Deep Learning based Classification of FDG-PET Data for Alzheimer's Disease

by

Shibani Singh

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Yalin Wang, Chair
Baoxin Li
Jianming Liang

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Alzheimers Disease (AD), a neurodegenerative disease is a progressive disease that affects the brain gradually with time and worsens. Reliable and early diagnosis of AD and its prodromal stages (i.e. Mild Cognitive Impairment(MCI)) is essential. Fluorodeoxyglucose (FDG) positron emission tomography (PET) measures the decline in the regional cerebral metabolic rate for glucose, offering a reliable metabolic biomarker even on presymptomatic AD patients. PET scans provide functional information that is unique and unavailable using other types of imaging. The computational efficacy of FDG-PET data alone, for the classification of various Alzheimers Diagnostic categories (AD, MCI (LMCI, EMCI), Control) has not been studied. This serves as motivation to correctly classify the various diagnostic categories using FDG-PET data. Deep learning has recently been applied to the analysis of structural and functional brain imaging data. This thesis is an introduction to a deep learning based classification technique using neural networks with dimensionality reduction techniques to classify the different stages of AD based on FDG-PET image analysis. This thesis develops a classification method to investigate the performance of FDG-PET as an effective biomarker for Alzheimer's clinical group classification. This involves dimensionality reduction using Probabilistic Principal Component Analysis on max-pooled data and mean-pooled data, followed by a Multilayer Feed Forward Neural Network which performs binary classification. Max pooled features result into better classification performance compared to results on mean pooled features. Additionally, experiments are done to investigate if the addition of important demographic features such as Functional Activities Questionnaire(FAQ), gene information helps improve performance. Classification results indicate that our designed classifiers achieve competitive results, and better with the additional of demographic features.

To my mother and father, for always being there for me,

for giving me everything I wanted and more, for making sure I always smiled.

To my part family in Sweden for all the love and support.

To my grandparents, for the countless blessings.

To God,

for looking out for me, making sure I always had hope in life,

and making me strong with time.

ACKNOWLEDGMENTS

First and foremost I express my sincere gratitude to my advisor Dr. Yalin Wang without whom this thesis would not have been possible. He stood by with immense knowledge, support and motivation. His guidance throughout the year helped me move forward and develop on my thesis work.

I would also like to thank my thesis committee members, Dr. Jianming Liang and Dr. Baoxin Li, for their support and encouragement.

My sincere thanks to my fellow labmates: Jie, Liang, Wani, Michael, Duyan and Wen, who have always been ready to help me out, and seeing them work in the lab has been a source of inspiration for me. It has been a great experience to be a part of the Geometry Systems Laboratory at ASU.

I would like to thank my companion Anant, for persistently standing by and supporting me throughout these years, and while I worked on my dissertation.

My deepest gratitude to my parents for never letting distance be a factor in the amount of love and care they provided. They taught me the importance of sharing knowledge and they made me a better person.

A huge thank you to all my friends who stood by me through tough times, and stayed in touch when far, with words of encouragement.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

[1] http://adni.loni.usc.edu/study-design/background-rationale

Chapter 1

INTRODUCTION

Alzheimer's Disease (AD), is a condition named after the German physician Alois Alzheimer. Characteristic neuropathological structures are present in the brain and lead to progressive dementia. There is general memory loss, followed by further functional and cognitive decline. Patients are then unable to even perform basic tasks de Leon (1999). This progressive neurodegenerative disease results in the loss of brain function, for which no cures or prevention methods have been discovered yet. However, treatments based on symptoms help patients to maintain mental functionality and manage behavioral symptoms. Clinical trials aim to lower the risk of development of the disease or help delay the onset and progression of AD H.-W. Klafki and Wiltfang (2006). Alzheimer's is a progressive disease that progresses in stages. The onset is detected in the Mild Cognitive Impairment stage (Early or Late). We study the classification of subjects from various stages of Alzheimers 2.1.1 to correctly predict the disease stages for the subjects. The FDG-PET images for all subjects is normalized, after which dimensionality reduction techniques are used to remove noise from the data. A simple neural network architecture is then used to classify these images.

- Chapter 2 discusses the background of clinical Alzheimer's categories, and the works on the application of deep learning to Alzheimer's disease analysis.

- Chapter 3 focuses on the dataset and the methods used in the whole classification process.

- Chapter 4 discusses the various experiments performed to configure the neural

network.

- Chapter 5 presents our results and analysis.
- In Chapter 6 we discuss our research further and state the limitations of our framework.

Our work has three main contributions, a coherent and efficient deep learning framework that well explores the possibility of FDG-PET for AD diagnosis. Secondly, we evaluated our work in a relatively large dataset and achieved competitive results. Thirdly, we exhibit the effective increase in classification performance with the addition of demographic variables (Age, Gender, APOE 1, APOE 2, & FAQ score) to our max-pooled (intensity) data. We compare our results to those based on FDG-PET analysis. We hope our work will inspire more deep learning based work on FDG-PET analysis and advance preclinical AD research.

Chapter 2

BACKGROUND

2.1 Problem Background

2.1.1 AD Diagnostic Categories

The progression of AD in patients can be categorized into the following stages:

- Cognitively Normal (CN): The control subjects in the ADNI study that show no signs of cognitive impairment, depression or dementia.
- Significant Memory Concern (SMC): SMC patients indicate they have a concern and display slight forgetfulness, correlated with a higher likelihood of progression. This does not indicate consistent forgetfulness.
- Mild Cognitive Impairment (MCI): The level of Mild Cognitive Impairment can be further classified into two and is determined using the Wechsler Memory Scale Logical Memory II (WMS II) Hartwig (1990). WMS II assists in differentiating individuals with various cortical impairments. Memory is impaired in MCI, but general cognitive function is preserved.
    1. Early Mild Cognitive Impairment (EMCI)
    2. Late Mild Cognitive Impairment (LMCI)
- Alzheimer's Disease (AD): The subjects meet the NINCDS/ADRDA G. McKhann and Stadlan (1984) criteria for probable AD.

2.1.2 *Neuroimaging and Biomarkers*

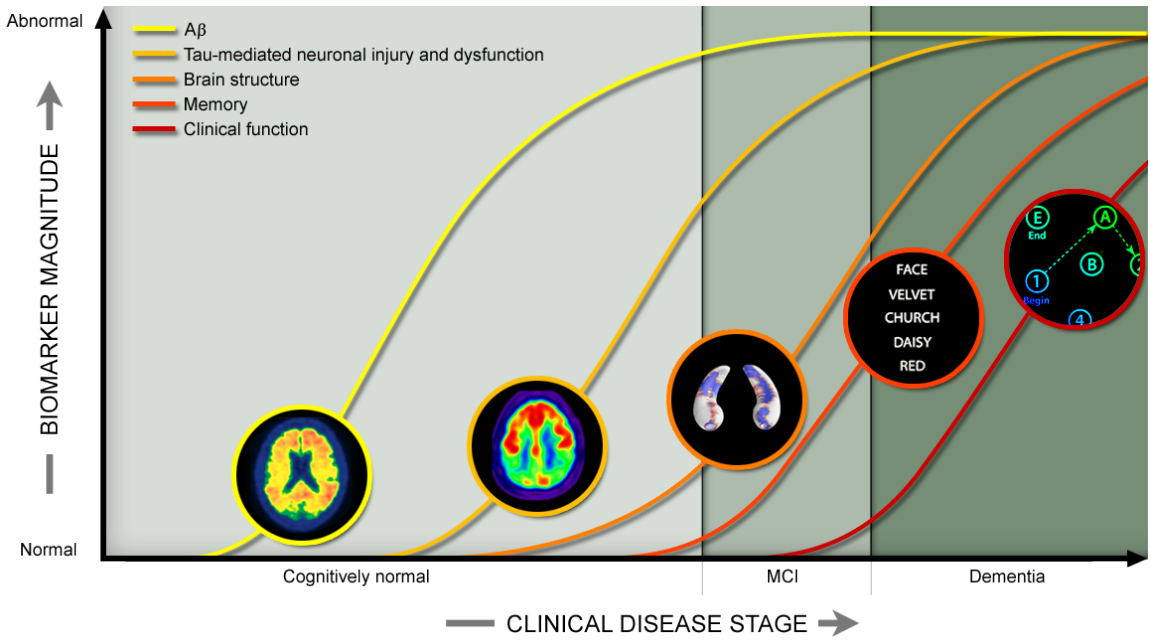

Figure 2.1: ADNI Disease Stages with Varied Biomarker Indications [1]

Alzheimer's Disease Neuroimaging Initiative 2 (ADNI2) extends the work of two previous studies, ADNI1 and ADNI-GO Weiner (2012) developed to study the progression of the disease using biomarkers. The five following biomarkers are studied in ADNI2:

1 Amyloid Beta

2 Neurodegeneration

3 Brain Atrophy

4 Memory Loss

5 General Cognitive Decline

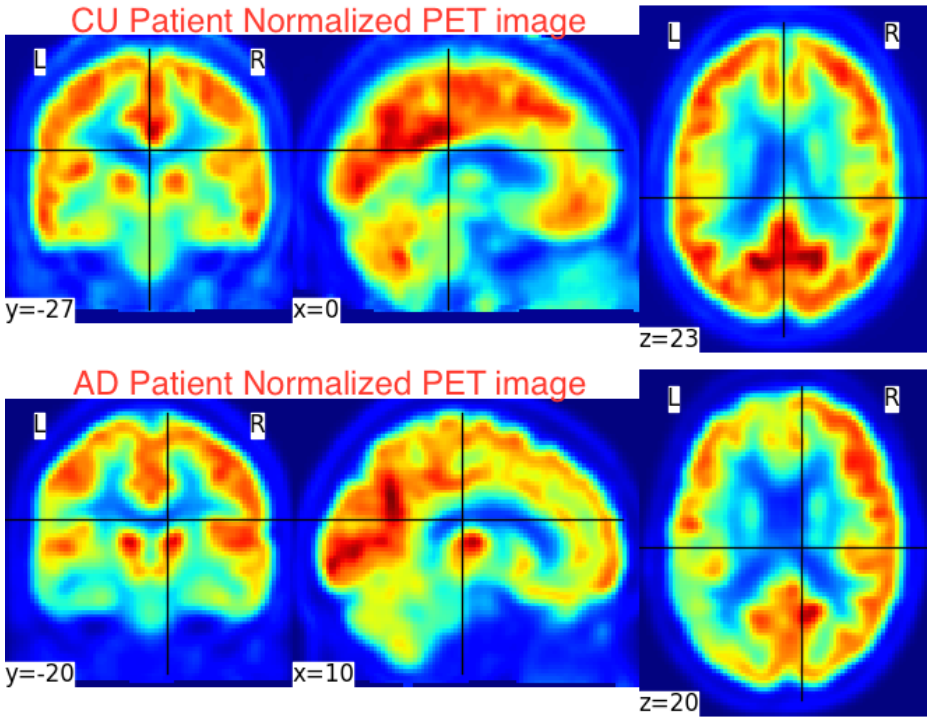

Figure 2.2: Normalized PET Image Slices for CU and AD Subjects

Out of these, **Ameloid beta** and **Neurodegeneration** are indicated by FDG-PET, and are significant predictors of cognitive decline.

We see in Figure 2.1, Amyloid Beta and CSF Tau species are detected by FDG-PET, and these values vary greatly between cognitively normal and dementia subjects. Since these measures vary for FDG-PET scans, these images may have great potential in clinical group classification.

In the study of Alzheimers disease (AD), neuroimaging based measures have shown high sensitivity in tracking changes over time and thus were proposed as possible biomarkers to evaluate AD burden and progression and response to interventions. In addition to the pathological amyloid and tau imaging measurements for AD, fluorodeoxyglucose (FDG) positron emission tomography (PET) characterizes the cerebral glucose hypometabolism related to AD and AD risk, offering a reliable metabolic biomarker even at its presymptomatic stage. Fig. 2.2 visualizes the neural activity in

normalized PET scans of AD and normal subjects. We see that the central image in both cases displays clear loss of functionality for AD patients as compared to normal.

While the difference between AD and Normal is clearly displayed in these images, there has not been much work in the classification of categories within MCI (Late MCI and Early MCI). The dataset (FDG-PET from ADNI2) we experiment on is a huge dataset compared to data that has been available to experiment with till date. The number of AD cases would reduce by an estimated 10% in 2050, if both the onset and progression are delayed by a year. Early identification of presymptomatic patients is important to allow the recruitment of participants for clinical trials. **The aim of our study is to validate FDG-PET images as predictors and generators of outcomes for use in clinical trials of AD treatment.**

2.2 History of Deep Learning for Alzheimer's

Deep learning (also known as deep structured learning, hierarchical learning or deep machine learning) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations. In short, deep learning refers to artificial neural networks that are composed of many layers.

There has been growing interest to study FDG-PET for AD and AD risk and particularly to identify and predict mild cognitive impairment (MCI). Although numerous analysis tools have been developed, much of the prior work (e.g.Reiman *et al.* (2001)), has relied on voxel-wise analysis corrected by multiple comparisons to discover group-wise differences and the general trend in data. However, there are a number of issues in extending the group analysis framework to compute AD risk on individual basis. For example, prior work has showed that the statistically significant

pixels obtained in group difference studies do not necessarily carry strong statistical power for predictions Sun *et al.* (2009). To develop an effective precision medicine, one needs some system which may be able to measure subtle difference and make robust prediction/classification on an individual basis. Thus far, it is still challenging to build FDG-PET imaging diagnosis and prognosis systems because of the tremendous difficulty to optimally integrate global functional image information.

Recently deep learning has helped achieve state-of-the-art classification results in myriad classification problems in the areas of signal, speech, text, image processing and medical imaging Hazlett *et al.* (2017). Deep learning based feature representation using auto-encoders was recently used to achieve high accuracies using MRI and PET data Suk and Shen (2013). Deep learning has also been used for classification using MRI and PET data Li *et al.* (2015).Classification has been improvised using a combination of multiple imaging modalities to improvise on neuroimaging biomarkers, requiring less labeled data Liu *et al.* (2015). The advance in deep learning research inspires us to develop novel deep learning methods to advance the FDG-PET analysis research which may facilitate their use in preclinical and clinical AD treatment development.

Deep Learning has recently been used to perform feature learning, feature representation, and classification using combinations of modalities Liu *et al.* (2015). High level feature extraction with multi-layer stacked auto-encoders Pascal Vincent (2008) have shown state of the art improvement in data representation of PET images Siqi Liu and Feng (2014).

Deep Learning has served two purposes in medical imaging analysis, mainly dimensionality reduction (using feature extraction Suk and Shen (2013) and feature selection Heung-II Suk and Shen (2015)) and classification Li *et al.* (2015). Classification relying only on FDG-PET using machine learning techniques has previously

resulted in accuracies greater than 85% I.A. Illan (2011). Information extracted from serial FDG-PET through regional analysis has been used to achieve similar accuracies on classification Katherine R. Gray and Rueckert (2012).

Chapter 3

METHODS

This section presents our workflow pipeline implemented to evaluate the performance of binary classification of diagnostic categories. Our experiments show that the proposed system is promising for AD diagnosis research.

3.1 Data

Data used in the preparation of this work 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 coinvestigators 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.

We work on FDG-PET data from the ADNI-2 dataset. This dataset contains FDG-PET data that has been manually labeled into diagnostic categories by an expert. The baseline data of patients includes 186 healthy control (CU), 336 Mild Cognitive Impairment (MCI) with 158 Late MCI and 178 Early MCI, and 146 AD. The size of each FDG-PET image is $79 \times 95 \times 79$. Table 3.1 shows the age distribution for our subjects. We normalize the data to linearly align all the images into a common space using the software toolkit Statistical Parametric Mapping Penny *et al.* (2011). The normalized FDG-PET images are of size $79 \times 95 \times 79$. Each value is a voxel intensity value. We use the intensity values for the whole brain in our experiments. Each voxel is a feature and hence the feature dimensionality is $592895(f_{dim})$ per image data sample. Since the number of data samples is much less than the number of features($n \ll f_{dim}$), we use dimensionality reduction techniques to reduce f_{dim}. This is discussed in the next section. We use a multilayer perceptron classifier to perform binary classification. We measured F1-measure, precision, recall, negative and positive predictive values with 10-fold cross validation.

A gene called APOE can influence the risk for the more common late-onset type of Alzheimer's. There are three types of the APOE gene, called alleles: APOE2, E3 and E4. The E2 allele is the rarest form of APOE and carrying even one copy appears to reduce the risk of developing Alzheimer's by up to 40%. APOE3 is the

Category	Age ± SD	Age Range	Males	Females
AD	74.74 ± 8.16	56 ∼ 90	85	61
MCI	71.88 ± 7.34	55 ∼ 91	186	150
LMCI	72.50 ± 7.51	55 ∼ 91	84	74
EMCI	71.34 ± 7.20	55 ∼ 88	102	76
CU	73.56 ± 6.25	56 ∼ 89	89	97

Table 3.1: Age Distribution for Subjects

most common allele and doesn't seem to influence risk. The APOE4 allele, present in approximately 20% of people, increases the risk for Alzheimer's and lowers the age of onset. The National Institutes of Health recommends genetic testing for APOE status to advance drug research in clinical trials. APOE4 is just one of many risk factors for dementia and its influence can vary across age, gender, race, and nationality (Farrer *et al.* (1997),Liu *et al.* (2013)).

We use this information to further enhance our classification performance. We also use the age, gender and FAQ (Functional Activities Questionnaire) scores for each subject. FAQ (for Alzheimer's Disease Initiative *et al.* (2003)) is an informant-based measure of functional abilities. Informants provide performance ratings of the target person on ten complex higher-order activities. We have the FAQ scores, APOE1 and APOE2 values for each of our subjects.

When the training examples are linearly separable, we can set the parameters of a linear classifier so that all the training examples are classified correctly. We use linear SVM, which is based on the implementation LIBLINEAR Fan *et al.* (2008) to test our max-pooled data for linear separability. We then perform classification on max-pooled data, and max-pooled data combined with age, gender, apoe 1, apoe 2,

and FAQ values

3.2 Classification Pipeline

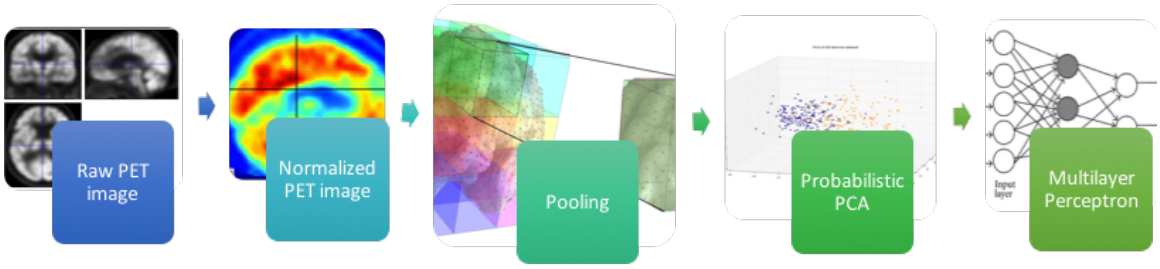

Figure 3.1: Classification Pipeline.

Our classification pipeline as shown in Fig. 3.1 is as follows.

1. Normalized PET data is the initial input data.
2. Max-pooling/Mean-pooling for two classification pipelines performed on each subject's data to reduce the number of features.
3. Dimensionality Reduction using Probabilistic Principal Component Analysis to further reduce the number of features.
4. The reduced features per image then passed to train a Multilayer Feed-forward Neural Network.
5. The neural network outputs predicted class labels for binary classification problems

We initially perform experiments to configure our neural network. We then follow the pipeline described above for binary classification experiments. We then retrieve demographic data for the subjects and investigate the importance of these features in improvising classification performance.

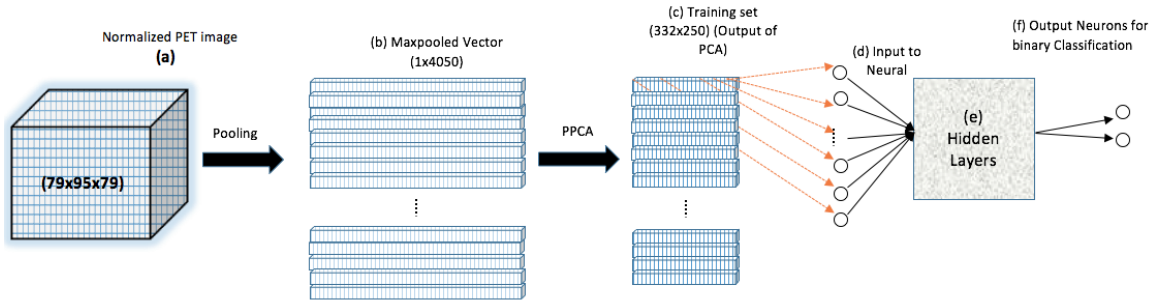

Figure 3.2: (a) Pre-processed PET data for one sample (size=79x95x79) is the initial input data to (b) Max-pooling to reduce the number of features from 79x95x79 to 1x4050. (c) Dimensionality Reduction using Probabilistic Principal Component Analysis to further reduce the number of features from 4050 to 250(specific for AD/Normal). (d)The reduced features per image then passed to train a Multilayer Feed-forward Neural Network with (e)hidden layers. (f)The neural network outputs predicted class labels

3.3 Feature Selection using Max-pooling

Feature selection, variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features. Boureau *et al.* (2010) show that max-pooling has better performance than other pooling operations. Pooling is widely used to reduce the number of features, to help boost classification performance. Since our dataset consists of 668 data samples with 592895 features each, the number of samples is much less compared to the number of features. Learning based on this representation does not help better classification. We therefore perform max-pooling to reduce the number of features by a large count, for a sample count in hundreds. Max-pooling with patches (of size $10 \times 10 \times 10$) reduces our feature dimensionality to 4050 per data sample. This also helps remove noise from our data, as the maximum value is chosen from 1000 values in a patch. The patches overlap each other in halves. We perform max-pooling on the 3-dimensional PET images to make them 2-dimensional. We then convert this 2-dimensional matrix into a uniformly formed 1-dimensional vector. Max-pooling, as shown in the transition from Fig.3.2a to Fig.3.2b converts each 3-dimensional image to a 1-dimensional vector. As shown

in our results, max-pooling outperforms mean-pooling for our binary classification experiments. We also perform experiments to show how the performance varies for varied patch sizes, please refer to Section 4.9. Since varying patch sizes with the same proportion(0.5) of overlapping between patches, also varies the number of max-pooled values (from all patches) returned for a data sample, we show the number of patches formed for varied patch sizes in Table 3.2.

Patch Size	5	6	7	8	9	10	11	12	13	14	15
#Patches	66424	18750	18750	7942	7128	4050	2160	2016	1200	1200	891

Table 3.2: Patch Size vs Number of Patches

3.4 Dimensionality Reduction using Probabilistic Principal Component Analysis

In machine learning, dimensionality reduction is the process of reducing the number of random variables under consideration, via obtaining a set of principal variables. Dimensionality reduction is the introduction of new feature space where the original features are represented. The new space is of lower dimension than the original space.

Tipping and Bishop. (1999) proposed a closed form solution to estimate Maximum Likelihood for Probabilistic Principal Component Analysis (PPCA). Principal Component Analysis (PCA) is widely used to transform data into reduced dimensionality. PCA maximizes the variance of projected data(x), which is represented in a lower dimensional space using a set of orthonormal vectors W. PPCA is the following latent variable model:

$z \sim \mathcal{N}(0, I)$;

$x \sim \mathcal{N}(Wz + \mu, \sigma^2 I)$,

where $x \in \mathbb{R}^p$ is one observation and $z \in \mathbb{R}^q$ is a latent variable vector, usually $q \ll p$. The error covariance structure in PPCA is $\sigma^2 I$. The Maximum Likelihood solution

for PPCA is obtained as:

$W_{MLE} = U_q(\Lambda_q - \sigma^2_{MLE}I)^{1/2}R,$

where U_q is a matrix of q leading principal directions(eigen values of the covariance matrix), Λ_q is a diagonal matrix of corresponding eigenvalues.

$\sigma^2_{MLE} = \frac{1}{d-q}\sum_{j=q-1}^{d}\lambda_j$ represents the variance lost in the projection and R is an arbitrary $q \times q$ rotation matrix (corresponding to rotations in the latent space).

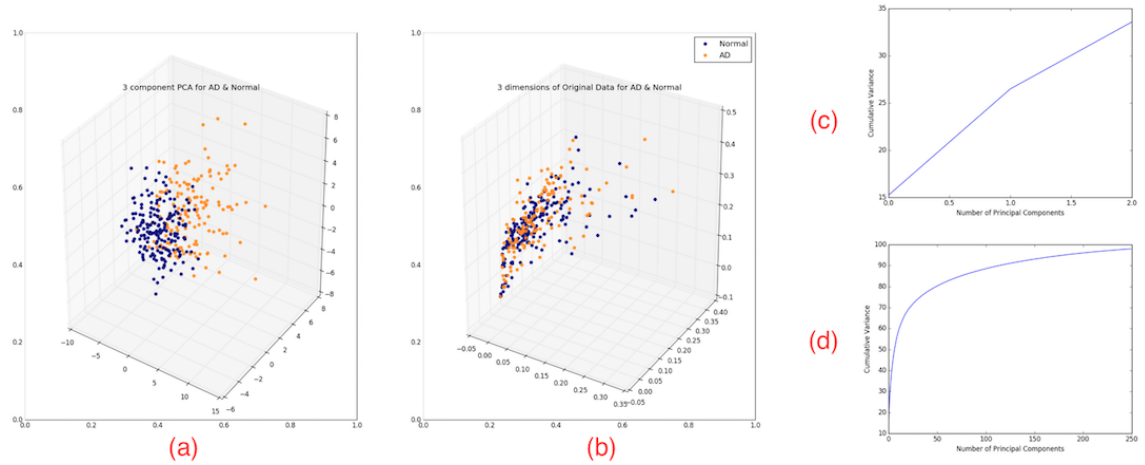

Figure 3.3: (a) 3 component PPCA for AD and Normal shows a good separation of AD and Normal subjects (b) Displaying the first 3 dimensions out of 4050 of the max-pooled data, (c) Cumulative Variance for 3 component PCA (d) Cumulative Variance for up to 250 features

Fig. 3.3(a) shows that PPCA, even with 3 components separates AD from Normal to a great extent, whereas in Fig. 3.3(b) we see that 3 of the 4050 max-pooled features are closely represented with possible overlaps for AD and Normal. The cumulative variance displayed in Fig. 3.3(c) is low ($\sim 35\%$), whereas for 250 components PPCA has a high cumulative variance ($\sim 97\%$) shown in Fig. 3.3(d). We further need to reduce our feature dimensionality from 4050 to a count in hundreds, as training a neural net with features close to the number of samples will give us a model that can perform better classifications. Hence we use PPCA to reduce our 4050 max-

pooled/mean-pooled features to features in the range 250 to 300. This range of feature dimensionality count gives the best variance from PPCA, as shown in Fig. 3.3(d).

3.5 Multilayer Perceptron (MLP) for Binary Classification

3.5.1 Perceptron

A perceptron produces a single binary output given several binary inputs x_1,x_2, and so on. Fig 3.4 below shows a schematic of Rosenblatt's Perceptron with m inputs x_1, x_2, x_3 ... x_m.

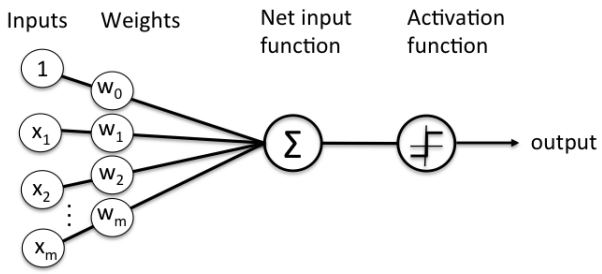

Figure 3.4: Schematic of Rosenblatt's Perceptron

Rosenblatt [1] proposed a simple way to compute the output. He assigned weights w_1, w_2, ..., to inputs signifying the importance of inputs in the determination of the output.

In the modern sense, the perceptron is an algorithm for learning a binary classifier: a function that maps its input x (a real valued vector) to an output value $f(x)$ (a single binary value):

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

$$f(x) = \begin{cases} 1 & w \cdot x + b > 0 \\ 0 & otherwise \end{cases}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product $\sum_{i=0}^{m} w_i x_i$, where m is the number of inputs to the perceptron and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value.

3.5.2 Activation Function

There are three main types of activation functions that are majorly used in MLPs, these are as follows:

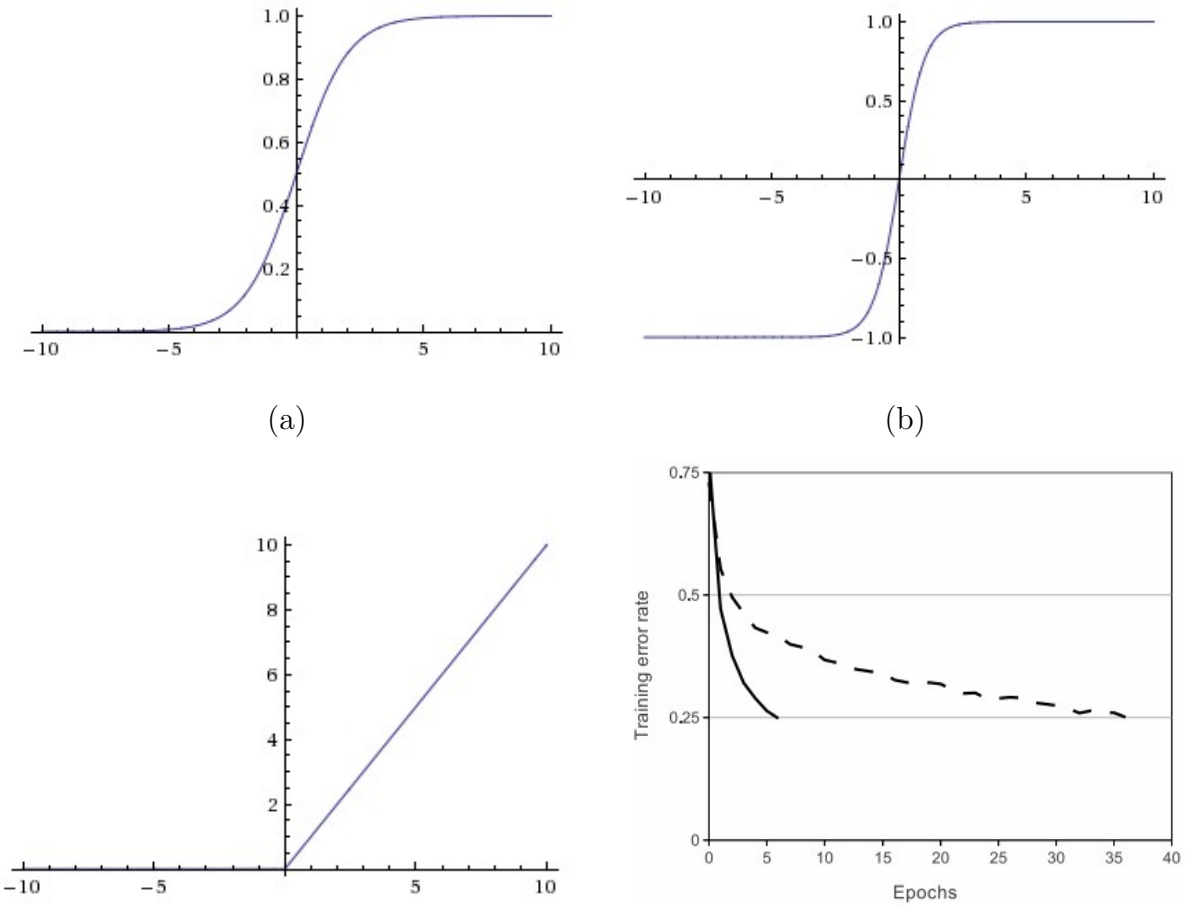

Figure 3.5: (a) Sigmoid activation function (b) Tanh activation function (c) ReLu activation function (d) 6x improvement using ReLu when compared to Tanh

1. Logistic

   Let the input for the activation function be given by:

$$\mathbf{x} = W^T x = \sum_{i=1}^{n_{features}} W_i x_i + b$$

   The logistic or *sigmoid* function can be given by

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{x})}$$

Sigmoid outputs are not zero-centered. Large numbers are squashed into a range between 0 and 1. Large positive numbers become 1 and large negative numbers become 0. The gradient at 0 and 1 is almost zero, which means there

2. Tanh

   Tanh can be considered as a rescaled version of *sigmoid*:

$$f(\mathbf{x}) = \tanh(\mathbf{x}) = \frac{e^{\mathbf{x}} - e^{-\mathbf{x}}}{e^{\mathbf{x}} + e^{-\mathbf{x}}}$$

   Tanh output is zero-centered.

3. Linear Rectification(ReLu)

   ReLu computes the function

$$\mathbf{x} = \max(0, x)$$

   , which is basically a threshold at zero. It has been found that ReLu greatly accelerates (e.g. a factor of 6 in Krizhevsky *et al.* (2012)) the convergence of stochastic gradient descent compared to the sigmoid/tanh functions for a Convolutional Neural network. It is argued that this is due to its linear, non-saturating form, shown in Figure 3.5. This is further discussed in Section **?**.

We use ReLu as it induces sparsity in our network, and it performs better as shown in 4.7. Figure 3.6 shows how the 1^{st} hidden layer has certain deactivated neurons because the output from their respective ReLu activation functions is zeroed out. Which means they do not contribute as inputs to the second hidden layer, and only 2 of the 6 neurons in the 1^{st} hidden layer contribute as inputs to the 2^{nd} hidden layer. Similarly sparsity is induced with ReLu in the 2^{nd} hidden layer.

Experimental results show engaging training behavior of this activation function, especially for deep architectures Bengio *et al.* (2009), i.e., where the number of hidden

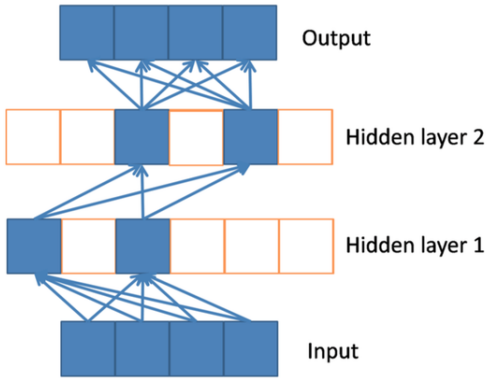

Figure 3.6: Rectified Linear Units Inducing Sparsity in a Neural Network. Bengio *et al.* (2009)

layers in the neural network is 3 or more. Which means that training proceeds better when operating neurons in a network are either off or operating mostly in a linear regime.

3.5.3 Backpropagation

Backpropagation is a method of training artificial Neural Networks used in conjunction with an optimization method (such as gradient descent). Backpropagation calculates a gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function. Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as autoencoders.
Backpropagation is used for computing the error δ^l and the gradient of the cost function. The Backpropagation algorithm is as follows:

1. Input x: Set a^1(activation) for the input layer

2. Feedforward: For each $l = 2, 3, ...L$, compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$

3. Output Error: (δ^l)

   Compute the vector $\delta^l = \nabla_a C \odot \sigma'(z^l)$

4. Backpropagate the Error: For each $l = L-1, L-2, ..., 2$ compute

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

5. Output: Gradient of the cost function is given by: $\frac{\delta C}{\delta w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\delta C}{\delta b^l_j} = \delta^l_j$

3.5.4 MLP

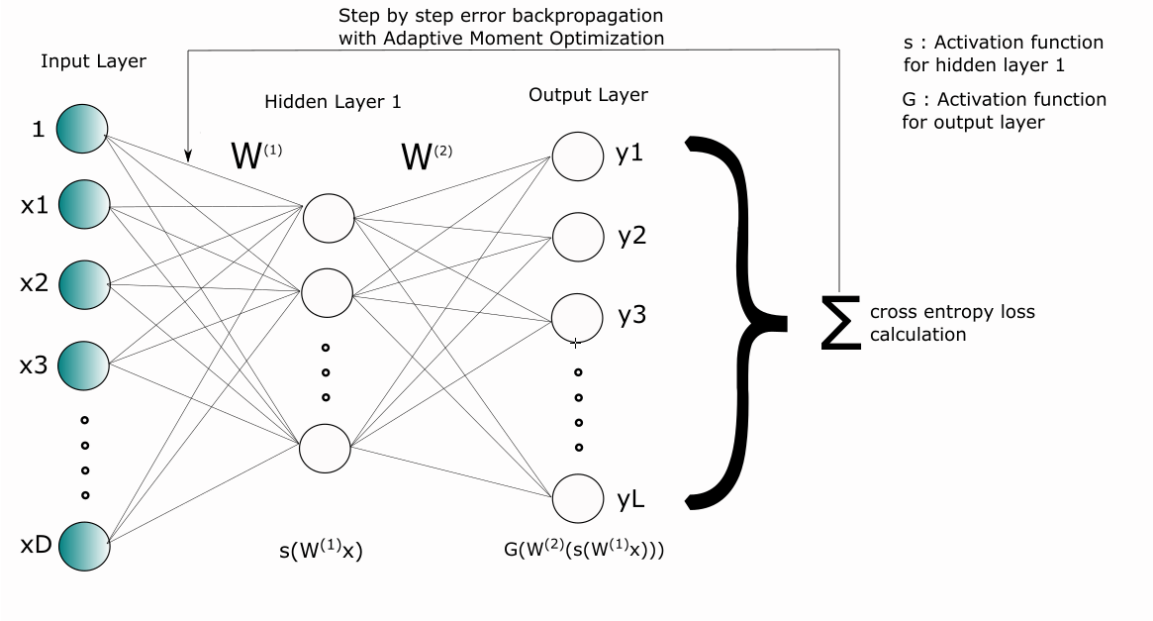

Figure 3.7: Multilayer Perceptron with 1 hidden layer

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique

called backpropagation for training the network. MLP networks are typically used in supervised learning problems. This means that there is a training set of input-output pairs and the network must learn to model the dependency between them. The supervised learning problem of the MLP can be solved with the back-propagation algorithm. The algorithm consists of two main steps, forward propagation and back propagation. In the forward pass, the predicted outputs corresponding to the given inputs are evaluated, by applying a set of weights to the input data. For the first forward propagation, the set of weights is selected randomly. In the backward pass, partial derivatives of the cost function with respect to the different parameters(W, b) are propagated back through the network. Back propagation measures the margin of error of the output and adjusts the weights accordingly to decrease the error.

A one hidden layer MLP can be represented using the function:

$$f : R^D -> R^L$$

where D is the size of input vector x and L is the size of the output vector $f(x)$.

Where,

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x)))$$

with bias vectors $b^{(1)}$, $b^{(2)}$; weight matrices $W^{(1)}, W^{(2)}$ and activation functions G and s.

We use the activation function Rectified Linear Unit for the activation units in the MLP. The rectifier activation function allows a network to easily obtain sparse representations, hence inducing the sparsity effect on networks.

MLP using a backpropagation algorithm is the standard algorithm for any supervised learning pattern recognition process and the subject of ongoing research in computational neuroscience and parallel distributed processing. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to

get approximate solutions for extremely complex problems like classification.

3.5.5 Using MLP for Binary Classification

After reducing the feature dimensionality for each sample, we pass these as inputs to our MLP. We explore various configurations to obtain the best performing models, for details go to Section 4. We vary the number of hidden layers and the number of neurons in each layer for every classification experiment. The activation function for each neuron is Linear Rectification, which given an input y, returns $f(y) = max(0, y)$. The learning rate for MLP is set to 0.001, and the loss minimization (Gradient Descent Optimization) is performed using the Adam(Adaptive Moment Estimation) Optimizer, Diederik P. Kingma (2015).

The loss minimization problem can be given by:

$$\underset{min}{W} \left\{ L(W) := \frac{1}{m} \sum_{i=1}^{m} \ell(W; x_i, y_i) + \lambda r(W) \right\}$$

where, $\{x_i, y_i\}_{i=1}^{m}$ are training instances for (x_i) and corresponding labels; (y_i)
W - network parameters to learn; $\ell(W_i; x_i, y_i)$ - loss of network parameterized by W w.r.t (x_i, y_i); $r(W)$ - regularization function. $(e.g. \|W\|_2^2$, square of the L2 norm); $\lambda < 0$ - regularization weight.
Optimization methods must be, **First-order** - update based on objective value and gradient only; and **Stochastic** - update based on subset of training examples.
$L_t(W) := \frac{1}{b} \sum_{j=1}^{b} \ell(W; x_{i_j}, y_{i_j}) + \lambda r(W)$
$(x_i, y_i)_{j=1}^{b}$- random mini-batch chosen at iteration t.
The update rule for Adam Optimization is:

$$W_t = W_{t-1} - \alpha \frac{\hat{M}_t}{\sqrt{\hat{R}_t + \epsilon}}$$

where,

$M_t = \beta_1 M_{t-1} + (1 - \beta_1)\Delta L_t(W_{t-1})$ (1^{st} moment estimate);

$R_t = \beta_2 R_{t-1} + (1 - \beta_2)\Delta L_t(W_{t-1})^2$ (2^{nd} moment estimate);

$\hat{M}_t = M_t/(1 - (\beta_1)^t)$ (1^{st} moment bias correction);

$\hat{R}_t = R_t/(1 - (\beta_2)^t)$ (2^{nd} moment bias correction).

The Hyper-parameters are: $\alpha > 0$ - learning rate (choice : 0.001);

$\beta_1 \in [0, 1)$ - 1^{st}moment decay rate (choice : 0.9);

$\beta_2 \in [0, 1)$ - 2^{nd} moment decay rate (choice : 0.999);

$\epsilon > 0$ - numerical term (typical choice: 10^{-8}).

Adam adaptively selects a separate learning rate for each parameter. Parameters that would ordinarily receive smaller or less frequent updates receive larger updates with Adam (the reverse is also true). This speeds learning in cases where the appropriate learning rates vary across parameters.

3.6 Cross Validation

Cross validation is a widely used method to measure the predictive performance of a model, Stone (1974). Cross validation helps maximize the total number of data points used for testing. We use K-fold Cross Validation with the value of $K = 10$. In 10 fold cross-validation the data is divided into 10 equally sized segments. After the dataset is divided into 10 parts, one part is held out for testing and the model is trained on the remaining 9 parts, and tested on the held out part. The confusion matrix is stored in a 1-dimensional matrix $M_{2\times 2}$. This process is repeated for each segment, and an addition over all such matrices M is performed to get the overall

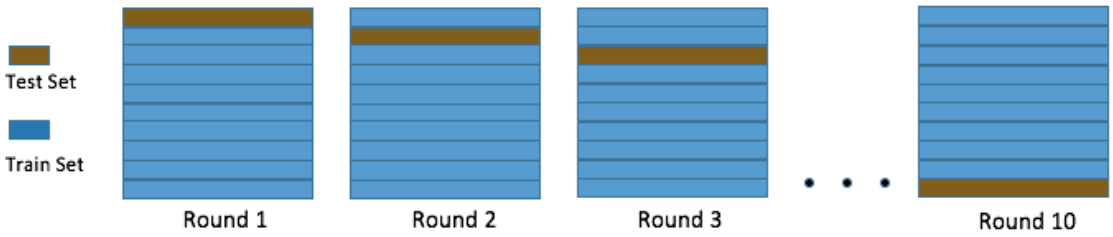

Figure 3.8: 10-fold Cross Validation

performance metrics of our model. Figure 3.8 helps visualize the cross validation procedure for the entire dataset. In Round 1 the first segment is held out for testing, in Round 2 the second segment is used for testing and so on, till Round 10 where the 10^{th} segment is used for testing. This way we are able to use the entire dataset once for testing.

3.7 Performance Metrics

		Predicted class	
		Class 1	Class 0
Actual class	Class 1	true positives (TP)	false negatives (FN)
	Class 0	false positives (FP)	true negatives (TN)

Figure 3.9: Confusion Matrix

Figure 3.9 is a confusion matrix generally used to measure binary classification performance. Out of the various performance measures derivable from a confusion matrix, the performance measures used in this work to compare various classification models are:

- True Negatives (TN): The number of examples belonging to the negative class, with a predicted output of negative class i.e. examples correctly classified as

negative.

- True Positive (TP): The number of examples belonging to the positive class, with a predicted output of positive class i.e. number of examples correctly classified as positive.

- Precision:

$$\frac{TP}{TP + FP}$$

  The ratio of the number of positive examples correctly classified as positive divided by the total number of examples classified as positive class.

- Recall:

$$\frac{TP}{TP + FN}$$

  The ratio of the number of positive examples correctly classified to the total number of positive examples

- F_1 Score:

$$\frac{2TP}{2TP + FP + FN}$$

  , is a measure that is the harmonic mean of precision and recall. Recall and Precision are evenly weighed in this measure.

Chapter 4

EXPERIMENTS

We compare the performance for various configurations of our neural network.

4.1 System Configuration

The experiments performed in this section have been performed on a node assigned on Saguaro(HPC cluster). MATLAB is used for pooling the FDG-PET data using max-pooling and mean-pooling. The Software toolkit used for dimensionality reduction(PPCA), classification(MLP) and evaluation (10 fold cross-validation) is Scikit-Learn [1] . Batch scripts were used to perform lengthier experiments (comparison).

4.2 Assumptions

These experimentations are done with an assumption that no stand-alone methods exist, which can classify even with binary experiments, all diagnostic categories, as explained in Section 2.1.1. We also have a dis-balance of classes in our binary experiments. We assume that since there is no high amount of dis-balance between class counts, we can safely ignore this and tune our parameters for a better classifier. With a high amount of dis-balance it is possible that the classifier trained might be biased to the class present in a larger number.

Our classifiers also do not specifically consider regional differences between data samples. Regional analysis included in classifiers may result into better performance.

[1] https://www.scikit-learn.org

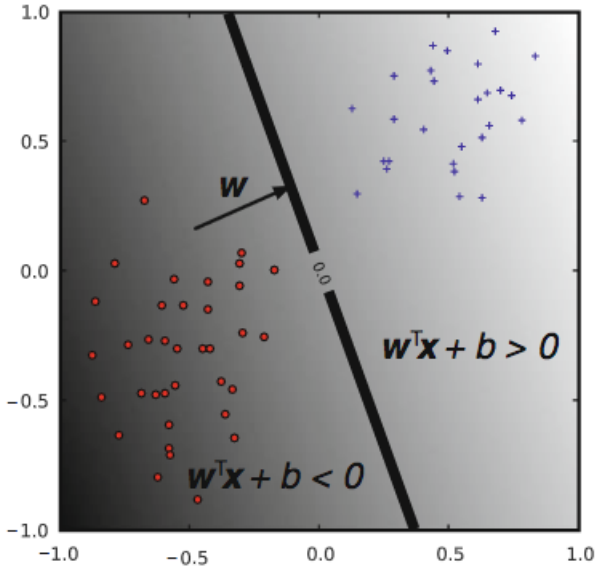

Figure 4.1: Linear Classifier (SVM). Ben-Hur and Weston (2010)

We run Linear Support Vector Machine (SVM) on our data to test its linear separability. Running linear SVM as shown previously by Asa Ben-Hur et. al. Ben-Hur and Weston (2010) can display the extent of linear separability between the various categories of data. Fig. 4.1 shows how linearly separable classes can be classified using a linear classifier like SVM. The hyper-plane (line in 2-d) is the classifiers decision boundary. A point is classified according to which side of the hyper-plane it falls on, which is determined by the sign of the discriminant function.

Running Linear SVM on max-pooled data gives us results (measured as f_1-score) as shown in Table 4.1. From the table we see that AD and CU are to a large extent linearly separable (with a few incorrect classifications), and LMCI EMCI are not linearly separable, because the f_1-score is 0.62.

Linear SVM	AD CU	AD MCI	CU MCI	EMCI AD	LMCI AD	EMCI CU	LMCI EMCI	LMCI CU
F1-Score	0.9291	0.8433	0.7082	0.8138	0.6882	0.6377	0.6260	0.6507

Table 4.1: Linear SVM

4.4 Sparse Representation of Data

This non-linearity of data can be visualized as a dense representation of the data points. Hence sparse representations are used for the following reasons (for details please refer to Glorot *et al.* (2011)):

- Information disentangling: To disentangle the factors explaining the variations in the data. Lets say we have a densely populated region with data points (having dense representations, for example smaller dimensions) from various classes. Learning a model on this representation does not guarantee correct classification for points that slightly vary in feature values, from the points the model has trained on.

  If a representation is both sparse and robust to small changes in feature values, the non-zero features will be conserved throughout training.

- Efficient variable size representation: Due to the varying amount of information contained in every input, the number of active input neurons will vary. This may help control dimensionality of the representation for every input.

- Linear Separability: Sparse representations induce linear separability of data due to high dimensionality of sparse representations.

We use Rectified Linear Units (activation function 3.5.2) to induce sparsity in our Neural Network.

4.5 Maximum Variance Explained in PPCA

For each of the 8 binary experiments we estimate the number of Principal Components with maximum variance explained. We choose to go with the number of components that can together explain a variance of around ∼99%.

Table 4.2 shows our choice of the number of Principal Components chosen using PPCA.

Experiment	AD CU	AD MCI	CU MCI	AD EMCI	AD LMCI	CU LMCI	CU EMCI	EMCI LMCI
#PCs	300	400	500	300	300	320	340	310

Table 4.2: #Principal Components Chosen for Each Experiment.

4.6 Effect of Number of Hidden Layers

To compare the effect of number of hidden layers in a neural network on classification performance, we construct 2 types of neural networks, one hidden layer MLP(with x number of neurons, as given formerly in Section 2.5) and 2 hidden layer MLP ($inp+1$ neurons in the first, and 10 in the second hidden layer), the performance comparison for max-pooled data is given in Table 4.3. The values in bold clearly depict that a 2 hidden layer MLP outperforms a one hidden layer MLP.

Table 4.3: Performance Comparison: 1 Hidden Layer vs 2 Hidden Layer(HL) MLP

MLP	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
	F1	0.90	0.85	**0.74**	0.82	0.67	0.68	0.60	**0.64**
1 HL	Prec	0.94	0.87	0.79	0.79	0.67	0.70	0.61	0.66
	Recall	0.86	0.82	0.70	0.84	0.67	0.66	0.59	0.63
	F1	**0.92**	**0.86**	0.73	0.82	**0.70**	**0.69**	**0.63**	0.62
2 HL	Prec	0.97	0.90	0.77	0.79	0.69	0.69	0.65	0.63
	Recall	0.87	0.86	0.70	0.84	0.71	0.68	0.60	0.60

We compare the performance on Max-pooled Data vs. Meanpooled data

4.7 Comparison of Max-Pooled Data with Mean-Pooled Data

We performed Binary Classification experiments on 2 types of datasets (one with max-pooling and the other mean-pooling). We perform max-pooling on our data and mean-pooling on our data. We then compare the two datasets based on classification performance achieved by using a Multilayer Perceptron(MLP) classifier. Table 4.4 shows that **max-pooling helps achieve better performance in majority of the binary classification experiments**.

4.8 Varying Number of Neurons for F1 Measure of Accuracy

We also compare the performance based on the number of neurons in a 1 hidden layer MLP. This comparison is done on AD and CU subject data, with max-pooled features and PPCA resulting into 300 components. This data is then passed through a 1 hidden layer Multilayer Perceptron. Fig. 4.2 shows a comparison of number

Table 4.4: Classification Comparison for Max-Pooled and Mean-Pooled Data

Performance Comparison	Pooling	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
F-1 score	Max	0.92	0.86	**0.75**	0.82	0.72	**0.70**	**0.63**	**0.64**
	Mean	**0.93**	0.86	0.71	**0.85**	**0.74**	0.63	0.57	0.61
Precision	Max	0.97	0.90	0.82	0.80	0.73	0.73	0.65	0.64
	Mean	0.96	0.89	0.73	0.87	0.77	0.62	0.59	0.59
Recall	Max	0.87	0.83	0.70	0.84	0.71	0.67	0.60	0.64
	Mean	0.90	0.83	0.70	0.82	0.71	0.65	0.55	0.63
NPV	Max	0.82	0.57	0.37	0.88	0.73	0.58	0.55	0.60
	Mean	0.87	0.58	0.42	0.77	0.66	0.72	0.54	0.70
PPV	Max	0.97	0.90	0.82	0.80	0.73	0.73	0.65	0.64
	Mean	0.96	0.89	0.73	0.87	0.77	0.62	0.60	0.59

of neurons vs f_1-measures of accuracy obtained. The number of neurons have been varied from 5 to 1000, and we see there is **no specific pattern that displays a trend in f_1 measure values varying with numbr of neurons**. The maximum f_1 score in this case is obtained for 700 neurons and the minimum f_1 measure is obtained for 15 neurons.

4.9 Effect of Patch Size on Classification Performance

We compare performance based on patch size for AD and CU subject data. For this experiment we used max-pooled data with age, gender, apoe1, apoe2 and FAQ score for all data samples. In max-pooling 3-dimensional patches of size $p_{size} \times p_{size} \times$

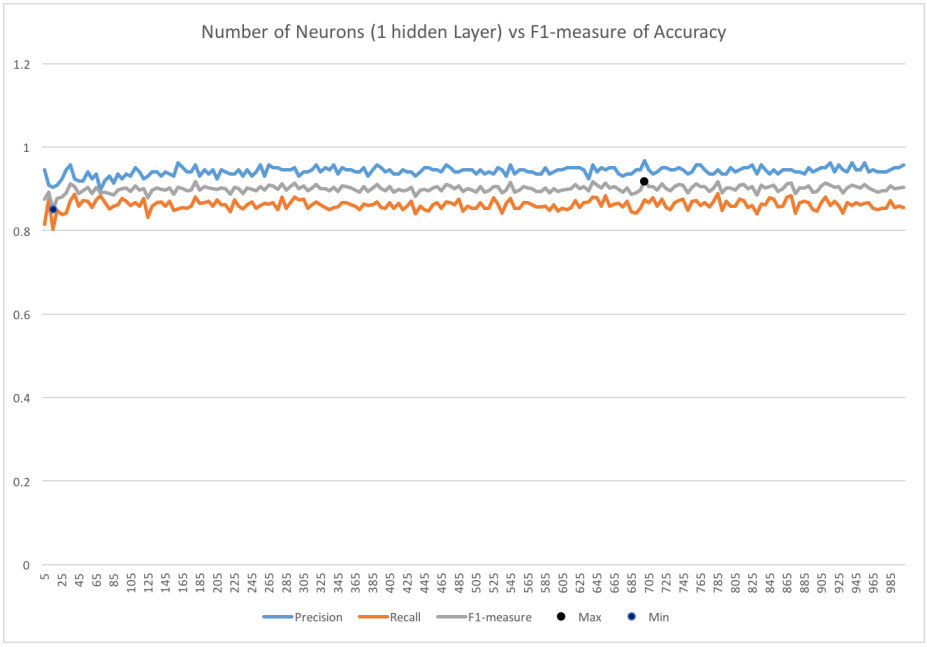

Figure 4.2: Number of Neurons in One Hidden Layer MLP vs F_1-Measure with Max-pooled Data

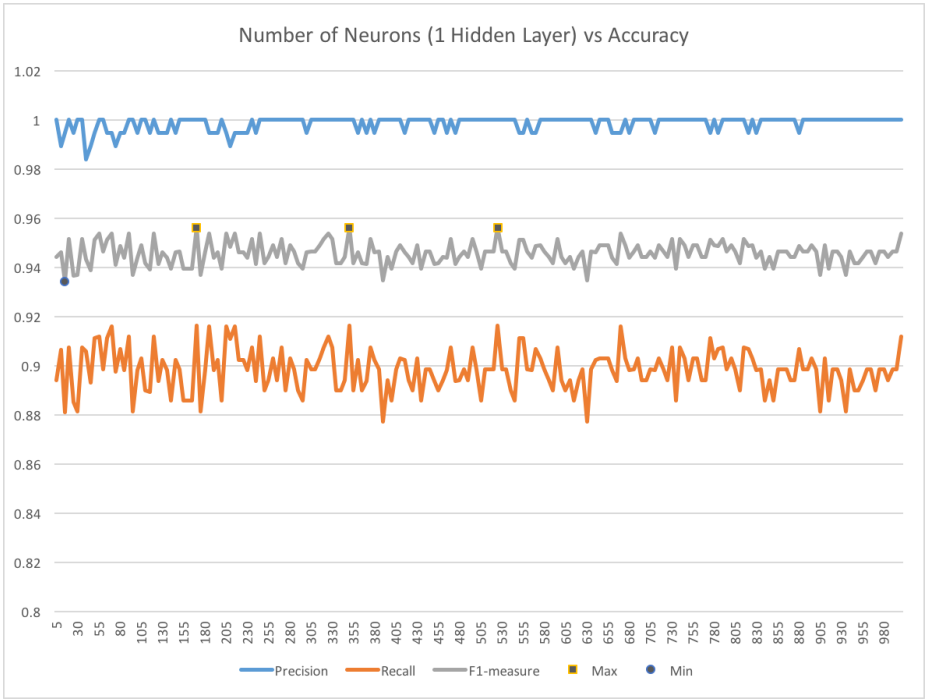

Figure 4.3: Number of Neurons in One Hidden Layer MLP vs F_1-Measure with Max-pooled + Demographic Data

Table 4.5: Performance Comparison: Patch Size vs F1-Accuracy

	Without Demographics			With Demographics		
Patch Size (n)	F1	Precision	Recall	F1	Precision	Recall
5	**0.9312**	0.9462	0.9167	0.9474	0.9677	0.9278
6	0.9175	0.9570	0.8812	0.9708	0.9839	0.9581
7	0.9044	0.9409	0.8706	0.9710	0.9892	0.9534
8	0.9271	0.9570	0.8990	0.9735	0.9892	0.9583
9	0.9231	0.9677	0.8824	**0.9737**	0.9946	9536
10	0.9255	0.8867	0.9677	0.9735	0.9839	0.9632
11	0.9251	0.9624	0.8905	0.9661	0.9946	0.9391
12	0.9299	0.9624	0.8995	0.9661	0.9946	0.9391
13	0.8144	0.7822	0.8495	0.9561	0.9946	0.9204
14	0.9199	0.9570	0.8856	0.9634	0.9892	0.9388
15	0.9133	0.9624	0.8689	0.9609	0.9892	0.9340

p_{size} are extracted uniformly from the 3-dimensional intensity value data. We vary the patch sizes in the range $p_{size} = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$. For the experimentation without demographic features and just max-pooled values, the number of hidden layers is given by $n_{hidden} = 4$ and the number of neurons in each hidden layer are $n_{neurons} = [1000, 500, 100, 10]$. For the experimentation with demographic features in addition to max-pooled values as the input, the number of hidden layers is given by $n_{hidden} = 7$ and the number of neurons in each hidden layer are $n_{neurons} = [1000, 800, 600, 400, 200, 100, 10]$.We compare both max-pooled data with demographics and performance on just max-pooled data. Table 4.5 Figures 4.4,4.5,4.6 depict graphs for the same. This shows an increase in F_1, precision and recall for 7 out of the 8 experiments, except for EMCI vs LMCI.

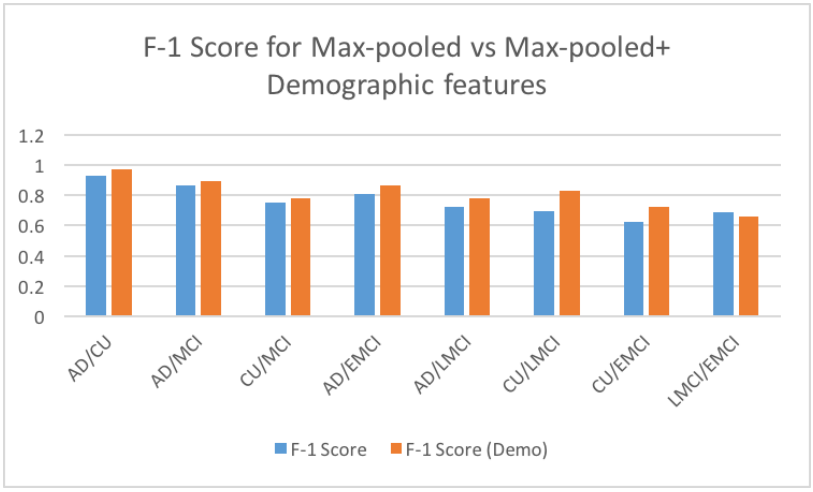

Figure 4.4: F_1 Measure Comparison with Addition of Demographic Features

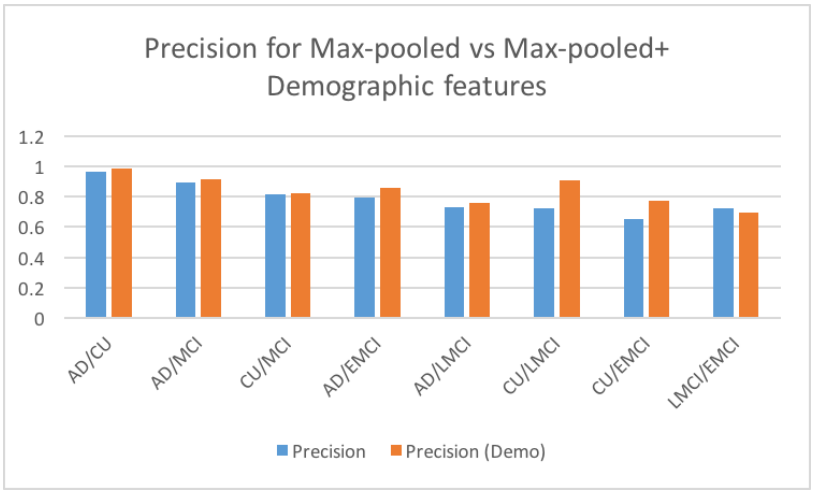

Figure 4.5: Precision Measure Comparison with Addition of Demographic Features

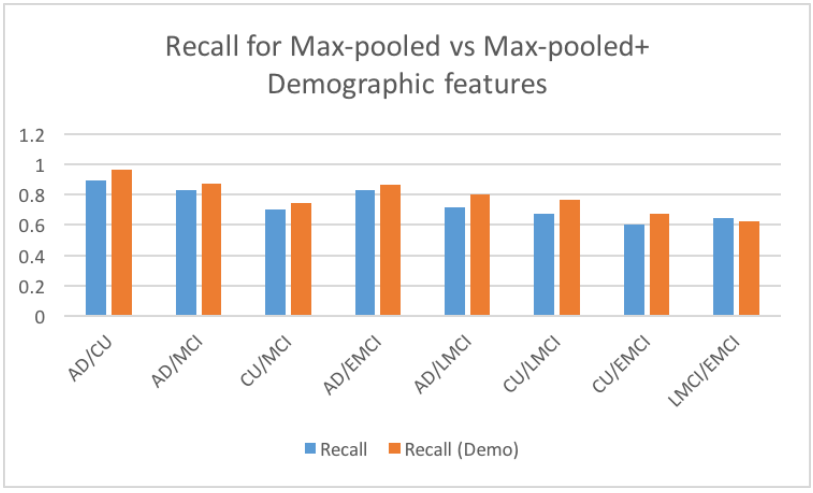

Figure 4.6: Recall Measure Comparison with Addition of Demographic Features

We select two patch sizes ($p_{size} = 9, p_{size} = 10$) to experiment with, on **Max-pooled data along with 4 demographic features**. The performance of both patch sizes is comparable and almost similar. Comparison for the two is shown in Table 4.6.

Table 4.6: Performance Comparison: Patch Size 9 vs Patch Size 10

Patch Size	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
$p_{size} = 9$	F_1 score	**0.9737**	0.8899	0.7713	**0.8632**	0.7778	0.8060	**0.7246**	**0.6809**
	Precision	0.9928	0.9137	0.7679	0.8425	0.7671	0.8710	0.7850	0.7191
	Recall	0.9536	0.8672	0.7748	0.8849	0.7887	0.75	0.6728	0.6465
$p_{size} = 10$	F_1 score	0.9734	**0.8953**	**0.7830**	0.8621	**0.7789**	**0.8325**	0.72	0.656
	Precision	0.9839	0.9167	0.8214	0.8562	0.7603	0.9086	0.7742	0.6910
	Recall	0.9632	0.8750	0.7480	0.8681	0.7986	0.7682	0.6729	0.6244

4.10 Effect of Activation Function for Classification

We compare performance based on activation function. There are three main types of activation functions that are majorly used in MLPs, these are as follows, explained further in Section. 3.5.2:

1. Tanh
2. Logistic
3. Linear Rectification(ReLu)

The experiments performed are on a Multilayer Perceptron with configuration: $n_{hidden} = 4$ and $n_{neurons} = [1000, 500, 100, 10]$.

Table 4.7: Performance Comparison: Activation Function vs F1-Accuracy for AD/CU

Activation Function	F1	Precision	Recalll
Tanh	0.9057	0.9032	0.9081
Logistic	0.7181	1.0	0.5602
ReLu	**0.9255**	0.8867	0.9677

We can conclude that the ReLu is better at classification performance in this case.

4.11 Finding an Optimal Configuration

We further experiment to find an optimal deep neural network configuration for the MLP architecture. Our experimentation is as shown in the algorithm , which is purely trying to estimate in one fixed direction. According to this algorithm, we estimate a one hidden layer MLP that gives the maximum f_1 score when neurons are varied from 5 to 1000. We then fix the number of neurons in the first layer to the one that gives us the best f_1 score, and vary the number of neurons in the second layer from 5 to 1000, and estimate the number of neurons that give us the best f_1 score. We keep doing this for iterations counting the number of hidden layers given as input. Since this is purely based on an assumption that fixing the previous neural network configurations and varying the new hidden layer configuration will lead to better results, this may not be true. Also we only vary the number of neurons from 5 to 1000 at intervals of 5, increasing the upper and lower bound for this experiment may lead to different results. To investigate this approach, we use it for the classification of 7(excluding CU MCI) of the binary classification experiments. The results are as shown in Tables 4.8-4.14

Table 4.8: Estimating an Optimal Configuration for AD CU Classification

	AD vs CU		AD vs CU with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(700)	**0.9430**	(525)	0.9563
2	(700,555)	0.9415	(525,880)	0.9737
3	(700,555,305)	0.9403	(525,880,880)	0.9761
4	(700,555,305,25)	0.9368	(525,880,880,255)	0.9812
5	(700,555,305,25,10)	0.9393	(525,880,880,255,775)	**0.9814**

Table 4.9: Estimating an Optimal Configuration for AD MCI Classification

	AD vs MCI		AD vs MCI with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(85)	0.8684	(160)	0.9086
2	(85,120)	0.8727	(160,270)	0.9125
3	(85,120,110)	**0.8743**	(160,270,405)	0.9083
4	(85,120,110,625)	0.8734	(160,270,405,350)	0.9086
5	(85,120,110,625,120)	0.8677	(160,270,405,350,215)	**0.9140**

Table 4.10: Estimating an Optimal Configuration for EMCI AD Classification

	EMCI vs AD		EMCI vs AD with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(755)	0.8696	(80)	0.9003
2	(755,55)	0.8736	(80,190)	0.9011
3	(755,55,625)	**0.8747**	(80,190,380)	**0.9036**
4	(755,55,625,15)	0.8641	(80,190,380,425)	0.9000
5	(755,55,625,15,585)	0.8644	(80,190,380,425,550)	0.8950

Table 4.11: Estimating an Optimal Configuration for LMCI AD Classification

	LMCI vs AD		LMCI vs AD with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(380)	0.7561	(215)	**0.8288**
2	(380,660)	**0.7706**	(215,105)	0.8193
3	(380,660,70)	0.7688	(215,105,150)	0.8098
4	(380,660,70,55)	0.7679	(215,105,150,600)	0.8086
5	(380,660,70,55,535)	0.7580	(215,105,150,600,260)	0.8086

Table 4.12: Estimating an Optimal Configuration for LMCI CU Classification

	LMCI vs CU		LMCI vs CU with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(915)	0.6512	(560)	0.7324
2	(915,20)	0.6536	(560,490)	**0.7774**
3	(915,20,690)	**0.6539**	(560,490,35)	0.7747
4	(915,20,690,170)	0.6471	(560,490,35,40)	0.7735
5	(915,20,690,170,15)	0.6507	(560,490,35,40,75)	0.7671

Table 4.13: Estimating an Optimal Configuration for EMCI CU Classification

	EMCI vs CU		EMCI vs CU with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(5)	0.6044	(930)	0.6564
2	(5,25)	0.6192	(930,860)	0.6866
3	(5,25,5)	**0.6388**	(930,860,130)	0.6961
4	(5,25,5,185)	0.6287	(930,860,130,385)	**0.7015**
5	(5,25,5,185,5)	0.6293	(930,860,130,385,750)	0.6859

Table 4.14: Estimating an Optimal Configuration for EMCI LMCI Classification

	EMCI vs LMCI		EMCI vs LMCI with Demo	
#HL	Config	F_1 Score	Config	F_1 Score
1	(525)	0.6258	(125)	0.6214
2	(525,360)	**0.6275**	(125,585)	0.6205
3	(525,360,285)	0.6032	(125,585,5)	0.6467
4	(525,260,285,250)	0.5828	(125,585,5,430)	**0.6519**
5	(525,360,285,250,750)	0.6024	(125,585,5,125)	0.6103

This helps us reach better accuracies than a random brute force approach. A better approach would be to try all permutations and combinations for each hidden layer, but this approach is computationally expensive.

```
function hiddenLayerConf = findHiddenLayerSizes(int n){

  hiddenLayerSizes = zeros(n)
  for(int i = 1; i <= n ;i++){
    maxF1=0
    maxNN=0
    for(nn = 5; nn <= 1000, nn += 5){
      hiddenLayerSizes[i] = nn
      model = MLPClassifier(hiddenLayerSizes, 'adam', 'relu')
      model.fit(trainKfold,labelKfold)
      f1Score = model.predict(testData)
      if (maxF1 < f1Score){
        maxF1 = f1Score
        maxNN = nn
      }
      hiddenLayerSizes[i] = maxNN
    }
  }
  return hiddenLayerSizes
}
```

Figure 4.7: Algorithm to Improvise Performance of an n-Hidden Layer MLP

Algorithm Pseudo Code for Estimating an Optimal Configuration for n hidden layers

arr_{sizes} =emptyList([])

$n_{iter} = n$

for i $\leftarrow$ [1,2,..n_{iter}] **do**

for numNeurons = 5;numNeurons+= 5;numNeurons$\leq$ 1000 **do**

for k_{fold} in 10 fold Cross Validation **do**

model$\leftarrow$MLPClassifier(hiddenLayerSizes=arr_{sizes}.append(numNeurons),activation ='ReLu', solver='Adam',maxIterations=1000)

{Now train on k-1 folds and test on k^{th} fold}

model.fit([$1_{fold}, 2_{fold}, ...k-1_{fold}$], trainLabels)

{Add all confusion matrices for k-folds}

confusionMatrix$\leftarrow$confusionMatrix+confMat(testLabels,model.predict(k^{th}fold))

end for

{Store the number of neurons for which we have the maximum f_1 score}

if confusionMatrix.$f_1 >$ max f_1 **then**

max $f_1 \leftarrow$confusionMatrix.f_1

bestNumNeurons$\leftarrow$ numNeurons

end if

end for

arr_{sizes}[i]$\leftarrow$bestNumNeurons

end for

Chapter 5

RESULTS

Figure 5.1 and 5.2 show the ROC and AUCs for each of our experiments with demographic data and without demographic data. In this section, we perform experiments to validate our proposed method on the ADNI2 dataset for evaluating its performance. We use FDG-PET baseline scans from the ADNI2 dataset. We use the software toolkit Statistical Parametric Mapping (SPM) Penny *et al.* (2011) to linearly align all the images into a common space. We measure the classificaton accuracy using f_1-measure. To evaluate the performance of our method, we perform 10-fold cross validation.

Our experiments show that the proposed system is promising for AD diagnosis research. Whether or not this approach provides more statistical power than those afforded by other classification work requires careful validation for each application. We anticipate that this work will inspire more work that builds deep learning based systems for FDG-PET data analysis. We compare our results with the best achieved results as of yet using FDG-PET images long with other biomarkers, our results show

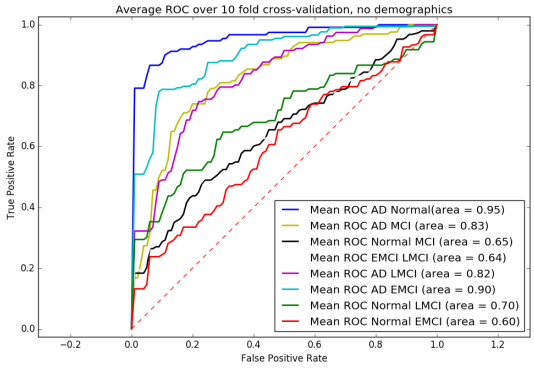

Figure 5.1: ROC for 10 Fold Cross Validation(MLP), No Demographics

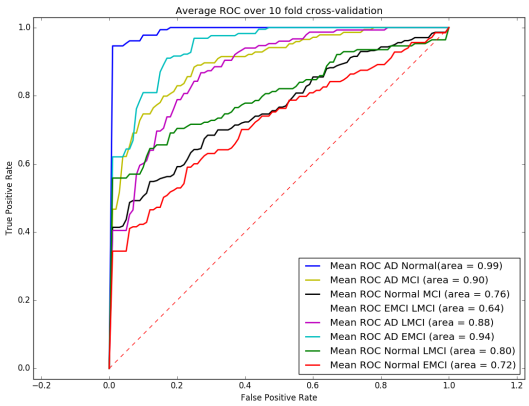

Figure 5.2: ROC for 10 Fold Cross Validation(MLP), with Demographics

an increase by 4.94% in the accuracy for AD Normal Zhang *et al.* (2011), and an increase by 14.85% for MCI Normal classification. Our technique performs much better compared to methods where manifolds are constructed based on pairwise similarity measures derived from random forest classifiers Gray *et al.* (2013).

5.1 Effect of Demographic Features

We also include age and gender of the subjects as additional features to our max-pooled data. Our data matrix, input to the neural network is in this case of size $n \times 4052$ where n is the number of samples (training + testing). n is 332 in the case of CU AD binary classification experiment (186 CU samples, 146 AD samples). We see in Table 5.1 the improvisations with the addition of age and gender(assigned values 0 for female and 1 for male) to max-pooled data. We further add the gene information available (apoe1 and apoe2), and FAQ scores as features to investigate the effects on performance of our classifier. Our results clearly display a difference with the addition of two demographic features and 4 demographic features(with APOE-gene information and FAQ scores). This means an addition of more features from ADNI subjects (such as MMSE score) will further help improvise prediction results. We try random permutations and combinations to achieve these f_1 score values.

Table 5.1: Performance Comparison: Max-Pooled Data with Addition of Demographic Features

MLP	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
Max-pooled Data	F1	0.9275	0.8612	0.7527	0.8112	0.7230	0.6976	0.6253	**0.6844**
	Prec	0.9624	0.8958	0.8155	0.7945	0.7328	0.7258	0.6505	**0.7247**
	Recall	0.895	0.8292	0.6990	0.8286	0.7133	0.6716	0.6020	0.6482
Max-pooled + Age + Gender	F1	0.9326	0.8632	0.7531	0.8211	0.7569	0.7413	0.6064	0.6813
	Prec	0.9677	0.9018	0.8125	0.8014	0.7466	0.8011	0.6129	0.6966
	Recall	0.9	0.8278	0.7018	0.8417	0.7676	0.6898	0.6	**0.6667**
Max-pooled + Age + Gender + APOE + FAQ Score	F1	**0.9734**	**0.8954**	**0.7830**	**0.8621**	**0.7790**	**0.8325**	**0.72**	0.656
	Prec	**0.9839**	**0.9167**	**0.8214**	**0.8562**	**0.7603**	**0.9086**	**0.7742**	0.6910
	Recall	**0.9632**	**0.875**	**0.7479**	**0.8681**	**0.7986**	**0.7682**	**0.6729**	0.6244

5.2 Comparison with Other Classification Algorithms

We compare our MLP results with other machine learning algorithms. This comparison has been done on max-pooled data combined with age, gender, APOE 1, APOE 2, and FAQ score information using 10 fold cross validation for each of the methods.Table 5.2 shows that MLP outperforms other methods when used with PPCA to reduce dimensions. PPCA was applied to other techniques as well, to compare the performance of classifiers on attained equivalent features for each experiment.

5.3 Comparison with Other Dimensionality Reduction Algorithms

Table 5.3 shows comparison of PPCA as a dimensionality reduction technique to other prominently used techniques. This experimentation was done on max-pooled

Table 5.2: Performance Comparison: MLP vs Other Machine Learning Algorithms

Method	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
PPCA+MLP	F1	**0.9734**	**0.8954**	**0.7830**	**0.8621**	**0.7790**	**0.8325**	**0.72**	**0.656**
	Prec	0.9839	**0.9167**	**0.8214**	**0.8562**	**0.7603**	**0.9086**	**0.7742**	**0.6910**
	Recall	0.9632	0.875	0.7479	**0.8681**	**0.7986**	**0.7682**	**0.6729**	**0.6244**
PPCA + Linear SVM	F1	0.9558	0.8781	0.7625	0.8522	0.7279	0.7413	0.6598	0.6136
	Prec	**0.9892**	0.8899	0.75	0.8493	0.7329	0.7473	0.6882	0.6067
	Recall	0.9246	0.8667	**0.7754**	0.8552	0.7230	0.7354	0.6337	0.6207
PPCA + SGD Classifier	F1	0.9551	0.8846	0.7463	0.8255	0.6957	0.7287	0.6773	0.5977
	Prec	0.9731	0.8899	0.7440	0.8425	0.7123	0.7366	0.6828	0.5843
	Recall	0.9378	**0.8794**	0.7485	0.8092	0.6797	0.7211	0.6720	0.6118
PPCA+GNB Classifier	F1	0.9080	0.8113	0.7098	0.7451	0.6351	0.6841	0.6067	0.6011
	Prec	0.8495	0.8125	0.7024	0.7808	0.6438	0.7043	0.6344	0.6180
	Recall	**0.9753**	0.8101	0.7173	0.7125	0.6267	0.6650	0.5813	0.5851

data along with demographic features(age, gender, apoe and FAQ scores), and 10 fold cross validation to evaluate performance. We see that PPCA outperforms the other techniques. These accuracies are achieved by trying random combinations of neural networks.

We follow the algorithm in Section 4.11, and our best results for the classification of FDG-PET data with and without demographics is shown in Table 5.4. These results are obtained for varying configurations of MLP, each of which has been discussed in this section and the previous sections.

Table 5.3: Performance Comparison: PPCA vs Other Dimensionality Reduction Algorithms

Method	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
PPCA + MLP	F1	**0.9734**	0.8954	**0.7830**	**0.8621**	**0.7790**	**0.8325**	0.72	**0.656**
	Prec	0.9839	0.9167	**0.8214**	**0.8562**	**0.7603**	**0.9086**	0.7742	**0.6910**
	Recall	**0.9632**	**0.875**	0.7479	0.8681	**0.7986**	0.7682	0.6729	**0.6244**
Truncated SVD + MLP	F1	0.9526	**0.9053**	0.7734	0.7473	0.7596	0.79	0.6667	0.6062
	Prec	0.9731	**0.9673**	0.7619	0.7192	0.7466	0.8495	0.7097	0.6573
	Recall	0.9330	0.8508	**0.7853**	0.7778	0.7730	0.7383	0.6286	0.5625
Kernel PCA + MLP	F1	0.9659	0.8937	0.7598	0.8489	0.7622	0.8082	**0.735**	0.64
	Prec	**0.9859**	0.9137	0.7530	0.8082	0.7466	0.8495	**0.7903**	0.6742
	Recall	0.9436	0.8746	0.7667	**0.8940**	0.7786	**0.7707**	**0.6869**	0.6091

Table 5.4: Summary of Best Results

Data	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
No Demo	F_1 score	0.9430	0.8743	0.7527	0.8747	0.7706	0.6976	0.6388	**0.6844**
Demo	F_1 score	**0.9814**	**0.9125**	**0.7858**	**0.9036**	**0.8288**	**0.8325**	**0.72**	0.656

Chapter 6

DISCUSSION

We discussed our experiments and analysed them in the previous two chapters 4 and 5. We will further analyse our findings through the experiments we performed, and the limitations in our work, which can further be improved by use of the better frameworks available for deep learning.

6.1 Findings

As seen in Section 4.6, increasing the number of layers in a Multilayer Perceptron helps increase performance on classification. This is because increasing the depth of the network generally helps the classifier learn complex functions that can fit over the data. That is why in our later experiments we use at least 4 layers, which when compared to lesser number of layers gives us better results.

Max-pooling as shown before by Boureau *et al.* (2010), has performed better than other pooling methods. We try the method on our 8 classification experiments in Section 4.7 and max-pooling performs better for the majority. The difference between the two pooling methods is significant for the classes that are in general difficult to classify (CU, EMCI, LMCI), hence we use Max-pooled data for all our later experiments.

One major problem is to find the optimal configuration for an MLP. This requires a lot of experimentation based on theory, and some brute force trials. According to the Universal Approximation theorem [1] , one hidden layer Multilayer Perceptrons are 'universal approximators'. The theorem thus states that simple neural networks can

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

represent a wide variety of interesting functions when given appropriate parameters. We therefore fix the activation function for a one hidden layer MLP to ReLu, and vary the number of neurons from 5 to 1000. As seen in 4.8, while changing the number of neurons and measuring f_1 accuracies respectively we see that there is no specific pattern to how the accuracy may vary with respect to the number of neurons. This leads us to follow a brute force method to find the optimal configuration for the number of neurons per hidden layer.

To estimate the optimal patch size for max-pooling we vary patch sizes and select a patch size of 9 and 10 to experiment with. We then compare the performance on patch size 9 and 10, as shown in 4.9. The hidden layer sizes are varied to many extents to find the optimal sizes for performance on the classification tasks. The optimal configurations for each experiment vary, as we can see in Table **??**.

Activation functions play a huge role in the performance of neural networks. We see in Section 3.5.2 that ReLu induces sparsity in the network which helps improve the performance on complex functions of data point distribution. We also later verify this for AD CU classification, where in Table 4.7

Using demographic features such as Age, Gender, APOE and FAQ improves the results significantly. Table 5.1 shows the effect of adding age and gender as features, and further adding APOE and FAQ scores as features. There is significant increase in accuracies. Age and gender might not seem as important criteria for classification, but as we know in machine learning, two irrelevant input features to a learning algorithm can together become relevant. By irrelevant features we mean that these features when used individually do not contribute to learning. We see that there is a slight increase in performance with the addition of age and gender attributes. This further motivated us to add gene information and FAQ scores. These additions proved fruitful and helped us achieve better results.

6.2 Limitations

Our work purely concentrates on the usage of Multilayer Feed Forward Neural Networks for classification. Our aim was to choose MLP as a classifier and find the best configuration to efficiently perform classification. FDG-PET image is 3 dimensional data, conforming to a voxel-wise structure of the brain. There are neural network configurations to address the spatial configurations of data, and learn spatial relations between features, such as (Parallel Multi-Dimensional Long Short Term Memory, Grid Long Short Term Memory). These can further be used on this dataset in an effort to improve classification performance.

This project includes experimentation purely on the baseline (first visit) data from ADNI2. Integration of data collected from later visits in time can further help analyse the dataset. Classification with time, (i.e. if a subject is diagnosed as CU (healthy), and in a later visit diagnosed with MCI(Early or Late)) successful prediction on converters and non-converters with neural networks is a technique yet to be explored.

Since our optimal configurations were achieved by a brute force method and an algorithm designed by us, there is a possibility of our results improving further. Further experimentation with neural network configurations may help achieve even better classification accuracies.

Chapter 7

CONCLUSION

This thesis develops a classification method to investigate the performance of FDG-PET as an effective biomarker for Alzheimer's clinical group classification. This involves dimensionality reduction using Probabilistic Principal Component Analysis on max-pooled data and mean-pooled data, followed by a Multilayer Feed Forward Neural Network which performs binary classification. Max pooled features result into better classification performance compared to results on mean pooled features. Additionally, experiments are done to investigate if the addition of important demographic features such as Functional Activities Questionnaire(FAQ), gene information helps improve performance. Classification results indicate that our designed classifiers achieve competitive results, and better with the additional of demographic features. This work has three main contributions, a coherent and efficient deep learning framework that well explores the possibility of FDG-PET for AD diagnosis. Secondly, we evaluated our work in a relatively large dataset and achieved competitive results. Thirdly, we exhibit the effective increase in classification performance with the addition of demographic variables (Age, Gender, APOE 1, APOE 2, and FAQ score) to our max-pooled (intensity) data. The use of additional demographic data that is unique to each subject has been investigated and proven to be a contributor to enhancing the performance of our neural net architecture. We compare our results to those based on FDG-PET analysis using machine learning techniques.

We hope our work will inspire more deep learning based work on FDG-PET analysis and advance preclinical AD research.

REFERENCES

Ben-Hur, A. and J. Weston, "*A users guide to support vector machines*", Data mining techniques for the life sciences pp. 223–239 (2010).

Bengio, Y. *et al.*, "*Learning deep architectures for AI*", Foundations and trends® in Machine Learning **2**, 1, 1–127 (2009).

Boureau, Y.-L., J. Ponce and Y. LeCun, "*A theoretical analysis of feature pooling in visual recognition*", in "Proceedings of the 27th international conference on machine learning (ICML-10)", pp. 111–118 (2010).

de Leon, M. J., *An atlas of Alzheimer's disease* (Parthenon Publishing Group, London, 1999).

Diederik P. Kingma, J. B., "*Adam: A Method for Stochastic Optimization.*", The International Conference on Learning Representations (ICLR) (2015).

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang and C.-J. Lin, "*LIBLINEAR: A library for large linear classification*", Journal of machine learning research **9**, Aug, 1871–1874 (2008).

Farrer, L. A., L. A. Cupples, J. L. Haines, B. Hyman, W. A. Kukull, R. Mayeux, R. H. Myers, M. A. Pericak-Vance, N. Risch and C. M. Van Duijn, "*Effects of age, sex, and ethnicity on the association between apolipoprotein E genotype and Alzheimer disease: a meta-analysis*", Jama **278**, 16, 1349–1356 (1997).

for Alzheimer's Disease Initiative, C. C. N. *et al.*, "*Tools for early identification, assessment, and treatment for people with Alzheimer's disease and dementia*", Alzheimer's Association and National Chronic Care Consortium (2003).

G. McKhann, M. F. R. K. D. P., D. Drachman and E. M. Stadlan, "*Clinical diagnosis of Alzheimers disease - report of the NINCDS-ADRDA work group under the auspices of Department of Health and Human Services task force on Alzheimers disease.*", Neurology **34**, 939–944, URL `http://www.neurology.org/content/34/7/939.abstract` (1984).

Glorot, X., A. Bordes and Y. Bengio, "*Deep Sparse Rectifier Neural Networks.*", in "Aistats", vol. 15, p. 275 (2011).

Gray, K. R., P. Aljabar, R. A. Heckemann, A. Hammers, D. Rueckert, A. D. N. Initiative *et al.*, "*Random forest-based similarity measures for multi-modal classification of Alzheimer's disease*", NeuroImage **65**, 167–175 (2013).

H.-W. Klafki, J. K., M. Staufenbiel and J. Wiltfang, "*Therapeutic Approaches to Alzheimer's disease.*", Brain **129**, 2840–2855, URL `http://dx.doi.org/10.1093/brain/awl280` (2006).

Hartwig, W., "*The unique contributions of the modified Logical Memory section of the Wechsler Memory Scale II towards localization and differential diagnoses.*", Archives of Clinical Neuropsychology **5**, S176–176 (1990).

Hazlett, H. C., H. Gu, B. C. Munsell, S. H. Kim, M. Styner, J. J. Wolff, J. T. Elison, M. R. Swanson, H. Zhu, K. N. Botteron *et al.*, "*Early brain development in infants at high risk for autism spectrum disorder*", Nature **542**, 7641, 348–351 (2017).

Heung-II Suk, S.-W. L. and D. Shen, "*Deep sparse multi-task learning for feature selection in Alzheimers disease diagnosis.*", The Alzheimers Disease Neuroimaging Initiative. **221** (2015).

I.A. Illan, J. R. D. S. G. M. M. L. F. S. R. C. M. G.-R. C. G. P., J.M. Gorriz, "*18F-FDG PET imaging analysis for computer aided Alzheimers diagnosis.*", The Alzheimer's Disease Neuroimaging Initiative. **181**, 903–916 (2011).

Katherine R. Gray, R. A. H. P. A. A. H., Robin Wolz and D. Rueckert, "*Multi-region analysis of longitudinal FDG-PET for the classification of Alzheimer's disease.*", The Alzheimer's Disease Neuroimaging Initiative. **60**, 221–229 (2012).

Krizhevsky, A., I. Sutskever and G. E. Hinton, "*Imagenet classification with deep convolutional neural networks*", in "Advances in neural information processing systems", pp. 1097–1105 (2012).

Li, F., L. Tran, K.-H. Thung, S. Ji, D. Shen and J. Li, "*A robust deep model for improved classification of AD/MCI patients*", IEEE journal of biomedical and health informatics **19**, 5, 1610–1616 (2015).

Liu, C.-C., T. Kanekiyo, H. Xu and G. Bu, "*Apolipoprotein E and Alzheimer disease: risk, mechanisms and therapy*", Nature Reviews Neurology **9**, 2, 106–118 (2013).

Liu, S., S. Liu, W. Cai, H. Che, S. Pujol, R. Kikinis, D. Feng, M. J. Fulham *et al.*, "*Multimodal neuroimaging feature learning for multiclass diagnosis of alzheimer's disease*", IEEE Transactions on Biomedical Engineering **62**, 4, 1132–1140 (2015).

Pascal Vincent, Y. B. P.-A. M., Hugo Larochelle, "*Extracting and composing robust features with denoising autoencoders.*", International Conference on Machine learning. **25**, 1096–1103 (2008).

Penny, W. D., K. J. Friston, J. T. Ashburner, S. J. Kiebel and T. E. Nichols, *Statistical parametric mapping: the analysis of functional brain images* (Academic press, 2011).

Reiman, E. M., R. J. Caselli, K. Chen, G. E. Alexander, D. Bandy and J. Frost, "*Declining brain activity in cognitively normal apolipoprotein E ϵ4 heterozygotes: a foundation for using positron emission tomography to efficiently test treatments to prevent Alzheimer's disease*", Proceedings of the National Academy of Sciences **98**, 6, 3334–3339 (2001).

Siqi Liu, W. C. H. C. S. P. R. K. M. F., Sidong Liu and D. Feng, "*High-level feature based PET image retrieval with deep learning architecture.*", The Journal of Nuclear Medicine **55** (2014).

Stone, M., "*Cross-validation and multinomial prediction*", Biometrika pp. 509–515 (1974).

Suk, H.-I. and D. Shen, "*Deep learning-based feature representation for AD/MCI classification*", in "International Conference on Medical Image Computing and Computer-Assisted Intervention", pp. 583–590 (Springer, 2013).

Sun, D., T. G. van Erp, P. M. Thompson, C. E. Bearden, M. Daley, L. Kushan, M. E. Hardt, K. H. Nuechterlein, A. W. Toga and T. D. Cannon, "*Elucidating a magnetic resonance imaging-based neuroanatomic biomarker for psychosis: classification analysis using probabilistic brain atlas and machine learning algorithms*", Biological psychiatry **66**, 11, 1055–1060 (2009).

Tipping, M. E. and C. Bishop., "*Probabilistic Principal Component Analysis.*", Journal of the Royal Statistical Society, Series B. **21**, 611–622 (1999).

Weiner, e. a., M.W., "*The Alzheimer's Disease Neuroimaging Initiative: a review of papers published since its inception.*", Alzheimers Dement **8**, S1–68, URL `https://www.ncbi.nlm.nih.gov/pubmed/22047634` (2012).

Zhang, D., Y. Wang, L. Zhou, H. Yuan, D. Shen, A. D. N. Initiative *et al.*, "*Multimodal classification of Alzheimer's disease and mild cognitive impairment*", Neuroimage **55**, 3, 856–867 (2011).