and the receptive field share to an overlap of 6 results in an accuracy of 79.77%, 78.40%, 82.50% and 87.84% respectively for sets 1, 2, 3 and 4. It means that, by increasing the size of receptive field, the rate of accuracy is reduced. By decreasing the RF size to 5, we see that the rate of accuracy is also reduced. The results with the receptive filed size of 7 and 9 are very close and for performance in running time and better accuracy on set 5 (which has more different structure from other sets), we took the 9 as our best size of receptive filed. In all of these tests, set 1 has been selected for feature training. For more details see table 3.4.

Set	RF size	RF share	**Accuracy %**	RF size	RF share	**Accuracy %**	RF size	RF share	**Accuracy %**	RF size	RF share	**Accuracy %**
set1	5	2	**80.86**	7	2	**85.92**	9	3	**85.11**	19	6	**79.77**
set2	5	2	**81.96**	7	2	**87.64**	9	3	**86.32**	19	6	**78.40**
set3	5	2	**91.57**	7	2	**94.31**	9	3	**92.18**	19	6	**82.50**
set4	5	2	**90.40**	7	2	**92.83**	9	3	**91.83**	19	6	**87.84**
set5	5	2	**88.30**	7	2	**90.90**	9	3	**91.58**	19	6	**88.59**

TABLE 3.4: Effect of changing the size of receptive fields

3.4.6 Changing the number of features (hidden units)

In this test all configuration and initialization was done as in section 3.4.1 except that we increased the number of features to 25 and decreased them from 16 to 9 and 7. It is clear that a small number of hidden units (features) generalizes better than a bigger number, but might be not precise enough. On the other hand, a bigger number of features might be precise in training, but might fail to generalize due to over-fitting the training data. The results of the experiments are shown in table 3.5. In all tests we used set 1 for training the features.

Set	Features	**Accuracy %**	Features	**Accuracy %**	Features	**Accuracy %**	Features	**Accuracy %**
set1	25	**86.42**	16	**86.76**	9	**85.11**	7	**85.15**
set2	25	**86.98**	16	**86.99**	9	**86.32**	7	**86.00**
set3	25	**93.40**	16	**93.66**	9	**92.18**	7	**92.56**
set4	25	**92.51**	16	**92.50**	9	**91.83**	7	**91.58**
set5	25	**90.92**	16	**90.85**	9	**91.58**	7	**90.54**

TABLE 3.5: The effect of changing feature numbers

By decreasing the features from 9 to 7, the rate of accuracy is also reduced a little bit (see table 3.5), whereas increasing the number of features takes much more computing time, but did not increase the performance significantly. Therefore, in our case 9 features present the best compromise between generalization error and over-fitting error.

3.4.7 Classification with random features

It is interesting to compare the network performance with a network of random features. In this case we do not learn any features but initialize the feature layer of network by random values.

Now, to test the network with random features we set the feature layer by random values of $[-0.5, +0.5]$ with uniform distribution and then tried to train the classification layer. The accuracy rate obtained for set 2 and set 4 was 58% and 63%, respectively. These amounts show that using random features can not provide good result for our tasks.

3.4.8 What kind of feature does the network learn ?

It is interesting to visualize the feature extractor filters that the network learned after training. In other words, we want to know if our filter or feature extractor looks like one of those filters found in literature, e.g. a Gabor filter, a differential of Gaussian (DOG) or some rotated bars. For this, we plot the weights of a receptive fields of some features as images in figure 3.15. They can be interpreted as filters. This features are trained in parallel, so we learned all features simultaneously.

The number of features was nine for these sets. The figure shows a set of quite complex filters. Here, we used the neighbors influence as specified in formula 3.37. The reason was that it is proved that interaction in mammalian nervous is more locally influenced. For instance, neural signal correlations between neighboring points in the neural fields is described in [65].

The resulting weights and therefore the features are the same in any run of the network weights adaptation starting with random initial weights, but the numberings are different, see figure 3.16.

This depends only on the starting conditions of the weights, and for this result they should be in $[-0.01, +0.01]$. By increasing the upper/lower boundary to around ± 0.02, the result will not be unique, and by decreasing the boundary to around

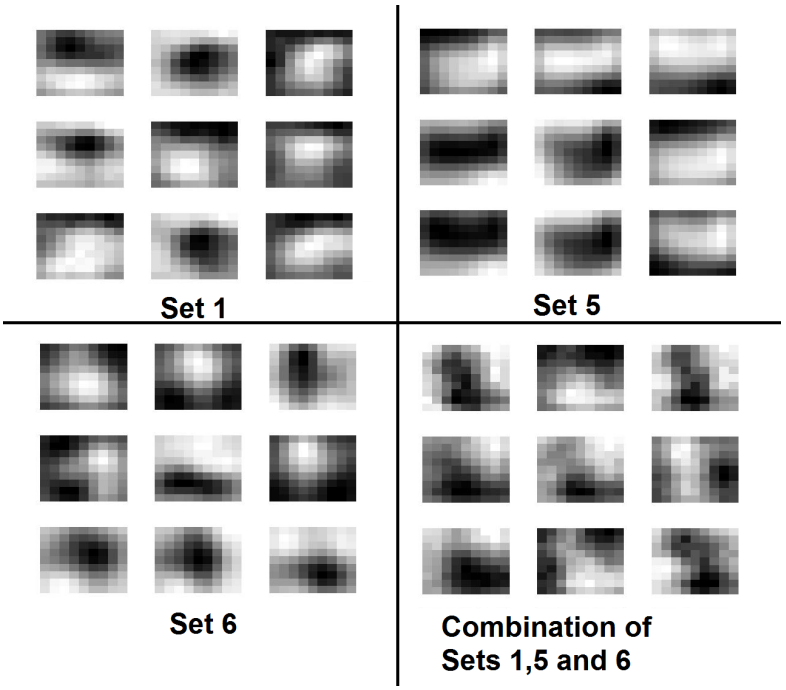

FIGURE 3.15: Receptive field weights learned by different training sets.

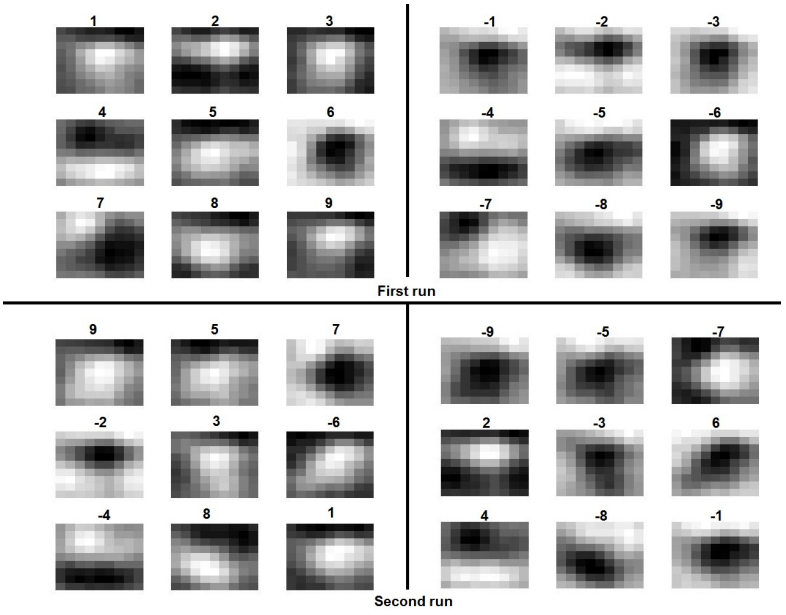

FIGURE 3.16: Receptive field weights resulting from different runs but the same training pattern set using random starts. "-" sign refers to the negative of the weights

$\pm 10^{-5}$ the result is unique, but most of them looks like each others (see figure 3.17). To understand that how the weights are similar together among different run of program, we drew the similarity curve of the weights in figure 3.18. We

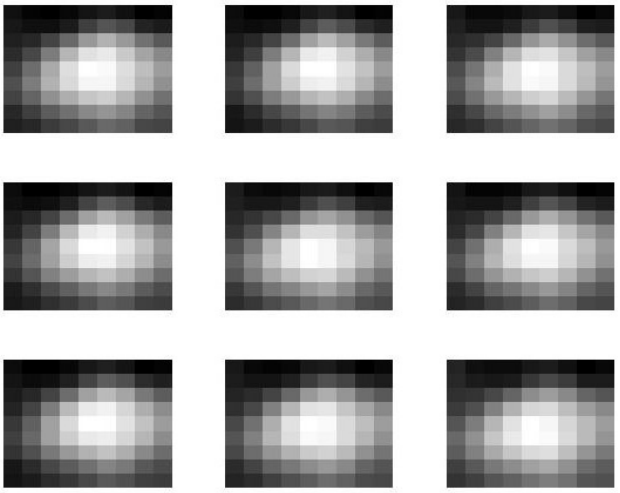

FIGURE 3.17: Receptive field weights resulting from very small random initialization weights (0.00001) .

computed the similarity between two weights **w1** and **w2** by this formula

$$Sim(\mathbf{w1}, \mathbf{w2}) = \left| \frac{\mathbf{w1}.\mathbf{w2}}{\|\mathbf{w1}\|\|\mathbf{w2}\|} \right| \tag{3.42}$$

It may be interesting to see the weights for the case where we drop the neighborhood influence. In figure 3.19 you can see the result when we drop the neighborhood influence. It is shown that in this case the weights are less smooth than before. Consider that by influencing the weights by their neighbors the performance does not change significantly (just around 1% in accuracy rate). This influence is biologically plausible and produces results which may be interpreted as filters.

3.4.9 The extracted features and Gabor filters

It is well known [66, 67] that the simple cell (V1) response in the visual cortex of mammalian brain can be modeled by a Gabor filter. Now, how do the extracted features compare to Gabor filters? For this comparison, we picked up some of the extracted features and measured the Euclidean distance among them and Gabor

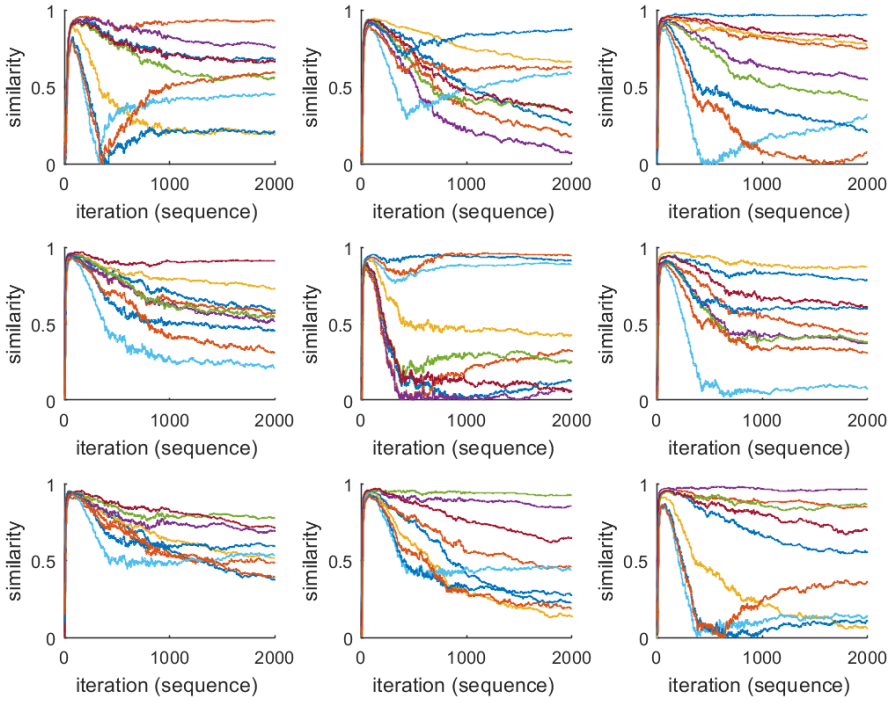

FIGURE 3.18: Similarity among weights

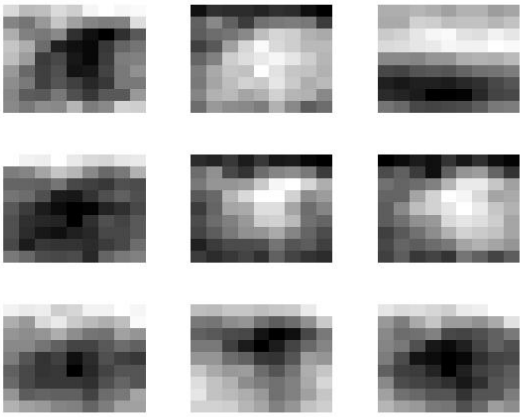

FIGURE 3.19: Receptive field weights when we drop the neighborhood influence.

filters with different scale and orientation. It means that for each extracted feature, we tried to find the best match Gabor filter which has the minimum distance with our selected feature as a pair of our features. In figure 3.20, the Euclidean distance among Gabor and our extracted filter is displayed between nine extracted features from set 1, set 2 and set 3 and the appropriate Gabor filter. In this figure, the

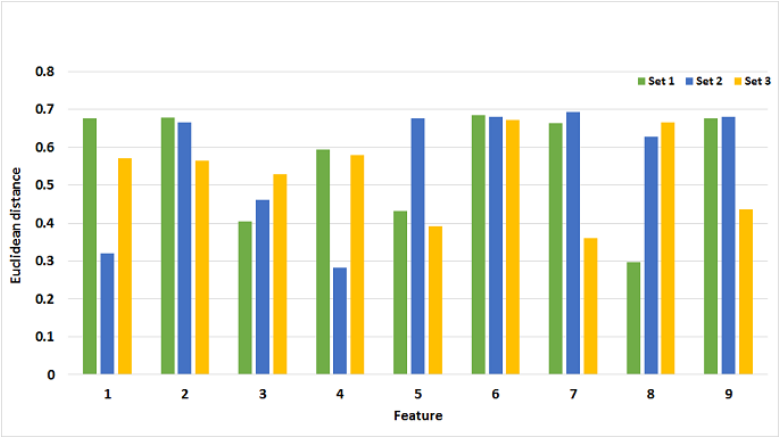

FIGURE 3.20: Euclidean distance among nine extracted features from set 1, set 2 and set 3 with Gabor filters. The distance values are displayed between zero (minimum distance) and one (maximum distance).

distance values are displayed between zero (minimum distance) and one (maximum distance). We see that the extracted feature are similar to Gabor filters with the average distance of 0.55. The reason that the extracted features are not very close to Gabor filters my be refer to some preprocessing algorithms like whitening. For instance in [68] the extracted features were more like to the Gabor filters after they applied Mahalanobis whitening on the training images.

3.4.10 Shallow network features vs. auto-encoder features

In this section, we compare the features of the proposed method with the features formed by an auto-encoder for reconstructing the input. Are the best features for classification also those who are the best for reconstruct-ing the input? To answer this question, we set the input layer and hidden units of an auto-encoder to be the same as our best configuration for classification. Then we set the input layer of an auto-encoder to 81 units and the hidden units to 9. The auto-encoder (AE) is a simple network that tries to reproduce at its output what is presented at the input. The basic AE is, in fact, a simple neural network with one hidden layer and one output layer, subject to the number of output neurons is equal to the number of inputs. In figure 3.21 the structure of auto-encoder is shown with one hidden layer.

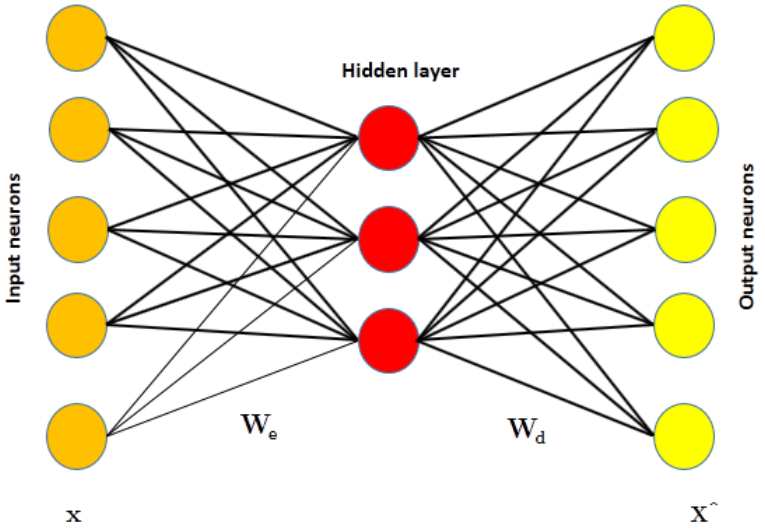

FIGURE 3.21: A typical Architecture of an auto encoder with one hidden layer.

The relation between input neurons ($\mathbf{x}$) and output neurons ($\hat{\mathbf{x}}$) in a typical auto-encoder is as:

$$h(\mathbf{x}) = s(\mathbf{W_e}x + b) \tag{3.43}$$

and

$$\mathbf{x} = s(\mathbf{W_d}h(\mathbf{x}) + c) \tag{3.44}$$

where s is a squashing function (e.g. $tanh$) and b and c are bias constants. To use the auto-encoder as feature extraction layer of the classifier, we have to train it with some unlabeled natural images. In this test we used the same data sets which used for our proposed shallow network but here we used them as unlabeled data and extracted some patches from them randomly as a train sets for auto-encoder. As in any neural network, here also we have to define an objective function which is to be optimized. In general, there are several possible choices for the objective function e.g. mean squared error, cross entropy and etc. To have a good comparison between the result of classification by auto-encoder and our shallow network, we used cross entropy as objective function because our objective function in the shallow network was based on cross-entropy. Then, the objective function is defined by:

$$R = -\sum_k x_k \ln(\hat{x_k}) + (1 - x_k) \ln(1 - \hat{x_k}) \tag{3.45}$$

The gradient descent method is employed to minimize this objective function. Table 3.6 illustrates the accuracy rate of classification using the auto-encoder . To have a good comparison, we used *tanh* for squashing function in hidden layer and sigmoid for the output layer which we did for our shallow network. Comparing the result of two tables, it is obviously clear that the average classification rate for two methods are very close to each others ($88.93 \pm 0.60\%$ for shallow network and $89.31 \pm 0.43\%$ for auto-encoder). The interpretation of this may be that the auto-encoder tries to learn the features which are best for reconstruction while our network tries to learn the features which are better for classification, which is evidently not the same.

Test Set \ **Training Set**	**Set 1**	**Set 2**	**Set 3**	**Set 4**
Set 1	85.39	85.19	83.27	84.83
Set 2	86.53	85.85	86.29	86.91
Set 3	92.89	93.45	93.31	94.38
Set 4	92.93	91.86	92.77	93.19

TABLE 3.6: Classification accuracies for various combinations of training and test sets using auto-encoder.

3.5 Summary

In this chapter we proposed a new method for universal feature extraction.

First, we used an information theory approach to design a proper risk function which leads to cross-entropy minimization. It is emphasized in some literatures that the cross-entropy risk function has significant, practical advantages over mean squared-error approaches [69, 70]. We developed a feed forward neural network as basic structure to extract universal features.

Second, to reduce the number of parameter to learn, as constraint we used a weight sharing method for all receptive fields. In addition of reducing the number of learning parameters it has the benefit that the shared weights makes all neurons detecting the same features, independent of their different positions in the input

image. As draw-back, it should be noted that this decision is not covered by the original proofs [23] for the approximation properties of two-layer neural networks. Additionally, the labeling of the filter properties of the first layer as "features" is plausible, but arbitrary.

Nevertheless, the results show that those *universal features* are unique and can be successfully applied in very different image processing applications e.g., hand written digit classification, recognition of natural or artificial objects which are placed in the natural or artificial background images and recognition of texture.

We used very different image sets for training and testing image features for classification purposes. Additionally, we changed several network parameters (e.g network layer, number of hidden unit and size of receptive field) to get the best results. By these tests, we can give some answers to our questions posed in section 3.1.

- What is the best size of a receptive field (patch)? The optimal RF size is 9×9.
- What is the optimum number of hidden units? The number of hidden units seems to be 16.

Although the universal features are a good start, for really recognizing natural objects in images additional questions have be studied:

- How can we make the system invariant to the position of objects in image so it could recognize objects not only in the center of background image, but also in any places of image?
- How can we make the system to adapt to different shadings and object sizes?
- How can the optimal size of the receptive fields be obtained automatically?
- The approach has shown the abilities of shallow networks - but what about deep networks [19, 71]? Is there a performance increasing possible?

In the next chapter we evaluate the capability of deep network with three layers for universal feature extraction approach.

Chapter 4

Deep Neural Networks for Universal Feature Extraction

4.1 Introduction

In this chapter, a second hidden layer will be added to the shallow neural network from the previous chapter. The chapter is divided into a short introduction into deep architectures, then we will look at the equations and updated learning rules for a three and four-layer neural network as well as the network specifications. Lastly the performance of the three and four layers network for different parameters will be tested.[1]

4.2 Motivation for using a deep neural network

Deep neural networks are compositions of many layers of non-linear units, in other words, they are cascades of parameterized non-linear modules that contain trainable parameters at every level. The outputs of the intermediate layers are akin to intermediate results on the way to computing the final output. While deep neural networks are well-known in the machine learning field since the 1980s, they were

[1] Thanks Mrs. Mackert for help in simulation of the result of this cahpter.

not popular among researchers until recently. The main reason for that was a highly cited publication by George Cybenko in 1989 [72]. Cybenko showed that one hidden layer of nodes is sufficient for any multivariate function approximation problem. While Cybenko [72] and Kolmogorov [73] have shown that convergence is possible, the resource requirements are never addressed. In other words, a large number of hidden layer nodes may be needed to solve any generalized functional mapping. Even with this property, shallow networks are popular, since many tasks are simple enough to be solved by neural networks with one hidden layer and deep networks are widely known as being hard to train. In the last decade deep neural networks have gained much more attention because of several breakthroughs in research such as *deep belief nets* [74] and *deep convolutional neural networks* [71]. There is one promising quality that deep neural networks offer, that shallow networks don"t have:

- If another hidden layer is added to the shallow network that was introduced in chapter two, the universality theorem is satisfied for the task of feature extraction. This should affect the search for universal features positively, because now the deep neural networks can represent a significantly larger set of functions than the shallow network.

4.3 Learning highly-varying functions

Expressing complex behaviors (such as universal feature extraction) requires highly varying mathematical functions, or high-level abstractions. Deep multi-layers neural networks have many levels of non-linearities allowing them to compactly represent highly non-linear and highly-varying functions. The reason why shallow networks are not able to represent these functions is that they have a large number of variations in the domain of interest e.g., they would require a large number of pieces to be well represented by a piecewise-linear approximation. Deep architectures allow the representation of a wide family of functions in a more compact form than shallow architectures, because they can trade space for time (or breadth

for depth). The learning of more abstract functions is much more efficient when it is done sequentially, by composing previously learned concepts [75]. With a second hidden layer the architecture will have one more layer of abstraction. Each layer in a multi-layers neural network can be seen as a representation of the input obtained through a learned transformation. More detailed, the first hidden layer will extract low-level features and the second hidden layer will build on this knowledge and extract more abstract, higher-level features. Finally the higher-level features will be classified in the last layer. This technique shares many similarities with the mammalian vision system as can be seen in figure 4.1. Here one can see that the model of multi-layers feed-forward architectures for computational visual processing bears a lot of resemblance to the organization of biological vision in the brain.

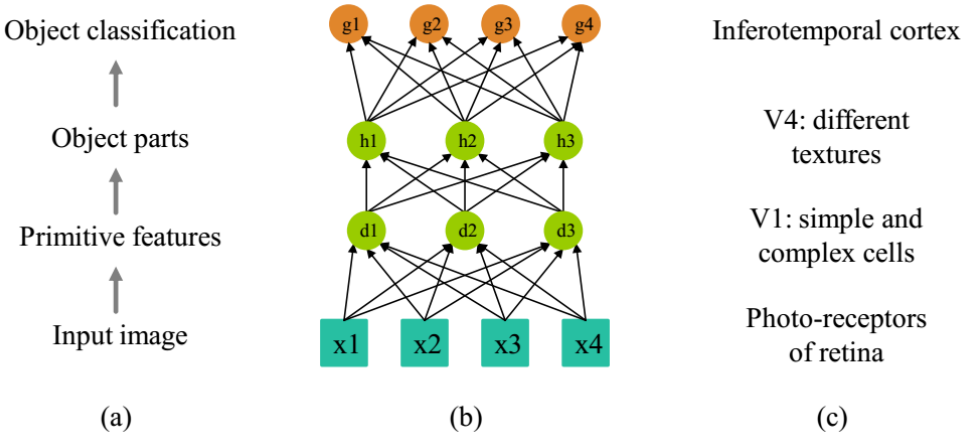

FIGURE 4.1: Hierarchical organization of visual scenes. In (a) a general hierarchical multi-layers model is described where each of the descriptions correspondents to the neighboring layer of the artificial feed-forward neural network in (b). In (c) the components of the biological vision system are briefly explained. First the visual stimuli are captured by the photo-receptors of the retina and then this information is relayed to primary visual cortex V1, where it is coded in terms of edge detection. Visual area V4 is tuned for simple geometric shapes. The last region in the hierarchy of visual processing in the inferotemporal cortex, its primary function is object recognition.

4.4 Training neural networks

Although deep neural networks with many layers can represent deep circuits, training deep networks has always been seen as somewhat of a challenge and has contributed greatly to their bad reputation. A notable exception to this is the convolutional neural network architecture that has a sparse connectivity from layer to layer. Training a convolutional network can be performed with stochastic (on-line) gradient descent, computing the gradients with a variant of the back-propagation method. While convolutional nets are deep (generally 5 to 7 layers of non-linear functions), they do not seem to suffer from the convergence problems that plague deep fully-connected neural networks. There is no definitive explanation for this, but it is suspected that this phenomenon is linked to the heavily constrained parameterization, as well as to the asymmetry of the architecture [75]. Another successful training algorithm is used in the deep belief net that can be seen as a composition of simple, unsupervised networks such as restricted Boltzmann machines. Experiments have been performed on the MNIST and other datasets to try to understand why the Deep Belief Networks are doing much better than either shallow networks or deep networks. The results are reported and discussed in [74]. A common explanation for the difficulty of deep network learning is the presence of local minima or plateaus in the loss function. Gradient-based optimization methods that start from random initial conditions appear to often get trapped in poor local minima or plateaus. The problem seems particularly dire for narrow networks (with few hidden units or with a bottleneck) and for networks with many symmetries (i.e., fully-connected networks in which hidden units are exchangeable) [75]. Another practical implication of using deep multi-layers architectures is the effect of the gradient progressively getting more dilute. It can happen that the correction signal that is propagated to the first few layers is getting smaller and smaller with each layer, therefore having little impact on the weights of the first layers. Therefore it is important to be aware of the potential issues that can occur when expanding a shallow network to a deep network.

4.5 Three layer neural network for extracting several features

In this section, we will expand the shallow network from previous chapter to a deep neural network with three layers. In figure 4.2 you can see the general design of a three-layer neural network for extracting several features from one receptive field that consists of xl pixels. Here the additional Layer ($\mathbf{e}$) is integrated between the first layer ($\mathbf{u}$) and the output layer ($\mathbf{w}$) of the network. ($\mathbf{e}$) represents an additional feature extraction step.

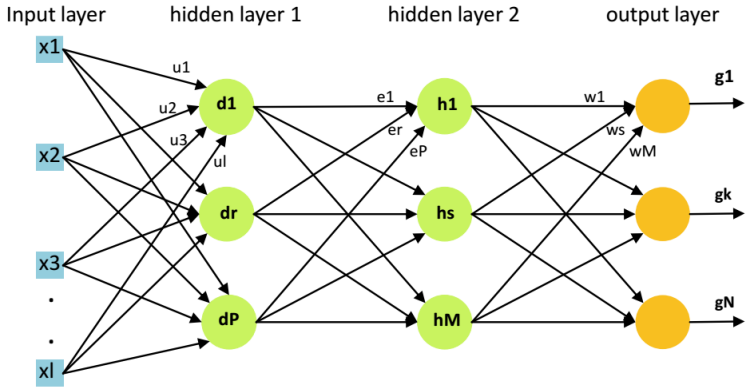

FIGURE 4.2: Deep neural network with 2 hidden layers, extracting and classifying information from one receptive field.

In figure 4.3 the entire deep neural network is shown. Here, you can see how the outer layer connects to the second hidden layer and how multiple receptive fields are extracted. The first task is obtaining the new learning equations to update the weights in each layer accordingly.

The extraction result $\mathbf{d}_j$ of the first layer, now consists of P features and is performed for each of the receptive fields j. We again chose $tanh$ as the squashing function for the hidden layers

$$\mathbf{d}_j = (d_1(\mathbf{u}_1, \mathbf{x}_j), d_r(\mathbf{u}_r, \mathbf{x}_j), \ldots, d_P(\mathbf{u}_P, \mathbf{x}_j))^T \tag{4.1}$$

with

$$d_{rj}(\mathbf{u}_r, \mathbf{x}_j) = S\left(o_{rj}\right) = tanh(o_{rj}), \quad o_{rj} = \mathbf{u}_{rj}^T \mathbf{x}_j \tag{4.2}$$

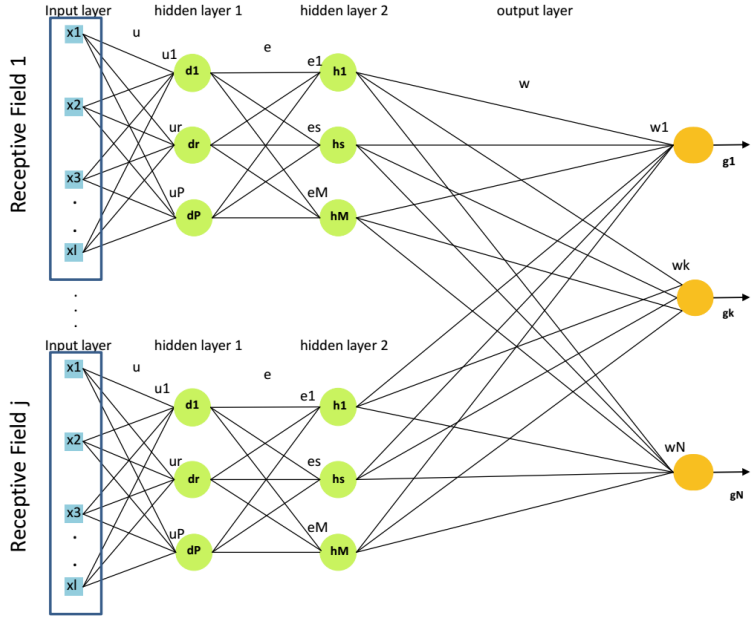

FIGURE 4.3: Deep neural network with 2 hidden layers, extracting and classifying information from one receptive field.

In the second hidden layer of the network, a different set of features is extracted with size M. This extraction result can be thought of as the extraction of important characteristics of the features from the first layer

$$h_j = \left(h_1\left(\mathbf{e}_1, \mathbf{d}_j\right),\ h_s\left(\mathbf{e}_s, \mathbf{d}_j\right), \ldots, h_M\left(\mathbf{e}_M, \mathbf{d}_j\right)\right)^T \tag{4.3}$$

with

$$h_{sj}(\mathbf{e}_s, \mathbf{d}_j) = S\left(z_{sj}\right) = tanh\left(z_{sj}\right), \quad z_{sj} = \mathbf{e}_{sj}^T \mathbf{d}_j \tag{4.4}$$

The number of hidden units in both hidden layers is equal in our application, therefore we have $P = M$. For the object detection function we chose the Fermi function, as we did before for the shallow network.

$$g\left(\mathbf{w}, \mathbf{h}\right) = \ S_F\left(v\right) \quad with \ \ S_F\left(v\right) = \frac{1}{1 + \ \exp(-v)} \ and \ v = \mathbf{w}^T \mathbf{h} \tag{4.5}$$

For the N output units the number of inputs becomes $M \times m$ since

$$\dim\left(\mathbf{w}\right) \ = \dim\left(\mathbf{h}\right) \ = Mm$$

Now we will define the weight update functions, starting with $\mathbf{w}$

$$\mathbf{w}\,(t+1) = \mathbf{w}\,(t) + \gamma(t)\frac{\partial R(\mathbf{u},\mathbf{e},\mathbf{w})}{\partial \mathbf{w}} \tag{4.6}$$

We can calculate the derivative of the stochastic gradient objective function, so the learning equation for w is

$$\mathbf{w}_k\,(t+1) = \mathbf{w}_k\,(t) - \gamma(t)\,(g_k - L_k)\,\mathbf{h} \tag{4.7}$$

With k being the $k-th$ output unit. $\mathbf{e}$ is then

$$\mathbf{e}\,(t+1) = \mathbf{e}\,(t) + \gamma(t)\frac{\partial R(\mathbf{u},\mathbf{e},\mathbf{w})}{\partial \mathbf{e}} \tag{4.8}$$

And again we calculate the derivative of the objective function. For the $s-th$ feature $\mathbf{e}_s$ is now

$$\mathbf{e}_s\,(t+1) = \mathbf{e}_s\,(t) - \gamma(t)(g-L)\frac{\partial v}{\partial \mathbf{e}_s} \tag{4.9}$$

With the activity of v

$$v = \mathbf{w}^T\mathbf{h} = \sum_{i=1}^{M}\sum_{j=1}^{m} w_{ij}h_{ij} \tag{4.10}$$

Containing the $s-th$ feature of the $j-th$ receptive field, we get the derivative of

$$\frac{\partial v}{\partial \mathbf{e}_s} = \frac{\partial}{\partial \mathbf{e}_s}\sum_{i=1}^{M}\sum_{j=1}^{m} w_{ij}h_{ij} = \sum_{i=1}^{M}\sum_{j=1}^{m} w_{ij}\frac{\partial}{\partial \mathbf{e}_s}h_{ij}(\mathbf{e}_{ij},\mathbf{d}_j) \tag{4.11}$$

Because the $s-th$ feature extraction function h_{ij} only depends on the $s-th$ parameter vector $\mathbf{e}_s$, the outer sum is redundant, the term is reduced for $i=s$ and the remaining derivative is calculated as:

$$\frac{\partial v}{\partial \mathbf{e}_s} = \sum_{i=1}^{M}\sum_{j=1}^{m} w_{ij}\frac{\partial}{\partial \mathbf{e}_s}h_{ij}(\mathbf{e}_{ij}, \mathbf{d}_j) = \sum_{j=1}^{m} w_{sj}\ S'\left(z_{sj}\right)\mathbf{d}_j \tag{4.12}$$

The learning equation for $\mathbf{e}$ is now

$$\mathbf{e}_s\left(t+1\right) = \mathbf{e}_s\left(t\right) - \gamma(t)(g-L)\sum_{j=1}^{m} w_{sj}\ S'\left(z_{sj}\right)\mathbf{d}_j \tag{4.13}$$

Finally, considering all N output units, $\mathbf{e}_s$ is

$$\mathbf{e}_s\left(t+1\right) = \mathbf{e}_s\left(t\right) - \gamma(t)\sum_{k=1}^{N}\left(g_k - L_k\right)\sum_{j=1}^{m} w_{ksj}S'\left(z_{sj}\right)\mathbf{d}_j \tag{4.14}$$

Where $S'\left(z_{sj}\right) = 1 - tanh^2(z_{sj})$

Next we will calculate the learning equation for $\mathbf{u}$

$$\mathbf{u}\left(t+1\right) = \mathbf{u}\left(t\right) + \gamma(t)\frac{\partial R(\mathbf{u},\mathbf{e},\mathbf{w})}{\partial \mathbf{u}} \tag{4.15}$$

The derivative of the objective function becomes

$$\mathbf{u}_r(t+1) = \mathbf{u}_r(t) - \gamma(t)(g-L)\frac{\partial v}{\partial \mathbf{u}_r} \tag{4.16}$$

With the activity v from equation (4.10) the derivative $\frac{\partial v}{\partial \mathbf{u}_r}$ is

$$\frac{\partial v}{\partial \mathbf{u}_r} = \sum_{q=1}^{P}\sum_{j=1}^{m}\sum_{i=1}^{M} w_{ij}\frac{\partial}{\partial \mathbf{u}_r}h_{ij}(\mathbf{e}_{ij}, d_{qj}) = \sum_{q=1}^{P}\sum_{j=1}^{m}\sum_{i=1}^{M} w_{ij}S'\left(z_{ij}\right)e_{r,i}\ \frac{\partial}{\partial \mathbf{u}_r}d_{qj}(\mathbf{u}_{qj}, \mathbf{x}_j) \tag{4.17}$$

Again the $r-th$ feature extraction function d_{qj} only depends on the $r-th$ parameter vector $\mathbf{u}_r$, the outer sum is redundant, the term is reduced for $q = r$ and the rest of the derivative is calculated as:

$$\frac{\partial v}{\partial \mathbf{u}_r} = \sum_{j=1}^{m}\sum_{i=1}^{M} w_{ij} S'(z_{ij})\, e_{ri}\ \frac{\partial}{\partial \mathbf{u}_r} d_{rj}(\mathbf{u}_{rj}, \mathbf{x}_j) = \sum_{j=1}^{m}\sum_{i=1}^{M} w_{ij} S'(z_{ij}) e_{ri}\ S^{'}(o_{r,j})\ \mathbf{x}_j \tag{4.18}$$

The learning equation for u is

$$\mathbf{u}_r(t+1) = \mathbf{u}_r(t) - \gamma(t)(g - L)\sum_{j=1}^{m}\sum_{i=1}^{M} w_{ij} S^{'}(z_{ij})\, e_{ri}\ S^{'}(o_{rj})\, \mathbf{x}_j \tag{4.19}$$

Finally, considering all N output units, the update rule for $\mathbf{u}_r$ is

$$\mathbf{u}_r(t+1) = \mathbf{u}_r(t) - \gamma(t)\sum_{k=1}^{N}(g_k - L_k)\sum_{j=1}^{m}\sum_{i=1}^{M} w_{kij}\ S^{'}(z_{ij})\, e_{ri}\ S^{'}(o_{rj})\, \mathbf{x}_j \tag{4.20}$$

Where $S^{'}(z_{ij}) = 1 - tanh^2(z_{ij})$ and $S^{'}(o_{rj}) = 1 - tanh^2(o_{rj})$.

At this point the neighboring effect of the weights is calculated, just like we did for the shallow network in previous chapter.

$$u_{mn}(t+1) = u_{mn}(t) + \aleph(m, n, r, s) u_{rs}(t+1) \tag{4.21}$$

where

$$\aleph(m, n, r, s) = e^{-((r-m)^2 + (s-n)^2)/2\sigma^2} \tag{4.22}$$

is the neighborhood function. Variable σ is related to the neighborhood radius.

4.6 Experimental results

As training and testing data, we used the same data sets and preprocessing methods which have been done for the shallow network in the previous section. Images from the Amsterdam library of object images are used [58]. Further processing

included placing the selected object in the middle of a natural (e.g. beach or a landscape) background image and shifting the objects position by maximum 3 pixels to the left, right, top or bottom of the image. The entire set of data holds 40,000 images with size 80×60 pixels. The data is labeled, with each image having a label, explaining what class the image belongs to. Here also there are 10 classes in total; therefore, the deep neural network has 10 output units. In this section, the performance of the deep neural network will be evaluated in terms of the correct classification of the extracted features. We rely on prior research that was presented in the second section for frame conditions concerning the deep neural network architecture and the choice of parameter values.

4.6.1 Important parameters for testing

The weight initialization: In a deep network, weight initialization has a big influence on the learning process since the feed-forward and back-propagation algorithms now traverse longer routes and have to perform more calculations. Various weight initialization methods will be reviewed for the deep neural network.

The number of features to be extracted: It certainly could be the case that universal features that are extracted with a deep neural networks exhibit a different composition than features that are extracted in a shallow network. Therefore, it is appropriate to experiment with the number of features.

Receptive field size and receptive field overlap: In the shallow network it was determined that the optimal receptive field size was 9×9 with 3 shared pixels. Universal feature extraction in a deep network might benefit from a different sized receptive field since there are now higher-level features involved. A significant benchmark for training and testing is the shallow network itself since it allows us to compare certain outputs like the classification accuracy and the extracted features of the first layer. It definitely is a notable help in determining how certain parameters affect the deep neural network and of course, what is most interesting:

Does the deep neural network perform the task of universal feature extraction better than the shallow neural network?

4.6.2 Measuring classification quality

For evaluating how a certain set of parameters affects the network, the most meaningful information is found in the accuracy of the classification. In this section, we used the same formula and methods to measure the quality of classification which has been used in the previous chapter for shallow network. Then the formula to calculate the classification accuracy is:

$$Accuracy = 0.5(Prob(PT) + Prob(NF)) \tag{4.23}$$

where $Prob(PT)$ is the Probability of a positive (true) neural network system classification decision or the percentage of true classifications that were correctly identified as such and $Prob(NF)$ the negative (false) classification decision or the percentage of negative classifications that are correctly identified as such. To compute $Prob(PT)$ and $Prob(NF)$ the classification output of the neural network units is used. The output that is calculated at each unit in the network is a probability. To assign an object to a class, the maximum value of the output is selected. This is in accordance with the Bayes classification rule that was introduced before. Thus, the object is the member of a class with the maximum output value:

$$choose\ C_i\ if\ g_i = max(g_j)\ j = 1..N \tag{4.24}$$

4.6.3 Testing specifications

The optimal parameters for the deep neural network are not yet known, so as initial parameters we pick the parameters that gave the best classification accuracy in the shallow network and work from there:

Optimal weight initialization method: Random uniform distribution in the interval $[-0.01, +0.01]$

Optimal feature number: 9

Optimal receptive field size and receptive field share: size 9×9, with share 3.

Unless explicitly stated, the tests will be performed with set 1 from the ALOI database. The accuracy values in this sections are the average classification accuracy (with maximum 1.5% variation) values for four completed training and test runs of the network for a respective parameter set. This number leads to a good approximation of the accuracy for the respective parameters and reflects the effect of the parameters sufficiently. Ideally, each accuracy value would be a mean over more than four completed runs, however in this regard, we are restricted by the available computational power. A bigger number would lead to a slightly more precise accuracy, however, this does not influence the process positively, therefore, it can be disregarded. The classification accuracy was calculated after training the network with 80% of the images from the respective image set and testing it with the remaining 20% of the images. To test how well the deep network executes the task of extracting and classifying universal features, the accuracy is calculated. A more concrete and descriptive evaluation is given by the learned weights. Generally, weights are tunable parameters that are adjusted according to the information that is gathered from the feed-forward algorithm and the risk function. Maybe an even more important property: weights themselves are adapted to the image data as part of the training process. After thousands of iterations, the feed-forward and back-propagation algorithms create a model of decision making that can perform the presented task to a certain degree. The functionality of this model is best represented through the weights. Weights can be understood as filters and after a sufficiently large number of training samples, the weights of each layer show a representation of the features that are learned in that layer. Consequently after determining the optimal parameters of the deep neural network, we will take a look at the extracted features.

4.6.4 Optimal weight initialization method

For the prior first execution of the deep neural network, the weight values were initialized with the random uniform distribution in the interval $[-0.01, +0.01]$. The shallow network performed best with this method of weight initialization. Now we want to test which method leads to a better performance in the deep neural network. First, we will review whether the suggested method from [26] results in a good accuracy. While it was not the best method for the shallow network, it may lead to a better performance in the deep network. We did not follow the calculation of the interval value from [26] which is as:

$$|w_{ij}| < 2.4/n_{input} \tag{4.25}$$

completely because we also included the upper bound as a test value, but as we will see this is a good idea. The interval values for first layer is then $[-0.03, +0.03]$ and for the second and third layer is $[-0.27, +0.27]$. Because the hidden and last layer has smaller units, the weight interval for them is bigger. The resulting accuracy for the deep neural network is 90.92%. This is already better than initializing the weights in the interval $[-0.01, +0.01]$. We also initialized the weights with smaller values, with similar proportion. The results can be viewed in table 4.1. Here we can see that the best classification accuracy is reached with the proposed upper bound from 4.25 with 90.84%. The interval $[-0.005, +0.005]$ for $\mathbf{u}$ is lower than the lower bound is given in equation 3.38 and as we can see it leads to a comparatively bad classification. The reverse approach of initializing the weights of the first hidden layer in a bigger interval than the rest of the network resulted in a worse classification for the deep neural network that reached about 85%. The classification gets gradually worse with smaller intervals. So far we dont know all the optimal parameters, therefore, we can only compare the calculated accuracies in table 4.1 to table 4.2 and 4.3.

In table 4.2 we start from the optimal weight interval value of the shallow network, increasing the weight interval for $\mathbf{e}$ to analyze the effect the additional hidden layer has on the classification accuracy. The best accuracy in this table is achieved

Interval u	**Interval e,w**	**Accuracy %**
[-0.03,+0.03]	[-0.27,+0.27]	**90.84**
[-0.025, +0.025]	[-0.23, +0.23]	89.71
[-0.02, +0.02]	[-0.19, +0.19]	89.31
[-0.015, +0.015]	[-0.15, +0.15]	88.93
[-0.01, +0.01]	[-0.11, +0.11]	88.88
[-0.005, +0.005]	[-0.07, +0.07]	87.54

TABLE 4.1: Accuracy of the classification for different weight initialization values, stepwise decreasing from the upper bound given in [26].

Interval e	**Accuracy %**	**Interval e**	**Accuracy %**
[-0.02, +0.02]	86.37	[-0.42, +0.42]	87.72
[-0.07, +0.07]	85.98	[-0.47, +0.47]	88.93
[-0.12, +0.12]	87.94	[-0.52, +0.52]	88.97
[-0.17, +0.17]	86.37	[-0.57, +0.57]	89.4
[-0.22, +0.22]	87.13	[-0.62, +0.62]	88.92
[-0.27, +0.27]	87.08	[-0.67, +0.67]	89.29
[-0.32, +0.32]	88.29	[-0.72, +0.72]	**89.42**
[-0.37, +0.37]	87.94	[-0.77, +0.77]	89.24

TABLE 4.2: Accuracy of the classification for different weight initialization intervals. Values of weights **u** and **w** are initialized in the interval $[-0.01, +0.01]$ while **e** is initialized with increasing interval values.

for the interval [-0.72,+0.72] with 89.42%. An interesting observation is that the accuracy is better if the weight interval of the second hidden layer is comparatively bigger than the interval of the first hidden layer and the last layer. Initializing the weights of the second hidden layer in intervals under [-0.32,+0.32] leads to a classification accuracy that is nearly as low as initializing the weights of all layers in the interval [-0.01,+0.01]. Ultimately a classification with the approach in table 4.2 never manages to classify as well as the upper bound method from 4.25. What is apparent in table 4.3 is that the deep neural network classifies more accurately if the weights are initialized in a bigger interval, although the accuracy is declining slowly above [-0.37,+0.37]. Some degree of the meaningful information that is needed for a good classification is lost in the feed-forward algorithm with small weights. In table 4.3 the most accurate classification is given when all weights are initialized in the interval [-0.25, +0.25] with 90.47%.

The conclusive findings are that the best classification accuracy can be achieved

with the weights of the deep network initialized in the intervals specified by equation 4.25. As previously mentioned, we are not following the method completely because we actually use the upper bound. The clear advantage to this method is that it considers the number of inputs in each layer. Now we could also try to find another similar method, however, it would not be as thoroughly tested as the proposed method from [26], therefore, it is reasonable to continue with intervals specified by equation 4.25.

Interval	**Accuracy %**	**Interval**	**Accuracy %**
[-0.01, +0.01]	85.77	[-0.29, +0.29]	89.14
[-0.05, +0.05]	87.85	[-0.33, +0.33]	90.18
[-0.09, +0.09]	88.32	[-0.37, +0.37]	89.97
[-0.13, +0.13]	89.25	[-0.41, +0.41]	89.73
[-0.17, +0.17]	89.18	[-0.45, +0.45]	87.73
[-0.21, +0.21]	89.69	[-0.49, +0.49]	87.99
[-0.25, +0.25]	**90.47**	[-0.53, +0.53]	87.76

TABLE 4.3: Accuracy of the classification for different weight initialization intervals. All weights are initialized in the same interval.

4.6.5 Optimal number of features (hidden units)

The aim of this Section is to determine what the optimal number of features is in the deep neural network. Up until now we used 9 features since this was the optimal number for the shallow network. It is important to note that the weight initialization intervals of the second hidden layer and the last layer are now dependent on the number of inputs the neurons of the respective layers receive, in this case, that is the feature number. So the weights of the first hidden layer will still be initialized in [-0.03,+0.03], while the weights of the remaining layers will be initialized in the appropriate interval. Generally, the number of hidden units influences the network in the following way: A small number of hidden units may generalize better but might cause under-fitting of the data. Too many hidden units may cause the network to tend to over-fit the training data, thus reducing generalization accuracy. Additionally, a big number of hidden units might make

the training unnecessarily slow. The factors have to be taken into account besides the classification accuracy when choosing the number of features.

Features	**Accuracy %**	**Features**	**Accuracy %**
5	85.41	17	89.69
7	87.48	19	87.25
9	**90.39**	21	88.86
11	89.41	23	88.28
13	88.14	25	89.11
15	89.30	27	87.09

TABLE 4.4: The classification accuracy for different numbers of features, trained and tested with set 1.

In table 4.4 you can see the accuracy of the classification for different numbers of features. For a feature number smaller than 9 the classification accuracy is accordingly worse. For set 1, 9 is the optimal number of features, however, feature numbers 11, 15 and 17 lead to a similarly good classification. The classification accuracy is decreasing for 25 and more features. If we compare the results from table 4.4 with the results of shallow network for set 1 from previous chapter we can see that the classification accuracy of the deep neural network is about 1.6% worse for 9 features. However for comparably smaller and bigger feature numbers it performs better than the shallow network. For 7 features the classification accuracy is 0.60% better and even 4.63% better for 25 features. That said the accuracy is still highest for 9 features.

Now we will expand the test data and review how well the deep network classifies images that were not part of the training process. Different feature numbers were tested for image sets 2, 3 and 4 from the ALOI database. It is interesting that for some image sets, the classification is significantly higher if the network was trained with images from a different image set. For example for image sets 3 and 4, similarly to the results of the shallow network, the features that were extracted with set 1 are apparently general enough to lead to a good classification of objects from sets 3 and 4. For set 2 the accuracy is comparable worse throughout the test results. Set 2 consists of images of an artificial object (e.g. toys or cups) in front of natural backgrounds. Set 1 consists of images of natural objects in front of

natural backgrounds. Features that are learned with natural objects do not reflect the characteristics of artificial objects well enough and therefore, the classification is worse.

Set	**Features**	**Accuracy %**	**Set**	**Features**	**Accuracy %**
2	7	87.57	2	9	86.91
3	7	94.87	3	9	95.71
4	7	93.08	4	9	93.04

TABLE 4.5: The classification accuracy for 7 and 9 features, trained with set 1 and tested with image sets 2, 3 and 4 from the ALOI database.

Set 3 consists of natural objects in front of artificial backgrounds and leads to the best classification overall with an average classification accuracy of 94.59% for the 4 tested feature numbers. The comparatively high accuracy for set 3 implies that objects are easier to classify if the properties of the background image differ from the object properties. However, that only applies if the network was also trained with the same objects, because if its not the accuracy decreases, as we have seen in the results of set 2.

Set	**Features**	**Accuracy %**	**Set**	**Features**	**Accuracy %**
2	16	88.85	2	25	89.63
3	16	93.69	3	25	94.1
4	16	92.55	4	25	93.23

TABLE 4.6: The accuracy for 16 and 25 features, trained with set 1 and tested with image sets 2,3 and 4 from the ALOI database.

Objects from set 4 are classified correctly in the deep neural network with an best accuracy of 93.08% and 93.04% which is taken by 7 and 9 features respectively. In set 4, images are artificial objects in front of artificial backgrounds. While the accuracy is not as good as for set 3 it still proves that the deep neural network is capable of extracting universal features and successfully applying them in the classification of objects that have a different composition.

4.6.6 Optimal size of the receptive field and the receptive share

In this section, we want to determine if size 9×9 of the receptive field still gives better results than other receptive field sizes. The size of the receptive field equates to the number of input units in the network. As a consequence weight, **u** will be initialized in the appropriate interval that is specified by the upper bound in equation 4.25. The remaining weights **e** and **w** will be initialized in the interval [-0.27,+0.27] for all the tests in this section. The appropriate size of the receptive field is strongly dependent on the size of the first hidden layer and the size of the image itself. By increasing the receptive field size each image patch holds more information and the image is covered by fewer image patches. Bigger receptive fields will yield a smaller network and decrease its complexity, which can also be a disadvantage in learning meaningful features. On the other hand, smaller receptive fields would encourage the development of more precise features, however, that would also expand the network size and might over-fit the training data. A good receptive field size should result in an image patch that is sufficiently characteristic of the observed image to allow a successful feature extraction. In table 4.7 you can see the accuracy of the classification for different receptive field sizes, tested on set 1 of the ALOI images. The size of the receptive share was chosen as $\lfloor (RF\_Size)/3 \rfloor$. Its interesting to see that the receptive field size has more direct influence on the classification accuracy than for example the number of features.

RF size	RF share	Accuracy %	RF size	RF share	Accuracy %
5	1	86.76	15	5	87.01
7	2	89.17	17	5	84.66
9	3	90.94	19	6	83.18
11	3	86.32	22	7	82.15
13	4	86.28	25	8	84.75

TABLE 4.7: Accuracy of a classification of 9 features and different receptive field sizes.

The worst classification accuracy is obtained for a receptive field size of 22×22 pixels with 82.15%. Receptive Field size 9 is still the best choice for set 1 with 7

being the second best while the remaining test sizes lead to a significantly worse classification. For the deep neural network, a smaller receptive field size leads to a better classification than in the shallow network. In fact, receptive field size 7×7 has a classification accuracy increase of 2.69% in the deep neural network compared to the shallow network. It seems that the extraction of higher-level features benefits from lower-level features that are extracted from a smaller receptive field. In Tables 4.8 and 4.9 we also looked at the classification accuracy of different ALOI image sets.

Set	**RF size**	**RF share**	**Accuracy %**	**Set**	**RF size**	**RF share**	**Accuracy %**
2	7	2	88.42	2	9	3	87.72
3	7	2	94.96	3	9	3	94.43
4	7	2	93.97	4	9	3	93.06

TABLE 4.8: Classification accuracies for receptive field sizes 7×7 and 9×9. The deep network was trained with set 1 and tested with sets 2, 3 and 4.

Set	**RF size**	**RF share**	**Accuracy %**
2	19	6	77.43
3	19	6	82.06
4	19	6	86.69

TABLE 4.9: Classification accuracies for receptive field size 19×19. The deep network was trained with set 1 and tested with the respective sets.

For set 4, the best classification accuracy is obtained for a receptive field size of 7×7. In the shallow network set 4 achieved an accuracy of 94.68% on average, thats significantly better than the classification accuracy of the deep network with 91.06%. So far set 4 is the test set that exhibits the most stable classification accuracy in all parameter tests. Receptive field size 9x9 lead to a particularly good classification when the network was trained and tested with set 1. However receptive field size 7×7 lead to a better classification accuracy when we used image sets 2,3 and 4 as the test sets and set 1 as the training set. Generally, the deep network performed better with an input layer with size 7×7 as can be seen in tables 4.8 and 4.9 . With smaller receptive fields the extracted features in the first hidden layer are more precise. More expressive information is captured in the features, therefore improving the classification.

Now, we can answer the questions from section 4.6.1 for the deep neural network:

- The optimal number of features is 9, just like in the shallow network.
- The optimal size of the receptive field and the receptive share is 7×7 and 2 respectively.
- The optimal weight initializing interval for the first layer ($\mathbf{u}$) is $[-0.03, +0.03]$ and for the second ($\mathbf{e}$) and third ($\mathbf{w}$) layers are $[-0.27, +0.27]$.

Consider that the optimal weight initialization interval for the deep neural network depends on the number of input units of the respective network layer. In section 4.6.3 the significance of the weights was exemplified and a first comparison was conducted in section 4.6.4.

4.6.7 Extracted features from the deep network

In this section we will take a closer look at the features that are extracted within the architecture using the optimum parameters obtained experimentally in previous sections.

In figure 4.5 you can see the features extracted with the deep neural network. They look quite unique and also exhibit more variation than the features extracted with the shallow network, presented in figure 4.4, although they are smaller in size.

4.7 Comparing the deep network to the shallow network

In this section, we compare the shallow and deep network in the terms of transferring features or universality. In tables 4.11 and 4.10 the average classification of different data sets using the features which are originally extracted to classify other sets for the shallow and deep network is shown. By comparing the results of

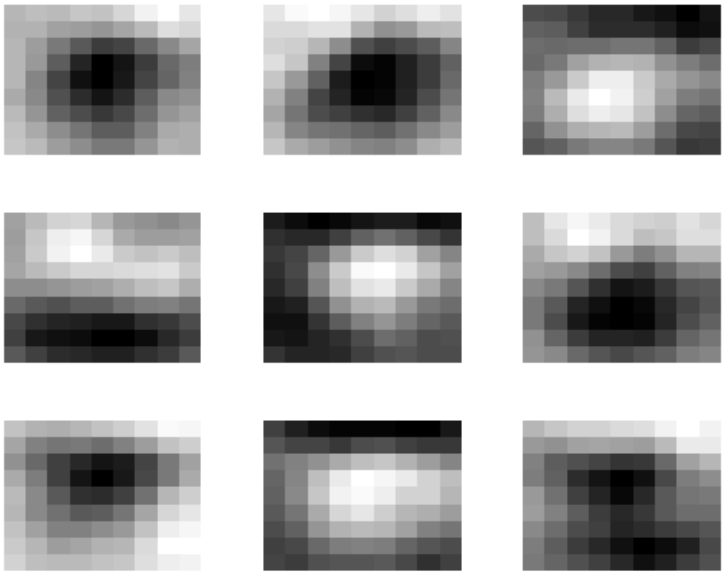

FIGURE 4.4: Weight plots (extracted features) of the first layer from the shallow network using set 1 as training set.

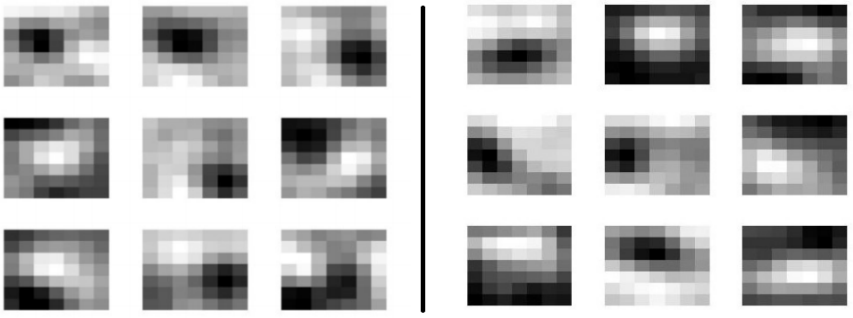

FIGURE 4.5: Weights of the first hidden layer (features)of deep network. The features on the left were trained with set 1 and the features on the right were trained with set 2.

these two tables, it is clarified that the deep network has better accuracy when the train set and test set are the same or close to each other. For instance the values of the diagonal of the table for shallow network is less than the related values for deep network. The average classification rate for shallow network is 88.92% while the average of the classification accuracy for deep network is 91.36%.

Another interesting result is that the shallow network has better performance than the deep network when we compare the accuracy rate of classification for set 5 (MNIST hand-written digits) as a test set. The average value of accuracy for shallow network is 91.72% while this value for deep network is 88.94%.

The interpretation of these results could be that in the deep network the features become more specific from lower layers to higher layers. On the other hand, the features computed by the last layer of a trained network must depend greatly on the chosen dataset and task.

Test Set \ Training Set	Set 1	Set 2	Set 3	Set 4
Set 1	85.11	82.28	85.49	84.57
Set 2	86.32	85.37	87.02	87.30
Set 3	92.18	90.13	94.47	93.66
Set 4	91.83	91.97	92.66	92.48
Set 5	91.58	90.97	91.91	92.10

TABLE 4.10: Shallow network classification accuracies for various combinations of training and test sets.

Test Set \ Training Set	Set 1	Set 2	Set 3	Set 4
Set 1	89.80	85.64	87.1	85.85
Set 2	90.09	89.32	86.97	89.55
Set 3	96.76	95.12	95.97	94.52
Set 4	93.88	93.77	92.75	94.78

TABLE 4.11: Multi-layers network with three layers classification accuracies for various combinations of training and test sets.

Test Set \ Training Set	Set 1	Set 2	Set 3	Set 4
Set 1	97.20	96.10	96.40	96.20
Set 2	97.80	98.20	97.40	98.20
Set 3	97.10	96.20	97.10	96.40
Set 4	98.80	98.90	95.70	98.90
Set 5	93.10	93.10	92.60	93.10

TABLE 4.12: Multi-layers network with four layers classification accuracies for various combinations of training and test sets.

4.8 Summary

In this chapter, we added a second hidden to the shallow neural network from the previous chapter. After finding the optimum parameters for the network like

we did for shallow network, we test the performance of the three and four layers network. By comparing the results of shallow network with multi-layers network, it is clarified that the multi-layer has better accuracy when the train set and test set are the same or close to each other. Surprising result is that the shallow network has better performance than the multi-layer network when we compare both accuracy rates of classification for set 5 (MNIST hand-written digits) as test set. It could be interpreted that the in the multi-layers network the features become more specific from lower layers to higher layers. On the other hand, the features computed by the last layer of a trained network must depend greatly on the chosen dataset and task.

Chapter 5

Object Learning and Recognition by Neural Mappings

One of the main problem for object recognition in computer vision is the transformation of an object (e.g. by shift, rotate, scale or depth deformation) within several views of that object. Object recognition in human vision might be explained by special temporary mappings where an unknown object is compared to stored object prototypes of standardized size and view[1]. How does this mapping work? One solution to this problem is to use dynamic links for a mapping between the input object and the target object. There already exist several approaches for that. In this chapter, we introduce a new dynamical self-organized mapping which can be used for dynamical object recognition and classification. Firstly, we explain two well-known dynamical object recognition system (Kohonen and Häussler systems), and in the rest of the chapter we describe our proposed system which is based on information theory.

[1]The result of this chapter published as: Self-organized neighborhood preserving projections using information, *The* 28^{th} *annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2016).*

5.1 Introduction

In vertebrates, especially in the human brain, the neural connections are not genetically fixed from birth, like in insects, but are developing during their lifetime. This is well known, but the question is still unresolved: how? There are many models for the self-organization of neural connections. For instance, the mapping of 3D sensory input distribution of the body to the human brain, which extends in a 2D way, is observed in the *primary somatosensory cortex*, as it receives the bulk of the thalamocortical projections from the sensory input fields. Typically, sensory neighbor regions are mapped to neighbor regions within the cortex. So, those mappings are termed *neighborhood-preserving mapping* which should not be confused with topology-preserving mappings. The latter guarantees the neighborhood while the former only approximates it. Mathematically, it is impossible to construct a topologically correct mapping between a 3D space and a 2D space, but it can be done by approximations, giving rise to a mapping which has discontinuities where the neighborhood abruptly changes. Such mappings are observed generally in the brain of many animals. For instance, in the bat brain, the echo signals are mapped to a 3D space representations [76]

There is a vast amount of literature on models for this phenomena. One of the most popular ones is the Kohonen map [77] which has many variations, e.g. the supervised or unsupervised Kohonen map, the neural gas and other approaches [14]. In all those approaches the input dimension is given by the number of inputs. The n-dimensional input space is spanned by all possible values of those n sensory input lines. In contrast to this, the mapping problem can be formulated as a mapping of sensory inputs, arranged in a 2D or 3D way, by neural projections. This means, for a 2D input having n lines and m rows of inputs sensors, we have $k = m \times n$ input values forming a 2D pattern, not k input dimensions. For each input neuron, the projection provides a way how it projects to an output neuron. In figure 5.1 both approaches are illustrated. On the left-hand side, all inputs from one variable x_1 are directed to all neurons, arranged in a 2D-way. The input from other variables is similar and not drawn. Thus, by 6 input sensors we have a 6D to

2D projection. On the right-hand side, also, the output neurons are arranged in a 2-dimensional way. Nevertheless, the input neurons are arranged as a 2D plane and present a 2D pattern which is mapped to the 2D output.

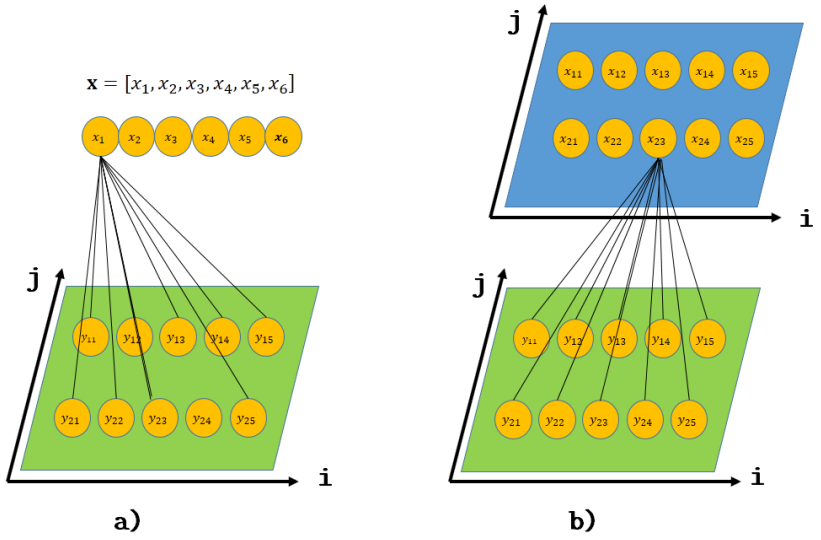

FIGURE 5.1: a) 6D to 2D SOM mapping. Here, **X** is a 6 dimensional vector. b) 2D to 2D neural projections.

In this chapter, we propose a new neighborhood preserving self-organizing map (SOM) algorithm, based on information theory. We present a new similarity measure based on the distance of local pattern entropies. Experimental results illustrate the robustness and efficiency of our new algorithm.

Let us now regard the neural projection approach more closely. One of the early work on this subject was done by Willshaw and Malsburg [78]. They formed a model where neighbored neurons of the output layer are enforcing activity while more distant neurons are inhibiting it. If the activity surpasses a threshold, also the synapses (weights) are adapted. Additionally, a continuously working fiber growing mechanism tries to test new connections. If the synapses are enforced by activity, they will remain, otherwise deleted. The local mechanism is completed by a global one which makes the local activities globally consistent. All feedback is within the output layer.

An important idea was also contributed by Häussler and Malsburg [22]. They tried to model the retina-tectum connections by neighboring retinal cell projections

through their fibers onto neighboring cells of the tectum. For this, they developed a set of differential equations. The fiber connections are two-fold: short-time connections are used to determine if an object is already stored, while the long-time connections are learned for storing the patterns. The idea of neighboring short-time connections and long-time projections was termed *Dynamic Link Architecture* [79, 80]. The idea of a *dynamic link* architecture evolved also by the work of Lücke et al. [81]. Their work interprets the visual columns as macrocolumns which have a state determined by a system of differential equations. Another early work was done by Kohonen [82]. In the rest of this chapter firstly, we explain two well-known self-organizing systems which proposed by Kohonen and Häussler [22, 82] and then, will describe our approach for a self-organizing system.

5.2 The Kohonen self-organizing neural projection

In this section, we explain the architecture of the Kohonen self-organizing neural projection which is proposed in [82].

Like many self-organizing systems, the goal was to get a mapping between an input pattern and an output pattern, starting from the full-connected situation on the left side of the figure 5.2 and developing to the finally ordered state on the right side after some iterations.

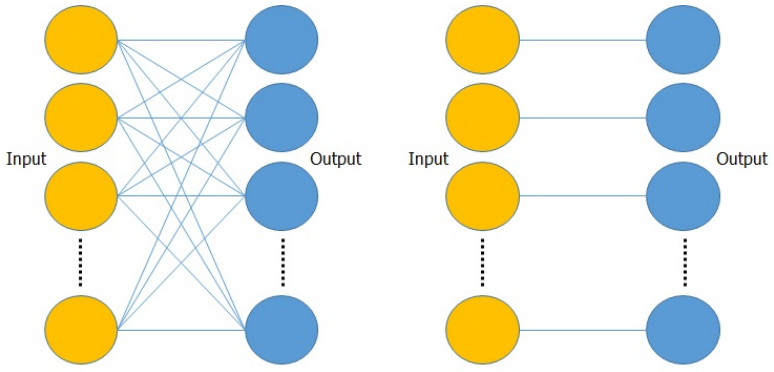

FIGURE 5.2: Illustration of the neural connectivity before (left) and after (right) self-organizing

To avoid the complexity of human brain structure, the transfer function of the network is assumed as linear:

$$y_i = \sum_{j=1}^{N_1} w_{ij} x_j \tag{5.1}$$

where:

- x_j is the input neuron activity.
- w_{ij} is the variable strength of the synapse between input neuron j and output neuron i.
- N_1 is the number of neurons in the input layer.

To make an adaptation equation, Kohonen tried to use the modified Hebbian law as a basic formula. In the Hebb rule, a synapse between two neurons is strengthened when the activity of the neurons in input and output side of it are highly correlated.

In the Kohonen formula, the strength of a synapse depends on the activity of input neuron on the input side but on the output side, it also depends on the activity of its neighbors proportional to the distance of these neurons to it. Another important change is that the strength of a synapse also depends on the original value of it. Then, the adaptation equation is as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)[1 + \beta w_{ij}(t)]u_i(t)x_j(t) \tag{5.2}$$

where:

- $\alpha(t)$ is a time-variable learning rate coefficient which starts from a small value between zero and one and decrease after each step of running. One common function for $\alpha(t)$ can be $\alpha(t) = 1/(1 + \gamma t)$ with e.g. $\gamma = 0.01$ or 0.0001.

- β is a constant value that describes the synapse-dependent emphasis in learning.

- $u_i(t)$ is the modified Hebbian law which not only depends on the activity of output neuron y_i, but is also affected by the activity of its neighbors, y_h, according to the distance from them. Then this effect decreases when the distance increases. This function can be a Gaussian modulation-type function which is used in this work as:

$$u_i = e^{az_i} \tag{5.3}$$

where a is constant control parameter and z_i defines a linear superposition of the activity of the neighbors of output neuron y_{ih} and interaction kernel g_{ih} as

$$z_i = \sum_{h=1}^{N_2} g_{ih} y_h \tag{5.4}$$

Here, N_2 is the number of output neurons.

The kernel function g_{ih} is defined to model its effect from location h to location i for an one-dimensional layer in the following way:

$$g_{ih} = \frac{1}{1 + |i - h|/c} \tag{5.5}$$

where c is a constant. Therefor, g_{ih} is a number between zero and one and decreases when the distance between two neuron i and h increases.

Let $\mathbf{w}_j$ be the column j of the connectivity matrix $\mathbf{W}$. Then the new updated values for the weights w_{ij} are normalized using Euclidean norm as:

$$w_{ij}(t+1) = \frac{w_{ij}(t)}{||\mathbf{w}_j||} \tag{5.6}$$

5.2.1 Simulations with one-dimensional layers

In this section, in order to compare the Kohonen approach with our new system, we should first characterize the Kohonen self-organized mapping by using some experiments.

We did the one-dimensional mapping with 20 input neurons and 20 output neurons. All parameters and input data patterns are selected as done by Kohonen in [82]. Then, to prepare the input data for training, in order to mimic complex input patterns that may consist of several independent parts, we used random mixtures of Gaussian patterns. Here, we used the biological motivation that the output activity of a neuron, controlled by the number of synapses, is centered on the neuron and decreases by the distance to the neuron. Each Gaussian was shifted randomly in the input layer. The mathematical forms of the Gaussians are simple and exactly definable, whereas their random mixture has a complex statistical distribution. Let the input pattern consist of K Gaussian components, centered at random in the input layer:

$$x_i = \sum_{k=1}^{K} e^{-(i-d_k)^2/2\sigma^2} \tag{5.7}$$

Where the d_k are selected randomly from the range $[1, N_1]$ with a uniform distribution (i.e. from the whole input layer), and σ is a variance of the normal function that may or may not be time-variable. In the first place, we wanted to demonstrate the point-wise organization of the projection when starting with randomly interconnected layers. The initial values of the w_{ij} were selected at random from the range $[0, 1]$ with uniform probability, after which, for each i, the sets of the input weights for j were normalized using the Euclidean norm.

As in the original work of Kohonen [82], in order to obtain good results, we have to normalize the kernel function. At each training step, we have to divide each element of it by Euclidean norm of the vector like we did in equation 5.6. Thus the function values u_i become:

$$u_i(t+1) = \frac{u_i(t)}{||\mathbf{u}||} \tag{5.8}$$

The final connectivity matrix with 20 input neurons and 20 output neurons is illustrated in figure 5.3, where row i corresponds to the $i-th$ output neuron, and column j to its $j-th$ synapse, or the $j-th$ input neuron, respectively. We got this result by setting $\sigma = 5$, $\beta = 1000$, $c = 5$ and time-dependent variable $\alpha(t) = 0.2/(1+0.001t)$ from equation 5.1. The parameter a in the equation 5.3 is set to 4. The ideal output in one-dimensional approach should be a diagonal matrix like as it is shown in the right side of this figure. As you can see in the left

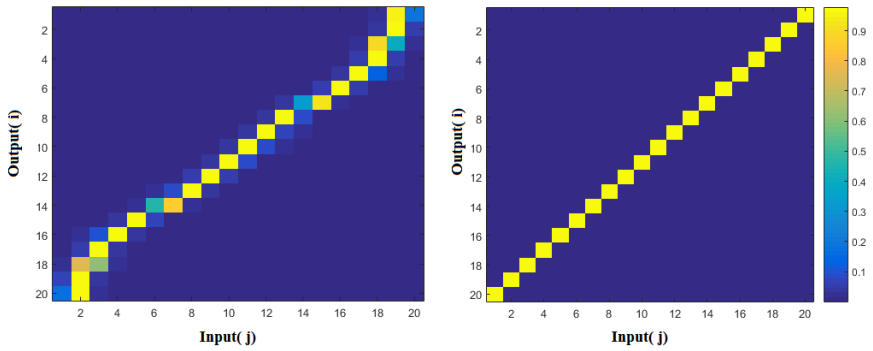

FIGURE 5.3: The connectivity matrix of Kohonen SOM after 25000 iterations (left) and the ideal connectivity (right).

side of figure 5.3, the desired connectivity is not really as same as the ideal output which is shown in the right side of this figure specially in the ridges.

5.3 The Häussler self-organizing system

In this section, we demonstrate another self-organizing system which is invented by Häussler and Von der Malsburg [22]. This system is known as *Häussler system*. In this system, the development of the synaptic connection between two neurons (e.g retina and tectum[2]in mammalian brain) from a random initial state toward a topological projection (e.g one to one) is described. The differential equations are based on two concepts: *cooperation* and *competition* among input and output neurons which are defined in [22] as :

- **Cooperation:** *The synaptic contacts on neighboring tectal cells which are in the same retinal region help each other to be strengthened* (see part (a) of figure 5.4).

- **Competition:** *The contacts made by one fiber compete with each other* (see parts (b and c) of figure 5.4).

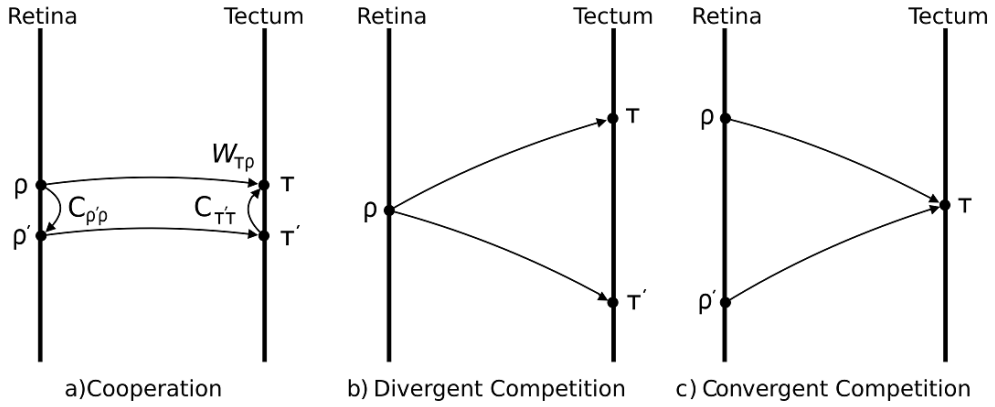

FIGURE 5.4: Cooperative (a) and competitive (b and c) processes between source (retina) and target(tectum) regions. (b) illustrates the divergent competition and (c) displays the convergent competition. (figure is taken from [21])

The change of the synaptic contact between input neuron (e.g retina) r and output neuron (e.g tectum) , w_{rt}, is a differential equation as follows :

$$\dot{w}_{rt} = f_{rt}(\mathbf{W}) - w_{rt}B_{rt}(f(\mathbf{W})) \tag{5.9}$$

where $f_{rt}(\mathbf{W})$ in this formula is the *cooperation* term described by the following equation,

$$f_{rt}(\mathbf{W}) = \alpha + \eta + w_{rt}\sum_{r't'} C\left(r, r', t, t'\right) w_{r't'} \tag{5.10}$$

where :

- α is an unspecific synaptic growth rate.

[2]"In [83], the general function of the tectal system is defined as: "to direct behavioral responses toward specific points in egocentric (*body-centered*) space".

- η is a small noise.
- $\mathbf{W}$ is a neural connectivity matrix.
- $C\left(r, r', t, t'\right)$ is a separable coupling function, modeled as a product of Gaussians which models the effect of neighbors on the activity of each other according to the distance among them as :

$$C\left(r, r', t, t'\right) = \exp \frac{(r - r')^2}{2\sigma^2} \exp \frac{(t - t')^2}{2\sigma^2} \tag{5.11}$$

In equation 5.9 there is another term, $B_{rt}(f(W))$, which describes the *competition* among neurons as:

$$B_{rt}\left(f(W)\right) = \frac{1}{2}\left(\frac{\sum_{r'} f_{r't}(W)}{N_r} + \frac{\sum_{t'} f_{rt'}(W)}{N_t}\right). \tag{5.12}$$

where in this formula:

- N_r and N_t refer to the number of neurons in the retina and tectum respectively.

As you see in the proposed formula, this system will be symmetric if the coupling functions of equation 5.10 are selected as Gaussian functions. In this case, when we start the iteration of the connectivity matrix $\mathbf{W}$ with an uniform random distribution, there are two possible final states in the one-dimensional approach. We will illustrate these states in the next section.

5.3.1 Simulation of a one-dimension Häussler system

In this section, in order to compare the Häussler self-organized mapping with our new system, we should first characterize the Häussler system by using some experiments. To avoid the complexity of two dimensional system, we do this in one dimensional approach as we did for Kohonen system before.

In difference to the Kohonen system which was presented in the previous section and our information based system which we present afterwards, this system does not need any input pattern. Then we only need to set the parameters α, η, N_r , N_t and σ in the equations 5.10 and 5.11. In this case, we put the number of the neurons both in the input and output to 20. The small noise value η, is generated from a uniform random distribution within the interval $[-0.01, +0.01]$.

Figure 5.5 illustrates the final state of the system after 5500 iterations using $\alpha = 0.1$ and $\sigma = 2$ in the first 5000 iterations and $\sigma = 1$ in the remaining 500 iterations. We changed the neighborhood effect parameter after some steps to get better result. As we discussed above, this system is symmetric. Therefore, there are two possible final states in the one-dimensional approach. In comparison with the Kohonen system, it is obviously clear that the final result shows that the mapping is completely diagonal in coherence to the desired result. However, as we will show afterwards, the run time complexity of this system is worse than the run time complexity of the Kohonen system. About the evaluation of the complexity of these systems we discuss afterward.

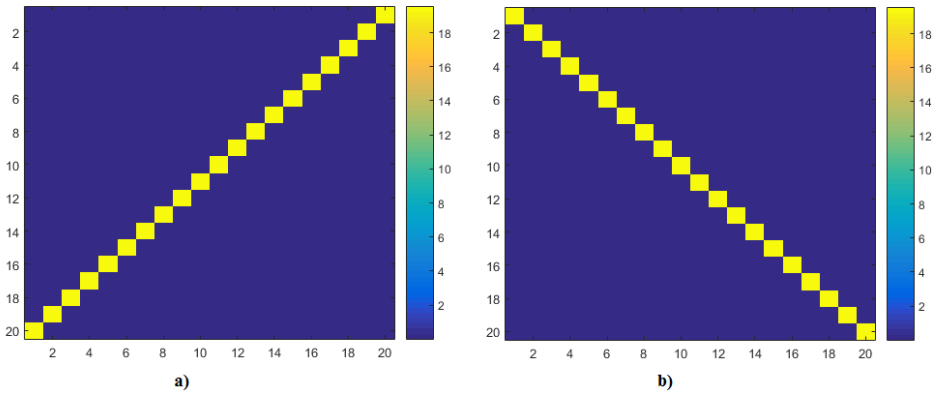

FIGURE 5.5: The final states of the Häussler system with $\alpha = 0.1$ after 5500 iterations with $\sigma = 2$ in first 5000 iterations and $\sigma = 1$ in the rest 500 iterations. a) left diagonal final state b) right diagonal final state.

5.4 The information-based self-organizing system

In this section, we will describe our approach for a self-organizing system, preserving locality information. It is well known that for classification, the Bayes optimal decision strategy is realized by choosing the object class with the highest posterior probability. In the average, this is implemented by choosing the class with shares the maximal information with the input. This has also proposed by other authors, e.g. by [84, 85] who claims that the information criteria is successful in the field of self-organization.

5.4.1 Model architecture

Our goal is to get a mapping between an input pattern and an output pattern, starting from the full-connected situation on the left side of figure 5.2 and ending up to the ordered state on the right side after some iterations.

This mapping should be formed in a self-organized way, preserving the neighborhood of input data (the local information).

As a base for comparing the patterns, let us use the Shannon information measure, based on the distribution $p(x)$ of patterns x over the event space X as follows:

$$H(X) = \sum_x p(x) \log p(x) \tag{5.13}$$

Our similarity measure is based on the information difference $H(x_i) - H(y_j)$ between the input and the output as we already proposed in the previous chapter. Additionally, we want to punish big differences much more than smaller ones. Thus, we define our similarity cost function R as a Gaussian

$$R(X_i, Y_j, r) = exp\left(-\frac{(H(X_i) - H(X_j))^2}{\sigma}\right) \tag{5.14}$$

where:

- σ is a control parameter which controls the influence of the information measurement according to biological topology.
- $H(X_i(r))$ is the local entropy of the activity $X_i(r)$ of the other neurons around neuron i within neighboring distance r.

$$H(X_i(r)) = \sum_{x \in X_i(r)} p(x) \log p(x) \tag{5.15}$$

Please note that here, we use the topological distance between the neurons, not the pattern distance or activity difference. The learning equation modulates all weights of the weight matrix $\mathbf{W}$ according to the observed pattern similarity:

$$w_{ij}(t+1) = w_{ij}(t) + \gamma w_{ij}(t) R(X_i, Y_j, r) \tag{5.16}$$

where:

- γ is the learning rate.
- The function R calculates the similarity measurement, based on local entropy between neuron i of the input neurons and neuron j of the output neurons.

Because all parameters in above equation are positive, all weights will only increase towards infinity and the learning process of the equation 5.16 will diverge. This is a common problem, often observed in unrestricted systems [82]. To avoid this, we normalize each column $\mathbf{w}_k$ of connectivity matrix $\mathbf{W}$ by

$$\mathbf{w}_k(t+1) = \frac{\mathbf{w}_k(t)}{||\mathbf{w}_k||} \tag{5.17}$$

5.5 Experimental results

In this section, we experimentally evaluate the proposed mapping self-organization. First, let us start with a simple one-dimensional mapping, comparing it to the previously described approaches of Kohonen in section 5.2 and Häussler in section 5.3, and then go on to a two-dimensional mapping used in image recognition. As training input patterns, let us use some natural images from SUN database [86]. We used some arbitrary gray scale images with pixel grids of 20×20. In figure 5.6, you can see some examples of these images.

FIGURE 5.6: Some natural images for training

5.5.1 One-dimensional input and output mapping

To feed these images to the neural network with one dimension, we used one column of each image as one pattern in the training set. By this, we extracted 34,180 pattern samples from 1709 images and used them as training set for all consecutive experiments. Within training, all samples are drawn randomly from this set. In the first step, we want to demonstrate the self-organization when starting with randomly interconnected layers in one dimension. The initial values of the weight matrix $\mathbf{W}$ were selected randomly from the range $[0, 1]$ with uniform

distribution. The initial connection with 20 input neurons and 20 output neurons is illustrated in figure 5.7. Here, row i corresponds to the $i-th$ input neuron weight value and column j corresponds to the $j-th$ output neuron, respectively. The colors encode the weight values between the $i-th$ input and the $j-th$ output.

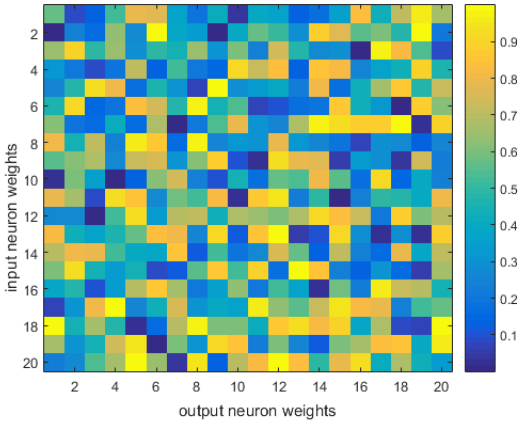

FIGURE 5.7: Illustration of the initial connectivity values

As we expect, the initial mapping is completely random. To calculate the local entropy of each input or output neuron t within a neighbor radius of r, we used the elements of input values or output values from $r-t$ until $r+t$ in the one-dimensional approach. If the indices are out of the borders we assume that the list of inputs is circularly connected (see figure 5.8).

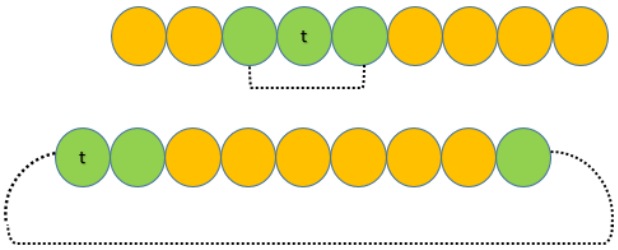

FIGURE 5.8: Two examples of neighbors of a neuron t in one-dimensional approach with a radius of neighboring $r = 1$. The neighbors of the neuron t are shown with green color.

Now, we try to illustrate the effect of each parameter (σ, γ and r) on the speed of convergence and quality of the final mapping result. Figure 5.9 shows the results of training after 1000, 5000, 10000 and 25000 iterations respectively for $\sigma = 2, \gamma = 0.05$ and $r = 2$.

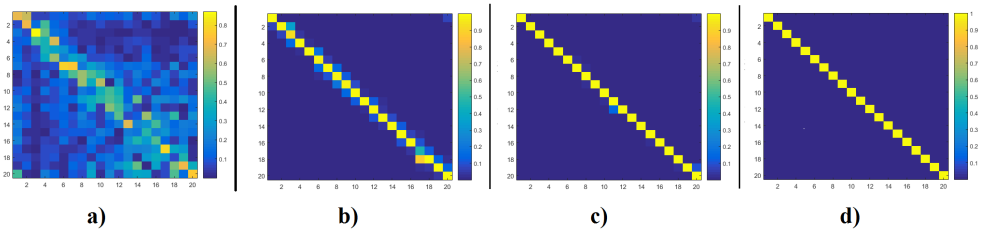

FIGURE 5.9: Neural connectivity with $\sigma = 2$ and $r = 2$ after a) 1000 iterations b) 5000 iterations c) 10,000 iterations d)25,000 iterations

As we can see, a regular mapping evolves which becomes precise during the learning process. Please note that we do not use any direct pattern comparison, only the statistics of them. To compare the effect of the similarity modulation parameter σ in equation 5.16, we performed a test with three different values of this parameter. In figure 5.10 the matrix connectivity is shown with the same neighborhood and learning rate as in figure 5.9, i.e. $r = 2$ and $\gamma = 0.05$, but with $\sigma = 2, 5$ and 10.

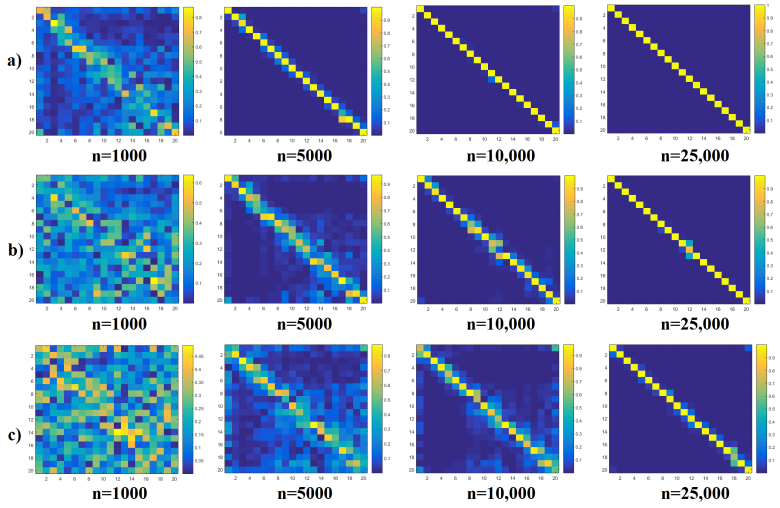

FIGURE 5.10: Neural connectivity development with $r = 2$, $\gamma = 0.05$ and with a) $\sigma = 2$ b) $\sigma = 5$ c) $\sigma = 10$. Iteration number n are shown underneath the panel. Row i corresponds to the $i - th$ input neuron weight value and column j corresponds to the $j - th$ column in output neuron respectively.

As you can see, when we increase σ from 2 to 10, the mapping still converges, but the speed of convergence decreased considerably. Now, how does the neighboring radius r influences the mapping? In figure 5.11, we try to show the effect of r, both in the speed of convergence and the quality of the result. This results has been taken by setting $\sigma = 2$, $\gamma = 0.05$ and $r = 2, 5$ and 10.

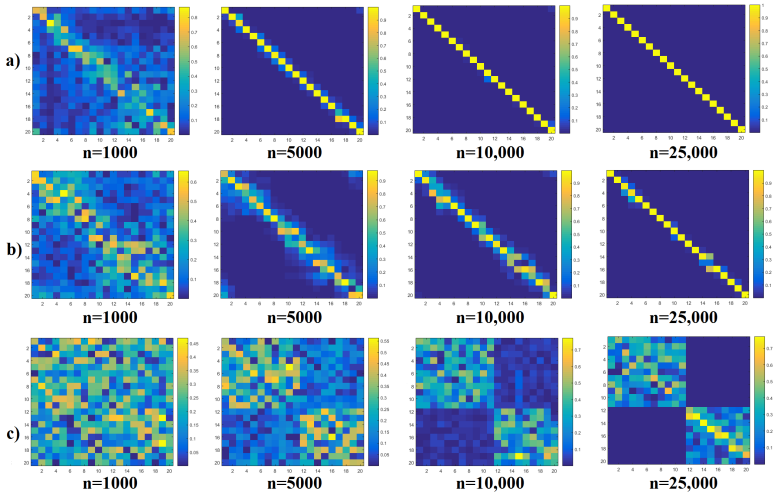

FIGURE 5.11: The connectivity matrix when we changed the radius of neighboring. In all cases $\sigma = 2$ and $\gamma = 0.05$ a) with $r = 2$ b) with $r = 5$ c) with $r = 10$. Iteration numbers n are shown underneath the panel. Row i corresponds to the $i - th$ input neuron weight value and column j corresponds to the $j - th$ column in output neuron respectively.

Compared with the results in the figure 5.9, especially the result with the neighborhood $r = 10$, it is clear that, by increasing this parameter from 2 to 10, the network converges slower and the quality of the final result is not as well as in other tests.

5.5.2 One-dimensional mappings with different number of inputs and outputs

Now, we test the case when the dimensionality of the input and output layers are different. This property can tackle the problem of scale-invariant in object recognition problems. When the objects in our view seem smaller or bigger, we have to map those objects to a constant size of the prototype. For instance, when we change the scale of an object to make it smaller, the mapping system have to map bigger number of input pixels to smaller number of output pixels. To do this, for instance, we might change the output dimension from 20 to 15. Therefore, we need training samples with different sizes for input and output. For input, we used the same training samples of the previous section and for the output, we resized the input pattern to 15 pixels by a nearest-neighbor interpolation algorithm which

is a simple, one of many algorithms for scaling an image. In this algorithm, the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered. Figure 5.12 describes an example of this algorithm which scales six pixels to three pixels.

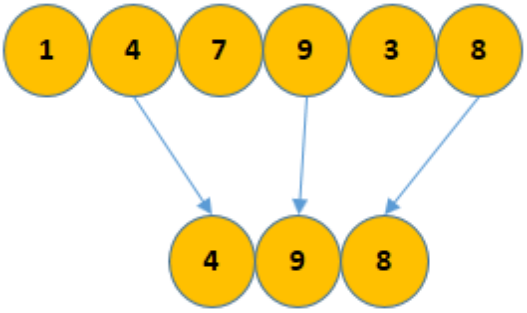

FIGURE 5.12: Example of nearest neighbor algorithm for scaling the six pixels to three pixels.

Figure 5.13 shows the results of our new model with the parameters $\sigma = 3$ and $\gamma = 0.05$. In this case, we did the test with three different approaches. In the first approach, we trained the network with a constant value of neighboring radius during the whole procedure, while in the second and the third approach we changed the neighboring radius after 40,000 and 50,000 iterations respectively.

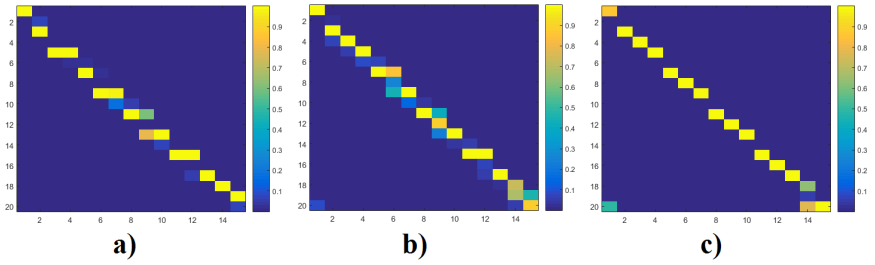

FIGURE 5.13: Neural connectivity when the number of input neuron and output neurons are different after a) 50,000 iterations with $r = 2$. b) 50,000 iterations with $r = 2$ in first 40,000 iterations and $r = 1$ in the rest 10,000 iterations. c) 100,000 iterations with $r = 2$ in first 50,000 iterations and $r = 1$ in the rest 50,000 iterations

In all three cases, we observe that the mapping still converges, although the input-output pattern pairs have changed considerably. The effect of parameters σ and

r on convergence is shown in figure 5.14. We trained the network with a) $r = 2$ and $\sigma = 10$, b) $r = 3$ and $\sigma = 3$ and finally, c) $r = 3$ and $\sigma = 10$.

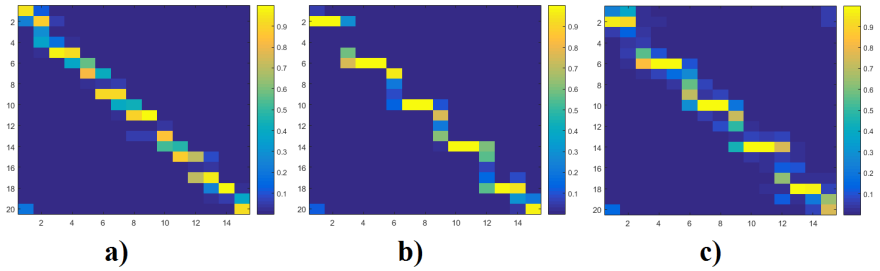

FIGURE 5.14: Connectivity matrix when the number of input neuron and output neurons are different after 50,000 iterations. a) with $r = 2$ and $\sigma = 10$, b) with $r = 3$ and $\sigma = 3$, c) with $r = 3$ and $\sigma = 10$

By comparing the results of the figure 5.13 and 5.14, we determined as best parameters $r = 2$ and $\sigma = 3$.

Now, we want to compare this behavior with the results of the Kohonen and Häussler systems when the number of input neurons and output neurons are different. In figure 5.15, this is shown for two systems with 20 neurons in the input and 15 neurons in the output.

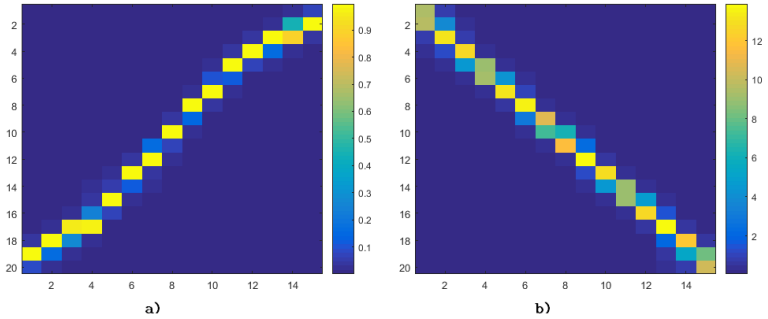

FIGURE 5.15: Connectivity matrix of Kohonen (left) and Häussler (right)systems when the number of input neuron and output neurons are different.

It is clear that in this case, the final states of both system like our proposed system, is not completely diagonal (the ideal final state for these types of mapping system when the number of input and output neurons are the same).

5.5.3 Two-dimensional mappings

In this section, we want to investigate the result of self-organizing the mappings for two-dimensional input and output layers, e.g. for the purpose of image object recognition. The mapping between two-dimensional layers is more complicated. To calculate the local entropy, in this case, we used the neighbors of a related point in both directions, in rows and columns. For instance, if we set the parameter r in equation 5.16 to 3, we have to calculate the local entropy in the 3×3 square area around the related point as a center. If the indices are outside of the borders, we see the sample as circular like we did for one dimension. Here also, like in the one-dimensional approach, we started with randomly interconnected layers by the weights in the range of $[0, 1]$ from a random uniform distribution(see figure 5.16).

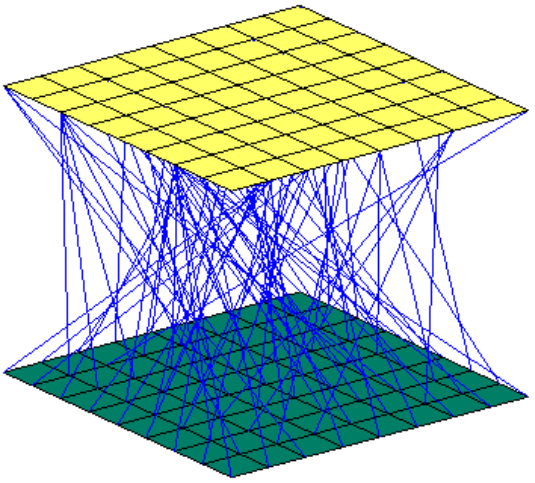

FIGURE 5.16: The initial mapping in the two-dimensional approach

The initial mapping has $9 \times 9 = 81$ input neurons and $9 \times 9 = 81$ output neurons. To visualize the four-dimensional matrix $\mathbf{W}$, we just display the strongest connection between an input neuron and all of its output neurons. In this picture, the upper surface refers to the input neurons and the lower surface refers to the output neurons.

Figure 5.17 shows the result after 25,000 iterations of the two-dimensional mapping. As you see, finally here also we got one-to-one connections between input and output neurons like in the one-dimensional approach.

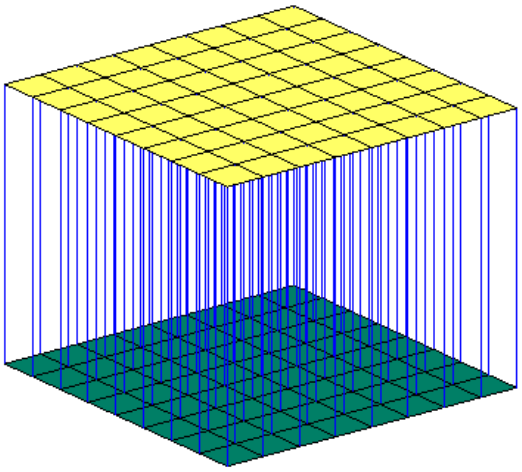

FIGURE 5.17: Matrix connectivity values in two-dimensional approach after 25000 iterations with $\sigma = 3$ and $r = 2$.

5.6 Comparing the computational complexity of three mapping algorithms

In this section we compare the three algorithm of sections 5.2, 5.3 and 5.4 regarding their complexity of computing in the one-dimensional approach. To do this, we make some assumptions and then compute the complexity.

- First, we assume that any simple mathematical operation *add*, *multiply*, *division* and *exponential* runs in $O(1)$ time.
- Second, we ignore the time of initializing the parameters and preprocessing the input patterns for all three systems.
- Third, for simplicity, we assume that the number of input neurons and output neurons has the same value N.

We only calculate the complexity of updating one synapse from input neurons to the output neuron.

Run time complexity of the Kohonen system: We need N summations in equation 5.1 to get each y_h. To calculate each z_i using equation 5.4, we need to execute calculating of y_h for N times. Therefore, the run time complexity for each

z_i will be $O(N^2)$. Computing u_i from z_i in equation 5.3 and then normalizing it by equation 5.2 can be done in $O(N)$ since we have to calculate the length of ui in each time which takes O(N). So, the run time complexity of each u_i is as:

$$O(N^2) = O(N^2) + O(N)$$

Since all elements of equation 5.2 calculated before, $w_{ij}(t+1)$ can be done in $O(1)$. Here also the normalization of each w_{ij} can be done in $O(N)$. Therefore, the total run time complexity of the Kohonen system is as:

$$O(N^2) = O(N^2) + O(N)$$

Run time complexity of the Häussler system: To calculate the complexity of the Häussler system, we have to compute the complexity of function $f_{rt}(W)$ using equation 5.10. In this equation, the neighbor effect function, $C(r, r', t, t')$, can be calculated for one time and saved in a lookup table. So, to obtain $f_{rt}(W)$, we need $N^2 = N \times N$ summations. Therefore the run time complexity of $f_{rt}(W)$ is $O(N^2)$.

Then, the complexity of the competition term $B_{rt}(f(W))$ using equation 5.12 can be computed as:

$$O(N^3) = 2 \times N \times O(N^2)$$

Therefor, the whole run time complexity of the Häussler system is as:

$$O(N^3) = O(N^3) + O(N^2)$$

Run time complexity of our proposed system: To evaluate the complexity of our information-based network, according to equations 5.14, 5.15 and 5.16, we should measure the complexity of calculating the entropy around a neuron. Because we used a histogram to estimate the probability of each input in equation 5.13, the complexity of measuring the entropy in one dimension will be $O(r)$ where r is the radius of the neighborhood to be included and it is always less than N.

Since we need to calculate the entropy both around input neuron and output neuron, the complexity of computing the entropy for two neurons will be

$$O(N) >= 2 \times O(r) = O(r)$$

using 5.15. Therefore, for all N neurons the complexity will be $O(N^2)$.

Comparing the run time complexity, the Kohonen network and our proposed system has the same run time complexity, while the Häussler system has the worst run time complexity.

5.7 Summary

In this chapter, we proposed a new method for self-organization system. We used an information theory to define a new similarity measurement. To measure the similarity between input and output neuronal activity, we employed an objective function, based on a localized Shannon entropy and weighted by a Gaussian function. The resulting neural mapping developed in two and one-dimensional cases. To compare the quality of the final result and the computational complexity of the proposed method with two other methods which discussed in section 5.2 and 5.3, we can say that the final result of our proposed method is better than Kohonen system (see figures 5.3 and 5.9). Comparing with the Häussler system, we can say that the final result of both system are the same but our proposed system has less computational complexity than Häussler system.

Chapter 6

Application on Deformation and Depth Invariance Object Recognition

In this chapter, we try to apply our proposed self-organized neural projection method developed in the previous chapter and, for comparison, a Häussler system on the task of deformation invariant object recognition.

6.1 Introduction

One of the biggest challenges in any object recognition or machine vision is invariant pattern recognition. We as humans can quickly detect and map one view of an object to other possible views (e.g. translated or rotated in any direction or angle, distorted, scaled or changed in lighting conditions) of the object at the same time, but in computer vision, this task is one of the hardest tasks. In figure 6.1, as example an object and different views of it are shown.

One instance of this task is the case when we have an object and want to find all objects in a particular data set which are similar to it. In computer vision, this problem is defined as *mapping* or *matching* between an object and the transformed

FIGURE 6.1: An object (shown within a square) and some transformations of it.

version of it. One idea to solve this issue is to extract features from the image domain and the model domain which are resistant to changes in view or illumination and to map them. In [87] they used Gabor features (*jets*) as feature extractions for a mapping mechanism using POI, which is based on a macro-columnar cortical model, and used dynamic links to perform the mapping. In [88] they treated the mapping problem as a graph-matching problem to find corresponding points (POI) in the source image and the target domain.

Now, how can these systems be applied to deformation invariant object recognition, especially also on 3D depth invariant object recognition?
For this, the rest of this chapter is as follows:
In section two, we explain both our information based system and the system which was introduced by Tomas and Von der Malsburg in [21] to solve the problem of deformation and depth invariance in a **supervised approach**, followed by some experimental results. In section three, we describe the architecture of an unsupervised approach for their system with also some experimental results on 3D depth invariant object recognition. In section four, a summary and conclusion of the results of this chapter is given.

6.2 Supervised training for depth invariant object recognition

For the task of 3D depth invariant object recognition, a dynamic link matching architecture based on a Häussler system has been proposed in the literature. The performance of this system will be compared with our new self-organized system, based on information criteria.

6.2.1 A DLM system for depth invariant object recognition

The Häussler system introduced in the previous chapter, uses a similarity measure between input and output neurons. As we already told in the previous chapter, the Häussler system does not need any input or output pattern to converge to the desired one-one projection. In comparison to the Häussler system in the previous chapter, for rotation invariant object recognition, the authors of [21] added a cross-activity term to the cooperation term of equation 5.10 and got a new cooperation term:

$$f_{rt}(\mathbf{W}) = g_{rt} + \eta + w_{rt} \sum_{r't'} C\left(r, r', t, t'\right) w_{r't'} \tag{6.1}$$

where g_{rt} is a cross-activity term which depends on the similarity between the input and output pattern:

$$g_{rt} = \alpha_m \delta_{rt} e^{-(I_r - O_t)^2 / 2k_g^2} \tag{6.2}$$

Here, we have:

- I_r and O_t are the activity of input and output neurons r and t in the source and target, respectively.
- α_m defines the maximum effect of cross-activity term in the whole system in equation 6.1.

- k_g is the parameter which controls the width of the Gaussian neighbor function.

- The function δ_{rt} which has the binary output of zero or one, is defined to block the synapses to outgrow at regions outside of the segment. They used this function to restrict the growth of synapses which connect the black regions (which has no information) of input and output patterns.

6.2.2 The new self-organized neural projection

For the task of depth invariant recognition, we also use our self-organized neural projection (SONP), which was proposed in the previous chapter. This also solves the deformation (including shift, rotation in both depth and surface) invariance problem in the task of object recognition. For this task, instead of using the same pictures as input (source) and output (target) pattern, as targets we apply the transformed versions of the source images. We explain more details about this in the section of this chapter with the experimental results.

Although we already described our system with more details in the previous chapter, let us explain now our proposed SONP briefly. The learning equation modulates all weights of the weight matrix $\mathbf{W}$ according to the observed pattern similarity:

$$w_{ij}(t+1) = w_{ij}(t) + \gamma w_{ij}(t) R(X_i, Y_j, r) \tag{6.3}$$

where γ is the learning rate. The function $\mathbf{R}$ calculates the similarity measurement, based on local entropy between neuron i of the input neurons and neuron j of the output neurons. We defined our similarity cost function $\mathbf{R}$ as a Gaussian

$$R(X_i, Y_j, r) = exp\left(-\frac{(H(X_i) - H(X_j))^2}{\sigma}\right) \tag{6.4}$$

where σ is a control parameter, and $H(X_i(r))$ is the local entropy of the activity $X_i(r)$ of the other neurons around neuron i with neighboring distance r. Consider

that we normalize each column $\mathbf{w}_k$ of connectivity matrix $\mathbf{W}$ by

$$\mathbf{w}_k(t+1) = \frac{\mathbf{w}_k(t)}{||\mathbf{w}_k||} \tag{6.5}$$

the transform should be neutral in the intensity.

6.2.3 Experimental results

In this section, we experimentally evaluate the Häussler system and our proposed mapping system for the shift and rotation in depth transformations. For training, let us use some natural images like we did in previous chapters. We used some arbitrary gray scale images with pixel grids of 20×20 as input pattern and transformed version of them as output pattern in supervised approaches. In figure 6.2, you can see some examples of these images. To avoid the complexity and to get better performance in run time we used only the one-dimensional approach of these systems. To feed these images to the neural network with one dimension, we used one column of each image as one pattern in the training set. By this, we

FIGURE 6.2: Some natural images for training

extracted 34,180 pattern samples from 1709 images and used them as the training

set for all consecutive experiments. Within training, all samples are drawn randomly from this set. To do this, we used some natural images as input pattern and transformed version of those pictures as output.

Before talking about the result and make some comparison, we want to have a close look at the concept of *rotation in depth*. When we rotate an object in depth and map it in 2D space, the output image can be seen as a *projective transformation* of the original image (see figure 3.2).

In projective transformation angles between lines are not invariant, so the parallel line in an original image are not parallel in the output image [25]. But for small patches of an object, the projective transformation can be approximated as changing the scales of a patch in width and height. In figure 6.3, we displayed an object and a projective transformation of it. In part (c) of this figure, we extracted a small patch from the original image (left picture) and its transformed patch from the transformed image (right picture). These two small patches can be seen as scaling of each other, whereas the deformation due to the 3D effect is nearly not visible. Thus, the scaling approach to deformation can be seen as the first, dominant linear term in the development of a non-linear function, restricted to a small local patch of image. The whole nonlinear deformation mapping is performed by a big number of small patches, each one performing only individual scaling. The patches cover the image as the overlapping receptive fields introduced in chapter 3(see figure 3.2). In the first step, we want to demonstrate the self-organization by starting with randomly interconnected layers in one dimension. The initial values of the connectivity matrix, $\mathbf{W}$ were randomly selected uniformly from the range $[0, 1]$. The initial connections with 20 input neurons and 20 output neurons are illustrated in figure 6.4. Here, all connections are indicated by a color: from blue for no connection until yellow for a good one.

As we expect, the initial mapping is completely random. In figure 6.5, the results of mapping for both systems are illustrated when we used a scaled version of input pattern as output pattern. As you can see, the final results of both systems are qualified as mappings for small patches (Here 20 by 15) and can be used for the

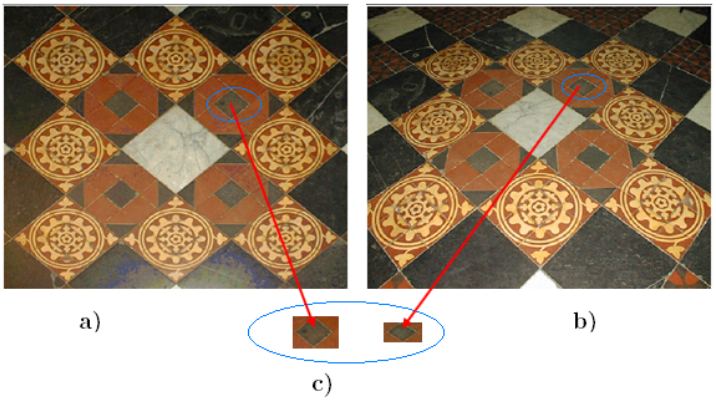

FIGURE 6.3: Example of an image and its projective transformation. a) original image b) projective transformation of the original image c) A small patch from the original image and its transformed patch in the output image. Images of (a) and (b) are taken from [25].

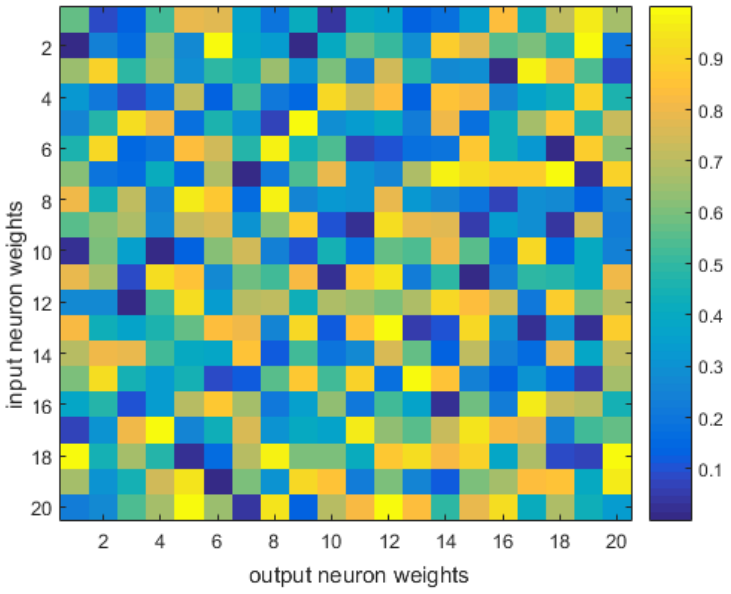

FIGURE 6.4: Initial states of the connectivity matrix.

task of mapping rotations in depth when the patches are small, although they are not perfect. The ideal final state for these types of mapping when the number of input and output neurons are the same and should be completely diagonal as we saw in the previous chapter. But with different numbers of input and output neurons the final state is not completely diagonal because the diagonal matrix can be defined only for a square matrix.

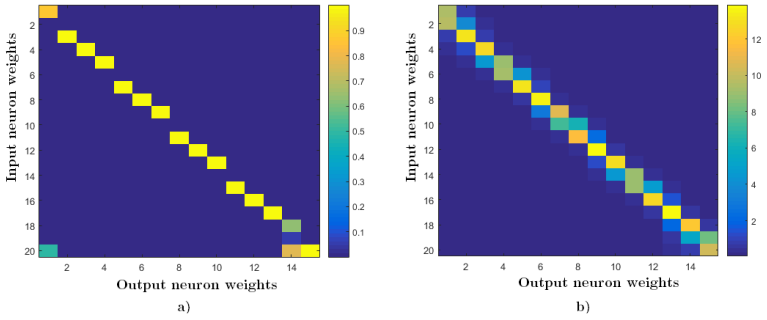

FIGURE 6.5: The final neural connection using scaled version of input as output pattern a) in our proposed SONP b) in the system proposed by [21]. The output pattern is taken by 75% scaling down of the input pattern.

6.3 Checking an unsupervised model architecture

In this section, we explain the unsupervised architecture of the system which was introduced by Tomas and Von der Malsburg in [21]. They applied it for the shift and rotation in 2D space. The good results motivated us to check its usability for the case when the transformation is in depth, not only in 2D. Again, we check the ability for small patterns, i.e. for scalings.

As we mentioned before, their system has two main possible structures: a supervised and an unsupervised model. In the supervised model, they applied the transformed version of input pattern for output pattern. Therefore, after some iterations, the system learns the related transformation. This, we presented in the previous section.

In this section, we will use the unsupervised training. In the unsupervised model, they applied a more powerful approach that does not need any pattern in output (target) and the target image initialized by some random value in $[0, 255]$.
Instead of only one connection between each input and activity pattern I_r and output activity pattern O_t there are multiple synaptic connections, each under the command of the particular control unit.

During the learning process, these control units compete for the representation of maps between input and output pattern. To achieve this, they have cooperation

and competition mechanisms, working together in parallel.

Control units control the state of a set of synapses projections corresponding to a particular mapping. The basic structure of this system is illustrated in figure 6.6. As you see in this figure, their mappings connect the source domain (input neurons I_r) to the target domain (output neurons O_t) each one controlled by a related control unit (CU1, CU2 and CU3). Then they labeled the synaptic weights between input neuron r and output neuron t which is governed by control unit u by $\mathbf{W}_{rt}^{u}$. In this figure, each group of synapses which is controlled by a specific control unit is distinguished by color (e.g. red or green) from others. Now, we

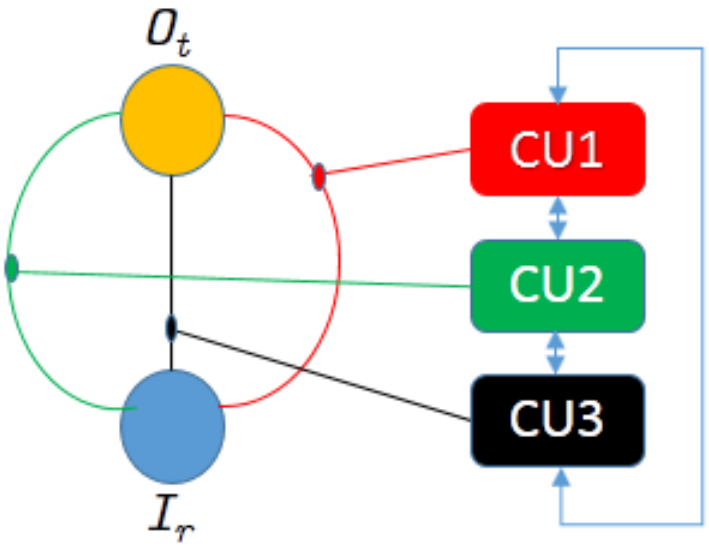

FIGURE 6.6: Controlled mappings between input neuron I_r and output neuron O_t. In this figure, a controlled mapping structure with three units is shown. Control units have competition and cooperation functions in parallel.

want to talk about the control unit activity algorithm, target pattern update rule and synaptic weight changes based on this structure.

6.3.1 Control unit activation

They changed the primary Häussler system (see equation 5.9) in the cooperation term and the activity of control units to get the new update rule by multiplication

of a term $S(u)$ as:

$$\dot{w}_{rt}^{u} = S(u)\left[f_{rt}(\mathbf{W}^{u}) - w_{rt}^{u} B_{rt}(f(\mathbf{W}^{u}))\right] \tag{6.6}$$

with the new cooperation term defined in equation 6.1. In this equation, the activity function $S(u)$ has the value of $0, 1$ and acts to indicate *active* and *inactive* states of the control unit u. Therefore, only the active units can update their relevant synaptic connection and others can not. Therefore, the main problem would be the decision about activation and deactivation of control units which should be done by this function. In the rest of this section we present the elements which compose the new function $S(u)$. So, $S(u)$ is defined as:

$$S(u) = \begin{cases} 1, & \text{if } P(u) > randomreal(0,1) \\ 0, & \text{otherwise} \end{cases} \tag{6.7}$$

Where *randomreal* is a function that generates a random value in $[0, 1]$ with uniform distribution. The definition of $P(u)$ will be given later on.

Activation and deactivation of each control units depend on two things: (1) activation of other units, especially the activation of the units which has less distance to current unit, and (2) similarity between input and output targets, weighted by synaptic connectivity between them. The former is done by a function $L(u)$ and the latter by a function $E(u)$. Each unit will be active with the probability of

$$P(u) = \frac{1}{1 + e^{-k_s(E_{tot}(u)+\beta)}}, \tag{6.8}$$

wherek_s and β are the control parameters which control the steepness of the function and the bias, respectively. Here, they defined E_{tot}, the *total excitation* of the each control unit u, as a weighted sum of two terms, the excitatory input of the control unit,$E(u)$and interaction among control units, $L(u)$ as:

$$E_{tot}(u) = bE(u) + (1 - b)L(u). \tag{6.9}$$

They defined $E(u)$ as weighted summations of the similarity of the input and output patterns and got the equation:

$$E(u) = \frac{1}{N^u} \sum_r \sum_t W_{rt}^u e^{-\frac{(I_r - O_t)^2}{2k^2}} \tag{6.10}$$

with

$$N^u = \sum_{rt} W_{rt}^u \tag{6.11}$$

where k is a parameter which controls the influence of similarity on the whole function $E(u)$. I_r and O_t are the input and output activity patterns respectively, r and t are indices of source and target points in and , respectively, and N^u is a normalization factor. Please note that here, the similarity in the excitatory term is measured by the mean squared error again which is sensitive to outliers.

The interaction function $L(u)$ of the control units is computed as:

$$L(u) = \sum_{u' \neq u}^{U} C(d_{uu'}) S(u'), \tag{6.12}$$

where U is the number of control units, $S(u')$ is the activity state of unitu' , and $C(.)$ is the Gaussian coupling function defined as follows:

$$C(d_{uu'}) = e^{-d_{uu'}^2 / 2\sigma^2} \tag{6.13}$$

where $d_{uu'}$ is the distance between the control units and σ is the local strength control parameter. If the indices are out of the borders they assumed that the list of inputs is circularly connected as we did for the distance between neurons in our SNOP system.

Output pattern updates rule

As we talked before, in the unsupervised model, they applied a more robust approach that does not need any pattern as output (target) and the target image is initialized by some uniform random values from $[0, 255]$. Then, the update

equation for the output pattern is:

$$\Delta O_t = \gamma(I'_t - O_t) \tag{6.14}$$

where γ is an update rate and $I'_t = I_r$ is the activity of the source domain unit with the strongest active connection $W^{u'}_{r't}$ to t.

$$r' = \underset{r,t}{\operatorname{argmax}}\, S(u) W^u_{rt} \tag{6.15}$$

making a strong connection even stronger.

6.3.2 Experimental results of the unsupervised approach

In this section, we want to see the result of applying the unsupervised model above to small 3D mappings (scaling) using a system with six control units. As we talked before, in this case we do not have any output pattern, and output patterns start with a random value in the range $[0, 1]$ with uniform distribution. For the input pattern we used the original input pattern and transformed version of it with scaling factors of 0.9, 0.8,0.7, 0.6 and 0.5. To get the same image size, we filled the extra space in the images with zeros, see figure 6.7. As we told before, to avoid the

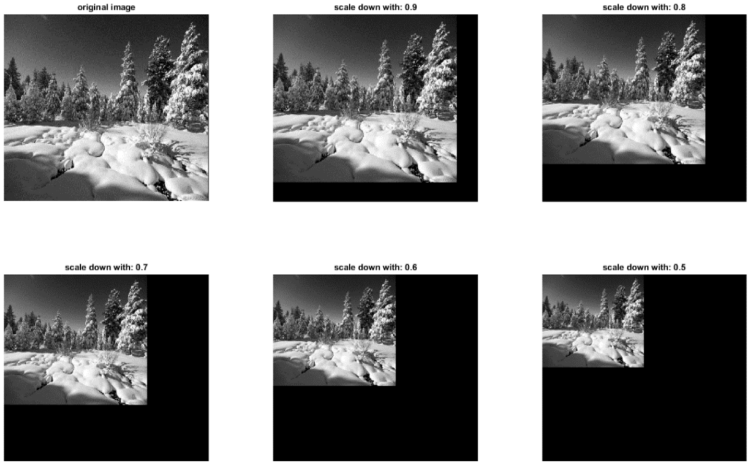

FIGURE 6.7: Sample image (top left) and its transformed version with multiple scaling factors.

complexity of running time performance, we used the one-dimensional approach of

the system. In figure 6.8, the final mappings for different scaling factors after some iterations is illustrated. As showed in figure 6.8, after some iterations starting with

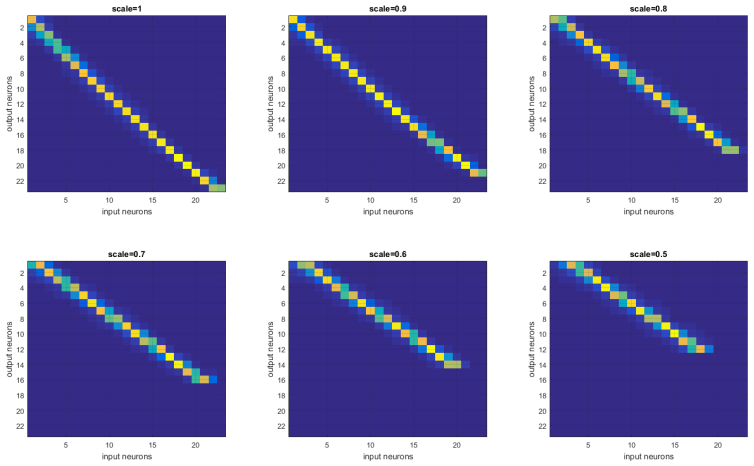

FIGURE 6.8: Results for the 1D-mapping algorithm. The weight values are shown for the sample image (top left) and its transformed version with multiple scaling factors.

randomly initialized weights and also random output pattern the system could do a consistent mapping between the input pattern and non-zero parts of output pattern,. Therefore after convergence, if we put any image as an input pattern, we can get the six different scaled transformed image of it as output pattern with six control units.

The results show the robustness and capability of this system for the task of depth deformation invariant object recognition.

Nevertheless, a self-organized, probability based approach is also needed for this unsupervised task. Thus we will develop our SNOP system in future also in this direction.

6.4 Summary

In this chapter, first we compared the *supervised* system proposed by Tomas et. al. for the task of deformation invariant object recognition to our information based mapping. We used the improved architecture of the Häussler system which has been proposed by Tomas and Malsburg in [21]. They applied it for the shift

and rotation in 2D space. The good results were a motivation for the case when the transformation is in 3D depth, not only in 2D. The results qualified our new system at least as good as the Häussler system, although it uses less computational power and is less influenced by outliers due to the probability-based approach. Then, in the case of the *unsupervised* approach, we showed the capability of the improved Häussler system for the task of rotation in depth.

Certainly, for the unsupervised case, an information-based approach is desired also. For this, we will extend our system in this direction in future work.

Chapter 7

Discussion and Conclusion

In this thesis, we presented multiple work for the task of universal feature extraction and an information based self-organized system to perform the task of invariant object recognition. Different methods exist for selecting features and performing deformation invariant image recognition. As we mentioned, the methods based on POI have the problem that they are task specific and need some hand-crafted algorithms which change from application to application. Statistical based methods for feature extraction like PCA and ICA have the assumption that the original data should be separated in a linear approach, but this assumption is not valid for most of the data in the real world. ANN-based methods try to solve the nonlinearity problem of PCA and ICA, but they have the problem that they need many training data (around millions) especially when the input data are transformed version of each other. Another problem is that these methods are not robust in the case of transformation, translations or the change in the position of the objects or it is very difficult to do this task by ANNs. Although the dynamic link model DLM shows interesting results, this system has the problem of a high computational complexity. In addition, because it uses the least mean squared error as risk function, the performance for classification is also not optimal. For random values where outliers are present, this system may not work well because outliers influence the mean squared error classification much more than

probability-based systems. Therefore, the DLM system should be completed by a new approach.

7.1 Universal feature extraction

In our contribution, firstly, we tried to find features which can be used for multiple classification tasks and second, we introduced a new system which employs the information criteria (i.e. probabilities) to overcome the problem of DLM system and has a smaller computational complexity. First, in Chapter 3 we proposed a new method for universal feature extraction which is more general than the previous tasks which have been done before e.g. transfer learning, semi-supervised learning and self-taught learning. To do this, we used an information theoretic approach to design a proper risk function which leads to cross-entropy minimization. We developed a feed forward neural network as a basic structure to extract universal features. Additionally, to reduce the number of parameters to learn, we used a weight sharing method for all receptive fields as constraint. In addition to reducing the number of learning parameters it has the benefit that the shared weights make all neurons detecting the same features, independent of their different positions in the input image. Although the labeling of the filter properties of the first layer as "features" is plausible, but arbitrary, the results showed that those *universal features* are unique and can be successfully applied in very different image processing applications e.g., handwritten digit classification, recognition of natural or artificial objects which are placed in the natural or artificial background images and recognition of texture. In chapter 4, we added a second hidden to the shallow neural network from the previous chapter. By comparing the results of the shallow network with the multi-layers network, it is clarified that the multi-layer has better accuracy when the train set and test set are the same or close to each other. The surprising result is that the shallow network has better performance than the multi-layers when we compared the accuracy rate of classification for MNIST hand-written digits as a test set. It could be interpreted as in the multi-layers network the features become more specific from lower layers

to higher layers. On the other hand, the features computed by the last layer of a trained network must depend greatly on the chosen dataset and task. Comparing the result of shallow network and multi-layers network clarifies that also by adding one layer to the shallow network we got a better result but sometimes by changing the other parameters (e.g. number of hidden unit or initialization variable) it is possible to achieve the result as same as multi-layers. This result agrees with the results of Adam et.al. in [68]. They showed that several factors, such as the number of hidden nodes in the model, may be as important as the choice of learning algorithm or the number of hidden layer. For instance by tuning the parameters in K-means, they could get better accuracy rate of classification than the state of the art results of the deep neural network, deep belief network and convolutional neural network for the task of NORB objects classification. About the transferability of features in layers above the first layer of the deep network, we found that two research attempts has been done simultaneously to our work, see ([89] and [90]). Unlike our work, in their work the source and target task was related such that they reused the features of seven layers extracted of the classification of objects in the ImageNet database for the objects of PASCAL VOC dataset([89]). In [90] the authors tried to use the features of one subset of ImageNet dataset for another subset of this data set. Like our work, they also found that transferring features even from distant tasks can be better than using random features. Like our work, in these two works the features become also more specific from lower layers to higher layers.

7.2 Deformation invariant object recognition

In chapter 5, we proposed a new method for self-organization system. We used an information theoretic approach to define a new similarity measurement. To measure the similarity between input and output neuronal activity, we employed an objective function, based on a localized Shannon entropy and weighted by a Gaussian function. The resulting neural mapping developed in two-dimensional (image) and one-dimensional cases. To compare it with another state of the art

self-organized system, we can say that the final result of our proposed method is better than Kohonen system. Comparing with the Häussler system, we can say that the final result of both systems is roughly the same, but our proposed system has less computational complexity than Häussler system. In chapter 6, first we compared the supervised system proposed by Tomas et. al. for the task of deformation invariant object recognition to our information based mapping. Here, we used the improved architecture of the Häussler system which has been proposed by Tomas and Malsburg in [21]. They applied it for the shift and rotation in 2D space. The good results were a motivation for the case when the transformation is in 3D depth, not only in 2D. The results qualified our new system at least as good as the Häussler system, although it uses less computational power and is less influenced by outliers than the Häussler system due to the probability-based approach. Additionally, in the case of the unsupervised approach, we showed the capability of Häussler system for the task of depth-rotation. Certainly, for the unsupervised case an information-based approach is desired also. For this, we will extend our system in this direction in future work. To compare our work with the work of Kishore Konda [91], the latter can only detect orthogonal transformations and it does not work for the task of rotation and specially rotation in depth.

7.3 Future work

In invariant object recognition by unsupervised Häussler systems, for each whole mapping we used a specific control unit, but this is not realistic in the real world. In the real world, we guess that human brain uses combinations of some local transformation to perform the whole mapping. Therefore, in the real world each control unit is responsible for a small local transformation and by cooperation between these many control units, we can understand the whole mapping. This mechanism can solve a restriction of the present model to deal not only with rigid transformations but with deformable mappings as well. For this reason, we employed the deformation of object surfaces during rotation in depth. By

developing our proposed method for the task of unsupervised depth-deformation, we hope to overcome this problem.

Bibliography

[1] D. J. Felleman and D. C. Van Essen. Distributed Hierarchical Processing in the Primate Cerebral Cortex. *Cerebral Cortex*, 1(1):1–47, jan 1991.

[2] Mary C Potter. Short-Term Conceptual Memory for Pictures. *Journal of Experimental Psychology: Human Learning & Memory*, 2(5):509–522, 1975.

[3] JamesJ. DiCarlo, Davide Zoccolan, and NicoleC. Rust. How Does the Brain Solve Visual Object Recognition? *Neuron*, 73(3):415–434, 2012.

[4] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[5] D Marr and E Hildreth. Theory of Edge Detection. *Biological Sciences*, 207(1167):187–217, 1980.

[6] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[7] T Lindeberg. Edge detection and ridge detection with automatic scale selection. *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, 30(2):465–470, 1996.

[8] A Willis and Yunfeng Yunfeng Sui. An algebraic model for fast corner detection. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2296–2302. IEEE, sep 2009.

[9] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988.

[10] T. Lindeberg. *Scale-Space Theory of Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I"*, pages 404–417. Springer Berlin Heidelberg, 2006.

[13] K. Mohiuddin Anil K. Jain, Jianchang Mao. Artificial Neural Networks: A Tutorial. *IEEE Computer*, 29(3):31–44, 1996.

[14] Teuvo. Kohonen, M. R. Schroeder, and T. S. Huang. *Self-organizing maps.* Springer-Verlag Berlin Heidelberg, 2001.

[15] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[16] Xinyu Guo, Xun Liang, and Xiang Li. A Stock Pattern Recognition Algorithm Based on Neural Networks. In *Proceedings of the Third International Conference on Natural Computation - Volume 02*, pages 518–522, Washington, DC, USA, 2007. IEEE Computer Society.

[17] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable Object Detection using Deep Neural Networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162. IEEE Computer Society, jun 2014.

[18] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qiu, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy, and Xiaoou Tang. DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2015.

[19] Jürgen Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[20] Davide Zoccolan. Invariant visual object recognition and shape processing in rats. *Behavioural Brain Research*, 285:10–33, 2015.

[21] T Fernandes and C von der Malsburg. Self-Organization of Control Circuits for Invariant Fiber Projections. *Neural Computation*, 27(5):1005–1032, 2015.

[22] A F Häussler and C von der Malsburg. Development of retinotopic projections: an anlytic treatment. *J. Theor. Neurobiol.*, 2:47–73, 1983.

[23] K Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2):251–257, 1991.

[24] M Haibach. *Informationsgesteuerte, Adaptive Extraktion von Merkmalen*. PhD thesis, Goethe-university, Frankfurt am Main, 2011.

[25] Richard. Hartley and Andrew. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[26] M. Fernández-Redondo and C. Hernández-Espinosa. Weight initialization methods for multilayer feedforward. In *Proceedings of the European Symposium on Artificial Neural Networks*, number April, pages 119–124, apr 2001.

[27] Jianjun Shi. Principal Component Analysis and Factory Analysis. In *Stream of Variation Modeling and Analysis for Multistage Manufacturing Processes*, chapter 5, pages 91–113. CRC Press, dec 2006.

[28] Shifei Ding, Hong Zhu, Weikuan Jia, and Chunyang Su. A survey on feature extraction for pattern recognition. *Artif Intell Rev*, 37(3):169–180, 2012.

[29] R A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[30] Hamid Abrishami Moghaddam Youness Aliyari Ghassabeh, Frank Rudzicz. Fast incremental LDA feature extraction. *Pattern Recognition*, 48(6):1999–2012, 2015.

[31] Aleix M Martínez and Avinash C Kak. Pca versus lda. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(2):228–233, 2001.

[32] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[33] D J Bartholomew. Factor Analysis for Categorical Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(3):293–321, 1980.

[34] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[35] Qiang Yang. a Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[36] Zhijie Xu and Shiliang Sun. Part-based transfer learning. In *Advances in Neural Networks – ISNN 2011: 8th International Symposium on Neural Networks, ISNN 2011, Guilin, China, May 29–June 1, 2011, Proceedings, Part III*, pages 434–441, Guilin, China, 2011. Springer Berlin Heidelberg.

[37] Huaxiang Zhang. Transfer Learning through Domain Adaptation. In *Advances in Neural Networks – ISNN 2011: 8th International Symposium on Neural Networks, ISNN 2011, Guilin, China, May 29–June 1, 2011, Proceedings, Part III*, pages 505–512. Springer Berlin Heidelberg, 2011.

[38] K Nigam, A K McCallum, S Thrun, and T Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134, 2000.

[39] Alexander Chapelle, Olivier and Scholkopf, Bernhard and Zien. Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006)[Book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[40] G E Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, jul 2006.

[41] R Raina, A Battle, H Lee, B Packer, and A Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning.*, pages 759–766. ACM, 2007.

[42] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

[43] Siddharth Saxena and Rajeev Kumar Singh. A Survey of Recent and Classical Image Registration Methods. *International Journal of Signal Processing Image Processing and Pattern Recognition*, 7(4):167–176, 2014.

[44] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A Survey on Shape Correspondence. In *EuroGraphics: State-of-the-Art Report*, pages 61–82, 2010.

[45] M Lades, J C Vorbruggen, J Buhmann, J Lange, C von der Malsburg, R P Wurtz, and W Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42(3):300–311, mar 1993.

[46] Lindi J Quackenbush. A Review of Techniques for Extracting Linear Features from Imagery. *Photogrammetric Engineering & Remote Sensing*, 70(12):1383–1392, dec 2004.

[47] Dong Ping Tian. A Review on Image Feature Extraction and Representation Techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396, jul 2013.

[48] Tommy W S Chow and M K M Rahman. A New Image Classification Technique Using Tree-structured Regional Features. *Neurocomput.*, 70(4-6):1040–1050, jan 2007.

[49] Chih-Fong Tsai and Wei-Chao Lin. A Comparative Study of Global and Local Feature Representations in Image Database Categorization. In *NCM*, pages 1563–1566, 2009.

[50] Dan C. Cirean, Ueli Meier, and Jurgen Schmidhuber. Transfer learning for Latin and Chinese characters with Deep Neural Networks. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, 2012.

[51] David G. Duda, Richard O. and Hart, Peter E. and Stork. *Pattern Classification*. Wiley & Sons, New York, 2001.

[52] T Cover and J Thomas. *Elements of Information Theory*. Wiley & Sons, New York, 1991.

[53] T Mitchell. *Machine Learning*. McGraw Hill, 1997.

[54] P Sibi, S.Allwyn Jones, and P.Siddarth. Analyses of Different Activation Functions Using Back Propagation Neural Networks. *Journal of Theoretical and Applied Information Technology*, 47(3):1264–1268, jan 2013.

[55] Bekir Karlik and Ahmet Vehbi. Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 1(4):111–122, 2011.

[56] Wlodzislaw Duch, Rafal Adamczak, and Norbert Jankowski. Initialization and Optimization of Multilayered Perceptrons. In *3rd Conf. on Neural Networks and Their Applications*, pages 105–110, 1997.

[57] Y K Kim. Weight Value Initialization for Improving Training Speed in the Backpropagation Network. In *Proc. of Int. Joint Conf. on Neural Networks*, volume 3, pages 2396–2401, 1991.

[58] Jan-Mark Geusebroek, Gertjan J. Burghouts, and Arnold W.M. Smeulders. The Amsterdam Library of Object Images. *International Journal of Computer Vision*, 61(1):103–112, jan 2005.

[59] Le Cun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann, 1990.

[60] B Caputo, E Hayman, and P Mallikarjuna. Class-specific material categorisation. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1597–1604 Vol. 2, oct 2005.

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

[62] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, jun 2006.

[63] Charles E Metz. Basic Principles of ROC Analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, oct 1978.

[64] Christopher M Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[65] Christoph von der Malsburg. Network Self-Organization in the Ontogenesis of the Mammalian Visual System. In S F Zornetzer, J Davis, C Lau, and Th. McKenna, editors, *An Introduction to Neural and Electronic Networks (Second Edition)*, pages 447–462. Academic Press, 1995.

[66] S. Marcelja. Mathematical description of the responses of simple cortical cells. *J Opt Soc Am*, 70(11):1297–1300, nov 1980.

[67] D H Hubel and T N Wiesel. Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, 148(3):574–591, 1959.

[68] A Coates, H Lee, and A Y Ng. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 215–223. JMLR W&CP, 2011.

[69] Douglas M. Kline and Victor L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing and Applications*, 14(4):310–318, jul 2005.

[70] P. Golik, P. Doetsch, and Hermann Ney. Cross-Entropy vs. Squared Error Training:a Theoretical and Experimental Comparison. In *INTERSPEECH, ISCA*, pages 1756–1760, 2013.

[71] Alex Krizhevsky, I Sutskever, and G E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[72] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[73] V. M. Tikhomirov. On the Representation of Continuous Functions of Several Variables as Superpositions of Continuous Functions of one Variable and Addition. In *Selected Works of A. N. Kolmogorov*, pages 383–387. Springer Publishing Company, Incorporated, 1991.

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

[75] Yoshua Bengio and Yann LeCun. *Scaling Learning Algorithms towards AI*, pages 321–360. Number 1. MIT Press, 2007.

[76] N Suga, H Niwa, I Taniguchi, and D Margoliash. The personalized auditory cortex of the mustached bat: adaptation for echolocation. *J Neurophysiol*, 58(4):643–654, oct 1987.

[77] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[78] David J Willshaw and Christoph von der Malsburg. How patterned neural connections can be set up by self-organization. In *Proceedings of the Royal Society London*, volume B194, pages 431–445, 1976.

[79] Christoph von der Malsburg. Nervous Structures With Dynamical Links. *Ber. Bunsenges. Phys. Chem.*, 89(6):703–710, 1985.

[80] Christoph von der Malsburg. Self-Organization in the Brain. In M A Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 1002–1005. The MIT Press, second edi edition, 2003.

[81] J. Lücke, C. von der Malsburg, and R. Würtz. Macrocolumns as decision units. In *Proceedings of the International Conference on Artificial Neural Networks , ICANN '02*, volume 2415 of *LNCS 2415*, pages 57–62. Springer-Verlag, London, UK, 2002.

[82] Teuvo Kohonen. Self-organizing neural projections. *Neural Networks*, 19(6-7):723–733, 2006.

[83] P Dean, P Redgrave, and G W Westby. Event or emergency? Two response systems in the mammalian superior colliculus. *Trends Neurosci*, 12(4):137–147, 1989.

[84] Ryotaro Kamimura. Input information maximization for improving self-organizing maps. *Appl. Intell.*, 41(2):421–438, 2014.

[85] Ryotaro Kamimura. Self-Organized Mutual Information Maximization Learning for Improved Generalization Performance. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 1613–1618, 2015.

[86] Jianxiong Xiao, James Hays, Krista A Ehinger, and Antonio Torralba. SUN Database : Large-Scale Scene Recognition from Abbey to Zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485 – 3492. IEEE Computer Society, 2010.

[87] Yasuomi D Sato, Christian Wolff, Philipp Wolfrum, and Christoph Von Der Malsburg. Dynamic Link Matching between Feature Columns for Different Scale and Orientation. In *Neural Information Processing, 14th International*

Conference, pages 385–394, Kitakyushu, Japan, 2007. Springer-Verlag Berlin Heidelberg.

[88] Wolfgang Konen and Jan C Vorbruggen. Applying Dynamic Link Matching to Object Recognition in real world images. In Stan Gielen, editor, *International Conference on Artificial Neural Networks (ICANN)*, pages 982–985, Amsterdam, 1993. Springer London.

[89] Maxime Oquab, Maxime Oquab, Ivan Laptev, Josef Sivic Learning, Transferring Mid-level, Maxime Oquab, and Leon Bottou. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, Columbus, OH, 2014. Ieee.

[90] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27 (Proceedings of NIPS)*, volume 27, pages 1–9. Curran Associates, Inc., nov 2014.

[91] Kishore Reddy Konda. *Unsupervised relational feature learning for vision.* PhD thesis, Goethe University Frankfurt am Main, 2016.

Curriculum Vitae

Mohammad Amiri

Birth date & place: 06.05.1977- Sary, Iran

Robert Mayer Str. 11-15

60325 Frankfurt, Germany

Email: amiri@ fias.uni-frankfurt.de

Education

- 2013 - Present, Goethe University, Frankfurt, Ph.D. in computer science.
- 2000 - 2003, Isfahan University, Iran , M.Sc in computer science.
- 1995 - 2000, Sharif University of Technology, Iran , Bachelor in computer engineering.
- 1991 - 1995, Firouzjaeean High School, Iran, Diploma in mathematics and physics.

Teaching or Teacher Assistance Experience

- Image Processing Seminar , 2013 - 14, Goethe University, Frankfurt, Germany
- Matlab Software, 2010 - 12, Payame Noor University, Sari, Iran
- C++ Programming Language, 2006-10, Payame Noor University, Sari, Iran
- Logic Circuits, 2006 - 12, Payame Noor University, Sari, Iran
- Microprocessors, 2007 - 12, Payame Noor University, Sari, Iran

- Assembly Language , 2006 - 12, Payame Noor University, Sari, Iran

Industrial Working Experience

- Computer and Industrial Automation
  Working with micro-controllers to control DC motors ,Jahad Daneshgahi Sharif, 2000 - 2001.

- Computer and Industrial Automation
  Working with micro-controllers in measuring and monitoring of process such as temperature and pressure.Iran research organization for science & technology, 2001 - 2003.

- Expert of Field Instruments and Measuring
  Working with measuring instrument such as flow meter and temperature controller and industrial networks, Mazandaran wood & paper industry, Iran, 2002 - 2003.

Awards and Honers

- Ranked 4^{th} amongst 500 examinees in the regional-wide keen students entrance exam for special high school.

- Ranked 200^{th} amongst 5000 examinees in the Irans national university entrance examination (Konkoor) for M.Sc. in computer science and engineering.

- Ranked 230^{th} amongst $50,000$ examinees in the Irans national university entrance examination (Konkoor) for B.Sc. in engineering and science.

- Outstanding graduate
  (Graded A) of the M.Sc. program, Department of Computer Engineering, Isfahan University, Isfahan, Iran.

Skills

- Programming
  C/C++, Matlab, Python , Linux environment and OpenCV.
- Languages
  Persian :native English: fluent

Publications

- M. Amiri, R. Brause, Self-organized neighborhood preserving projections using information,The 28^{th} annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2016).(accepted)

- M. Amiri, R. Brause, Information Based Universal Feature Extraction in Shallow Networks, *International Journal of Pattern Recognition and Artificial Intelligence* (accepted)
- M. Amiri, R. Brause, Information based universal feature extraction. *In Seventh International Conference on Machine Vision (ICMV2014), volume 9445 of , page 94450D, February 2015.*
- M. Amiri, M. Monemizadeh, Hadamard-Gabor filter: a rotation invariant feature extraction algorithm (under review)
- M. Amiri, M. Monemizadeh, Fast texture similarity search using Walsh-Hadamard transform (under review)
- M. Amiri, M. Rahimi Resketi, An Edge Method of Steganography. *World congress of computer, science and electrical engineering, Dubai, Jan. 2009*
- M.amiri, N. Babazadeh, Hiding messages in the noise pixels of a bitmap picture as background. *National conference of computer application, Lahijan, Feb.2009.*

- M. Amiri, M. Rahimi Resketi, Hiding messages in the text with combination of steganography and cryptography , *National conference of computer and electronic, Gonabad, Sep. 2009.*

Some Passed Courses

- Machine learning I, machine learning II, image processing, theoretical Neuroscience, artificial neural networks and digital signal processing.

Research Interests

- Machine vision, machine learning, image processing and artificial neural networks.

References

- **Prof. Rüdiger Brause**
  Goethe University
  Department of computer science and mathematics
  Robert-Mayer Str. 11-15, D- 60054 Frankfurt
  Email: *R.Brause@ informatik.uni-frankfurt.de*

- **Prof. Christoph von der Malsburg**
  Goethe-University Frankfurt
  Ruth-Moufang-Str. 1
  D-60438 Frankfurt a. M. Germany
  Room: 2.202, Email: *malsburg@ fias.uni-frankfurt.de*

- **Asst. Prof. Morteza Monemizadeh**
  Computer Science Institute
  Charles University
  Malostranske nam. 25, 118 00 Prague, Czech Republic
  Email: *monemi@ iuuk.mff.cuni.cz* & *m.monemizadeh@gmail.com*