Fall 2009

Multistructure segmentation of multimodal brain images using artificial neural networks

Eun Young Kim
University of Iowa

MULTISTRUCTURE SEGMENTATION OF MULTIMODAL BRAIN IMAGES USING ARTIFICIAL NEURAL NETWORKS

by

Eun Young Kim

A thesis submitted in partial fulfillment of the
requirements for the Master of Science
degree in Biomedical Engineering in the
Graduate College of The
University of Iowa

December 2009

Thesis Supervisor: Assistant Professor Hans J. Johnson

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Eun Young Kim

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Biomedical Engineering at the
December 2009 graduation.

Thesis Committee: _____
Hans J Johnson, Thesis Supervisor

_____
Joseph M Reinhardt

_____
Vincent A Magnotta

_____
Michael A Mackey

_____
Mona K Garvin

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation

Anatomical Brain MR Image Research

The segmentation of distinct anatomical regions from medical images is a very common task used to quantify the morphologic qualities of disease. Currently hand tracing is regarded as a golden standard for brain segmentation, but it is labor intensive and suffers from inconsistency. Consequently, manual segmentation is often impractical for large-scale or longitudinal projects. The development of robust image segmentation methods is necessary to overcome the limitations of manual segmentation.

One important application of anatomical segmentation is to study neurological and psychiatric diseases of the human brain. Magnetic resonance (MR) imaging is one of the most important diagnostic tools used to visualize and understand the human brain. Segmented brain structures from MR imaging are commonly used to provide quantitative information about volume and shape. Analysis of the quantitative morphometric measures has identified important relationships between brain morphology and disease status for many mental illnesses such as Huntington's Disease, Schizophrenia, Alzheimer, Parkinson's, isolated clefts of the lip or palate, and many others [3, 6, 25, 31, 38].

The automated methods and procedures described in this work are motivated by the need to segment brain structures from multi-modal MR brain images collected by large-scale, multi-site, longitudinal studies. The implementation of the tools, however, is suitable for more general segmentation problems that could include other imaging modalities or segmentation targets.

Practical Difficulties

The complex and intertwined human brain structure make it difficult for spatial locations of image intensities alone to provide enough data for precise segmentation. The fact that tracing experts come to well-defined segmentation conclusions based on various external experience related knowledge, such as neuroanatomy, pathology, physiology and radiology [36] suggests the necessity of more information. In addition, the segmentation method for MR image analysis is complicated by the intensity non-uniformity caused by MR machines' intrinsic nature [16, 33]. These obstacles have to be addressed for satisfactory segmentation results.

The lack of ability for machines to have brain segmentation strategies compatible to humans is mainly due to lack of relative knowledge about clinical images. Most of image segmentation algorithms for clinical images depend solely on intensity values of a single image, but intensity information by itself is often not enough for an algorithm to give satisfactory differentiation of target structure from high folded and inter-connected neighbors. Tracing experts come to the well-defined tracing results based on information gathered from many sources and many subjects, such as neuroanatomy, pathology, physiology and radiology in order to arrive at a reasonable image interpretation for any selected scan[36].

1.2 Thesis Overview

In this research, an automated brain segmentation method was investigated compared to the expert manual tracing method. Chapter 1 provides the introduction and motivation of automated brain segmentation method. Chapter 2 discusses necessary background on machine learning. Chapter 3 discusses the implementation method used in this research. Chapter 4 provides detailed analysis on the results with reliability tests. Chapter 5 concludes with a summary of important results and future work.

CHAPTER 2
BACKGROUND

2.1 Desired Properties

Numerous criteria for assessing the performance of brain segmentation methods exist. To achieve reliable segmentation methodology in this research, three general criteria have to be addressed: robustness, accuracy and precision of the application. It is introduced in the later section for quantitative approaches using mathematical or statistical formula for testing these qualities. Here we introduce descriptive explanations for each criterion.

Robustness

The first requirement for reliable segmentation method is robustness against noise of an image or so-called noise-tolerance. The noise corruption of the clinical image is challenging problem not only for MR images but also for any other image modalities. At the same time, however, noise is unavoidable practical issues for any image modalities. So the intrinsically produced noise of the image should not cause failure of the segmentation method. The robust segmentation method has to have the ability to deal with this actual circumstance so that it can be used for practical image analysis. Another aspect of robustness is that the automated method must perform similarly across anatomical variability ranging from severely diseased to normal healthy controls.

Accuracy

The second requirement is accuracy of segmentation. The accuracy is addressed by measuring the volumetric differences, relative overlap, and similarity index of automated segmentation to manual segmentation. The automated segmentation method's accuracy is improved as volumetric differences approaches zero, and relative overlap

and similarity index approach one.

Consistency

The lastly emphasized criterion for reliability of our method is consistency. The consistency is to show the linear relationship between automated segmentation method and manual tracer for structure volumes that span disease and health population.

The fulfillment of these criteria of automated method is very important to provide he robust image segmentation for clinical researches. Each criteria is investigated throughout this research.

2.2 General Overview on Machine Learning

To understand a general scheme of machine learning for Artificial Neural Net (ANN) a conceptual understanding of Artificial Intelligence (AI) is needed. AI is a study of how to make computers do things, which at the moment people do better [29], especially for ability of information generalization and pattern recognition. Since the speed of numerical calculation of a machine exceeds that humans, researchers began investigating the possibility of an 'intelligent machine' and various area of machine learning has been developed. As the machine learning has several specialty areas, there are several perspectives of 'machine learning'. Here we introduce two methodologies based on different views of 'learning', one is called a *symbolic perspective* and the other is a *connectionist perspective* on machine learning. After a brief introduction of the two perspectives, the concept followed by ANN will be explained in more detail.

Symbolic Model Perspective on Machine Learning

Symbolic-oriented AI approach is based on a well-defined domain representation [22]. A learning process of machine can occur for different interest of domain one by

one separately like different human specialists in different area. Since the machine can store massive amount of data, which can be analogous to the human's memory, it focused on how information can be expressed in the form to be used effectively. In this scenario, the learning ability of machine will be proportional to the size of storage. The database as information storage for machines, it can be considered as learning while it is 'storing' information. The researchers were not just focus on put as many data as possible, but also more on how relational information, such as hierarchy, causality, direction and properties between any type of data, could be represented for machine learning.

Connectionist Perspective on Machine Learning

The connectionists approach is to simulate the functioning of the human brain biologically using computer [29]. Connectionists are more influenced by biologists and are more concentrated on how they can simulate the human brain, especially the way of neuron's signal exchanges and the human brains organizational structure. Even though brain reasoning mechanisms are not fully understood, the humans ability to learn by example exceeds that of machines. Machines, however, are better at rapid reproducible numerical computation. Connectionists recognized those facts and invented brain-like machines to mimic the mechanical functioning of the human brain. Constructing a brain-like machine mainly consists of two parts: (1) nodes, which is analogous to the biological dendrites as a signal receiver and (2) connections, which are analogous to the biological axons to deliver signals between nodes. J.A Feldman et al[9] described the concept of nodes as individual computing units intricately connected to each other to allow machine learning. The notion of a more complete neural network was started by McCulloch and Pitts [19]. A great amount of research interest in machine learning existed from the mid 1940's to the early 1970's, but then came to virtual halt in the 1970s due to the lack of computational resources [29] to realize the proposed algorithms. Now that rich computing resources have again become

available, research activity in connectionist model have become popular again.

2.3 Segmentation Method with Artificial Neural Network

Different artificial neural networks have been employed for image segmentation. The method using artificial neural network for segmentation were introduced in [2]. Moghaddam improved the ANN segmentation method by adapting Geometric Moment Invariants (GMIs) to the training stage [23]. One of the popular segmentation method utilizing ANN is a Self-Organizing Map (SOM) [1, 28, 17] Another popular approach of ANN is Probabilistic Neural Network (PNN)[24, 14, 37, 4, 34, 8]. The method uses PNN to form probability density function for estimating of each region of interest. These methods allows the segmentation module to identify unique distributions other than standard distribution for region of interests. To overcome the limitations of the semi or fully automated segmentation method by using ANN, hybrid methods have also been proposed. Reddick et al. suggested the combination of SOM and ANN for brain image segmentation[27], and Middleton et al proposed a different approach combining ANN and active contour model [21] . A hybrid method of wavelet with ANN has also been introduced [18].

2.4 More Background on Artificial Neural Network

The ANN follow the connectionist perspectives and represent a simplified version of the human brain. The ANN cannot think like human does, but it is good at adaptive learning which can be used for pattern recognition or its related work. Here we introduce the perceptron, the basic building unit of ANN, and basic learning mechanisms, which will develop further in later chapter for more proper use.

2.4.1 Knowledge Representation: Perceptron

As briefly introduced previously, the fundamental unit of an ANN is the neuron-like construct called a 'perceptron'. Frank Rosenblatt proposed the concept of perceptron in 1958 as an artificial neuron. A perceptron's function is analogous to what the neuron does in human brain. As a biological neuron receives signals from dendrites and propagates those signals to the axon to deliver to the next level of perceptrons. In an ANN, the perceptron aggregates and weights the signals from several inputs and propagates a single combined signal to the output end.

In addition, a certain number of perceptrons are connected to each other like numerous neurons do in the nerve system. Other analogous feature between artificial and biological neurons is the mechanism of signal summation. In human nervous system, inputs from more than one synapse can result in summation of the synaptic potentials, which may then trigger an action potential on postsynaptic cells. Input synapses can be *Excitatory*, which encourages firing or *Inhibitory*, which counter acts on signal summation for postsynaptic cell activation[35]. The perceptrons in an ANN produce their output status by adding up connected input nodes in the similar way. A graphical interpretation is shown in figure 2.1

The last characteristic of perceptron is the activation function, which is biologically analogous to the action potential of a neuron. The biological neuron decides whether it is going to fire based on the strength of signal. If the strength of signal is below the action potential, it does not propagate its signal to the next neuron, otherwise it propagate to the next one. The activation function closely resembles the action potential mechanism. It takes summed all the input values and decides the output status.

Figure 2.1: A perceptron (left), computing unit of ANN, can be analogous to a biological neuron (right). Like a biological neuron exchange its signal in the form of potential, a perceptron does with numerical values.

2.4.2 Learning Mechanism: Backpropagation

First a basic procedure for ANN learning mechanism is explained, including how perceptrons are utilized in the ANN architecture model. A more advanced algorithm, which is used for our implementation, for learning is developed from this general approach and introduced in the method section.

2.4.2.1 General Learning Mechanism for ANN

The adjustment process for an ANN's nodes weights and connection process so that desired outcome can be acquired for all classes of inputs is the learning or training of the ANN. In nervous system, there are millions of neurons and usually several steps are involved in one decision. Each neuron decides its firing status based on its own input stimulus independently, and the perception is configured similarly. As figure 2.2 shows, the connection of perceptrons forms the ANN architecture that consists of one input layer, one output layer and zero or more hidden layers between the input and output layer. Each layer can have any number of nodes and connections to neighbored layers' nodes. The simplest structure is a two-layered ANN architecture having input and output layer only (figure 2.2) and the architecture with one or more than one hidden layer is called 'multi-layered architecture'.

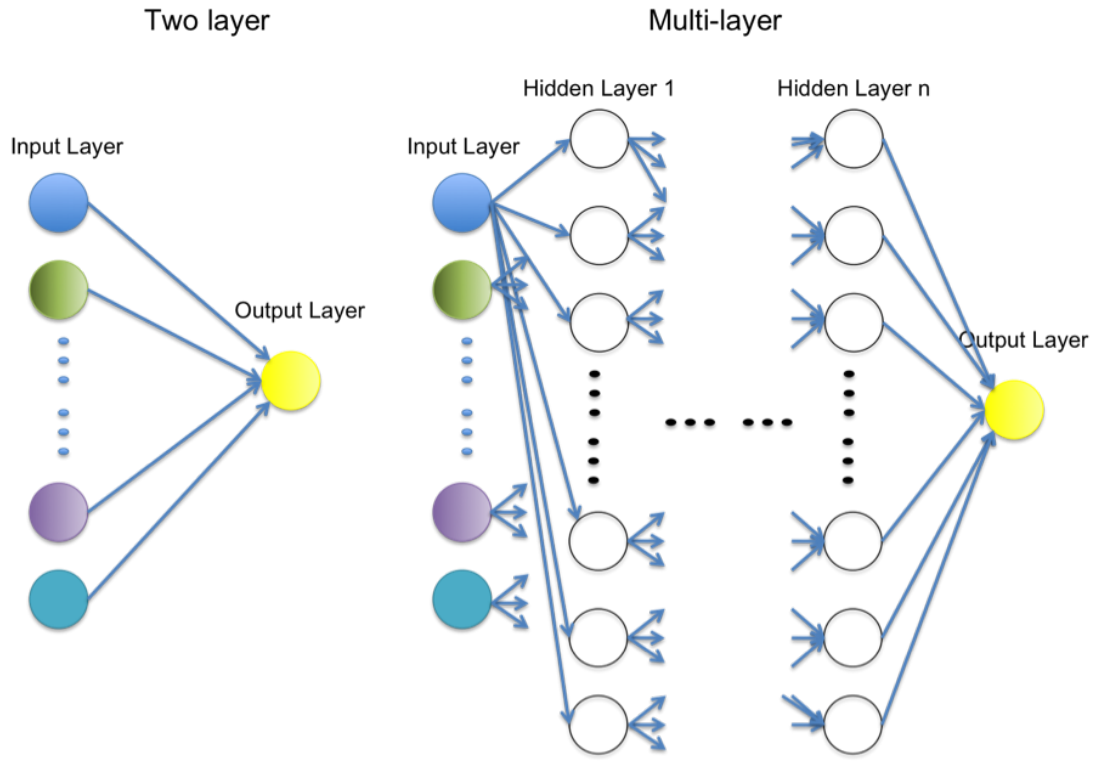


Figure 2.2: A two-layered (left) and a multi-layered (right) ANN architecture: A two-layered ANN architecture is the simplest one and multi-layered ANN architecture has one or more than one hidden layer in-between input and output layer.

Abilities of Single vs. Multi-layered Architecture in Solving Problem

When we deal with linearly separable problems, the perceptron convergence theorem, described by Rosenblatt [1963], guarantees that the perceptron will find a correct solution with large enough number of training. Minsky and Papert [1969] stated that, however, even if the convergence theorem holds for linearly separable data classification problem, most of real world data does not provide that condition. The counter example is XOR problem, which is not linearly separable problem since we cannot draw a line between different outputs given input(see figure 2.3). Here the multilayer network plays its role, separation of non-linear cases. Some years later they found that the multi-layered neural net generally solves any given problem

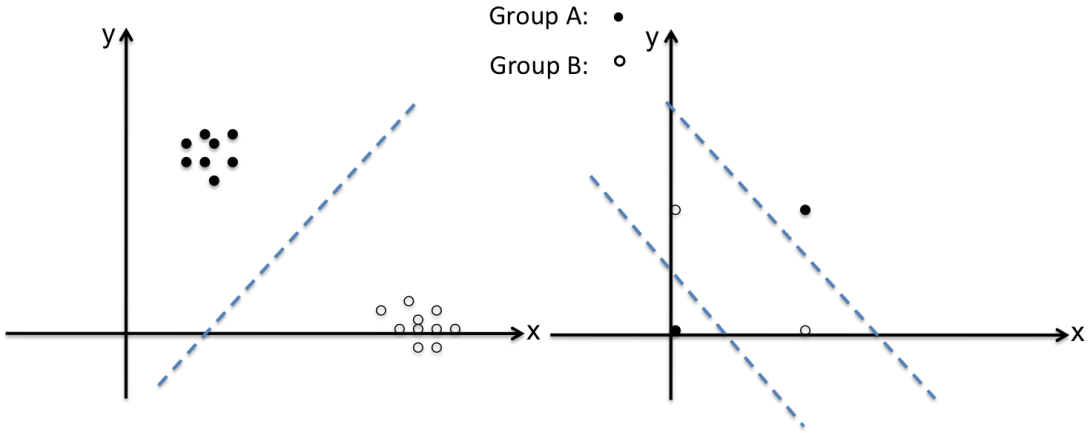


Figure 2.3: A solvable and unsolvable graphical example for the simplest ANN: Left graph shows that the case of linearly separable problem, which can be easily solved with the simplest ANN, and right represents non-linearly separable problem, which needs at least two lines for separation and thus cannot be solved with same architecture.

even though the perceptron convergence theorem loses its power, Kurt Hornik [13] and Cybenko [7] independently proved that multi-layered neural net with one hidden layer can approximate any given mapping function with any desired accuracy. [29] Hornik stated that any lack of success in applications must arise from insufficient numbers of hidden units or the lack of a deterministic relationship between input and target.

Supervised Learning

The learning process during neural network training involves modifying connections between nodes to generate the desired output. Even though it is not guaranteed to converge for multilayered neural networks, several algorithms produce heuristically best solution. Hinton [12] divided connectionist learning procedure into three broad classes: 1) Supervised procedures, 2) reinforcement procedures and 3) unsupervised procedures. This work has focused on supervised learning procedures that used training data set in the form of input vector coupled with desired output. The general notion of supervised learning is same to human learning with teacher. When children

see cat or dog at the first time, they might not distinguish one from the other. After some time, if teacher tell them right or wrong whenever they saw cat or dog and guessing about it, they would learn which one is cat and which one is dog in their own way. One might distinguish them by size, others may by shape of faces. This is how ANN learns to classify by example. From initial randomized guess about output for given input, the ANN will learn by comparing its guessed output and given desired output.

2.4.2.2 Backpropagation algorithm

The backpropagation algorithm is one of the most popular neural network learning algorithms; it uses gradient descent to find the correct combination of node's weights to minimize errors in a known training set. It is necessary to describe the components and topological network features before describing the backpropogation algorithm.

Activation Function

The activation function, analogous to the action potential of the biological neuron, has several forms for different purposes. For backpropagation a continuously smooth activation function is desirable. The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights that minimizes the error function is considered to be a solution of the learning algorithm [10]. A differentiable activation function is required so that computation of gradient descents at every iteration can be used to find a minimum of the error function. One of the most popular activation functions is the sigmoid function.

$$sigmoid(t) = \frac{1}{1 + e^{-tx}} \tag{2.1}$$

The slope of sigmoid function is varied by the constant t, the larger value for t, the sharper slope. As t goes infinity, it asymptomatically converges to the step function

[10].

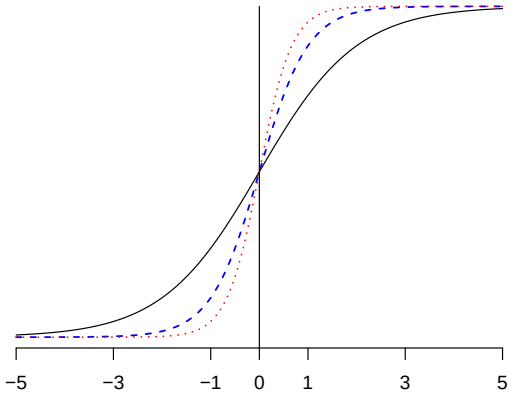


Figure 2.4: Sigmoid functions with different value for t: The Black is $t = 1$, the blue dashed line is $t = 2$ and the red dotted line is $t = 3$. As t value increases, the slope gets shaper. If the t goes infinity, the sigmoid function asymptotically converges to the step function.

Fully Connected Network

Fully connected network means that the all the existing nodes are connected only between neighbored layers. It is general to use fully connected network compare to partially connected one since the unimportant or negligible connection can be reflected with a very small valued weighted connection. The example for the fully connected network can be seen on the figure 2.5.

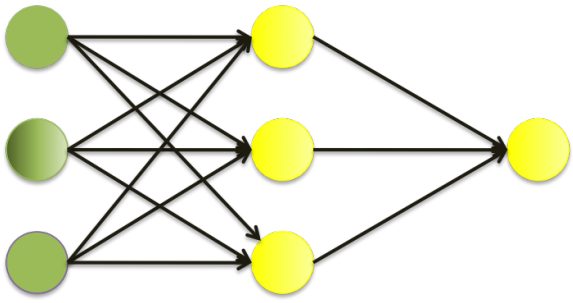


Figure 2.5: A Fully connected network: All neighboring nodes are fully connected each other

Feed Forward Network

The feed forward network has only one direction of data flow: from input layer to the output layer. The nodes and weights only have effect in the direction of output layer, so there is no loops or cyclic connections in the architecture. In the figure 2.6 feed forwarded data appears as right directional arrow. This means that the decision making process can only depend on previous nodes, which are located in the input nodes side of current nodes.

Backpropagation Algorithm

Back propagation algorithm is the most commonly used method in training the multi-layer ANN. As previously mentioned, the learning process of ANN is the process of adjusting weights for hidden nodes that the cumulative error of a given training set is minimized. The back propagation algorithm starts from very small random weights assigned to each node as an initial guess. An ANN is known to be good at finding patterns, the ANN finds the matching patterns of input vector to output vector. Given a training vector, which is ordered pair of input(x_i) and output(t_i),(x_i, t_i), the ANN guesses the answer at random(g_i), then adjust weights by looking at the error function, f_e:

$$f_e = \frac{1}{2} \sum || \ g_i - ti \ ||$$

The adjusting weights for next iteration goes in the direction that minimizes for error function. As mentioned earlier, the gradient is used to compute local minimization. Calculation of gradient descent needs to be updated at each iteration time and the equation follows:

$$\Delta w_i = -\gamma \frac{\delta E}{\delta w_i}$$

, where w_i is i^{th}, $\bigtriangledown E = (\frac{\delta E}{\delta w_1}, \frac{\delta E}{\delta w_2}, \ldots, \frac{\delta E}{\delta w_n})$, which is the gradient descent for error function, and γ is the learning rate, which specified by initial parameter. The repetition of the 'guessing output' and 'adjusting weights' is called 'training' for ANN

model and the data used for training is called 'training data'. Detailed algorithm is on the table 2.1.

Table 2.1: General Backpropagation Algorithm for a fully connected, feed foward network, adapted from [10] and modified.

Feed forward :	The input x is fed into the network. Calculate each connected nodes from input layer to the output layer.
Backpropagation:	After evaluating the error function, update each nodes' weights by the rule provided above.

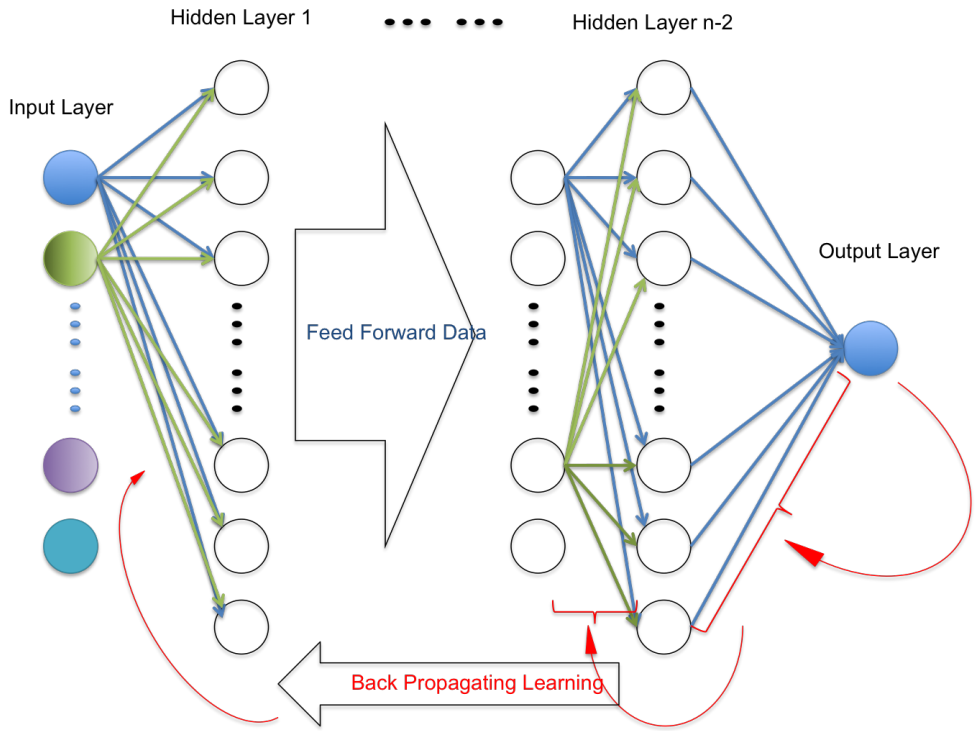


Figure 2.6: A Feed-forward network with backpropagation learning: input data flows the direction from left to right side ('Feed Forward Data' arrow) and learning process goes on left-to-right direction to adjust nodes' weights to improve next trial ('Back Propagating Learning' arrow).

CHAPTER 3
METHOD

The process of ANN segmentatiodn for brain MR consists of three main steps:(1) Preparation for the training data set, (2) Constructing ANN architecture and training and (3) test for validation and verification of the application. Iterations of these steps are essential to optimize the performance of application.

3.1 Training Data Preparation

Training data preparation is the process that takes all the subjects' multi-modal images into one well representative vector form to be fed onto the established ANN architecture. The topology of the ANN architecture will be discussed later. The training vector is an ordered pair of input (x_i) and output (t_i) vector: (x_i, t_i). Each training vector pair has a corresponding spatial voxel location based on atlas space. Before extracting training vectors, we have to decide which image type will be used for training. This section describes selection of subjects for our experiments, different image types used in the training.

Training Set Selection

24 sets of multi-modal, T1 and T2, MR images were collected in our research interest domain. Each individual was chosen to have different characteristics in their volume, size, and shape so that our training set thoroughly covers normal anatomical variation, and was not biased towards one particular configuration. Out of 24 data sets, 16 data sets were chosen randomly for training and 8 data sets were used for testing purposes.

Brain Atlas

The brain atlas from *Montreal Brain Web* [5] adopted as our brain atlas. The brain atlas is used to offer a common spacial location to register in and register out the images in of our training data set. All subjects were transformed into the brain atlas space to compute the probability map. The opposite registration, i.e. the brain atlas registered on the subject image, is used to transform the probability map back into subject space.

Priors

Priors are spatial probabilistic distribution maps for different structures. The priors allow us to see the likelihood of the structure's given voxel location. The higher value the voxel has for the structure prior, the more probable it is likely to be the structure. These priors generated using all of 16 training set for every structure we are interested in: brain, left and right caudate, accumben, putamen, globus and thalamus.

Feature-Enhanced Images

Multi-modal images and two feature-enhanced images enhance the robustness of the ANN segmentation to image intensity noise. They also provide complimentary information that does not exist in any single modality. Both feature-enhanced images were created from acquired multi-modal data. The first feature-enhanced image is a tissue-classified image to enhance intensity uniformity across subjects and the other is a mean of gradient magnitudes image with emphasis on structure boundaries. Detailed explanation and a mathematical formula are following.

Tissue Classified Image

A tissue-classified image generated by discriminant analysis [11] is employed to provide intensity uniformity over different scans and subjects. A major problem for

automated MR image segmentation is the corruption with a smoothly varying intensity inhomogeneity or bias field [15, 33]. The image suffered from inconsistent intensities across subjects and tissue types can result in a general failure of segmentation. The method uses discriminant analysis on the multi-modal, T1- and T2-weighted, images to create soft- or continuous-tissue classified images. A soft-tissue classified image provides a spectrum of tissue types for white matter (WM), grey matter (GM), and cerebrospinal fluid (CFS). Pure WM, GM, and CSF have values of 250,130 and 10, respectively and values between these represent composition of more than one tissue type. The soft-tissue classified image as a normalized image for variability of different scans allows our method to have global idea about general intensity distributions across different scans.

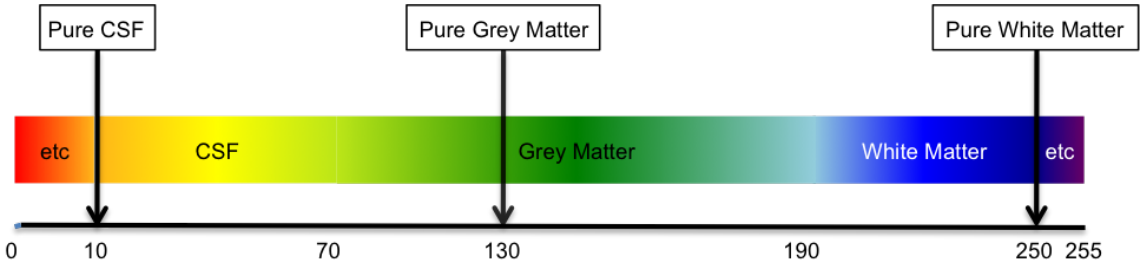


Figure 3.1: A spectrum reference for a soft-tissue classified image. A pure grey matter (GM), white matter (WM), and cerebrospinal fluid (CSF) have values of 250, 130 and 10 respectively. Values in-between pure tissue types are considered to be mixture of more than one tissue types [11].

Mean of Gradient Magnitudes Image

The average of gradient magnitudes of *T1*- and *T2*-weighted images is created for highlighting boundaries of structures. A *T1*-weighted image shows relatively distinct GM and WM boundaries while a *T2*-weighted image provides more on CSF boundaries. In addition, the gradient magnitude along intensity descents direction traditionally gives contour information of image. By putting all informa-

tion together, the average of gradient magnitudes for *T1*- and *T2*-weighted images allows our method to identify possible boundaries of the brain structures better. The computation formula follows the below equation.

$$\| \bigtriangledown f(x,y,z)\| = \|(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})\|,$$

where $f(x,y,z)$ is a scalar function.

$$Grad\_Avg(T1,T2) = \frac{\| \bigtriangledown f_{T1}(x,y,z)\| + \| \bigtriangledown f_{T2}(x,y,z)\|}{2},$$

where f_{T1} and f_{T2} represent the gradient magnitude for *T1*- and *T2* images.

3.2 Features for Input Vector

From the above images, multi-modal MR images and feature enhanced images, features for input vector for each voxel created. Features are selected to represent well enough for each structure's characteristics based on spacial information.

Modified Spherical Coordinate to Reflect Symmetry of Brain

The modified version of spherical coordinates for a given voxel is added to the input vector as a sub-element as appeared on figure 3.2. The modification is due to take the symmetry of the brain shape along to the AC-PC line into account in the learning process. The traditional definition of spherical coordinate system is following:

$$rho = \sqrt{(x^2 + y^2 + z^2)}$$
$$phi = arccos(\frac{z}{r})$$
$$theta = arctan(\frac{y}{x})$$

The modified spherical coordinate to factor in the symmetry is following:

$$rho = \sqrt{(x^2 + y^2 + z^2)}$$
$$phi = arctan(\frac{|x|}{y})$$

$$theta = arctan(\frac{|z|}{y}) \tag{3.1}$$

This modification allows the ANN application to have general tendency of left and

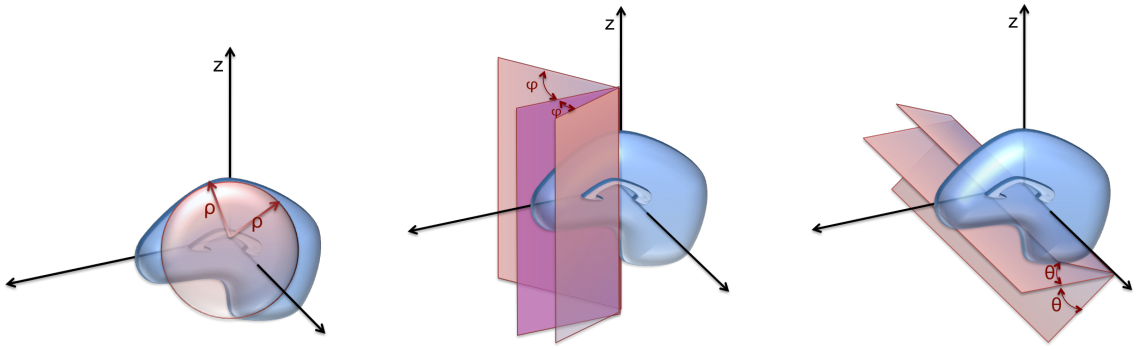


Figure 3.2: The modified definition of spatial voxel location: The demonstration figure for modified version of rho (left), phi (middle), and theta (right) for enhancement of training. The definition of rho follows original definition but definitions for phi and theta modified to address symmetry of brain structures

right structures.

Neighborhood based on Gradient Descent

Gradient descents are directional vector along probabilities, so that it can have directional consistence from inner side of the structure to out [26]. Intensity values along the gradient descents were used as a sub vector for input. The gradient descent can be thought as a pseudo-normal vector of expected surface for the structure based on the priors. Taking intensity values along the pseudo-normal vector from a few voxel inside to outside has a distinct advantage over using neighbor intensity values based on a rectilinear-based coordinate system. The directional consistency of input vector can be achieved by using gradient descents while extracting neighboring intensities, and a considerable amount of data can also be reduced by using only two instead of six neighbored values.

Figure 3.3: A neighborhood system comparison between a traditional (left) and an adapted (right) system: A traditional neighboring system considers 8 directions regardless of structures position while the adapted system here considers only two along the gradient descents direction of structure priors.

3.3 Training Vector Construction

A set of ordered pair vector, training vector, is constructed for image types and features described above. Training vector involves four steps.

1. Register the atlas to the subject image
2. Map all the ROI priors to the subject image space using the same deformation field created in step 1.
3. Extract features for voxel with prior values between zero and one.
4. Assign output vector by given location.

Input Vector

The input vector set is generated using all of above prepared images. The element of input vector has 16 elements, which can be divided by three sub-vectors.

$$I = [Sp, Ps, Gr]$$

Each sub-vector represents corresponding voxels characteristic given location.

- $Sp = [\rho, \phi, \theta]$

  Modified spherical coordinate system. Definition appeared on equation 3.1.

- $Ps = [s_1, s_2, \ldots, s_n]$

  Set of Boolean variables(s_i) for representing candidate structures based on spatial location of voxel. If the prior has a value bigger than zero for a given location, the corresponding variable is set to 'true'.

- Gr[I][L]

  Gr represent intensity values along the prior's gradient descent for corresponding Image Type(I). Image Type can be T1, T2, Tissue Classified Image(CL), or Average Gradient Magnitude(GrMg), and Location(L) can have any integer value between given neighboring range. Most common range is 1, which means that one-voxel distance neighbors are considered along the gradient descent direction. $[-1]$ indicates one voxel prior to the descents opposite direction, [0] is the voxel's own intensity itself, and $[+1]$ means one voxel after the current voxel to the descent's direction.

Output Vector

Each element of the output vector indicates whether the voxel under consideration voxel belongs one of the structure of interest. One voxel should belong to the one structure and cannot belong to more than one structure. Anatomical exclusiveness between different structures of brain is enforced. So if the voxel manually classified as the structure, which the first element of the vector indicates, only first element of the vector is 1 and others are 0. The number of structures being included in the model determines the size of the vector.

Pairing Input and Output Vectors

The training vector is comprised of two vectors: input and output. The pair is created for every voxel for each structure of interest that has its prior probability value between 0 and 1. If a voxel has a value of 0 for prior, the training vector is not generated because there is no possibility the voxel to be a part of structure. In

the same way, a voxel with a prior probability of 1 is disregarded and considered as a part of structure regardless of other features.

3.4 ANN Architecture: Multi-layer Neural Network

ANN architecture in our application includes three layers: Input, output and one hidden layer.

- Input layer: The input layer consists of input vector plus one bias node.
- Hidden Layer: Various number of Hidden nodes were tested and applied on each structure to find optimal ANN architecture for our model.
- Output Layer: The number of desired output, segmentation, is the number of nodes in the output layer.

3.5 Enhanced Back Propagation algorithm: RPROP algorithm

Our network utilizes the RPROP algorithm, which developed by Martin et al [30] for training. The algorithm is modified in a way to overcome a a drawback of traditional back-propagation methods: a local adaptation of the weight-updates determined by the behavior of the error-function. The RPROP algorithm, which stands for 'resilient propagation', performs a direct adaptation of the weight step based on local gradient information. Without given learning rate or momentum, this performs its learning by adapting based on previous error status. A key point on its strength is that it is independent from the magnitude of error only depends on sign. Individual update-value, Δ, introduced to determine the size of the weight-update:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \cdot \Delta_{ij}^{(t-1)} & , \quad \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)} & , \quad \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\ \eta^- \cdot \Delta_{ij}^{(t-1)} & , \quad \text{else} \end{cases} , \tag{3.2}$$

where $0 < \eta^- < 1 < \eta^+$. Once individual update-value, Δ, obtained, the weight-update follows:

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)} & , \text{ if } \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0 & , \text{ else} \end{cases} \tag{3.3}$$

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{3.4}$$

Overall algorithm for RPROP appeared on the algorithm 1.

Algorithm 1 RPROP Algorithm

for all weights and biases **do**
  if $\frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0$ **then**
    $\Delta_{ij}^{(t)} = minimum(\Delta_{ij}^{(t-1)} \cdot \eta^+, \Delta_{max})$
    $\Delta w_{ij}^{(t)} = -sign((\frac{\partial E^{(t)}}{\partial w_{ij}}^{(t)}) \cdot \Delta_{ij}^{(t)})$
    $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
  else if $\frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0$ **then**
    $\Delta_{ij}^{(t)} = maximum(\Delta_{ij}^{(t-1)} \cdot \eta^-, \Delta_{min})$
    $w_{ij}^{(t+1)} = w_{ij}^{(t)} - \Delta w_{ij}^{(t)}$
    $\frac{\partial E^{(t)}}{\partial w_{ij}} = 0$
  else if $\frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} = 0$ **then**
    $\Delta w_{ij}^{(t)} = -sign((\frac{\partial E^{(t)}}{\partial w_{ij}}^{(t)}) \cdot \Delta_{ij}^{(t)})$
    $w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}$
  end if
end for

3.6 Generate ANN Model

Training of ANN proceeds uses an RPROP algorithm fed with a training vector, input-output pair vector. Several number of hidden nodes were trained to find the optimal ANN architecture for segmentation.

Generation of an ANN model is a learning process for a pattern of input-output vector. The pair of input and output vector, a training vector, goes into the input layer of the predefined ANN model to generate an ANN model as described on the

background section. Traditional way of training ANN model starts with creating an over-fitted model. The over-fitted model can be identified by error function, which ideally looks like the solid line of the figure 3.4, which converges asymptomatically as training progresses. The over-fitted ANN model then are investigated for optimally trained point with test data vector. The test data vector has the identical format to the training vector but is created from different data set. Traditionally suggested shape of performance error function is appeared as a dotted line on the figure 3.4. The performance error reaches a point; at which the error increases rapidly while train error decreases continuously. The turning point of performance function is regarded as an optimally trained time point for the ANN model. After the time point, the ANN model stop generalizing to memorize a given input-output pattern of training vector. Because of its failure of generalization of the pattern time after the point, the error rapidly increases with testing vectors.

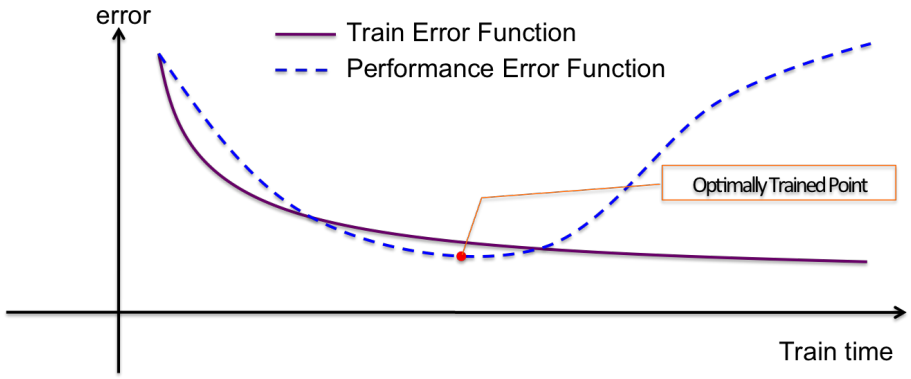


Figure 3.4: Error function for train and performance error: Ideally, while the train error function converges to the certain point as train time progresses, the performance function reaches a point, at which the error increases rapidly. This minimum critical point of performance function is regarded as the a optimally trained point.

3.7 Apply completely trained model

After constructing an optimized ANN model, the ANN model is applied to the test set for validation and verification. An ANN model that satisfies the criteria of

validation and verification, will be used for large-scale data analysis later. Validation and verification are described in the 3.8. This section explains the application process. The ANN application starts with registration of an atlas image onto the subjects' images to address anatomical variability. The transformations obtained from the registrations are to register each spatial probability maps on top of subjects' spaces. As previously explained, the probability maps are for extracting voxels of input vectors. The extracted set of input vectors for the subject now goes into the input layer of the constructed ANN model to compute output by the propagating mechanism called 'feed-forward'. The application process is completed by thresholding the ANN output at a certain level to generate binary images for regions of interest. A simple threshold method can be used for creating reasonable segmentation, which is described as a part of result section. However a special threshold method is also developed to satisfy the need for precise segmentation of adjacent structures, which is the case for sub-cortical brain structures. Remained section of this chapter describes about the threshold method for neighboring structures.

Threshold for neighboring structures

Even though the ANN model optimized to produce segmentation that matches the results of manual tracing, it is still problematic to precisely draw boundaries between adjacent structures. To address this issue, the threshold method for neighboring structures is developed. An ANN output right before a threshold process has continuous real values between zero and one since it is outcome of a sigmoid function (figure 2.4). These fuzzy segmentations could be interpreted as probability maps for regions of interest based on spatial location. Since brain sub-cortical structures are not only mutually exclusive with each other, but also defined without any holes between them, those fuzzy outputs should take these constraints into account to produce the final results. By bringing a little of mathematical adjustment before threshold process, these biological brain definition can be achieved from auto segmentation method as

well. Firstly, calculate the summation of ANN outputs for all regions of interests for a subject.

$$T = \sum_{r} A_r$$

Then, threshold the summed ANN output at certain value.

$$\bar{T} = threshold(T)$$

Finally, if a voxel survives the thresholding, then find which region of interest has a maximum value of ANN output and segment the corresponding voxel as belonging to the region with maximum ANN output.

$$S_r = \bar{T} \cdot A_r$$

, where $r = index\ of max(A_1, A_2, \ldots, A_n)$ for $r = 1 \ldots n$, $n = number\ of\ regions\ of\ interest$, and S_r is for segmentation for region r.

3.8 Validation and Verification Method

Binary result images from the ANN model are validated and verified in volumetrically through a set of measuresments: Mean and variance, Relative Overlap, Similarity Index, Intraclass Correlations and Pearson's correlation. Each measurement can have different interpretation as regards the relation between manual traced one and ANN results.

Mean and Variance

The volumetric comparison for sample means and variances conducted at the very first stage of validation and verification process. These simple measures allow us to see whether our automated method is worthy of further investigation. If the difference in volume between automatic and manual segmentation exceeds 10%, the ANN model will failed to learn the structures. In the same context, if the automated segmentations variance is larger than four times that of the manuals, this model also is disregarded for further investigations.

Relative Overlap(RO) and Similarity Index(SI)

Both relative overlap and similarity index are used to compare the accuracy of the automated segmentation to the manual segmentation. The Relative Overlap metric is commonly used to determine correspondence between two different images. The Definition of RO between structure A and B follows equation 3.5

$$RO(A, B) = \frac{(A \cap B)}{(A \cup B)} \tag{3.5}$$

where A and B are volume measures for structure A and B respectively. Similarity index is also used commonly in comparing.

$$SI(A, B) = \frac{2(A \cap B)}{(A + B)} \tag{3.6}$$

These two metrics may give different number but the tendency should be same. If two structures have more agreement between their shapes, the measures should both have higher scores, and vice versa.

Pearson's Correlation Coefficient (r)

Pearson's correlation coefficient, simply r, is computed to see correlation between two groups.

$$r = \frac{\sum(A_i - \bar{A})(B_i - \bar{B})}{(n-1)s_A s_B} \tag{3.7}$$

Intraclass Correlation Coefficient(ICC)

Intraclass correlation coefficients (ICC), as a measure of the reliability between two different judgments, is employed to interpret our results in segmentation. Out of numerous definition of ICC, we used the ICC (C,1), which is consistency and ICC (A,1) that is absolute agreement followed the chart by McGRAW and WONG [20]. From the chart (figure 3.5), we can conclude that a 'consistency' and 'absolute agreement' are our most interest for segmentation value analysis. ICC (C,1) and ICC (A,1) can be corresponding to the definition of ICC (3,1), ICC (2,1) of Shrout and Fleissis [32], respectively. Shrout and Fleissis says that if two judges are used

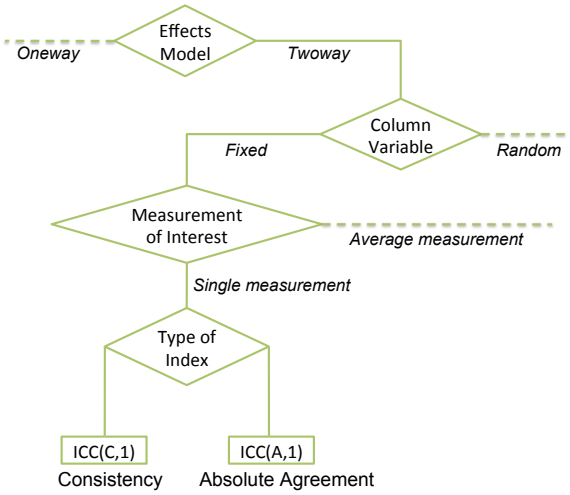


Figure 3.5: A portion of flow chart for selecting ICC values: A ICC consistency and absolute agreement are selected based on this chart suggested by McGRAW and WONG [20].

to rate the same number of targets, the consistency of the two rating is measured by $ICC(3,1)$, treating the judges as fixed effects. To measure the agreement of these judges, $ICC(2,1)$ is used, and the judges are considered random effects; in this instance the question being asked is whether the judges are interchangeable [32]. McGRAW and Wong also suggested that the $ICC(A,1)$ is to measure the absolute agreement between two groups and $ICC(C,1)$ is interested in correlations. Here we call the ICC-agreement for $ICC(A,1)$ or $ICC(2,1)$, and ICC-consistency for $ICC(C,1)$ or so called $ICC(3,1)$. As so McGRAW and Wong mentioned, when the mean differences between two different judges are small, there will be little difference between ICC-agreement and ICC-consistency. As mean differences increase, however, ICC-agreement will decrease while ICC-consistency remains [20].

CHAPTER 4
RESULTS

Results are presented in three sections. Section 4.1 provides ANN segmentation results from individually trained model and finding optimized models and section 4.2 analyzed the result from threshold method for neighboring structures. Lastly, the section 4.3 presents the multi-structure segmentation results from the simultaneously trained ANN model for all the sub-cortical structures.

4.1 ANN Segmentation Result for Individually Trained Model

Segmentation results from individually trained ANN models for sub-cortical structures are presented here. Models are trained for left and right side structures simultaneously. A total of six sub-cortical regions of interest were trained on 16 image training data set and tested with an 8 image testing data set.

Selected ANN model features for each experiment are following.

- Multi-Layered ANN with one Hidden Layer

  Number of Input nodes varied by different size along the gradient descent.

- A number of neighbors from the current voxel along the gradient descents in and out. Two different size were tried and denoted by $Grad = n$, where n is 1 or 2.

- After 4 different number of Hidden nodes were tried for $Grad = 1$, based on the result, two different number of Hidden nodes were investigated for $Grad = 2$.

  Number of hidden nodes denoted by $HN = h$, where $h = 50, 60, 70$ and 80.

A total of six ANN models were investigated to see how ANN behaves with different model constructions. Generally, increasing the number of neighbors along the descent does not result in improvement, and the model with $HN = 60$ and

$Grad = 1$ have the most high perfromance for the most of our regions of interest.

4.1.1 Convergence Plot

The convergence plot contains two error functions at each time point of training iteration. One is the training error function, which is the mean square error calculated during training, the other is the performance error function, which uses the identical measure with test vector set. The training error function is to see whether the model converges at certain status, and the performance error functions is to identify the optimally trained point.

To make sure the training model over-fit the data set, so that the optimal training point can be investigated later, a sufficient number of iterations were specified fo each training run. The plotted points are acquired every 100 epoch iterations and ran through 400 iterations.

As we can see in the convergence plot, for each structures with various models in the figure 4.1, they error function value decreases and the curve flattens as iteration increase. This means that the model trained well toward the proper direction to minimize the result of error function. The optimally trained time point, which represented as dotted vertical lines, was all appeared after convergence occurred. Convergence plot for the entire ANN model appeared on Appendix A.

4.1.2 Selection of Optimal Threshold

Based on the convergence plot the number of iterations is determined for the optimally trained model. Then the optimal threshold value was investigated, given that the neural net's output has values between zero and one. Two measures were used in determination of the optimal threshold value: Relative overlaps (RO) and intraclass correlation (ICC) versus threshold value. The figure 4.2,which is graph for the ANN model with $HN = 60$ and $Grad = 1$, shows that the most of maximum value for ROs were located around threshold value 0.4. At a threshold of 0.4 it is also

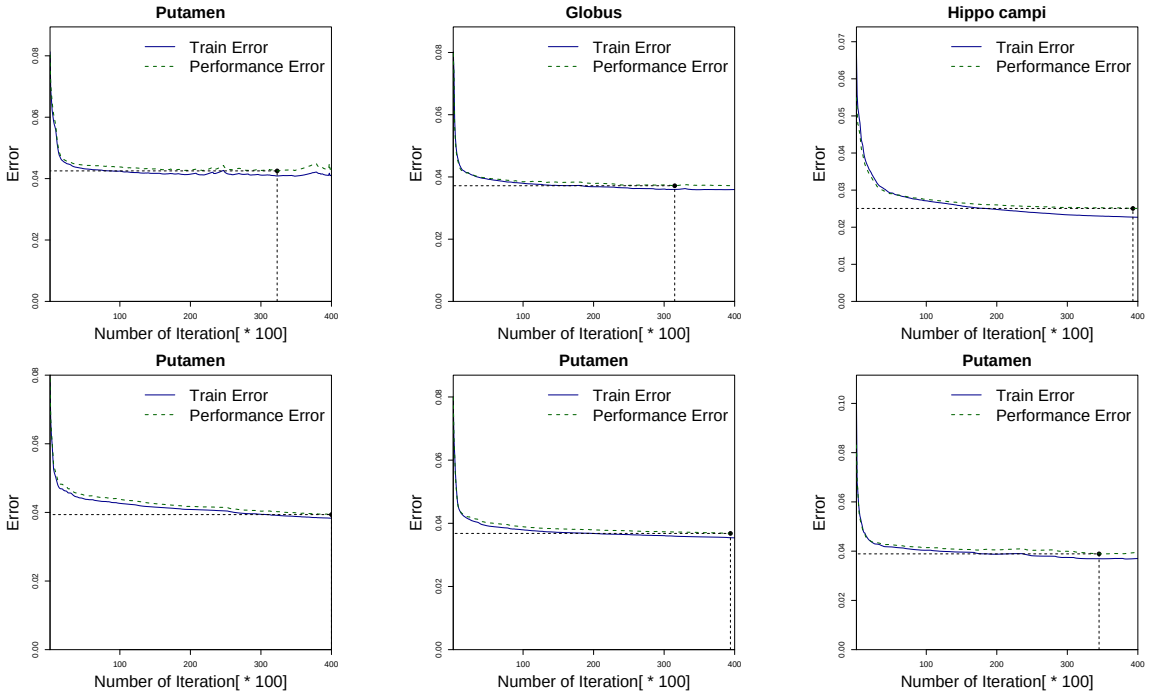


Figure 4.1: Mean Square Error for Individually Trained ANN Model: Train error (solid blue) and performance error (dotted green) graph for investigation of optimally trained point. The first row of the figure is different structures from same setting of training ($HN = 60$, $Grad = 1$), and the second row represents ANN model trainings for same structure across different training settings (from left to right, $HN = 50, 60$and 70). All ANN model over-fitted well for further investigation.

true that the results have a minumum variance. The variance is represented as dotted line. Based on the relative overlap measure graph, therefore, we concluded that the global optimal threshold value is around 0.4. What we can also infer from these RO versus threshold graph is that the accumben and globus do not have RO value as high as others. This was the case for most of other training models introduced in this research. Addtional graphs for other individually trained ANN models are presented in Appendix A.

Intraclass correlation (ICC) measures were evaluated followed by RO. As figure 4.3 shows, the ICC agreement has similar shape with relative overlap measure, which lead us to coincide conclusion of threshold value with RO investigation. The ICC consistency, however, allows us to see different aspect of training result. The ICC-

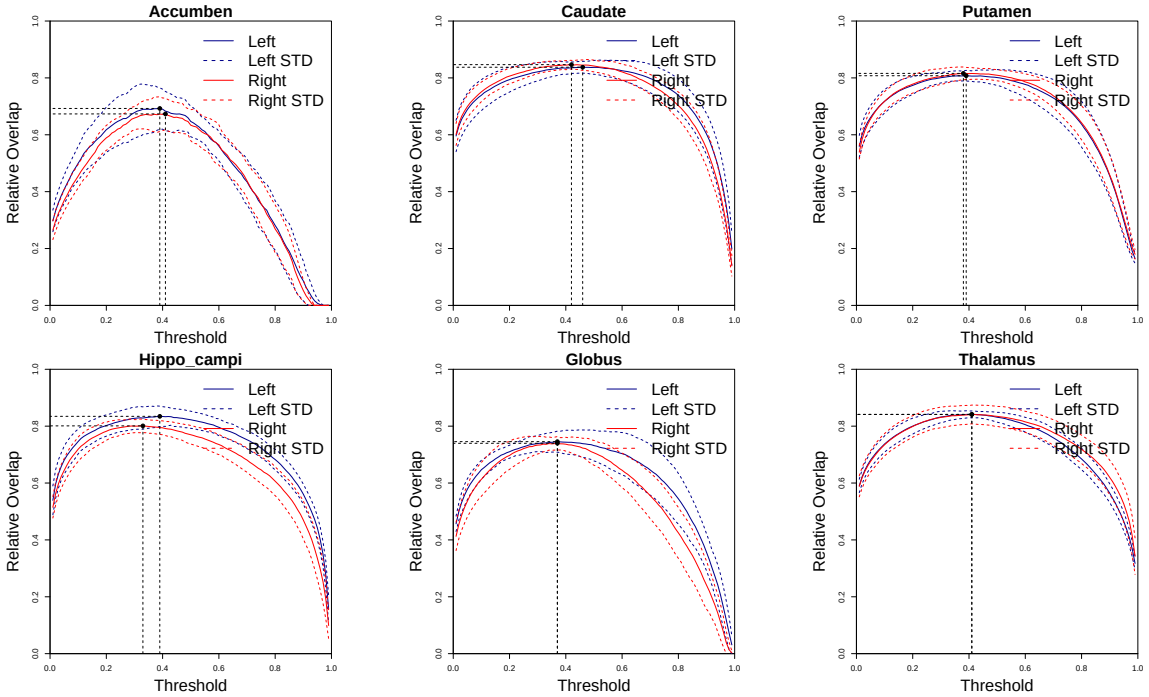


Figure 4.2: Relative Overlap versus Threshold for Individually Trained ANN Model: Given set of ANN model, the relative overlap computed between manual and ANN results to find optimal threshold value. The value around 0.4 is chosen as an optimal threshold value for our model. The training parameter is length of neighbors along the gradient direction ± 1 with Number of Hidden Nodes 60.

consistency tells us that any threshold value between 0.2 and 0.7 would produce the result as consistent as threshold 0.4. As long as the threshold value stays in the range, the ANN result will consistent to the manually traced data. Both *ICC* agreement and consistency suggests that ANN will produce consistent result to human between 0.2 and 0.7, however if absolute agreement were required for research, the choice of 0.4 for threshold would be more proper. Our choice of threshold was 0.4 for entire structure as a global optimal.

4.1.3 Segmentation Results

In this section, statistical results for comparison between ANN segmented and manually delineated images. Each sub-cortical structure applied by ANN models with various parameters, HN and $Grad$.

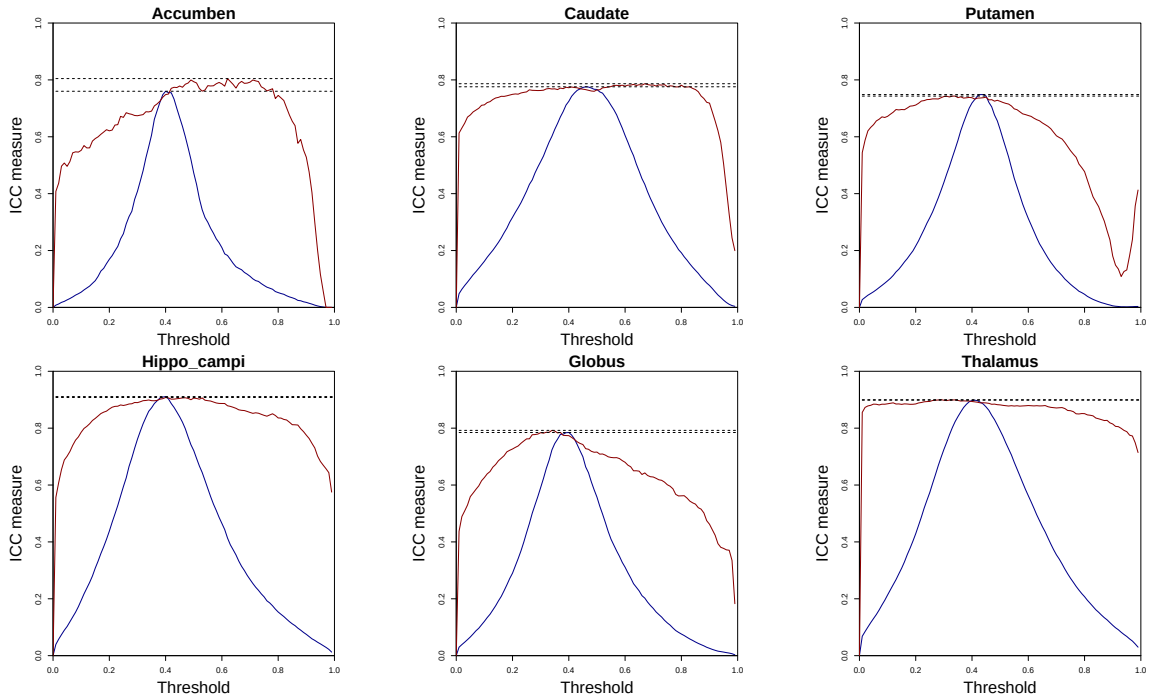


Figure 4.3: ICC measures versus Threshold for Individually Traind ANN: ICC consistency (Red) and ICC agreement (Blue) for $HN = 60$ with $Grad = 1$

Nucleus Accumben

A summary of the ANN segmentation results for the nucleus accumben is shown in table 4.1. First column shows the manual mean volume and standard deviation of the 8-test data sets, which is compared with the rest of experimental values. We observed that the size of neighbors along the descent rather disturbs training with its larger value. Even though RO or SI shows the best value with the model of $HN = 50$, $Grad = 1$, the highest value for ICC is the model with $HN = 60$, $Grad = 1$. The visual inspection figure for the one of the best and worst result appeared on figure 4.4.

Table 4.1: Accumben Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	322.1	50.9	-	-	-	-	
1	50	324.5(0.75%)	56.5(11.0%)	0.757± 0.0853	0.741 ± 0.0830	0.74	0.73	0.73
1	60	320.6(-0.47%)	62.2(22.2%)	0.752± 0.0874	0.737 ± 0.0832	0.76	0.75	0.72
1	70	319.9(-0.68%)	65.2(28.1%)	0.748± 0.0838	0.744 ± 0.0743	0.66	0.65	0.67
1	80	319.1(-0.93%)	60.6(19.1%)	0.755± 0.0816	0.739 ± 0.0888	0.66	0.64	0.65
2	60	313.8(-2.58%)	62.1(22.0%)	0.747± 0.0832	0.727 ± 0.0806	0.71	0.70	0.72
2	80	302.9(-5.96%)	62.6(23.0%)	0.729± 0.0899	0.722 ± 0.1009	0.58	0.60	0.61

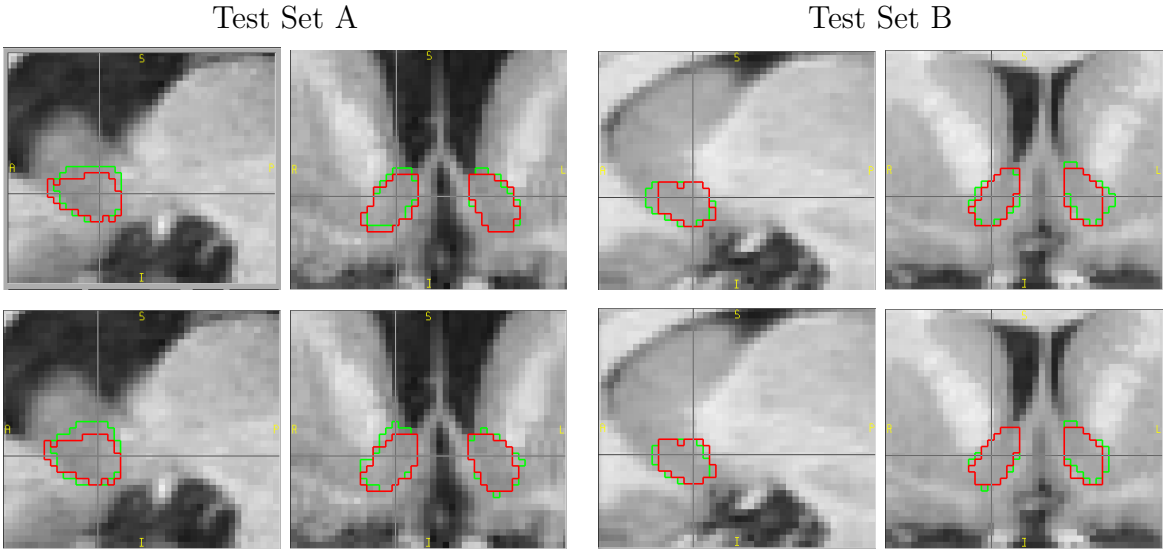


Figure 4.4: Visual Example for Individually Trained Segmentation: Accumben. $Grad = 2$ and $HN = 80$ upper row and $Grad = 1$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Lime) segmentation appeared on the view of (121,123,134) for (Sagital, Axial, Coronal)

Caudate

Summarization of ANN segmentation result for caudates is in table 4.2. The first column shows the manual mean volume and standard deviation of 8-test data set to be compared to the rest of experimental values. The optimally trained model seems to be with $HN = 60$, $Grad = 2$, since it has the highest measures for the

all statistics. The ANN model for caudates only case that has best with increased number of neighbor along the descent ($Grad$). We also identified that the ANN model with $HN = 80$, $Grad = 2$, has a relatively low agreement with high consistency. The visual inspection figure for the one of the best and worst result appeared on figure 4.5.

Even if there exists some variability in their statistical results, the entire results could be interpreted to show that the model were successfully trained. Shrout and Fleiss [32] suggested that the ICC value above 0.75 would be good enough for general application. In addition to that, result comparing to the result presented by Powell et al. [26], RO measures were increased from 0.83 with similar variances. This indicates that the newly adapted features for sub-cortical training contributed to the better result.

Table 4.2: Caudate Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	3178.3	323.2	-	-	-	-	
1	50	3315.2(4.4%)	426.0(31.8%)	0.874± 0.0420	0.877 ± 0.0397	0.75	0.79	0.82
1	60	3324.9(4.7%)	407.2(26.0%)	0.872± 0.0425	0.880 ± 0.0388	0.74	0.77	0.79
1	70	3334.6(5.0%)	425.7(31.7%)	0.873± 0.0422	0.877 ± 0.0406	0.73	0.78	0.79
1	80	3178.2(0.1%)	427.4(32.2%)	0.876± 0.0416	0.879 ± 0.0389	0.76	0.79	0.82
2	60	3277.9(3.2%)	416.6(28.9%)	0.879± 0.0416	0.880 ± 0.0392	0.78	0.80	0.82
2	80	3178.2(0.1%)	385.9(19.4%)	0.878± 0.0413	0.878 ± 0.0386	0.58	0.82	0.83

Globus

The Globus structure summarization is in table 4.3. The first column shows the manual mean volume and standard deviation of 8-test data set to be compared to the rest of experimental values The relative overlaps or similarity index with $HN = 80$,

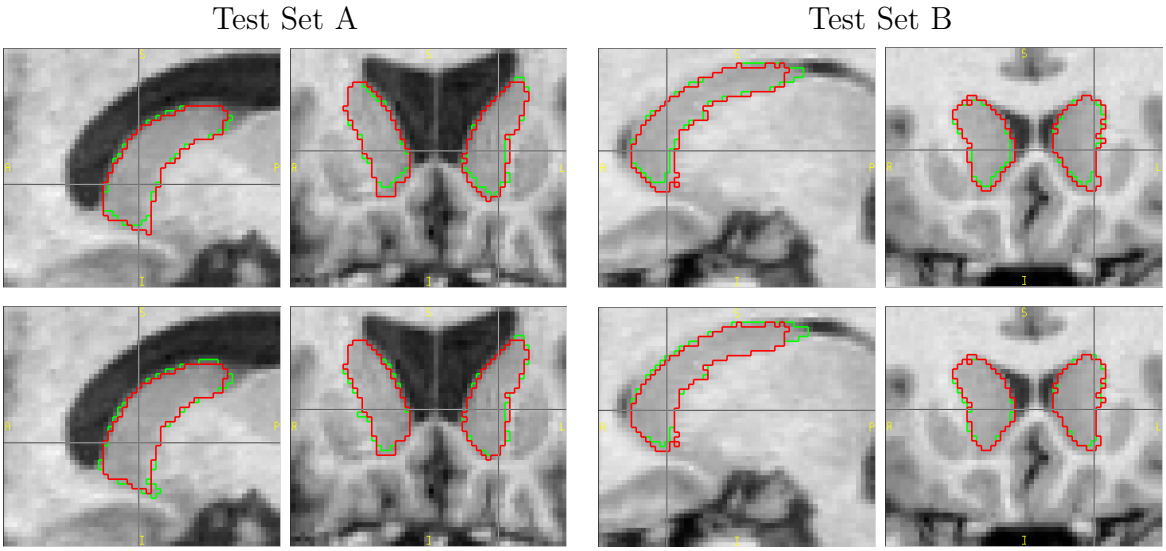


Figure 4.5: Visual Example for Individually Trained Segmentation: Caudate for $Grad = 2$ and $HN = 80$ upper row and $Grad = 2$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Lime) segmentation appeared on the view of (141,131,142) for (Axial,Coronal,Sagital) Plain

$Grad = 2$ can be said by best ICC and other statistics, however indicates that the ANN model with $HN = 60$, $Grad = 1$ is the best choice for optimal ANN trained model for globus The visual inspection figure for the one of the best and worst result appeared on figure 4.6.

Table 4.3: Globus Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	1581.2	196.0	-	-	-	-	-
1	50	1573.7(-0.5%)	236.2(20.5%)	0.790± 0.0661	0.787± 0.0635	0.77	0.76	0.78
1	60	1577.6(-0.2%)	253.6(29.4%)	0.798± 0.0658	0.792± 0.0610	0.78	0.77	0.80
1	70	1539.2(-2.7%)	256.8(31.0%)	0.790± 0.0682	0.790± 0.0630	0.74	0.74	0.77
1	80	1559.2(-1.4%)	231.7(18.2%)	0.790± 0.0669	0.793± 0.0617	0.78	0.77	0.78
2	60	1567.4(-0.9%)	232.6(18.7%)	0.801± 0.0618	0.805± 0.0593	0.76	0.75	0.76
2	80	1558.4(-1.4%)	238.6(21.7%)	0.807± 0.0612	0.798± 0.0611	0.75	0.74	0.75

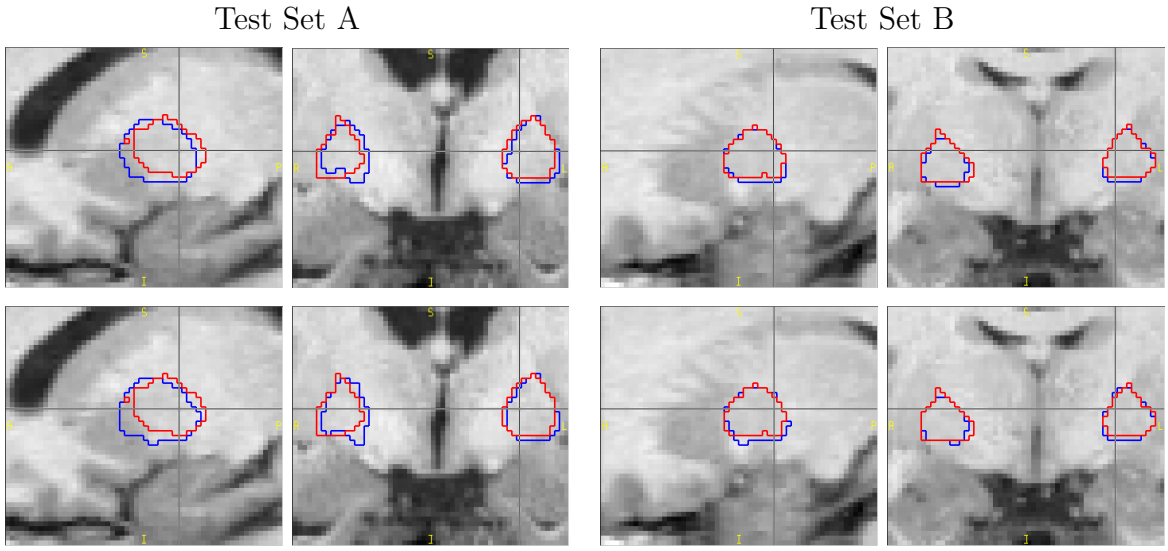


Figure 4.6: Visual Example for Individually Trained Segmentation: Globus for $Grad = 2$ and $HN = 80$ upper row and $Grad = 1$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Blue) segmentation appeared on the view of (145,131,120) for (Axial,Coronal,Sagital) Plain

Hippo campi

Both side of hippo campi were trained with gradient descents size 1 for various number of hidden nodes. The result of hippo campi produced one of the most successful result. As *ICC*-agreement shows in the table 4.4, these ANN model gave highly reliable segmentation result compared to manual golden standard. Among all of our trained ANN model, the 60's hidden nodes model seems to be thethe optimal architecture with the highest scores for any measurement we computed. The *ICC*-consisency also remains at high score for broad range of threshold value, which means any threshold value between 0.2 and 0.8 would produce consistent result to the human rater.

Table 4.4: Hippo campi Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	2198.4 246.1	-	-	-	-	-	
1	50	2175.4(-1.0%)	260.9(5.7%)	0.865± 0.0471	0.839± 0.00510	0.90	0.89	0.90
1	60	2181.8(-0.8%)	258.8(4.8%)	0.872± 0.0476	0.840± 0.00509	0.91	0.91	0.91
1	70	2173.6(-1.1%)	280.9(13.8%)	0.859± 0.0473	0.836± 0.00512	0.89	0.89	0.90
1	80	2161.3(-1.7%)	259.1(4.9%)	0.870± 0.0461	0.837± 0.00513	0.90	0.90	0.90
2	60	2172.4(-1.2%)	258.4(4.7%)	0.871± 0.0475	0.838 ± 0.0512	0.91	0.91	0.91
2	80	2174.9(-1.1%)	275.0(11.4%)	0.870 ± 0.0463	0.845± 0.0491	0.90	0.90	0.90

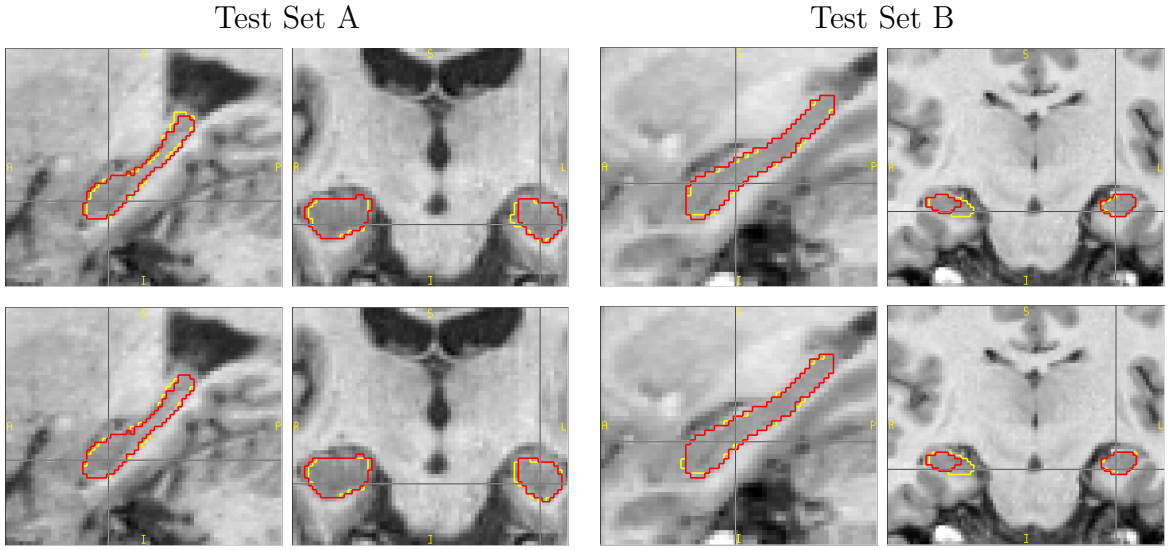


Figure 4.7: Visual Example for Individually Trained Segmentation: Hippo campi for $Grad = 2$ and $HN = 80$ upper row and $Grad = 1$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Yellow) segmentation appeared on the view of (154,113,109) for (Axial,Coronal,Sagital) Plain

Putamen

Putamen structure summarization is on table 4.5. First column shows the manual mean volume and standard deviation of 8-test data set to be compared to the rest of experimental values. The optimally trained model seems to be with $HN = 60$, $Grad = 1$, which has the highest measures for the every statistics. For putamen, it

seems like that the larger number of neighbors along the gradient descents (*Grad*) especially disturbs in learning process of ANN model. The visual inspection figure for the one of the best and worst result appeared on figure 4.5. Relative overlap measure were about to similar to the result by Powel et al. [26], but with larger variances. While mean volume indicates that ANN result overestimated the putamen, variances are generally appeared to be higher than human rater.

Table 4.5: Putamen Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	4670.6	321.5	-	-	-	-	-
1	50	4809.8(3.0%)	480.6(49.5%)	0.844± 0.0483	0.854± 0.0460	0.64	0.67	0.72
1	60	4788.3(2.5%)	474.4(47.6%)	0.850± 0.0469	0.857± 0.0459	0.72	0.74	0.79
1	70	4737.8(1.4%)	497.5(54.7%)	0.845± 0.0494	0.854± 0.0459	0.70	0.67	0.76
1	80	4764.1(2.0%)	528.7(64.5%)	0.831± 0.0532	0.844± 0.0478	0.56	0.56	0.63
2	60	4726.3(1.2%)	443.9(38.1%)	0.841± 0.0496	0.858± 0.0465	0.65	0.65	0.69
2	80	4733.1(1.3%)	465.4(44.8%)	0.847± 0.0473	0.855± 0.0478	0.64	0.64	0.68

Thalamus

Segmentation results for thalamus were one of the best results out of six sub-cortical structures. Thalamus structure summarization is on table 4.6. Any trained ANN model has very high *ICC* agreement, which is above 0.84 and the highest one reached at 0.90. Thalamus also has the higher RO measures compared to the Powells study. By inspecting one of the correlation graph between manual and ANN segmentation, the fitted line is well along the reference $V(manual) = V(ANN)$ line, which means perfect agreement.

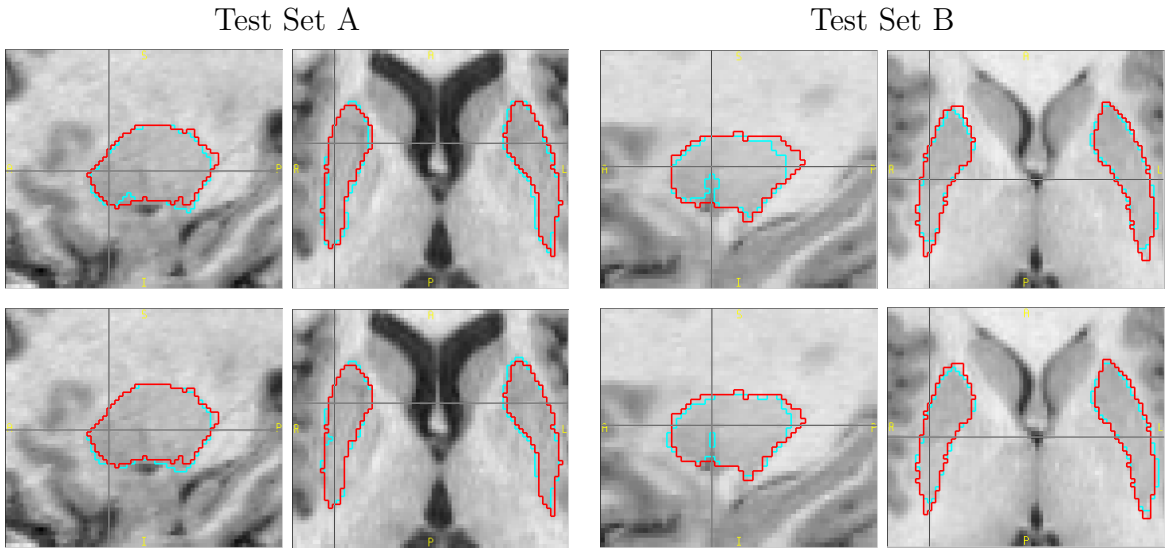


Figure 4.8: Visual Example for Individually Trained Segmentation: Puamen for $Grad = 2$ and $HN = 80$ upper row and $Grad = 1$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Light Blue) segmentation appeared on the view of (102,132,134) for (Axial,Coronal,Sagital) Plain

Table 4.6: Thalamus Individually Trained Nets Summary

Grad	*HN*	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
-	manual	6428.1	751.9	-	-	-	-	-
1	50	6523.3(1.5%)	822.2(9.3%)	0.880± 0.0388	0.878± 0.0447	0.84	0.84	0.84
1	60	6488.6(0.9%)	809.1(7.6%)	0.878± 0.0386	0.876± 0.0456	0.90	0.89	0.90
1	70	6512.4(1.3%)	809.1(7.6%)	0.878± 0.0395	0.878± 0.0449	0.90	0.88	0.90
1	80	6507.3(1.2%)	778.9(3.4%)	0.876± 0.0400	0.879± 0.0463	0.89	0.89	0.88
2	60	6510.9(1.3%)	821.6(9.3%)	0.878± 0.0392	0.880± 0.0430	0.88	0.87	0.88
2	60	6564.6(2.1%)	802.3(6.7%)	0.875± 0.0395	0.881± 0.0429	0.86	0.86	0.86

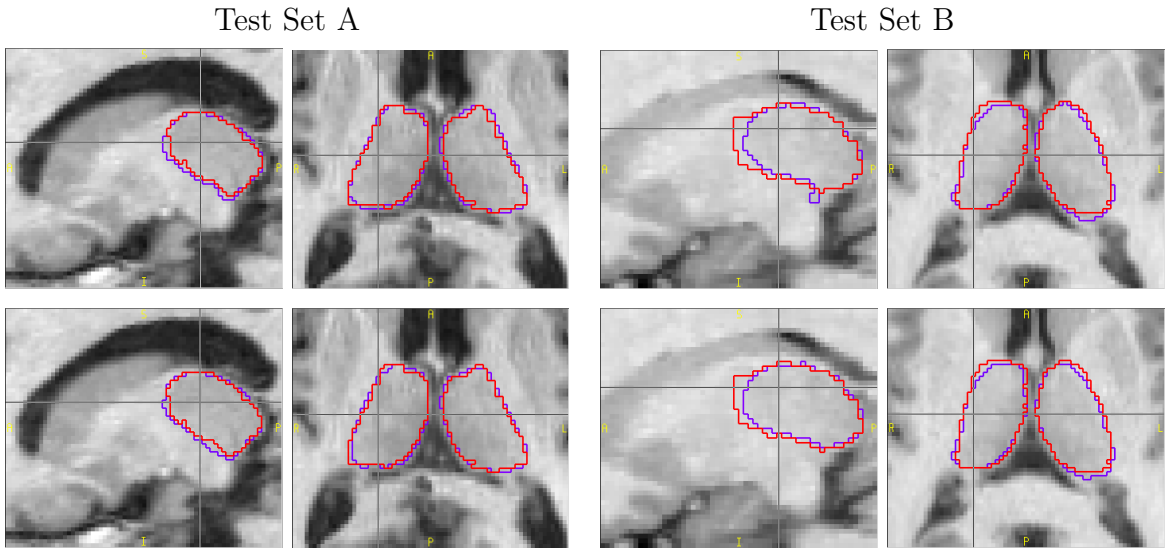


Figure 4.9: Visual Example for Individually Trained Segmentation: Thalamus for $Grad = 2$ and $HN = 80$ upper row and $Grad = 1$ and $HN = 60$ at the lower row. Manual (Red) and ANN (Purple) segmentation of (114,139,111) for (Axial,Coronal,Sagital) Plain

4.2 Segmentation Result with Thresold method for Neighboring Structures

The threshold method for neighboring structures was applied to the identical test set and compared with section 4.1 results. The relative overlap measures on figure 4.10 shows that the method does not disturb any segmentation and the relative overlap tendency against the threshold were preserved after the threshold applied. The summary of statistics appear in both figure 4.11 and table 4.2.

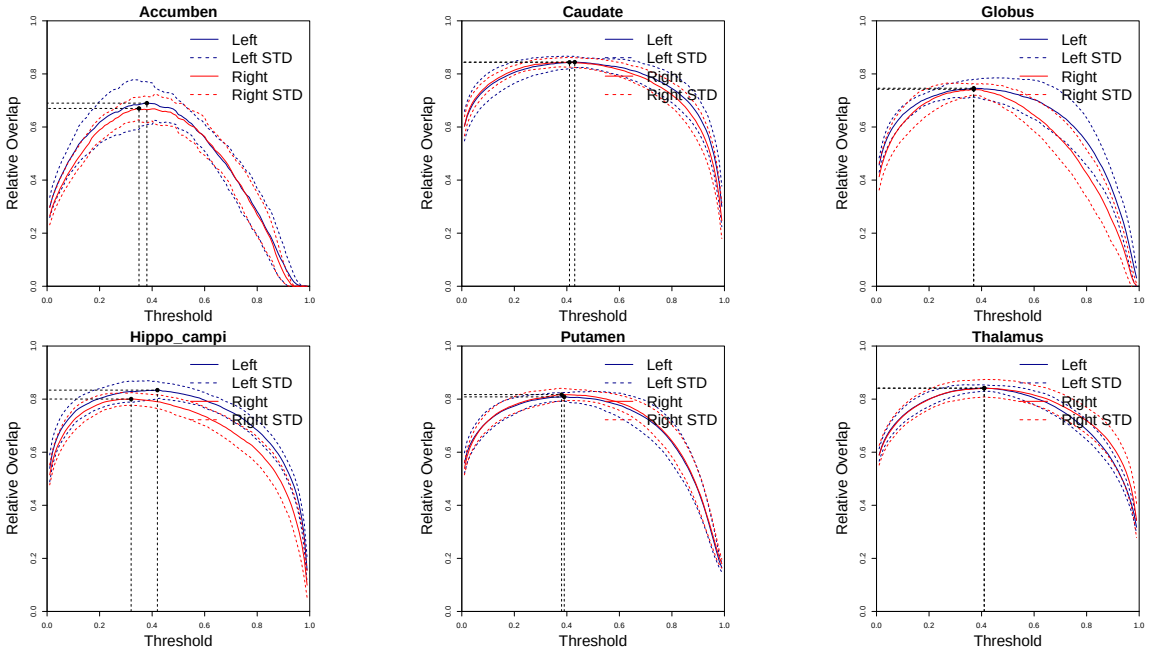


Figure 4.10: Relative overlap measure for the threshold method for neighboring structures

While the regions were in good agreement with the individually trained ANN model, such as Hippo campi and thalamus retained their reliability. The globus and putamen, however, increased their agreement by using the method.

As we observe from figure 4.12, the globus and putamen are adjacent structures. They

Table 4.7: Threshold result's Summary for Neighboring Structures from Optimal Indevidually Trained Nets Summary

structure	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
accumben	337.75(4.9%)	60.841(19.6%)	0.750 ± 0.091	0.734 ± 0.078	0.73	0.75	0.76
caudate	3294.6(3.7%)	419.05(29.7%)	0.878 ± 0.041	0.879 ± 0.039	0.78	0.8	0.83
globus	1573.3(-0.5%)	248.72(26.9%)	0.799 ± 0.063	0.794 ± 0.060	0.8	0.79	0.81
Hippo campi	2188.3(-0.5%)	259.58(5.5%)	0.870 ± 0.047	0.838 ± 0.050	0.91	0.91	0.91
putamen	4796.6(2.7%)	472.67(45.2%)	0.850 ± 0.046	0.857 ± 0.046	0.73	0.75	0.81
thalamus	6495.9(1.1%)	809.63(6.8%)	0.877 ± 0.038	0.876 ± 0.045	0.89	0.89	0.89

are close together, with no gap between them. The threshold method not only cleans up the overlapped area by creating disjoint definition for each structure, but also fills any gaps based on ANN output. Those two structures therefore improved more in their results. The nucleus accumben is adjacent to the caudate, and the result for it are also improved. Despite of improvement, because of its small size, the statistics for the accumben nucleus are not as significant as with other structures.

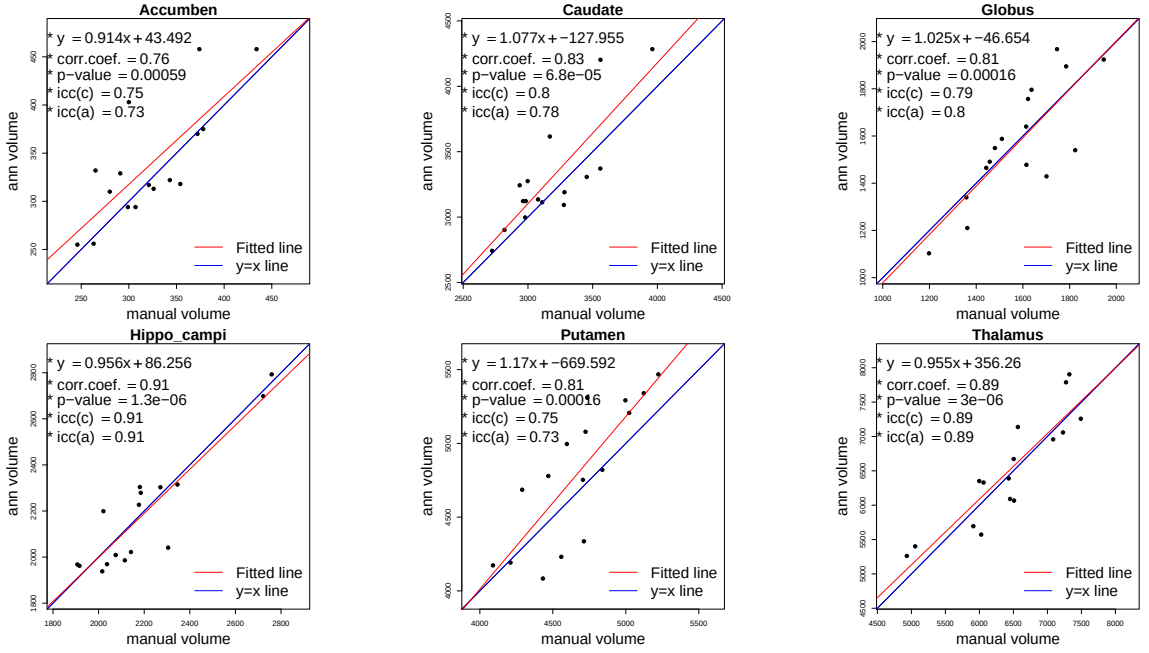
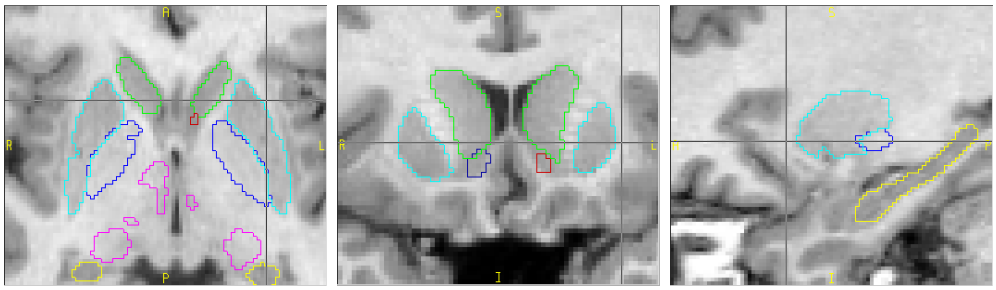


Figure 4.11: Summerization Graph for Threshold Method for Neighboring Structures

Overlapped Segmentation result from Individually Trained Net



Cleaned up overlapped area and filled the gaps in between structures

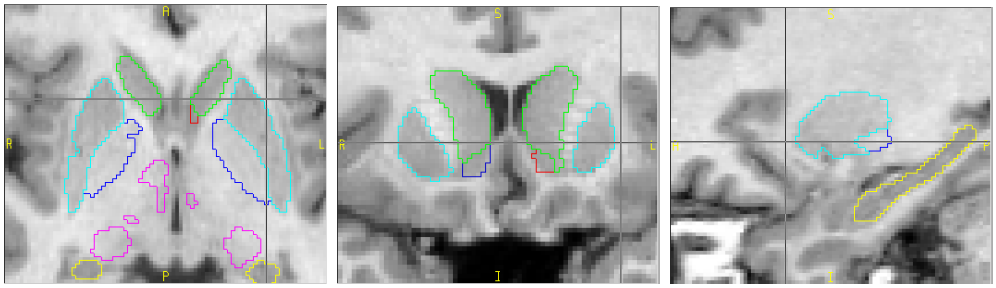


Figure 4.12: Comparison for Threshold method for neighboring structures

In summary, the threshold method helps to improve our individually trained ANN model segmentation results, especially for closely attached structures. Globus and putamen increases mostly and some improvement for accumben and caudate were observed as well. The segmentation result for thalamus and Hippo campi, which already in good agreement with the gold standard, retained their high degree of reliability.

4.3 ANN Segmenation Result for Simultaneously Trained Model

Segmentation results with simultaneously trained ANN models are presented here. Sub-cortical brain structures are defined as being disjoint, and having no spatial gaps between neighboring structures. In addition, brain structures are defined each other dependently, so if one structure needed to be defined, the neighboring structure's definition should be account. To take account these facts, the ANN model were trained to recognize all sub-cortical structures simultaneously. In general, the results show that relatively small structures were improved significantly, however it was not true for those that had good result with previous segmentation trials. Here we present how to select the optimal time point with the over-fitted ANN model, a threshold value choosing strategy, and segmentation results for sub-cortical structures.

4.3.1 Convergence Plot

Convergence plots were investigated in the same way as previous trials to find the optimal for ANN model. As shown in the figure 4.13, the error functions show some protruding points on the way their convergence. Except for those points, however, all convergence plots were flattened as the number of iteration increases. Interestingly enough, the protruding points occured more often as the number of hidden nodes get larger. It can be interpreted as meaning that there were so many gradient descents directions for RPROP algorithm to try out to find local minima.

Another observable phenomenon is that the training error came back to the stable point just after deviated from a convergence line, which is the way of RPROP algorithm works. The existed plots can be introduced because the algorithm works in this following way: if current update of weight increases the mean square error, it will revert to the previous state. After having the minimal value for performance

error function, we take that as the optimal time point.

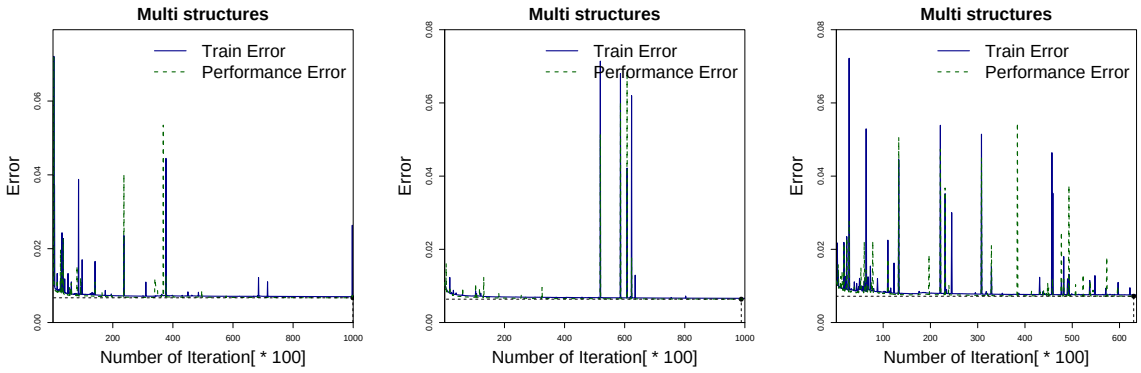


Figure 4.13: Convergence plot for Simultaneously trained model, from left to right with number of hidden nodes is 50,60,80.

4.3.2 Threshold

The optimal threshold with this method appears to be similar to the previous training experiment, at 0.4. The ICC versus threshold plot was notable for its increase, especially for the smallest volume sturcture, accumben. Since the accumben is located next to the caudate, putamen and globus, training all of those structures simultaneously should have helped to delineate boundaries more distinctly. For the globus and putamen, however, it turned out that the ICC values were not as high as independently trained ANN results. The other structures, such as Hippo campi and thalamus preserved their high status for ICC agreement and consistency.

4.3.3 Segmentation Result

This section describes segmentation results from different number of hidden nodes for simultaneously trained ANN, 'Multi-ANN' model. Even though the literature said that a larger number of hidden nodes would result in more accurate result [29], it was not applicable in this case. Especially the number of hidden nodes of 80 shows that the poorest performance on globus. Globally, even though results show that ICC-agreement were far less significant than previous trials, the ICC-consistency

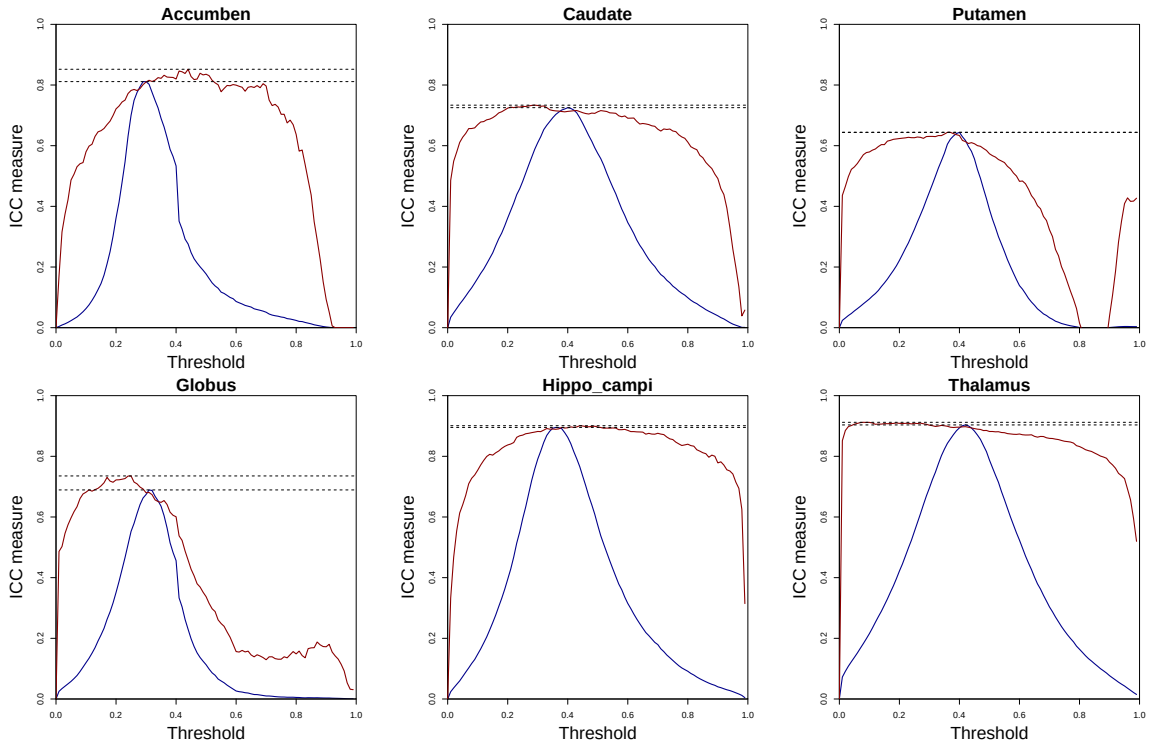


Figure 4.14: Intraclass correlation graph for simultaneously trained model with $Grad = 1, HN = 50$.

values retained their high values. In the next three sections, we will cover in more details for each Multi-ANN models' results of different number of hidden nodes.

Segmentation Result with the $HN = 50$

Figure 4.14 shows that the statistics between manually traced result and ANN model produced result. In the graph the reference line (blue) is represented for '$manual = ANN\_Result$', and we observed that the fitted line (red) by our data, assembles a lot of reference. Hippo campi and thalamus shows that fitting line highly follows the reference line with p-value less than 0.00001. For the accumbens, even though ICC-agreement suggests that the ANN output is not coincide to the golden standard, the ICC-consistency values still stays in high and also the p-value is less than 0.0001. For globus, the ANN segment application tends to over-estimate the structure than manual segmentation.

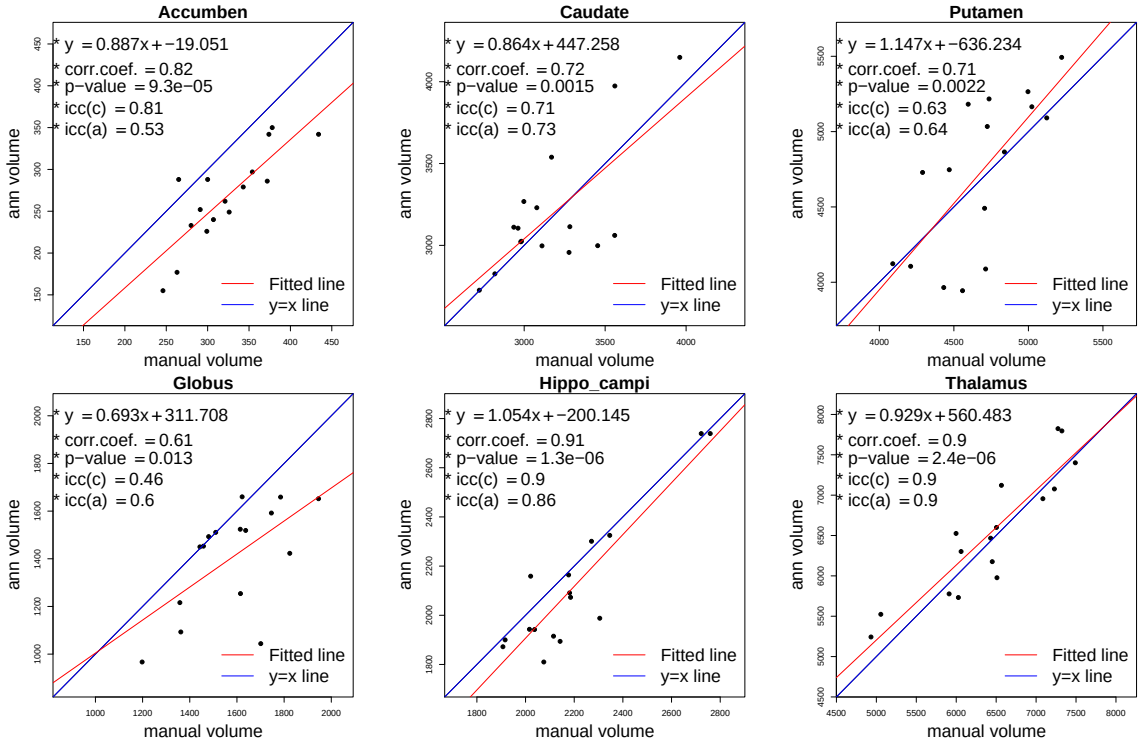


Figure 4.15: Intraclass correlation graph for simultaneously trained model with $Grad = 1, HN = 50$.

Segmentation Result with the $HN = 60$

The data with the $HN = 60$ shows more promising results of most of structures except for globus. The caudate, Hippo campi and thalamus has high correlation with the p-value less than 0.0001, and shows high ICC-agreement values, which is above 0.8. The putamen and accumben show weaker correspondences in all statistics than those structures but still p-value is small, which is less than 0.001. The globus segmentation, however, appeared to be less corresponding to the golden standard.

Segmentation Result with the $HN = 80$

We observed the weakest training result with the largest number of hidden nodes, 80 in multi-structure training case. This result makes no differences from individually trained net, which showed the weakest training for most of the structures. It means that the ANN model has begun to memorize the individual input-output pat-

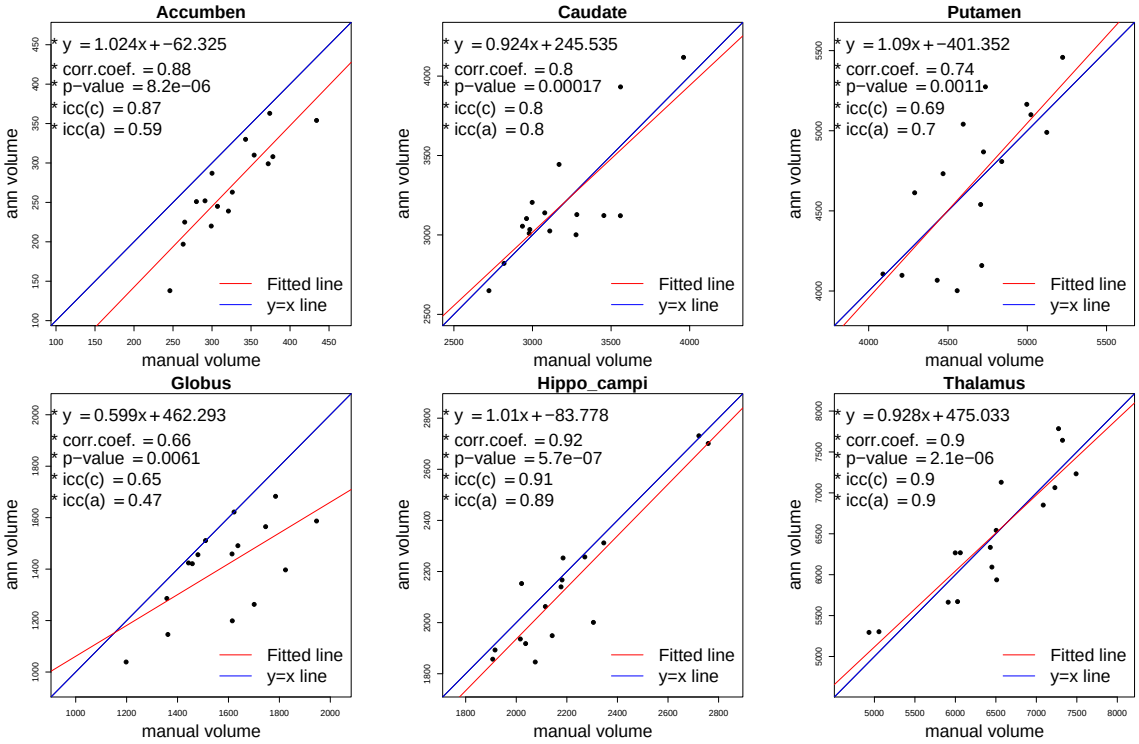


Figure 4.16: Intraclass correlation graph for simultaneously trained model with $Grad = 1, HN = 60$.

terns rather than settling for weights that generally describe the mapping for all case [29]. The over-fitting problem comes from either the larger number of iterations or larger number of hidden nodes, so that they can make a 'look-up table' for individual training set.

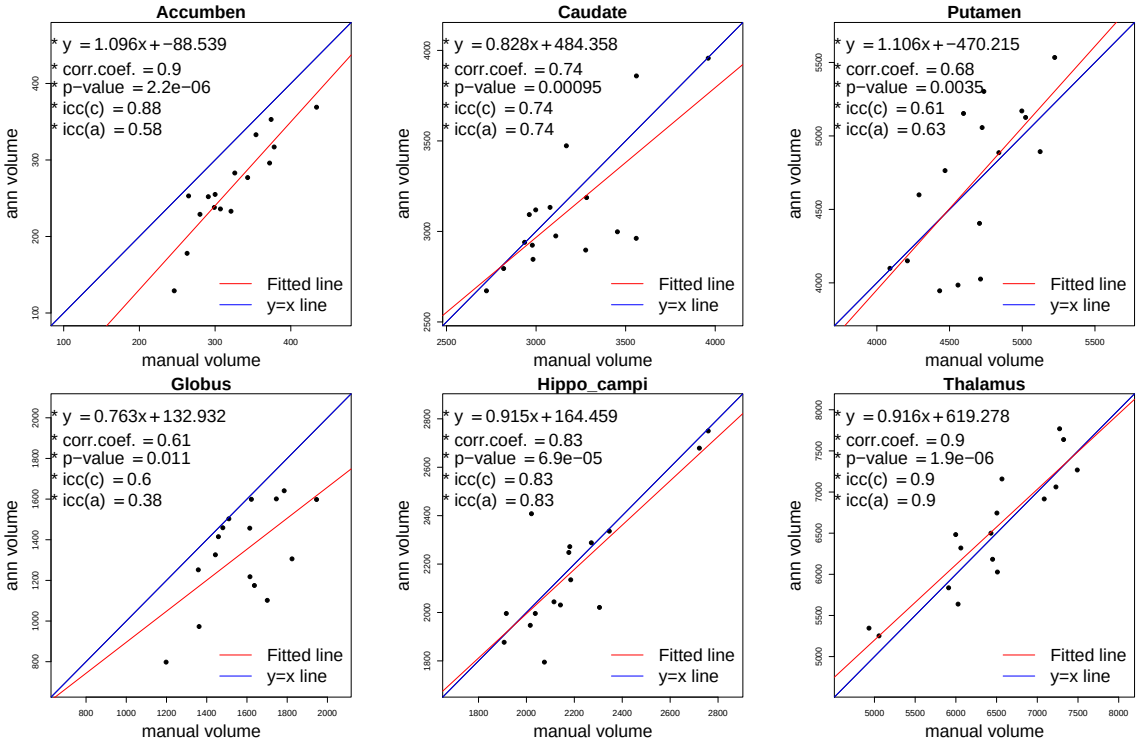


Figure 4.17: Intraclass correlation graph for simultaneously trained model with $Grad = 1, HN = 80$.

CHAPTER 5
CONCLUSION

MR imaging is widely used to see brain structures for diagnostic and research purposes. Sub-cortical brain structure research have been limited by current technology's inability to process large numbers of volumetric images efficiently and reliably. The automated segmentation method proposed here is motivated by these facts.

In this research, the ANN methodology, one of many approaches in Artificial Intelligence, was investigated for a sub-cortical brain segmentation application. The previous work by Powel et al. was adapted and modified for improvement in its usage and performance. The adapted method was tested in three main aspects: 1) newly utilized features' performance with improved registration method, 2) the ability of threshold method to deal with neighboring structures, and 3) efficiency and reliability of simultaneously constructed models for all sub-cortical structures.

Before diving into those three investigations in this research, the achievement of time efficiency should be mentioned first. Even if there is no standard for how fast it has to be, our method out-performed manual tracings. The manual segmentation task takes several hours of human labor time for one structure segmentation. In addition, a human must be trained for several days or weeks to become a reasonably good tracing expert. The segmentation time for ANN model would depend on performance of the computer used, however it takes about less than an hour and sometimeseven less than a half hour with laboratory computer for segmentations. Most of the time is taken up by image registration, so once registration has been acquired for a subject, the segmentation process for rest of the structures will finish in a few minutes.

The utilization of feature-enhanced images, which include a soft-tissue classified image and mean of gradient magnitude image, improves reliability of our segmentation. It is quantitatively proven that the volumetric overlaps with manual traces has

been increased to some extent compared with previous work [26]. And it has shown to be very reliable compared to the manual segmentation. What is more, structures with a relatively small volumes (nucleus accumben and hippo campi) or vague intensities (globus, thalamus and also nucleus accumben) were successfully identified in this work.

A mathematically adjusted threshold method also proved to contribute to increase the reliability of the method with negligible time cost. The utilization of threshold for neighboring structures allows our results to delineate more natural biological brain structure definitions in regarding of disjoint and fully defined nature between structures. This method was especially successful for a small volume structures.

The simultaneously trained ANN model for all of sub-cortical structures showed that its reliability for most of structures as high as individually trained and optimized ANN models. In addition, since it does not have to be run more than one times to get segmentations for several structures for a subject, it is even more efficient than individually trained ANN model.

Even though our ANN models were specially designed to produce human subcortical brain segmentation, it can be also easily adapted for more general segmentation work with any available image modalities and desired structures.

There is, however, still room for improvement. Firstly the simultaneously trained ANN models have met the obstacles in optimizing for each structure. Despite its time efficiency, it could not be optimized for every single structure, and thus a few structures' segmentation showed relatively low reliability compared with individually optimized one. Secondly, the ANN model is intrinsically not interpretable about its trained architecture itself, including nodes and their weights. If interpretable ANN model can be developed by adapting new technology such as computer ontology, it would give new impetus to ANN segmentation method.

APPENDIX A
ADDITIONAL GRAPH AND TABLE

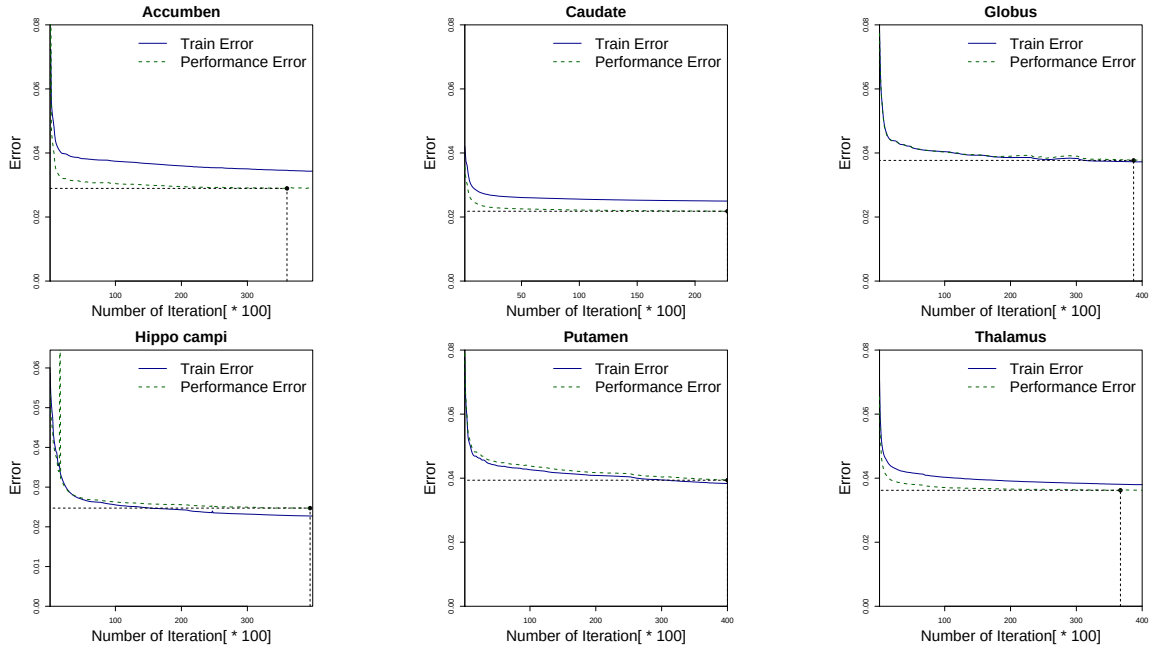


Figure A.1: Error function Graph for $HN = 50$ and $Grad = 1$.

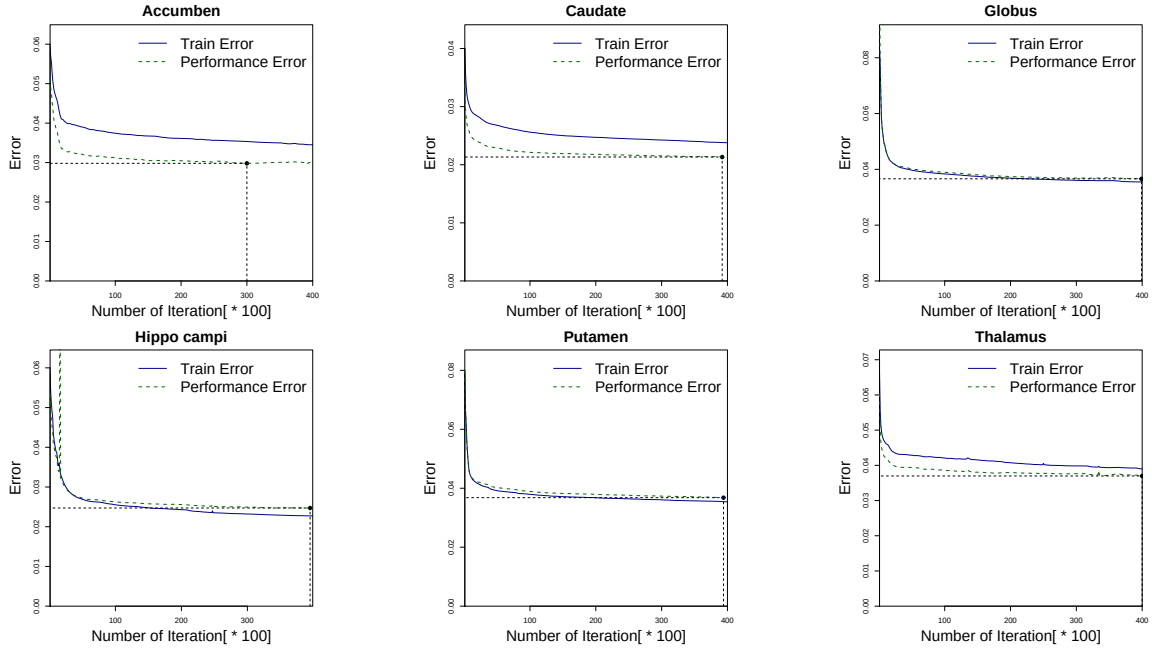


Figure A.2: Error function Graph for $HN = 60$ and $Grad = 1$.

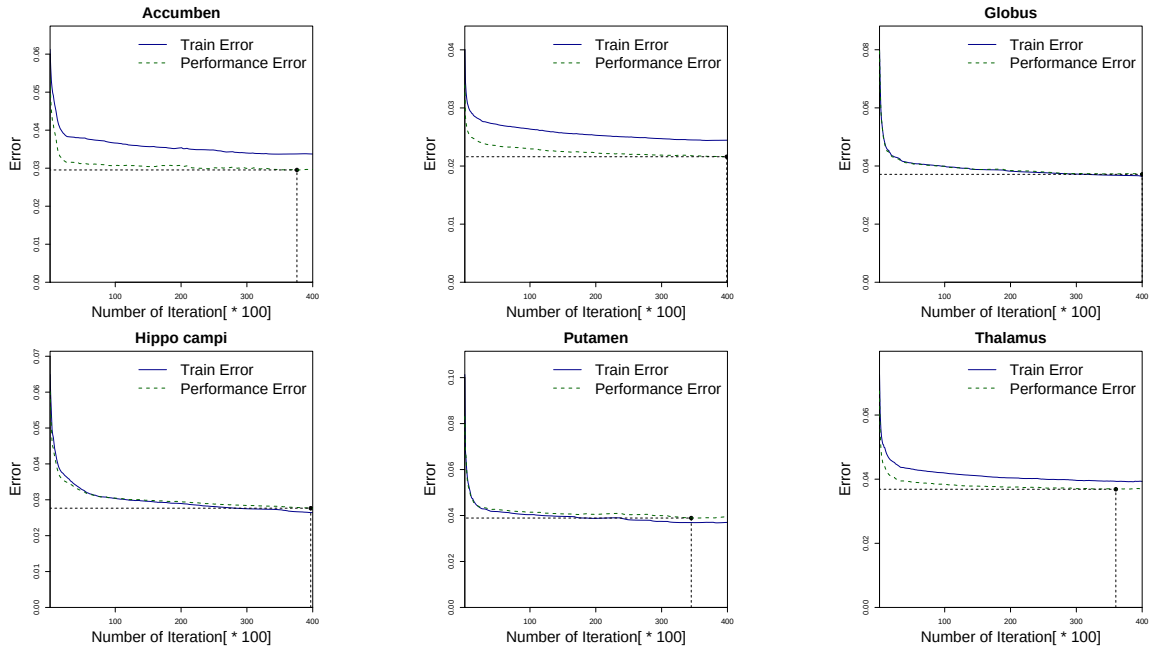


Figure A.3: Error function Graph for $HN = 70$ and $Grad = 1$.

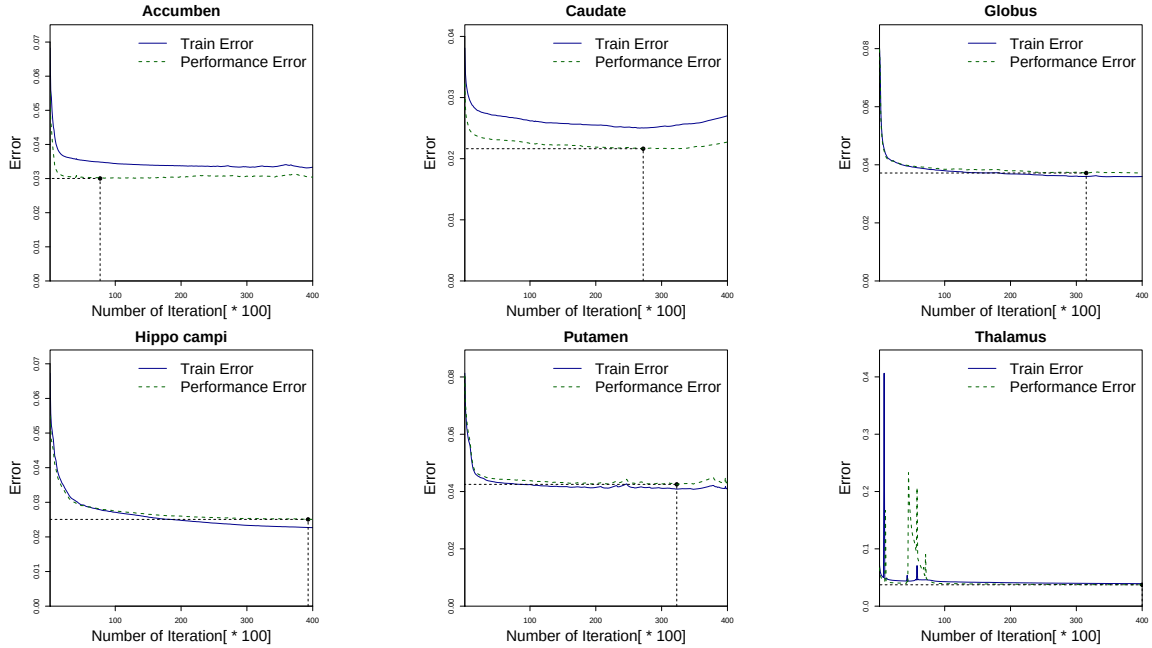


Figure A.4: Error function Graph for $HN = 80$ and $Grad = 1$.

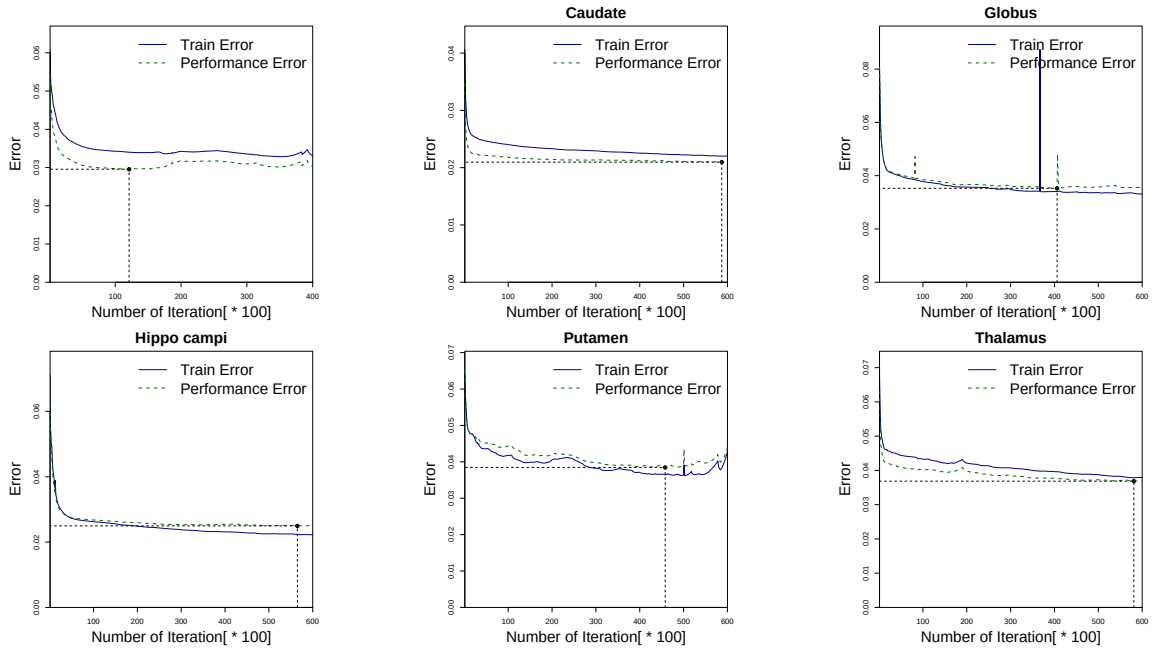


Figure A.5: Error function Graph for $HN = 60$ and $Grad = 2$.

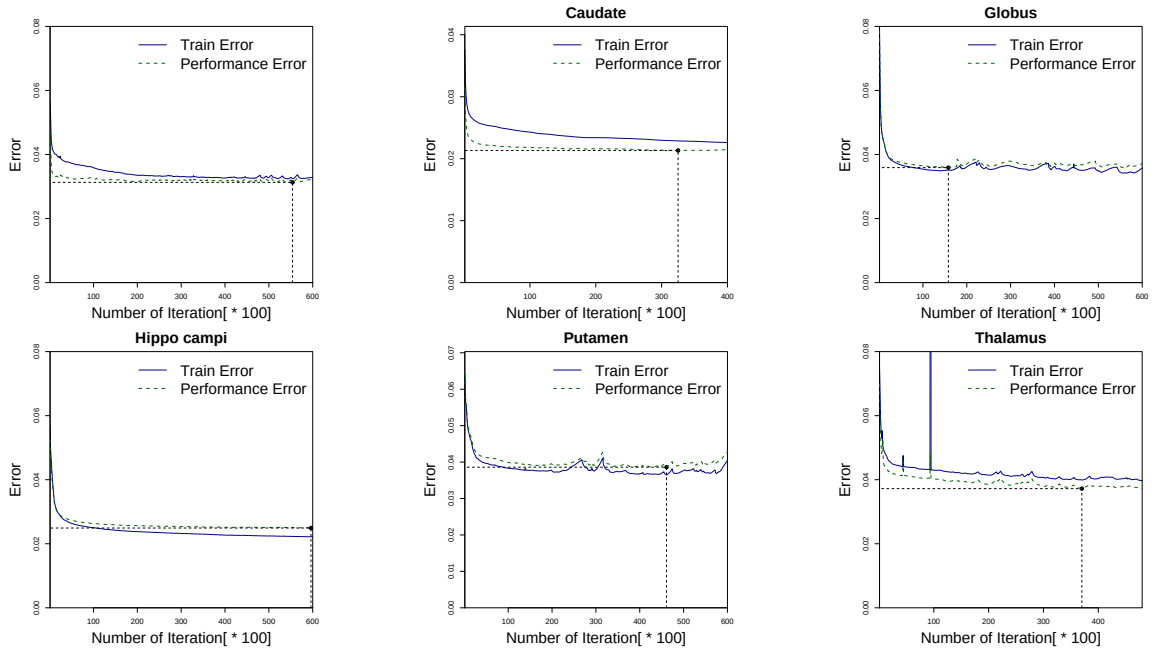


Figure A.6: Error function Graph for $HN = 80$ and $Grad = 2$.

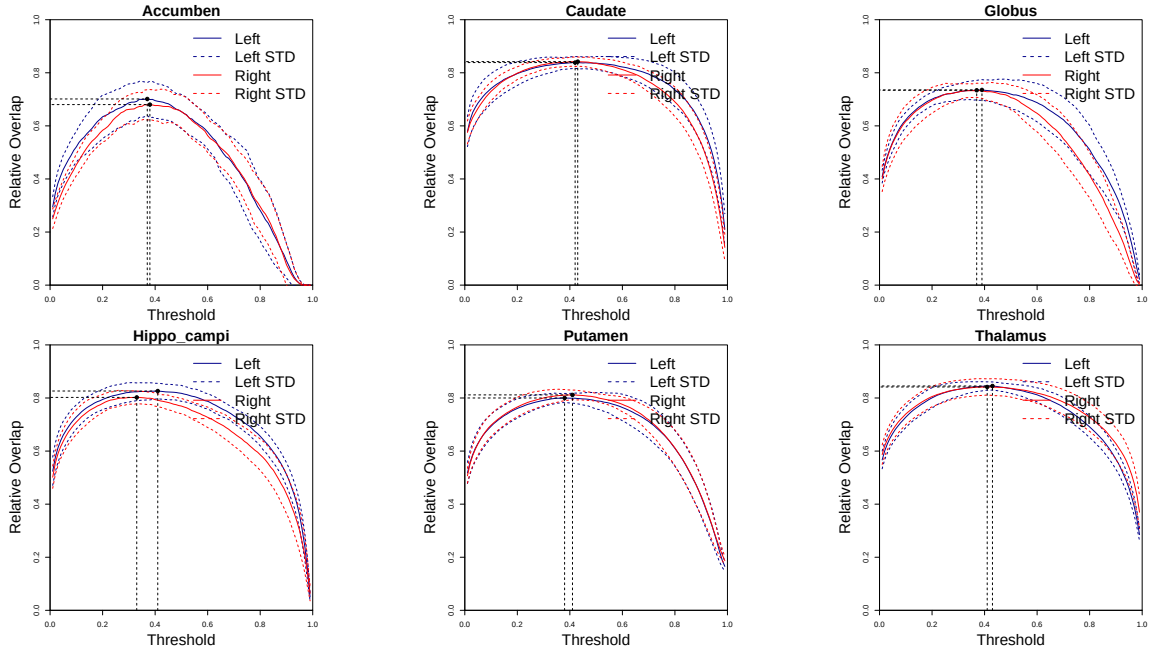


Figure A.7: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 50$

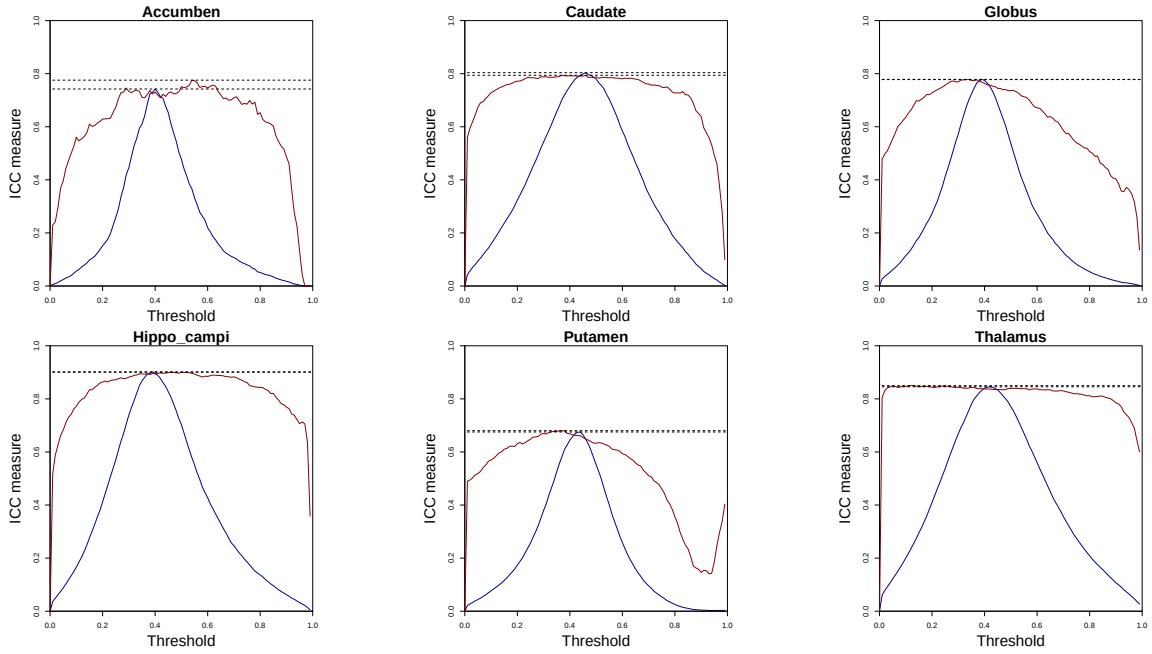


Figure A.8: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 50$

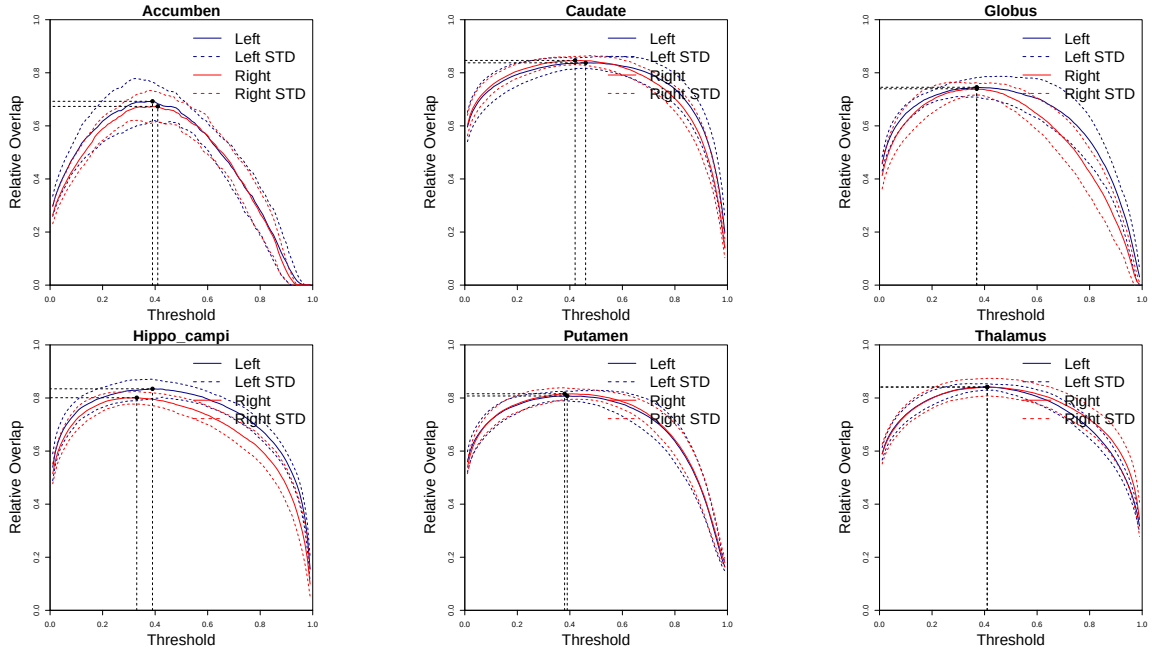


Figure A.9: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 60$

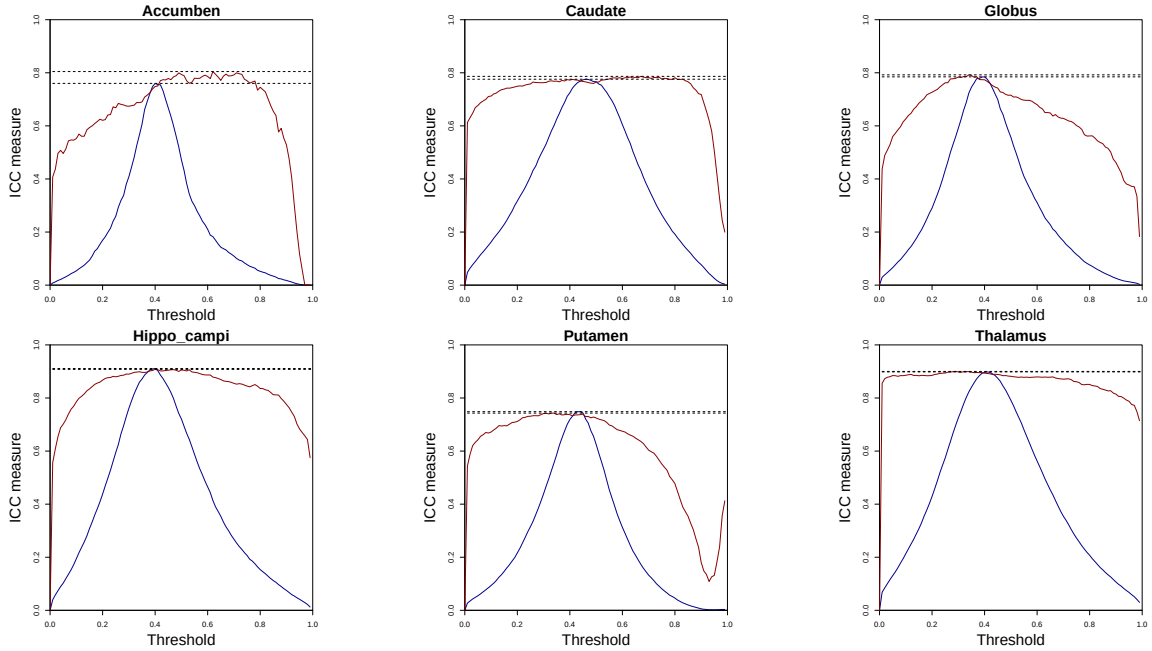


Figure A.10: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 60$

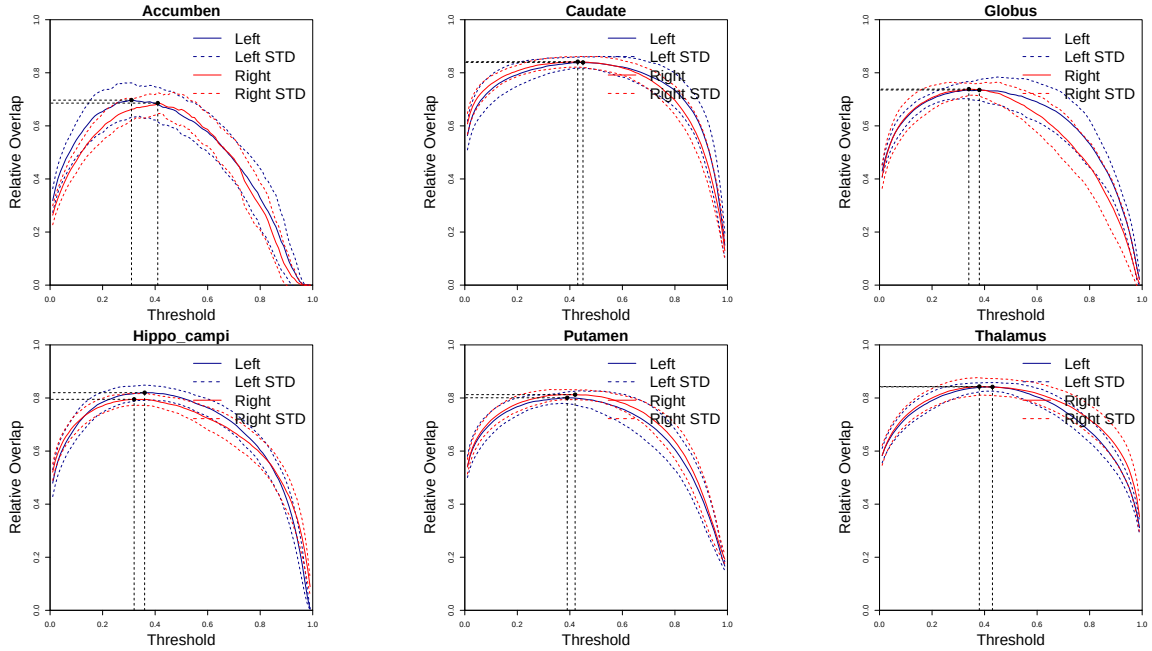


Figure A.11: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 70$

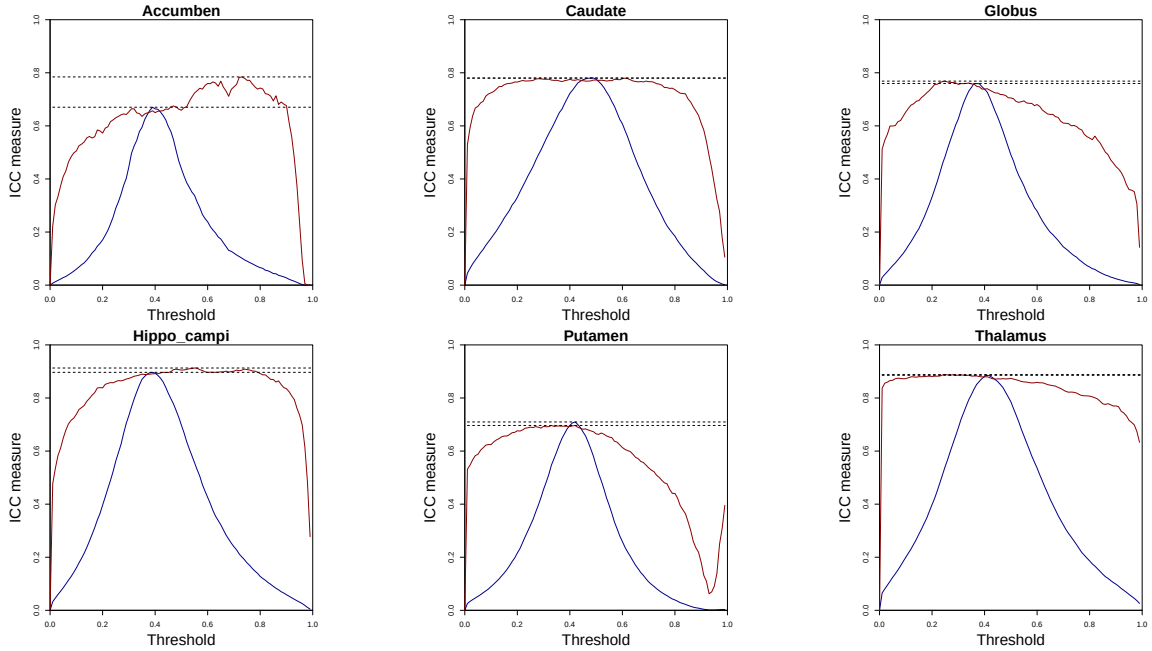


Figure A.12: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 70$

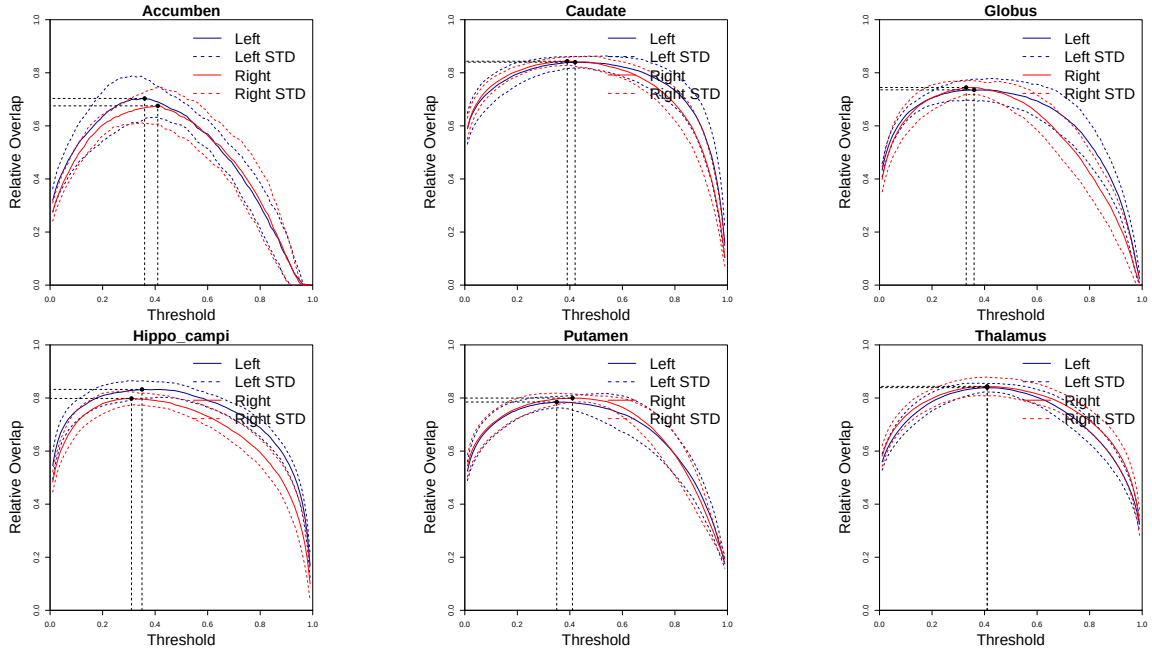


Figure A.13: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 80$

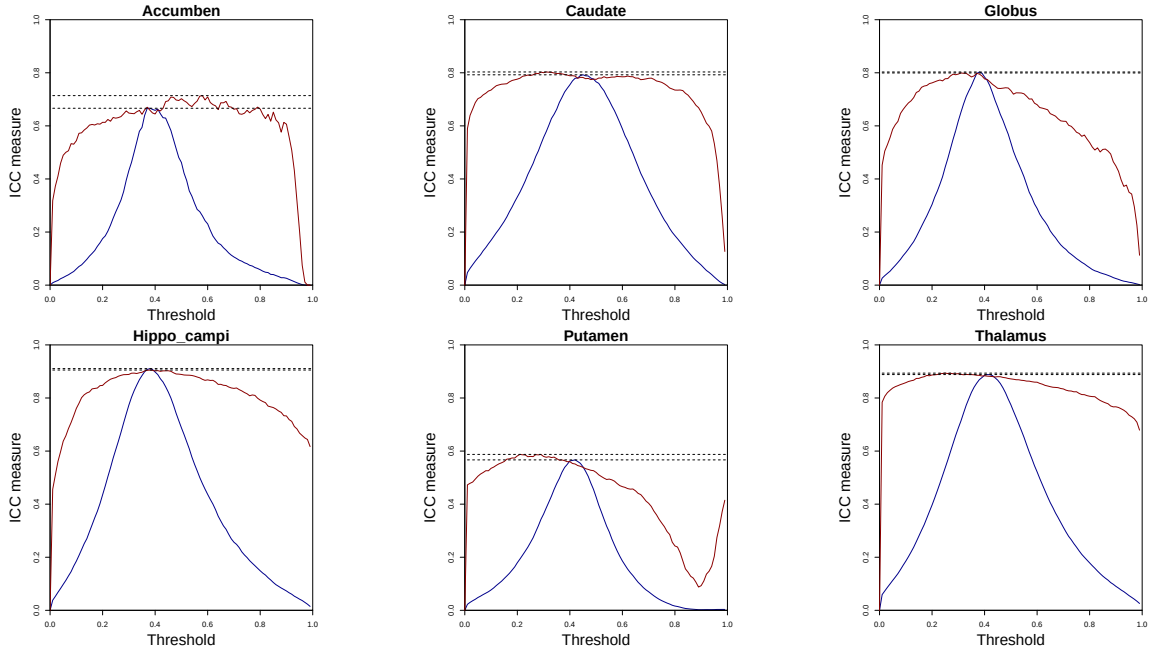


Figure A.14: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 80$

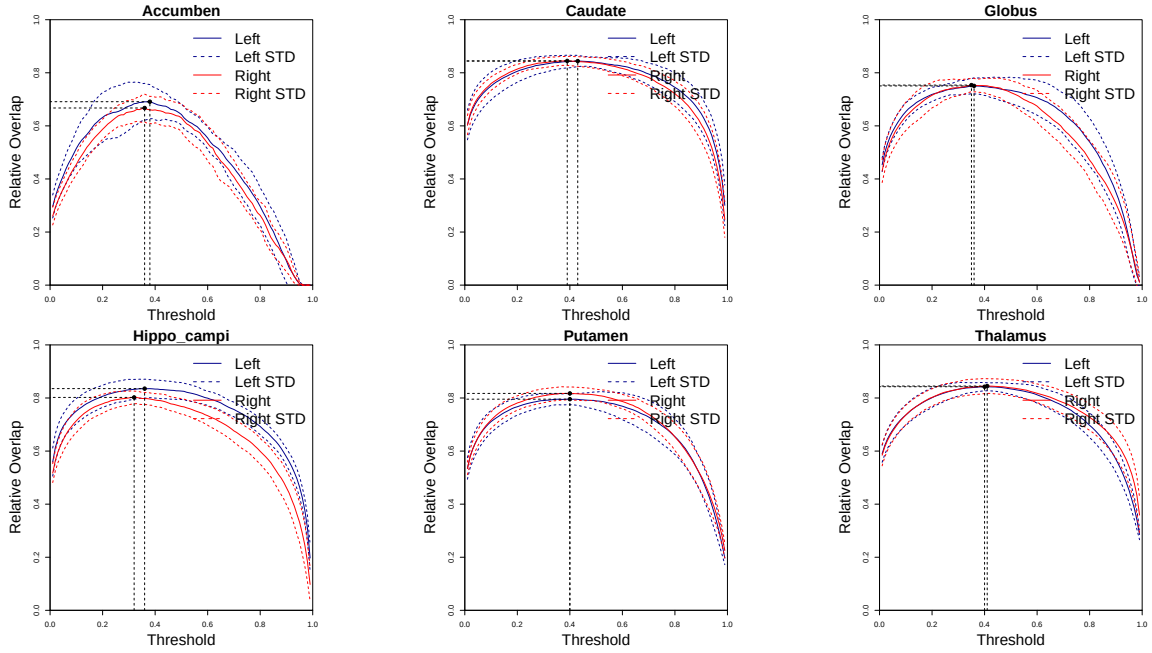


Figure A.15: Relative overlap (RO) measurements versus threshold for $Grad = 2$ $HN = 60$

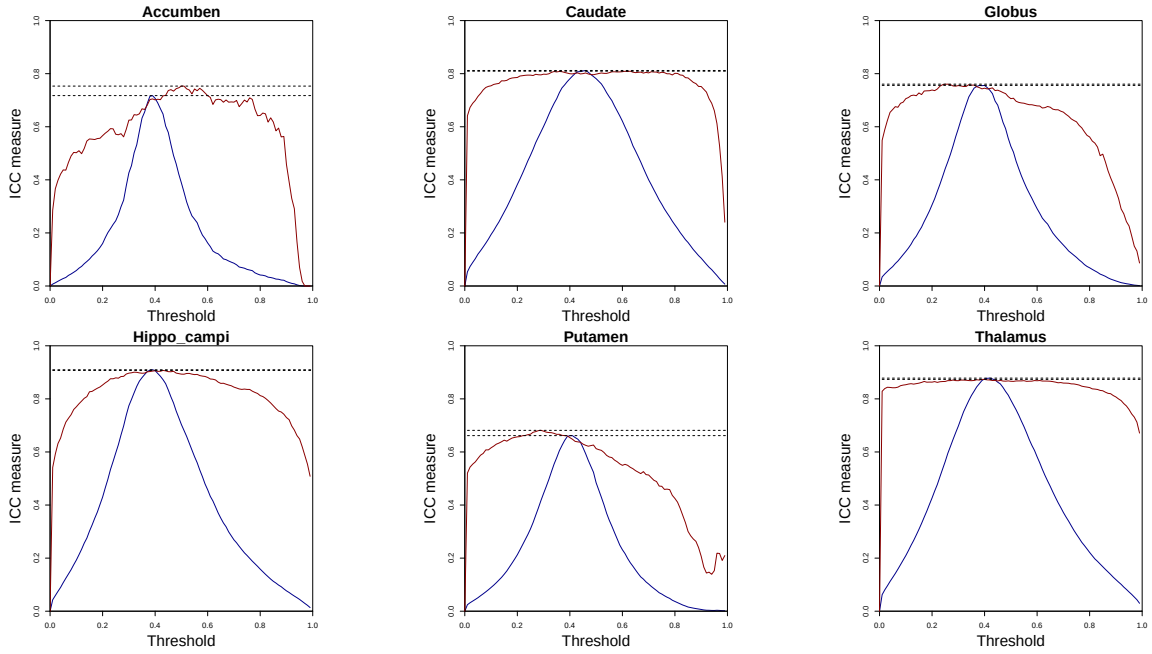


Figure A.16: ICC measeres vs. threshold value for $Grad = 2$ and $HN = 60$

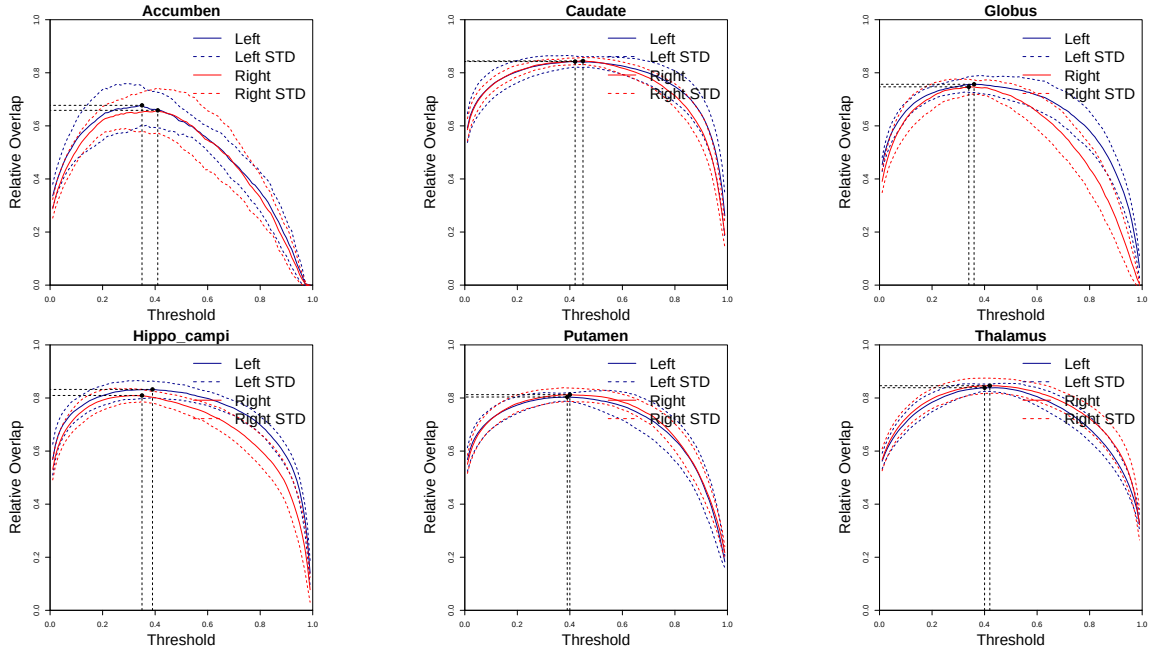


Figure A.17: Relative overlap (RO) measurements versus threshold for $Grad = 2$ $HN = 800$

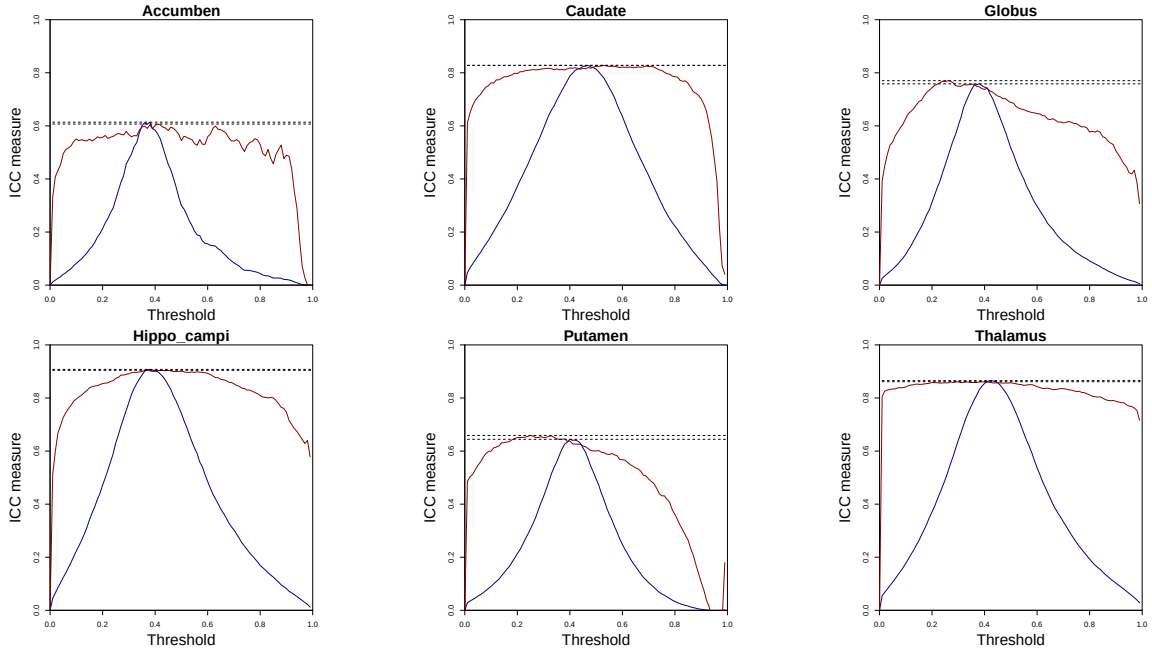


Figure A.18: ICC measeres vs. threshold value for $Grad = 2$ and $HN = 80$

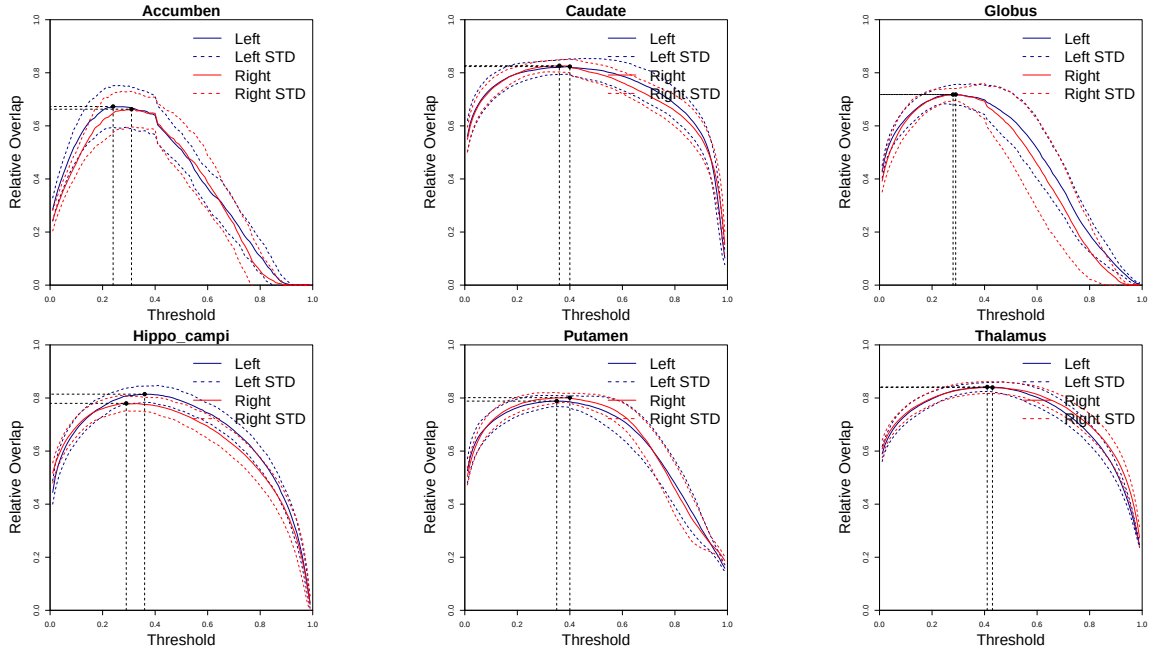


Figure A.19: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 50$

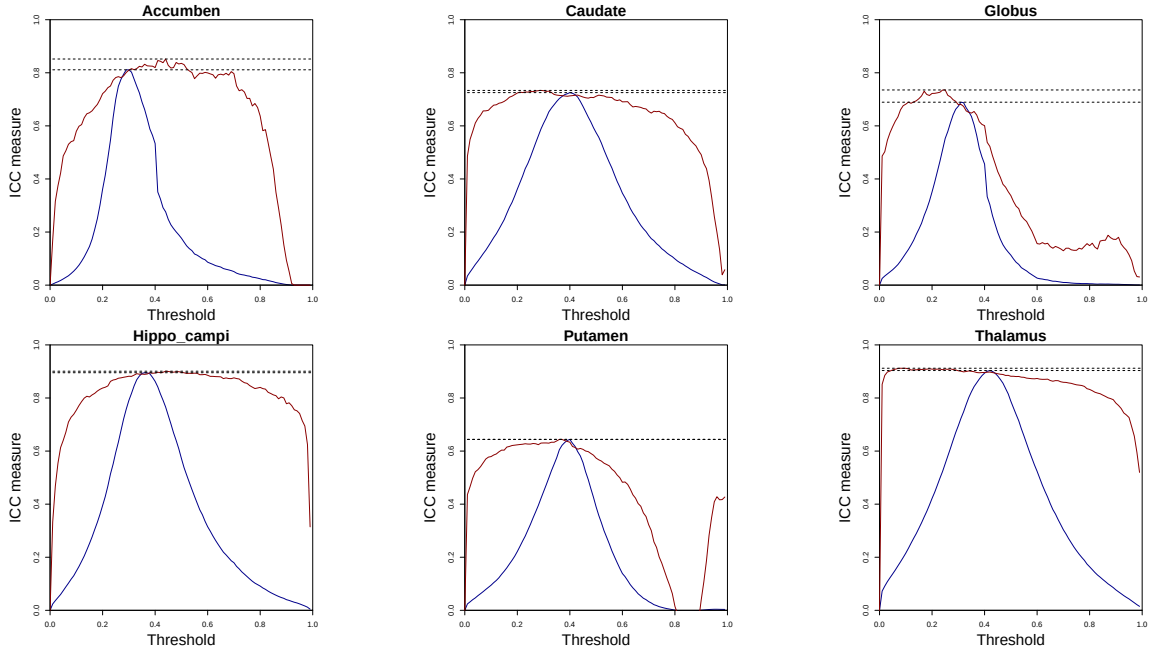


Figure A.20: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 50$

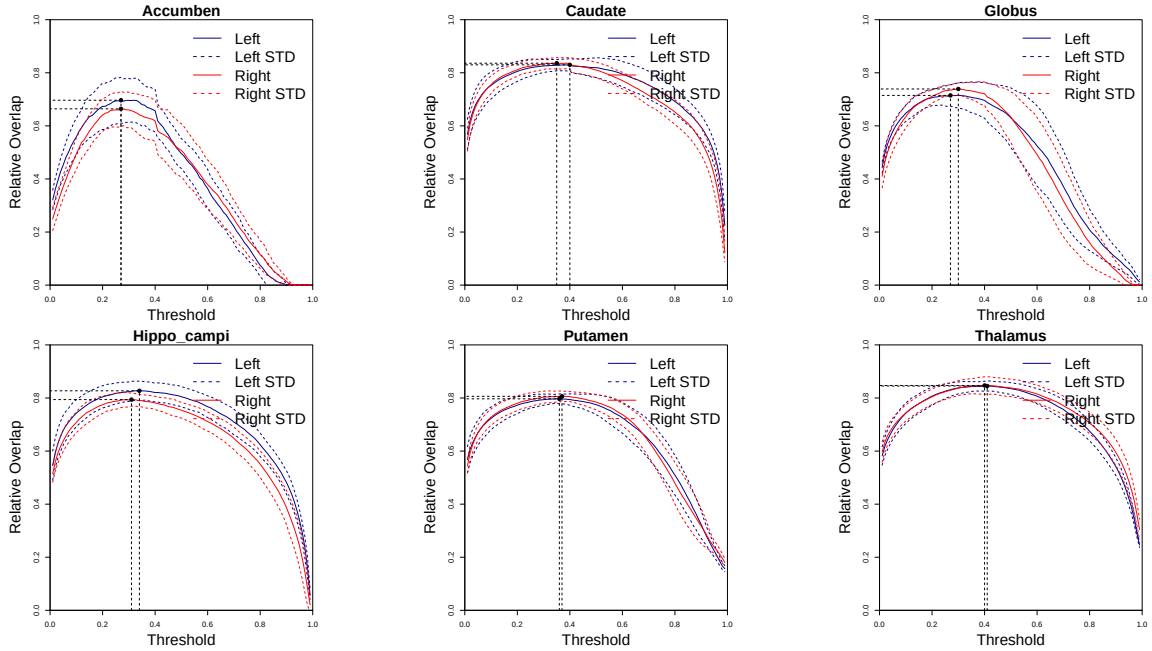


Figure A.21: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 60$

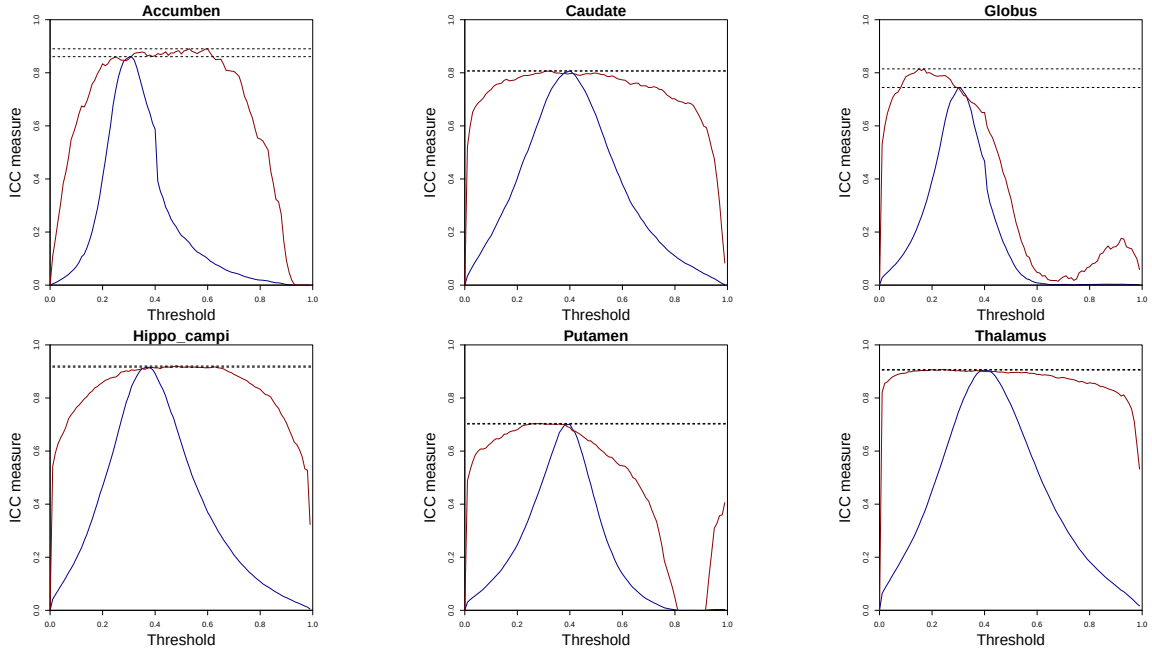


Figure A.22: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 60$

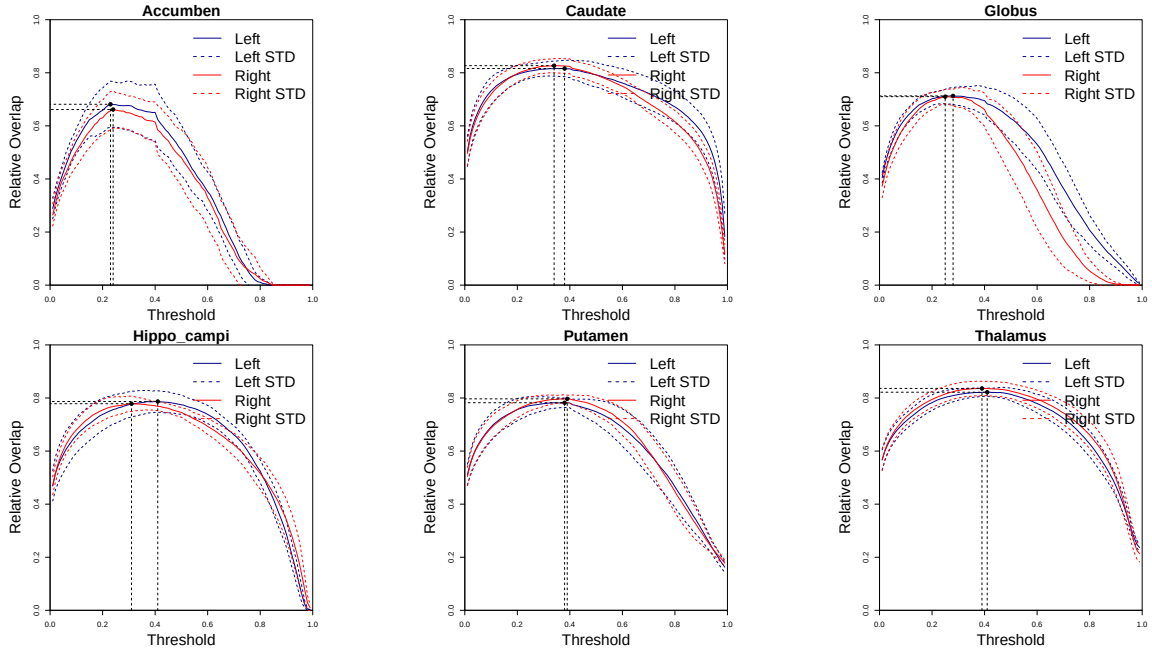


Figure A.23: Relative overlap (RO) measurements versus threshold for $Grad = 1$ $HN = 80$

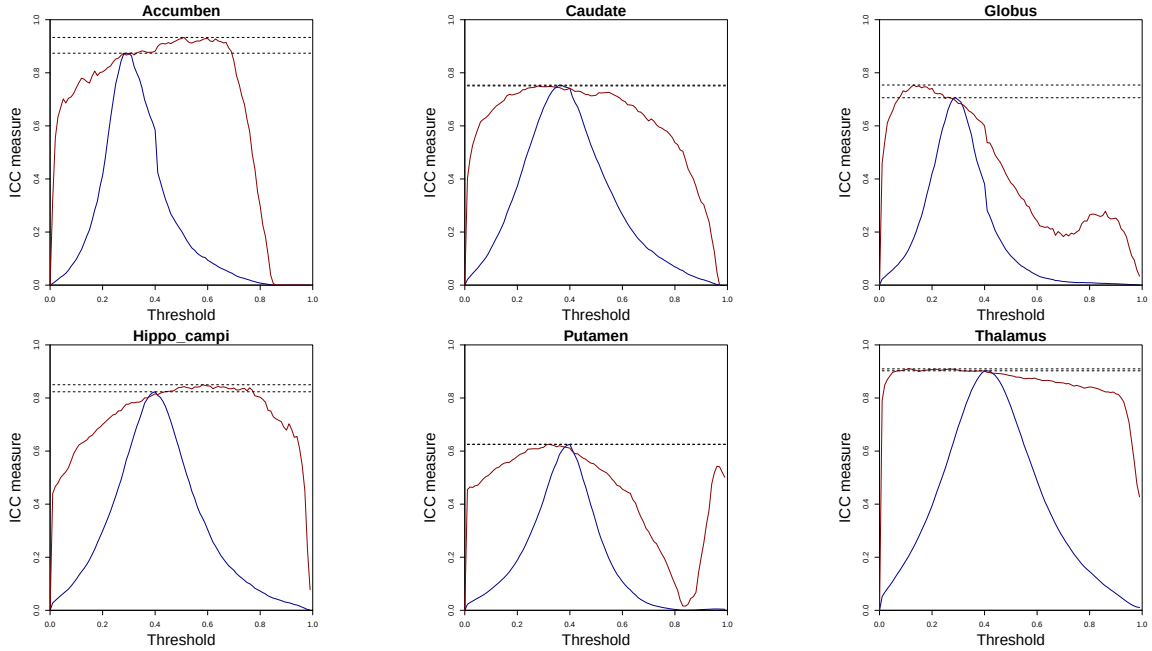


Figure A.24: ICC measeres vs. threshold value for $Grad = 1$ and $HN = 80$

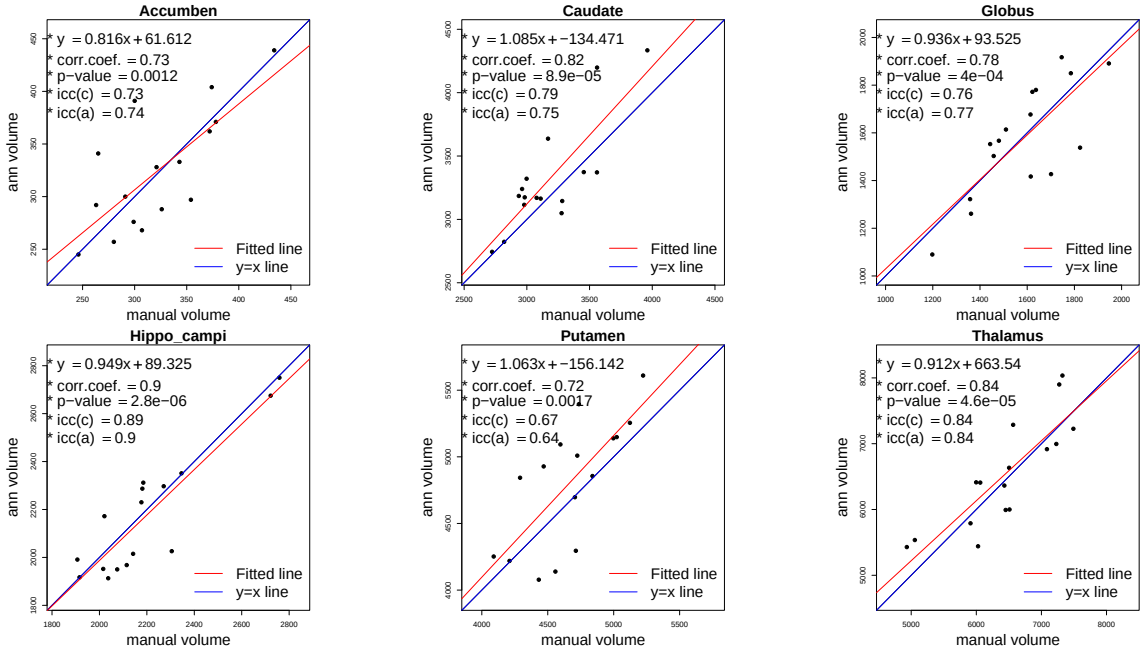


Figure A.25: ICC Corelation graph with reference line (blue) for $Grad = 1$ and $HN = 50$

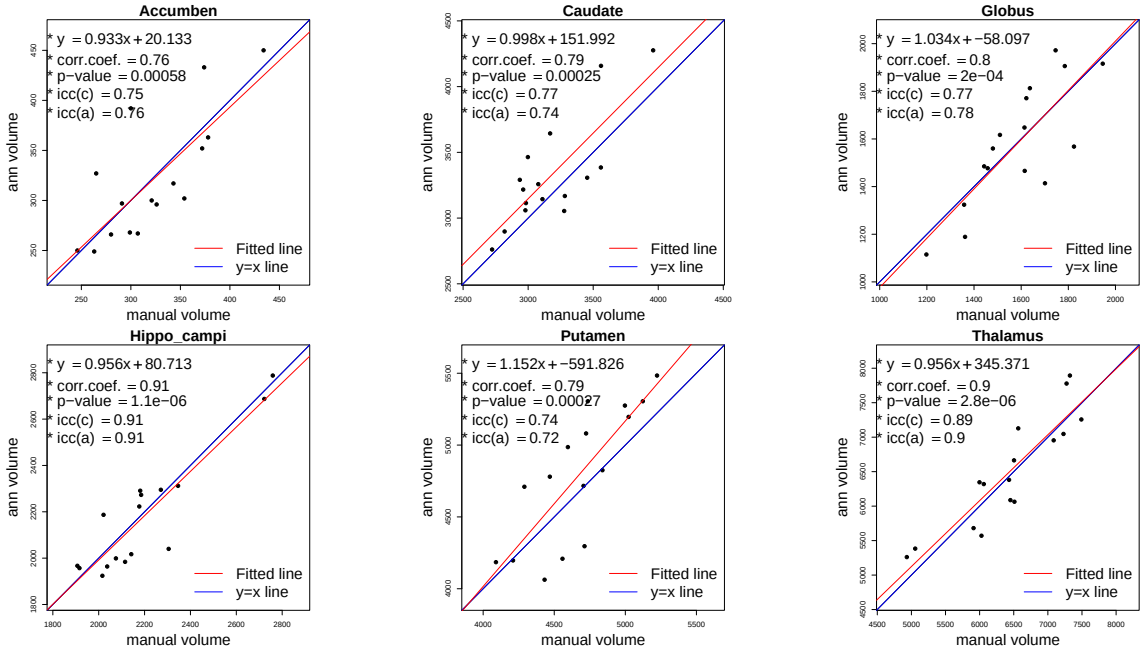


Figure A.26: ICC Corelation graph with reference line (blue) for $Grad = 1$ and $HN = 60$

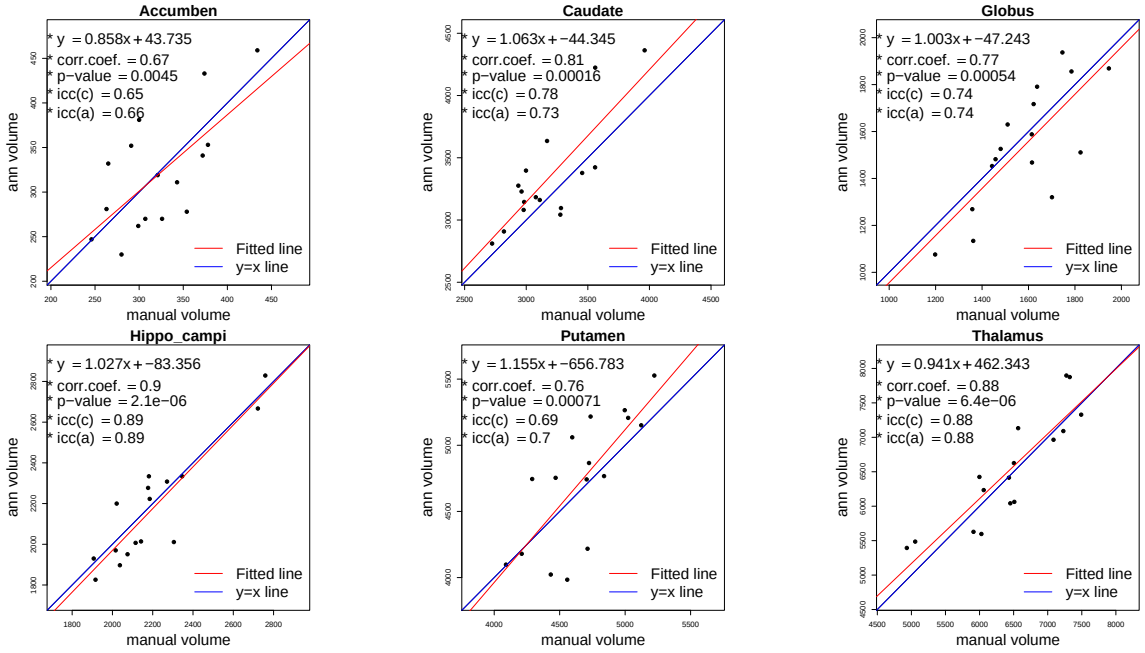


Figure A.27: ICC Corelation graph with reference line (blue) for $Grad = 1$ and $HN = 70$

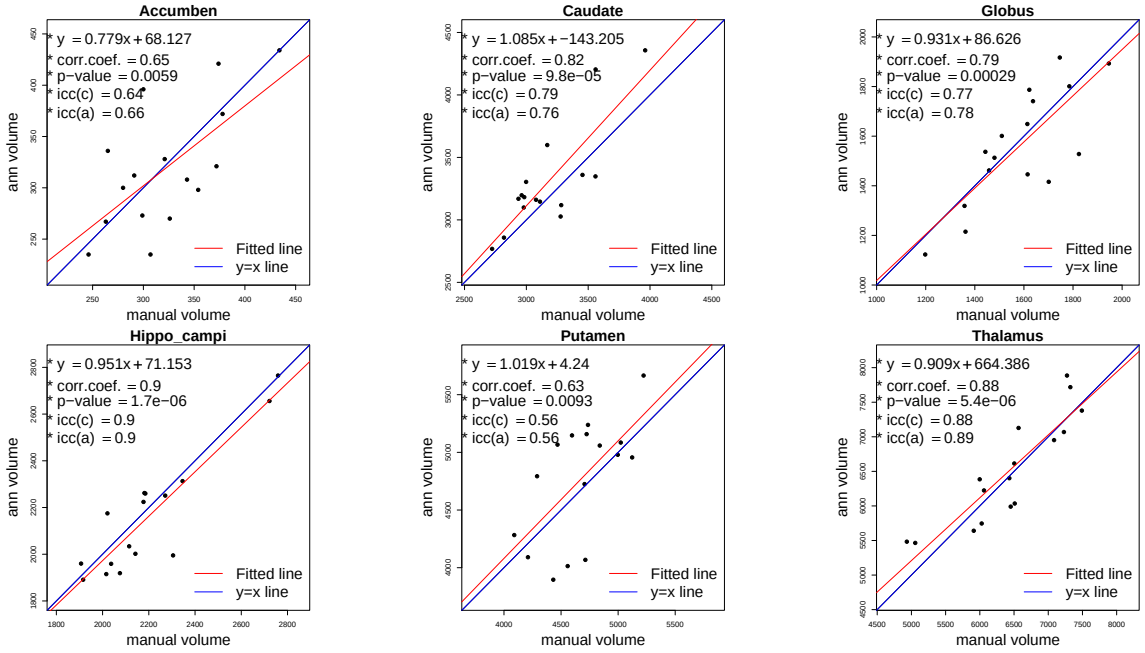


Figure A.28: ICC Corelation graph with reference line (blue) for $Grad = 1$ and $HN = 80$

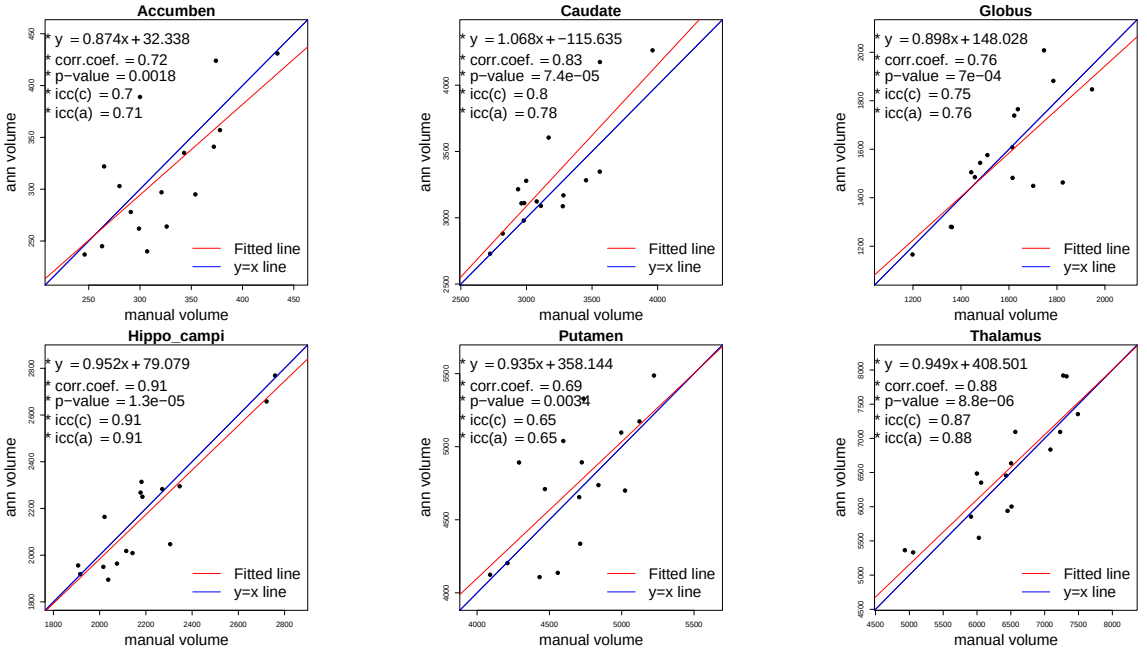


Figure A.29: ICC Corelation graph with reference line (blue) for $Grad = 2$ and $HN = 60$

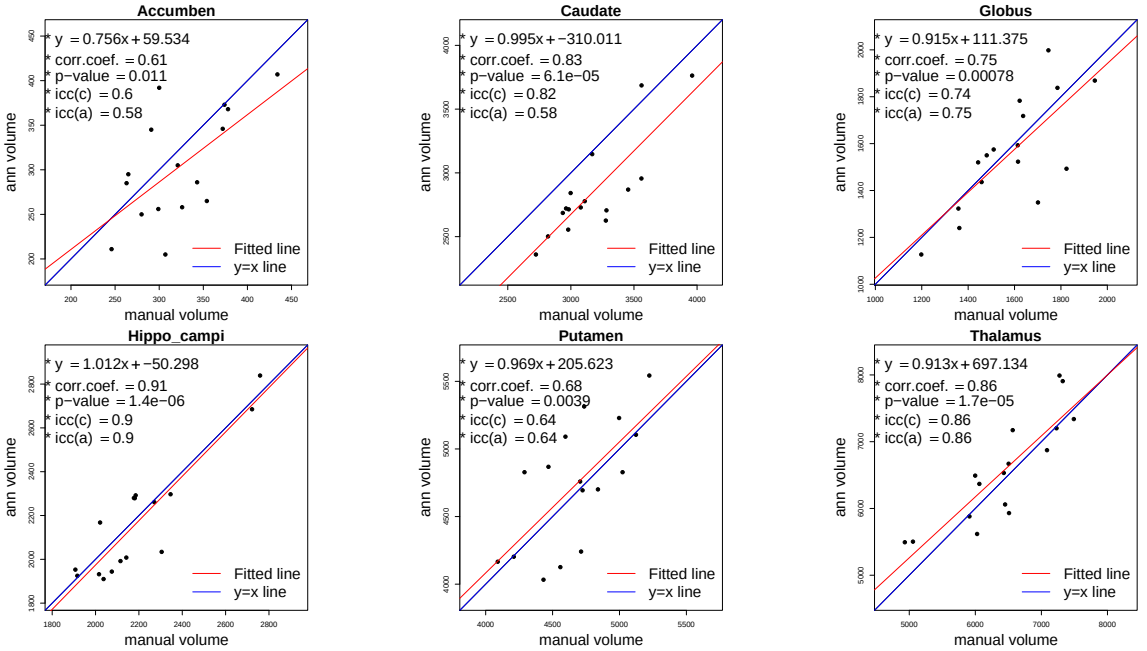


Figure A.30: ICC Corelation graph with reference line (blue) for $Grad = 2$ and $HN = 80$

Table A.1: Simultaneous multi-structure segmentation summary with $HN = 50$

structure	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
accumben	266.62(-17.2%)	54.899(7.9%)	0.709±0.0946	0.7156±0.0877	0.53	0.81	0.82
caudate	3193.8(0.49%)	385.66(19.3%)	0.863±0.0471	0.8646±0.0453	0.73	0.71	0.72
globus	1406.8(-11.0%)	224.24(13.4%)	0.759±0.0784	0.7552±0.0850	0.60	0.46	0.61
hippo campi	2115.9(-3.8%)	285.99(16.2%)	0.853±0.0506	0.8205±0.0566	0.86	0.90	0.91
putmane	4719.0(1.0%)	527.16(62.0%)	0.832±0.0516	0.8454±0.0474	0.64	0.63	0.71
thalmus	6531.2(1.6%)	784.48(3.5%)	0.877±0.0397	0.8756±0.0417	0.90	0.90	0.90

Table A.2: Simultaneous multi-structure segmentation summary with $HN = 60$

structure	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
accumben	267.56(-16.9%)	59.431(16.8%)	0.724 ± 0.097	0.690 ± 0.100	0.59	0.87	0.88
caudate	3181.7(0.1%)	371.08(14.8%)	0.868 ± 0.043	0.872 ± 0.042	0.80	0.80	0.80
globus	1409.3(-10.9%)	179.89(-8.2%)	0.757 ± 0.084	0.779 ± 0.069	0.47	0.65	0.66
hippo campi	2136.1(-2.8%)	270.91(10.1%)	0.860 ± 0.050	0.829 ± 0.053	0.89	0.91	0.91
putamen	4688.8(0.4%)	479.48(47.3%)	0.839 ± 0.050	0.848 ± 0.047	0.70	0.69	0.70
thalamus	6442.5(0.2%)	781.93(3.2%)	0.880 ± 0.039	0.882 ± 0.044	0.90	0.90	0.90

Table A.3: Simultaneous multi-structure segmentation summary with $HN = 80$

structure	*Mean Vol.*	*Std. Vol.*	*RO*	*SI*	*ICC(A)*	*ICC(C)*	*R*
accumben	264.43(-17.9%)	62.01(21.9%)	0.717± 0.1141	0.686± 0.0994	0.58	0.88	0.90
caudate	3114.3(-2.0%)	359.5(11.2%)	0.856± 0.0503	0.863± 0.0466	0.74	0.74	0.74
globus	1338.9(-15.3%)	243.4(24.2%)	0.757± 0.0776	0.727± 0.0866	0.38	0.60	0.81
hippo compi	2176.4(-1.0%)	271.3(10.2%)	0.833 ± 0.0585	0.820± 0.0532	0.83	0.83	0.83
putmane	4693.5(0.5%)	526.9(61.9%)	0.829 ± 0.0516	0.841± 0.0480	0.63	0.61	0.68
thalmus	6508.9(1.3%)	770.5(1.7%)	0.861 ± 0.0433	0.873± 0.0442	0.90	0.90	0.90

APPENDIX B
BRAINSCUT MANUAL

This section is the manual for BRAINSCut, which is implementation of ANN for sub-cortical brain structure segmenatation application. The program is available at *"www.nitric.org"*. The first part explains how to create a model for any population desired. The second part describes how to apply existing model on the large data set. The BRAINSCut application configuration and input are provided in an xml file. The xml file contains all the informaton it needs to complete the task. Here we introduce how to create well defined xml file to run BRAINSCut, and an example command line.

B.1 Create Trained Model

B.1.1 Preparation

Preparation process including choosing research interest, atlas and registration type. The information prepared should go into an xml file, the *AutoSegProcessDescription* xml element. B.1 is an example input file.

```
<AutoSegProcessDescription>
 <!--                    ...  ...                                    -->

   <DataSet Name="33057700" Type="Train">
       <Image Type="T1"    Filename = "[directory]/[filename]" />
       <Image Type="T2"    Filename = "[directory]/[filename]" />
       <Image Type="CL"    Filename = "[directory]/[filename]" />
       <Image Type="SG"    Filename = "[directory]/[filename]" />
       <Image Type="[Image type name]" Filename = "[directory]/[filename]" />
       <!--                    ...  ...                                    -->
       <!--                    as many as you want                         -->

       <Mask Type="left_putamen"       Filename = "[directory]/[filename]" />
       <Mask Type="right_putamen"      Filename = "[directory]/[filename]" />
       <Mask Type="left_accumben"      Filename = "[directory]/[filename]" />
       <Mask Type="[mask type name]"   Filename = "[directory]/[filename]" />
       <!--                    ...  ...                                    -->
       <!--                    as many as you want                         -->
       <Registration SubjToAtlasRegistrationFilename="[directory]/[filename]"
                     AtlasToSubjRegistrationFilename="[directory]/[filename]"
                     ID="CL" />
   </DataSet>
   <!--                ...  ...                                        -->
   <!--                as many as you want                             -->

 <!--                ...  ...                                        -->
```

```
</AutoSegProcessDescription>
```

XML Ex. B.1: XML file example for listing training set. *AutoSegProcessDescription* at the first and the last line indicates the start and end of the documentation

Choose Research Population and representative sample

A representative sample should be chosen from the population of interest for the research. These samples should well cover the population in volume, size, and intensity distributions. After choosing a set of data, divide those into two groups: one is for training and the other is for testing and optimization purposes. Usually one third of the entire sample used for testing and rest of them for training. All the necessary information regarding training data set for each subject should be appeared on *DataSet*. The *DataSet* has two manditory attributes, *Name* and *Type*, and one optional attribute, *OutputDir*. For training data set, the optional attribute *OutputDir* won't be used here so will be explained later. The *DataSet* has three major elements called *Image*, *Mask* and *Registration* with their own attributes as appeared on XML example B.1. *Image* describes the location and type of the image being used for training, and *Mask* refers the binary file for region of interest. Lastly, *Registration* indicates the location the registered result to be stored.

Choose Atlas

Choice of atlas is important in ANN model construction since it is used as spacial standard across any subjects. There can be two possible options in selecting atlas. One is that the brains which has average in size, volume and shape, can be used as atlas. In this case, this subject will be excluded for training process. The other choice is to use the synthesized brain by using BrainWeb [5] The declaration of atlas in the xml file appleared on anywhere between *AutoSegProcessDescription* but disjointly with any other element. Naturally, in contrast to declaring any number of *Training* data set, we can have only one number of *Atlas* type *DataSet*.

```
<AutoSegProcessDescription>
```

```
<!--                              ... ...                                    -->
<DataSet Name="[anyname]" Type="Atlas">
        <Image Type="T1"                    Filename = "[directory]/[filename]" />
        <Image Type="T2"                    Filename = "[directory]/[filename]" />
        <Image Type="CL"                    Filename = "[directory]/[filename]" />
        <Image Type="SG"                    Filename = "[directory]/[filename]" />
        <Image Type="[Image type name]"     Filename = "[directory]/[filename]" />
</DataSet>

<!--                              ... ...                                    -->
</AutoSegProcessDescription>
```

XML Ex. B.2: Atlas image file declaration example.

Choose Registration

The choice of registration is very important in image processing, as comparative analysis requires images to be as closely aligned as possible. Image Registration is a big subject for image processing research in and of itself. BRAINSCut application uses script file to do any type of registration so that it can be flexible to variety type of registration. Any registration out of BRAINS package can be used, and also few well defined registration scripts are provided with BRAINSCut program. Registration is also declared in the xml file with *Type*,*Commands*, *ImageType* and *ID*. The *ImageType* indicates that the what kind of image will be used for registration out of several image listed for data set, and *ID* labels is for registration specifically in case of we have more than one declared registration in one file. This *ID* is identified at the *Registration*'s *ID* of the *Training* data set.

```
<AutoSegProcessDescription>
<!--                              ... ...                                    -->
<RegistrationParams Type        = "ThirionDemons"
                    Command     = "[directory]/[filename]"
                    ImageType   = "CL"
                    ID          = "CL" />

<RegistrationParams Type        = "[RegistrationType]"
                    Command     = "[directory]/[filename]"
                    ImageType   = "[ImageType]"
                    ID          = "[Registration ID]" />
<!--                  as many as you want                                    -->
<!--                              ... ...                                    -->
</AutoSegProcessDescription>
```

XML Ex. B.3: Registration declaration example.

B.1.2 Generate Probability Map

In addition to those data set declarations, we have to specify region of interest for segmentation in the *ProbabilityMap*. An xample file appears in the XML example B.4. The *ProbabilityMap* has 7 attributes; *StructureID*, *Gaussian*, *GenerateVector*, *Filename*, *rho*, *phi*, and *theta*. Each region of interest has to have its own *ProbabilityMap* with distintive *structureID*, and this *structureID* should correspond to the binary file declared under the *DataSet* as *Mask*. The smoothness degree is defined for a *Gaussain* with any positive number. When the *GeneratingVector* term is true, the BRAINSCut application will generate the vector using this probability map. Otherwise, it will not generate training vectors for the probability maps but will be referenced to determine candidates structures while neighboring structures being processed. The *Filename* gives name for the probability map including location, and the modified version of *rho*,*phi*, and *theta* defined here.

```
<AutoSegProcessDescription>
  <!--                    ... ...                                    -->

  <ProbabilityMap StructureID    = "left_caudate"
                  Gaussian       = "1.0"
                  GenerateVector = "true"
                  Filename       = "[directory]/[filename]"
                  rho            = "[directory]/[filename]"
                  phi            = "[directory]/[filename]"
                  theta          = "[directory]/[filename]"
                  />
  <ProbabilityMap StructureID    = "right_caudate"
                  Gaussian       = "1.0"
                  GenerateVector = "true"
                  Filename       = "[directory]/[filename]"
                  rho            = "[directory]/[filename]"
                  phi            = "[directory]/[filename]"
                  theta          = "[directory]/[filename]"
                  />

  <ProbabilityMap StructureID    = "[mask_type_name]"
                  Gaussian       = "[any_positive_number]"
                  GenerateVector = "[true|false]"
                  Filename       = "[directory]/[filename]"
                  rho            = "[directory]/[filename]"
                  phi            = "[directory]/[filename]"
                  theta          = "[directory]/[filename]"
                  />
  <!--                  as many as you want                          -->
  <!--                  ... ...                                      -->
</AutoSegProcessDescription>
```

XML Ex. B.4: Probability map declaration example file.

Now we are ready to run very first command to create spatial probability by running following command

```
BRAINSCut --generateProbability --netConfiguration [xmlfilename]
```

This command takes while if no registration result presented.

After creating probablity maps for all of region of interest, the next step is to create training vectors. In addition to the data set declaration, we now need the parameters for ANN training. This can be varied by cases, we introduces here most generally applicable parameters.

```
<AutoSegProcessDescription>
  <!--                    ... ...                                    -->
  <ANNParams
            Iterations              = "400"
            MaximumVectorsPerEpoch  = "700000"
            EpochIterations         = "100"
            ErrorInterval           = "1"
            DesiredError            = "0.000001"
            NumberOfHiddenNodes     = "60"
  />

  <NeuralNetParams MaskSmoothingValue  = "0.0"
                   GradientProfileSize = "1"
                   TrainingFilename    = "[directory]/[filename]"
                   Filename            = "[directory]/[filename]"
                   TestVectorFilename  = "[directory]/[filename]"
  />

  <ApplyModel CutOutThresh   = "0.05"
              CutOutGaussian = "0"
              MaskThresh     = "0.33"
              DefDir         = "[directory]/[filename]"
              OutputDir      = "[directory]/[filename]"
  />
  <!--                    ... ...                                    -->

</AutoSegProcessDescription>
```

XML Ex. B.5: Training Parameter setting xml file example.

To create the training vector, the *ANNParams* and *NeuralNetParams* elements are used. The *ApplyModel* element will not be used until we get to the optimization, or applyModel steps, so this will be explained later part of the manual. The number of epoch Iteration between error report is given by *EpochIterations*, and so total training iteration will be *Iterations* times *EpochIterations*. The *MaximumVectorPerEpoch* depends on machine, which how much memory size is available for training. It is better to read in the whole training vector at once if enough memory is available. To

generate an overfitted model to determine an optimally trained ANN model, use a small number for *DesiredError*. We are using multi-layered neural network with one level of hidden node. The *NumberOfHiddenNodes*, in that sense, decides how many nodes for the hidden nodes for training. Many literature says it is beneficial to have as many as hidden nodes possible, but practically it takes long time to finish training with large number of hidden nodes. In our experience in training ANN models for sub-cortical brain structures, a model with 60 hidden nodes behaves well for most problems. If this is not working, start from increasing and decreasing the number of hidden nodes by 10. In this way, the model gives some idea what number would fit for given problem.

```
BRAINSCut --createVectors --netConfiguration [xmlfilename]
```

Above commend line will do the job.

B.1.3 Training and Optimization of Model

With those selected parameters and data set, now we can proceed to the training and optimization of ANN model. Because the training of ANN is the process of computing weights for each nodes, the time depends on the number of nodes exist in the model. The following line will start and finish training as instructed in the xml file.

```
BRAINSCut --trainModel --netConfiguration [xmlfilename]
```

After overfitted trained model is obtained by running the above command, the optimally trained point based on train error and preformance error should be searched. To get the performance error, the performance vector set is created using test data set. With the performance vector file, the performance error can be obtained by running following command.

```
BRAINSCut --trainModel --netConfiguration [performancexmlfilename]
    --doTest
```

This command line is alomost identical except for *–doTest*, which will apply the testing vector file to the created model and then compute mean square error with

known output vector. With *–doTest* option, the model will not change their nodes' weights but just compute performance error. For computing performance error with *–doTest* option, the performance vector file should be indicated at the *TestVector-Filename* instead of *Filename* in the *NeuralNetParam* on the XML example B.5 By plotting train and performance error, we can find the optimally trained point. For further explanation on optimization, please see 3.6.

B.2 Apply on Existing Model

Apply the existing ANN model onto new data set is simple process with similar xml file.

```
<AutoSegProcessDescription>
   <DataSet Name="33057700" Type="Apply" OutputDir="[directory]">
      <Image Type="[Image type name]" Filename="[directory]/[filename]" />
      <!--                    ... ...                                    -->
      <!--            same type of images used in the training          -->

      <Mask Type="[mask type name]" Filename="[directory]/[filename]" />
      <!--                    ... ...                                    -->
      <!--             same type of masks used in the training           -->
      <Registration SubjToAtlasRegistrationFilename="[directory]/[filename]"
                    AtlasToSubjRegistrationFilename="[directory]/[filename]"
                    ID="CL" />
      <!--          same type of registration used in the training       -->
   </DataSet>
   <!--                   ... ...                                  -->
   <!--                   as many as you want                      -->
   <DataSet Name="template" Type="Atlas">
        <!--          same type of registration used in the training      -->
        <!--                    ... ...                                   -->
   </DataSet>

   <RegistrationParams Type="ThirionDemons"          ... ...              />
   <!--          same type of registration used in the training           -->
   <ProbabilityMap StructureID="right_accumben"   ... ...                 />
   <!--          same probability maps used in the training               -->

   <ANNParams
            ... ...
              Iterations="[optimal_point]"
            ... ...
   />
            <!--          same  setting but Iteration numbers              -->

   <NeuralNetParams MaskSmoothingValue="0.0"       ... ...                />
   <!--          same setting used in the training                  -->


    <ApplyModel CutOutThresh="0.05"                ... ...                />
    <!--          same setting used in the training                  -->

</AutoSegProcessDescription>
```

XML Ex. B.6: Training Parameter setting xml file example.

The data set to apply were given as the *DataSet* labeld as *apply* and the desired output directory *OutputDir*. After assign data set, which trained model applied on, every other setting should be same but the *Iterations*, which has to correspond value of the optimal point above.

```
BRAINSCut --applyModel --netConfiguration [xmlfilename]
```

As usual, applying ANN model onto the new data set with well-formed xml file can be performed by above command line.

Choice of optimal threshold

The result file from the ANN model for region of interest is not a binary file but probability map, which have values between zero and one. Since most of cases we want to have one binary file for specific region of interest, the proper threshold value is necessary. Finding proper threshold values can be different for different research interests; if desired result is consistency to the reference binary image, the *ICC* consistency can be measured between reference and ANN results binary images. Then the threshold value which has the highest *ICC* consistency value on average can be used for overall threshold value. If the ANN result has to agree to the reference binary images, the ICC agreement or Relative Overlap measure can be used. Those numbers will give the point with the most agreeable threshold value.

REFERENCES

[1] Mohamed N Ahmed and Aly A Farag. Two-stage neural network for volume segmentation of medical images. Nov 1997.

[2] J Alirezaie, M Jernigan, and C Nahmias. Neural network-based segmentation of magnetic resonance images of the brain. *Nuclear Science, IEEE Transactions on*, 44(2):194 – 198, Apr 1997.

[3] F Beacher, E Daly, A Simmons, V Prasher, R Morris, C Robinson, S Lovestone, K Murphy, and D G M Murphy. Alzheimer's disease and down's syndrome: an in vivo mri study. *Psychological medicine*, 39(4):675–84, Apr 2009.

[4] J Christensen. Normalization of brain magnetic resonance images using histogram even-order derivative analysis. *Magnetic Resonance Imaging*, Jan 2003.

[5] C Cocosco, V Kollokian, R Kwan, and A Evans. Brainweb: Online interface to a 3d mri simulated brain database. *NeuroImage*, Jan 1997.

[6] J Coyle, D Price, and M DeLong. Alzheimer's disease: a disorder of cortical cholinergic innervation. *Science*, 219(4589):1184Ĩ190, Mar 1983.

[7] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control*, Jan 1989.

[8] Tim B Dyrby, Egill Rostrup, William F C Baaré, Elisabeth C W van Straaten, Frederik Barkhof, Hugo Vrenken, Stefan Ropele, Reinhold Schmidt, Timo Erkinjuntti, Lars-Olof Wahlund, Leonardo Pantoni, Domenico Inzitari, Olaf B Paulson, Lars Kai Hansen, Gunhild Waldemar, and LADIS study group. Segmentation of age-related white matter changes in a clinical multi-center study. *NeuroImage*, 41(2):335–45, Jun 2008.

[9] J Feldman and D Ballard. Connectionist models and their properties. *Cognitive Science*, Jan 1982.

[10] R. Rojas J. Feldman. *Neural networks: a systematic introduction.* Springer, 1996.

[11] G Harris, N Andreasen, T Cizadlo, J Bailey, J H Bockholt, and V Magnotta. Improving tissue classification in mri: A three-dimensional multispectral discriminant analysis .... *Journal of Computer Assisted Tomography*, Jan 1999.

[12] G Hinton. Connectionist learning procedures. *Artificial intelligence*, Jan 1989.

[13] K Hornik, M Stinchcombe, and H White. Multilayer feedforward networks are universal approximators. *Neural Networks*, Jan 1989.

[14] A Kostopoulos, D Glotsos, and P Spyridonos. ... of feedforward and probabilistic neural networks for the automatic classification of brain .... *Proc 1st Int. Conf. from Scientific Computing to ....*

[15] K VAN LEEMPUT, F MAES, D VANDERMEULEN, and P SUETENS. Automated model-based bias field correction of mr images of the brain. *IEEE transactions on medical imaging*, Jan 1999.

[16] K Van Leemput, F Maes, D Vandermeulen, and P Suetens. Automated model-based tissue classification of mr images of the brain. *Medical Imaging, IEEE Transactions on*, 18(10):897 – 908, Oct 1999.

[17] Karen Chia-Ren Lin, Miin-Shen Yang, Hsiu-Chih Liu, Jiing-Feng Lirng, and Pei-Ning Wang. Generalized kohonen's competitive learning algorithms for ophthalmological mr image segmentation. *Magnetic Resonance Imaging*, 21(8):863–70, Oct 2003.

[18] Y Liu, B Li, D Elliman, P Morgan, and D Auer. Automatic segmentation of putamen from brain mri. *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 2006.

[19] W McCulloch and W Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, Jan 1943.

[20] K McGraw and S Wong. Forming inferences about some intraclass correlation coefficients. *Psychological methods*, Jan 1996.

[21] Ian Middleton and Robert I Damper. Segmentation of magnetic resonance images using a combination of neural networks and active contour models. *Medical engineering & physics*, 26(1):71–86, Jan 2004.

[22] M Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, Jan 1991.

[23] M Moghaddam and H Soltanian-Zadeh. Automatic segmentation of brain structures using geometric moment invariants and .... *Proceedings of the 21st International Conference on ...*, Jan 2009.

[24] M Morrison and Y Attikiouzel. A probabilistic neural network based image segmentation network formagnetic resonance .... *Neural Networks*, Jan 1992.

[25] P Nopoulos, V Magnotta, A Mikos, and H Paulson. Morphology of the cerebral cortex in preclinical huntington's disease. *American Journal of Psychiatry*, Jan 2007.

[26] S Powell, V Magnotta, H Johnson, and V Jammalamadaka .... Registration and machine learning-based automated segmentation of subcortical and cerebellar brain .... *NeuroImage*, Jan 2008.

[27] W Reddick, J Glass, E Cook, T Elkin, and R Deaton. Automated segmentation and classification of multispectral magnetic resonance images of brain using artificial neural networks. *Medical Imaging, IEEE Transactions on*, 16(6):911 – 918, Dec 1997.

[28] W E Reddick, R K Mulhern, T D Elkin, J O Glass, T E Merchant, and J W Langston. A hybrid neural network analysis of subtle brain volume differences in children surviving brain tumors. *Magnetic Resonance Imaging*, 16(4):413–21, May 1998.

[29] Elain Rich and Kevin Knight. *Artificial Intelligence, 2nd ed.* Mcgraw Hill, 1991.

[30] M Riedmiller and H Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. *Neural Networks, 1993., IEEE International Conference on*, pages 586 – 591 vol.1, Jan 1993.

[31] H Rusinek, MJ de Leon, AE George, LA Stylopoulos, R Chandra, G Smith, T Rand, M Mourino, and H Kowalsk. Alzheimer disease: measuring loss of cerebral gray matter with mr imaging. *Radiology*, 178:109˜114, 1991.

[32] P.E Fleiss J.L Shrout. Intraclass correlation: Uses in assessing rater reliability. *Psycology Bulletin*, 86:420–428, Sep 1979.

[33] J Sled and G Pike. Understanding intensity non-uniformity in mri. *LECTURE NOTES IN COMPUTER SCIENCE*, Jan 1998.

[34] Tao Song, M Jamshidi, R Lee, and Mingxiong Huang;. A modified probabilistic neural network for partial volume segmentation in brain mr image. *Neural Networks, IEEE Transactions on*, 18(5):1424 – 1432, Sep 2007.

[35] Eric P. Widmaier Hershel Raff Kevin T. Strang. *Vander's Human Physiology the mechanism of body function.* McGraq-Hill Higher Education, 2008.

[36] G Székely and G Gerig. Model-based segmentation of radiological images. *KI*, Jan 2000.

[37] Yue Wang, T Adali, M Freedman, and S Mun. Mr brain image analysis by distribution learning and relaxation labeling. *Biomedical Engineering Conference, 1996., Proceedings of the 1996 Fifteenth Southern*, pages 133 – 136, Feb 1996.

[38] Dr. Peter J. Whitehouse, MD, PhD 1 *, Donald L. Price, MD 1 2, Arthur W. Clark, MD 1 2, Joseph T. Coyle, MD 3, Mahlon R. DeLong, and MD. Alzheimer disease: Evidence for selective loss of cholinergic neurons in the nucleus basalis. *American Neurology Association*, Dec 1981.