Automatic Classification of Alzheimer's Disease from Structural MRI

Master's Thesis in Computer Science

Eivind Arvesen
for the Alzheimer's Disease Neuroimaging Initiative*

May 15, 2015
Halden, Norway

Østfold University College www.hiof.no

*Data used in preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: `http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf`

– Annoying! The same old story! When one has finished one's house one realizes that while doing so one has learnt unawares something one absolutely *had* to know before one – began to build. The everlasting pitiful 'too late!' – The melancholy of everything *finished*!...

— Friedrich Nietzsche, *Beyond Good and Evil*, subsection 277
(English translation by R. J. Hollingdale)

Abstract

Alzheimer's disesase, the world's most common form of dementia, is projected to boom in the coming years. The disease is very financially costly, with a poorly understood cause and no curative treatment. Early diagnosis of Alzheimer's and its prodromal stage is very important for possible delay of the disease, and there is thus a great deal of interest in the development of new methods for earlier detection. Structural irregularities of the brain are a sensitive feature of the disease (observable on MR images), and one of several known biological markers of the disease. Machine learning models, and maybe particularly techniques from the machine learning branch deep learning, might be able to learn features from high-dimensional data like structural MRI, and thereby enable automatic classification of Alzheimer's disease. This thesis first presents information about Alzheimer's disease relevant to the task at hand. It then reviews and summarizes some of the most important relevant machine learning research from later years. Based upon this information, we design and perform some experiments to investigate the possibility of automatic classifaction of Alzheimer's disease from structural MRI, using several methods of dimensional reduction and variations in the formulation of the learning task via different schemes of merging diagnostic groups, and performed with various machine learning approaches. We discover that decision trees trained on a dataset that had been dimensionally reduced via principal component analysis, with learning posed as a binary classification problem between Alzheimer's disease and all other diagnostic groups yielded the best results — comparable to related work. We hope that this thesis can provide a jumping off point for further research on this problem.

Keywords : Machine Learning, Computer Vision, Pattern Recognition, Alzheimer's Disease, Deep Learning, Artificial Neural Networks, Decision Trees

Acknowledgments

This thesis marks the conclusion of my master's degree at the Faculty of Computer Sciences, Østfold University College. It has been an interesting couple of years, and I have learned much during this challenging, but joyful time. My project has been very exciting to work on, and though the work itself was sometimes challenging, the many people that have helped me have been nothing but helpful and supportive.

I would like to thank my first thesis supervisor, Dr. Roland Olsson, for his support, helpful advice, friendly input and for a large extent of autonomy in this project.

Thanks also to my second supervisor, Dr. Øystein Haugen, for stepping in on short notice in the final month and providing insight regarding the structure and contents of this report.

I would also like to thank my friends and family, who have endured my partial absence, yet have been there for me.

Finally, I would like to thank ADNI and its collaborators for their great efforts, large amounts of work and willingness to share their data, without which this thesis and the original work described herein would not be possible.

Data collection and sharing for this project was funded by the Alzheimer's Disease Neuroimaging Initiative (ADNI) (National Institutes of Health Grant U01 AG024904) and DOD ADNI (Department of Defense award number W81XWH-12-2-0012). ADNI is funded by the National Institute on Aging, the National Institute of Biomedical Imaging and Bioengineering, and through generous contributions from the following: Alzheimer's Association; Alzheimer's Drug Discovery Foundation; BioClinica, Inc.; Biogen Idec Inc.; Bristol-Myers Squibb Company; Eisai Inc.; Elan Pharmaceuticals, Inc.; Eli Lilly and Company; F. Hoffmann-La Roche Ltd and its affiliated company Genentech, Inc.; GE Healthcare; Innogenetics, N.V.; IXICO Ltd.; Janssen Alzheimer Immunotherapy Research & Development, LLC.; Johnson & Johnson Pharmaceutical Research & Development LLC.; Medpace, Inc.; Merck & Co., Inc.; Meso Scale Diagnostics, LLC.; NeuroRx Research; Novartis Pharmaceuticals Corporation; Pfizer Inc.; Piramal Imaging; Servier; Synarc Inc.; and Takeda Pharmaceutical Company. The Canadian Institutes of Health Research is providing funds to support ADNI clinical sites in Canada. Private sector contributions are facilitated by the Foundation for the National Institutes of Health (www.fnih.org). The grantee organization is the Northern California Institute for Research and Education, and the study is coordinated by the Alzheimer's Disease Cooperative Study at the University of California, San Diego. ADNI data are disseminated by the Laboratory for Neuro Imaging at the

University of Southern California.

Prerequisites

Because the problems that this thesis deals with covers several aspects of computer science (as well as other fields), we will not cover every detail of all subjects mentioned. Additionally, machine learning is a very specialized (though broad) field itself, incorporating different aspects of and ideas from mathematics, physics, biology, etc. Graduated computer science students should easily be able to understand the contents of this thesis, though a deeper understanding would require some knowledge about probability, statistics, linear algebra and calculus from readers. Some basic familiarity with machine learning and related terms and techniques (supervised learning, classification, neural networks and decision trees in particular) is expected, although we give a shallow review when presenting newer relevant techniques.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Background and Motivation

Alzheimer's disease (AD) is the most common form of dementia, for which no cure or effective treatment is currently known. There is a projected "boom" of patients in the coming years, and there is a great deal of interest in early diagnosis of the disease, as this may lead to better treatment outcomes. Diagnosis of Alzheimer's Disease has traditionally relied mainly on clinical observation and cognitive evaluation. Recent studies, however, indicate that image analysis of neuroimaging scans may be a more reliable and sensitive approach. More attention has thus been shifting to finding biomarkers and applying machine learning techniques to perform automatic early detection of Alzheimer's. The Alzheimer's Disease Neuroimaging Initiative (ADNI), one of the world's leading research projects in the field, has contributed significantly to the further understanding of the disease by providing reliable clinical data for research purposes, including a labeled dataset of patients from different diagnostic groups consisting of magnetic resonance images. Recent research has yielded very good results on images from the ADNI dataset using deep learning methods and articifical neural networks. Artificial Neural Networks have been successfully applied to *Clinical Decision Support Systems*[1] as well as diagnostic assistance in the medical field, and there is great interest in leveraging machine learning technology for use in radiology, cardiology, oncology, etc. in order to develop more cost-effective and easy-to-use systems for supporting clinicians. This type of *Computer-Aided Diagnosis*[2] is especially interesting in the context of early diagnosis, which is very important in the case of Alzheimer's Disease. Stuctural Magnetic Resonance Imaging (MRI) seems to be an interesting diagnostic modality, as it is non-invasive, widely used, and as there are changes in brain morphology that are strongly associated with Alzheimer's disease. There also exists a lot of relevant data in the form of the ADNI standardized dataset. The problem at hand is also interesting from a machine learning point of view, as artificial neural networks and deep learning techniques in particular have proven to be well suited for dealing with high-dimentional data like that of brain scans.

[1] Interactive expert system, for assistance in clinical decision-making.

[2] The application of machine learning and image processing techniques in the field of radiology for the purpose of assisting clinicians in the interpretation of medical images.

1.2 Research Question and Methods

1.2.1 Research Question

This thesis aims to explore the possiblity of training a machine learning system to perform automatic classification of Alzheimer's disease from structural magnetic resonance images of subjects' brains. It evaluates the different approaches taken by measuring the generalization abilities of the trained models via their rate of misclassification on a previously held out validation portion of the dataset.

The work in this thesis focuses mainly on the evaluation of the generalization errors of the trained models, but it also looks at how different methods of dimensional reduction of the training data and the choice of machine learning technique affects the results yielded in attempting to produce a performant classifier. Different variations of every technique employed in the experiments are trained on every kind of dimensionally reduced data (e.g. variations in architecture in the case of neural networks). Additionally, each of these combinations are used as basis for several rounds of experiments, each of which pose the learning problem in a slightly different way based upon merging two or more of the diagnostic groups, i.e. output classes (e.g. classifying Normal/MCI/Alzheimer's or Alzheimer's/Other). Finally, the models trained on all the combinations of variations in the different parts of the experiments are compared and evaluated.

The research, prototyping and evaluation of this thesis focused on the following research question:

RQ 1 *How can a classifier be trained to differentiate between structural MR images of Alzheimer's Disease and other diagnostic groups?*
Secondary relevant research questions are:

- **RQ 1.1** *What kind of dimensional reduction is most suited for the problem at hand?*
- **RQ 1.2** *Which machine learning approach yields the best results on the dataset?*
- **RQ 1.3** *How does reformulating the multiclass classification problem as binary classification affect the models' performance?*
- **RQ 1.4** *How does the different results compare to similar research?*

1.2.2 Methodology

The field of machine learning is experimental in nature, often relying on empiricism in terms of explaining methodology and evaluation of results. This thesis is no exception to this, and it takes an experimental approach to the problem at hand. We deal with applied research.

In order to answer the previously stated research question, we need to experiment on several levels: method of dimensional reduction, formulation of the classification problem (i.e. different schemes for merging classes), and method of machine learning. We also need to combine the different variations in order to directly compare which combination yields the best reults, as well as how all the experimental alternatives affect performance.

As the dataset used in the work described in this thesis is high dimensional, dimensional reduction is needed. This is because the original instances in the dataset are 3D images with 8-bit voxel values representing density. In addition, as morphological changes in the brain (particularly in a few specific sites) is a sensitive feature of Alzheimer's, it would

be reasonable to assume that the entirety of the brain (as well as the air surrounding the head) is not needed for evaluation; rather, the relative structural proportions of the cerebral regions would in all probability suffice as features.

The learning problem is posed in several different ways, all supervised classification tasks, by merging the classes (i.e. diagnostic groups) of learning instances:

- Normal, MCI, Alzheimer
- Normal, Other
- Other, MCI
- Other, Alzheimers.

We do this because the binary classification problems resulting from the latter three merged variants of the set could reveal where the clearest distinctions are on the Normal-to-Alzheimer-spectrum. This may also affect general performance, in the case of neural networks in particular.

We also attempt to solve the learning problem by applying different machine learning techniques, as these types of models have different strengths and weaknesses. Tree-based approaches (e.g. decision trees and their ensemble variant, random forests) are one of the techniques that have achieved relatively high performance in research on automatic and assisted clinical and diagnostic systems. Neural networks, on the other hand, have proven to be extremely performant in later years, as deep learning has evolved as a subfield. Deep neural nets have shown unparallelled performance on several tasks, set state of the art on various computer vision problems, and proved to be useful in dimensional reduction.

The following criteria were used to answer the research questions. For each question respectively, we attempt to answer by:

RQ 1 Evaluating the results of the experiments and see which factors or combinations of these lead to the greatest performance. Additionally, any insight gained from performing the experiments might point to promising new directions.

- **RQ 1.1** Comparing performance between all experiment variants with respect to the method of dimensional reduction used
- **RQ 1.2** Evaluating what approach to machine learning that produced the single lowest misclassification rate across all experiments
- **RQ 1.3** Comparing performance between all experiments with respect to scheme of merged groups
- **RQ 1.4** Comparing the best performance measure yielded by experimentation to literature presenting recent research on the same dataset

The methods mentioned below have been used in the process:

- Identification of research objectives
- Literature
- Development of a data conversion tool
- Evaluation of machine learning tools
- Experimentation with different approaches to dimensional reduction, machine learning and formulation of the classification problem
- Training, testing and validating the different models on the converted dataset variants

1.3 Report Outline

- Chapter 2 deals with the background for the research presented in this thesis.It presents pertinent information regarding Alzheimer's disease, including biomarkers such as measurable difference in brain structure between diagnostic groups. Then, we give a brief explanation of the technology behind Magnetic Resonance Imaging and of the resulting images. After this, we describe both organizations related to the research of Alzheimer's, as well as organizations that facilitates research on the disease — such as the Alzheimer's Disease Neuroimaging Initiative, from which the dataset used in this thesis and the work related to it was procured. Afterwards, relevant fields of machine learning, and techniques thereof, are explored. This includes modern approaches like deep learning and related techniques that have proved effective and have gained popularity in industry during later years. We then deal with the same machine learning techniques applied in research, as well as approaches which have given the best results on problems regarding medical imaging and Alzheimer's Disease in particular.

- In chapter 3, we deal with the details of how the experiments were designed. The dataset that was used in this project is presented. Containing 2182 MR images of patients diagnosed with Alzheimer's Disease and Mild Cognitive Impairment as well as healthy controls, it is the largest available dataset of its kind. Different software, tools and implementations of machine learning techniques are discussed and evaluated. We then deal with the custom tools and other code and intermediary products that were produced during the course of this project.

- Chapter 4 gives details about the experiment runs. We present experiment results from decision trees and neural networks run on data reduced dimensionally via PCA, histogram and downscaling with the original 3 classes, as well as with three two-class merging schemes (resulting in binary classifiers).

- In Chapter 5, we discuss the findings from the previous chapter, and evaluates the effects of machine learning approach, method of dimensional reduction and class merging scheme on performance, relating the results to the research questions posed in this chapter. We also compare our results to performance achieved in related work.

- Chapter 6 sums up the work and the experimental findings and answers the research questions posed in this introduction, presents our conclusions, as well as points out some potential avenues of investigation.

Chapter 2

Background

2.1 Alzheimer's Disease

Alzheimer's disease (AD) is a neurodegenerative disease that causes problems with behavior, memory and thinking, and which is ultimately fatal. It is the most common form of dementia[1], accounting for 60 to 80 percent of all cases of dementia [9]. It is one of the most costly diseases to society in developed countries. Alzheimer's affected 26.6 million people worldwide in 2006, and is predicted to affect 1 in 85 people globally by 2050 [16]. The World Alzheimer Report of 2010 [68] projected that there would be 65.7 million people living with dementia by 2030, and 115.4 million by 2050. The same report also points out that almost two thirds of people affected by dementia live in low- and middle-income countries, which are expected to see the largest increase in patients in the coming years [68, p. 4], as the regions are developing rapidly. This will be challenging for several reasons, one of which is that dementia patients in these countries largely rely on *informal care*[2] — a practice that may prove exceedingly difficult to maintain as the older segment of these populations increase in numbers and disease prevealence climbs [68]. Dementia has an enormous societal cost at present, accounting for 1.01% of the total sum of worldwide Gross National Products [68, p. 24]. It is thought that this problem will become exacerbated in the coming years, with an estimated 85% worldwide increase in societal cost by 2030 [68, p. 38], assuming that no potential background factors (e.g. macroeconomic factors, dementia incidence and prevalence of dementia, availability and effectivity of treatment) change.

While some of its symptoms may appear somewhat similar to typical signs of advanced aging, it is important to note that AD (and indeed dementia in general) is **not** a normal part of aging. As the disease progresses over time, dementia symptoms gradually worsen. At present, there is no curative treatment for Alzheimer's disease; The goal is rather to slow the progression of the disease, improve symptoms, address behavioral problems and improve quality of life. However, current treatments can temporarily decelerate the development of dementia symptoms if the disease is diagnosed in an early stage. While more effective treatments and ultimately prevention or even a cure is a very important (long term) goal, earlier diagnosis may give patients comparatively better treatment outcomes.

The precise cause for Alzheimer's is still unknown, except for the few cases of identifiable

[1] A syndrome that impairs cognition and behavior, which can be caused by a number of disorders [68, p. 8].

[2] Unpaid help from friends and family.

genetic differences. Current research indicates, however, that it is associatied with *neuritic plaques* and *neurofibrillary tangles* (both of which are further detailed in Section 2.1.3) in the brain [60]. While Amyloid beta, the protein that neuritic plaques are made up of, is known to be strongly implicated in the development of the disease, it is still debated whether or not it is a causative factor, as many believe it to be. It is, however, generally acknowledged to be a marker of the disease.

The current trend is that there is an ever-increasing ability to see the disease and track its progress before symptoms occur. Later years have brought about advances in research, most importantly identification of biomarkers[3] (notably brain imaging techniques) that allow identification and visualization of AD-related processes months, years and even decades before clinical symptoms appear.

The biomarkers for AD can be separated into early biomarkers, which typically measure amyloid deposition in the brain (e.g. PET imaging, CSF amyloid), and later biomarkers, which typically measure neurodegeneration (e.g. structural MRI, FDG PET, CSF tau).

Brain image scans are often used to exclude other causes for symptoms, but can also be an indicator of whether or not Alzheimer's disease is present. In their 1996 article, Laakso et al. states that "In clinical practice, the diagnosis of Alzheimer's disease is based on typical features of the disease and exclusion of other conditions causing dementia" [41].

The only known way to know for absolutely certain whether a person was afflicted by the disease is a post-mortem brain tissue examination. However, both neurofibrillary tangles and neuritic plaques seems to play a role in the development and evolution of Alzheimer's disease(see subsection 2.1.3, p. 9).

Although a great deal of research has been done on Alzheimer's, there is still a need for an early (non-invasive) diagnostic tool for the disease.

2.1.1 Mild Cognitive Impairment

Mild Cognitive Impairment (MCI) is thought to be an early stage Alzheimer's disease, though it is somewhat disputed whether MCI corresponds to a different diagnostic stage, or to a prodromal[4] stage of AD.

In MCI-patients, brain changes have already been going on for quite some time, and symptoms are only just beginning to appear. It does not (yet) result in problems that are severe enough to make a major impact on day-to-day functioning — which would be considered dementia. MCI is further detailed in Section 2.1.5.

2.1.2 Risk Factors

While the exact cause of the disease remains unclear, certain risk factors are clearly associated with development of Alzheimer's. All of the following risk factors are discussed more in depth in their appropriate contexts in Section 2.1.5.

Age

Age is very clearly associatied with Alzheimer's. After the age of 65, risk of Alzheimer's disease doubles every five years, and is at almost 50 percent after age of 85 [8].

[3] A biomarker, or biological marker, is an indicator, measurement or substance of a biological state. Biomarkers may exist before clinical symptoms arise [4].

[4] Symptoms that can be seen as precursors to the disease.

Family History

Close family member with Alzheimer's disease are more likely to develop it themselves. If more than one family member has the illness the risk increases. Either or both of hereditary and environmental factors may play a role when diseases tend to run in families [8].

Genetics

Genetics may also play an important role in the development of AD. There are two categories of genes that influence Alzheimer's development:

- Risk genes (increase in likelihood)
- Deterministic genes (direct cause of disease)

If an autosomal dominant form of Alzheimer's (i.e. familial Alzheimer's disease) is present (i.e. mutations in APP, PSEN1, PSEN2), development of MCI is might be an early symptom that indicates the start of Alzheimer's Dementia. A large majority of those affected by this develop early onset Alzheimer's Dementia. However, there is variable certainty about the evolutional progression from MCI to AD dementia in these individuals. The presence of one or two $\epsilon 4$ alleles[5] in the apolipoprotein E (APOE) gene is also broadly accepted as an increasing risk for late-onset AD dementia. The $\epsilon 2$ allele, on the other hand, decreases this risk.

2.1.3 Pathophysiology

Alzheimer's is a complex disease, and its precise mechanism of disease is not known.

Biochemistry

AD has been identified as a protein misfolding disease due to the accumulation of abnormally folded amyloid beta (β-amyloid) protein in the brains of Alzheimer's patients [9, 60]. Amyloid beta is a short peptide that is an abnormal byproduct of the protein amyloid precursor protein (APP), whose function is unclear but thought to be involved in neuronal development. The amyloid fragments, which are sticky, clump together and forms plaques (these are also called neuritic plaques). These plaques block signaling (i.e. communication) between cells, which seems to trigger immune reactions leading to programmed cell death of the disabled neurons.

Amyloid plaques are "a hallmark feature of a pathological diagnosis of AD..." [7], and are reflected in biomarkers that can detect and quantify the accumulation of the amyloid in the brain. The protein can be measured directly in cerebrospinal fluid (CSF) and plasma. Positron emission tomography (PET) can also be used. The clinical criteria for mild cognitive impariment due to Alzheimer's states the following of the amyloid correlation:

"Current evidence suggests that markers of amyloid pathology (i.e., CSF and PET) precede evidence of neuronal injury. This does not prove that Aβ is the initiating factor for the disease. However, it does suggest that these different categories of biomarkers seem to provide different sorts of information about the progress of disease in the brain" [7].

[5] An allele is a variant form of a gene.

AD is also considered a tauopathy[6] due to abnormal aggregation of the tau protein, a microtubule[7]-associated protein expressed in neurons that normally acts to stabilize microtubules in the cell cystoskeleton[8]. Usually tau makes microtubules straight, which makes molecules able to pass freely. In AD, however, the protein collapses into twisted strands (i.e. tangles), which the tubes disintegrate, obstructing nutrients from reaching nerve cells — leading to cell death.

The MCI criteria argues that changes in tau and phosphorylated-tau could reflect general damage to neurons and synapses.

The accumulation of β-amyloid plaques and neurofibrillary tangles is in turn though to lead to loss of neurons and synapses (breaking down the brain's structure), which results in "memory impairment and other cognitive problems" [64].

However, plaques and tangles have also been found in the brains of older individuals that did not meet the criteria for Alzheimer's disease during life [52, 60]. For example, allocortical[9] NFT and neocortical[10] NP (less commonly) in brains of elderly subjects without dementia was shown in neuropathological examinations. These changes are rare in the brains of cognitively intact younger subjects. These AD-related changes in some older subjects may suggest that they are representative of "pathological aging" [22] or preclinical AD [52, 13] (which means that the disease is pathologically present, but is not associated with clear, clinically detectable cognitive changes) - since AD and normal aging currently cannot be distinguished clearly from eachother.

Morris et al. performed a study in 1996 [52], examining the relationships of cognitively normal aging, very mild Alzheimer type dementia and the presence of neocortical senile plaques. Their results suggested that neuritic plaques may not be part of the normal aging process, but rather represent presymptomatic or unrecognized early symptomatic Alzheimer's disease. They found this to be "consistent with the hypothesis that beta-amyloid deposition is an initial pathogenetic event in the development of AD" [52].

Dickson et al. [22], on the other hand, find strong support for the hypothesis that cerebral amyloid deposition (plaques) is not necessarily assosiated with clinically apparent cognitive dysfunction. Rather, they argue that additional factors, such as neuronal or synaptic loss or widespread cystoskeletal abberations (tangles) are necessary for dementia in AD.

Berg et al. [13] found that the brains of all participants in their group of persons with dementia (except for one, which had a non-Alzheimer's form of dementia) had substantial densities of neocortical senile plaques regardless of dementia severity. The brains of the subjects in their control group, on the other hand, had very few senile plaques. They also found moderate correlations between duration and severity of dementia and certain (both gross and microscopic) neuropathological lesions. Densities of neocortical neurofibrillary tangles were related to degree of dementia. They conclude that plaque densities differentiate very old subjects with AD from nondemented controls, but point out that there

[6] A class of neurodegenerative diseases associated with the pathological aggregation of tau protein in the human brain.

[7] Networked tubes used to transport nutrients between cells.

[8] Networks of fibers that provides support for and helps maintain the shape of cells.

[9] The allocortex is one of two types of cerebral cortex, the outermost layered structure of brain tissue. The hippocampus, which are involved in integrating information from short-term to long-term memory (amongst other things), is usually described as belonging to the allocortex.

[10] The neocortex is the largest part of the cerebral cortex, covering the two cerebral hemispheres. It is involved in higher functions (e.g. sensory perception, spatial resoning, language) in humans. The neocortex is the most recently evolved part of the cerebral cortex.

is a need for more postmortem studies of older persons who are free of dementia. They also found densities of neocortical neurofibrillary tangles to be most closely related to the degree and duration of dementia.

In 2004, Tiraboschi, Hansen, Thald and Corey-Bloom performed a study [60] that sought to determine the relation of neuritic plaques and neurofibrillary tangles to the development and evolution of Alzheimer's disease by studying an autopsy series of 102 patients with dementia and pathologically confirmed Alzheimer's and 29 normal control subjects. Cases of Alzheimer's were stratified according to their last Mini-Mental State Examination before death. neuritic plaques and neurofibrillary tangles were enumerated in the midfrontal, inferior parietal, superior temporal, hippocampal, or entorhinal cortices using thioflavin-S preparations.

They found that 87% of the normal control group had allocortical neurofibrillary tangles, while only 37% displayed neocortical neuritic plaques. 19% of the control group showed hippocampal plaques. None of the controls exhibited neocortical tangles (except for one case with a single tangle).

Among Alzheimer's patients, however, neocortical tangles were detected in less than 10%, and tangles were even absent in nearly 50% of those with mild Alzheimer's disease at death. Plaques were found in all patients with Alzheimer's. Although neurofibrillary tangles and neuritic plaques density increase with severity of dementia, significant differences consistently emerged for neurofibrillary tangles alone.

When comparing the normal control group and patients with mild Alzheimer's disease, the only significant difference found was that of the number of neuritic plaques. The authors therefore found neurofibrillary tangles' sensitivity as a marker of Alzheimer's disease lower than that of neuritic plaques.

They concluded that deterioration in Alzheimer's disease appears to be driven by neuritic plaques and neurofibrillary tangles at different stages of disease. The significant increase in plaques (but not in tangles) even in patients with mild Alzheimer's at death compared to the normal control group suggests that only neuritic plaques are associated with the earliest symptoms of the disease.

Neuropathology

The symptoms that occurs during development of Alzheimer's disease are caused by the changes in the brain (i.e. structural irregularities), which is a sensitive feature of the disease. The hippocampi of persons affected by Alzheimer's disease are atrophied early on in the course of the disease - a phenomenon that can be reliably detected by volumetric (i.e. structural) MRI for diagnostic purposes [41]. This loss of neurons and synapses from the disease lead to clearly visible differences in brain tissue, as illustrated in Figures 2.1 and 2.2.

The disease is regularly characterized by degeneration of cerebral cortex and hippocampus, leading to cortical atrophy in temportal, frontal and parietal areas [66]. Ventricles are enlarged compared with healthy individuals, as can be seen in Figure 2.2.

Microscopic changes in the brain begin long before the first signs of memory loss, with one recent study finding presence of mediotemporal lesions up to 5.6 years before the clinical diagnosis of Alzheimer's [14]. The same study found that there was an absence of atrophy in the frontal lobes, which was consistent with studies indicating that they are affected closer to the time at which diagnosis is made. The authors had findings consistent with studies indicating that the frontal lobes are affected close to the time of diagnosis.

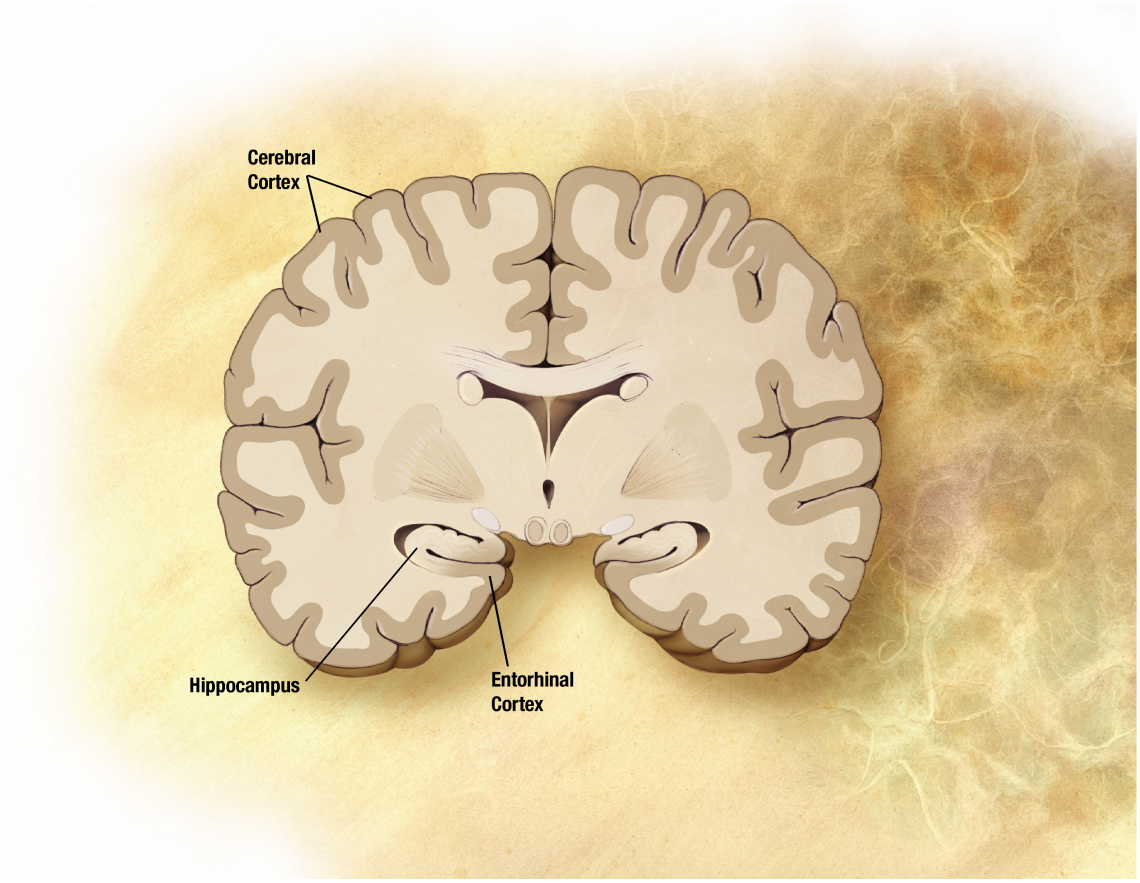

Figure 2.1: Diagram of a normal brain.

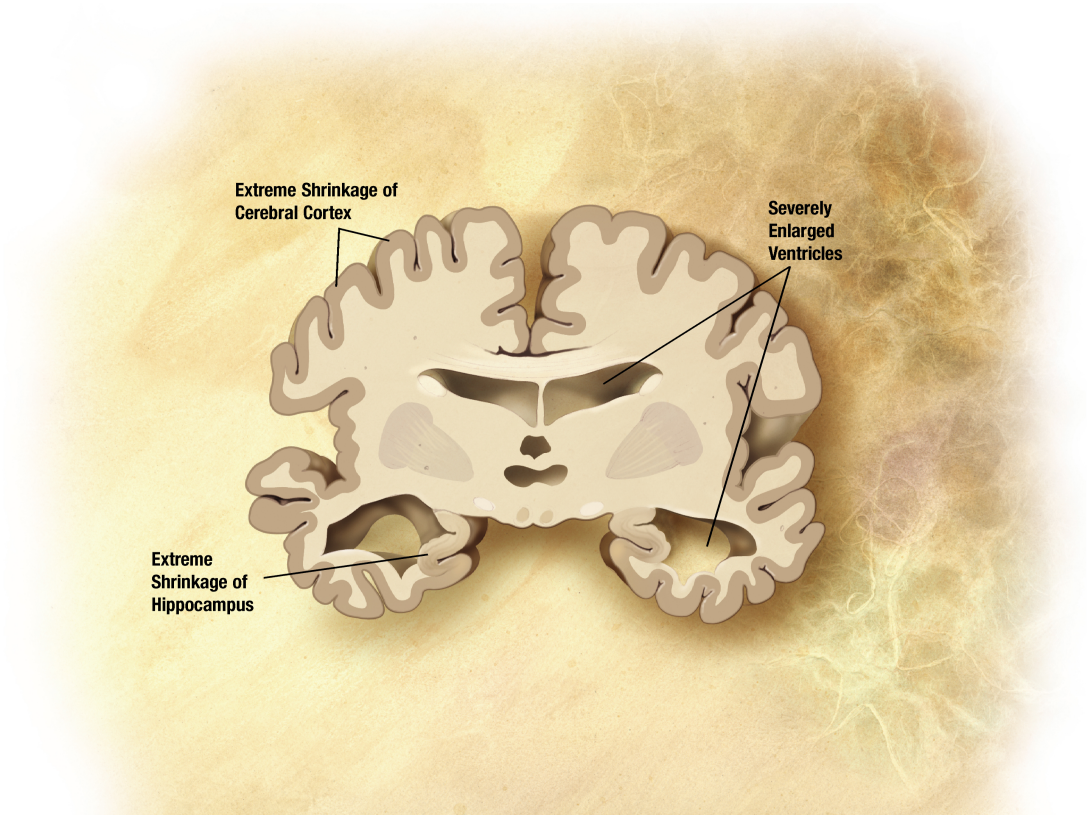

Figure 2.2: Diagram of a the brain of a person with Alzheimer's Disease.

They also argue that volume reduction of the posterior cingulate cortex may be involved later in the course of the disease.

Magnetic resonance imaging (MRI) enables researchers and clinicians to assess these structural changes in the brain assosicated with Alzheimer's disease non-invasively. There are also other modalities[11]such as the imaging technique Pittsburgh Compound B PET (PiB PET), which clearly show sites and shapes of beta amyloid deposits in the brain. This technique is more invasive, however, as it requires a contrast agent — a radioactive sugar — which is absorbed in the brain of the patient. Additionally, it is much newer, and not as available as MRI.

In their 1995 study [41], Laakso et al. investigated the specificity of hippocampal volumes between 59 patients with mild to moderate Alzheimer's Disease (AD), 9 patients with vasculular dementia (VaD), 12 patients with idiopathic parkinson's disease wiout dementia (PD), 8 patients with parkinson's and dementia (PDD), and 34 elderly control subjects using a 1.5-T MR imager. They found that all patient groups had significantly smaller hippocampal volumes (on both sides) than the control group, and that the absolute volumes were even smaller in the group of patients with parkinson's and dementia than in the alzheimer's group. They postulated that "hippocampal atrophy does not seem to be a specific phenomenon of dementia in AD but also occurs in VaD and PDD, and even in PD when no dementia is present" [41]. They did, however, also point out that coexistence of Alzheimer's pathology in their Parkinson's and vasculular dementia patients cannot be ruled out. The researchers concluded that hippocampal atrophy is a sensitive feature of Alzheimer's disease, but specificity of this atrophy seems to limit its use in clinical practice.

Biomarkers

As Alzheimer's disease progresses over time, biomarker magnitudes reach abnormal levels in a predictable order (see Figure 2.3).

Figure 2.3 shows biomarkers as indicators of dementia, the curves indicate changes caused by five studied biomarkers [4] (in chronological order):

1) **Amyloid beta** imaging detected in CSF and PET amyloid imaging

2) Neurodegeneration detected by rise of **CSF tau species** and synaptic dysfunction, measured via FDG-PET[12]

3) **Brain atrophy** and neuron loss measured with MRI (most notably in hippocampus, caudate nucleus, and medial temporal lobe)

4) **Memory loss** measured by cognitive assessment

5) **General cognitive decline** measured by cognitive assessment

The first three biomarkers in the list can be observed prior to diagnosis of dementia, while the last two are "the classic indicators of dementia diagnosis" [4].

In order to be able to offer appropriate therapy (where available), biomarkers are also being incorporated into the diagnostic framework, although these are primarily meant for

[11]A modality, in this context, is the mode in which biomarkers are recorded, i.e. MRI, PET or CSF.

[12]PET scan performed with a short-lived radioactive tracer isotope that is chemically incorporated into the biologically active fluorodeoxyglucose (FDG) molecule.

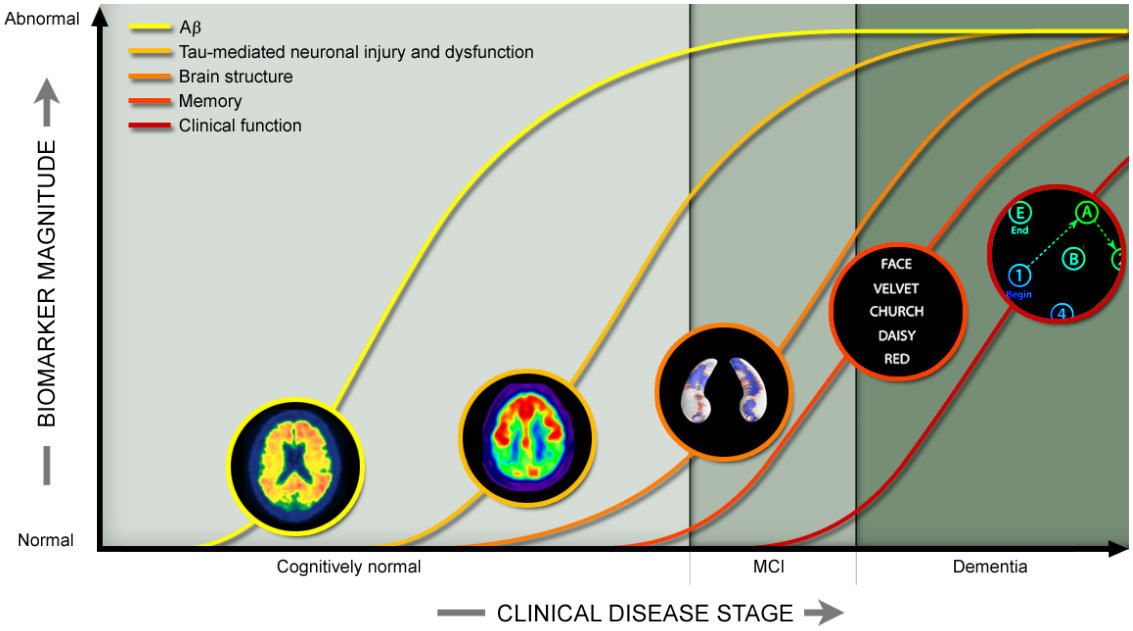

Figure 2.3: Alzheimer's biomarkers over the course of the disease.

use in research. These include biomarkers that directly reflect the pathology of AD, such as beta-amyloid protein (Aβ) and tau; and biomarkers that provide less direct or nonspecific evidence of the disease by tracking indices of neuronal injury, which are somewhat specific for AD because of the regional pattern of abnormalities. Whilst primarily associated with Alzheimer's, these biomarkers are also seen in other disorders. When the two are observed in the same subject there is a very strong reason to suspect Alzheimer's.

2.1.4 Symptoms

Symptoms of Alzheimer's disease typically start with difficulty remembering newly learned information, leading to increasingly severe symptoms such as "disorientation, mood and behavior changes; deepening confusion about events, time and place; unfounded suspicions about family, friends and professional caregivers; more serious memory loss and behavior changes; and difficulty speaking, swallowing and walking" [9] as it advances. Symptoms may in some cases be more apparent to family or friends than to the affected individual.

Destructive pairing of plaques and tangles starts in hippocampus (responsible for forming memories). Short term memory loss is usually the first symptom of AD. The proteins then progressively invade other parts of brain, creating unique changes, signal various stages of disease. At front of brain: destroy ability to process logical thought. Then decreased emotion control, resulting in erratic mood changes. Then (at top of brain) cause paranoia and hallucinations. Then, in the brain's rear, the proteins work together to erase the mind's deepest memories. Finally control centers (governing heartrate and breathing) are overpowered, resulting in death. Still not well understood. Immensely destructive nature of disease.

2.1.5 Diagnosis

New diagnostic criteria were issued for Mild Cognitive Impairment due to Alzheimer's disease, as well as Dementia due to Alzheimer's disease in 2011, by expert international workgroups convened by the Alzheimer's Association and the National Institute on Aging (an agency of the U.S. National Institutes of Health - NIH). This was due to the growing consensus in the field that the disease includes a phase when affected individuals experience a gradual worsening of cognitive capabilities that results from the accumulation of Alzheimer's disease pathology in the brain. The patient is diagnosed with AD dementia when symptoms are sufficiently pronounced (i.e. with great cognitive impairment). The writers of the criteria notes that "as AD is a slow, progressive disorder, with no fixed events that define its onset, it is particularly challenging for clinicians to identify transition points for individual patients" [7], and that the point in time at which a patient transitions from, for example, the asymptomatic phase to the symptomatic predementia phase - or from the symptomatic predementia phase to dementia onset, for that matter - is thus difficult to identify. There is also greater diagnostic uncertainty earlier in the disease process. They also point out that is important to incorporate this continuum of impairment into clinical and research practice. Alzheimer's cannot not currently be diagnosed by a laboratory test, but requires a clinician's judgment. Etiologies[13] in addition to AD pathophysiological processes may coexist in individuals that meet the criteria (i.e. other biological, psychological and/or sociocultural factors).

Mild Cognitive Impairment due to Alzheimer's Disease

Mild cognitive impairment due to Alzheimer's disease (MCI) is a syndrome defined by "clinical, cognitive and functional criteria" [7]. It is the symptomatic predementia phase of Alzheimer's disease. The 2011 revised diagnostic criteria "assume that it is possible to identify those with AD pathophysiological processes as the likely *primary* cause of their progressive cognitive dysfunction" [7], and lists the following criteria (for the clinical and cognitive syndrome):

- Concern regarding a change in cognition
- Impairment in one or more cognitive domains
- Preservation of independence in functional abilities
- Not demented

If a patient meets the criteria, the medical practitioner has to assess whether (and to what degree) there is objective evidence of cognitive decline from reports by the patient, and/or other informants. He or she must also performing cognitive testing. Individuals affected by MCI typically score "1 to 1.5 standard deviations below the mean for their age and education matched peers on culturally appropriate normative data..." [7]. Since other cognitive domains can be impaired in MCI patients, clinicians mus also examine other domains than memory, such as: executive functions, language, visuospatial skills and attentional control. There are a number of validated clinical neuropsychological tests available for this purpose. If these tests are not feasible, there are also a variety of simple informal techniques that can be used for the same purpose. The authors also point out

[13] The cause(s) and origin of an illness/disease.

that it must be recognized that "atypical clinical presentations of AD may arise, such as the visual variant of AD (involving posterior cortical atrophy) or the language variant (sometimes called logopenic aphasia), and these clinical profiles are also consistent with MCI due to AD" [7].

This diagnosis usually entails mild functional impairment for complex tasks, but performing daily activities is typically not problematic, and the person should not meet criteria for dementia. It is important to obtain longitudinal assesment of cognition whenever possible, since evidence of progressive decline in cognitive abilities provides additional evidence that the mild cognitive impairment is due to Alzheimer's and would strenghten the accuracy of the diagnosis. Once the clinical and cognitive syndrome is determined to be consistent with that associated with Alzheimer's, but the subject is not demented, the likely primary cause must be determined. The diagnostic criteria states that normally "this information is derived from further historical information and ancillary testing (e.g., neuroimaging, laboratory studies, and neuropsychological assessment) that may prove informative" [7]. This is because, in order to meet the core clinical criteria, other systemic or brain diseases that could account for the decline in cognition must be ruled out. This is also important because the pathological features of some other disorders can co-exist with Alzheimer's disease, especially among patients at an advanced age.

Genetics may also play an important role in the development. If an autosomal dominant[14] form of Alzheimer's (i.e. familial Alzheimer's disease) is present (i.e. mutations in APP, PSEN1, PSEN2), development of MCI is most likely an early symptom that might indicate the start of Alzheimer's Dementia. A large majority of those affected by this develop early onset Alzheimer's Dementia. However, there is variable certainty about the evolutional progression from MCI to AD dementia in these individuals. The presence of one or two $\epsilon 4$ alleles in the apolipoprotein E (APOE) gene is also broadly accepted as an increasing risk for late-onset AD dementia. The $\epsilon 2$ allele, on the other hand, decreases this risk.

AD also causes a wide range of structural and functional changes in the brain. These have diagnostic and prognostic value in dementia and MCI, and appear to reflect damage to neurons and synapses. The diagnostic criteria states that "Many of these changes have topographic specificity for the neural damage or dysfunction that occurs in AD" [7]. These downstream neuronal injuries may be measured by structural and/or functional measures, including MRI, PET and SPECT[15] imaging.

Some structural and functional patterns are characteristics of AD as well. Loss of hippocampal volume, for example, can be detected on MRI, and reduced glucose metabolism or perfusion in temporoparietal cortex may be seen with PET or SPECT scanning. The criteria notes that "Although these biomarkers have been associated with the neuropathology of AD, regional atrophy, global atrophy, and regional hypometabolism and hypoperfusion are not specific for AD. These measures appear to provide evidence about the stage or severity of disease that may not be provided by Aβ biomarkers" [7]. There is also the issue of biochemical events characterizing AD, including "oxidative stress (e.g., isoprostanes) and inflammation (e.g., cytokines). CSF, plasma..." [7], the imaging markers of which could provide information suggestive of underlying pathology. Additional work in this area

[14] A way that a disease, condition or trait can be inherited, which requires only an abnormal gene from one parent.

[15] Single-photon emission computed tomography, a nucleare medicine tomographic imaging technique using gamma rays. Provides true 3D information.

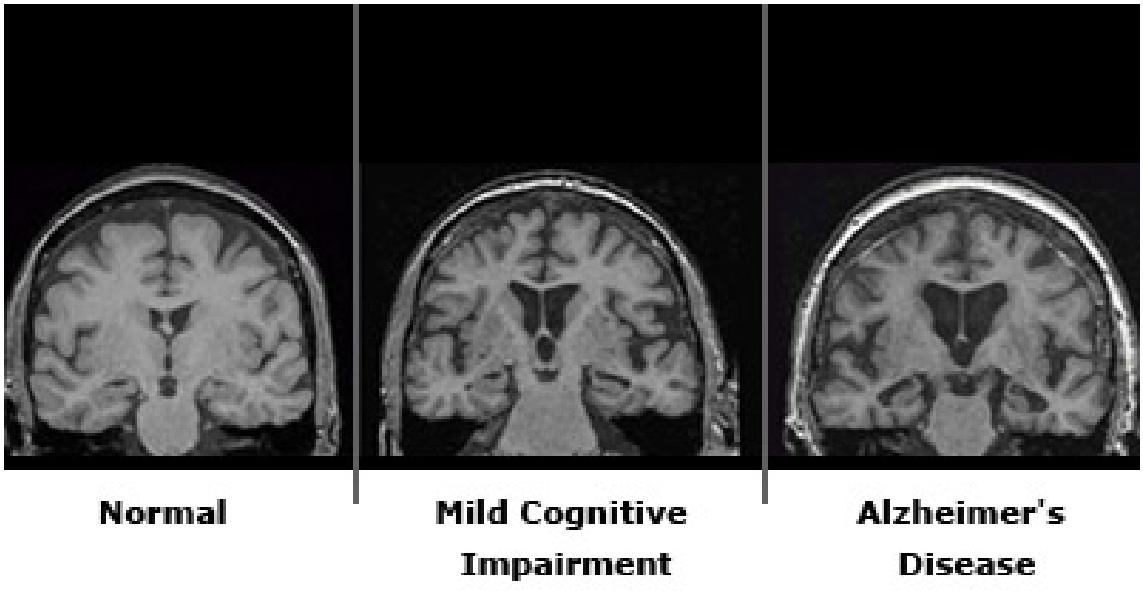

Figure 2.4: MR images of normal, MCI- and AD-affected brains.

needed to further assess usefulness of markers. Biomarkers of neuronal injury "(which refers to CSF tau/p-tau, hippocampal, or medial temporal lobe atrophy on MRI, and temporoparietal/precuneus hypometabolism or hypoperfusion on PET or SPECT)" [7] is perhaps one of the most interesting features of Alzheimer's disease for development of tools for early (non-invasive) clinical screening. This can be seen in Figure 2.4. The standardization of biomarkers of AD is currently limited, differing between locales. There is also limited experience with cut-points for diagnosis and different accessibility of biomarkers in different settings [7].

Dementia due to Alzheimer's Disease

The revised criteria for Alzheimer's dementia set forth in 2011 is for use by both "general healthcare providers without access to neuropsychological testing, advanced imaging, and cerebrospinal fluid measures, and specialized investigators involved in research or in clinical trial studies who would have these tools available" [48]. It includes criteria for all-cause dementia and for AD dementia. It also integrates biomarker evidence, for probable and possible AD dementia for use in research settings. Biomarker evidence is "expected to enhance the pathophysiological specificity of the diagnosis of AD dementia" [48], although much work lies ahead for validating the biomarker diagnosis of AD dementia. The criteria this revision supercedes have been quite successful, surviving for over 27 years, have been reliable for diagnosis of probable AD, and have had a sensitivity[16] of 81% and specificity[17] of 70% across more than a dozen clinical pathological studies. One of the features that required revision was that there was no inclusion of the results of MRI or PET imaging, nor cerebrospinal fluid assays in decision-making. Another was the implication that memory impairment is always the primary cognitive deficit in all patients with AD dementia. The

[16] A statistical measure of the proportion of correctly identified positives.
[17] A statistical measure of the proportion of correctly identified negatives.

following are the must meet criteria for all-cause dementia. Patients are diagnosed when there are cognitive or behavioral (neuropsychiatric) symptoms that:

1. Interfere with the ability to function at work or at usual activities; and
2. Represent a decline from previous levels of functioning and performing; and
3. Are not explained by delirium or major psychiatric disorder;
4. Cognitive impairment is detected and diagnosed through a combination of (1) history-taking from the patient and a knowledgeable informant and (2) an objective cognitive assessment, either a "bedside" mental status examination or neuropsychological testing. Neuropsychological testing should be performed when the routine history and bedside mental status examination cannot provide a confident diagnosis.
5. The cognitive or behavioral impairment involves a minimum of two of the following domains:
    a) Impaired ability to acquire and remember new information—symptoms include: repetitive questions or conversations, misplacing personal belongings, forgetting events or appointments, getting lost on a familiar route.
    b) Impaired reasoning and handling of complex tasks, poor judgment—symptoms include: poor understanding of safety risks, inability to manage finances, poor decision-making ability, inability to plan complex or sequential activities.
    c) Impaired visuospatial abilities—symptoms include: inability to recognize faces or common objects or to find objects in direct view despite good acuity, inability to operate simple implements, or orient clothing to the body.
    d) Impaired language functions (speaking, reading, writing)—symptoms include: difficulty thinking of common words while speaking, hesitations; speech, spelling, and writing errors.
    e) Changes in personality, behavior, or comportment—symptoms include: uncharacteristic mood fluctuations such as agitation, impaired motivation, initiative, apathy, loss of drive, social withdrawal, decreased interest in previous activities, loss of empathy, compulsive or obsessive behaviors, socially unacceptable behaviors.

The differentiation of dementia from MCI rests on whether or not there is significant interference in the ability to function at work or in usual daily activities. The authors of the diagnostic guidelines also propose terminology for classifying individuals with dementia caused by AD:

1. Probable AD dementia
2. Possible AD dementia
3. Probable or possible AD dementia with evidence of the AD pathophysiological process (intended for research purposes)

The diagnosis of probable AD dementia includes meeting the criteria for dementia (mentioned earlier), as well as the following characteristics: Insidious onset, symptoms have

gradual onset over months to years; clear-cut history of worsening of cognition by report or observation; cognitive deficits in the form of either amnestic presentation (including impairment in learning and recall of recently learned information and evidence of dysfunction in at least one other cognitive domain) or nonamnesic presentation, such as language presentation (word finding), visuospatial presentation (deficits in spatial cognition) and executive dysfunction (impaired reasjoning, judgment and problem solving). The diagnosis should not be applied should a differential diagnosis yield candidate conditions. Increased levels of certainty can be achieved for this diagnosis from evidence of progressive cognitive decline, formal neuropsychological evaluation or standardized mental status examinations. If there is evidence of a causative genetic mutation (in APP, PSEN1 or PSEN2) the certainty that the condition is caused by AD pathology is to be considered increased. Possible AD dementia, on the other hand, needs to have an atypical course - meeting the core clinical criteria, but either having a sudden onset or demonstrating insufficient historical detail or objective cognitive documentation of progressive decline, or an etiologically mixed presentation. An etiologically mixed presentation meets all core clinical criteria, but has evidence of candidate condition(s).

The diagnosis of probable AD dementia with evidence of the AD pathophysiological process (for use in research) includes biomarkers for the pathophysiological process of AD. These include low CSF Aβ_{42}, positive PET amyloid imaging (biomarkers of brain amyloid-beta, Aβ); and elevated CSF tau (both total tau and phosphorylated tau/p-tau) and "disproportionate atrophy on structural magnetic resonance imaging in me-dial, basal, and lateral temporal lobe, and medial parietal cortex" [48]. Although total tau and p-tau are treated equivalently, p-tau may have more specificity for AD than other dementing diseases. Biomarker evidence may increase the certainty that the basis of the clinical dementia syndrome is the AD pathophysiological process in persons who meet the core clinical criteria for probable AD dementia. The writers of the criteria do not, however, advocate the use of biomarker tests for routine diagnostic purposes at present because the core clinical criteria provide very good diagnostic accuracy and utility in most patients; More research needs to be done to ensure that criteria that include use of biomarkers have been appropriately designed; The limited Standardization of biomarkers between locales; and access to biomarkers is limited to varying degrees in community settings. They suggest that the use of biomarkers to enhance certainty may be useful in "investigational studies, clinical trials, and as optional clinical tools for use where available and when deemed appropriate by the clinician" [48].

There is also the category "possible AD dementia with evidence of the AD pathophysiological process" [48], which is for persons who meet clinical criteria for a non-AD dementia, but who either meet the neuropathological criteria for AD or have biomarker evidence of the AD pathophysiological process. The authors indicate the usage of both, since the diagnosis of possible AD dementia with evidence of the AD pathophysiological process does not preclude the possiblity of another pathophysiological condition's presence. The authors also give a few considerations related to the incorporation of biomarkers into the critera, namely: Since AD dementia is a part of a continuum of clnical and biological phenomena, there will invariably be some cases where ambiguous or indeterminate results will be obtained from imaging biomarkers (which can be interpreted in both a qualitative or quantitative manner). They write that

"Although sophisticated quantitative and objective image analysis methods do exist, at present, accepted standards for quantitative analysis of AD imaging tests are lacking. Standard clinical practice in diagnostic imaging is qualitative in nature. Therefore, quantification of imaging biomarkers must rely on local laboratory specific standards. [...] Quantitative analytic techniques are, and will continue to be in evolution for some time. Therefore, practical use of biomarkers must follow best-practice guidelines within laboratory-specific contexts, until standardization has been fully accomplished" [48].

The authors also notes that "A sequence of events has been described with Aβ pathophysiological processes becoming abnormal first and downstream neuronal injury biomarkers becoming abnormal later... " [48]. This could imply a hierarchical ranking of Aβ biomarkers over downstream neuronal injury biomarkers for diagnostic purposes. The reliability of such a hierarchical scheme has, however, not been sufficiently well established for use in AD dementia at this time. They argue that it is inevitable that different combinations of test results can occur, given the number of different AD biomarkers. Individual cases might, for example, be encountered with a positive Aβ and negative neuronal injury biomarker, or a positive FDG-PET and negative tau measure, and so on. "At present, the data are insufficient to recommend a scheme that arbitrates among all different biomarker combinations. Further studies are needed to prioritize biomarkers and to determine their value and validity in practice and research settings. [48]

2.1.6 Treatment

There is currently no known cure for Alzheimer's. The current goal of treatments is to slow the progression of the disease and manage its symptoms. Although this is very difficult, it is possible to a certain extent if it is diagnosed relatively early on. Treatments mainly manage symptoms, such as cognitive and psychological issues, and behavioral problems; environmental adjustment to enable subjects to better perform daily activities; and support caregivers, such as family members.

2.2 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) is unlike conventional X-ray and CT[18] scans, in that it does not rely on ionizing radiation. The imaging technique can generate three dimentional images at any depth and in any orienation. It is far superior to other imaging tools in providing non-invasive images at high resolution, and is the diagnostic tool of choice for soft tissue. Although it is not normally necessary, MRI patients are sometimes injected with a contrast agent to make abnormalities (such as tumors) appear clearer. The imaging technique works by essentially mapping the position of water molecules, which exists in different densities in different types of tissue.

2.2.1 Technology

In most units, MRI works by using radio waves to redirect the axes of spinning protons, which are the nuclei of hydrogen atoms (from water in body), whilst inside a magnetic

[18]Computed Tomography, x-ray CT being the most common form of which.

field generated by passing an electric current through wire coils (electromagnets - creating a magnetic field by conducting electricity.) [53].

The main magnet in the machine (surrounding the patient) is a superconducting electromagnet: an electromagnet that has no resistance to electricity and no heat generation (due to cooling by liquid helium).

The patient is positioned so that the part of the body that is of interest is in the middle, or isocenter, of the magnetic field.

2.2.2 Physics

In the nucleus of every hydrogen atom in the water in a person's body there is a positively-charged proton that spins (or precesses) around an axis. This makes it generate its own magnetic field. When these atoms are placed in a more powerful magnetic field the protons' axes realign with the more powerful field: around half of them face in the direction of the field, around the other half in the opposite direction. A few more atoms line up with the magnetic field in the low-energy configuration (north-south north-south) than in the (north-north south-south) configuration, which requires more energy. "A few" means ca. 9 out of 2 million protons in a 1.5 tesla[19] (the field strength which is most common in clinical practice.) magnet powered MRI [53]. The few "leftover" protons (not cancelled out by a proton lined up in the opposite direction) are the protons that the MRI scanner will use. Whilst inside a more powerful magnetic field, these hydrogen nuclei "precess about the magnetic field direction like gyroscopes" [1], a behavior termed Larmor precession.

The Larmor Frequency

A radio frequency coil is placed inside the machine along with the subject, and sends radio frequency pulses (RF pulses) at the desired location. The pulses are precisely timed, adjusted to a specific range of frequencies at which Hydrogen protons precess (using the Larmor frequency).

"The frequency of Larmor precession is proportional to the applied magnetic field strength as defined by the *Larmor frequency*,

$$\omega_0 = \gamma B_0 \tag{2.1}$$

where γ is the *gyromagnetic ratio* and B_0 is the strength of the applied magnetic field. The gyromagnetic ratio is a nuclei specific constant. For hydrogen, γ=42.6MHz/Tesla" [1].

The "leftover" protons absorb the energy of the RF pulses, which causes them to flip on their axes (still in line with magnetic field, but now in opposite direction: the high-energy configuration). In other words: unmatched protons flip as the RF pulses are turned on [53]. When the RF pulse stops, protons release the absorbed energy and return to their previous alignment. In doing so, they emit a signal back to the coil.

Signal Weighting

Signal weighting will affect the contrast of the image between tissues. T1-weigthed images are especially well suited for brain scans, and shows significant differences between grey matter and white matter.

[19] Unit of magnetic field strength, defined by the International System of Units.

2.2.3 Imaging

The returned signal is then turned into an electric current that is digitized by the scanner, which generates a series of images, each of which shows a thin slice of the body. The resulting images have various shades of gray that reflect different densities, based on the fact that areas with lower water content have fewer hydrogen protons that emit signals back to the RF coils. Different machines display this differently (depending on the signal weighting), but using T1 weighting will produce images where dense bone, air, and other matter containing few hydrogen protons will be fairly dark; fat will be light and so on [53]. The voxels can be one of 255 shades of grey depending on signal strength, where 0 equals black and 255 equals white.

2.2.4 Slicing

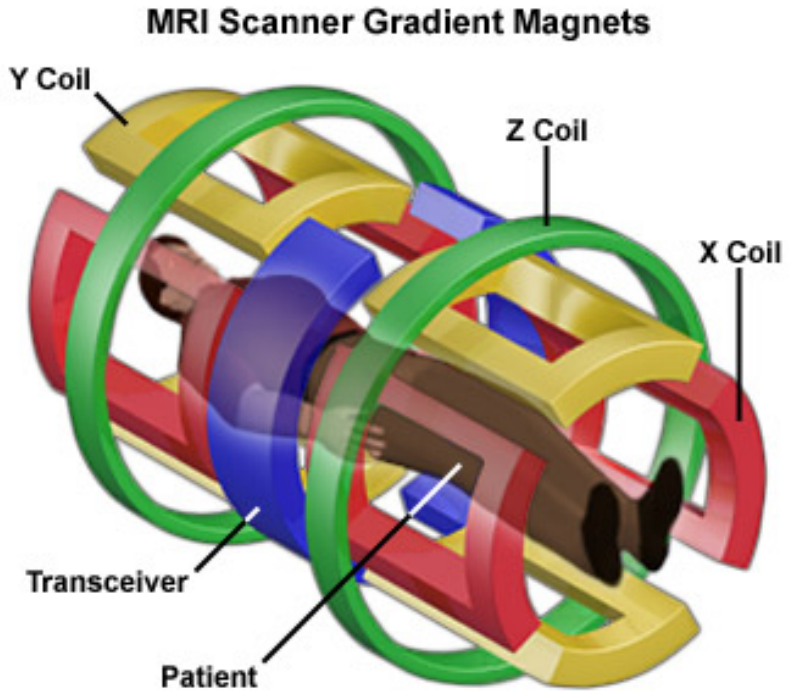

Figure 2.5: MRI scanner coils.

Inside the MRI scanner are also an additional three magnets, the gradient magnets (illustrated in Figure 2.5), which are called X, Y and Z. These are far less powerful than the main magnet, and each of them are oriented along a different plane of the patient's body. The gradient magnets "modify the magnetic field at very particular points and work in conjunction with the RF pulses to produce the scanner's picture by encoding the spatial distribution of the water protons..." [53] in the patient's body. The gradient magnets are what allows the scanner to image the body in slices when rapidly turned on and off.

The combinations of X-Y (transverse or axial planes) slice top to bottom, X-Z (coronal planes) slice lenghtwise (from front to back), and Y-Z (sagittal planes) slice lenghtwise (from side to side). The resulting slices can be seen illustrated in Figure 2.6.

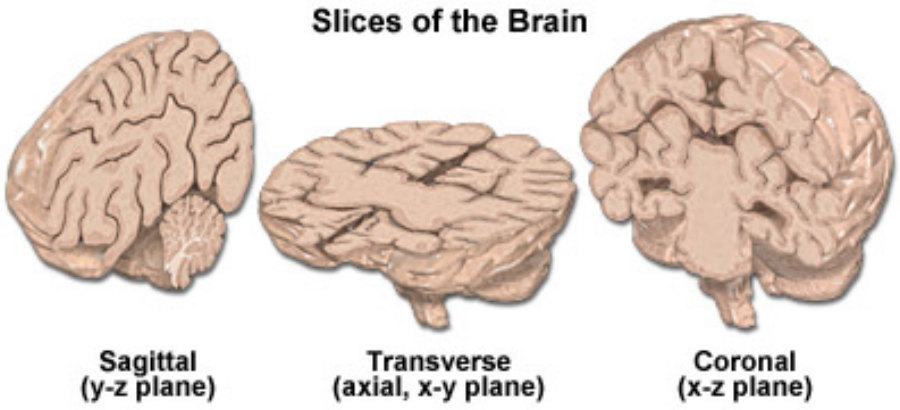

Figure 2.6: Slices of the brain.

2.3 Organizations

This subsection deals with organizations that work towards a better understanding of and research on Alzheimer's Disease, some of which produce and make available clinical data for research use (including the dataset described in Chapter 3.2), and without whom the work described in this thesis — not to mention all the other research it has facilitated — could not have been performed.

2.3.1 The Alzheimer's Association

The Alzheimer's Association is a non-profit American voluntary health organization which focuses on care, support and research for Alzheimer's disease. They host the Alzheimer's Association International Conference on Alzheimer's Disease (ICAD) as a part of their research focus to update knowledge about dementia. ICAD is the largest group of international leaders in Alzheimer research and care. They also have the official publication *Alzheimer's & Dementia* (published by Elsevier), presenting the latest original, peer-reviewed, basic and clinical research advances in the field, including early detection, prevention and treatment.

2.3.2 The Alzheimer's Disease Neuroimaging Initiative

The Alzheimer's Disease Neuroimaging Initiative (ADNI) is an ongoing, longitudinal collaborative study that aims to "develop clinical, imaging, genetic, and biochemical biomarkers for the early detection and tracking of Alzheimer's disease" [3]. The six-year ADNI1 study began in 2004, including 400 subjects with MCI, 200 subjects with early AD and 200 elderly control subjects. The study was extended with ADNI GO from 2009 to 2011, adding 200 participants that were identified as having early MCI in order to examine biomarkers in an earlier stage of disease. The worldwide project provides researchers with reliable clinical data on Alzheimer's Disease. They aim to better understand the pathology of Alzheimer's using biomarkers, hopefully enabling earlier diagnosis of Alzheimer's disease; to provide clinical trial data in order to support new research methods pertaining to intervention, prevention and treatment of the disease; and continue and improve the sharing of their database. During ADNI1 the aim was to develop more accurate diagnostic

methods to detect Alzheimer's in the earliest stage and define the pathology using biological markers. The initiative successfully developed early stage detective methods for Alzheimer's (including CSF biomarkers, β-amyloid 42 and tau), standardized methods for clinical tests (including MRI, PET and CSF biomarkers) and established their database of datasets containing brain scan images.

In a review [64] of the ADNI project that summarizes the results of all papers published as of February 2011[20], Weiner et al. list the following major accomplishments:

1. The development of standardized methods for clinical, magnetic resonance imaging (MRI) and positron emission tomography (PET) and cerebrospinal fluid (CSF) biomarkers in a multi-center setting.

2. Elucidation of the patterns and rates of change of imaging and CSF biomarker measurements in control, MCI and AD patients. CSF biomarkers are consistent with disease trajectories predicted by β amyloid (Aβ) cascade and tau mediated neurodegeneration hypotheses for AD while brain atrophy and hypometabolism levels show predicted patterns but exhibit differing rates of change depending on region and disease severity.

3. The assessment of alternative methods of diagnostic categorization. Currently, the best classifiers combine optimum features from multiple modalities including MRI, FDG-PET, CSF biomarkers and clinical tests.

4. The development of methods for the early detection of AD. CSF biomarkers, Aβ42 and tau as well as amyloid PET may reflect the earliest steps in AD pathology in mildly or even non-symptomatic subjects and are leading candidates for the detection of AD in its preclinical stages.

5. The improvement of clinical trial efficiency through the identification of subjects most likely to undergo imminent future clinical decline and the use of more sensitive outcome measures to reduce sample sizes. Baseline cognitive and/or MRI measures generally predicted future decline better than other modalities whereas MRI measures of change were shown to be the most efficient outcome measures.

6. The confirmation of the AD risk loci CLU, CR1 and PICALM and the identification of novel candidate risk loci.

7. Worldwide impact through the establishment of ADNI-like programs in Europe, Asia and Australia.

8. Understanding the biology and pathobiology of normal aging, MCI and AD through integration of ADNI biomarker data with clinical data from ADNI to stimulate research that will resolve controversies about competing hypotheses on the etiopathogenesis of AD thereby advancing efforts to find disease modifying drugs for AD.

9. The establishment of infrastructure to allow sharing of all raw and processed data without embargo to interested scientific investigators throughout the world.

ADNI is an important step towards developing improved diagnostic methods, and the possible development of an effective treatments to slow the progression, and ultimately prevent AD.

[20] Approximately 200 papers were published between the initiative's inception and February 2011.

2.3.3 The Laboratory of Neuro Imaging

The Laboratory of Neuro Imaging (LONI) is a research laboratory that is part of the Institute for Neuroimaging and Informatics at the University of Southern California (as of 2013), and is one of the leading neurological research centers in the U.S. They work mainly with [2]:

- conducting research and building population-based and disease-specific digital brain atlases;
- helping in the training of research investigators; and
- fostering communication of medical information.

They have also developed software to aid in understanding the relationships between structural and functional aspects of the brain. LONI currently focusing on researching the construction of detailed brain atlases. They are also addressing "... the problem of comparing data across individuals as well as across modalities ..." [2], and focusing on statistical manipulation of the anatomic and functional data sets' geometry, as well as visualization techniques for efficient communication of results. LONI is heavily involved in a variety of collaborations both nationally and internationally, for some of which LONI acts as a hub. The latter deals with the application of "novel image analysis approaches to investigate brain structure and function in health and disease" [2]. Some of their brain mapping research has dealt specifically with Alzheimer's disease, and LONI hosts the data from ADNI in their data archive.

2.4 Machine Learning

Machine learning is an inherently multidisciplinary field, drawing on research from a diversity of fields, such as statistics, artificial intelligence, neurobiology and philosophy. In his book "Machine Learning", Tom Mitchell defined the field of machine learning as one concerned with "the question of how to construct computer programs that automatically improve with experience" [49, p. xv]. In the same book, he later defines learning (in the context of machine learning) in the following way: "A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its experience at tasks in T, as measured by P, improves with experience E" [49, p. 2]. For instance, a program that learns to play checkers could improve its performance measured by its ability to win with respect to the class of tasks involved with playing checkers through experience obtained by playing games of checkers against itself [49, p. 5].

Witten, Frank and Hall gave an operational definition of learning in their book 'Data Mining : Practical Machine Learning Tools and Techinques', stating that "Things learn when they change their behavior in a way that makes them perform better in the future" [69, p. 7], effectively tying learning to *performance* rather than *knowledge*. However, they did see problems with this definition in relation to computers. They pointed out that learning is a slippery concept, and mentioned difficulties in distingushing training from learning, as well as the philosophical difficulty of defining intention and purpose.

A machine learning system consists, generally, of three parts; training data, a model, and a training (i.e. learning) algorithm. The learning can then be explained as fitting the model's parameters to the training data supplied. This is not always the case, though,

e.g. in the case of decision trees (which are more of a generative approach). In order to measure the trained model's real performance, it is then applied to test data, a portion of the data previously withheld. The goal of the learning process is for the system to be able to generalize from its experience (with the training data), performing well on previously unseen instances of data.

Machine learning can (roughly) be divided into three learning paradigms, namely *supervised* and *unsupervised learning*, and *reinforcement learning*. Supervised learning deals with labelled example data, where inputs are coupled with desired output values. This can be viewed as a parallel to the psychological concept of *concept learning*[21]. In unsupervised learning, on the other hand, the learning examples are unlabeled (i.e. no known output). This results in there not being any error or reward signal with which to evaluate a potential solution. Reinforcement learning deals with attempting to reach a specified goal by performing an action in a dynamic environment in order to maximize a reward, without being explicitly told if the learner is approaching its goal.

Machine learning employs the same methods as data mining[22] (of which machine learning provides the technical basis) in unsupervised learning, or in preprocessing steps.

There is a wide varity of machine learning models, all of which make different prior assumptions[23] about the possible input-output mappings or data distribution, in supervised and unsupervised learning respectively. The models make these different assumptions by necessity, since the problem they deal with is *ill-posed*[24], and the training data is thus not sufficient for the models to find the solution by themselves. For instance, a mapping may not exist, or there may not be enough information to reconstruct it, or there may be an unavoidable presence of noise that makes a perfectly fit model useless in practice. These sets of assumptions that the learning algorithms make in order to make learning possible is called *inductive bias*. Mitchell defined a learning algorithm's inductive bias as "a set of assertions that, together with the training examples, deductively entail subsequent predictions made by the learner" [49, p. 322]. Inductive bias is thus important because it describes how the learner generalizes beyond the observed training examples.

In choosing the type of training experience from which the system will learn, one deals with three important attributes:

1. **Direct or indirect training experience:** In the case of the checkers example, learning from individual board states and correct moves (direct), or move sequences and final outcomes (indirect). The latter leads to the problem of credit assignment, which entails determining how much credit each move in the sequence deserves for the outcome.

2. **The learner's degree of control over the sequence of learning examples:** For instance, given informative board states with correct move for each vs. self-selected board states where the learner asks for the correct move upon encountering

[21] Learning attributes, or formulating concepts, that the learner can use to classify given examples as belonging to a category based on a set of relevant features.

[22] Discovering unknown properties in data, as opposed to predicting properties based on what was learned from training data.

[23] Based on a priori information — prior knowledge, or *domain knowledge*.

[24] A problem that does not fit Jacques Hadamard's definition of *well-posed problems*, which are mathematical models of physical phenomena that should have the following properties: A solution exists; The solution is unique; The solution's behavior changes continuously with the initial conditions [65].

particularly confusing states vs. complete control over states and classifications (e.g. playing against itself, no teacher involvement).

3. **How representative training examples are of the distribution of the final test examples over which the final system performance is measured.** While learning is most reliable when these distributions are similar (most machine learning theory rests on the assumption that they are identical), they will often differ somewhat in real-world applications because of practical limitations.

Since they can have a significant impact on the performance of the learner, these are all important parts of designing a learning stystem.

In machine learning, it is often useful to, as Mitchell says, "reduce the problem of improving performance P at task T to the problem of learning some particular *target function*[25]" [49, p. 7] such as choosing which move the learner will pick next (from a set of legal moves, given the state of the board).

The process of learning the target function is often called *function approximation* because we often expect the algorithm to learn only some approximation to the target function [49, p. 8].

The learning algorithm attempts to minimize some measure of error (i.e. loss function), such as mean squared error or least mean squares. It does this by trying to choose the weights that best fit the set of training examples, i.e. minimizing the loss function between the observed training examples (and other prior constraints or knowledge) and the predicted values.

When choosing which algorithm to use for a given problem (and a given target function), one should take into concideration the relationship between size of hypotheses space, completeness of it, number of training examples available, prior knowledge held by learner and confidence one can have that a hypothesis that is consistent with training data will correctly generalize to unseen examples.

The inductive learning hypothesis is defined by Mitchell as "Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples" [49, p. 23].

In this report, I will focus mainly on supervised learning using (deep) artificial neural networks. Additionally, I will also cover some some unsupervised learning techniques for preprocessing purposes, as well as regularization methods.

2.4.1 Artificial Neural Networks

Artificial neural networks (ANNs) are models that provide robust, general methods for learning. In part inspired by complex webs of interconnected neurons that make up biological learning systems (i.e. animal central nervous systems, particularly brains), ANNs consist of a set of interconnected units, each unit taking a number of real-valued inputs (e.g. output from other units) and producing a real-valued output (which may in turn be used as input to other units). There are, however, complex facets of biological neural systems that ANNs do not model, as well as features of ANN systems that are known to

[25] The type of knowledge that will be learned and how it will be used by the performance program [49, p. 7], i.e. what to learn.

be inconsistent with biological systems — ANN units that output a single constant value, while "biological neurons output a complex time series of spikes" [49, p. 82], for instance.

Unless otherwise specified, in this report the term "Neural Networks" refers to artificial feedforward neural networks (illustrated in Figure 2.7): ANNs in which the connections between the units do not form a directed cycle[26]. Feedforward neural networks are thus multilayered networks, feeding information from input nodes (through any eventual hidden nodes) to output nodes.

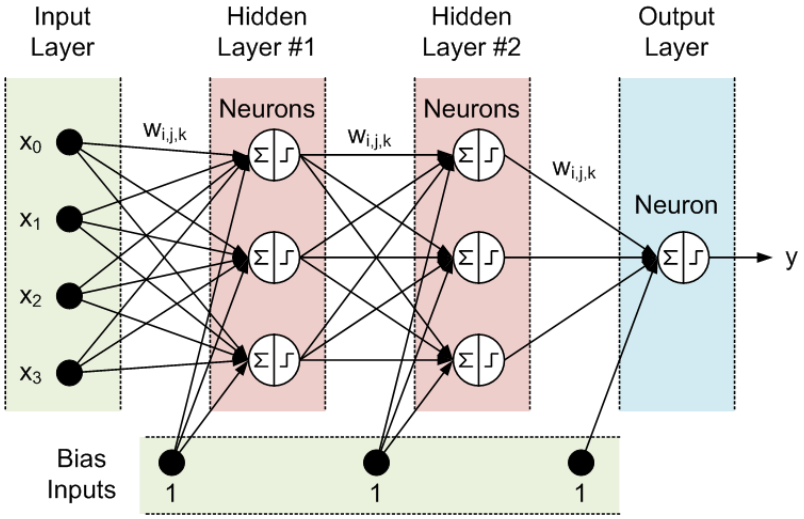

Figure 2.7: A feedforward neural network.

One very simple type of feedforward network is the *perceptron* (illustrated in Figure 2.8), which is a special kind of artificial neuron[27] that performs binary classification. A perceptron takes a real-valued vector as its input and calculates a linear combination of the inputs, computing a weighted sum of the inputs and their corresponding *weights* — their counterparts in a vector of real-valued constants — upon which a bias is applied. It is then passed to an *activation function* (further detailed in Section 2.4.6), which in perceptrons' case is the sign function ($sgn()$). The sign function works in the following way: If the result is greater than some threshold, the perceptron outputs a 1 — Otherwise, it outputs -1. There also exists alternative designs to perceptrons for these primitive units (i.e. neurons), such as linear and sigmoid units.

Single layer perceptrons are capable only of learning linearly separable patterns, making them capable of representing the primitive boolean functions AND, OR, NAND and NOR [49, p. 87]. Other boolean functions, such as XOR, cannot be represented by a single perceptron. Multilayer perceptrons[28], on the other hand, are able to learn linearly nonseparable patterns by virtue of their nonlinear activation function (e.g. the hyperbolic tangent function $tanh()$, the logistic function or softmax — the former two of which are sigmoids). A network of perceptrons need only be two levels deep in order to be able to represent *every* boolean function, although the network in question would be very big and employing a large number of neurons.

[26] As opposed to recurrent neural networks, where the units do just that.
[27] A primitive unit used to build articifical neural networks.
[28] Network of perceptrons.

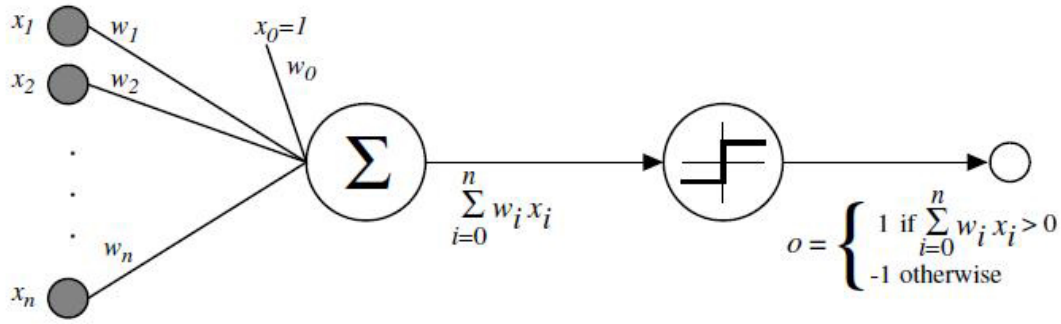

Figure 2.8: A perceptron.

The ANN training process consists of adjusting the network's weights to minimize a cost function (e.g. mean square error, cross entropy, negative log likelihood). Multilayer perceptrons, for instance, are trained with the standard *backpropagation* algorithm, updating the weights via *gradient descent* in order to best fit a training set of input-output pairs. While there are many training strategies for neural networks, the most popular strategies are mostly gradient-based methods (some of which are further detailed in Section 2.4.4).

Artificial neural networks are well suited for learning problems that deal with noisy, complex sensor data (e.g. audio and images) as they are robust to noise in training data. They have been successfully applied to a wealth of problems, and are currently very popular as well as effective for certain types of problems in a diversity of domains such as of autonomous vehicle control, game-playing, facial recognition, natural language processing, medical diagnosis, data mining and automated trading systems. Neural networks have recently seen a renewed interest, which has been maintained in later years in large part due to the advent of deep learning.

2.4.2 Deep Learning

Deep learning is a subfield of machine learning that is concerned with multiple levels of non-linear operations. It deals with high dimentional data (e.g. images) and how to extract meaningful, learning representation from it and model this in hierarchies of increasingly higher levels of abstractions. They work on features of increasingly higher levels through successive layers of transformation, essentially building levels of concepts eachother, with each layer dealing with a more complex task, given its inputs from the previous layer. This also gives deep models the ability to perform *feature learning* (further discussed in the form of unsupervised pre-training in Section 2.4.3), i.e. automatically extract useful features from input. These automatically learned features are in many cases better than hand-designed features. Another important concept is that of distributed representations, which essentialy means that concepts are represented by patterns of activity over several neurons, and that each neuron takes part in the representation of more than one concept. These concepts are illustrated in Figure 2.9.

Architectures of this type is inspired by biological models [57] of the primate visual cortex, particularly the apparent processing of information "through stages of transformation and representation", building more complex processing stages upon eachother [10, p. 6]. Deep learning encompasses learning techniques such as Deep Neural Networks (DNNs), Deep Belief Nets (DBNs) and Deep AutoEncoders (DAE), to name a few, with some of

the most successful methods involving ANNs.

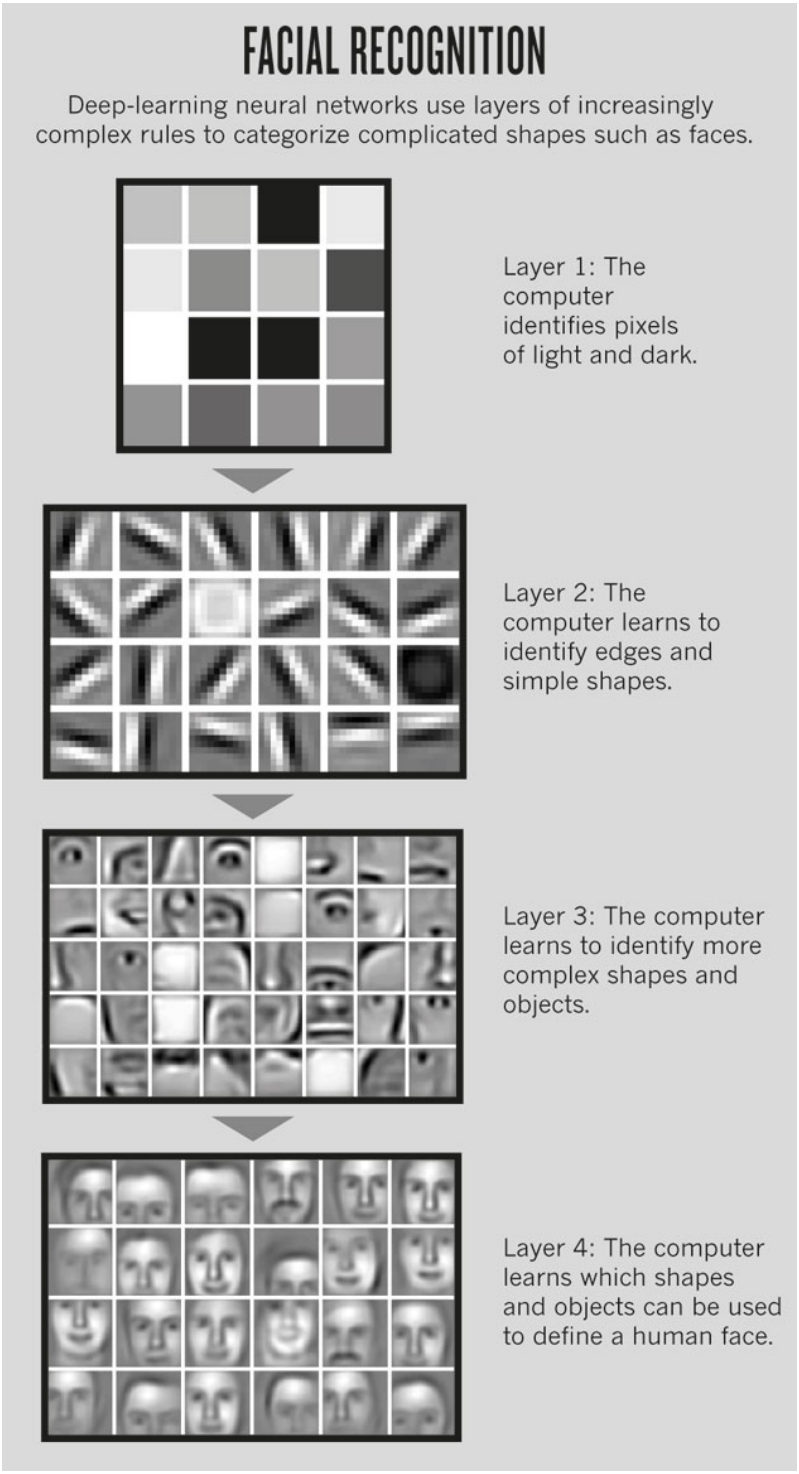

Figure 2.9: Illustration of how DNNs learn distributed representations in hierarchical concepts.

Deep learning approaches have shown very good performance in later years, especially with respect to computer vision problems. As of October 2014, deep neural networks

based on GPU-implementations using some form of dropout regularization (detailed in Section 2.4.5) is the state-of-the-art on several standardized image classification datasets. Deep neural networks has yielded remarkable reults on important standardized benchmark datasets (which are further detailed in Section 2.5.1) in recent years, in some cases comparable to human performance — and in others [19] beating humans altogether. In the latter case, the network in question was a GPU-based implementation of a deep convolutional neural network performing supervised learning. This particular case is further detailed in Section 2.5.2. Machine learning research have produced techniques during recent years that have enabled deep learning techniques to be practical to apply, such as sparse initialization, pre-training and regularization techniques like Dropout. All of these are further detailed later in this chapter. As well as in machine learning research, advances in hardware (GPUs in particular) have made these methods feasible, drastically reducing running times of training algorithms. In fact, Ciresan, Meier, Gambardella and Schmidhuber found training big and deep neural networks (plain multilayer perceptrons) using online backpropagation on GPUs to be more than 40 times faster than on CPUs in 2010 [18].

Dominance of Deep Learning

Comparisons to biological models suggested that deep architectures may be necessary in order to learn complicated functions that can represent high-level abstractions (e.g. computer vision) [10, p. 6]. Deep architectures were, however, much more difficult to train than shallow ones. They tended to get stuck in apparent local minimum, and purely supervised training consistently got worse for deep architectures, according to a 2010 article [23]. The problems could be explained by vanishing gradient, local optima and pathological curvature problems (which known more serious in deep architectures). Because of the "poor training and generalization errors" that were usually obtained from them [10, p. 31], deep models were largely ignored until unsupervised pre-training [34] (detailed in Section 2.4.3) was discovered. Since then, different deep architectures have been proposed and successfully applied in many areas, beating state of the art in certain applications.

Since then, more approaches and alternatives to pre-training appeared [62, 42], many of which are very similar to Hintons original technique in that a DNN is first pre-trained by unsupervised algorithm, then fine tune by other classical supervised learning methods (e.g. backpropagation). Besides pre-training, Sutskever, Martens, Dahl and Hinton found that classical methods like stochastic gradient descent (first-order method) may perform decently on deep learning [59], provided that better initialization schemes (like "sparse initialization" [47] and carefully tuned momentum methods) are used.

The last few years have seen a decrease in image processing for computer vision problems, as the models used in many cases makes it unnecessary — as made evident by its lack of appearance in state-of-the-art research. A technique that has seen increased usage the last few years is the augmentation [26, 37, 18] of image datasets used for learning in order to "inflate" the dataset (i.e. increase the instances of training data). This is achieved via deformations and transformations.

During the last decade or so, many of the the pioneers and leading researchers of deep learning have been hired or acqui-hired[29] by huge technology companies. This includes:

- Geoffrey Hinton (University of Toronto) and his grad students working for Google;

[29] Hired as a result of their company or employer being acquired.

- Yann LeCun (New York University) working for Facebook;
- Andrew Ng (Stanford University) working for Baidu.

Also, in early 2014 Google acquired DeepMind [58], an english artificial intelligence company, for over $500 million. DeepMind published a paper in 2013, detailing their research that dealt with learning a deep convolutional neural network to play seven Atari 2600 games using reinforcement learning on raw pixel input. Their model outperformed all previous approaches on six of the games, and beat a human expert on three of them [50].

As these types of models achieve impressive performance on several pattern-analysis tasks, they have been applied to real world challenges by major players in the field of technology, such as Apple's Siri and the Google search-by-voice service [31]. In their 2014 article, Lee, Xie, Gallagher, Zhang and Tu identifies the sharing of experiences and open sourcing of software by researchers in the deep learning field as factors that have helped both the adoption and the advancements of the techniques used [42].

In later years, GPU-based implementations of deep feedforward neural networks (often convolutional NNs) have been the state-of-the-art on many problems, notably in the field of computer vision [70].

2.4.3 Architectures

Convolutional Neural Networks

A convolutional neural network (CNN) is a kind of a feedforward artificial neural network, that is popularly used for image recognition purposes. CNNs are inspired by the mammelian visual cortex [10, p. 43], and designed to work with little to no preprocessing. They can operate directly on raw input like pixels, which effectively automates feature selection. In these models, individual neurons are tiled to respond to overlapping regions in the visual field, and in each layer neurons are organized in groups to look at only a portion of the input image. Convolutional neural networks were the only kind of deep network that was practical to train before the use of unsupervised pre-training [10, p. 43], and are still among the best performing models for pattern recognition.

Deep convolutional networks have achieved great results on computer vision problems in recent years (see Section 2.4.2 and 2.5.2 for details).

Deep Neural Networks

Deep neural networks (DNNs) are articifical neural networks with multiple hidden layers. Although "depth" refers the many hidden layers of the models, they are generally not just deeper, but larger in all dimensions. This entails more neurons, more layers and more connections. This, in turn, means that more hardware or computational power is a prerequisite for these models. Google Brain[30], for instance, had a million neurons and a billion connections, and ran on a cluster of 16'000 computers, according to a 2012 article in the New York Times [46]. As deep models they can, if learned well, represent input through multi-level feature representation, where features in higher layer model higher level abstractions in input.

[30] A Google deep learning project which famously learned to recognize cats from watching youtube videos.

The idea of deep networks is not a new one, but DNNs has been impractical to use until recently because of difficulties with the training algorithm. After the appearence of unsupervised pre-training, however, they have rapidly become one of the most popular machine learning methods in use.

The concept of *pre-training* has been an important development, which addresses the problem of how to set the initial weights in a network. If the weights are too small it will take a very long time before anything interesting happens. If the weights are too large, on the other hand, it raises the likelihood of becoming trapped in a local optimum prematurely. Pre-training can be said to prime systems to follow a more fruitful path, by making it learn generally useful feature detectors.

Unsupervised pre-training is an important strategy in deep learning, allowing one to train very deep NNs very effectively by performing unsupervised training on one layer at a time before supervised training [12]. The strategy consists of two stages:

- Pre-training: Pre-training one layer at a time in a greedy way. Training building block model for each layer, and stacking them.
- Fine-tuning: Training with backpropagation to "fine-tune" the network.

Different models can be used as building blocks in the pre-training phase, such as Restricted Boltzmann Machines (RBMs), as Hinton originally used when discovering the strategy [34], and various Autoencoders (AEs) [10, p. 32]. The act of performing supervised training on the network after pre-training is called "fine-tuning", since it is much easier to adjust the weights of a pre-trained deep net than to train it with randomly initialized weights. Although this technique was very important factor in DNNs becoming viable models, it is decreasingly used on state-of-the-art computer vision problems, for which convolutional neural networks are often used [40]. It may, however, still be useful in cases where there is not a large amount of labeled training data available, as it will reduce the amount of computational power/time needed to reach good performance (at least in non-convolutional networks).

Related Models

The following models are mostly relevant in the sense that they are used as building-block models in the unsupervised pre-training process introduced by Hinton [34] (e.g. Restricted Boltzmann machines, varous Autoencoders).

Boltzmann Machines

Boltzmann Machines are a type of stochastic recurrent neural networks that are theoretically useful, but have as of yet not proven practical in their unconstrained form (i.e. with intra-level connectivity). It consists of symmetrically coupled stochastic binary units. Restricting visible-to-visible and hidden-to-hidden connectivity results in a Restricted Boltzmann Machine (compared to a general Boltzmann machine in Figure 2.10), which can be learned using contrastive divergence [32].

Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) are Boltzmann Machine with no connections between neurons within the same layer — the neurons must form a bipartite graph. Thanks

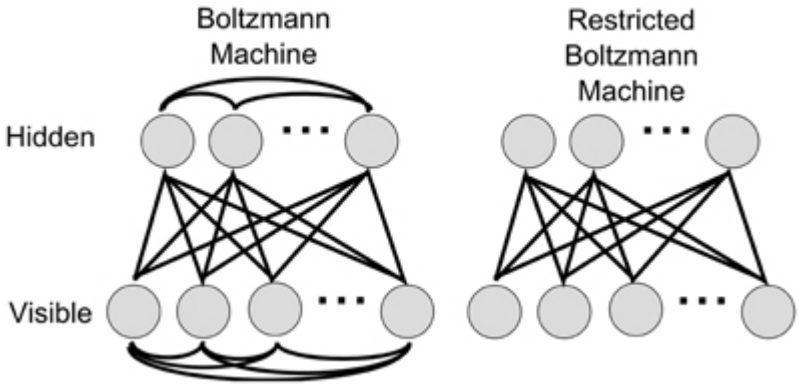

Figure 2.10: A comparison between a general Boltmann machine and a restricted boltzmann machine.

to this independence between hidden and visible units, the model can be trained with more efficient training algorithms, especially contrastive divergence [32]. RBMs are mainly interesting beacuse of their capability of pre-training deep neural networks in an unsupervised way. They were the original building block model in unsupervised pre-training by Hinton [34, 12].

Deep Belief Networks

Deep Belief Networks (DBNs) are probabilistic generative models, composed of multiple hidden layers [35]. The two top layers have undirected, symmetric connections between them. The lower layers (i.e. hidden layers) act as visible layers for the layer below them. DBNs basically extends the RBM architecture to multiple hidden layers, and can be viewed as a model of stacked RBMs or Autoencoders. After pre-training DBNs in a greedy manner, the models can be fine-tuned via backpropagation.

Autoencoders

An autoencoder (AE) is a special type of feedforward ANN that aims to learn a compressed encoding of some input, typically for reducing dimentionality. Autoencoders have as many output nodes as input nodes, because they are not trained to predict a value given inputs, but rather to reconstruct their inputs.

Basic autoencoders are very similar to RBMs in functional form, though their respective training procedures are very different.

Several variations of autoencoders have appeared in recent years, some of which can outperform RBM models on some benchmarks. Stacking autoencoders usually will usualy yield almost as good a classification performance as stacking RBMs [11].

Basic and Sparse Autoencoders

The training of basic autoencoders involves finding parameters that minimize the reconstruction error on a set of training examples. Any kind of backpropagation method can be used in training. The autoencoder will try to approximate the identity function, but by placing constraints on the network, sparsity (i.e. limiting number of hidden units) for

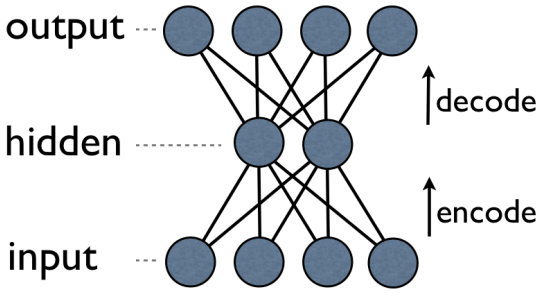

Figure 2.11: Illustration of a general autoencoder.

instance, the network can discover interesting structures in the data that results in it learning a compressed representation of the data. This means that the autoencoder will encode the learning representation between the input layer and the hidden layer, and decode it between the hidden layer and the output layer (illustrated in Figure 2.11).

These sparse autoencoders are popularly used as feature extractors[31], as they are very adept at learning useful features from various input domains [54]. Recently, many types of autoencoders that are very useful in deep learning and feature extraction. Deep autoencoders, for instance, are a special form of deep neural networks that is very useful for dimentionality reduction.

Denoising Autoencoder

Another strategy to improve upon the autoencoder model is the denoising autoencoder [62], which attempts to repair partially corrupted input. This means that, unlike basic autoencoders, denoising autoencoders cannot learn the identity function of the input data. Denoising autoencoders can be be stacked to pre-train deep NNs, similarly to RBM stacking. They have yielded better results on different datasets.

Deep Autoencoders

Deep autoencoders are autoencoders with many hidden layers. As previously mentioned, they can facilitate other tasks on high-dimentional data (e.g. classification) by reducing dimentionality, and are considered very useful for certain tasks. They, like RBMs, were introduced long ago, but only became viable by Hinton's discovery of pre-training strategy. Deep autoencoders can be considered a nonlinear generalization of Principal component analysis (PCA), and produce more compressed codes compared to PCA.

2.4.4 Training Algorithms

Optimization methods play an important part in supervised learning with neural networks. First-order methods like Steepest Gradient Descent used to be most popular, but had problems like local minima and slow convergence and overfitting. Optimization techniques like initialization methods and momentum techniques been shown to improve upon them in experiments.

[31]The act of feature extraction produces a (reduced) set of features from input data. A typical classifier (e.g. an ANN or an SVM) can then be trained on the extracted features. Feature extraction is a special form of dimensional reduction in image processing and pattern recognition.

Backpropagation

Backpropagation[32] is a learning method for artificial neural networks, which is a generalization of the delta rule to multilayered feedforward networks. It is commonly used with optimization methods like gradient descent. Backpropagation works by having error-correcting signals from each network layer to the previous, making corrective adjustments to the weights in the direction that cause the greatest change in output. The best possible weight assignment is not guaranteed, as it can get trapped in a local optimum, and there is a risk of bogging down the procedure in deeper networks.

The backpropagation algorithm assumes that the ANN corresponds to a directed graph, and that it possibly contains cycles. Learning with packpropagation corresponds to choosing a weight for each of the edges in the graph [49, p. 83].

It tries to minimize the squared error by trying to choose the weights that best fit the set of training examples, i.e. minimizing the loss function for the observed training examples (and other prior constraints or knowledge).

There is, however, the risk of "overfitting[33]", i.e. the neural network learning the data too well, including irrelevant details that are not present outside the training set. This results in a poor ability to generalize. This may be combated to some extent by for instance using more data, ensembling techniques, and/or model averaging, or by applying regularization techniques like weight decay (L2-regularization) or sparsity (L1-regularization).

2.4.5 Regularization

Dropout

In a 2012 paper [36], Hinton, Srivastava, Krizhevsky, Sutskever and Salakhutdinov proposed random *dropout*, an efficient way to average many large neural nets, thereby reducing overfitting. It prevents complex co-adaptations on the training data [36]. The technique works by setting a randomly selected subset of activations to zero within each layer, i.e. randomly omitting each hidden unit with probability of 0.5 each time a training example is presented [33]. A hidden unit can therefore not rely on other hidden units' presence. The authors write that another way to look at the procedure is "as a very efficient way of performing model averaging[34] with neural networks. A good way to reduce the error on the test set is to average the predictions produced by a very large number of different networks" [36]. Instead of training a host of separate networks and applying each of theses to the test data (which is computationally expensive), it is possible to "train a huge number of different networks in a reasonable time"[36] using dropout.

Hinton et al. used the stochastic gradient descent procedure for training on small batches of training cases, but modified the penalty term. They set an upper bound on the L2 norm of the incoming weight vector for the hidden units instead of penalizing the squared length (L2 norm) of the whole weight vector, renormalizing the weights of the hidden units by division if a weight-update violated the constraint. The authors noted that this use of a constraint rather than a penalty prevented weights from growing very large no matter how large the proposed weight-update was, which made it "possible to start with a very large learning rate which decays during learning, thus allowing a far more

[32] An abbreviation for "backward propagation of errors."

[33] As opposed to underfitting, which occurs when there is not enough representative training data or the learning algorithm gets stuck in local minima.

[34]

thorough search of the weight-space than methods that start with small weights and use a small learning rate" [36]. When testing, they used the "mean network" [36], containing all the hidden units, the outgoing weights of which were halved due to the fact that twice as many of them were now active. The expected value then stayed the same at training and test time. This led to a performance similar to that of averaging over many dropout networks [36].

The authors note that using the mean network is the equivalent to "aking the geometric mean of the probability distributions over labels predicted by all $2N$ possible networks" [36] in networks with one hidden layer and a softmax output layer for computing the probabilities of the class labels.

Hinton et al. [36] explored the effectiveness of dropout with the MNIST dataset, which "contains 60,000 28x28 training images of individual hand written digits and 10,000 test images" [36]. While previous works had achieved greatly improved performance using a variety of techniques[35] (some of which were very similar to the work [26] of Goodfellow et al.), Hinton et al. achieved fewer errors than the best published results for a standard feedforward network (160): by "using 50% dropout with separate L2 constraints on the incoming weights of each hidden unit" [36] and dropping out a random 20% of the pixels they reduced it to about 110 errors. Using dropout in combination with generative pre-training, the authors achieved an error of 92 when fine-tuning a publically available pre-trained deep belief net "using 50% dropout of the hidden units" [36]. In comparison, the net got 118 errors when fine-tuned using only the standard back-propagation. Using the CIFAR-10 benchmark, Hinton et al. [36] were able to improve upon the best published error rate on the test set, without using transformed data, from 18.5% to 15.6% "by using a neural network with three convolutional hidden layers interleaved with three “max-pooling” layers that report the maximum activity in local pools of convolutional units. These six layers were followed by one locally-connected layer... " [36] and using dropout in the last hidden layer. They also achieved similar gains in performance using the dropout technique with deep Boltzmann machines and hidden Markov models.

DropConnect

A year later, dropout was later generalized by Wan, Zeiler, Zhang, LeCun and Fergus [63]. Their generalization, "DropConnect" "... sets a randomly selected subset of *weights* within the network to zero" [63] rather than the activations within each layer. DropConnect often outperformed Dropout during the authors' experiments, and they note that it "yields state- of-the-art results on a variety of standard benchmarks using our efficient GPU implementation of DropConnect" [63]. Their code is available on the project's website[36].

2.4.6 Activation Functions

The activation function of a node is what defines its output given its input. Popular types of activation functions include threshold function, sigmoid function and logistic function. Neurons are activated, or "firing" to use a biologically inspired metaphor, when their output is close to one. Two types of activation functions which have recently appeared show

[35] Such as augmentation of the training data, wiring spatial transformation-based knowledge into a convolutional neural network or the use of generative pre-training.

[36] `http://cs.nyu.edu/~wanli/dropc/`

particular promise: Rectifiers (as employed in rectified linear units), which sidestep vanishing and exploding gradients; and Maxout [25], where the activation function is not fixed, but rather learned. The latter is well suited for use in conjunction with the regularization technique Dropout, as it improves its accuracy.

2.5 Related Work

The recent trend for computer vision problems have been to use GPU-accelerated implementations of deep convolutional neural network with a dropout-based regularizer (e.g. DropConnect) as well as performing augmentation of the dataset in order to improve the learning system's performance. These kinds of models have produced state-of-the-art results several times. Little to no preprocessing is done to the image datasets, since convolutional neural nets are able to operate on raw pixels.

2.5.1 Benchmarking Datasets

The following are standardized classification datasets that are widely used for benchmarking purposes.

MNIST

MNIST[37] is a database of black and white images of handwritten digits. The dataset contains 60 000 training examples and 10 000 test examples, all 28x28 pixels. It is commonly used as an evaluation set for deep learning image classification systems.

CIFAR-10

CIFAR-10[38] is one of the first benchmark datasets new machine learning methods are typically tested on. It consists of 50 000 training images and 10 000 test images, containing 10 classes of 6000 32x32 color images. The classes, which are mutually exclusive, are ones like "airplane", "automobile", and so on.

CIFAR-100

This dataset[39] is identical to CIFAR-10, except that it contains 100 classes of 600 images each. It has 500 training images and 100 testing images per class. In this dataset, all of the 100 classes are grouped into 20 superclasses, and each image has two corresponding labels (one for its class, and one for the class' superclass). Examples of these superclasses are "large carnivores" and "large man-made outdoor things", which contain "leopard" and "skyscraper" respectively.

STL-10

The STL-10 dataset[40] is similar to CIFAR-10, but it has fewer labeled training examples in each class. It also includes a large number of unlabeled examples for unsupervised

[37] http://yann.lecun.com/exdb/mnist/
[38] http://www.cs.toronto.edu/~kriz/cifar.html
[39] http://www.cs.toronto.edu/~kriz/cifar.html
[40] http://web.stanford.edu/~acoates/stl10/

pretraining-purposes. The dataset contains 10 classes of 500 labeled training images and 800 labeled testing images — as well as 100 000 unlabeled images (for unsupervised learning), which in addition to the classes in the labeled set contains similar but other types of classes (e.g. animals or vehicles) than those encountered in the labeled set. All of the 96x96 images were originally found on ImageNet.

Street View House Numbers

SVHN[41] is a real-world dataset taken from house numbers in Google Street View images. It contains over 600 000 images from 10 classes (1 for each digit), and is available in two formats: Original images with character bounding boxes; and 32x32, MNIST-inspired images centered around a single character.

ImageNet

ImageNet[42] is an enormous database of images (containing upwards of 14 million images as of late September 2014) that are organized hierarchically.

The ImageNet Large Scale Visual Recognition Challenge of 2012 (ILSVRC12)[43] is widely used for benchmarking. Its training data contains 1.2 million images from 1000 categories. Validation and test data includes 150 000 images, which are labeled with the presence or absence of 1000 categories.

NIST SD 19

The NIST Special Database 19[44] dataset is a proprietary set of handwritten forms and characters. The dataset includes handprinted forms from 3600 different writers, 810 000 classified separate character images.

CICDAR 2011 chinese handwriting recognition competition

These datasets[45] for evaluating character recognition of isolated handwritten chinese characters are offline and online datasets available in both feature data and original sample data formats. They are divided into training set of 240 writers and a testing set of 60 writers.

CIJCNN 2011 Traffic Sign Recognition Competition

This german traffic sign recognition benchmark dataset[46] contains more than 50 000 images of more than 40 classes, with each real traffic sign being unique (i.e. each physical traffic sign is only represented once). The images vary in size between 15x15 and 250x250 pixels, and are not necessarily square. The traffic signs are not necessarily centered in the image.

[41] http://ufldl.stanford.edu/housenumbers/
[42] http://www.image-net.org
[43] http://www.image-net.org/challenges/LSVRC/2012/
[44] http://www.nist.gov/srd/nistsd19.cfm
[45] http://www.nlpr.ia.ac.cn/databases/handwriting/Download.html
[46] http://benchmark.ini.rub.de/?subsection=gtsrb&subsection=news

NORB

The NORB dataset[47] is a database intended for 3D object recognition from shapes. It contains image pairs (shot by two cameras) of fifty toys imaged under various ligthing conditions, elevations and azimuths. The toys imaged belonged to 5 generic classes.

2.5.2 Computer Vision and Pattern Recognition

In 2014, a new method called deeply-supervised nets [42] was introduced by Lee, Xie, Gallagher, Zhang and Tu. The method was based on convolutional neural networks, but enforced supervision on both the hidden layers and the output layer. They introduced a *companion objective* (i.e. local output) to each individual hidden layer, which was used as additional constraint to learning. This meant that they backpropagated from the local outputs as well as from the final layer. The authors comment that their empirical result suggests that the companion objective leads to a reduction on testing error but not necessarily in training error, and that it results in faster convergence, especially with little training data [42]. They followed a training protocol [40] that was previously used by Krizhevsky et al., and achieved state-of-the-art results on the benchmark datasets CIFAR-10, CIFAR-100 and SVHN. As of October 2014, they are still the best[48] on all these three (with the technical exception of CIFAR-10, the score board for which is currently lead by human being's manual classification of 400 training images).

Lee, Grosse, Ranganath and Ng presented the *convolutional deep belief network*, "a hierarchical generative model which scales to realistic image sizes" [43] in their 2009 paper, which built upon the Deep Belief Network (Hinton et al., 2006)! Their network consisted of multiple max-pooling convolutional restricted boltzmann machines (CRBM), which are similar to RBMs, but with the weights between the hidden and visible layers being shared among all locations in an image. The network is a "scalable generative model for learning hierarchical representations from unlabeled images" [43]. It performed well in a variety of visual recognition tasks. The authors pointed out that they believed their approach held "promise as a scalable algorithm for learning hierarchical representations from high-dimensional, complex data" [43].

Building on their work [17] from the previous year, Cireşan, Meier and Schmidhuber developed the first method to achieve near-human performance on the MNIST handwriting benchmark [19], which also (in a world's first) outperformed humans by a factor of two on a traffic sign recognition benchmark. The authors claim that properly trained wide and deep deep neural networks (DNN) can "outperform all previous methods" [19] and demonstrate that initialization/pre-training is not necessary (although it might sometimes be helpful, especially when using datasets with relatively few samples of each class). They also show that their multi-column DNN ("further decreases the error rate by 30-40%") improves the state-of-the-art performance by "30-80%" [19] on many image classification datasets (including "MNIST, NIST SD 19, Chinese characters, traffic signs, CIFAR10 and NORB" [19]). Their method was fully supervised, and did not use any extra unlabeled data for training purposes. They conclude that while single DNN are already enough for obtaining new state-of-the-art results, combining them into Multi Column DNNs gives dramatic boosts in performance, e.g. a relative improvement of 41% on MNIST and of 39% on CIFAR10 [19].

[47] `http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/`

[48] According to the crowd sourced list on `http://rodrigob.github.io/are_we_there_yet/build/`

In 2012, Krizhevsky, Sutskever and Hinton trained a deep convolutional neural network to classify images in the ImageNet LSVRC-2010 contest (1.2 million high-resolution images) into 1000 different classes [40]. They achieved top-1 and top-5[49] error rates, of 37.5% and 17.0% respectively, which was considerably better than previous state-of-the-art solutions. Their resulting neural network had 60 million parameters and 650 000 neurons, and consisted of five convolutional layers. They also took advantage of a recently-developed regularization method, "dropout" [36], that proved to be very effective. The only preprocessing the authors used were down-sampling of the images to a resolution of 256×256, as well as "subtracting the mean activity over the training set from each pixel" [40], resulting in them training their network "on the (centered) raw RGB values of the pixels" [40]. The network was trained using "stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005" [40]. The authors also note that they found the small amount of weight decay an important factor in order for the model to learn. They conclude that a large, deep convolutional neural network "is capable of achieving record- breaking results on a highly challenging dataset using purely supervised learning" [40], and that the depth is very important for achieving the best results.

Goodfellow, Bulatov, Ibarz, Arnoud and Shet published a paper in 2013, in which they described their work on a deep convolutional neural network that operated directly on image pixels from Street View imagery, recognizing arbitrary multi-digit numbers [26]. The dataset they used consisted of around 200 000 images of street numbers, as well as individual digit bounding boxes, totaling at around 600 000 digits. While previously published work typically tried to recognize individual letters in cropped images, Goodfellow et al. instead focused on recognizing all of the multiple digits in the original images. Their preprocessing consisted of generating "several randomly shifted versions of each training example, in order to increase the size of the dataset" [26]. This augmentation gained them "about half a percentage point of accuracy" [26]. Their best results were that of 95.64% coverage[50] at 98% accuracy using confidence tresholding. The system achieved a character-level accuracy of 97.84% - slightly better than the previous state of the art. The training of this model took "approximately six days using 10 replicas in DistBelief" [26]. The authors hypothesize that the depth of their network was crucial to their success and that neural networks of such depth need to be trained on a big amount of data in order to be successful as a consequence of their high representational capacity. They found that the performance of their approach increased with the depth of the convolutional network, with their deepest architecture (with eleven hidden layers) achieving the best performance. They also found it very interesting that neural networks "learn to perform complicated tasks such as simultaneous localization and segmentation of ordered sequences of objects" [26], and that this end-to-end system approach could be applicable to problems such as "general text transcription or speech recognition" [26].

In his 2013 paper, Howard investigated several techniques to "improve upon the current state of the art deep convolutional neural network based image classification pipeline" [37], using the Imagenet Large Scale Visual Recognition Challenge dataset, models based upon Krizhevsky, Sutskever and Hinton's work [40] and using code[51] provided by Krizhevsky as a starting point. His model structure is identical to that of Krizhevsky et al., except from

[49] The fraction of test images for which the correct label is not among the five labels considered most profitable by the model.

[50] The undiscarded proportion of inputs after evaluating "the probability of the most likely prediction being corect" [26], where inputs below the threshold of this confidence value were discarded.

[51] Available at `http://code.google.com/p/cuda-convnet`

having twice as big, fully connected layers, a facet which did not improve the top-5 error rate. The training methodology used was the same as Krizhevsky et al. used. Since the performance of deep neural networks can be improved a great deal by adding more training data, the author added two image transformations to augment the dataset, in addition to what Krizhevsky et al. had performed previously, namely extending image crops into extra pixels and performing additional color manipulations to the random lighting noise Krizhevsky et al. used. His final image classification system, which was composed of "10 neural networks made up of 5 base models and 5 high resolution models" [37] had a test set top-5 error rate of 13.6% - an improvement on the previous state of the art (16.4% [40]), but short of the best result (11.7%) in 2013. The author speculates that the methods described in the paper could be used to improve upon the current state of the art.

2.5.3 Alzheimer's Disease

In 2008 Klöppel et al. developed a robust method that could be generalized across different medical centres, using linear support vector machines to classify the "grey matter segment of T1-weighted MR scans" [39]. from diagnosed Alzheimer's patients and cognitively normal controls, which had been obtained from "two centres with different scanning equipment" [39]. They also tested the support vector machines's ability to differentiate control scans from subjects with no post-mortem confirmation of AD, as well as differentiating scans of patients affected by AD from scans of patients with frontotemporal lobar degeneration. Their solution classified up to 96% of pathologically verified AD patients correctly using whole brain images. Data from different centres achieved comparable results to the separate analyses, allowing support vector machines to be trained on data from one center, and used to accurately differentiate between AD and normal scans obtained from another center with different subjects and scanning equipment. The method also correctly assigned 89% of patients with post-mortem comfirmed diagnosis of AD or frontotemporal lobar degeneration to their group, and correctly separated patients with mild clinically probable AD and age/sex matched controls in 89% of cases - a feature which was "compatible with published diagnosis rates in the best clinical centres" [39]

In their 2009 paper, Magnin et al. presented and evaluated a new method of classification of whole-brain (1.5-T) MRI to discriminate between AD patients and control subjects based on Support Vector Machines. They parcellated the brain into Regions Of Interest by using a previously developed anatomically labelled template of the brain, and created a mask to exclude voxels of the skull. The authors obtained "94.5% mean correct classification for AD and control subjects (mean specificity, 96.6%; mean sensitivity, 91.5%)" [45], where few 34 out of 38 subjects had a mean correct classification rate greater than 90%. They conclude that their method has a potential for early diagnosis of Alzheimer's.

Morra et al. performed a comparison of four automated methods for hippocampal segmentation using different machine learning algorithms in 2010. The methods they compared were "(1) hierarchical AdaBoost, (2) Support Vector Machines (SVM) with manual feature selection, (3) hierarchical SVM with automated feature selection (Ada-SVM), and (4) a publicly available brain segmentation package (FreeSurfer)" [51]. In their report, they show that all of the methods were "capable of capturing both disease related effects and correlations between cognition and structure for these well known, widespread effects" [51].

In her Ph.D. Thesis, Katherine Gray describes her work with imaging biomarkers

for Alzheimer's disease. A multi-modal classification framework based on similarities derived from random forests is applied to the combination of MRI, PET, cerebrospinal fluid biomarkers and genetic information, and out-performs classification based on any individual modality (see footnote 11, p. 12). The author argues that her findings suggests that "volumetric MRI can reveal structural brain changes that precede the onset of clinical symptoms" [28], and that it may be useful for potentially providing a useful tool for early screening, to help measure outcome for clinical trials or to help identify the early signs of neurodegeneration in otherwise healthy elderly people. She also suggests that an potential avenue for future work "could be to compare the subjects that are mis-classified using different machine learning methods based on the various available modalities" [28].

In 2013 Gupta, Ayhan and Maida "used a sparse autoencoder to learn a set of bases from natural images and then applied convolution to extract features from the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset" [29]. They then classified MRI instances into three categories of AD, MCI and healthy control (HC). Their approach achieved high diagnostic accuracy, and was competitive with or better than other approaces, despite "being very simple ..." [29], and not incorporating prior domain-knowledge in their data processing steps.

In their now withdrawn[52] 2014 study, Liu and Shen trained a deep convolutional neural network for Alzheimer's Disease and Mild Cognitive Impairment classification. They then tested it's performance on a subset of MRI images from the Alzheimer's Disease Neuroimaging Initiative, containing 200 subjects of each group (AD, MCI and health control group), and showed that the learned deep features perform significantly better than conventional MRI based AD classification methods, as well as better than recent multimodal classification methods. They obtained "new state-of-the-art results: an accuracy of 97.18%, 93.21% for AD and MCI identification (from HC) respectively, and an accuracy of 94.64% for classifying AD from MCI" [44]

Vemuri et al. developed a tool [61] for Alzheimer's diagnosis via classification of structural MRI via Support Vector Machine in 2008. They developed three different classification models based on data from 190 patients with probable AD and 190 cognitively normal subjects, with 140 from each group used for training and 50 for testing. The two models usig covariate data — demographics and Apolipoprotein E genotype respectively — in addition to MRI achieved accuracy-scores (aSTAND) of 88.5% and 89.3% respectively. In addition, anatomic patterns that differentiated the two groups best were consistent with known Alzheimer's Disease pathology.

In 2011, Westman et al. compared MRI data from the ADNI-program with data from the European Union AddNeuroMed, another large multi-center initiative [67] using the same MRI data acquisition scheme. By using the automated segmentation scheme of Freesurfer, they discovered that the different cohorts showed similar patterns of atrophy, and that classifiers trained on one of them were applicable to the other. The combined cohort model was used to predict conversion to AD at baseline of MCI subjects at 1 year follow-up, and the predictive powers obtained from the different models were all between 80 and 90%.

A 2015 paper by Payan and Montana [55] presented their findings from experiments with deep learning methods on the ADNI dataset. They used sparse autoencoders for feature extraction, and trained 3D convolutional neural networks predict disease status

[52] A comment on the latest version of the article on arXiv (`http://arxiv.org/abs/1404.3366`) states that the paper has been withdrawn "due to an error in the MRI data used in the experiments".

based upon MRI brain scans, producing state-of-the-art results. They demonstrated that 3D convolutions on the whole MRI image yield better performance on the data than 2D convolutions on slices, and report classification results from "a 3-way classifier (HC vs. AD vs. MCI) and three binary classifiers (AD vs. HC, AD vs. MCI and MCI vs. HC)." [55] Their 3-way classifier achieved prediction accuracy of 85.53% and 89.47%, using 2D and 3D convolutions respectively. Binary classifiers achieved 95.39% (2D convolutions) and 95.39% (3D convolutions) on AD vs. HC, 82.24% (2D convolutions) and 86.84% (3D convolutions) on AD vs. MCI, and 90.13% (2D convolutions) and 92.11% (3D convolutions) on HC vs. MCI.

2.5.4 Machine Learning and Medical Imaging

In a 2002 study [71], Zhou, Jiang, Yang and Chen proposed an automatic lung cancer diagnosis system based on articifical neural network ensembles and image processing techniques. Their model was a two level ensemble architecture: the first level ensemble judged whether a cell was a cancer cell with high confidence; the second level ensemble classified what kind of cancer cell had been passed to it by the first level ensemble. The final system showed an average error (5-fold cross validation) error of 11.6%, 2.7% and 4.5% over three measures.

In a 2010 paper [24], Ganesan, Venkatesh, Rama and Palani describe their attempt to make use of neural networks in a pre-clinical carcinogenesis[53] study. They trained a multilayer artificial neural network on demographic data from 100 lung cancer patients using backpropagation. Their model achieved an accuracy of over 87%. The authors concluded that artificial neural networks show promise in assisting clinicians with tasks like diagnosis, and that automatic diagnosis "...is an important, real-world medical problem" [24].

Microsoft has a medical image analysis research project called Inner eye[54], which among other things allows for automatic detection and localization of anatomy based on CT-scans of patients via state of the art machine learning techniques. This algorithm has been approved by the FDA, and is currently being used in clinical settings. Also stuff about semantic navigation, segmentation, workflow, search by image region, etc.

2.5.5 Computer Aided Diagnosis

In 1997 Bottaci et al. trained six neural networks for "the prediction of death within 9, 12, 15, 18, 21, and 24 months" [15] for colorectal cancer patients. The networks were fully connected multilayer feedforward networks using backpropagation. All of the networks achieved and overall accuracy greater than 80% for the prediction of death for individual patients. The 12-month network was later applied to 2-year follow-up data from patients from a second institution. The network performed better than two consultant colorectal surgeons at an overall accuracy of 90% (95% confidence interval 84-96), beating their accuracy of "79% (71–87) and 75% (66–84)" [15].

[53] The creation of cancer.

[54] The project has a website at `http://research.microsoft.com/en-us/projects/medicalimageanalysis/`

Chapter 3

Methods

3.1 Design

This chapter presents the design of the experiments performed to answer the research questions posed in Chapter 1.2.1, i.e. find out how a classifier can be trained to differentiate between structural MR images of Alzheimer's Disease and other diagnostic groups.

Similar problems (and indeed this problem) have had deep neural networks applied to them during the last few years, with one very recent attempt [55] yielding good results when a deep 3D convolutional network was applied to the problem formulated as several binary classification tasks.

In order to answer the research questions, we needed to do the following:

Compare different kinds of dimensional reduction: As the instances in the dataset were high-dimensional, some form of dimensional reduction was needed in order to make training feasible.

Perform machine learning using several approaches: Tree-based methods (including random forests) as well as SVM have yielded promising results in previous research. Additionally, deep learning via neural networks have proven to be very accurate on computer vision problems in later years, and would appear to be well suited for this type of machine learning problem.

Train several binary classifiers as well as a multiclass classifier: In addition to training a three-way classifier, training three binary classifiers (using a "one-vs.-rest" approach to classification) would likely fit to (and reveal) any drastic differences between classes, and could eventually be combined later.

Compare our results with the results of similar research: See what configuration of variation in the three previous steps lead to the lowest classification error, and how this compared to different relevant research.

The problem we dealt with during this project was one of supervised classification, as the dataset had labeled instances (i.e. a diagnostic group associated with each MR brain image). Because of the dataset's characteristics, the curse of dimensionality[1] was an issue.

[1] Various problematic phenomena that arise when analyzing and organizing high-dimensional data. Their unifying feature being that as the dimensionality increases, the volume of the space increases so fast that the available data become sparse.

3.1.1 Overview

As the complete ADNI dataset is relatively large, and the instances (i.e. MR images) are very high-dimensional, some adjustments to the data were needed in order to make learning practical. This is because of the previously mentioned curse of dimensionality with regard to the machine learning problem at hand; since the dataset has very high dimensionality a classifier would need a very large amount of training data to ensure that several instances of each possible combination of values were present. We therefore performed three kinds of dimensional reduction: Principal Component Analysis (PCA), Histogram, and downscaling (resampling the images at lower resolution). The approaches used and the resulting dimensionally reduced dataset variants are described later in this chapter.

Additionally, one of the machine learning tools used (C5.0) in the experiments were unable to work with datasets larger than approximately 2GB. In order to facilitate a fair comparison, the processed datasets used in the experiments were thus limited accordingly in size, leading to variation in the total amount of images represented between experiments (further detailed later in this subsection). This also has the advantage of providing an added perspective, namely that of the amount of images vs. the amount of voxels.

All of the images (which had varying resolution) were first resized to 192x192x160 (the lowest resolution in the dataset[2]) using 3D spline interpolation for downsampling.

The three-dimentional MR images were represented as axial/transverse slices (i.e. the images' Z-axis, or through the top of the subjects' heads — as they were lying down when imaged).

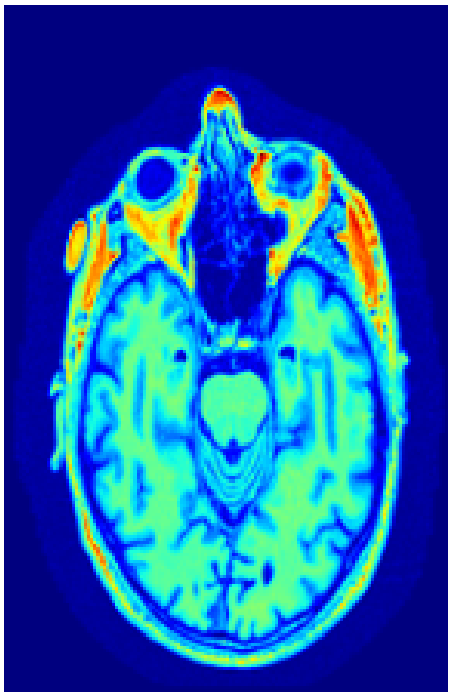

Figure 3.1: A (colorized) slice from one of the original MR images used (before any resizing and dimensional reduction.).

[2]The highest resolution in the dataset was 256x256x166.

The slices may be visualized as illustrated in Figure 3.1. At this point, dimesional reduction was performed (described in Chapter 3.1.2). After dimensional reduction, the pixels of the returned image(s) were then converted to continuous values suited for use with the machine learning software in the next step. At this point, diagnostic groups were merged (described in Chapter 3.1.3). The resulting datasets also contained metadata in the form of diagnostic group (i.e. a label for each training instance).

The resulting training data can be described as follows:

From the ".names" file used with C5.0, using some dimensional reduction method and the first class merging scheme

```
diagnosis.              | target attribute

...
    (pixel values)
...

diagnosis:              Normal, MCI, AD.
```

Specification-files included pixel-value attributes for the training instances (processed images).

Each resulting dataset was then used for training in experiments with several different machine learning techniques (and several configurations of these).

3.1.2 Dimensionality reduction

Some researchers have performed automatic segmentation with software such as Freesurfer [39, 45, 51, 67], or feature selection using autoencoders [29, 55], but in the last decade minimal preprocessing (like resizing of the images) have been popular in computer vision research — because of use in conjunction with (sometimes convolutional) deep neural networks, and using modern "tricks", such as dropout.

As the instances in the dataset (i.e. complete 3D images) contains extra information — such as skin, skull and space around subject heads — and as there are specific key areas of the brain that is associated with atrophy (in addition to general shrinkage of the brain) in the case of Alzheimer's disease, we suspect that much of the information in the images is unneeded for classification purposes. By successfully reducing the dimensionality of the data, we stand to gain in terms of classification rates, training time and data size. Reducing the dimensionality of the dataset is not without its pitfalls, though. If we were to reduce too drastically, or if an approach is not well suited for this specific problem, we risk losing valuable information — resulting in classifiers that perform worse.

PCA

For dimensionality reduction, every kept slice of each MR image was normalized, before the well known PCA[3] method (specifically, the scikit-learn[4] implementation of randomized

[3]Principal Component Analysis

[4]Project website: `http://scikit-learn.org`

PCA.) was applied to the slices, extracting 20 components.

In order to further decrease the size of the dataset, only every 4th slice of every image was kept for further use. This effectively reduced the dimensions of each instance from 192x192x160 (after the original resizing) to 48x192x20. This variation included a total of 1760 images.

Histogram

In this version of the dataset, each MR image (in its entirety) was standardized by scaling each voxel-value to range (0,1). A histogram was then computed over the 3D image, using 32 bins of equal width. Since this dramatically reduced the size of each instance, all of the 2182 images were used in this variation of the dataset.

Downscaling

The last version of the dataset was reduced by simply downscaling all MR images to 69x95x79 via 3D spline downsampling (as with the the initial resizing mentioned earlier). This variation included 570 images in total.

3.1.3 Merging diagnostic groups

All dimensionally reduced datasets were produced in four variations of merged classes (to facilitate binary classifications, as per research question 1.3):

1. Normal / MCI / AD
2. Normal / Other
3. Other / MCI
4. Other / AD

3.1.4 Machine learning tools

The experiments used different of machine learning techniques and approaches. The following tools were chosen, as they were known to have good implementations of the algorithms in question.

Decision trees

Decision trees models are interpretable by humans, and work well in many cases. They are a popular method for various ML problems, even though they they have a tendency to overfit to their training set. Tree-based learning methods (including random forests, which is an ensemble method based on decision trees) have also performed well with regard to similar problems [28].

C5.0's implementation was chosen because it is widely known to be a very good implementation of decision trees (in addition to decision rules[5]).

Some of C5.0's options include:

[5]Linearized version of decision trees, e.g. "if condition1 and condition2 then outcome."

Boosting Constructing several classifiers, where each new classifier pays more attention to the cases where the previous classifier made errors. The different trees are then combined by voting.

Winnowing Removing attributes that do not increase the model's predictive power with regard to the classification task at hand.

Pruning Removing parts of the tree that are predicted to have a relatively high error rate.

Neural networks

Neural networks are robust to noisy data and missing variables. They can combat overfitting easier than tree-based techniques, using regularization techniques. However, produced models can be difficult to understand for humans, and they take a long time to train well compared to methods. Also, hyperparameters (such as learning-rate, mini-batch, and number of hidden units and layers) must be tended to, and a certain amount of experimentation must be performed.

Since we wanted to experiment with some newer techniques, we had to make an evaluation of some of the most popular modern neural network tools. This is described in detail in Chapter 3.3. As our focus was on ease of experimentation and support for relatively recent tricks and techniques, we chose the bleeding edge research library Pylearn2.

3.1.5 Procedure

The procedure can thus be described in the following way; First, the data conversion:

- Resizing of images to lowest resolution in dataset
- Dimensional reduction
- Merging of classes (i.e. diagnostic groups)
- Output to specified format

After this, scripts that run specified experiments were executed, taking about two days to finish for all experiments (i.e. with all combinations of dimensional reduction and merged classes) for one machine learning technique (e.g. decision trees).

3.1.6 Criteria and Analysis

Commonly used performance measurements in the field of machine learning include: p-values, f-measure, specificity and sensitivity, confidence intervals, significance, characterization of errors (e.g. bias and variance), confusion matrix, ROC curve, and error percentages on standardized benchmark datasets (examples of which can be found in Section 2.5.1).

In this thesis, the classification accuracy (and classification error) is used as performance metrics, as this is what has been used in most comparable related work. We evaluate the performance of the techniques by comparing them to results from related work.

3.1.7 Practical information about replication

All relevant project scripts can be found on a dedicated public github repository[6].

There should be no issue running these experiments with C5.0, as the tools is freely available in source form (a single-threaded GPL-version) from its creator's website[7]. The project scripts also include configuration parameters.

Pylearn2, on the other hand, can be somewhat complicated to install, set up and use, though some of the project scripts may be somewhat helpful during this process.

Also included are scripts that describe and perform the training runs with both C5.0 and Pylearn2.

As the complete (and the converted versions of) the ADNI dataset is very large (the original is over 80GB in total) and I don't have permission to redistribute it, it must be collected manually. This can be done by applying for access at LONI's website[8]. All logs and result-files are included.

3.2 ADNI Standardized MRI Dataset

The complete dataset[9] contains 2182 three-dimensional T1-weighted magnetic resonance images of patients (an example of which can be seen in Figure 3.2), obtained using machines with a field strength of 1.5 teslas, from the following three groups:

- 200 Alzheimer's Disease (mild)
- 400 Mild Cognitive Impairment
- 200 Normal (control)

The dataset was created by the Alzheimer's Disease Neuroimaging Initiative [6] - a world-wide project that provides reliable clinical data for the research of pathology principle, prevention and treatment of Alzheimer's disease.

To the author's knowledge, this is the largest available dataset of its kind.

The images in the dataset were all corrected in the following ways [5]:

1. Gradwarp – gradwarp is a system specific correction of image geometry distortion due to gradient non-linearity. The degree to which images are distorted due to gradient non-linearity varies with each specific gradient model. It is anticipated that most users would want to use images which have been corrected for gradient non-linearity distortion in analyses.

2. B1 non-uniformity – this correction procedure employs the B1 calibration scans noted in the protocol above to correct the image intensity non-uniformity that results when RF transmission is performed with a more uniform body coil while reception is performed with a less uniform head coil.

3. N3 – N3 is a histogram peak sharpening algorithm which is applied to all images. It is applied after grad warp and after B1 correction for systems on which these two

[6] `https://github.com/eivind88/master_code`

[7] `https://www.rulequest.com/download.html`

[8] `http://adni.loni.usc.edu/data-samples/access-data/`

[9] The complete 3 year 1.5 tesla dataset from the ADNI1 study.

correction steps are performed. N3 will reduce intensity non-uniformity due to the wave or the dielectric effect at 3T. 1.5T scans also undergo N3 processing to reduce residual intensity non-uniformity.

The need to perform the aforementioned preprocessing corrections varied with manufacturer and system RF coil configuration [5]. In addition, phantom based scaling measures were associated with each image, and masks created by the MR Core as part of preprocessing are included in "Intracranial Space".

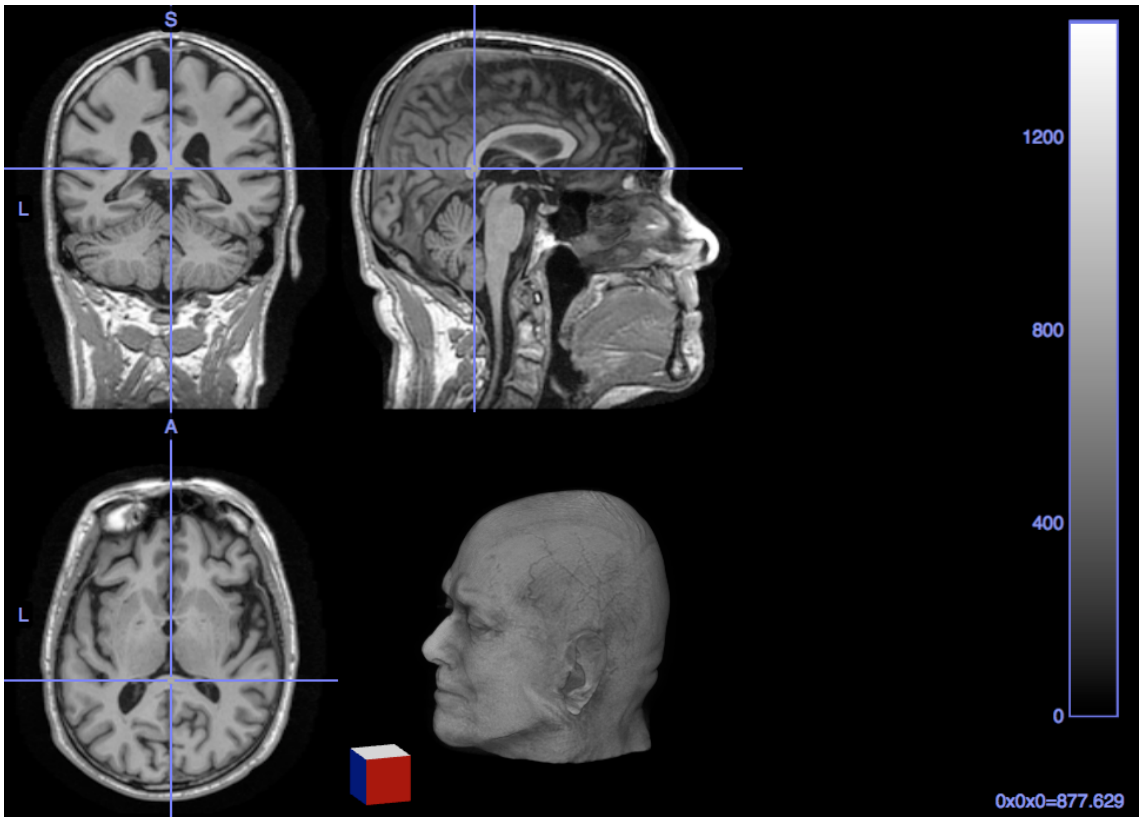

Figure 3.2: Example image from dataset.

The dataset also contains corresponding metadata for each brain scan, which includes information such as gender, age and diagnostic group.

Data used in the preparation of this article were obtained from the Alzheimer's Disease Neuroimaging Initiative (ADNI) database (`adni.loni.usc.edu`). The ADNI was launched in 2003 by the National Institute on Aging (NIA), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), the Food and Drug Administration (FDA), private pharmaceutical companies and non-profit organizations, as a $60 million, 5- year public-private partnership. The primary goal of ADNI has been to test whether serial magnetic resonance imaging (MRI), positron emission tomography (PET), other biological markers, and clinical and neuropsychological assessment can be combined to measure the progression of mild cognitive impairment (MCI) and early Alzheimer's disease (AD). Determination of sensitive and specific markers of very early AD progression is intended to aid researchers and clinicians to develop new treatments and monitor their effectiveness, as well as lessen the time and cost of clinical trials. The Principal Investigator of this initiative is Michael

W. Weiner, MD, VA Medical Center and University of California – San Francisco. ADNI is the result of efforts of many co-investigators from a broad range of academic institutions and private corporations, and subjects have been recruited from over 50 sites across the U.S. and Canada. The initial goal of ADNI was to recruit 800 subjects but ADNI has been followed by ADNI-GO and ADNI-2. To date these three protocols have recruited over 1500 adults, ages 55 to 90, to participate in the research, consisting of cognitively normal older individuals, people with early or late MCI, and people with early AD. The follow up duration of each group is specified in the protocols for ADNI-1, ADNI-2 and ADNI-GO. Subjects originally recruited for ADNI-1 and ADNI-GO had the option to be followed in ADNI-2. For up-to-date information, see `www.adni-info.org`.

3.3 Neural Network Tools: Overview and Evaluation

Machine learning has exploded in popularity during the last decade, and neural networks in particular have seen a resurgence due to the advent of deep learning. In this chapter, we examine modern neural network software-libraries and describe the main differences between them, and attempt to showcase their strengths and weaknesses and characterize situations in which they would be well suited for use.

3.3.1 Introduction

Neural network/computer vision research have moved with breakneck speed in later years, and the field of deep learning in particular, helping researchers and businesses to tackle a variety of challenging problems such as development of autonomous vehicles, visual recognition systems and computer-aided diagnosis. Many new tricks and techniques have been developed, and several frameworks, libraries and toolkits released for research purposes. These support various functionality and implement different techniques. While most of the more general alternatives are quite similar in terms of functionality, they nonetheless differ in terms of approach and design goals. The primary focus is on modern tools; Most of these will have some connection to deep learning, as it is an approach that is currently producing many state of the art results in fields like computer vision.

For the problem of answering the research questions of this thesis, there were a few important factors when chosing software tools. Models needed to be easy to experiment with, as experiments would largely consist of comparison between different models trained on different variations of the dataset. They would also have to be relatively efficiently implemented, as many experiments would be run. Lastly, modern techniques such as dropout regularization would have to be supported, since they have enabled increased performance in certain circumstances.

3.3.2 Tools

Pylearn2

Pylearn2 [27] is a cutting edge machine learning library for Python developed at the LISA-lab[10] at The University of Montreal, built with research in mind. It has a particular focus on deep learning. The library is focused on easy configurability for expert users

[10]One of recent years' most prominent labs, spearheaded by one of the leading researchers on deep learning, Yoshua Bengio.

(i.e. machine learning researchers) —unlike "black-box" libraries, which provide good performance without demanding knowledge about underlying algorithms from users. One way to put it would be that the library values ease of experimentation over ease of use.

Though Pylearn2 assumes some technical sophistication on the part of users, the library also has a focus on reusability, and the resulting modularity makes it possible to combine and adapt several re-usable components to form working models, and to only learn about the parts of the library one wants to use. One of the library's goals is to contain reference implementations of all models and algorithms published by the LISA-lab, and it has been used to set the state of the art on several standardized datasets, including a test error of 0.45% on MNIST in 2013 [25] —the best performance without data augmentation at that point —using a convolutional maxout-network with dropout regularization.

Pylearn2 makes use of a YAML[11]-interface, which allows users to set up and perform experiments rapidly by defining their models in an almost declarative style, using (pre-)defined components as building blocks and specifying hyperparameters. Alternatively, experiments may also be specified through a Python script.

The library is built upon the Theano library —a numerical computation library for Python and C/CUDA-compiler —which is also developed at the LISA-lab. Theano can also compute gradients automatically; Users need only specify architecture and loss function. This makes it very easy to experiment quickly.

GPU-based models are also enabled via compilation with Theano, as it can compile both to GPU and CPU. Pylearn2 also provides wrappers to widely used and efficient third-party GPU-implementations of convolutional nets like CUDA-convnet and Caffe.

As the library is under heavy development, its documentation is somewhat sparse, although the source code itself is thoroughly commented. Though its website has a few notebook[12]-style and short examples, users will likely have to spend some amount of time reading source code to figure out how to implement their own models and extensions, as well as how to integrate them (for use) with the library. Pylearn2 can thereby be somewhat hard to understand, extend, modify and debug for beginners. There is, however, a very active community around the library, including usergroups/mailinglists and the official source code repository[13]. Despite its somewhat steep learning-curve, Pylearn2 is very powerful.

Pylearn2 is to a large extent created and supported by it's users (students at the LISA-lab and other contributors), and since it has its focus on research, features are added as they are needed, meaning that users will likely have to code up some things themselves —unless they are only trying to replicate published results or run variations on old experiments/datasets.

Pylearn2 comes with support for/ready-to-use standardized benchmark datasets (such as MNIST and CIFAR-10) and unsupervised learning out of the box. It appears to be growing steadily in popularity, and is apparently popular amongst contestants in Kaggle contests.

[11] A human-readable data serialization format.

[12] A web-based (locally interactive) environment provided by the enhanced Python-shell and -project IPython.

[13] Available at `https://github.com/lisa-lab/pylearn2`

Torch

Torch [20] is an open source scientific computing framework that supports many different machine learning algorithms. It uses a JIT compilation-based implementation of the Lua-language for ease of use and efficiency, on top of a C/CUDA-implementation. Torch7 (its current version) is currently in use at and being contributed to by DeepMind (Google), Facebook, Twitter and New York University, according to the project's website[14].

It is somewhat similar and comparable to Theano/Pylearn2 (used in conjunction with Numpy and the rest of the packages in the "standard" Python scientific stack), complete with suitable datatypes which supports mathematical operations, stastistical distributions, and BLAS[15] operations, as well as supporting rapid experimentation via a REPL[16]. It is also composed of reusable parts that can be combined in many variations. Torch also has a large ecosystem of community-driven packages, and as with Pylearn2, other packages brings support for things like image processing —including a relatively recently released package[17] of CUDA extensions for deep learning by Facebook AI Research.

Torch allows neural network models to be run on GPUs through the help of CUDA. This can be achieved by users via simple typecasting.

Like Pylearn2, Torch7 includes scripts to load several popular datasets. Third party loaders are also available for datasets other than those that are supported by default.

According to a 2011 paper [21] by some of Torch's maintainers, Torch7 was faster than Theano on most benchmarks. The author's noted, however, that Theano was "faster than any existing implementation" (including Torch5, Matlab with GPUmat and EBLearn), going as far as saying that it *crushed* the alternatives in benchmarks when run on a GPU. It is also important to remember, as the authors comment, that only larger network architectures will benefit from GPU-acceleration.

In the realm of deployment, Torch may prove to be easier in use than for instance Pylearn2 (and MATLAB, which only supports deployment of pre-trained networks), as the LuaJIT environment is embeddable into a host of environments, including smart-phone applications and video games.

Torch is very practical to use for deep learning, as the library has a focus on these approaches. Like Pylearn2, Torch has an active community, including mailing lists/user groups, community wiki, online chat and an official source code repository on github.

Matlab Neural Network Toolbox

Matlab supports Neural Networks via the Neural Network Toolbox, and networks built with this can in turn be parallelized and run on GPUs when used in conjunction with the Parallel Computing Toolbox.

While it is not supported "out of the box", deep learning architectures can be used in Matlab via the Deep Learning Toolbox[18] third-party software. The toolbox includes support for Deep Belief Nets, Stacked Autoencoders, Convolutional Neural Nets, Convolutional Autoencoders and Feedforward Backpropagation Neural Nets. The toolbox also

[14] `http://torch.ch`
[15] Basic Linear Algebra Subprograms
[16] Read–eval–print loop, i.e. an interactive shell
[17] Available at `https://github.com/facebook/fbcunn`
[18] Available at `http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox`

allegedly works with GNU Octave[19], and includes sample scripts to get news users started. However, the public source code repository does not appear to have been updated since May 2014.

Matlab is very much established both in industry and academia, and it is therefore easier to find helpful tips and snippets of code online, as there is a larger userbase (and indeed more use cases, as it is a language of its own). Matlab also includes integrated development environment.

Scikit-learn

Scikit-learn [56] is another machine learning library for Python. Building upon widely used packages like numpy (N-dimensional arrays) and scipy (scientific computing), it contains a variety of classification, regression and clustering algorithms, providing machine learning techniques for supervised and unsupervised problems. The package "focuses on bringing machine learning to non-specialists" [56], and is thus more accessible than some alternatives. It does therefore, unlike Pylearn2, not require users to possess knowledge of the models' implementations. Also unlike Pylearn2, which focuses mostly on neural networks, scikit-learn has a wide varity of machine learning techniques available, including support vector machines, logistic regression, nearest neighbors and random forest, covering tasks like classification, regression, clustering, dimensionality reduction, model selection and preprocessing. The libraries website includes comprehensive documentation. It includes limited support for neural networks, but it is not very practical to use for deep learning.

Weka

Weka is machine learning software [30] from the University of Waikato, New Zealand, written with focus on data mining. It is written in Java, and is usable on its own (through it's GUI "explorer"), as well as callable from one's own Java code. While Weka has been viewed as a helpful tool, it seems to be falling out of favor as big data and deep architectures become more prevalent. It can to some extent still be helpful, but it cannot handle large datasets very well —though it is possible in some cases, with certain models. Although it works well as an exploratory tool for (smaller) datasets, Weka has typically not been used in state of the art machine learning research in later years, and is thought not to be appropriate for heavy lifting amongst some. This seems to be a pervasive attitude on several popular forums and other internet sites. Although how helpful Weka is in various cases is debatable, it is certainly not well suited for use with very large datasets as it's explorer will not be able to handle this even if one adjusts the JVM settings like heap size before launching it.

Features / GPUs and Convolutions

Caffe

Caffe[38] is a very efficient deep learning framework by Berkeley Vision and Learning Center at UC Berkely. It is mostly used for computer vision, i.e. training deep convolutional neural nets. Caffe supports training on CPU/GPU. It is implemented in C++, and has Python and MATLAB bindings. It is a research library, and includes bleeding edge functionality and very high performance.

[19]Open-source alternative to MATLAB, mostly cross-compatible.

CUDA-convnet(2)

CUDA-convnet (also sometimes referred to as AlexNet in literature) is a high-performance CUDA-based GPU-implementation of convolutional neural networks. It is made by Alex Krizhevsky, and has been used to achieve state of the art performance on CIFAR-10. CUDA-convnet is also a research library and a new version, which is more efficient on newer GPUs, has recently been released.

cuDNN

CuDNN is a GPU-accelerated library of deep neural network primitives recently released by NVIDIA.

3.3.3 Comparison of libraries

* = Support via third party.

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Multilayer perceptron	Yes	Yes	Yes	No	Yes
Autoencoder	Yes	Yes	Yes*	No	No
Boltzmann Machine	Yes	Yes*	Yes*	Yes	No
Convolution	Yes	Yes*	Yes*	No	No
Recurrence	Yes	Yes*	Yes	No	No
Deep Belief Network	Yes	No	Yes*	No	No

Table 3.1: Network Models

As can be seen in table 3.1, Scikit-learn and Weka do not support many different network architectures. This is somewhat unsurprising, as neither is specialized in neural networks, but rather general machine-learning toolkits. Pylearn2, Torch7 and Matlab, on the other hand, all have support (though in some cases via third party) for many different architectures.

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Support Vector Machine	Yes	Yes*	Yes	Yes	Yes
K-means	Yes	Yes*	Yes	Yes	Yes
Principal Component Analysis	Yes	Yes*	Yes	Yes	Yes
Decision Trees	No	No	Yes	Yes	Yes
Random Forest	No	No	Yes*	Yes	Yes

Table 3.2: Other Models

In table 3.2 we see Scikit-learn, Weka and Matlab support tree-based models as well. The fact that Pylearn2 and Torch7 do not is because they are neural network libraries.

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Sigmoid	Yes	Yes	Yes	No	Yes
Tanh	Yes	Yes	Yes	No	No
Linear	Yes	Yes	Yes	No	No
ReLU	Yes	Yes	Yes*	No	No
Softplus	No	Yes	Yes	No	No
Softmax	Yes	Yes	Yes	No	No
Maxout	Yes	No	No	No	No

Table 3.3: Transfer Functions

Table 3.3 shows that Scikit-learn and Weka once again does not support much in terms of neural networks: Weka supports a single one of the transfer functions compared, and Scikit-learn none. Amongst the three remaining contenders, Pylearn2 is the only one that implements Maxout (the discoverer of which created Pylearn2).

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
(Minibatch) Stochastic Gradient Descent	Yes	Yes	Yes	No	No
Batch Gradient Descent	Yes	Yes	Yes	No	No
Levenberg-Marquardt	No	No	Yes	No	No

Table 3.4: Training Algorithms

The tools with focus on neural nets all support the usual gradient descent training algorithms, but only Matlab supports Levenberg-Marquardt in table 3.4.

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
L1 Weight Decay	Yes	Yes	Yes*	No	No
L2 Weight Decay	Yes	Yes	Yes*	No	No
Momentum	Yes	Yes	Yes*	No	No
Dropout	Yes	Yes	Yes*	No	No

Table 3.5: Regularization

In table 3.5, all of the tools with focus on neural nets support all of the regularization techniques compared (though Matlab only via third party).

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
GPU acceleration	Yes	Yes	Yes	No	No
Support for standardized benchmark datasets	Yes	Yes	No	No	No
Bindings for Caffe	Yes	No	No	No	No
Bindings for CUDA-convnet	Yes	Yes	No	No	No
Bindings for cuDNN	Yes	Yes	No	No	No

Table 3.6: Other Features

Table 3.6 shows that Pylearn2 has bindings for Caffe, but is otherwise comparable to Torch7. Matlab, on the other hand, only supports GPU acceleration. Scikit-learn and

Weka have none of the compared features, as most of them are only applicable with regard to neural networks.

3.3.4 Conclusion

In this chapter, we have explored several modern neural network software-libraries, particularly those that support state of the art techniques (including deep learning.)

Most of the libraries are very similar. Support for deep models and GPU-acceleration are relatively common between the alternatives (at least the modern and/or specialized ones)—if not out of the box, then by third-parties. Many of the newer options among the software are either partially implemented in or had bindings to a succint, expressive language. Also common (especially among the newer libraries) are relatively liberal, open source licenses (e.g. BSD), and having very active communities of users/developers contributing to further discussion and development.

Pylearn2 values ease of experimentation over ease of use. It has minimal documentation, but it makes it very easy to experiment with powerful models.

Torch7 is also a serious contender, and is very efficient. It has a large selection of third-party packages, and many leading research-groups contributes to its development.

Both Pylearn2 and Torch7 are modern research-tools that enable deep learning and extremely good performance because of (almost transparent) GPU-acceleration.

Matlab is tried and true, documentation is available and one is sure to find available expertise and code if one needs help. State of the art performance and models, however, are not guaranteed out of the box, as MATLAB is not as close to the bleeding edge as other tools. There is a deep learning toolbox available from a third party, but its source code repository has currently not been updated for a year, and the field of neural networks (and deep learning in particular) is in rapid development.

While Weka may be of use in exploratory stages, it is not really usable for deep learning, and is indeed not very focused on neural networks.

Scikit-learn has a broad focus when it comes to machine learning techniques, and is also very much approachable for non-expert users. However, it does not offer neither the efficiency nor the large collection of neural network-related models that some of the alternatives do, as it is currently mostly limited to unsupervised learning in the form of RBMs[20].

As he most important criteria in this project were ease of experimentation, support for modern techniques and efficiency of implementation, Pylearn2 was chosen as the software tool for neural networks models.

3.4 Custom tools

During the work described in this thesis, a few custom tools, programs and scripts were produced. Here, we give a short description of the most central tools, which can be found as appendices to this thesis.

The tools described here, as well as others, can be found in the same public repository[21] as described in Chapter 3.1.7.

[20] Restricted Boltzmann Machines

[21] `https://github.com/eivind88/master_code`

3.4.1 Dataset Converter

The dataset converter A is a standalone command line interface application that performs concurrent conversion of the dataset according to specifications.

3.4.2 Pylearn2 Dataset Class

The dataset class B is a python class for use with the Pylearn2 library, that loads the converted ADNI-dataset and automatically splits it into training- test- and validation-subsets.

Chapter 4

Results

4.1 Introduction

This chapter presents the outcomes of the research work described in the previous chapter.

All experiments were run with a 70%/15%/15% dataset split (training/testing/validation). This was done to ensure there was enough test/validation-data available with the dataset variations with fewer instances.

As described in Chapter 3.1.3, we used the following schemes for merging classes (i.e. diagnostic groups) in order to facilitate both a 3-way and three binary classifiers:

1. Normal / MCI / AD
2. Normal / Other
3. Other / MCI
4. Other / AD

4.1.1 Decision Trees

In some cases the number of boosting trials may have been reduced (if the last classifier was very inaccurate) or abandoned entirely (if there were too few classifiers). The decision trees were not cross validated.

4.1.2 Neural Networks

All networks were multilayer perceptrons using tanh as activation function, with the termination criteria: maximum 10000 epochs; no decrease in misclassification rate in 25 epochs; misclassification rate lower than 0.01. All networks were run with 3-fold stratified cross validation.

Baseline 1 hidden layer (20 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), no regularization

Baseline_faster 1 hidden layer (20 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-2), no regularization

More_neurons 1 hidden layer (50 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), no regularization

Two_layers 2 hidden layers (16 hidden neurons, tanh)(8 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), no regularization

Two_layers_big-batch 2 hidden layers (16 hidden neurons, tanh)(8 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 100, learning rate of 5e-3), no regularization

Larger_layers 2 hidden layers (150 hidden neurons, tanh)(300 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), no regularization

Three_layers 3 hidden layers (30 hidden neurons, tanh)(90 hidden neurons, tanh)(60 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), no regularization

Three_layers_dropout 3 hidden layers (30 hidden neurons, tanh)(90 hidden neurons, tanh)(60 hidden neurons, tanh), Stochastic Gradient Desccent (batch size of 10, learning rate of 5e-3), Dropout regularization (probability of .5, scaled by 2)

4.2 Merging scheme 1

Under this merging scheme, the diagnostic groups (i.e. classes) in the dataset were merged so that the resulting classes were: Normal, MCI, AD.

4.2.1 PCA

Decision Trees

Training Error	Test Error	Description
1.0%	54.5%	N/A
17.3%	54.1%	Rules
0.0%	47.0%	Boosting (10 trials)
0.0%	51.2%	Boosting (10 trials), Rules
0.0%	45.3%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	50.0%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	47.0%	Boosting (20 trials), No Pruning
0.0%	48.6%	Boosting (20 trials), No Pruning, Rules
0.0%	53.3%	Boosting (20 trials), No Pruning, Winnowing
0.0%	51.4%	Boosting (20 trials), No Pruning, Winnowing, Rules
0.0%	45.1%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
1.0%	49.0%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.1: Results from experiment with decision trees on merged data (Normal/MCI/AD) reduced via PCA.

From the many low training errors in Table 4.1, we see that overfitting on the training data is a big problem. Both boosting and winnowing appear to help give the classifier greater accuracy.

Neural Networks

Training Error	Testing Error	Valid Error	Description
49.2%	57.5%	57.2%	Baseline
48.0%	57.4%	58.6%	Baseline_faster
48.7%	59.2%	57.6%	More_neurons
55.0%	55.0%	55.0%	Two_layers
55.0%	55.0%	55.0%	Two_layers_big_batch
54.9%	55.0%	54.9%	Larger_layers
55.0%	55.0%	55.0%	Three_layers
55.0%	55.0%	55.0%	Three_layers_dropout

Table 4.2: Results from experiment with neural networks on merged data (Normal/MCI/AD) reduced via PCA.

The neural networks in Table 4.2 consistently perform worse than chance.

4.2.2 Histogram

Decision Trees

Training Error	Test Error	Description
7.2%	49.5%	N/A
26.5%	49.2%	Rules
0.1%	45.6%	Boosting (10 trials)
26.7%	54.0%	Boosting (10 trials), Rules
0.0%	44.0%	Boosting (10 trials), Pruning (confidence level 80%)
12.3%	49.0%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	40.8%	Boosting (20 trials), No Pruning
9.4%	47.9%	Boosting (20 trials), No Pruning, Rules
0.0%	47.8%	Boosting (20 trials), No Pruning, Winnowing
27.4%	49.5%	Boosting (20 trials), No Pruning, Winnowing, Rules
0.0%	42.6%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
15.8%	46.9%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.3: Results from experiment with decision trees on merged data (Normal/MCI/AD) reduced via histogram.

The results are improving in Table 4.3, and the test errors are now consistently better than chance with decision trees.

Neural Networks

Training Error	Testing Error	Valid Error	Description
53.7%	54.1%	53.4%	Baseline
52.9%	54.3%	53.2%	Baseline_faster
52.6%	53.8%	53.0%	More_neurons
53.6%	53.6%	53.6%	Two_layers
53.6%	53.6%	53.6%	Two_layers_big_batch
53.6%	53.6%	53.6%	Larger_layers
53.6%	53.6%	53.6%	Three_layers
53.6%	53.6%	53.6%	Three_layers_dropout

Table 4.4: Results from experiment with neural networks on merged data (Normal/MCI/AD) reduced via histogram.

The neural network results in Table 4.4 have improved slightly, but they still have a valid error of over 50% in all cases. The fact that many of the error rates have stopped at precisely the same spot indicates that there might be a problem with the networks getting stuck in local minima.

4.2.3 Downscaling

Decision Trees

Training Error	Test Error	Description
0.8%	50.3%	N/A
16.3%	50.3%	Rules
0.0%	39.8%	Boosting (10 trials)
0.0%	44.4%	Boosting (10 trials), Rules
0.0%	47.4%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	42.1%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	41.5%	Boosting (20 trials), No Pruning
0.0%	46.2%	Boosting (20 trials), No Pruning, Rules
31.4%	45.0%	Boosting (20 trials), No Pruning, Winnowing
1.8%	43.9%	Boosting (20 trials), No Pruning, Winnowing, Rules
0.0%	43.9%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
4.3%	47.4%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.5: Results from experiment with decision trees on merged data (Normal/MCI/AD) reduced via downscaling.

In Table 4.5, Downscaling yielded better results than the best with histograms, but also worse than the worst. Training errors hint at overfitting.

Neural Networks

Training Error	Testing Error	Valid Error	Description
59.3%	57.0%	55.8%	Baseline
51.8%	61.0%	55.0%	Baseline_faster
45.2%	60.8%	55.8%	More_neurons
57.0%	57.0%	57.0%	Two_layers
57.0%	57.0%	57.0%	Two_layers_big_batch
57.0%	57.0%	57.0%	Larger_layers
57.0%	57.0%	57.0%	Three_layers
57.0%	57.0%	57.0%	Three_layers_dropout

Table 4.6: Results from experiment with neural networks on merged data (Normal/MCI/AD) reduced via downscaling.

Error-rates have risen from the improvement seen with histograms in Table 4.6.

4.3 Merging scheme 2

Under this merging scheme, the diagnostic groups (i.e. classes) in the dataset were merged so that the resulting classes were: Normal, Other.

4.3.1 PCA

Decision Trees

Training Error	Test Error	Description
1.0%	41.9%	N/A
5.5%	36.8%	Rules
0.0%	35.0%	Boosting (10 trials)
0.0%	39.2%	Boosting (10 trials), Rules
0.0%	34.8%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	35.6%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	29.5%	Boosting (20 trials), No Pruning
0.0%	29.9%	Boosting (20 trials), No Pruning, Rules
29.6%	34.8%	Boosting (20 trials), No Pruning, Winnowing
20.8%	36.4%	Boosting (20 trials), No Pruning, Winnowing, Rules
9.5%	35.4%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
27.9%	35.8%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.7: Results from experiment with decision trees on merged data (Normal/Other) reduced via PCA.

Table 4.7 shows lower test error rates than the same combination of PCA for dimensional reduction and decision trees for training did inTtable 4.1, but training errors are still low,

indicating a problem with overfitting.

Neural Networks

Training Error	Testing Error	Valid Error	Description
34.1%	35.1%	35.9%	Baseline
31.9%	36.2%	35.6%	Baseline_faster
32.8%	35.1%	34.2%	More_neurons
33.3%	33.3%	33.3%	Two_layers
33.3%	33.3%	33.3%	Two_layers_big_batch
33.3%	33.3%	33.3%	Larger_layers
33.3%	33.3%	33.3%	Three_layers
33.3%	33.3%	33.3%	Three_layers_dropout

Table 4.8: Results from experiment with neural networks on merged data (Normal/Other) reduced via PCA.

Results are consistently better in Table 4.8 when compared to Table 4.2, but there still appears to be a big problem with local minima.

4.3.2 Histogram

Decision Trees

Training Error	Test Error	Description
28.5%	34.7%	N/A
28.6%	32.2%	Rules
27.6%	34.7%	Boosting (10 trials)
30.5%	33.7%	Boosting (10 trials), Rules
27.3%	34.2%	Boosting (10 trials), Pruning (confidence level 80%)
30.2%	35.3%	Boosting (10 trials), Pruning (confidence level 80%), Rules
27.4%	33.4%	Boosting (20 trials), No Pruning
29.0%	32.5%	Boosting (20 trials), No Pruning, Rules
34.2%	34.5%	Boosting (20 trials), No Pruning, Winnowing
34.2%	34.5%	Boosting (20 trials), No Pruning, Winnowing, Rules
34.2%	34.5%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
34.2%	34.5%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.9: Results from experiment with decision trees on merged data (Normal/Other) reduced via histogram.

In Table 4.9, results have gotten a tiny bit better since Table 4.7.

Neural Networks

Training Error	Testing Error	Valid Error	Description
33.2%	33.2%	33.2%	Baseline
33.2%	33.2%	33.2%	Baseline_faster
33.0%	33.3%	33.1%	More_neurons
33.2%	33.2%	33.2%	Two_layers
33.2%	33.2%	33.2%	Two_layers_big_batch
33.2%	33.2%	33.2%	Larger_layers
33.2%	33.2%	33.2%	Three_layers
33.2%	33.2%	33.2%	Three_layers_dropout

Table 4.10: Results from experiment with neural networks on merged data (Normal/Other) reduced via histogram.

Table 4.10 indicated a problem with local minima even stronger than before, with all error rates being more or less equal.

4.3.3 Downscaling

Decision Trees

Training Error	Test Error	Description
0.5%	33.3%	N/A
6.3%	33.9%	Rules
0.0%	35.7%	Boosting (10 trials)
0.0%	35.7%	Boosting (10 trials), Rules
0.0%	28.7%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	36.8%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	46.8%	Boosting (20 trials), No Pruning
0.0%	34.5%	Boosting (20 trials), No Pruning, Rules
6.0%	41.5%	Boosting (20 trials), No Pruning, Winnowing
24.6%	36.8%	Boosting (20 trials), No Pruning, Winnowing, Rules
5.0%	38.6%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
13.3%	36.3%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.11: Results from experiment with decision trees on merged data (Normal/Other) reduced via downscaling.

Test errors in Table 4.11 are more or less comparable to those in Table 4.9, but with a tiny bit more variance. The training errors are once again very low, indicating that overfitting is taking place.

Neural Networks

Training Error	Testing Error	Valid Error	Description
36.2%	36.0%	34.7%	Baseline
36.7%	34.9%	34.9%	Baseline_faster
33.4%	36.7%	35.2%	More_neurons
35.4%	35.4%	35.4%	Two_layers
35.4%	35.4%	35.4%	Two_layers_big_batch
35.4%	35.4%	35.4%	Larger_layers
35.4%	35.4%	35.4%	Three_layers
35.4%	35.4%	35.4%	Three_layers_dropout

Table 4.12: Results from experiment with neural networks on merged data (Normal/Other) reduced via downscaling.

??

Table **??** show that local minima is still a big problem.

4.4 Merging scheme 3

Under this merging scheme, the diagnostic groups (i.e. classes) in the dataset were merged so that the resulting classes were: Other, MCI.

4.4.1 PCA

Decision Trees

Training Error	Test Error	Description
1.7%	42.9%	N/A
11.2%	44.7%	Rules
0.0%	38.6%	Boosting (10 trials)
0.0%	42.9%	Boosting (10 trials), Rules
0.0%	42.5%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	40.9%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	36.0%	Boosting (20 trials), No Pruning
0.0%	38.4%	Boosting (20 trials), No Pruning, Rules
30.4%	47.0%	Boosting (20 trials), No Pruning, Winnowing
20.0%	44.7%	Boosting (20 trials), No Pruning, Winnowing, Rules
29.8%	45.7%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
24.3%	45.3%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.13: Results from experiment with decision trees on merged data (Other/MCI) reduced via PCA.

The results in Table 4.13 are very similar to those in Table 4.7, indicating the same issue with overfitting.

Neural Networks

Training Error	Testing Error	Valid Error	Description
44.9%	44.9%	44.9%	Baseline
42.3%	45.6%	44.9%	Baseline_faster
38.6%	45.9%	44.3%	More_neurons
44.9%	44.9%	44.9%	Two_layers
44.9%	44.9%	44.9%	Two_layers_big_batch
43.7%	44.9%	44.8%	Larger_layers
44.9%	44.9%	44.9%	Three_layers
44.9%	44.9%	44.9%	Three_layers_dropout

Table 4.14: Results from experiment with neural networks on merged data (Other/MCI) reduced via PCA.

Table 4.14 shows that local minima is still an issue, and the networks now seem to get stuck even earlier than before.

4.4.2 Histogram

Decision Trees

Training Error	Test Error	Description
31.4%	45.8%	N/A
40.6%	42.3%	Rules
40.1%	41.7%	Boosting (10 trials)
44.1%	44.9%	Boosting (10 trials), Rules
39.9%	41.2%	Boosting (10 trials), Pruning (confidence level 80%)
36.0%	42.9%	Boosting (10 trials), Pruning (confidence level 80%), Rules
33.8%	39.1%	Boosting (20 trials), No Pruning
43.5%	43.4%	Boosting (20 trials), No Pruning, Rules
42.0%	44.9%	Boosting (20 trials), No Pruning, Winnowing
42.0%	44.9%	Boosting (20 trials), No Pruning, Winnowing, Rules
42.0%	44.9%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
41.3%	46.6%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.15: Results from experiment with decision trees on merged data (Other/MCI) reduced via histogram.

In Table 4.15, training error once again increases, indicating that the apparent overfitting is less of a problem when histogram is used for dimensionality reduction with decision trees.

Neural Networks

Training Error	Testing Error	Valid Error	Description
45.3%	46.8%	45.8%	Baseline
45.1%	47.7%	44.1%	Baseline_faster
44.5%	48.0%	44.4%	More_neurons
46.4%	46.4%	46.4%	Two_layers
46.4%	46.4%	46.4%	Two_layers_big_batch
42.6%	47.9%	44.5%	Larger_layers
46.4%	46.4%	46.4%	Three_layers
46.4%	46.4%	46.4%	Three_layers_dropout

Table 4.16: Results from experiment with neural networks on merged data (Other/MCI) reduced via histogram.

The networks appear to become stuck around the same place in Table 4.16 as in Table **??**, but there is more variance in errors this time around.

4.4.3 Downscaling

Decision Trees

Training Error	Test Error	Description
1.3%	35.7%	N/A
5.8%	40.4%	Rules
0.0%	28.7%	Boosting (10 trials)
0.0%	35.7%	Boosting (10 trials), Rules
0.0%	36.8%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	29.8%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	30.4%	Boosting (20 trials), No Pruning
0.0%	35.1%	Boosting (20 trials), No Pruning, Rules
14.8%	40.4%	Boosting (20 trials), No Pruning, Winnowing
6.0%	47.4%	Boosting (20 trials), No Pruning, Winnowing, Rules
16.6%	32.7%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
19.1%	40.9%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.17: Results from experiment with decision trees on merged data (Other/MCI) reduced via downscaling.

Table 4.17 shows some issue with overfitting, but rest results at a low for this merging scheme.

Neural Networks

Training Error	Testing Error	Valid Error	Description
43.0%	43.0%	43.0%	Baseline
39.7%	45.0%	40.9%	Baseline_faster
33.9%	44.7%	42.0%	More_neurons
43.0%	43.0%	43.0%	Two_layers
43.0%	43.0%	43.0%	Two_layers_big_batch
43.0%	43.0%	43.0%	Larger_layers
43.0%	43.0%	43.0%	Three_layers
43.0%	43.0%	43.0%	Three_layers_dropout

Table 4.18: Results from experiment with neural networks on merged data (Other/MCI) reduced via downscaling.

Most of the networks in Table 4.18 appear to become stuck around the same local minima.

4.5 Merging scheme 4

Under this merging scheme, the diagnostic groups (i.e. classes) in the dataset were merged so that the resulting classes were: Other, AD.

4.5.1 PCA

Decision Trees

Training Error	Test Error	Description
0.9%	26.2%	N/A
3.1%	24.6%	Rules
0.0%	14.2%	Boosting (10 trials)
0.2%	17.3%	Boosting (10 trials), Rules
0.1%	16.5%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	20.9%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	14.4%	Boosting (20 trials), No Pruning
0.0%	17.9%	Boosting (20 trials), No Pruning, Rules
0.0%	16.9%	Boosting (20 trials), No Pruning, Winnowing
16.0%	19.3%	Boosting (20 trials), No Pruning, Winnowing, Rules
0.1%	16.5%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
19.1%	21.5%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.19: Results from experiment with decision trees on merged data (Other/AD) reduced via PCA.

Table 4.19 present drastically lowered test errors, though training errors indicate that overfitting is once again an issue.

Neural Networks

Training Error	Testing Error	Valid Error	Description
21.8%	21.8%	21.8%	Baseline
21.8%	21.8%	21.8%	Baseline_faster
21.8%	21.8%	21.8%	More_neurons
21.8%	21.8%	21.8%	Two_layers
21.8%	21.8%	21.8%	Two_layers_big_batch
21.8%	21.8%	21.8%	Larger_layers
21.8%	21.8%	21.8%	Three_layers
21.8%	21.8%	21.8%	Three_layers_dropout

Table 4.20: Results from experiment with neural networks on merged data (Other/AD) reduced via PCA.

Local minima still poses a big problem in Table 4.20. This time all the errors are exactly the same.

4.5.2 Histogram

Decision Trees

Training Error	Test Error	Description
20.2%	22.0%	N/A
19.2%	21.1%	Rules
19.2%	21.1%	Boosting (10 trials)
19.2%	21.1%	Boosting (10 trials), Rules
19.6%	21.8%	Boosting (10 trials), Pruning (confidence level 80%)
19.2%	22.7%	Boosting (10 trials), Pruning (confidence level 80%), Rules
18.2%	22.4%	Boosting (20 trials), No Pruning
17.7%	23.1%	Boosting (20 trials), No Pruning, Rules
20.1%	22.3%	Boosting (20 trials), No Pruning, Winnowing
20.1%	22.3%	Boosting (20 trials), No Pruning, Winnowing, Rules
20.1%	22.3%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
20.1%	22.3%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.21: Results from experiment with decision trees on merged data (Other/AD) reduced via histogram.

In Table 4.21, errors rise a bit again.

Neural Networks

Training Error	Testing Error	Valid Error	Description
20.4%	20.4%	20.4%	Baseline
20.3%	20.5%	20.4%	Baseline_faster
20.4%	20.4%	20.4%	More_neurons
20.4%	20.4%	20.4%	Two_layers
20.4%	20.4%	20.4%	Two_layers_big_batch
20.4%	20.4%	20.4%	Larger_layers
20.4%	20.4%	20.4%	Three_layers
20.4%	20.4%	20.4%	Three_layers_dropout

Table 4.22: Results from experiment with neural networks on merged data (Other/AD) reduced via histogram.

Table 4.22 show errors that are lower than in Table 4.20, but local minima is still a very big problem.

4.5.3 Downscaling

Decision Trees

Training Error	Test Error	Description
0.3%	30.4%	N/A
2.3%	22.2%	Rules
0.0%	20.5%	Boosting (10 trials)
0.0%	17.0%	Boosting (10 trials), Rules
0.0%	18.1%	Boosting (10 trials), Pruning (confidence level 80%)
0.0%	19.9%	Boosting (10 trials), Pruning (confidence level 80%), Rules
0.0%	15.8%	Boosting (20 trials), No Pruning
0.0%	21.6%	Boosting (20 trials), No Pruning, Rules
0.0%	17.5%	Boosting (20 trials), No Pruning, Winnowing
15.6%	20.5%	Boosting (20 trials), No Pruning, Winnowing, Rules
15.8%	24.0%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing
11.8%	22.2%	Boosting (20 trials), Pruning (confidence level 80%), Winnowing, Rules

Table 4.23: Results from experiment with decision trees on merged data (Other/AD) reduced via downscaling.

Once again, variance increases between histograms in Table 4.21 and downscaling in Table 4.23. Some training errors fall to 0.0% again, indicating overfitting.

Neural Networks

Training Error	Testing Error	Valid Error	Description
21.6%	21.6%	21.6%	Baseline
20.6%	22.1%	21.1%	Baseline_faster
19.6%	21.6%	20.9%	More_neurons
21.6%	21.6%	21.6%	Two_layers
21.6%	21.6%	21.6%	Two_layers_big_batch
21.6%	21.6%	21.6%	Larger_layers
21.6%	21.6%	21.6%	Three_layers
21.6%	21.6%	21.6%	Three_layers_dropout

Table 4.24: Results from experiment with neural networks on merged data (Other/AD) reduced via downscaling.

Table 4.24 shows the same signs of local minima.

Chapter 5

Discussion

5.1 Answering the research questions

How can a classifier be trained to differentiate between structural MR images of Alzheimer's Disease and other diagnostic groups?

5.1.1 What kind of dimensional reduction is most suited for the problem at hand?

It is not easy to say what impact the form and degree of dimensional reduction (in combination with the externally imposed size-constraint) has had, as the data itself is very complex (and still somewhat highly dimensional in the case of the downscaled variants), not to mention that there really are quite few instances in the dataset.

There does not appear to be a clear winner from the current experimental results. In the experiments, histogram and downscaling usually lead to better results than PCA (with one exception), but none of the two performed consistently better than the other. Although, the single best results was achieved using PCA (see table 4.19, row 3).

PCA lead to the single best result (see table 4.19, row 3) when used with decision trees and distinguishing Alzheimer's from all other classes, but was not consistently better than the alternatives.

Histogram yielded comparatively good performance, at least when taking amount of dimensional (and data size) reduction into consideration.

Because of this drastic reduction, combined with the size constraints mentioned in Chapter 3.1.1, there were fewer learning instances in the dataset variants reduced via downscaling than those reduced via histograms (all 2182 vs. 570 images). While the two start out with relatively similar performance with decision trees (when looking at the tables in the order they are presented), downscaled images yield slightly better results than histograms with later merging schemes. Under merging scheme 3 (see table 4.15 and 4.17), and maybe 4 (see table 4.21 and 4.23) in particular, downscaling seems to lead to higher variability in test errors when compared to histograms, at least when used with decision trees (46.6-39.1% vs. 40.9-28.7%, and 23.1-21.1% vs. 30.4-17.0%), though this may be because of the lack of cross validation.

5.1.2 Which machine learning approach yields the best results on the dataset?

A direct comparison between machine learning approaches reveal that decision trees always performed slightly to moderately better than neural networks in the experiments.

In the case of decision trees, rulesets will generally be easier to understand for humans than decision trees, in addition to often having fewer rules than corresponding trees have leaves. In this case, there were not many sensible (or indeed easily interpretable) rules in the decision tree output, since the rules generally dealt with pixel-values or related concepts (i.e. bin- or component-values). Performance-wise, rules have yielded worse (0.0 - 8.2%) performance than trees. Boosting seems to make a positive difference across the board. Winnowing, on the other hand, seems to yield slightly worse performance. Pruning also seems to make a slightly negative contribution.

At present, the performance achieved with neural networks is generally not very good —in fact, they range from just worse than chance to an error rate of about 20 percent, and generally appeared to struggle with local minima.

The best results in the experiments came from decision trees (see table 4.19, row 3) under the 4th merging scheme. While (deep) neural networks intuitively should be a better fit for the problem at hand, they did not perform very well in the experiments. This is probably because we did not have time to experiment more with regard to complex architectures, costs and specialized activation functions. Deep convolutional networks, for instance, would likely perform better on this type of problem, as they have performed very well on image-related tasks in recent years.

5.1.3 How does reformulating the multiclass classification problem as binary classification affect the models' performance?

In general, the generalizing ability of most of the generated models appears to be poor to moderate, as is evident from the relatively high error-rates in many of the tables.

However, comparing results from decision trees trained on PCA-reduced (generally worst amongst dimensional reduction techniques in these experiments) data between merging scheme 1 (see table 4.1 and 4.2) and 2 (see table 4.7 and 4.8) reveals a drastic difference in test error rates (54.5-45.1% vs. 41.9-29.5% with regard to decision trees). PCA test error rates from merging scheme 4 (see table 4.19 and 4.20) are even lower (26.2-14.2% with regard to decision trees). Using merging scheme 3 (see table 4.13 and 4.14), on the other hand, the errors are somewhat higher (47.0-36.0% regarding decision trees), though this is unsurprising, as these classifiers distinguish between MCI (which is known to be associated with smaller changes in brain stucture, as it is a prodromal form of AD) and all other diagnostic groups.

In general, the misclassification error was reduced by reformulating the learning problem as binary classification via class merging schemes. Error was reduced moderately (around 5-15%) using merging scheme 2 and 3. With merging scheme 4, error was reduced drastically (around 20-25%). Merging scheme #4 (Other/AD) yielded the single best percentage as well as overall best performance (see table 4.19, row 3).

5.1.4 How does the different results compare to similar research?

Though direct comparison between studies is difficult, as they often use different datasets, preprocessing techniques, and eventual approaches to dimensional reduction, as well as somewhat different measurements and methods regarding comparisons, we will attempt to indicate the typical accuracy achieved.

The best binary classifier (85.8% correct classification) is in the general range of directly comparable research (machine learning on AD-classification from MRI; some on the ADNI-dataset), though recent research [55] achieved a [55] percentage on correct classification via downscaling, feature extraction via autoencoders and deep convolutional neural networks and binary classifiers.

Klöppel et al. performed a direct comparison of diagnostic accuracy between radiologists and computer methods in 2008, using a linear SVM to classify pathologically verified AD patients correctly using whole brain images. Their solution classified correctly up to 96% of the patients in their dataset [39].

In 2009, Magnin et al. used SVMs to discriminate between AD patients and control subjects using whole-brain (1.5-T) MRI. Their solution depended upon heavy preprocessing, parcellating the brain into Regions Of Interest by using a previously developed anatomically labelled template of the brain, and created a mask to exclude voxels of the skull. They obtained "94.5% mean correct classification for AD and control subjects (mean specificity, 96.6%; mean sensitivity, 91.5%)" [45]

Gray achieved a classification accuracy of 90% [28] on the task of separating AD patients from healthy controls in 2012, noting that this was in line with the current state-of-the-art machine learning techniques (both single- and multi-modality). The classifier also had an accuracy of 76% for separating MCI from healthy controls.She did this by training an SVM on pairwise similarity measures derived from random forest classifiers.

In 2013, Gupta, Ayhan and Maida used a sparse autoencoder for feature extraction, and then trained a convolutional neural network on the bases. Training this model with MRI-images, they achieved an accuracy of 93.8% on AD vs. healthy controls, 83.3% on MCI vs healthy controls, and 86.3% on AD vs. MCI [29]. They also trained a three-way classifier, which achieved an accuracy of 78.2%.

Payan and Montana's 2015 paper [55] describes their experiments with sparse autoencoders for feature-extraction and 2D and 3D deep convolutional neural networks for training. They achieved a prediction accuracy of 85.53% (2D convolutions) and 89.47% (3D convolutions) with their 3-way classifier, and binary classifiers achieved 95.39% (2D convolutions) and 95.39% (3D convolutions) on AD vs. HC, 82.24% (2D convolutions) and 86.84% (3D convolutions) on AD vs. MCI, and 90.13% (2D convolutions) and 92.11% (3D convolutions) on HC vs. MCI.

5.2 Limitations and threats to validity

The dataset we used was relatively small (few instances). While hold-one-out cross validation should have been applied consistently to the machine learning techniques, because of the time it took to run the experiments, there was no cross-validation performed on the decision trees. Arguably, the sampling could have been better. Instead of going by subject ID until reaching the maximum processed dataset size of about 2GB, sampling should have been stratified, increasing the representativeness of the sample.

5.3 Takeaways

Overall, both machine learning techniques exhibited problems during these experiments.

Overfitting was problematic for decision trees, in particular when they were training on dataset that had used PCA for dimensionality reduction. Neural networks struggled with getting stuck in local minima, as can be seen by the many networks stopping in the same place.

The fact that the relatively simple decision tree method gave relatively high predictive abilities was somewhat surprising, even though the tree-based ensemble method random forests have performed well in other research.

The methods of dimensional reduction we used were well known approaches, some of which drastically reduced the information present in each instance in the dataset. Surprisingly, downscaling yielded relatively good results compared to alternative methods of dimensionality reduction in our experiments, even though the downscaled datasets had a total of 570 images, while PCA had 1760, and histogram all 2182. Overall, downscaling seemed to yield consistently better results (though with higher variance within experiment), until it was beat by PCA in the single best performance figure using merging scheme 4 (see table 4.19, row 3).

Our merging (e.g. Others vs. AD), as opposed to just using two classes (e.g. Normal vs. AD), likely makes some differences in the experiment outcomes. Firstly, it leads to variance in number of instances between groups (i.e. *many* more members of "Other" than "AD"). But, the models could also be said to learn to differentiate between pseudogroups on the basis of the difference beteween Alzheimer's affected brains and any other brain, (i.e. defining features; what it *is*) rather than the difference between Alzheimer's affected brains and a healthy brain (i.e. disqualifying features; what it *is not*).

In these experiments, decision trees always performed better than neural networks, though this is possibly because of lacking experimentation with hyperparameters and architecture.

Merging classes in order to perform binary classification improved results drastically.

Performance wise, the best classifier was in the general range of directly comparable research, though in the lower end. The best classifier produced during these experiments was one that could distinguish Alzheimer's Patients' brains scans from those belonging other (i.e. MCI or normal control) diagnostic groups, using PCA for dimensional reduction and decision tree models for learning. This classifier had a test error rate of 14.2%.

We think that the research questions are fully answered by these findings, with the exception of research question 1.2, regarding which machine learning approach is better. It may be reasonable to suspect that other models are better suited for use with the distinguishing features of the diseased brains (which are visually observable), such as maybe a deeper neural network architecture (perhaps involving convolution.) Experimentation with network architecture, costs and activation functions, for instance, would likely improve upon the results presented here. Additionally, the same experiments should be run on SVMs and random forests for comparison. Each experiment with merged classes could also have been compared to selective sampling, i.e. picking instances from classes "AD" and "Normal" and comparing the results from using these with the results from merging scheme 4 ("Other" vs. "AD"). Some more experimentation with alternative methods of dimensional reduction and/or variation upon the learning techniques used here may yield slightly better results, but very good results may be contingent upon greater changes in

approach. We could have experimented with the number of bins and components kept, or the downscaling resolution in the experiments.

Chapter 6

Conclusion

6.1 Conclusion

In this thesis we have evaluated how well common machine learning techniques can distinguish between MR images of healthy brains, mildly cognitively impaired brains and brains affected by Alzheimer's Disease. Standard machine learning algorithms like decision tree- and rule-based ones, as well as neural networks were applied to the standardized ADNI magnetic resonance imaging dataset. The approach taken was an experimental and exploratory one, comparing results from machine learning different techniques in combination with different methods of dimensional reduction and different numbers of diagnostic groups (using different merging schemes). Random forests, support vector machines, and recently deep neural networks, have shown relatively good performance on similar problems. As Alzheimer's is a rapidly growing worldwide problem and the fields of computer aided diagnosis, computer vision and deep learning makes strides, the problem at hand (i.e. automatic classification of Alzheimer's from MRI) seems to play into the inherent strengths of some newer techniques.

From our experiments, we have seen that decision trees seems to be a viable machine learning approach to the problem of Alzheimer's classification via structural MRI — which is somewhat surprising, given its apparent simplicity — though its success is probably contingent upon use with an effective method of dimensionality reduction. These findings needs to be further examined with cross validation.

Our research question, *"How can a classifier be trained to differentiate between structural MR images of Alzheimer's Disease and other diagnostic groups?"* was answered via the following subquestions:

What kind of dimensional reduction is most suited for the problem at hand? While the best classification rate was achieved on a dataset that used PCA for dimensionality reduction, there was no clear winner that consistently performed best amongst the different methods.

Which machine learning approach yields the best results on the dataset? In our experiments, decision trees performed best. Neural networks generally performed poorly, but this may be because of lack of experimentation with regard to hyperparameters and architecture.

How does reformulating the multiclass classification problem as binary classification affect the models' performance? Binary classification via merging some classes had a positive effect on the performance of the machine learning models. When using this technique to learn the difference between the classes "Alzheimer's" and "Other", a classifier achieved a correct classification rate of 85.8%.

How does the different results compare to similar research? While our best results are in the general range of directly comparable research, it is in the lower end.

The main thing that this report demonstrates is that relatively simple machine learning techniques like decision trees are viable candidates even for seemingly complex problems like automatic classification of Alzheimer's disease from brain scans — provided that fitting methods of dimensional reduction are used. We also demonstrate that posing the learning problem as a binary classification task rather than a three-way classification task dramatically improves performance of the classifier.

6.1.1 Future Work

We suggest the following potential avenues of investigation for future experiments:

Variations on binary classification Comparing merging vs. using two specific classes (as is more widely done in literature).

Experiment with dimensional reduction Amount of bins/components to keep or resolution to resize to.

Little to no dimensional reduction Might work with newer models (i.e. deeper architectures). Some work has already been done with similar problems.

Train on additional data Supplement training data with AddNeuroMed and Oasis Brains, or apply systematic deformations in order to artificially inflate the dataset's size (as in number of instances). Could reduce chance of overfitting, andincrease generalizability ability of trained model.

Train a multimodal classifier A multimodal approach to predict MCI-AD conversion will probably offer better sensitivity/specificity than methods based on isolated modalities.

State of the art techniques Deep learning is a field in rapid development, and newer tips and techniques would likely help performance. 3D convolutional deep neural network seem particularly well suited. Maxout and Dropout might increase performance, as maxout was designed with dropout in mind, and the combination of the two have yielded good results in research.

Bibliography

[1] *2.2 Basic Principles of MRI*. URL: http://www.cs.sfu.ca/~stella/papers/blairthesis/main/node11.html.

[2] *About LONI | Laboratory of Neuro Imaging*. URL: http://www.loni.usc.edu/about_loni/.

[3] *ADNI | About*. URL: http://adni.loni.usc.edu/about/.

[4] *ADNI | Background & Rationale*. URL: http://adni.loni.usc.edu/study-design/background-rationale/.

[5] *ADNI | MRI Pre-processing*. URL: http://adni.loni.usc.edu/methods/mri-analysis/mri-pre-processing/.

[6] *ADNI | Standardized MRI Data Sets*. URL: http://adni.loni.usc.edu/methods/mri-analysis/adni-standardized-data/ (visited on 03/14/2014).

[7] Marilyn S Albert et al. "The diagnosis of mild cognitive impairment due to Alzheimer's disease: recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease". In: *Alzheimer's & dementia: the journal of the Alzheimer's Association* 7.3 (May 2011), pp. 270–279. ISSN: 1552-5279. DOI: 10.1016/j.jalz.2011.03.008.

[8] *Alzheimer's & Dementia Causes, Risk Factors | Research Center*. Alzheimer's Association. URL: http://www.alz.org/research/science/alzheimers_disease_causes.asp (visited on 04/08/2014).

[9] *Alzheimer's Disease & Dementia | Alzheimer's Association*. URL: http://www.alz.org/alzheimers_disease_what_is_alzheimers.asp (visited on 04/02/2014).

[10] Yoshua Bengio. "Learning deep architectures for AI". In: *Foundations and trends® in Machine Learning* 2.1 (2009), pp. 1–127. URL: http://dl.acm.org/citation.cfm?id=1658424 (visited on 06/03/2014).

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8 (2013), pp. 1798–1828. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6472238 (visited on 10/14/2014).

[12] Yoshua Bengio et al. "Greedy Layer-Wise Training of Deep Networks". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 153–160. URL: http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf.

[13] L Berg et al. "Neuropathological indexes of Alzheimer's disease in demented and nondemented persons aged 80 years and older". In: *Archives of neurology* 50.4 (Apr. 1993), pp. 349–358. ISSN: 0003-9942.

[14] Charlotte Bernard et al. "Time course of brain volume changes in the preclinical phase of Alzheimer's disease". In: *Alzheimer's & Dementia* 10.2 (Mar. 2014), 143–151.e1. ISSN: 15525260. DOI: 10.1016/j.jalz.2013.08.279. URL: http://ezproxy.hiof.no:2162/pubmed/?term=Time+course+of+brain+volume+changes+in+the+preclinical+phase+of+Alzheimer%27s+disease (visited on 04/23/2014).

[15] Leonardo Bottaci et al. "Artificial neural networks applied to outcome prediction for colorectal cancer patients in separate institutions". In: *The Lancet* 350.9076 (1997), pp. 469–472. URL: http://www.sciencedirect.com/science/article/pii/S014067369611196X (visited on 10/17/2014).

[16] Ron Brookmeyer et al. "Forecasting the global burden of Alzheimer's disease". In: *Alzheimer's & dementia: the journal of the Alzheimer's Association* 3.3 (July 2007), pp. 186–191. ISSN: 1552-5279. DOI: 10.1016/j.jalz.2007.04.381.

[17] Dan C. Ciresan et al. "Flexible, high performance convolutional neural networks for image classification". In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 2011, p. 1237. URL: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/download/3098/3425 (visited on 10/14/2014).

[18] Dan Claudiu Ciresan et al. "Deep, big, simple neural nets for handwritten digit recognition". In: *Neural computation* 22.12 (2010), pp. 3207–3220. URL: http://www.mitpressjournals.org/doi/full/10.1162/NECO_a_00052 (visited on 06/04/2014).

[19] Dan Cireşan, Ueli Meier, and Juergen Schmidhuber. "Multi-column Deep Neural Networks for Image Classification". In: *arXiv:1202.2745 [cs]* (Feb. 13, 2012). CVPR 2012, p. 3642-3649. URL: http://arxiv.org/abs/1202.2745 (visited on 05/12/2014).

[20] Ronan Collobert, Samy Bengio, and Johnny Marithoz. *Torch: A Modular Machine Learning Software Library*. 2002.

[21] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning". In: *BigLearn, NIPS Workshop*. 2011. URL: http://infoscience.epfl.ch/record/192376/files/Collobert_NIPSWORKSHOP_2011.pdf (visited on 03/01/2015).

[22] D W Dickson et al. "Identification of normal and pathological aging in prospectively studied nondemented elderly humans". In: *Neurobiology of aging* 13.1 (Feb. 1992), pp. 179–189. ISSN: 0197-4580.

[23] Dumitru Erhan et al. "Why does unsupervised pre-training help deep learning?" In: *The Journal of Machine Learning Research* 11 (2010), pp. 625–660. URL: http://dl.acm.org/citation.cfm?id=1756025 (visited on 10/14/2014).

[24] N. Ganesan et al. "Application of neural networks in diagnosing cancer disease using demographic data". In: *International Journal of Computer Applications (0975-8887)* (2010). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.7545&rep=rep1&type=pdf (visited on 06/10/2014).

[25] Ian J. Goodfellow et al. "Maxout Networks". In: *arXiv:1302.4389 [cs, stat]* (Feb. 18, 2013). arXiv: 1302.4389. URL: http://arxiv.org/abs/1302.4389 (visited on 10/17/2014).

[26] Ian J. Goodfellow et al. "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks". In: *arXiv:1312.6082 [cs]* (Dec. 20, 2013). URL: http://arxiv.org/abs/1312.6082 (visited on 05/06/2014).

[27] Ian J. Goodfellow et al. "Pylearn2: a machine learning research library". In: *arXiv:1308.4214 [cs, stat]* (Aug. 19, 2013). arXiv: 1308.4214. URL: http://arxiv.org/abs/1308.4214 (visited on 02/07/2015).

[28] Katherine Gray. "Machine learning for image-based classification of Alzheimer's disease". Available at http://www.doc.ic.ac.uk/~krg03/krgray_03102012_phdthesis.pdf. Doctoral dissertation. Imperial College London, Nov. 2012. URL: http://pubs.doc.ic.ac.uk/krgray-phdthesis-2012/.

[29] Ashish Gupta, Murat Ayhan, and Anthony Maida. "Natural Image Bases to Represent Neuroimaging Data". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 987–994. URL: http://machinelearning.wustl.edu/mlpapers/papers/icml2013_gupta13b (visited on 04/23/2014).

[30] Mark Hall et al. "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18. URL: http://dl.acm.org/citation.cfm?id=1656278 (visited on 02/09/2015).

[31] Brian Hayes. "Delving into Deep Learning". In: *American Scientist* 102.3 (2014), p. 186. ISSN: 0003-0996, 1545-2786. DOI: 10.1511/2014.108.186. URL: http://www.americanscientist.org/issues/id.16185,y.0,no.,content.true,page.99999,css.print/issue.aspx (visited on 10/15/2014).

[32] Geoffrey Hinton. "A Practical Guide to Training Restricted Boltzmann Machines". In: *Momentum* 9 (2010), p. 1. URL: http://www.learning.cs.toronto.edu/~hinton/absps/guideTR.pdf (visited on 06/05/2014).

[33] Geoffrey E. Hinton. "Dropout: A simple and effective way to improve neural networks". NIPS 2012. Lake Tahoe, Nevada, United States, Dec. 2012. URL: http://videolectures.net/nips2012_hinton_networks/ (visited on 05/19/2014).

[34] Geoffrey E. Hinton. "Learning multiple layers of representation". In: *Trends in cognitive sciences* 11.10 (2007), pp. 428–434. URL: http://www.sciencedirect.com/science/article/pii/S1364661307002173 (visited on 09/21/2014).

[35] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets". In: *Neural computation* 18.7 (2006), pp. 1527–1554. URL: http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527 (visited on 06/04/2014).

[36] Geoffrey E. Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv:1207.0580 [cs]* (July 3, 2012). URL: http://arxiv.org/abs/1207.0580 (visited on 05/15/2014).

[37] Andrew G. Howard. "Some Improvements on Deep Convolutional Neural Network Based Image Classification". In: *arXiv:1312.5402 [cs]* (Dec. 18, 2013). URL: http://arxiv.org/abs/1312.5402 (visited on 05/06/2014).

[38] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv:1408.5093 [cs]* (June 20, 2014). arXiv: 1408.5093. URL: http://arxiv.org/abs/1408.5093 (visited on 03/12/2015).

[39] Stefan Klöppel et al. "Automatic classification of MR scans in Alzheimer's disease". In: *Brain* 131.3 (2008), pp. 681–689. URL: http://brain.oxfordjournals.org/content/131/3/681.short (visited on 04/23/2014).

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[41] M P Laakso et al. "Hippocampal volumes in Alzheimer's disease, Parkinson's disease with and without dementia, and in vascular dementia: An MRI study". In: *Neurology* 46.3 (Mar. 1996), pp. 678–681. ISSN: 0028-3878.

[42] Chen-Yu Lee et al. "Deeply-Supervised Nets". In: *arXiv:1409.5185 [cs, stat]* (Sept. 18, 2014). arXiv: 1409.5185. URL: http://arxiv.org/abs/1409.5185 (visited on 10/07/2014).

[43] Honglak Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, 2009, pp. 609–616. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553453. URL: http://doi.acm.org/10.1145/1553374.1553453 (visited on 05/12/2014).

[44] Fayao Liu and Chunhua Shen. "Learning Deep Convolutional Features for MRI Based Alzheimer's Disease Classification". In: *arXiv:1404.3366 [cs]* (Apr. 13, 2014). URL: http://arxiv.org/abs/1404.3366 (visited on 04/23/2014).

[45] Benoît Magnin et al. "Support vector machine-based classification of Alzheimer's disease from whole-brain anatomical MRI". In: *Neuroradiology* 51.2 (2009), pp. 73–83. URL: http://link.springer.com/article/10.1007/s00234-008-0463-x (visited on 04/23/2014).

[46] John Markoff. "In a Big Network of Computers, Evidence of Machine Learning". In: *The New York Times* (June 25, 2012). ISSN: 0362-4331. URL: http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html (visited on 10/14/2014).

[47] James Martens. "Deep learning via Hessian-free optimization". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 735–742. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/icml2010_Martens10.pdf (visited on 06/04/2014).

[48] Guy M McKhann et al. "The diagnosis of dementia due to Alzheimer's disease: recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease". In: *Alzheimer's & dementia: the journal of the Alzheimer's Association* 7.3 (May 2011), pp. 263–269. ISSN: 1552-5279. DOI: 10.1016/j.jalz.2011.03.005.

[49] Tom M. Mitchell. *Machine Learning*. International Edition. Singapore: McGraw-Hill, 1997. 414 pp. ISBN: 0-07-115467-1.

[50] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv:1312.5602 [cs]* (Dec. 19, 2013). arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602 (visited on 10/17/2014).

[51] Jonathan H. Morra et al. "Comparison of AdaBoost and support vector machines for detecting Alzheimer's disease through automated hippocampal segmentation". In: *Medical Imaging, IEEE Transactions on* 29.1 (2010), pp. 30–43. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4957035 (visited on 04/23/2014).

[52] J C Morris et al. "Cerebral amyloid deposition and diffuse plaques in "normal" aging: Evidence for presymptomatic and very mild Alzheimer's disease". In: *Neurology* 46.3 (Mar. 1996), pp. 707–719. ISSN: 0028-3878.

[53] *National High Magnetic Field Laboratory: Introduction to Magnetic Resonance Imaging (Full Article)*. URL: http://www.magnet.fsu.edu/education/tutorials/magnetacademy/mri/fullarticle.html.

[54] Andrew Ng. *CS294A Lecture notes: Sparse autoencoder*. 2010. URL: http://www.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

[55] Adrien Payan and Giovanni Montana. "Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks". In: *arXiv:1502.02506 [cs, stat]* (Feb. 9, 2015). arXiv: 1502.02506. URL: http://arxiv.org/abs/1502.02506 (visited on 05/14/2015).

[56] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://dl.acm.org/citation.cfm?id=2078195 (visited on 02/10/2015).

[57] Thomas Serre et al. "A quantitative theory of immediate visual recognition". In: *Progress in brain research* 165 (2007), pp. 33–56. URL: http://www.sciencedirect.com/science/article/pii/S0079612306650048 (visited on 06/03/2014).

[58] Catherine Shu. *Google Acquires Artificial Intelligence Startup DeepMind For More Than $500M*. TechCrunch. URL: http://techcrunch.com/2014/01/26/google-deepmind/ (visited on 10/17/2014).

[59] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 1139–1147. URL: http://machinelearning.wustl.edu/mlpapers/papers/icml2013_sutskever13 (visited on 06/04/2014).

[60] P Tiraboschi et al. "The importance of neuritic plaques and tangles to the development and evolution of AD". In: *Neurology* 62.11 (June 8, 2004), pp. 1984–1989. ISSN: 1526-632X.

[61] Prashanthi Vemuri et al. "Alzheimer's disease diagnosis in individual subjects using structural MR images: validation studies". In: *NeuroImage* 39.3 (Feb. 1, 2008), pp. 1186–1197. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2007.09.073.

[62] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103. URL: http://dl.acm.org/citation.cfm?id=1390294 (visited on 06/05/2014).

[63] Li Wan et al. "Regularization of Neural Networks using DropConnect". In: Proceedings of the 30th International Conference on Machine Learning (ICML-13). Ed. by Sanjoy Dasgupta and David Mcallester. Vol. 28. JMLR Workshop and Conference Proceedings, 2013, pp. 1058–1066. URL: http://jmlr.org/proceedings/papers/v28/wan13.pdf (visited on 05/20/2014).

[64] Michael W Weiner et al. "The Alzheimer's Disease Neuroimaging Initiative: a review of papers published since its inception". In: *Alzheimer's & dementia: the journal of the Alzheimer's Association* 8.1 (Feb. 2012), S1–68. ISSN: 1552-5279. DOI: 10.1016/j.jalz.2011.09.172.

[65] *Well-posed problem*. In: *Wikipedia, the free encyclopedia*. Page Version ID: 611222467. July 22, 2014. URL: http://en.wikipedia.org/w/index.php?title=Well-posed_problem&oldid=611222467 (visited on 07/23/2014).

[66] Gary L. Wenk. "Neuropathologic changes in Alzheimer's disease". In: *The Journal of Clinical Psychiatry* 64 Suppl 9 (2003), pp. 7–10. ISSN: 0160-6689.

[67] Eric Westman et al. "AddNeuroMed and ADNI: Similar patterns of Alzheimer's atrophy and automated MRI classification accuracy in Europe and North America". In: *NeuroImage* 58.3 (Oct. 1, 2011), pp. 818–828. ISSN: 1053-8119. DOI: 10.1016/j.neuroimage.2011.06.065. URL: http://www.sciencedirect.com/science/article/pii/S1053811911007117 (visited on 05/14/2015).

[68] Anders Wimo and Martin Prince. *World Alzheimer Report 2010: The Global Economic Impact of Dementia*. 2. Available at http://www.alz.co.uk/research/files/WorldAlzheimerRepo Alzheimer's Disease International (ADI), Sept. 21, 2010. URL: http://www.alz.co.uk/research/world-report-2010.

[69] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining : Practical Machine Learning Tools and Techniques*. 3rd edition. Burlington, MA, USA: Morgan Kaufman Publishers, 2013. 629 pp. ISBN: 978-0-12-374856-0.

[70] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *arXiv:1311.2901 [cs]* (Nov. 12, 2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901 (visited on 10/15/2014).

[71] Zhi-Hua Zhou et al. *Lung Cancer Cell Identification Based on Artificial Neural Network Ensembles*. 2002.

Appendix A

Dataset Converter

Listing A.1: Dataset Converter

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Script that prepares ADNI-data for use with C5.0 and/or WEKA.

Takes a folder of xml files (metadata) and subfolders with .nii files (MRIs),
as downloaded from LONI archives via webbrowser (Java applet).
May perform dimensionality reduction (depending on arguments), and outputs
spec- and data-files for use with systems supplied as arguments.
"""
__author__ = "Eivind Arvesen"
__copyright__ = "Copyright (c) 2014-2015, Eivind Arvesen. All rights reserved."
__credits__ = ["Eivind Arvesen"]  # Bugfixers, suggestions etc.
__license__ = "GPL3"  # LGPL/GPL3/BSD/Apache/MIT/CC/C/whatever
__version__ = "0.0.3 Alpha"
__maintainer__ = "Eivind Arvesen"
__email__ = "eivind.arvesen@gmail.com"
__status__ = "Prototype"  # Prototype/Development/Production
__date__ = "2015/03/31 03:43:40 CEST"
__todo__ = [
    "In some serious need of generalizations/modularizations...",
    "Create logging method, so that the script doesn't fail"
    "silently or just quit...",
    "Check/try/error that number of XML and NIFTI are alike?",
    "Check that no return values (from methods) are empty...",
]
__bug__ = "None"

# Copyright (c) 2014-2015 Eivind Arvesen. All rights Reserved.

from itertools import izip_longest
from scipy import ndimage as nd
from sys import exit
import argparse
import collections
import copy_reg
import cPickle as pickle
import errno
import glob
import fnmatch
```

```
import multiprocessing as mp
import nibabel as nib
import numpy as np
import os
import re
import sys
import types
import xml.etree.cElementTree as ET


def _reduce_method(m):
    """Make␣instance␣methods␣pickleable."""
    if m.im_self is None:
        return getattr, (m.im_class, m.im_func.func_name)
    else:
        return getattr, (m.im_self, m.im_func.func_name)
copy_reg.pickle(types.MethodType, _reduce_method)


class AdniConverter(object):

    """
␣␣␣␣Convert␣ADNI␣dataset␣(MRI,␣NIfTI)␣to␣various␣formats.

␣␣␣␣DESCRIBE␣PARAMS,␣METHODS,␣ETC.
␣␣␣␣"""

    input_folder = None
    file_stem = None
    out_folder = {}
    outformat = None
    reduction = None
    reduction_dict = {"P": {"name": "PCA", "value": 20},
                      "H": {"name": "Histogram", "value": 32}}
    n_slices = None
    visits = None
    outfiles = []
    logging = True
    visits_dict = {
        0: "ADNI1␣Screening",
        1: "ADNI1/GO␣Month␣6",
        2: "ADNI1/GO␣Month␣12",
        3: "ADNI1/GO␣Month␣18",
        4: "ADNI1/GO␣Month␣24",
        5: "ADNI1/GO␣Month␣36",
        6: "No␣Visit␣Defined"
    }
    out_dict = {
        "C": {"filesuffix": ".data", "fileformat": "C5.0"},
        "D": {"filesuffix": ".log", "fileformat": "DEBUG"},
        "V": {"filesuffix": ".csv", "fileformat": "CSV"},
        "W": {"filesuffix": ".arff", "fileformat": "WEKA"}
    }
    new_resolution = (192, 192, 160)
    third_dim = None
    dx_groups = {}
    max_size = None
    merge = None
```

```
    merge_dict = [
        "Normal,MCI,LMCI,AD",
        "Normal,MCI,AD",
        "Normal,Other",
        "Other,MCI",
        "Other,AD"
    ]

    def __init__(self):
        """Initialize. Handle args, check that ouput-files does not exist."""
        np.seterr(divide='ignore', invalid='ignore')
        # fix PCA divide by zero "errors"...
        parser = argparse.ArgumentParser(description='A script for converting (a subset of) th
VISITS [-i] (ACCEPTED ARS, INCLUSIVE UPPER BOUND) | MERGED [-x] DX GROUPS
0 : ADNI1 Screening                              | 0: Normal / MCI / LMCI / AD
1 : ADNI1/GO Month 6                             | 1: Normal / MCI / AD
2 : ADNI1/GO Month 12                            | 2: Normal / Other
3 : ADNI1/GO Month 18                            | 3: Other / MCI
4 : ADNI1/GO Month 24                            | 4: Other / AD
5 : ADNI1/GO Month 36                            |
6 : No Visit Defined                             |
    ''')
        parser.add_argument(
            '-c', '--clean', help='remove any previous output',
            action='store_true', default=False)
        parser.add_argument(
            '-d', '--directory', required=True, help='directory to use',
            action='store')
        parser.add_argument(
            '-f', '--format', nargs='*', choices=['C', 'V', 'W'],
            default=['C', 'V'], help='Output format (C5.0, CSV, Weka)',
            action='store')
        parser.add_argument(
            '-g', '--getInfo',
            help='Show amount of images from visits and exit',
            action='store_true', default=False)
        parser.add_argument(
            '-i', '--visits', type=int,
            help='Latest visit to include (int <=6)', action='store')
        parser.add_argument(
            '-m', '--maxSize', type=float,
            help='Maximum output file size (in GiB)',
            action='store')
        parser.add_argument(
            '-n', '--slices', type=int, default=1, help='Use every Nth slice',
            action='store')
        parser.add_argument(
            '-r', '--reduction', choices=['P', 'H'], default=None,
            help='Method used for dimensionality reduction (PCA, Histogram)',
            action='store')
        parser.add_argument(
            '-s', '--scale', type=int, default=(192, 192, 160),
            help='Resolution to downscale to (x,y,z)', action='store', nargs=3)
        parser.add_argument(
            '-v', '--version', help='display version',
            action='version', version='%(prog)s ' + __version__)
        parser.add_argument(
            '-x', '--mergeGroups', type=int, nargs='*',
```

```
            help='Merge DX groups (int <=4)', action='store')
        self.args = parser.parse_args()

        self.input_folder = self.args.directory

        if self.args.getInfo:
            self.getInfo()
            sys.exit(0)

        if self.args.visits is not None:
            self.visits = self.args.visits
        else:
            if self.args.maxSize:
                self.visits = 6
            else:
                self.visits = 0

        if self.args.maxSize:
            self.max_size = self.args.maxSize

        self.new_resolution = tuple(self.args.scale)
        self.new_dimensions = self.args.scale

        if self.args.slices:
            self.n_slices = self.args.slices
            self.new_dimensions[0] = self.new_dimensions[0] / self.n_slices

        if self.args.reduction is not None:
            self.reduction = self.args.reduction
            self.new_dimensions[2] = self.reduction_dict[
                self.reduction]["value"]
            if self.reduction == "H":
                self.new_dimensions[1], self.new_dimensions[0] = 1, 1

        if self.args.mergeGroups is not None:
            self.merge = [x for x in self.args.mergeGroups]
        else:
            self.merge = [0]

        self.dx_groups = {x: {} for x in self.merge}

        self.file_stem = filter(lambda x: not re.match(
            r'^\s*$', x),
            [x.strip() for x in self.input_folder.split('/')])[-1]

        try:
            os.makedirs("Converted/")
        except OSError, e:
            if e.errno != errno.EEXIST:
                raise IOError('Error encountered during file creation.')

        for mergeGroup in self.merge:
            # Bruk os.path.join() - forrige gang ble sep'en en del av
            # fil/mappe-navnet...
            self.out_folder.update({mergeGroup:
                                    "Converted/" + self.file_stem +
                                    "_visits-" + str(self.visits) +
                                    "_nSlices-" + str(self.n_slices) +
```

```
                                    "_reduction-" +
                                    str(self.reduction) + "_scale-" +
                                    str(self.new_resolution[0]) + "x" +
                                    str(self.new_resolution[1]) + "x" +
                                    str(self.new_resolution[2]) +
                                    "_mergeGroups-" + str(mergeGroup) + "/"})

            try:
                os.makedirs(self.out_folder[mergeGroup])
            except OSError, e:
                if e.errno != errno.EEXIST:
                    raise IOError('Error encountered during file creation.')

        self.outfiles = list(self.out_dict[x.upper()]['filesuffix'] for x in self.args.format)
        if ("C" in self.args.format):
            self.outfiles.append(".names")
        if ("D" not in self.args.format):
            self.outfiles.append(".log")

        if self.args.clean:
            for mergeGroup in self.merge:
                for outfile in self.outfiles:
                    for fl in glob.glob(self.out_folder[mergeGroup] +
                                        self.file_stem + outfile):
                        try:
                            os.remove(fl)
                        except OSError, e:
                            raise IOError('Could not remove previous files.')
            print "Removed any and all previous output files."

        for mergeGroup in self.merge:
            for outfile in self.outfiles:
                if os.path.isfile(self.out_folder[mergeGroup] +
                                  self.file_stem + outfile):
                    print "The file " + self.out_folder[mergeGroup] + \
                        self.file_stem + outfile + " already existed."
                    print "Please (re)move it before attempting to run this \
script again."
                    print "Exiting..."
                    exit(1)
                else:
                    with open(self.out_folder[mergeGroup] + self.file_stem +
                              outfile, 'w') as fout:
                        fout.write('')

            try:
                os.makedirs(self.out_folder[mergeGroup] + "Results/")
            except OSError, e:
                if e.errno != errno.EEXIST:
                    raise

        self.outformat = [x.upper() for x in self.args.format]

        print ""

        print 'Initializing...'
        dict_forms = []
```

```
        for format in self.outformat:
            dict_forms.append(self.out_dict[format]['fileformat'])

        ofs = ' + '.join(str(p) for p in dict_forms)
        print ofs + ' format(s) selected.\n'

        print "Using every", self.n_slices, "slice"
        try:
            print "Dimensional reduction via", \
                self.reduction_dict[self.reduction]["name"]
        except Exception, e:
            print "No dimensional reduction."
        print "Scaling all images to", self.new_resolution
        print "Merging DX groups to scheme(s):"
        for x in self.merge:
            print str(x) + ":", self.merge_dict[x]
        print ""

        # parseXml(TRY return{...}, else fail>log)
        print 'Scanning input folder for XML-files from visits up until', \
            self.visits_dict[self.visits], '...'

        self.allFiles = self.filterFiles(
            self.getXmlFilesInFolder(self.input_folder), self.visits)

        gib_const = 10**9  # 1024**3 # 2**30
        if self.max_size:
            maxSize = self.max_size * gib_const
            print "Will stop writing when largest output-file exceeds limit \
of approximately", self.__greek__(maxSize) + "B."
        else:
            maxSize = float("inf")

        cores = mp.cpu_count()
        # Detect number of logical (not physical; i.e. HyperThreading) cores
        print "\nDetected", cores, "(virtual/logical) cores."

        p = mp.Pool()
        manager = mp.Manager()
        q = manager.Queue()

        for mergeGroup in self.merge:
            if ("V" in self.outformat):
                sizeFile = self.out_folder[mergeGroup] + self.file_stem + \
                    self.out_dict['V']['filesuffix']
            if ("C" in self.outformat):
                sizeFile = self.out_folder[mergeGroup] + self.file_stem + \
                    self.out_dict['C']['filesuffix']
            if ("W" in self.outformat):
                sizeFile = self.out_folder[mergeGroup] + self.file_stem + \
                    self.out_dict['W']['filesuffix']

        writer = mp.Process(
            target=self.__outputProcess__, args=(q, sizeFile, maxSize, p))
        writer.start()

        for xmlFile in self.allFiles:
            p.apply_async(
```

```
                self.__workerProcess__, args=(xmlFile, q, sizeFile, maxSize))
        p.close()
        p.join()  # Wait for all child processes to close.
        writer.join()

        print 'CONVERSION DONE!\n'

    def __outputProcess__(self, queue, sizeFile, maxSize, pool):
        """Listen for messages on queue. Perform all writing of output."""
        print'\nWriting spec-files...'

        if ("V" in self.outformat):
            self.writeCsvHeader()
            print 'Wrote ".csv" header.'
        if ("C" in self.outformat):
            self.writeNames()
            print 'Wrote ".names" file.'
        if ("W" in self.outformat):
            self.writeArffHeader()
            print 'Wrote ".arff" file HEADER.'

        print '\nStarting conversion of', self.allFiles.__len__(), \
            'NIfTI images.\n'

        images_used = 0
        buffer = {}
        while 1:
            m = pickle.loads(queue.get())
            if m == 'STOP':
                # SHOULDN'T THIS BE POOL INSTEAD?
                p.terminate()
                break
            current_image = images_used + 1
            if m['file_index'] != current_image:
                # save queue object in buffer if not current index
                buffer[m['file_index']] = m
            else:
                if os.path.getsize(sizeFile) < maxSize:
                    self.writeLine(m['data'], m['file_object'])
                    print 'Converted and wrote image', m['file_index'], 'of',\
                        self.allFiles.__len__(), "- Largest size-constricted \
output-file", self.__greek__(os.path.getsize(sizeFile)) + "B /",\
                        self.__greek__(maxSize) + "B."
                    images_used += 1
                    if os.path.getsize(sizeFile) >= maxSize:
                        queue.put('STOP')
                        break
                    else:
                        current_image += 1
                        while current_image in buffer:
                            m = buffer[current_image]
                            self.writeLine(m['data'], m['file_object'])
                            print 'Converted and wrote image', m['file_index'], 'of', self.all
                            del buffer[current_image]
                            images_used += 1
                            current_image += 1
                            if os.path.getsize(sizeFile) >= maxSize:
                                queue.put('STOP')
```

```
                            break
                    if ("DEBUG" in self.outformat):
                        for mergeGroup in self.merge:
                            self.log(
                                mergeGroup, self.prettyFormat(
                                                    self.parseXml(xmlFile)))
                else:
                    queue.put('STOP')
                    break
            if current_image > self.allFiles.__len__():
                queue.put('STOP')
                break
        self.logRun(self.allFiles.__len__(), images_used)
        print "Processed and wrote", images_used, "files and lines in total."

    def __workerProcess__(self, xmlFile, queue, sizeFile, maxSize):
        """Perform all data processing. Write results to queue."""
        if os.path.getsize(sizeFile) < maxSize:
            filei = self.allFiles.index(xmlFile) + 1
            fileo = self.parseXml(xmlFile)
            result = {'data': self.processData(
                fileo), 'file_index': filei, 'file_object': fileo}
            if os.path.getsize(sizeFile) < maxSize:
                # pickle numpy-array as binary data to increase performance
                queue.put(pickle.dumps(result, protocol=-1))
            else:
                queue.put("STOP")
        else:
            queue.put("STOP")

    def __greek__(self, size):
        """Return a string representing the greek/metric suffix of a size."""
        # http://www.gossamer-threads.com/lists/python/python/18406
        abbrevs = [
            (1 << 50L, 'P'),
            (1 << 40L, 'T'),
            (1 << 30L, 'G'),
            (1 << 20L, 'M'),
            (1 << 10L, 'k'),
            (1, '')
        ]
        for factor, suffix in abbrevs:
            if size > factor:
                break
        return ("%.2f" % float((size / factor))) + ' ' + suffix

    def getInfo(self):
        """Output visits from ADNI."""
        visit_groups = {}
        dx_groups = {}

        print "Scanning input folder for XML-files..."
        xml_files = self.getXmlFilesInFolder(self.input_folder)

        print "Counting subjects in visit and diagnostic groups...\n"
        for xml_file in xml_files:
            tree = ET.parse(xml_file)
            root = tree.getroot()
```

```
            visit = root.find(".//*visitIdentifier").text
            dx = root.find(".//*[@item='DX Group']").text

            if visit not in visit_groups:
                visit_groups.update({visit: 0})
            visit_groups.update({visit: (visit_groups[visit] + 1)})

            if dx not in dx_groups:
                dx_groups.update({dx: 0})
            dx_groups.update({dx: (dx_groups[dx] + 1)})

        i = 0

        visit_groups = collections.OrderedDict(
            sorted(visit_groups.items(), key=lambda x: (x[0][0:6], x[0][-2:])))
        dx_groups = collections.OrderedDict(
            reversed(sorted(dx_groups.items())))
        for e1, e2 in list(izip_longest(visit_groups.items(),
                                        dx_groups.items(), fillvalue=('', '')
                                        )):
            print i, ":", e1[0], "\t", e1[1], "\t\t\t  |  ", e2[0], "\t", e2[1]
            i += 1
        print ""

    def writeArffHeader(self):
        """Write header for Arff-file (WEKA)."""
        for mergeGroup in self.merge:
            with open(self.out_folder[mergeGroup] + self.file_stem + ".arff",
                      'w') as nfile:
                nfile.write('@RELATION ADNI\n\n')
                # nfile.write('@ATTRIBUTE ID string\n')
                # nfile.write('@ATTRIBUTE age numeric\n')
                # nfile.write('@ATTRIBUTE sex {M,F}\n')
                # nfile.write('@ATTRIBUTE "APOE A1" integer\n')
                # nfile.write('@ATTRIBUTE "APOE A2" integer\n')
                # nfile.write('@ATTRIBUTE "MMSE Total Score" numeric\n\n')

                # for number in range final resolution (192, 192, 160)
                # each slice reduced to X components and using every Nth slice:
                # THIS NEEDS TO TAKE VALUES FROM command line parameters etc.
                for number in range((self.new_dimensions[0]) *
                                    self.new_dimensions[1] *
                                    self.new_dimensions[2]):
                    nfile.write(
                        '@ATTRIBUTE "pixel ' + str(number + 1) + '" real\n')
                nfile.write(
                    '@ATTRIBUTE diagnosis {' + self.merge_dict[mergeGroup] +
                    '}\n')
                nfile.write('\n@DATA\n')

    def writeNames(self):
        """Write Names-file (C5.0)."""
        for mergeGroup in self.merge:
            with open(self.out_folder[mergeGroup] + self.file_stem + ".names",
                      'w') as nfile:
                nfile.write('diagnosis.              | target attribute\n\n')
                # nfile.write('ID:                     label.\n')
                # nfile.write('age:                    continuous.\n')
```

```
                # nfile.write('sex:                        M, F.\n')
                # nfile.write('APOE A1:                    discrete 4.\n')
                # nfile.write('APOE A2:                    discrete 4.\n')
                # nfile.write('MMSE Total Score:           continuous.\n\n')

                # for number in range final resolution (192, 192, 160),
                # each slice reduced to X components and using every Nth slice:
                # THIS NEEDS TO TAKE VALUES FROM command line parameters etc.
                nfile.write(
                    'diagnosis:               ' + self.merge_dict[mergeGroup] +
                    '.\n')
                for number in range((self.new_dimensions[0]) *
                                    self.new_dimensions[1] *
                                    self.new_dimensions[2]):
                    nfile.write(
                        'pixel_' + str(number + 1) + ':        continuous.\n')

    def writeCsvHeader(self):
        """Write header for CSV-file (?/Pylearn2)."""
        for mergeGroup in self.merge:
            with open(self.out_folder[mergeGroup] + self.file_stem + ".csv",
                      'w') as nfile:
                nfile.write('diagnosis,')
                for number in range((self.new_dimensions[0]) *
                                    self.new_dimensions[1] *
                                    self.new_dimensions[2]):
                    nfile.write('pixel_' + str(number + 1) + ',')
                nfile.seek(-1, os.SEEK_END)
                nfile.truncate()
                nfile.write('\n')

    def getXmlFilesInFolder(self, folder):
        """Get a list of all XML files in a folder (non-recursive search)."""
        xml_files = []
        for xml_file in os.listdir(folder):
            if xml_file.endswith(".xml"):
                xml_files.append(os.path.join(self.input_folder, xml_file))
        return xml_files

    def filterFiles(self, xmls, visits):
        """Filter out unwanted XML files, i.e. not within specified range."""
        print "Filtering through", len(xmls), "XMLs..."
        relevant_xmls = []

        if self.visits == 6:
            relevant_xmls = xmls

        else:
            for xf in xmls:
                tree = ET.parse(xf)
                root = tree.getroot()

                # if (root.find(".//*visitIdentifier").text not in visits):
                j = 0
                while j <= visits:
                    if(root.find(".//*visitIdentifier").text ==
                       self.visits_dict[j]):
                        relevant_xmls.append(xf)
```

```
                    j += 1

        print "Using", len(relevant_xmls), "XMLs."
        return relevant_xmls

    def parseXml(self, xml_file):
        """Get associated metadata and corresponding NIfTI file from XML."""
        tree = ET.parse(xml_file)
        root = tree.getroot()
        try:
            # 'ID': self.getId(root), 'Age': self.getAge(root),
            # 'Sex': self.getSex(root), 'APOE A1': self.getApoeA1(root),
            # 'APOE A2': self.getApoeA2(root),
            # 'MMSE Score': self.getMmseScore(root),
            return {'DX Group': self.getDxGroup(root),
                    'Nifti File': self.getNiftiFile(root)}
        except:
            e = sys.exc_info()[0]
            print "Error parsing:", e

    def getId(self, root):
        """Get subject ID from XML (root) element."""
        return root.find(".//*subjectIdentifier").text

    def getAge(self, root):
        """Get subject age from XML (root) element."""
        return root.find(".//*subjectAge").text

    def getSex(self, root):
        """Get subject sex from XML (root) element."""
        return root.find(".//*subjectSex").text

    def getApoeA1(self, root):
        """Get subject APOE A1 from XML (root) element."""
        return root.find(".//*[@item='APOE A1']").text

    def getApoeA2(self, root):
        """Get subject APOE A2 from XML (root) element."""
        return root.find(".//*[@item='APOE A2']").text

    def getMmseScore(self, root):
        """Get subject MMSE Total Score from XML (root) element."""
        return root.find(".//*[@attribute='MMSCORE']").text

    def getDxGroup(self, root):
        """Get subject diagnostic group from XML (root) element."""
        return root.find(".//*[@item='DX Group']").text

    def getNiftiFile(self, root):
        """Get corresponding NIfTI file from XML (root) element."""
        subjectIdentifier = root.find(".//*subjectIdentifier").text
        seriesIdentifier = root.find(".//*seriesIdentifier").text
        imageUID = root.find(".//*imageUID").text

        searchFor = 'ADNI_' + subjectIdentifier + '_*_' + 'S' + \
            seriesIdentifier + '_' + 'I' + imageUID + '.nii'

        matches = []
```

```
        for rootB, dirnames, filenames in os.walk(self.input_folder):
            for filename in fnmatch.filter(filenames, searchFor):
                matches.append(os.path.join(rootB, filename))
        if (matches.__len__() < 1):
            print 'There was no corresponding .nii match using the following \
pattern:'
            print searchFor
            print 'Exiting...'
            exit(1)
        elif (matches.__len__() > 1):
            print 'There was more than one corresponding .nii match using the \
following pattern:'
            print searchFor
            print 'Exiting...'
            exit(1)

        return matches[0]

    def mergeGroups(self, scheme, group):
        """Merge DX groups."""
        if scheme == 0 or scheme is None:
            pass

        elif scheme == 1:
            if group == 'LMCI':
                group = 'MCI'

        elif scheme == 2:
            if group != 'Normal':
                group = 'Other'

        elif scheme == 3:
            if group != 'MCI':
                if group == 'LMCI':
                    group = 'MCI'
                else:
                    group = 'Other'

        elif scheme == 4:
            if group != 'AD':
                group = 'Other'

        else:
            print "Failed to merge into another group."

        return group

    def prettyFormat(self, current_file):
        """Produce prettified ouput. For testing purposes."""
        # print "ID: ", current_file['ID']
        # print "Age: ", current_file['Age']
        # print "Sex: ", current_file['Sex']
        # print "APOE A1: ", current_file['APOE A1']
        # print "APOE A2", current_file['APOE A2']
        # print "MMSE Score: ", current_file['MMSE Score']
        print "DX Group: ", current_file['DX Group']
        print "Nifti File: ", current_file['Nifti File']
```

```
    def log(self, mergeGroup, message):
        """Write to log."""
        with open(self.out_folder[mergeGroup] + self.file_stem +
                  ".log", 'a') as logf:
            logf.write(str(message))

    def resize(self, img, new_resolution):
        """Resize (data from) 3D image matrix (numpy array)."""
        dsfactor = [w / float(f) for w, f in zip(new_resolution, img.shape)]
        new_img = nd.interpolation.zoom(img, zoom=dsfactor)
        return new_img

    def labelToInt(self, label, scheme):
        """Replace pre-merged label (string) with Int."""
        if scheme == 0 or scheme is None:
            if label == 'Normal':
                return '0'
            elif label == 'MCI':
                return '1'
            elif label == 'LMCI':
                return '2'
            elif label == 'AD':
                return '3'
            else:
                print "Failed to merge into another group."
                sys.exit(1)

        elif scheme == 1:
            if label == 'Normal':
                return '0'
            elif label == 'MCI':
                return '1'
            elif label == 'AD':
                return '2'
            else:
                print "Failed to merge into another group."
                sys.exit(1)

        elif scheme == 2:
            if label == 'Normal':
                return '0'
            elif label == 'Other':
                return '1'
            else:
                print "Failed to merge into another group."
                sys.exit(1)

        elif scheme == 3:
            if label == 'Other':
                return '0'
            elif label == 'MCI':
                return '1'
            else:
                print "Failed to merge into another group."
                sys.exit(1)

        elif scheme == 4:
            if label == 'Other':
```

```
                return '0'
            elif label == 'AD':
                return '1'
            else:
                print "Failed to merge into another group."
                sys.exit(1)

        else:
            print "Failed to merge into another group."
            sys.exit(1)

    def maybeReduceDimensionality(self, img_data):
        """Dimensional reduction of 3D image matrix (numpy array)."""
        # Iterating through a ndimensional array produces slices along
        # the last axis. This is equivalent to data[i,:,:] in this case
        img_data = img_data[::self.n_slices]

        if self.reduction is None:
            """No Reduction"""
            return img_data

        elif self.reduction == "H":
            """Histogram"""
            from sklearn import preprocessing

            img_data = np.asarray(img_data, dtype=float).flat

            min_max_scaler = preprocessing.MinMaxScaler()
            scaled_data = min_max_scaler.fit_transform(img_data)

            hist = np.histogram(scaled_data,
                                bins=self.reduction_dict["H"]["value"],
                                range=None, normed=False, weights=None,
                                density=None)[0]

            return hist.reshape(1, hist.shape[0])

        elif self.reduction == "P":
            """Slice-wise (randomized) Principal Component Analysis"""
            from sklearn.preprocessing import normalize
            from sklearn.decomposition import RandomizedPCA

            proj_data = []
            for img_slice in img_data:
                norm_data = normalize(img_slice)

                shaped_data = np.reshape(norm_data, norm_data.size)
                # shaped_data.shape

                rpca = RandomizedPCA(
                    n_components=self.reduction_dict["P"]["value"],
                    random_state=0)
                proj_slice = rpca.fit_transform(norm_data)
                # plt.imshow(proj_data)

                # feat_data = rpca.inverse_transform(proj_data)
                # plt.imshow(feat_data)
                # plt.imshow(norm_data)
```

```
            proj_data.append(proj_slice)

        return proj_data

    def processData(self, current_file):
        """Process data."""
        return self.maybeReduceDimensionality(self.resize(nib.load(
            current_file['Nifti File']).get_data(), self.new_resolution))

    def writeLine(self, img_data, current_file):
        """Write image data as line in dataset file(s)."""
        for mergeGroup in self.merge:
            for format in self.outformat:
                output_file = self.out_folder[mergeGroup] + self.file_stem + \
                    self.out_dict[format]['filesuffix']
                output_format = self.out_dict[format]['fileformat']

                with open(output_file, "a") as myfile:
                    # myfile.write(current_file['ID'] + ',' + \
                    # current_file['Age'] + ',' + current_file['Sex'] + \
                    # ',' + current_file['APOE A1'] + ',' + \
                    # current_file['APOE A2'] + ',')

                    if (format != 'W'):
                        if (format == 'V'):
                            myfile.write(
                                str(self.labelToInt(self.mergeGroups(
                                    mergeGroup, current_file['DX Group']), mergeGroup)) +
                                ',')
                        else:
                            myfile.write(
                                self.mergeGroups(mergeGroup,
                                                 current_file['DX Group']) +
                                ',')

                    i = 0
                    for data_slice in img_data:
                        np.savetxt(myfile, (data_slice * (10**6)).astype(int),
                                   delimiter=",", newline=',', fmt="%d")  # s/f
                        i += 1

                    # hack to remove single (illegal[?]) comma on end of line
                    # (MAY NOT WORK ON ALL PLATFORMS [i.e. Windows])
                    myfile.seek(-1, os.SEEK_END)
                    myfile.truncate()
                    if (format == 'W'):
                        myfile.write(
                            ',' + self.mergeGroups(mergeGroup,
                                                   current_file['DX Group']))
                    myfile.write('\n')

            group = self.mergeGroups(mergeGroup, current_file['DX Group'])
            if group not in self.dx_groups[mergeGroup]:
                self.dx_groups[mergeGroup].update({group: 0})
            self.dx_groups[mergeGroup].update(
                {group: (self.dx_groups[mergeGroup][group] + 1)})
```

```
    def logRun(self, total_files, images_used):
        """Output relevant information from current conversion."""
        for mergeGroup in self.merge:
            self.log(mergeGroup, "CONVERSION INFORMATION:\n\n")
            dict_forms = []
            for format in self.outformat:
                dict_forms.append(self.out_dict[format]['fileformat'])
            ofs = ' + '.join(str(p) for p in dict_forms)
            self.log(
                mergeGroup, "Started out using " + str(total_files) +
                " .nii files.\n")
            if self.max_size is not None:
                self.log(mergeGroup,
                         "Stopped conversion when largest ouput file reached "
                         + str(self.max_size) + "GiB.\n")
            self.log(
                mergeGroup, "Wrote " + str(images_used) + " lines in total.\n")
            self.log(mergeGroup, "Resized all NIfTI MR Images to " +
                     str(self.new_resolution) +
                     " (lowest resolution in set).\n\n")
            self.log(mergeGroup, "Included visits:\n")
            for x in xrange(0, self.visits + 1):
                self.log(mergeGroup, self.visits_dict[x] + "\n")
            self.log(
                mergeGroup, "\nUsed every " + str(self.n_slices) +
                " slice(s).\n")
            if self.reduction is not None:
                self.log(mergeGroup, "Reduced dimensionality of each slice to "
                         + str(self.reduction_dict[self.reduction]["value"]) +
                         " components/bins via method " +
                         str(self.reduction_dict[self.reduction]["name"]) +
                         ".\n")
            self.log(mergeGroup, "Converted to " + ofs + " format(s).\n")
            self.log(mergeGroup, "Final resolution was (" +
                     str(self.new_dimensions[0]) +
                     ", " + str(self.new_dimensions[1]) +
                     ", " + str(self.new_dimensions[2]) +
                     ").\n")
            self.log(mergeGroup, "DX Groups after (eventual) merging: " +
                     self.merge_dict[mergeGroup] + ".\n")
            self.log(mergeGroup, "\nSubjects in diagnostic groups:\n")
            groups = collections.OrderedDict(
                sorted(self.dx_groups[mergeGroup].items(), key=lambda t: t[0]))
            for k, v in groups.iteritems():
                self.log(mergeGroup, k + "\t" + str(v) + "\n")


if __name__ == '__main__':
    """Run only if the script is called explicitly."""

    obj = AdniConverter()
```

Appendix B

Pylearn2 Dataset Class

Listing B.1: Pylearn2 Dataset Class

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
Dataset-class for using custom preprocessed ADNI MRI dataset with Pylearn2.

A simple general csv dataset wrapper for pylearn2.
Can do automatic one-hot encoding based on labels present in a file.
Based upon work from Zygmunt Zajac's (zygmunt@fastml.com) CSVDataset class in the Pylearn2 rep

FIX based on answer from
http://stackoverflow.com/questions/27609843/pylearn2-csvdataset-typeerror
"""
__authors__ = "Eivind Arvesen"
__license__ = "3-clause BSD"
__email__ = "eivind.arvesen@gmail.com"

import numpy as np

from pylearn2.datasets import dense_design_matrix
from pylearn2.format.target_format import convert_to_one_hot
from pylearn2.utils.string_utils import preprocess
from pylearn2.config import yaml_parse


# Dynamically change class based on dataset size?
# http://stackoverflow.com/questions/9539052/python-dynamically-changing-base-classes-at-runti


class CSVDatasetPlus(dense_design_matrix.DenseDesignMatrixPyTables):

    """A generic class for accessing CSV files.

    Labels, if present, should be in the first column
    if there's no labels, set expect_labels to False
    if there's no header line in your file, set expect_headers to False

    Parameters
    ----------
    path : str
        The path to the CSV file.
```

```
    task : str
        The type of task in which the dataset will be used -- either
        "classification" or "regression". The task determines the shape of the
        target variable. For classification, it is a vector; for regression, a
        matrix.

    expect_labels : bool
        Whether the CSV file contains a target variable in the first column.

    expect_headers : bool
        Whether the CSV file contains column headers.

    delimiter : str
        The CSV file's delimiter.

    start : int
        The first row of the CSV file to load.

    stop : int
        The last row of the CSV file to load.

    start_fraction : float
        The fraction of rows, starting at the beginning of the file, to load.

    end_fraction : float
        The fraction of rows, starting at the end of the file, to load.


    *** NEW, EIVIND, ADNI ***


    yaml_src : string
        Lorem ipsum

    one_hot : bool
        Lorem ipsum

    num_classes : int
        Lorem ipsum

    which_set : string
        Lorem ipsum
    """

    def __init__(self,
                 path='train.csv',
                 task='classification',
                 expect_labels=True,
                 expect_headers=True,
                 delimiter=',',
                 start=None,
                 stop=None,
                 start_fraction=None,
                 end_fraction=None,
                 yaml_src=None,
                 one_hot=True,
                 num_classes=4,
```

```
                    which_set=None):
        """
        ..todo::..

        WRITEME
        """
        self.path = path
        self.task = task
        self.expect_labels = expect_labels
        self.expect_headers = expect_headers
        self.delimiter = delimiter
        if which_set is not None:
            self.start = start
            self.stop = stop
        self.start_fraction = start_fraction
        self.end_fraction = end_fraction

        self.view_converter = None

        if yaml_src is not None:
            self.yaml_src = yaml_parse.load_path(yaml_src)
        # self.yaml_src=yaml_parse.load_path("mlp.yaml")
        # eventually; triple-quoted yaml...
        self.one_hot = one_hot
        self.num_classes = num_classes

        if which_set is not None and which_set not in[
                                                    'train', 'test', 'valid']:
            raise ValueError(
                'Unrecognized which_set value "%s".' % (which_set,) +
                '". Valid values are ["train","test","valid"].')
        else:
            self.which_set = which_set
            if self.start is not None or self.stop is not None:
                raise ValueError("Use start/stop or which_set,"
                    " just not together.")

        if task not in ['classification', 'regression']:
            raise ValueError('task must be either "classification" or '
                             '"regression"; got ' + str(task))

        if start_fraction is not None:
            if end_fraction is not None:
                raise ValueError("Use start_fraction or end_fraction, "
                                 "not both.")
            if start_fraction <= 0:
                raise ValueError("start_fraction should be > 0")

            if start_fraction >= 1:
                raise ValueError("start_fraction should be < 1")

        if end_fraction is not None:
            if end_fraction <= 0:
                raise ValueError("end_fraction should be > 0")

            if end_fraction >= 1:
                raise ValueError("end_fraction should be < 1")
```

```
        if start is not None:
            if start_fraction is not None or end_fraction is not None:
                raise ValueError("Use␣start,␣start_fraction,␣or␣end_fraction,"
                                 "␣just␣not␣together.")

        if stop is not None:
            if start_fraction is not None or end_fraction is not None:
                raise ValueError("Use␣stop,␣start_fraction,␣or␣end_fraction,"
                                 "␣just␣not␣together.")

        # and go
        self.path = preprocess(self.path)
        X, y = self._load_data()

        # y=y.astype(int)
        # y=map(int, np.rint(y).astype(int))

        if self.task == 'regression':
            super(CSVDatasetPlus, self).__init__(X=X, y=y)
        else:
            # , y_labels=4 # y_labels=np.max(y)+1
            super(CSVDatasetPlus, self).__init__(
                X=X, y=y.astype(int), y_labels=self.num_classes)

    # new interface (get_...)?
    # see pylearn2/datasets/cifar10.py
    # see pylearn2/datasets/svhn.py

    def _load_data(self):
        """
␣␣␣␣␣␣␣␣..␣todo::␣..

␣␣␣␣␣␣␣␣␣␣␣␣WRITEME
␣␣␣␣␣␣␣␣"""
        assert self.path.endswith('.csv')

        if self.expect_headers:
            data = np.loadtxt(self.path,
                              delimiter=self.delimiter,
                              skiprows=1, dtype=int)
        else:
            data = np.loadtxt(self.path, delimiter=self.delimiter, dtype=int)

        def take_subset(X, y):
            self.num_classes = np.unique(y).shape[0]
            if self.which_set is not None:
                total = X.shape[0]
                train = int((total/100.0)*70)
                test = int((total/100.0)*15)
                valid = int((total/100.0)*15)
                train += total - train - test - valid
                if self.which_set == 'train':
                    self.start = 0
                    self.stop = train -1
                elif self.which_set == 'test':
                    self.start = train
                    self.stop = train + test -1
                elif self.which_set == 'valid':
```

```
                self.start = train + test
                self.stop = train + test + valid -1

        if self.start_fraction is not None:
            n = X.shape[0]
            subset_end = int(self.start_fraction * n)
            X = X[0:subset_end, :]
            y = y[0:subset_end]
        elif self.end_fraction is not None:
            n = X.shape[0]
            subset_start = int((1 - self.end_fraction) * n)
            X = X[subset_start:, ]
            y = y[subset_start:]
        elif self.start is not None:
            X = X[self.start:self.stop, ]
            if y is not None:
                y = y[self.start:self.stop]

        return X, y

    if self.expect_labels:
        y = data[:, 0]
        X = data[:, 1:]
        y = y.reshape((y.shape[0], 1))

    else:
        X = data
        y = None

    X, y = take_subset(X, y)

    if self.one_hot:
        y = convert_to_one_hot(
            y, dtype=int, max_labels=self.num_classes, mode="concatenate")

    return X, y  # .astype(float)
```