August 2022

Ph.D. Dissertation

Classification of Imaging Modalities with Alzheimer's Disease using Modified Parametric Layers in 3D CNN

Graduate School of Chosun University

Department of Information and Communication Engineering

Bijen Khagi

Classification of Imaging Modalities with Alzheimer's Disease using Modified Parametric Layers in 3D CNN

3D CNN에서 수정된 매개변수 레이어를 이용한 알츠하이머병의 영상 모달리티 분류

August 26, 2022

Graduate School of Chosun University

Department of Information and Communication Engineering

Bijen Khagi

Classification of Imaging Modalities with Alzheimer's Disease using Modified Parametric Layers in 3D CNN

Advisor: Prof. Goo-Rak Kwon

A dissertation submitted to the Graduate School of Chosun University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

April 2022

Graduate School of Chosun University

Department of Information and Communication Engineering

Bijen Khagi

This is to certify that the Doctoral degree dissertation of

Bijen Khagi

has been approved by examining committee for the thesis requirement for the Doctoral degree in Engineering.

Committee Chair : Prof. Jae-Young Pyun (Chosun University)

Committee Member: Prof. Young-Sik Kim (Chosun University)

Committee Member: Prof. Chanki Kim (Chosun University)

Committee Member: Prof. Inkyu Moon (DGIST University)

Committee Member: Prof. Goo-Rak Kwon (Chosun University)

June 2022

Graduate School of Chosun University

Table of Contents

List of Figures

List of Tables

Abstract (초 록)

3D CNN에서 수정된 매개변수 레이어를 이용한 알츠하이머병의 영상 모달리티 분류

비젠 카기

지도교수: 권구락

조선대학교 정보통신공학과

합성곱신경망(CNN)은 MRI 이미지와 피상적으로 작용하여 환자의 의학적 상태와 관련될 수 있는 영상 특징을 학습한다. 이에 따라, 더 높은 정확도와 과적합 문제를 해결하기 위해 알츠하이머의 영향을 받는 MRI의 분류에 CNN을 사용하려고 시도했다. CNN은 MRI 분류를 위해 특별히 설계된 몇 가지 새로운 매개변수 레이어와 함께 사용한다. 초기에 'divNet'이라는 아키텍처는 증가되는 필터 크기와 깊이에 따라 넓은 범위로 발산되는 수신 영역을 제안한다는 아이디어로 개발되었다. 이는 차례로 기능이 감소된 낮은 수준에서 높은 수준의 특징 추출 프로세스를 진행하며 중복된 특징을 낮춘다. 이 아키텍처는 정확한 최종 결과를 위해 일부 다른 기본 아키텍처 및 가변 하이퍼 파라미터와 비교한다. 또한, 데이터 크기 효과 및 데이터 유형(즉, MRI 또는 PET)도 이 아키텍처를 사용하여 분석한다. CNN 분류에서 베이스라인 아키텍처는 레이어 간 연구수행에서 압도적인 결과를 얻었다. 이에 CNN의 초기 레이어는 낮은 수준의 특징 추출에 관여한다는 사실을 알 수 있다. 이러한 프로세스는 정규화 기술에 크게

의존한다. 따라서 정규화 프로세스를 연구하고 학습이 용이한 고유한 정규화 계층을 제안한다. 이를 위해 기존의 일괄된 정규화 계층을 대체하기 위해 CNN을 위한 새로운 GAP(Gaussian Activated Parametric) 계층을 제안한다. 제안된 방법의 목표는 심층 CNN의 초기 및 중간 특징 레이어를 정규화하고 활성화하여 맞춤형 학습이 가능한 매개변수 레이어를 사용하여 특징을 구별할 수 있도록 한다. 이후 계층은 대상 도메인의 분류를 위해 조정한다. 기존의 GAP레이어는 MRI의 특징 벡터 정규화를 위해 설계되었다. 그러나 CIFAR-10, Caltech-256, 5-animals dataset 와 같은 자연적인 영상 데이터 셋에서 테스트했을 때에 몇 가지 경우 유사하거나 약 개선된 결과가 관찰되었다. 정규화 기술에서 얻은 몇 가지 이해를 바탕으로, 매개변수 계층을 활성화 함수로 사용하고 기준 모델의 ReLU 활성화를 대체하는 것으로 목표를 변경했다. 이를 위해 SGT 활성화라고 하는 스케일 감마 보정과 쌍곡선 탄젠트 함수의 조합을 기반으로 하는 새로운 활성화 함수를 제안한다. 제안된 SGT 활성화 함수는 ReLU, Leaky-ReLU 및 tanh와 같은 다른 활성화 함수와 비교하여 분석한다. 또한 기울기의 소실/폭주 문제에 대처하는 역할로 분석된다. 이전 연구와 유사하게 모든 결과는 히스토그램 분석, 가중치/편향 상관 분석 및 T-SNE 객관화로 내용을 보완한다.

이와 같은 방법으로 CNN 아키텍처 설계에서 단일 레이어 자체 설계를 진행한다. 이를 위해 레이어의 미세 작업을 이해하고 더 나은 결과를 위해 조정한다. 수행된 작업은 독립형 MRI 분류이지만 3D CNN을 사용하여 기본 분류 작업 내에서 미세 조작을 자세히 연구를 진행한 것이 좋은 결과를 얻었다. pooling layer, flattening layer, convolutional layer와 같이 아직 연구할 수 있는 레이어가 많기 때문에 더 많은 레이어를 사용자 정의 및 향상된 방식으로 풀 수 있다.

Abstract (English)

Classification of Imaging Modalities with Alzheimer's disease using Modified Parametric Layers in 3D CNN

Bijen Khagi

Advisor: Prof. Goo Rak-Kwon, Ph.D.

Dept. Info. and Comm. Engineering,

Graduate School of Chosun University

A Convolutional neural network (CNN) works superficially with magnetic resonance image (MRI) to learn its image-attributes, which may be correlated with the medical condition of the patient. This thesis work is an attempt to utilize CNN for the classification of Alzheimer's affected MRI to achieve higher accuracy and lesser overfitting issue. For which CNN was employed along with some novel parametric layers that were designed specifically for MRI categorization. Initially, a baseline architecture called 'divNet' was developed with the main idea of presenting diverging reception area by increasing the filter size and stride along with depth. This helped from a low level to a high-level feature extraction process with reduced feature redundancy. This architecture was compared with some other basic architectures and variable hyperparameters for the final accuracy result. Meanwhile, the effects of data size and datatype (i.e., MRI or PET) were also analyzed using this architecture. With the overwhelming results from this baseline architecture in CNN classification, the layer-to-layer study was performed. Later, it was noticed that the early layers in CNN were responsible for low-level feature extraction. These processes were heavily dependent on the normalization technique. Hence the research was shifted to study the normalization process and

propose a unique normalization layer with ease of training. For this, a novel Gaussian activated parametric (GAP) layer specifically for CNN to replace the traditional batch normalization layer was proposed. The goal of the proposed method was to normalize and activate the initial and intermediate feature layers of a deep CNN so that a customized learnable parametric layer can make the feature more distinguish. Later the layers were smoothly tuned for the target-domain classification. Originally the GAP layer was designed for MRI features vector normalization. However, when tested in natural image datasets like CIFAR-10, Caltech-256, and 5-animals dataset, similar or slightly improved results was observed in a few cases. With some insights from the normalization technique, the new concern was to use a parametric layer as an activation function and replace the traditional ReLU like activation layers from the baseline model. For this, a novel activation function was proposed based on the combination of scaled gamma correction and hyperbolic tangent function, named Scaled Gamma Tanh (SGT) activation. The behavior of the proposed SGT activation function was analyzed against other popular activation functions like ReLU, Leaky-ReLU, and tanh. Additionally, their role to confront vanishing/exploding gradient problems was analyzed. Similar to the previous studies, all of the findings were supported by histogram analysis, weights/bias correlation analysis, and T-SNE projection.

In this way, the research commenced from designing a CNN architecture till designing a single layer itself, so that micro-operation in layers can be understood and tweaked for better results. Though the performed task is a standalone MRI classification, with 3D CNN, it was beneficial to minutely study the micro-operation within the fundamental classification task. Since still there are many more layers to be studied like the pooling layer, flattening layer, and convolutional layer itself, many layers can be customized and unraveled in better ways. Considering deep neural network, a black box to uncover, this thesis might provide some insight and enthusiasm for those interested to study CNN working mechanism step by step.

Abbreviation

AD	Alzheimer’s Disease
AE	Auto Encoder
CNN	Convolutional Neural Networks
MCI	Mild Cognitive Impairment
NC	Normal Controls
CN	Controlled Normal
SVM	Support Vector Machine
ADNI	Alzheimer’s Disease Neuroimaging Initiative
BN	Batch Normalization
GAP	Gaussian Activated Parametric
SGT	Scaled Gamma Tanh
FCL	Fully Connected Layer
DNN	Deep Neural Network
CAD	Computer-Aided Diagnosis
t-SNE	t-Distributed Stochastic Neighbor Embedding
SGD	Stochastic Gradient Descent
ADAM	Adaptive Moment Estimation
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography

CHAPTER 1

Introduction

"Emotions are enmeshed in the neural networks of reason."

-Antonio Damasio

1.1 Introduction

Depending upon the method of acquisition of images, different types of imaging modalities can be used to visualize the physical and physiological condition of the human body. And the ones used for the brain are often referred to as neuroimaging modalities. The most popular ones being MRI, computed tomography (CT scan), and PET. MRI is broadly classified as functional MRI (fMRI), for metabolic function activity visualization, and structural MRI (sMRI), for anatomical structure visualization. Additionally, in sMRI, the most common one is T1 where only FAT tissues are bright whereas in T2 both FAT and water are isotropically bright. CT Scan and X-Ray being an ionizing radiation-based methods, are not often preferred for the brain. In the case of neurodegenerative disease like Alzheimer's there is a reduction in the volume of grey matter and shrinkage in the Hippocampus area visualized in MRI [1] whereas some other measurement techniques like quantitative susceptibility mapping (QSM) measure the change in iron, myelin, and calcium in vivo in brain-related dementia diseases. Similarly, brain FDG-PET and amyloid PET records the pattern of glucose metabolism and amyloid deposition respectively. Therefore, based on the imaging modalities, different metrics are used to assess patients with Alzheimer's disease (AD). This helps to measure the differences between normal and pathogenic cases and filter out the unusual changes, which may eventually be the topic of interest for medical study too.

AD is a pathological condition of dementia characterized by memory impairment and cognitive dysfunction of the brain. The microscopic cause for AD takes place in the brain nerve cell connection area called the synapsis, where the neurotransmitters are released [2]. The synapsis helps with the information flow caused by tiny bursts of chemicals that are released by one neuron and detected by a receiving neuron. During AD, there is an accumulation of ß-amyloid proteins, suspected to cause neurons death, and tau proteins, also known as tau tangles, block the supply of nutrients and other essential molecules inside the neurons and the synaptic region. With this outcome, there is a physical change in the common AD-related variation of anatomical brain structures such as the enlargement of ventricles, shrinkage of the hippocampus shape, change in the cortical thickness, and other cerebral areas containing white matter and gray matter brain tissue as well as cerebrospinal fluid. These changes and atrophies are rationally visualized through brain imaging by the clinicians using various imaging modalities like MRI or PET or CT scans. Then comes the use of image processing and machine learning (ML) techniques.

ML algorithms in image processing, assist to find a discriminative pattern of image features by collecting the same groups of images into one. It means that the patterns that are eventually discovered for AD patients will behave the same for other AD patients' recognition although it is differentiated with the CN and MCI-affected MRI. Once the MRI is translated into an image from the magnetic field gradients, it represents the pixel value for the participant's brain image. Ultimately, AD classification will be based on the features that are extracted from these brain image pixels. The main features required to accurately capture the major AD-related variations of the anatomical brain structure include the size of the ventricles, hippocampus shape, cortical thickness, and brain volume [3]. Although such alterations may resemble other brain-related diseases like Parkinson's disease (PD) and encephalitis [4]. In that case, more clinical and physiological tests should be performed on a clinical level.

Classification, as a part of ML, is the process of categorizing an image, or an image attribute, or any input vectors into some output target variable/label used as ground truth during training. That means a model can only categorize an input to the corresponding labels used in training. Hence CNN as a classifier simply categorizes the input vector into a target label, unlike the classical regression problem, the output has a fixed discrete value. This idea is simple but crucial for the data analysis process, in fact, other ML tasks like segmentation, object detection, ROI detection, etc. are just complex forms of classification. Convolutional neural network (CNN) is an advanced ML idea, which has been the dark horse in the field of deep learning (DL) since the success of LeNet-5, an emerging CNN in the late '90s for handwriting recognition [5][6]. The massive success behind the use of CNN is because of its capacity to accommodate a larger number of trainable model parameters which contributes to the accurate extraction of features for pattern recognition as in image classification (AlexNet [7], GoogleNet [8], ResNet [9]), Object recognition (R-CNN [10][11]), scene segmentation (SegNet [12][13]) and other tedious human perception-based tasks. The commonly used CNN has convolution filters as the key feature detector from primary level features like edge, color, corner, and line detection to higher-level features like texture, pattern, shape, and detection for its class identification [14][15][16]. Hence the weights of convolution filters are the key parameter to train and determine how a particular filter works. Besides the convolution filters, many other learnable layers also participate in weight update during training via backpropagation, so that they all work conjointly to produce final down-sampled features with their class-label properties. Traditionally in a CNN, only the convolution kernels and multilayer perceptron (MLP) layers used to have the learnable parameters, however, now other layers besides them also use learnable coefficients during training, updated via backpropagation based on the first-order partial derivative of the participating polynomial function.

In general, the performance of CNN can be defined as a function of N, A, and H, where N=number of parameters, A=architecture i.e., connection design between the layers, H is hyperparameters with training conditions and X is the optimizing or decision function in each layer, hence CNN performance (P) = f (N, A, H, X). In this thesis, I will explore this idea with MRI classification task using two imaging modalities i.e., MRI and PET. The used methods and methodology for the proposed architectures and layers are well explained in upcoming chapters distinctly with related theory and mathematical expressions.

1.2 Thesis motivation

This thesis is mainly based on the motivation of CNN itself, which is bio-inspired by the neural structure and cognitive role of the human brain. Hence in an abstract way, it is the human brain itself motivating the use of CNN and its understanding. However, the implementation into an application is a challenging task. The existing CNN models in medical image classification utilize the models and decision function based on those models which are primarily trained on natural image datasets like ImageNet [18], CIFAR-10 [19], and Caltech-256 [20]. Thus, we were motivated to do independent work, and scratch train CNN models through MRIs to support AD diagnosis. The presented work mainly includes chronographically done research from basic understanding and implementation to modification of the participating layers.

The foremost work of designing CNN architecture i.e., ‘divNet’, was motivated with the interest of using increasing filter and stride size against the normal practice of equal size filter like in LeNet or converging filter size as in AlexNet. Hence, the concept was ‘accommodating more features, for the deeper layers with wider filter windows’ to reduce the depth.

The second work i.e., Gaussian activated parametric (GAP) normalization was inspired by the work of Alex Krizhevsky [14] and Kaiming et al. [9]. In the work of A. Krizhevsky, it is reported that the trained convolutional parameters in early layers were mostly the edge detectors and color filters, which were translationally invariants and spatially distributed. And Kaiming et al. reported the filters of the first convolutional layers were mostly Gabor-like filters such as edge or texture detectors, and the results after full training showed that both positive and negative responses of the filters are revered. Hence, the Gaussian filter kernel as a normalization kernel was implemented to generate a mask for average smoothing, which was parametrically designed to work as a normalization function.

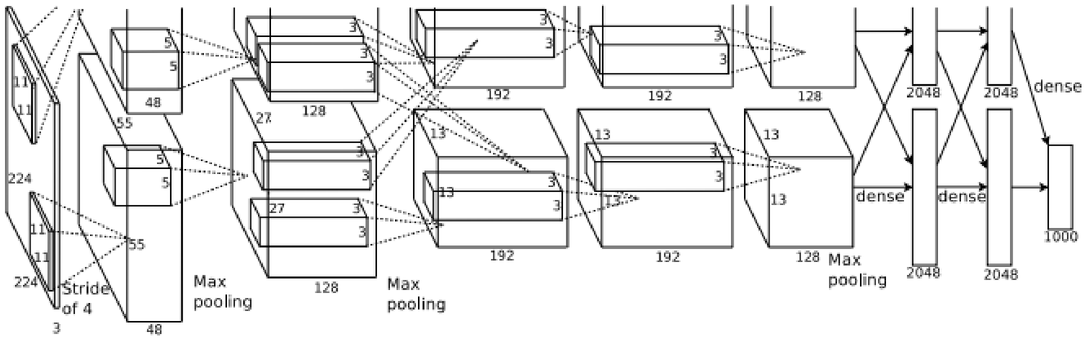
Later, in the design of the SGT activation function, we were motivated by the advantage of using gamma correction in image preprocessing and data augmentation. Hence, we implemented it as a learnable parameter-based function during activation for contrast enhancement and non-monotonic intensity mapping. Later it was combined with hyperbolic tangent function for thresholding and non-linear operation.

Therefore, in a broader sense with so many deep learning architectures and algorithms being successfully implemented in medical image identification tasks including segmentation of the brain [21], volumetric measurement of a brain tumor [22], cancer diagnosis [23], quantification of tissue materials [1], brain lesion detection [24][25], etc. I was also motivated to do similar task and with the original idea for AD identification. Though many Deep neural networks (DNN) including CNN, Convolutional autoencoder (CAE), and generative adversarial network (GAN) are already proposed for AD identification [26][27][28][29], I was motivated to design a simple and better 3D CNN architecture (most popular ones existing are 2D CNN) which was later customized layer-wise to enhance the feature extraction process in MRI.

1.3 Research objective

In current literature, we mainly find the use of sophisticated DL models like AlexNet, ResNet, GoogleNet, and VGG being tweaked to use in medical imaging analysis, though these models were originally trained and tested in natural image sets like ImageNet. In addition to the complexity of these models, these are not universally accepted for all medical imaging applications, so the goal was to design an independent model and make it widely acceptable for a similar range of medical imaging tasks. Hence, the primary objective of the research was to develop an independent robust standalone CNN classification model, specifically for MRI classification to support rapid AD diagnosis.

Another objective was to design novel layers that can be universally used for all kinds of perceptions i.e., input vector, however with more priority for MRI images. Moreover, in a broader sense, I wanted to study the correlation of CNN parameters with the classification categories and interpret the result for human understanding. This was one other avenue to explore which was partly implemented in the subsequent works. And lastly, besides finding a solution for the research problem, our other main objective is to perform ethical research directed at simplifying neural network applications rather than making them unnecessarily complex. Last but not the least, I also want to motivate early researchers working in similar fields accordingly.



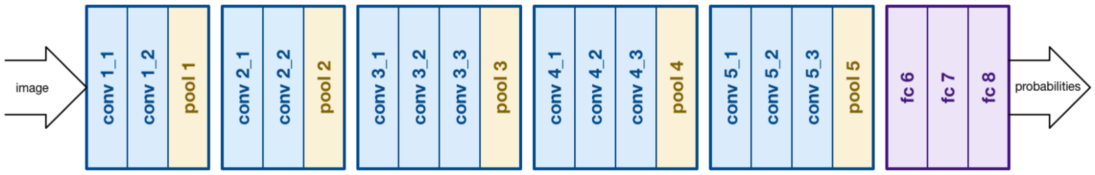


Figure 1.1: (a) The famous AlexNet architecture (2010) as reported in [7]. (b) VGG-16 architecture (2014) [17] is very similar to that of AlexNet, which outperformed AlexNet by a critical margin and became the baseline model for much deeper architecture.

1.4 Thesis contribution

This thesis explains its contribution to designing a baseline architecture for classification, proposed normalization layer, proposed activation layer, and the reason to propose those layers for MRI classification. It is highlighted below:

i. A diverging architecture-based 3D CNN referred to as ‘divNet’ was proposed for the supportive classification of both MRI and PET. Moreover, its sibling (i.e., slightly modified in window area) architectures have been thoroughly analyzed and investigated with experimental results.

ii. A novel Gaussian filter-based normalization layer referred as GAP layer was also proposed to integrate with deep neural networks. Also, instead of performing minibatch averaged scaling, same channel mean scaling was proposed for normalization within the layer. Additionally, a comparative analysis is performed for the proposed GAP in alternative to the Batch Normalization (BN) layer to study the feature extraction, histogram analysis, internal covariance problem via correlation test, and overfitting issue.

iii. An interesting activation function based on the combination of scaled gamma correction and hyperbolic tangent function, which is called Scaled Gamma Tanh (SGT) activation is proposed to replace traditional ReLU activation. The characteristics of the SGT activation function against other popular activation functions like ReLU, Leaky-ReLU, and tanh along with their role to confront vanishing/exploding gradient problems were analyzed thoroughly in a 3D CNN for the MRI classification task. More importantly to support our proposed idea I have presented a detailed analysis via histogram of inputs and outputs in activation layers along with weights/bias plot and TSNE-projection of fully connected layer (FCL) for the trained CNN models.

iv. Besides, I also wanted to coarsely interpret the proposed CNN models according to the distribution of its class weights i.e., AD, MCI, and CN, which were subsequently done in the analysis section.

1.5 Scopes and limitations

The performed study has a limited scope as the studied imaging modalities are only MRI and PET, PETs were later eliminated due to their poor result, concluded from the work in ‘divNet.’ Consequently, MRI was only used in the latter remaining works. Also, instead of using subcategorizing MCI into sMCI and pMCI, both subcategories were combined into a single MCI class, so only three main classes were used. In the case of the database, I selected only ADNI 1 project, baseline visits, whereas multiple visits are available in ADNI 2, ADNI GO, and ADNI 3 projects with plenty of MRIs under four categories. This narrow selection of MRIs was done to bring identical training conditions and avoid biased results for all three works. The selected MRIs were also class-wised balanced after the initial selection. Details on MRI and PET materials used for experiments are available in Chapter 2.

Another limitation is the use of CNN itself as many more superior deep learning networks like GAN, long-short term memory (LSTM), Fast r-CNN, etc. are readily available. The reason for using CNN is, though these superior models might bring higher accuracy, at the same time it is difficult to interpret them, even the popular CNN models like VGG-19, GoogleNet, and ResNet are very deep and immensely difficult to interpret. Hence simple models were proposed to interpret the model easily and simplify the process. Besides, any neural network either shallow or deep are prone to overfitting due to an excessive number of learnable parameters, hence I cannot be very confident that if tested on other MRIs (i.e., with different acquisition method), similar results will be reproduced by our models, hence lack of generality and universality is still a challenge faced by every deep learning researchers.

1.6 Thesis organization

This thesis accumulates the cumulative and representative research works done during my Ph.D. duration. The first chapter presents the general introduction of the research work where the research goal, objective, motivation, and contributions are highlighted. This chapter is made simple and non-technical so that general readers with lesser knowledge interested in this field can understand the research work and its intuition. Then the latter chapters become a bit technical where the proposed ideas are discussed in more technical details. Chapter 2 highlights the basic theoretical background and motivation behind the research. It discusses the theoretical aspect related to the design of CNN architecture, the need for transition from 2D to 3D CNN, the role of hyperparameters, dataset size, data selection and other related theory. Once we have the background on the basic CNN architecture, we move into the detail of the layers involved. In later part of Chapter 2, it discusses the theoretical aspects of two layers highly responsible for CNN performance i.e., normalization layer and activation layer. The proposed methods are slightly introduced and the motivation for the need of parametric layers are discussed in the end parts. Next, Chapter 3 presents the mathematical model and detail design for CNN architecture in section 3.1. Similarly, the proposed layers for normalization and activation layers viz, GAP normalization and SGT activation are discussed in section 3.2 and 3.3 respectively. Later in Chapter 4, the experimental results are shown. First with the proposed divNet architectures in section 4.1 and 4.2 and later using proposed GAP and SGT normalization in section 4.4 and 4.5. Additionally, the results are properly analyzed and discussed in sub sections 4.3, 4.2 and 4.6. Finally, a general conclusion for all of the proposed methods and performed research is highlighted in Chapter 5. Besides, the general concluding remarks, the future works that can be done is also highlighted in subsection 5.1. All of the references are presented in Reference section and some additional data in Appendix section. This is illustrated in figure 1.2, where readers can easily understand the content of the thesis with a single glance. Please note that this is a chronologically done work, and each work is connected to each other.

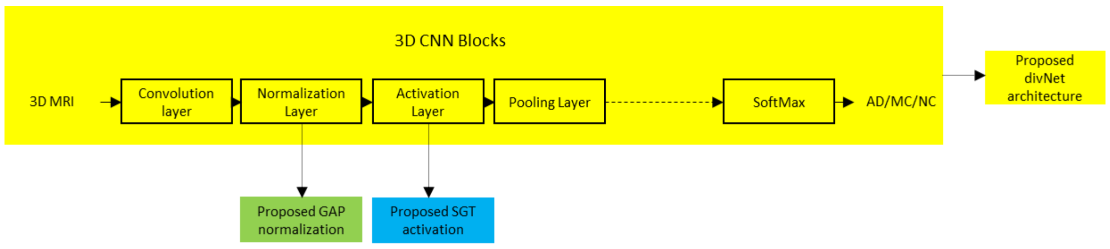


Figure 1.2: Block-diagram illustrating the thesis work.

CHAPTER 2

Theory and Background

2.1 CNN for MRI classification

In reference to the Alzheimer's Association Report (AAR) [30], the molecular and neurological causes for AD takes place in the brain nerve cell connection area called the synapsis, where the neurotransmitters are released. The synapsis helps with the information flow caused by tiny bursts of chemicals that are released by one neuron and are detected by a receiving neuron. During severe dementia, ß-amyloid proteins and tau proteins, also known as tau tangles, are accumulated around the synaptic region. This ß-amyloid is suspected to cause neuron death by interfering with neuron-to-neuron communication at the synapsis. In addition, the tau tangles block the supply of nutrients and other essential molecules inside the neurons. Brains with advanced AD have a dramatic shrinkage due to cell loss, inflammation, and widespread debris from dead and dying neurons. This causes memory loss problems (e.g., dementia) with the inclination of age. This is the molecular and physiological level analysis for AD. However, there is a corporal change in the common AD-related variation of anatomical brain structures such as the enlargement of ventricles, shrinkage of the hippocampus shape, change in the cortical thickness, and other cerebral areas containing white matter and gray matter brain tissue as well as cerebrospinal fluid. These changes and atrophies are rationally visualized through the brain imaging by the clinician while using a variety of medical imaging modalities like positron emission tomography (PET), magnetic resonance imaging (MRI), and computed tomography (CT) scanning. Here comes the true usage of image processing and machine learning. Image processing improves the quality of the image for better visualization of the brain whereas machine learning assists clinicians to perform other logical operations like segmentation, classification, and quantification, which can be time-consuming and sometimes baffling. The logical operation once modeled with proper supervision can later follow the designed algorithm to reach a prediction, the more the prediction is true, the better the model is, and the higher will be the chance of reliability. Mild cognitive impairment (MCI) is a transitional stage between normal aging and the preclinical phase of dementia. MCI is a potential early stage of AD, and it can either progress into AD (pMCI) or remain in the same stage throughout life, which is called stable MCI (sMCI). Here, both types of MCIs are combined into a single MCI group to ease the classification process. A healthy MRI is called normal aging/cognitively normal (CN). Since AD contains a genome that affects the disease, no known stimulant causing it is identified. However, the influencing factors for AD include genetics, low education or professional involvement, lack of-mental exercise, family chronicles, and external or internal brain injuries [31].

Image processing aims to find a discriminative pattern of image features by collecting the same groups of the MRI into one. Once the MRI is translated into an image from the magnetic resonance frequency, it represents the pixel value for each structure and these pixels will be assigned to a class. Ultimately, AD classification will be based on the features that are extracted from these brain image pixels. The main features required to accurately capture the major AD-related variations of the anatomical brain structure includes the size of the ventricles, hippocampus shape, cortical thickness, and brain volume [32]. Although such alterations may resemble other brain-related diseases like Parkinson’s disease (PD) and encephalitis [33]. In that case, more clinical and physiological tests should be performed on a genetic level. Consequently, the idea of identifying pathogenic scans from a healthy one seems easier than identifying a particular disease from a pool of pathogenic scans. Thus, imaging technique alone may not be the only valid proof to diagnose a person with AD. However, based on the brain phenotype reflected in the imaging, the discriminative features from the trained network can help identify AD prone images.

This study answers few questions related to the use of deep learning for medical imaging. It starts with the background story of CNN and recent literature reviews of its implication in medical image classification. Subsequently, I have surveyed with shallow to deep layers using different feature sampling region and finally came up with a diverging architecture being supportive in the case of both MRI and PET. The proposed architecture referred as ‘divNet’, and its sibling architectures have been thoroughly investigated and the results are presented, discussed, and analyzed in subsequent sections.

2.2 The Background story

2.2.1 3D CNN

Inspired by the neural network architecture of the mammalian cerebrum, an artificial neural network (ANN) tries to mimic the information flow and the decision-making pattern of the brain. As demonstrated by Hubel and Wesel [34], they recorded the activity of a single brain cell in cats. It was stated that some cortical cells respond to contours of a specific orientation. Aside, patterns of light stimuli are most effective in influencing units at one level, and they may no longer be the most effective for the next. Although millions of neurons and synapses receive the stimuli, only certain neurons are trained to respond to those specific features or aspects of an image [35]. Similar to the brain when we receive any stimuli, the neuron spike is generated for only a specific area, ANN will only have a few activated nodes for each shape, which may be a horizontal, vertical, or diagonal line. The node activated for each line is different and unique. This means that the node activated for a horizontal line in one image is activated for the horizontal line in another image and so on; this is the basic principle of an ANN. The layer-wise connection between the nodes may indicate the heavy connection between the neurons.

CNN is an advanced type of ANN, which has convolutional filter elements (weights) unlike single-node multiplication in ANN. Besides, CNN has extra feature investigators in the form of pooling and activation functions. Thanks to the newly developing algorithms that train the CNNs more effectively, which has ultimately surpassed human-level accuracy for natural image classification [35-36]. With a wide variety of CNN based topology, the prominent ones include residual (Resnet50, ResNet101 [55]), recurrent (RCNN [53]), inception (GoogLeNet [50]), encoder-decoder (U-net [39]), and so on. One can notice that the common element in all of the topologies is the encoder unit i.e., convolution-normalization-activation-pooling, which acts as the fundamental unit for feature generation. Therefore, I am building blocks of a combination of these encoding layers.

The existing ideas in the 3D CNN are mainly 'the best patch' or 'multiple patches trained for the CNN ensemble' based architectures [37]. In 'the best patch' approach, a single region of the brain is selected based on the recommended region of interest (ROI) or it is manually assisted from the anatomic region of atrophy, like the hippocampus and amygdala whereas in 'multiple patches trained for the CNN ensemble' multiple CNNs from multiple ROIs are trained separately for each region, later performing feature concatenation at the last FCL before classification. One of the reasons behind using only limited/selected/informative pixels to feed in 3D CNN may be due to GPU memory constraints and also to increase the information with quality. Non-discriminative parts although play a role in feature construction at a low level may not necessarily support the cohort classification, hence information becomes redundant using a whole-brain model. As well, selecting an ROI patch, or simply the best region makes the system semi-automatic; hence, the truest sense of automatic feature extraction is not applied in these cases. This research aims to make the classification simpler and candid rather than a multifaceted process. That's why I wanted to build an automatic and discriminative CNN that can work for MRI, PET, and any other 'pixelary' (pixel-based) object/entity irrespective of its input size.

Yechon et al. [32] works were mainly focused on the hippocampi region, where they proposed a multimodal 3D CNN that uses hippocampi region ROI from MRI and hippocampi and/or cortices ROI from PET, without segmentation as a prerequisite task. They separately trained the CNN referenced with VGG architecture, for MRI and PET modalities-based ROI and later concatenated from final FCL before final classification. In other similar attempt done for multimodality-based 3D CNN, Liu et al. [38] also proposed a simpler CNN model like Yechon et al. but then instead of concatenating the final FCL, the concatenation was done in the convolution layer, from each CNN (trained using PET and MRI patch) for sequential convolution until flattening features at FCL. They experimented with T1-MRI and FDG-PET based cascaded CNN, which utilizes a 3D CNN to extract features, and adopted another 2D CNN to combine multi-modality features for task-specific classification. In 2016, Hosseini-Asl et al. [43] proposed a deeply supervised and adaptable 3D CNN (DSA-3D-CNN), trained on structural MRI (sMRI) images, which gives the prediction for the AD vs. MCI vs. CN task. Similarly, Payan et al. [64] also used sparse auto-encoder (SAE) patch-based 3D CNN to classify MRI scans using dataset partitioning unlike Oh et al. [65], where they performed 5-fold cross-validations (CV) using convolutional auto-encoder (CAE) based volumetric or 3D CNN for AD vs. NC and supervised transfer learning for sMCI vs. pMCI classification.

2.2.2 Why move from 2D to 3D?

This study aims to explore one more dimension for CNN i.e., the depth. And the key question that needs to explore is: can we only depend on the 2D CNN results?

As mentioned, the 2D CNN can easily be misled [67] in the sense that a target domain trained CNN can only give a probability score for each trained class. Besides, a few pixel changes can make the prediction a disaster [67]. Some researchers have suggested possible improvement in performance over 2D images if 3D whole-brain structure is used to train CNN [39], due to its deeper architecture. But deeper architecture means more parameters (weights in layers) to train, and at the same time requires bigger and better training material. CNN either 2D or 3D follows a generic feature extraction pattern [40][41], here generic features might suggest CNN features, also called ‘off the shelf CNN features’ [42] which is basically the image features extracted from the multiple convolutional layers as the weights (as a decimal number) of the trained network, applying various activation functions. Typically, the final feature weights from the FCL are graphed out to decide the performance of CNN. This means a well-separated, class-based segmented graph generally depicts a well-trained classifier [43].

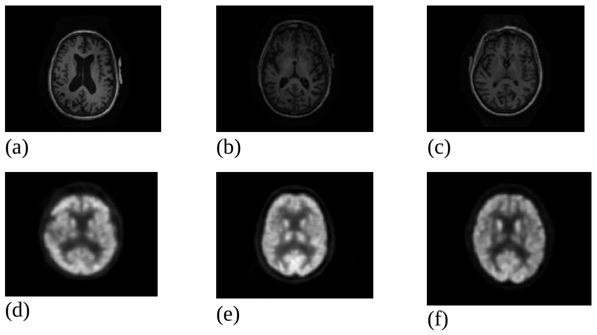
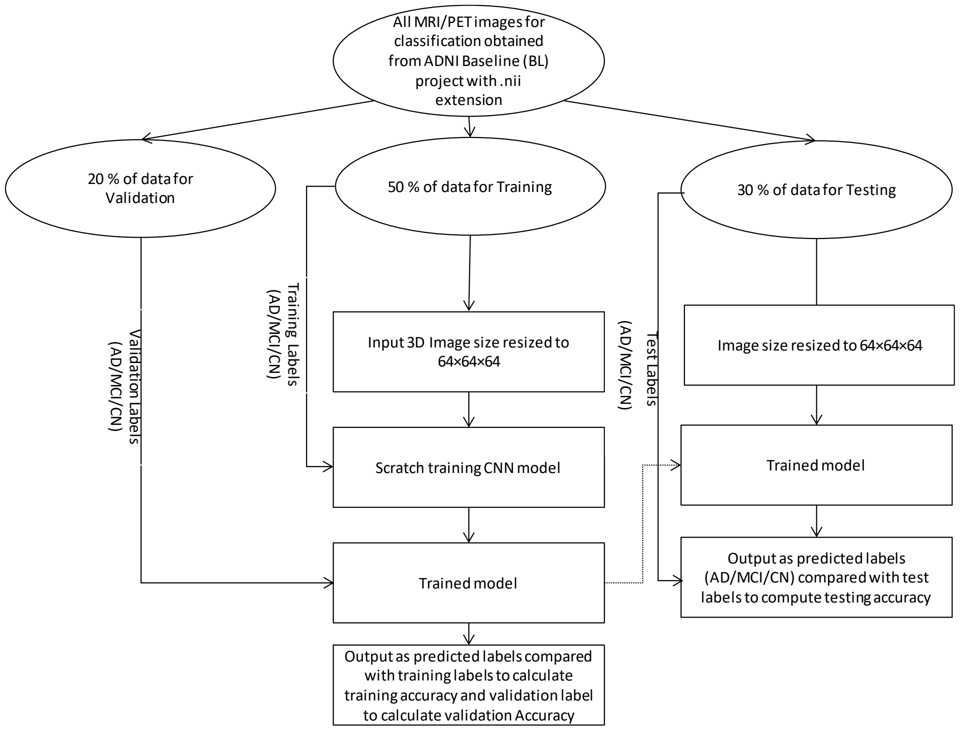
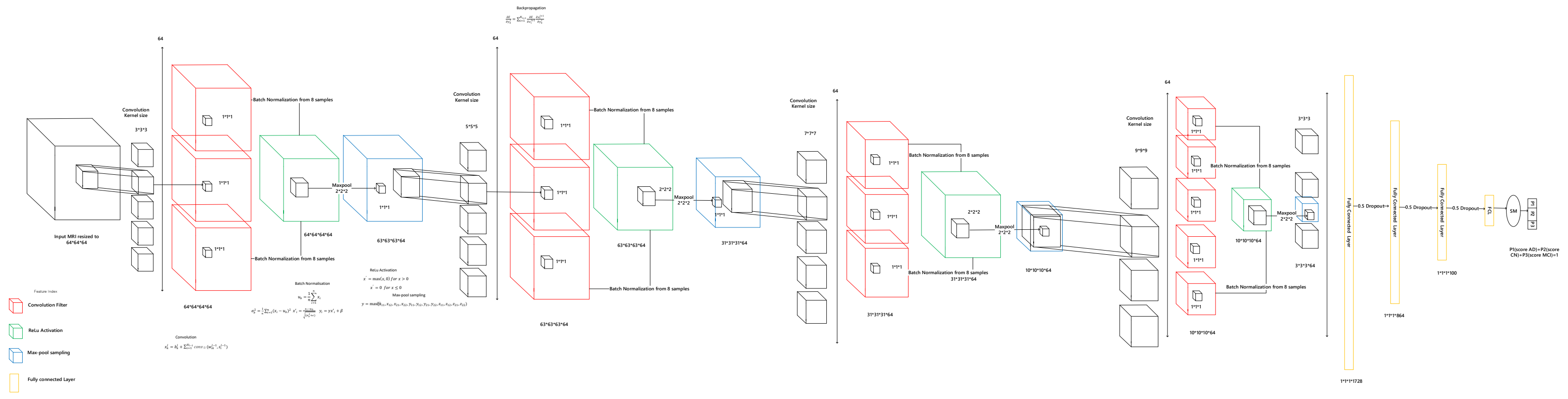


Figure 2.1: MRI and PET scans of: (a) AD prone MRI; (b) Healthy MRI; (c) MCI affected MRI; (d) AD prone PET; (e) Healthy PET; (f) MCI affected PET.

While classifying images with 2D CNN, the major problem is to select the appropriate slice or slices as training inputs. Several literatures suggest the 'best scan' [44] [45] or 'best multiple scans' [46] [43] for efficient performance, which rather mystifies the slice selection process. This is problematic and quite impracticable every time. Some important information might be missed if we focus only on limited scans or orientation. Consequently, the safest way to ensure is to use whole brain volume, which comes in a three-dimensional pixel value for x, y, and z dimension in planar geometry. In our previous work [47] we demonstrated that 2D CNN when trained from fewer MRI images results in poor classification performance, moreover the selection of slice or slices is still ambiguous. Besides, the transfer learning idea seems an inappropriate choice as the popularly available models like AlexNet [48], ResNet [49], GoogLeNet [50], ZNet [51] are all 2D based architecture. Furthermore, we need to make the architecture deeper and bulkier to accommodate thousands of images per class due to the dimension constraints of 2D-CNN. Hence to make the MRI classification universal and less tedious 3D CNN is used, to readily fit the 3rd dimension of whole MRI. Likewise, the use of 3D input requires fewer pre-processing steps like slice-extraction slice-correction and slice-selection. This reduces the manual processing steps and makes the procedure more automatic and robust, which is the goal of this study. Regarding image preprocessing only image resize, and normalization were performed before being fed into the CNN to work with the irrelevance of the imaging protocols and scanners selections. Aside, the obtained nifty files are already pre-processed from Alzheimer's disease Neuroimaging Initiative (ADNI), (we are not provided with the raw image from scanner, but a semi-corrected processed MRI). This can eventually be useful for the generalization of the trained model.


All MRI/PET images for classification obtained from ADNI Baseline (BL) project with .nii extension
20 % of data for Validation
50 % of data for Training
30 % of data for Testing
Validation Labels (AD/MCI/CN)
Training Labels (AD/MCI/CN)
Test Labels (AD/MCI/CN)
Input 3D Image size resized to 64×64×64
Scratch training CNN model
Trained model
Output as predicted labels compared with training labels to calculate training accuracy and validation label to calculate validation Accuracy
Image size resized to 64×64×64
Trained model
Output as predicted labels (AD/MCI/CN) compared with test labels to compute testing accuracy

(a)



(b)

Figure 2.2: (a) Workflow of the experiment (b) Pictorial representation of proposed 3D CNN architecture for the MRI/PET classification based on the diverging area of the reception, which is referred to as ‘divNet’. High resolution image for better visual presented in Appendix I.

2.2.3 Finding the correct architecture and hyper-parameters

Although CNN can be easily misled, it is quite smart as well. Because irrespective of the depth (deep or shallow layer), the training material (good or bad), or the training size (small or big), CNN finally learns something when it is trained. This ‘something’ may not typically relate to the human interpretable logical features however they will categorically learn some identifying features so it can be classified. Most of the time this features are basic shapes, edges, corners, and patterns on the objects. Therefore, we don’t need to worry about selecting architecture every time. Nevertheless, when it comes to finding the best architecture, with ease of training, and good performance, the trio gives an ultimate challenge to every deep learning researchers. Performance results, training time, validation period, the confidence of prediction, generalizability, and other factors are the key to determine the state-of-the-art winner. The results of our experiment are highlighted in tables 4.1,4.2 and 4.3.

2.2.4 How deep should we go?

Recent studies have recommended that a CNN can extract convenient features directly from a raw image, unlike a manually supervised learning algorithm and has a robust potential to locate key points and features in object detection tasks for natural images [52] [53]. This property of the CNN has been explored in a region-based convolutional neural network (R-CNN) for region-based detection in 2D images. Recent works in image classification using a CNN suggests that segmentation results itself do not contain information needed for the classification, hence not being a pre-requisite for the task. Consequently, the CNN can learn useful features without labeling the voxels itself [32]. These entire experimentations support the generic feature extraction property of CNN. But how deep should we go is still a major question. Our choice of going deeper is to extract more meaningful features to perform a relevant operation of classification or segmentation from the trainee dataset. In general, we will have more feature vectors with more layers, and subsequently a large pool of features to extract. This will help in terms of 'judging' the best out of the good features. Nevertheless 'we should go deeper' [54] doesn't necessarily mean for the deep learning models every time. The work of He et al. [55] in ResNet shows that a deeper network with 1,202 layers in comparison to 50, 101, and 152 convolutional layers has no significant improvement with an aggressive depth. With the additional cost of extra training, more depth for a network may make it more prone to overfitting by learning "too well" and this may not generalize the model at the cost of running expensive GPUs which makes it more challenging to build models, being able to understand all details [56].

2.2.5 Data as fuel for CNN, but how large should our data be?

The breakthrough of AlexNet in the ImageNet dataset classification suggests that better the data, better would be the result. To support this theory artificial dataset are also created with different augmentation techniques. And well, the result seems to be supported using extensive synthetic MRI for improved performance in segmentation and classification tasks [57] [58]. In case of ImageNet, we have 1000 classes with around 8000 images in each class, which means more classes with more distinctive images. Similar is the case with other datasets like CIFAR101, Caltech, etc. where data acts like oil to AI [59]. Having said that, what may be the case with the medical image? Considering labels as the most precious assets for the data scientist, how voluminous should the training materials be? In the case of medical images, the task is more challenging, with an image-based feature; we can rarely detect the atrophy pattern. Particularly if we look at AD vs. MCI or MCI vs. NC, MRI or PET [Figure 2.1]. Hence to solve this I am experimenting with various sizes of the datasets, one big and the other small for MRI and PET tests. The results are highlighted in Table 4.4. Detailed demographics for each dataset type tabulated in the Appendix I.

2.2.6 Visualizing features: What has the CNN extracted and learned?

A sequential CNN involves reduction of features from the input to the final classification layer. The same is with a 3D MR image, as input obtained in NIfTI (Neuroimaging Informatics Technology Initiative) format. Once input is read using *niftiread* function inbuilt in MATLAB, it can be resized from its original size 256×256×256 or 256×256×170 to 64×64×64. After multiple down-sampling via the max-pooling for each convolutional layer, it is reduced to 1,728 for the first FCL. To reduce overfitting, dropout is used and other conceding FCL to make the final output as input for the SoftMax layer. This idea of using multiple FCLs to map the target domain using pre-trained networks is often called target domain fine-tuning, which is the basis for transfer learning. The activated features in the initial convolutional layer can detect pixel changes based on attributes like line, edge, and color [60] in a small window filter. These edge-based features pass through the intermediate layers of the CNN, and they are combined in many filters, whose weights (initially kept at random weights or initialized using Xavier, He, Gaussian) is updated using backpropagation following a specific optimization path like ADAM or stochastic gradient descent (SDG). These intermediate layers detect the activated parts of the image whereas the final layer learns discriminative features in the shape and pattern amongst the target domains. Once training reaches convergence, which means no more weight changes occur and the training accuracy reaches its maximum, the training stops. The network is now trained and it’s a generic feature extractor, which is like a traditional algorithm that generates features. The generated features are the discriminative features that are used to distinguish the classes. This study uses multiple 3D filters that give 4D output in each layer i.e., one 3D feature map per filter (please see Figure 2.2). Convolving the image with these filters produce a feature map that detects the presence of those features in the image. This nature of a CNN is the essence of its auto feature extraction, and it helps in the automatic computer-aided design (CAD) system.

It is difficult to predict the features that a CNN can learn without training it; thus, making it a tedious task to analyze the features. Since a single network may contain millions of parameters and we cannot mathematically predict the final converged value in each filter without training them. Hence, every time I train the CNN, the learned features need to be investigated. Once trained, the CNN is loaded with the filter weights, which are used to make the predictions with the test images. It is convolved in each layer to obtain different results for the different MRIs. The trained network is used to obtain the features as described in Pseudo-code 1, 2, and 3.

2.3 Normalization layer in CNN

DNNs have been the dark horse in the field of DL after the success of LeNet-5, an emerging Convolutional neural network (CNN), in 1989 for handwriting recognition [72][73]. The massive success of DNN is due to its capacity to accommodate a larger number of trainable model parameters which contributes to the accurate extraction of feature for image-classification as in ImageNet using AlexNet [74], GoogleNet [75], ResNet [76], object recognition (R-CNN [77],[78]), scene segmentation (SegNet [79],[80]) and other tedious tasks for human perception. The commonly used DNN in image classification or pattern recognition is CNN, with convolution filter as the key feature detector from primary level features like color, edges, corners, line, etc. to higher-level features like texture, pattern, shape, etc. for its class identification [81][82][83][102]. Hence, CNN is basically an image attribute extractor. In CNN, the weights of convolution filters are the key parameter to train, and it determines how a particular filter works. Besides the convolution filters, many other learnable layers also participate in weight update during training via backpropagation, so that they all work conjointly to produce final down-sampled features with class-label properties. The training algorithm for optimization and the initialization techniques are the key components to escort the cross-entropy loss to minima, so that the training can be halted, i.e., the weights is no more changed and technically the model/network is said to achieve convergence. Achieving optimal convergence is our primary goal however, only reaching the lowest minima does not guarantee a high success rate in test (unseen) sample classification. i.e., the ‘test error’ can be high with a low ‘training error’. This familiar problem in ML is called overfitting or generalization error, which is the most serious issue in any DNN architecture. Many regularization techniques along with dropout, early stopping, random sampling, etc. are introduced to reduce this generalization error. However, being the state-of-the-art algorithm in several benchmark datasets, numerous ML algorithms are still not fully understood and working as a black box in many tasks, it can be realized by the fact that many standard DNN fails to generalize [84][107][108].

Any NN under scratch training gets affected by the randomness in the parameters, which brings some short of disparities in the input node distribution of layer during training time and makes it extremely challenging to train networks with saturating nonlinearities which is measured as covariant shift [85]. This phenomenon considered specifically as an internal covariant shift is sort of trouble for convergence and generality as well. To solve this problem, BN was introduced to minimize the effect of covariant shift. It also speeded up the training process and deal with a higher learning rate without exploding gradient. BN uses layer-wise whitening technique image i.e., mean zero and unit variance for normalization and decorrelation, with only two extra parameters per activation one for scaling and the other for shifting. It also reduces the training time and preserves the interpretation capacity of the network [86]. However, since the convolution weights are updated so quickly, the weights tend to move towards convergence faster while the validation accuracy still lags far behind. BN was intended for speedy training reducing internal covariant shift also, the claimed regularization technique was said to eliminate the use of dropout. Because of this BN also claims to reduce the generalization error as dropout does the same. However, at the same time, a prominent overfitting problem was observed with the use of BN in the base architecture. Along with BN, many other lately used activation functions for classification purposes are designed with learnable parameters for optimal model fitting and little overfitting risk [87]. Similar works done lately to design normalization layers without using minibatch mean in DNN includes filter response normalization (FRN) [109], group normalization (GN) [110], and layer normalization (LN) [111]. All these methods do not operate in batch dimension to avoid minibatch dependency for calculating scaling factor and only use activation map channel statistics. However, none of them uses any filtering function like Gaussian filter or image filters for sharpening effect in the layer as our proposed method. Therefore, the major interest was in designing a normalization layer along with activation parameters to address these issues of overfitting and covariant shift. For experiments, 4 layers of an encoder-based network are used as a base architecture with a normalization layer in each encoder. Likewise, AlexNet with local response normalization [74] which is similar in architecture to our base model is also used for scratch training all dataset to compare the classification result. The proposed method is explained mathematically via matrix equivalency with the CNN normalization layer. I have used three parameters per activation in the proposed layer, one for scaling the original feature, the other for scaling the masked feature, and the last one as offset or bias to shift the output. Our contribution in this section is:

i. A novel Gaussian filter-based normalization layer was proposed to integrate with deep neural networks, which we call GAP layer. Moreover, as a replacement for of performing minibatch averaged scaling, same-channel mean scaling is proposed for normalization within the layer.

ii. Comparative analysis was performed for the proposed GAP layer in alternate to BN in the base architecture to study the histogram analysis, feature extraction, internal covariance problem via correlation test, and overfitting, along with the comparison of the final classification results.

2.3.1 Background and motivation for GAP normalization

The convolutional filter weights in the initial layers of CNN tend to mimic some image filters once the network is fully trained. In Krizhevsky et al. [74] the trained convolutional weights/parameters were primarily the edge detectors and color filters. The edge filters could detect the vertical, horizontal and diagonal edges, as well these filters were translationally invariants in nature and so worked spatially for all the input images. Kaimming et al. [87] reported the filters of the first convolutional layers were mostly Gabor-like filters such as edge or texture detectors, and the end result after full training showed that both positive and negative responses of the filters are revered. With these previous results pertaining to the basic image filters and negative responses being equally important during training, it motivated in in designing a layer for early feature detection to imitate image filters. For this instead of manually initializing the filters, the goal was to find filters to smooth the overall result like edge, color, blob, etc. and not only the single features at a time. The result of other experiments showed the use of edge detection filters like Sobel, Prewitt, Robert, etc. [88][89][90][91] works unidirectionally without bringing feature variance and the classification result was also poor. For this, Gaussian filter was selected, which can be used as an average smoothing function along with the normalized input. However, using Gaussian smoothing with batch normalization seems inappropriate therefore, to normalize the features, channel mean normalization was utilized. In this normalization, the mini-batch properties in not entertained, instead the mean features from all the filter weights from all channels) are normalized for each batch dimension as described later in detail in section IV. As a result, the obtained mean feature vectors work only for its channel images without combining the mini-batch properties. In doing so, the addition of noise is reduced, and consequently single image property is only summed up.

In an experiment [92] to realize the effect of randomness in NN, the Gaussian function was applied to add noise to the input and bring non-uniformity in the signal by combining random pixels to the source image. This demolishes the relationship of training pixels with its label. Further to make the experiment more random, the image pixels were re-shuffled and re-sampled from a Gaussian distribution. However, the NN was not severely affected by this and was able to accommodate the test samples accurately. Thanks to the stochastic gradient descent algorithm [93] employed during the training. In the proposed methodology, Gaussian function were not used as a noise generator, but as the mask generator for the un-sharpening process. Instead of performing Gaussian smoothing on the whole image, the Gaussian kernels were used to generate matching mean and variance to the original image kernel. Kernel size is determined by the size of the preceding convolution filter kernel, to prevent mixing of filter properties during the smoothing process.

2.3.2 Gaussian filter and un-sharpening process

Gaussian filter [94] is a spatially weighted image filter, which functions as a point-spread function for any image-pixel distribution. It operates as a non-uniform low pass filter with a higher weight to the central pixel generating a normal distribution of pixels. Its rotationally symmetric nature provides directional unbiasedness for image morphological operation. This property also enhances to marginally preserve edges and brightness, while yielding smoothing results for image attributes. T. Lindeberg [95] derived the optimal discretized Gaussian kernel and proved Gabor functions can look very much like Gaussian derivatives. In order to operate in the CNN feature matrix, a discrete approximation of its kernel is required which is done as in equation (2.1) where $i, j,$ and $k$represents the matrix row, column, and depth and σ2 is the input variance which is equal to 1 for standard Gaussian output. The output for 3×3×3 and 5×5×5 matrix is shown in plot Figure 2.3.

$$h(i,j,k,\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^3} e^{\frac{-(i^2+j^2+k^2)}{2\sigma^2}} \quad , \tag{2.1}$$

the term $\frac{1}{\sqrt{2\pi}}$ is a normalization constant that comes from the truth that the integral over the exponential function is not unity. It is demonstrated that the convolution is a linear operation [94] so to bring non-linearity in the DNN, we need to add some non-linear functions like Leaky-ReLU for rectification. A 3D kernel is applied in designing GAP layers for both 2D and 3D CNN, in 2D the filter operates with the average of all convolution-based images, whereas in 3D the mean value is operated in the whole volume itself. Yet, in both scenario, the learning coefficient parameters are updated independently acknowledging the mean effect from its channel filters for the final output.

The un-sharpening process requires three steps i) blurring of image i.e., correlation operation of input image volume with the Gaussian kernels ii) subtraction of original input image volume with its blurred version to generate masking kernel iii) adding the masked version to the original input. Figure 3 demonstrates the operation involved inside the layer in more detail.

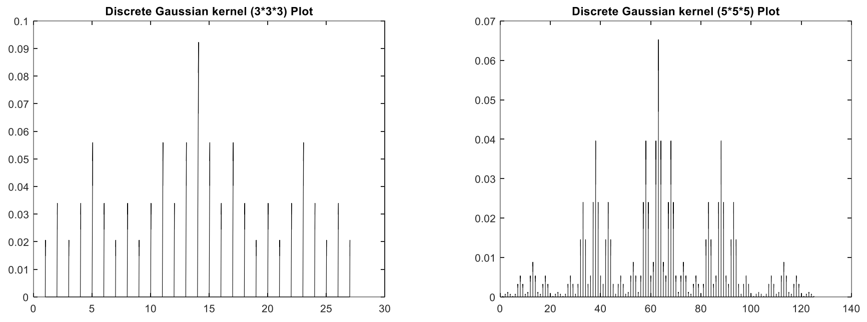


Figure 2.3: Visualization of Gaussian 3D kernel in the linear plot (a) 3*3*3 kernel with 27 weights matrix (b) 5*5*5 kernel with 125 weights matrix. Please note the higher central value for each corresponding peaks.

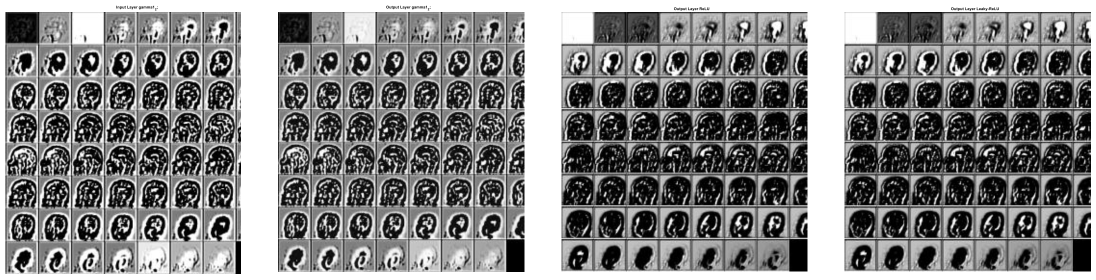
2.4 Activation functions in CNN

An activation function is used in DNN primarily for two purposes, first to add non-linearity in the whole model to learn complex patterns, second to normalize or threshold the output of each layer to decrease the computational burden. Here, for a CNN, if only linear activation *f(x)= wx+b* is used, then assembling multiple functions of *f(x)* generates only a single degree output (noting that the convolution layer itself is also a linear operation layer). Aside, the final values can monotonically explode to a maximal or minimal level causing difficulty in training to reach convergence. Thus, the learned polynomial expression should be in order larger than 1 to learn complex patterns due to multi-dimension features [114][115] i.e., the decision boundary needs to be non-linear. For this, the activation functions need to be chosen properly in deep networks as it has significant effects on the training dynamics and required task performance [114][116][117].

Traditional neural networks employing Multilayer Perceptron (MLP) used sigmoid function or hyperbolic tangent (tanh) as a non-linear operator in its node [118-121]. Latter with emerging complication in DNN, several other activation functions based on the non-linear operation were proposed. However, most of them were overly complex and intended for a very deep network for their high-level abstract representation in natural image datasets like ImageNet [122-124]. It made the network more complex to understand its working mechanism and feature extraction process [125]. Thus, still for may applications simpler non-linear rectifiers like ReLU [126] and its variants Leaky-ReLU [127] are the most popular ones. Besides, other new functions like Parametric ReLU (P-ReLU) [128], GELU [129], ELU [130], SELU [125] are being occasionally used in DNN [118][131][132][133]. ReLU described as $f(x)=max\ (0, x)$ completely blocks the negative input for positive gradient flow whereas its other variants allow a computed flow of negative input for small negative gradients loss. Although the vanishing gradient problem was solved with positive gradients loss in ReLU, it gave rise to another similar problem called ‘dying ReLU,’ which is confronted if higher negative input keeps on prevailing at the cost of sparsity. Later these problems were marginally solved using Leaky-ReLU and P-ReLU [126][128] with non-zero activation for negative inputs as $f(x)=\alpha x$, where α is a constant or a learnable parameter. Nevertheless, in the case of medical image classification like MRI and PET, ReLU and Leaky-ReLU are still the predominant ones due to their simplicity and training images being in greyscale format. Recent works in MRI classification utilizing DNN involves designing robust and better architecture, ensemble models along with clinical features, and experiments to apply new optimization algorithms [134-138]. While very few works have are done in designing novel activation functions specifically to MRI, as most researchers use the existing activation methods [139-141]. Hosseini-Asl et al. [134] used Sigmoid and ReLU function to design deeply supervised and adaptable 3D CNN (DSA-3D-CNN) trained on structural MRI (sMRI) images, for the prediction of AD vs. MCI vs. controlled normal (CN) task. Payan et al. [135] proposed sparse auto-encoder (SAE) patch-based 3D CNN using sigmoid activation function to classify MRI scans. Similarly, Oh et al. [65] performed 5-fold cross-validations (CV) using convolutional auto-encoder (CAE) based volumetric CNN with ReLU as the activation function for AD vs. NC classification along with supervised transfer learning for sMCI vs. pMCI classification. Gupta et al. [63] used CNN with sigmoid activation function to classify MRI into 3 classes with transferred features learned from natural images using autoencoder. E.Goceri[138] proposed Sobolev gradient-based optimization for 3D-CNN, results for MRI classification accuracy were reported higher with Leaky-ReLU in comparison to sigmoid and ReLU. Recently Huang et al. [139] implemented a combination of GELU and ReLU in their DNN model for brain tumor image classification and achieved a 95.49% success rate.

Generally, Gamma correction ($f(x)=x^{\gamma}$) [143] is about contrast enhancement and non-monotonically intensity mapping to new values, depending on the exponent γ for the input x. In deep learning scenario, Gamma correction is mostly used to produce augmented images (with defined γ values like $\gamma=0.5,1.5,2$, etc.) for increasing training material [144-146]. This idea seems helpful to increase the training result by producing multiple versions of gamma-corrected images using different values of γ in $f(x)=x^{\gamma}$. However, it should also be noted that some image's quality might deteriorate due to the unmatched version of gamma. With the higher value of γ, we can wash out the image whereas with the lower value of γ we might lose the important pixel information.

Hence ‘γ’ should be a ‘versatile’ constant or technically a learnable parameter as per channels rather than a ‘fixed’ constant. Hence our idea is to select appropriate gamma value for each image, or more specifically for all the images (or their features) obtained from all the channels output after BN. Hence our method is not to increase the number of augmented images rather find appropriate values of gamma for each filter output and bring non-linearity in the model at the same time without increasing the number of training samples which basically works as an activation function (please see Figure 2.4).



(a) (b) (c) (d)

Figure 2.4: Comparison of activation using different functions for a sample MRI observed in 2nd activation layer (22nd of 64 channels) corresponding to 63×63×63 image as a montage here. It can be observed that the output from the gamma layer using SGT has well preserved the feature attribute present in the first three and last few slices in comparison to ReLU (c) and Leaky-ReLU (d) where (a) is the input feature matrix.

In this work, a novel activation function is introduced with the stepwise combination of gamma correction technique and hyperbolic tangent function. Although zero centered symmetric functions like Sigmoid, tanh is advantageous for activation function for un-skewed gradients however, those functions proved to be not very worthy due to the vanishing gradient problem. Figure 2.5 shows the proposed activation function plot for different case of input *x* and its first derivative. Here blue curve represents the SGT activation function whereas the red curve represents its first-order derivative. Please see Appendix III for all the related equations also see figure 2_app for all the related equations. As each activation layer is preceded with BN layer, the idea is to distribute histogram with saturation at low and high intensities of input data, which was originally mean centered at zero with unit variance. In other words, the intensity profile is dispersed from the central region to the edges. This brings higher variance in weight distribution with significant discrimination in features to support the classification (please see histogram distribution figure 4.16).

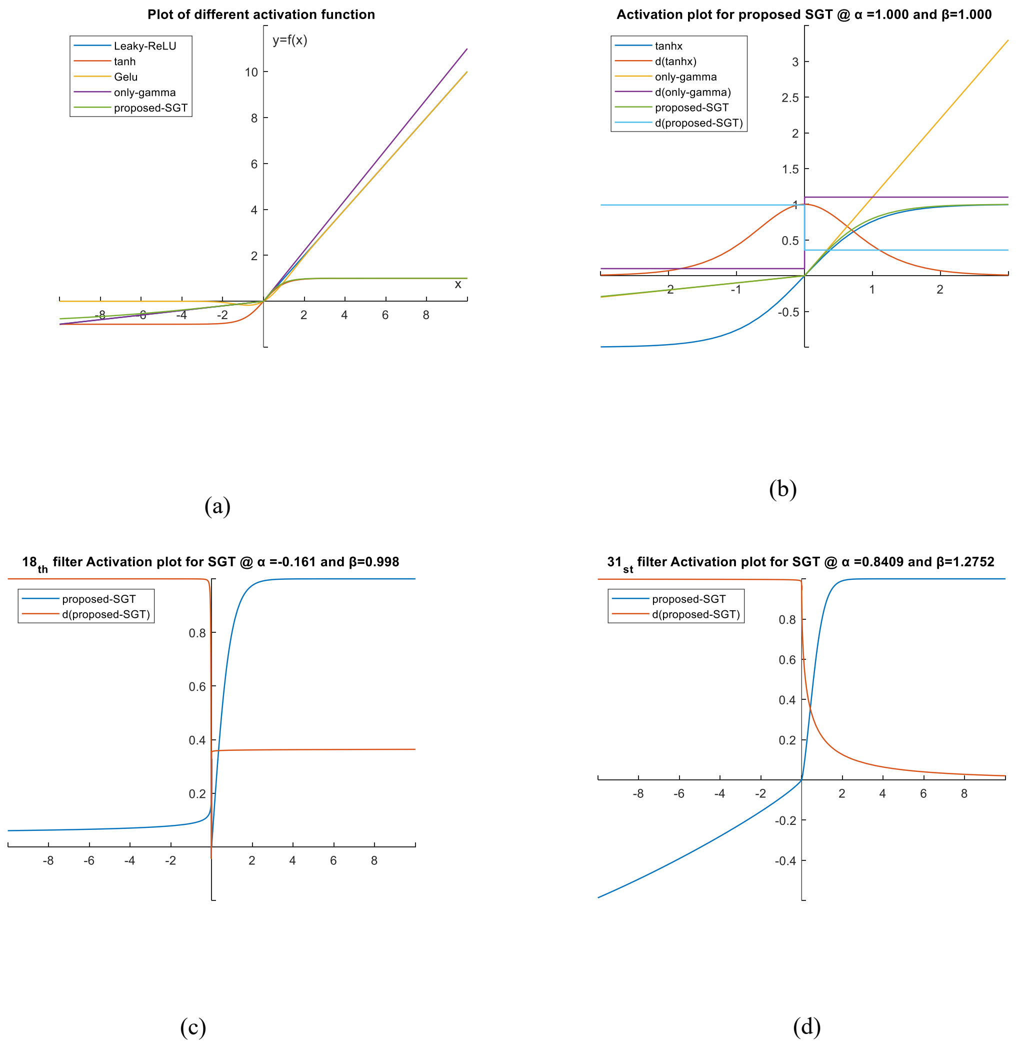


Figure 2.5: (a) Activation function plot for input *x* and *f(x)* along with other popular activation functions near x=0. (b) Activation (proposed-SGT) and first-order derivative (d(proposed-SGT)) plot with both exponents equal to 1 using a combination of gamma correction ('only-gamma') and hyperbolic tangent ('tanh') to illustrate the need for thresholding and squashing function. (c) Actual activation plot for the trained network in 18th filter (out of 64) in layer 4.

CHAPTER 3

Proposed Methods

3.1 Parameter initialization for divNet Architecture

Let's assume that the MRI/PET has a 64×64×64 matrix represented by I (i.e., $I = [I_{x_i y_i z_i}]_{i=1 \text{ to } 64}$). In total, this will result in 262,144 gray-scale values, which is the numerical representation for the 3D image. Since the input is a 3D MRI, we can call each of these values a voxel, not a pixel, with values for x, y, and z coordinates.

Each voxel mathematically assigns three coordinates, for easy representation the single vector notation v where, $v = [I_{x_i y_i z_i}]$ is used to make the computation simple. Let us consider the first convolution in the first layer as in equation (3.1). Here, b_1^1 and $w_{N,1}^1$ represent the initial bias and the weight of the first convolution kernel in the N^{th} filter, computed from the initialization algorithm. Note ⊗ represents element-wise multiplication. The window of the convolution operation then keeps on moving according to the stride size. To reduce this mathematical expression, it can be written with shorter terms as in equation (3.2). For each node of the 3D convolution filter:

$$[x_1^1, x_2^1, x_3^1, .., x_{64}^1] = [b_1^1, b_2^1, b_3^1, .., b_{64}{}^1] + [v_1, v_2, v_3, .., v_9] \otimes [(w_{1,1}^1, w_{1,2}^1, w_{1,3}^1 \ldots\ldots w_{1,9}^1)] , \quad (3.1)$$

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv._3 \left(w_{ik}^{l-1}, s_i^{l-1}\right) , \quad (3.2)$$

where, $conv._3$ is a regular 3-D convolution without zero paddings on the boundaries. Following equation (3.2), x_k^l is the input, b_k^l is the bias of the k^{th} neuron at layer l, and s_i^{l-1} is the output of the i^{th} neuron at layer l–1. w_{ik}^{l-1} is the kernel (weight) from the i^{th} neuron at layer l–1 to the k^{th} neuron at layer l. $conv._3$ represents an element-wise multiplication of the [3×3×3] kernel size. For the very first convolutional layer, the input s_i^{l-1} is the 3×3×3 matrix of the image pixel value (maybe normalized) that is scanned by a window of the same size.

When represented in a matrix/array or a discrete form, the N-dimensional convolution for the discrete, N-dimensional variables A and B can be defined with equation (3.3):

$$C(j_1, j_2, \ldots..j_N) = \sum_{k_1} \ldots.\sum_{k_N} A(k_1, k_2, \ldots., k_N) B(j_1 - k_1, j_2 - k_2, \ldots..., j_N - k_N) = conv._N (A, B) \quad , \tag{3.3}$$

Each $k_i$runs over all of the values that can lead to legal subscripts for A and B. Thus, the 3D convolution runs as follows. The layer convolves the input by moving the filters along the input vertically and horizontally. Afterward, it computes the dot product of the weights and the input, and then it adds a bias term. As the filter moves along the input, it uses the same set of weights and the same bias for the convolution; thus, forming a feature map.

In the SGD algorithm, the filter weights during the optimization are iteratively updated as shown in equation (3.4) and equation (3.5), where W_l^tdenotes the weights in the l^{th} convolutional layer for the t^{th} iteration and E denotes the cost function (updated using backpropagation for minimizing the cost function) over a mini-batch of size N.

$$W_l^{(t+1)} = W_l^t + V_l^{(t+1)} , \tag{3.4}$$

where $V_l^{(t+1)}$ is calculated as in equation (3.5)

$$V_l^{(t+1)} = m.V_l^t - \gamma^t.\alpha_l \frac{dE}{dW_l} \ . \tag{3.5}$$

Pseudo-code 1	Pseudo-code 2	Pseudo-code 3
% Activated feature extraction of convolution layer from a single MRI '*I*' % For the trained network 'N' at layer '*l*' we have, Filter feature (F_l) = weights of the filters at layer '*l*' with a size $k\times k\times k\times f$, where $k\times k\times k$ is the size of the filter and f the number of filters at layer '*l*' Initial activated feature (A_1) =*conv* (F_l, *I*). Here, *conv* performs the basic 3D convolution as described in section III. This A_1 goes through batch normalization and max-pooling to downsample its size and pass through the activation layer. Similarly, *Activated feature* (A_l) =*conv* (F_l, A_{l-1}) of size $k\times k\times k\times f$ [A_{lmax}] = maximum value of A_l Activated feature for Visualization= A_{lmax} of size $k\times k\times k$ is resized to 64×64×64 and each slice viewed separately in the 2D domain. When visualized, the clearer the visual, the better the features that are learned. The difference between the images in each MRI type represents the difference in the pattern of the MRI, and the complexity of the discrimination increases with the increasing layer number. This is used when comparing the output for the four different architectures as presented in Figure 4.2.	% Activated feature visualization of the FCL from the whole test set using the T-SNE projection % For the trained network 'N' at layer '*l*' similar operation to Pseudocode 1 is performed for the FCL so that, *FCL features* (*FCL_l*) = weights of FCL with size T× S, where T is the number of test subjects, S=O×I, and O, and I represent the output and input, respectively, of the FCL at the l^{th} layer. Now, $FCL_{l\text{-}tsne}$=T-SNE (*FCL_l*), where T-SNE perform T-distributed Stochastic Neighbor Embedding to find the 2-dimensional feature matrix of size T×2, while performing feature reduction from the N dimension. $FCL_{l\text{-}tsne}$ is plotted in the x-y plane against its target class-color to visualize the discriminative pattern. The better the separation of the same-colored set, the better the classification. The inclusion of a few odd color data in a cluster leads to errors in the test set. This is used when comparing the output for the four different architectures as presented in Figure 4.4.	% Final layer FCL weights of the trained networks for direct visualization % The previous two pseudocodes are used to examine the property of a single trained network; however, to compare the networks directly, the easiest way is to plot the final FCL weights. For a network 'N' with '*l*' as the last FCL layer and the number of classification categories 'n' FCL_l = Weights of the FCL_l of size O×n, where O is the output size of the penultimate FCL, which is 100 in our case, 'n' is the output size of the final FCL at layer l, which is equivalent to the number of classes (here n=3). FCL_l is an O×n matrix, which is simply plotted in the X-Y plane as a linear graph. By having more distance between the three lines, it becomes a better classifier. This is used when comparing the output for the four different architectures as presented in Figure 4.6.

Here, α_l in equation (3.5), is the learning rate for the l^{th} layer, m is the momentum due to the previous weight update in the current iteration, and γ is the scheduling rate that decreases the learning rate for the completion of each epoch. If α_l=0, then this depends on the value of *l*. All of the layers from 1: *l* is not updated in terms of their weight; hence, the weights are transferred in the final version of the trained model.

3.1.1 Parameter training

$$Total_{error(E)} = E(y_1^L, \ldots y_N^L) = \sum_{i=1}^{N_L}(y_i^L - t_i)^2 \ , \tag{3.6}$$

this error in equation (3.6) is a mean squared error, which is obtained by adding the MSE value of the deviation from each of the samples (i.e., training data (t_i) from the predicted value (y_i^L)). Here, the upper subscript L denotes the output for the final layer. Based on the obtained error (E), backpropagation (BP) is performed to update the weights for each parameter as in equation (3.7) [32]:

$$\frac{\partial E}{\partial w_{ik}^l} = \frac{\partial E}{\partial x_k^{l+1}}\frac{\partial x_k^{l+1}}{\partial w_{ik}^l} = \frac{\partial E}{\partial x_k^{l+1}} y_i^l \ \ , \tag{3.7}$$

Here, for the output of the x_k^{l+1} filter, 'k' is the number of filters in the l^{th} layer, and the weights of the previous layer '$l+1$' give the output y_i^l of the l^{th} layer during the BP. Similarly, the bias is also updated as equation (3.8):

$$\frac{\partial E}{\partial b_k^l} = \frac{\partial E}{\partial x_k^l}\frac{\partial x_k^l}{\partial b_k^l} = \frac{\partial E}{\partial x_k^l} \ \ , \tag{3.8}$$

Here the term $\frac{\partial x_k^l}{\partial b_k^l} = 1$ as x_k^l is unaffected by b_k^l. And for the final error, it is written for the whole length of 1 to $l+1$ layers; hence, it can be summed up as follow for N number of filters in the $l+1$ layer to obtain *y* in the l^{th} layer as in equation (3.9):

$$\frac{\partial E}{\partial y_k^l} = \sum_{i=1}^{N_{l+1}} \frac{\partial E}{\partial x_i^{l+1}}\frac{\partial x_i^{l+1}}{\partial y_k^l} \ . \tag{3.9}$$

During training, we need to backpropagate the gradient of the error ∂E through this transformation and compute the gradients with respect to the parameters as the BP transforms.

All experiments were conducted using MATLAB R2019a academic software on Windows 10

OS. Network models were trained on NVIDIA GeForce RTX 2070 GPU with 24 GB of memory and tested in Intel® Core™ i5-9600K CPU @ 3.70 GHz with 32 GB of memory. The trained mat file will be provided to researchers upon request to the authors.

3.2 Proposed GAP normalization layer

3.2.1 Architecture and training

Let's consider the initial output from the first convolution layer as in base architecture (see Table 4.1), which acts as input for the proposed first GAP i.e., g1 layer. This input can be represented as a 4D array for 2D CNN (5D array for 3D CNN) of its size as X = [227, 227, 32, 64] = [image_row, image_col, channel_size, minibatch_size]. Here the input X contains 2D images of size 227×227 each from 32 filter outputs, i.e., 227×227×32 with 32 different activated images for the same image, and the last dimension i.e., 4th dimension signifies the minibatch number for training. Each minibatch signifies different images for multiple classes so in total 64 different input images, with 105.53M pixels/weights as batch input for forward propagation in the first GAP layer. Similarly, for the 2nd layer, the output size after pooling is reduced to 113×113 so the output from the 2nd convolution layer is 113×113×64×64 (64 filters in 2nd convolution), so a total of 52.3M weight inputs for the 2nd GAP layer. To make it clearer, input X or 1st GAP input can be represented as the following block-matrix in equation (4.2), where X_b^n represents the activated image of n^{th} filter in b^{th} mini-batch of training:

$$X = \begin{bmatrix} X_1^1 & \cdots & X_1^{32} \\ \vdots & \ddots & \vdots \\ X_{64}^1 & \cdots & X_{64}^{32} \end{bmatrix}, \tag{4.2}$$

Here, the mean value is computed based on its 3rd dimension i.e., the number of filter (or channel, for 3D CNN it is calculated on the 4th dimension) hence can be represented as a matrix:

$$Xm = \begin{bmatrix} X_1^1 & \cdots & X_1^{32} \\ \vdots & \ddots & \vdots \\ X_{64}^1 & \cdots & X_{64}^{32} \end{bmatrix} = \begin{bmatrix} X_1' \\ \vdots \\ X_{64}' \end{bmatrix}, \tag{4.3}$$

Here, the column matrix on the right side comprises the mean value of each batch from different training MRI samples. The dimension now for = [227 227 1 64] which gives the averaged value of 'same images' activated 'differently' with convolutional filters. It is an empirical mean value of all channels so contains 64 different images for each batch updated during training, and is not used post-training, hence an empirical mean. Similarly, standard deviation is also calculated for Xs = [227 227 1 64]. Now we re-center and re-scale X using Xm and Xs as, $Xf = (X - Xm)\,/Xs$ which produces the averaged mean-centered and $1/Xs$ scaled output, Xf. Please note all the arithmetic operation in the matrix is an element-wise identical dimension operation. Hence the size of Xf = [227 227 32 64]. Here in equation (4.3), X_1' is the mean from the activated output of the first training images $X_1^1, X_1^2, \ldots\ldots X_1^{32}$ and so on. Hence, the variance is nominally very less, i.e., in general, $var(X_b^{'}) \approx var(X_b^n)$. In BN, once the training is finished each BN layer has 'n' number of trained mean and variance per activation stored in the trained network, which is later used to normalize the input during prediction. However, in our method, the layer does not store the trained mean and S.D instead, during prediction, the weight of convolutional kernels is passed below so the empirical mean for each image is calculated from each channel output, hence no need for pretrained mean and S.D.

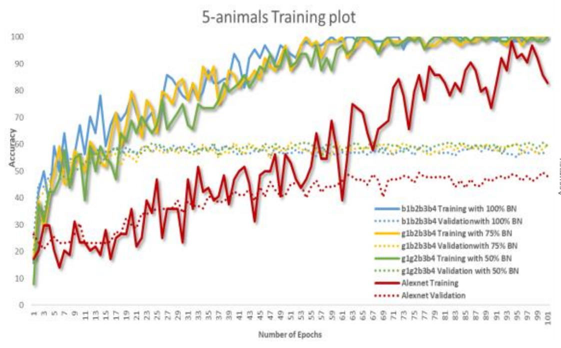
Channel normalization [112] in a CNN standardizes each channel independently for every training example, and scales and shifts the resulting input with a (learnable) scalar. It can be compared to instance normalization [113] and BN for a single training sample. In our case Xf is the normalized-unsharpened version of its original minibatch input, which is the shifted and scaled version of the input X. α, β works as a separate standardizing factor for the input and the unsharpened version respectively. However, the pretrained value of α, β, and γ scales each channel separately. $Xf$is now spatially correlated with a Gaussian kernel, i.e., filtering is done with the kernel of a size equivalent to the size of the previous convolution filter (here 3×3×3) for the first GAP layer as in equation (4.4), where h_{www} represents the Gaussian filter weights for the 3D kernel, calculated as equation (4.1) for i, j and k as w = 1 to 3 or 5. For the second GAP normalization i.e., g2 layer in g1g2b3b4 architecture, the used Gaussian kernel is obtained via convolution of the discrete Gaussian kernel with each other to get a second order filter response of the second normalization layer, distinct than the first normalization layer. Please note that 3D kernels are used in all GAP layers to adapt the normalization of feature in the activation map (the 3rd /channel dimension) and the filtering kernel moves throughout the whole Xf vector. In equation (4.4), the symbol $'o'$ represents the correlation operation which is similar to spatial filtering operation in image processing. Mathematically correlation process is similar as convolution in the time domain, except that the signal is not reversed before the multiplication process. The idea here is to filter all images in a replicated manner so that all 32×64 images are included. However, the blurring or smoothing effect works differently for each image from $X_1^1, X_1^2, \dots \dots X_1^{32}$on the first batch and so on. The obtained Gaussian images can be represented in a matrix as in equation (4.4), where each Xg_b^n represents an image after the Gaussian filter. The applied Gaussian kernel is the same size as the preceding convolution kernel to perform linear correlation. Hence, the concept of combining normalization and activation as a single procedure seemed not practicable, so we require an extra non-linear (activation) function to support the classification.

$$Xg = \begin{bmatrix} h_{11w} & \cdots & h_{1ww} \\ \vdots & \ddots & \vdots \\ h_{ww1} & \cdots & h_{www} \end{bmatrix} o \begin{bmatrix} Xf_1^1 & \cdots & Xf_1^{32} \\ \vdots & \ddots & \vdots \\ Xf_{64}^1 & \cdots & Xf_{64}^{32} \end{bmatrix} = \begin{bmatrix} Xg_1^1 & \cdots & Xg_1^{32} \\ \vdots & \ddots & \vdots \\ Xg_{64}^1 & \cdots & Xg_{64}^{32} \end{bmatrix} \quad , \tag{4.4}$$

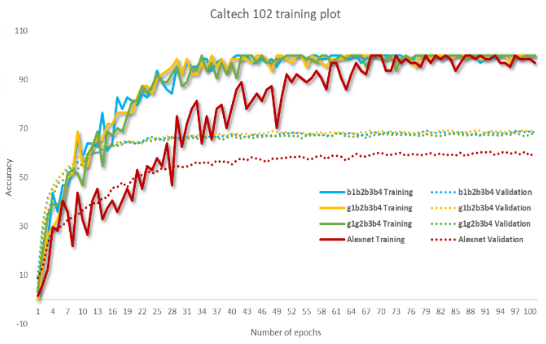
And the mask is obtained as $Xmask = X - Xg$ that subtracts the normalized Gaussian signal from its original version, for each channel output without inheriting the batch properties. Hence finally we forward the output from the layer as:

$$Z = \alpha X + \beta Xmask + \gamma \ , \quad (4.5)$$

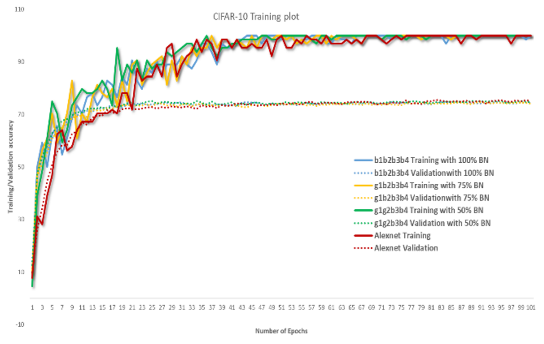
Here α, β and γ are learnable parameters each of size [1 1 N], which have unique values for 'N' filters and help to optimize the output value of Z. α, β and γ are initially selected between 0 and 1 and acts as scaling co-efficient to control the gradient outputs during backpropagation of weight update. And the second parametric term in equation (4.5) i.e., β is initially less than 1, however with weights update during backpropagation, the value tends to be $\beta > 1$ in which case acts as a high boosting filter, emphasizing the contribution of unsharp masking, and when $\beta < 1$ the sharpening mask is more emphasized (please see figure 3.2). The other learnable parameters α and γ are also updated during backpropagation, α works as scaling coefficient for making output equivalent as $(\beta.Xmask)$ and γ works as bias with no effect in the layer gradient loss. Figure 3.1 shows the result of experiments. In all the graphs training accuracy reaches convergence (100%) faster with 100% BN whereas the validation curve does not follow the training curve for a longer time, which means the weight update process ends sooner and cannot generalize better. In the case with 50% BN and 75% BN, the training curve achieves convergence lately and so does the validation. This helps the network to update weight slowly (at the same learning rate) and hence might reduce the overfitting (although we have a clear case of overfitting due to a large gap in training and validation accuracy in all cases, better architectures might change the result, however in this case the 'divNet' base architecture is used to test the proposed idea). Here, the CNN with GAP layers tends to reach 100% accuracy slowly so that validation accuracy is addressed for a longer time whereas with BN layers the training accuracy shoots up quickly causing a higher gap (coincidently we have named GAP for the proposed layer) between the training and validation layer during the early stages of training. Hence the overfitting problem is still not entirely tackled in batch normalization.



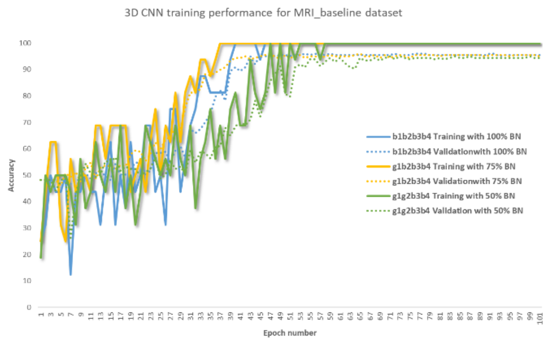
(a)



(b)



(c)



(d)

Figure 3.1: Training and validation accuracy curve using different normalization schemes for same training environment for (a) 5-animals dataset (b) Caltech-102 (c) CIFAR-10 (d) 3D MRI_BASELINE.

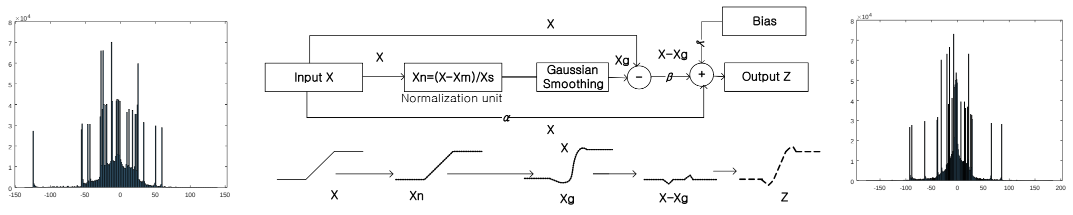


Figure 3.2: Schematic representation of proposed layer, along with input and output histogram for comparison. The input signal is represented as a ramp signal to demonstrate the edge detection process. However, in our experiment input X to the layer is the activated image matrix from the preceding convolution layer. The input passes through the normalization unit to produces a scaled and shifted version of X having a narrow range of feature values. Later, the Gaussian smoothing function transforms the feature vector Xn in a weighted-average fashion to produce a sharpened version of images i.e., Xg. The difference of X and Xg produces a masking vector Xmask, which is again added with the original X to produce Z. The learnable parameters α, β, and γ scales X, Xmask

and offset respectively.

SGD is the training algorithm to update the learnable parameters viz. weights, bias, offset, coefficients) values computed using a mini-batch. Instead of using the whole training set error at once as in standard ones [93], [96] here the loss is calculated on a mini-batch set by updating the network's parameters toward a negative gradient of loss at each iteration as in equation (4.6)

$$w_l^{t+1} = w_l^t - \gamma^t . \alpha_l \frac{dL}{dw_l^t} + r(w_l^t - w_l^{t-1}), \tag{4.6}$$

Here, weights or bias or offset w_l^{t+1} update in layer 'l' at each iteration $'t+1'$, uses the weights of the previous iteration w_l^t. α_l is the learning rate hyperparameter for the parameters of layer l, kept at value>0, and initially 0.001 in our experiments. Since the SDG is used with the momentum it lowers the oscillation of the parameter weight update and finds the path of steepest descent towards the optimal value. For this, the hyper-parameter r known as the rate of momentum is set for 0.95. Additionally, the negative term represents the gradient of the loss function in the layer updated as in back-propagation after every epoch.

The backpropagation calculates the derivative of loss with respect to (w.r.t) all trainable parameters in CNN. The layer gradient loss w.r.t to input X i.e., $\frac{dl}{dX}$ and other parameters *α, β,* and *γ* are updated as follow:

From equation (4.5) we get,

$$\frac{dl}{dX} = \alpha . \frac{dl}{dz} + \beta . k . \frac{dl}{dz} , \tag{4.7}$$

where $k = Xmask/X$, from experiment, it was found that 'k' value, when used, produces a small gradient value causing vanishing gradient problem, hence was selected to be 1. Hence for the GAP layer with 32 activations and 64 minibatch sizes we define gradient loss as follow:

$$\frac{dl}{d\alpha} = \sum_{b=1}^{64} \sum_{n=1}^{32} X_n^b . \frac{dl}{dz} , \tag{4.8}$$

$$\frac{dl}{d\beta} = \sum_{b=1}^{64} \sum_{n=1}^{32} Xmask_n^b . \frac{dl}{dz} \ , \tag{4.9}$$

$$\frac{dl}{d\gamma} = \ \sum_{b=1}^{64} \sum_{n=1}^{32} \frac{dl}{dz} \ . \tag{4.10}$$

Training in minibatch has a substantial impact in achieving convergence time, i.e., higher minibatch size makes the training quicker by reducing the number of iterations per epoch, on the other hand it also impacts the training accuracy. Hence, in the proposed method also, training of network is done in mini-batches i.e., the entropy loss is computed based on mini-batch input. Though, the normalization process is not the batch normalization when the used normalization layer is GAP. In normalization generally, the whitening process is desired where the input is linearly transformed to zero mean and unit variance, which is also considered to eliminate the ill effects of the internal covariate shift. While activation process is similar to a filtering process where the activation function determines the weight output. Similarly in the proposed method, the Gaussian filter works as an activation function along with a learned parameter to create a normalized mask. This mask works as the additional extracted feature ($\beta.Xmask$) with the original input so that the original signal is slightly boosted with its filter mean responses. Hence, later when linearly added to the original input adds the value according to the mask. If the masked value is only used, we would lose the entire input image property. Hence mask is included to the original input to bring a calculated variance, without losing the linear property of its input.

Few hyperparameters like learn rate drop factor, initial learn rate, learn-drop rate per epoch, etc. affect the training time and learning proportion however in the longer term, results are not significantly different. The lately achieved convergence slightly affects the testing result only. Fully connected layers act as a single-layered feed-forward network with all parameters connected from input to output. Because of this nature, FCL is often blamed for triggering overfitting in the DNN, hence potentials regularization techniques like dropout are used in between them [97][101].

3.3 Proposed SGT activation and training process

The proposed SGT activation is performed in two steps:

Step 1: $f(x)= y = ax^{\alpha}$ for $x<0$ and bx^{β} for $x>0$ (5.1)

Here the first step is getting the gamma corrected version of input x as in equation (5.1). x is an input defined by a 4D matrix/Tensor as X_l with each pixel/feature value X_n^b for 'b^{th}' batch and 'n^{th}' filter in layer 'l'. a and b are constant scaling factors that were set manually. For n filters, we have n values of learnable parameters (i.e., α or β) which implies that for all the different (or same)-class images belonging to the same mini-batch, the value of exponent (α and β) remains the same, whereas the value of exponents is different for the same-class images in different channels, hence are activated differently in each channel as shown in matrix representation in equation (5.2), ^ signifies operation performed in column-column element wise exponential operation.

$$Y_l = a.\begin{bmatrix} X_1^1 & \cdots & X_1^b \\ \vdots & \ddots & \vdots \\ X_n^1 & \cdots & X_n^b \end{bmatrix} \wedge \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = a.\begin{bmatrix} {X_1^1}^{\alpha_1} & \cdots & {X_1^b}^{\alpha_1} \\ \vdots & \ddots & \vdots \\ {X_n^1}^{\alpha_n} & \cdots & {X_n^b}^{\alpha_n} \end{bmatrix} = \begin{bmatrix} Y_1^1 & \cdots & Y_1^b \\ \vdots & \ddots & \vdots \\ Y_n^1 & \cdots & Y_n^b \end{bmatrix} \quad (\text{for } X_n^b < 0)$$

$$= b.\begin{bmatrix} X_1^1 & \cdots & X_1^b \\ \vdots & \ddots & \vdots \\ X_n^1 & \cdots & X_n^b \end{bmatrix} \wedge \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} = b.\begin{bmatrix} {X_1^1}^{\beta_1} & \cdots & {X_1^b}^{\beta_1} \\ \vdots & \ddots & \vdots \\ {X_n^1}^{\beta_n} & \cdots & {X_n^b}^{\beta_n} \end{bmatrix} = \begin{bmatrix} Y_1^1 & \cdots & Y_1^b \\ \vdots & \ddots & \vdots \\ Y_n^1 & \cdots & Y_n^b \end{bmatrix} \quad (\text{for } X_n^b > 0) \qquad (5.2)$$

where $X_l = \begin{bmatrix} X_1^1 & \cdots & X_1^b \\ \vdots & \ddots & \vdots \\ X_n^1 & \cdots & X_n^b \end{bmatrix}$ is the input to the layer l.

Here a and b are scaling constants selected manually to be 0.1 and 1.1 respectively. It is done to behave slightly as a monotonic function when the exponents are equal to 1 and resemble the Leaky-ReLU function in the first step (please see figure 4.2(a)). Later in the second step, when passed through the hyperbolic tangent (both exponents as 1) function, the output for the positive part will resemble tanh, and for the negative part will partly resemble the Leaky-ReLU function (please see figure 4.2(b)). However, on changing the exponent value and sign, different activation plots can be generated as shown in figures 4.2(c) and 4.2(d). Here it should be noted that only using step 1 for activation might explode the activated value in the positive region and can lead to vanishing gradient in the negative region (please see 'only-gamma' plot in figure 4.2(b)) which causes computational difficulty in convergence during training. Consequently, a thresholding function with non-linear and symmetric property in positive and negative axis is required, for which the tanh function was selected. The learnable parameters α and β values work as a positive gamma corrector, hence the weight updates of value α and β are calculated from the partial derivative of equation (5.1) during backward propagation as in equations (5.3) and (5.4):

$$\frac{dl}{d\alpha} = \sum_b \sum_n 0.1 \times real(log_{10}X_b^n).real\left(X_b^{n\alpha}\right).\frac{dl}{dz} \quad , \ for\ X_n^b < 0 \tag{5.3}$$

$$\frac{dl}{d\beta} = \sum_b \sum_n 1.1 \times real(log_{10}X_b^n).real\left(X_b^{n\beta}\right).\frac{dl}{dz} \quad , \ for\ X_n^b > 0 \tag{5.4}$$

Please note when $X_b^n = X$ is negative and α is a rational decimal number, the resulting X^α becomes a complex number, in that case, only use the *real* part of the complex number will be used. The same is the case with $log_{10}X$ and X^β. Likewise, the absolute values of α or β are used in equations (4.2), (4.3) and (4.4) for getting positive exponents.

Step 2: *z=tanh(y)* or in matrix form as:

$$Z_l = real\begin{bmatrix} tanh(Y_1^1) & \cdots & tanh(Y_1^b) \\ \vdots & \ddots & \vdots \\ tanh(Y_n^1) & \cdots & tanh(Y_n^b) \end{bmatrix} = \begin{bmatrix} Z_1^1 & \cdots & Z_1^b \\ \vdots & \ddots & \vdots \\ Z_n^1 & \cdots & Z_n^b \end{bmatrix} \tag{5.5}$$

Here, since all the operations are an element-wise matrix operation, the matrix calculated using (5.2) is passed to matrix calculation as in (5.5), then the output matrix Z_l of layer l is passed into the pooling layer. For the layer loss $\frac{dl}{dX}$, first the derivative of Y_l with respect to (w.r.t) X_l is calculated using equation (5.6), so that the output Y' dimension matches exactly the dimension of the layer input i.e., X_l.

$$X_l \quad = \frac{dY_l}{dX_l} = 1.1 \times \beta.real\left(X^{\beta-1}\right) , \qquad \text{for } X \geq 0 \tag{5.6}$$

Then, the overall gradient loss $\frac{dl}{dX}$ is calculated through the output of this layer as the derivative of Z_l w.r.t Y', which is backpropagated to the former layers using equation (5.7).

$$\frac{dl}{dX} = \frac{dZ_l}{dY'}.\frac{dl}{dZ} = \frac{dtanh(Y')}{dY'}.\frac{dl}{dZ} = sech^2(Y').\frac{dl}{dZ} \quad . \tag{5.7}$$

Here, $\frac{dl}{dZ}$ is the loss back-propagated from the deeper layers. Since *z* = *tanh(y)* is used as a squashing function, the final output value of the layer is non-uniformly scaled before passing out to the next layer resulting in *z* being a non-symmetric function centered at zero. This is shown in figures 4.2(c) and 4.2(d), where d(proposed-SGT) shows the plot for the final output of the first-order derivative of the proposed function. For condition with exponents α and β both being 1, the activation layer behaves like tanh in the positive part and Leaky-ReLU in the negative part, whereas for the case of derivative, the first-order derivative is a constant so behaves exactly like Leaky-ReLU with output constant 0.3592 and 0.99006 for positive and negative part respectively. Such behavior was observed in few filters with β(positive)>α(negative) as in the 18th filter which seems to be constant output as in two different filters non-lineared at 0. However, since both α and β are channel-wise learnable parameters, the value is not the same for all the channels (please see figure 4.17). The final value of α and β were examined to be between -0.2 and 1.3, and rarely were the identical values. Regarding our experiment, in most of the filters, the values of both α and β were a positive rational number with decimals, and β being greater than α in the majority case. More discussion on this is done in the discussion section. In the case with β(positive)>α(positive), follows the graph as in 31st filter (please see graph figure 4.2(d)) where the gradients value for positive *x* gradually keeps on decreasing with the value of *x*, however, the rate of decrease is lower than the tanh derivate. This prevents gradients values from becoming infinitely small, whereas in the negative derivative part the value is roughly constant for all cases. Therefore, the network becomes less prone to the vanishing gradient or exploding gradient. It is to note that when the input

X, α, β becomes 0, it causes an indeterminate form as *Sech (0)* = ∞ also *log (0)* = ∞ in this case, we simply replace the value of the parameters as 0.001 to continue training. Few α, β values were recorded undefined still after the convergence (please see Figure 4.18), however, they can be ignored.

For training the network and optimizing the parameters Adam [147] optimization technique was used. It is a first-order gradient-based optimization algorithm to update parameters until it reaches convergence. The learnable parameter (w_t) (weights/bias/defined terms like α and β) during t^{th} iteration is updated using Adam optimization as follow:

$$w_{t+1} = w_t - \frac{am_t}{\sqrt{v_t} + \varepsilon} \quad , \tag{5.8}$$

where a is the learning rate constant-value kept at 0.001 in our case, ε is a very small regularization constant value (10^{-8}) used as offset to keep a non-zero denominator. An element-wise moving average of parameters gradients (m_t) and its squared value (v_t) keeps on being updated as in equations (5.9) and (5.10), where b_1 and b_2 are decay rates for m_t and v_t kept at 0.9 and 0.990 respectively.

$$m_t = b_1 m_{t-1} + (1 - b_1) \nabla E(w_t) \ , \tag{5.9}$$

$$v_t = b_2 v_{t-1} + (1 - b_2)[\nabla E(w_t)]^2 \ , \tag{5.10}$$

Here, $\nabla E(w_t)$ represents the first-order derivative of loss (E) for the parameter w_t, which is the cross-entropy loss i.e.

$$loss\ (E) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{K} t_{ni} ln\ (y_{ni}) \ , \tag{5.11}$$

where for N is the total numbers of training samples with K mutually exclusive labels and t_{ni} is targeted output, and y_{ni} is the predicted value with its natural log ((ln) calculated for nth sample belonging to ith class.

CHAPTER 4

Experimental Results

4.1 DivNet architecture experiments

4.1.1 Test on different CNNs

To define an optimal number of layers for the input of 64×64×64 3D scan, an initial layer of encoder i.e., Convolution-Batch normalization-ReLU-max-pooling was used. Later, the encoder blocks were further applied on the L2, L3, L4, L5, and L6 layer consecutively as shown in Table 4.1. In L6, the final feature size from the sixth convolution was [2 2 2] for each of the 64 filters. This means that the filter kernels have only two pixels in length for each filter; hence, expanding this to the L7 layer would be an impractical idea and will eventually reduce the features. Hence, seven convolutions-based architecture was not experimented. Table 4.1 shows the result of classification on these layer-wise CNN, whereas Table 4.2 presents the result of classification using four different architectures based on the reception area i.e., window size of the convolution kernel. Similarly, the training and validation graph was also studied to observe, how the architectures affect the training and also help to better understand the convergence process of each CNN, Figure 4.1. Correspondingly, to understand the extracted features, from each convolution layer, a single MRI from each target domain was passed and the feature was observed as in Figure 2.4. On minute observation we could find the difference in the lines, edges, intensities, and other patterns based on the class domain. Moreover, FCL layers were visualized using t-SNE projection as in Figure 4.3 for each architecture to support our finding. Here, the features were visualized for the whole test set, thus this will help us to judge which architecture has segregated the feature in a better way. Finally, the results from different hyper-parameter settings and datasets are tabulated in Table 4.3 and Table 4.4 respectively.

4.1.2 Why diverging architecture?

The size of convolution filter determines the scanning window during the convolution and this window can be analogised as the reception area. The filter size increases by two strides in each consecutive layer so that the feature extracted will be sequentially extracted at a low level, an intermediate level, and a high level with a greater area of reception for the successive layers. The low-level features are extracted from the 3×3×3 filter window and it is max-pooled by the 2×2×2 windows with a stride of one from the first convolution layer (i.e., conv_1 to max-1) [Figure 2.2 (b)]. We can call this a diverging network because that the size of the filter kernel keeps on increasing with an increase in the step size or the stride. However, the number of filters in each layer is identical (i.e., 64) to maintain the channel size for the input of 64×64×64. Once the layer deepens, we can gather the features by increasing the window size for each layer. Consequently, the max-pool stride is also increased to lessen the redundancy in the feature. Conversely, the area of the reception keeps on diminishing with an initial filter size of 9×9×9 in the converging network, whereas in the equivalent architecture a uniform kernel size of 3×3×3 is used in each convolutional layer. The details of the architecture and the results of the experiment after training and testing are highlighted in Table 4.2, which includes the parameters in the second column.

4.1.3 PET or MRI or both?

To find the effect of the size of the training material, the L4 diverging network was trained with a variety of datasets and the results are shown in Table 4.4. The used MR images and PET images were all obtained from patients of ADNI BL visits obtained under the ADNI 1 project [71]. We used 3D scans of T1 weighted structural MR images of whole-brain; normalized and processed using ADNI pipeline also few scaled (listed in Appendix), whereas PET scans were also obtained from ADNI BL; processed for smoothing, co-registration, and few standardized (listed in Appendix). Our experiment showed that MRI is a better imaging modality than PET for 3D CNN classification. When the network is trained with the smallest dataset including MRI1 (see Table 4.4, 5th column for the type), the network gets under-fitted; hence, the testing accuracy was low at

74.5%, which is slightly lower than the validation accuracy. However, the training achieved convergence as the accuracy reaches 100%. The same network when trained with the BASELINE_MRI data (type MRI2, see Table 4.4) under the same environment achieved the highest testing accuracy of 94.5%. The reason behind the increased accuracy may be due to the higher scans per patient ratio (SPR), which decreases the variability for each scan and loses its generality in the network. The PET scan performed the worst in the L4 divNet with increased training time. The BASELINE_PET_SMALL dataset, PET1, has a testing accuracy of only 66.34%, whereas the bulkiest PET dataset (i.e., BASELINE_PET_ALL, PET2) testing accuracy reached only 50.21%, along with difficulties in achieving convergence with 100 epochs and GPU training time almost three times of PET1 though it is ten times bigger in size than PET1. Finally, the MRI2+PET1 datasets were merged and trained in a single network however, it could only reach a 90% training accuracy after convergence and reached the testing accuracy of up to 82%. As a result, it seems like MRI is a better choice for CNN, and PET only has a complementary role for the AD prediction. It is worth mentioning that the PET image is visually not so discriminative by the target class in comparison to the MRI image (see Figure 2.1), which may have resulted in the MRI's better performance.

4.2 Experimental result for divNet architecture

The results of all experiments are presented in the tables and figures below.

4.2.1 Test on different layered CNN

Table 4.1 presents the results from the diverging architecture-based configuration with different layers counts, starting with two convolution encoding layers to six. The parameter column lists the filter size, number of filters, max-pool filter size, stride, and FCL input and output number as indexed in each row. Training accuracy reached almost 100% for each configuration, whereas the validation and testing accuracy start dropping after the L4 layer. This could be the optimal case as plotted in training and validation loss against the epoch numbers as shown in Figure 4.1(a) to 4.1(f).

Table 4.1: Training and testing results for the diverging architectures with changing number of layers as specified in the parameters column. Here, C [W*W*W N, S] represents a convolutional layer with N filters sized W each dimension, moving by stride S and N biases. TC [W*W*W N, S] represents a transposed convolutional layer with N number of filter sized W each dimension, moving by stride S and N biases. BN [N] represents the batch normalization with an offset of N and N scale values as learnable parameters. R represents the ReLU activation. M [W*W*W S] represents the max pooling with W kernels with a stride S, FC[O*I] represents the fully connected layer with input I and the output O. CT, D, S, and C represent the Concatenation, Dropout, SoftMax, and the Classification layer, respectively. The training pattern is shown in Figure 2.3.

Diverging Architecture Length	Parameters	Training Accuracy	GPU training Time (min)	Validation Accuracy (%)	Testing Accuracy (%)
2 layer conv (L2)	C[5*5*5 64,1] BN[64] R M[2*2*2 2] C[9*9*9*64 64,1] BN[64] R M[2*2*2 4] FC[1728*32768] D FC[864*1728] D FC[100*864] D FC[3*100] S C	99	778	93.4	94.26
3 layer conv (L3)	C[5*5*5 64,1] BN[64] R M[2*2*2 2] C[7*7*7*64 64,1] BN[64] R M[2*2*2 3] C[9*9*9*64 64,1] BN[64] R M[2*2*2 4] FC[1728*1728] D FC[864*1728] D FC[100*864] D FC[3*100] S C	100	664	91.88	94.66
4 layer conv (L4)	C[3*3*3 64,1] BN[64] R M[2*2*2 1] C[5*5*5*64 64,1] BN[64] R M[2*2*2 2] C[7*7*7 *64 64,1] BN[64] R M[2*2*2 3] C[9*9*9*64 64,1] BN[64] R M[2*2*2 4] FC[1728*1728] D FC[864*1728] D FC[100*864] D FC[3*100]S C	100	842	95.43	95.59
5 layer conv (L5)	C[3*3*3 64,1] BN[64] R M[2*2*2 1] C[5*5*5*64 64,1] BN[64] R M[2*2*2 2] C[5*5*5*64 64,1] BN[64] R M[2*2*2 2] C[7*7*7*64 64,1] BN[64] R M[2*2*2 3] C[9*9*9*64 64,1] BN[64] R M[2*2*2 4] FC[1728*64] D FC[864*1728] D FC[100*864] D FC[3*100] S C	100	786	93.4	92.91
6 layer conv (L6)	C[3*3*3 64,1] BN[64] R M[2*2*2 1] C[5*5*5 64,1] BN[64] R M[2*2*2 2] C[5*5*5 64,1] BN[64] R M[2*2*2 2] C[7*7*7 64,1] BN[64] R M[2*2*2 3] C[7*7*7 64,1] BN[64] R M[2*2*2 3] C[9*9*9 64,1] BN[64] R M[2*2*2 4] FC[1728*64] D FC[864*1728] D FC[100*864] D FC[3*100] S C	100	780	95.43	92.57

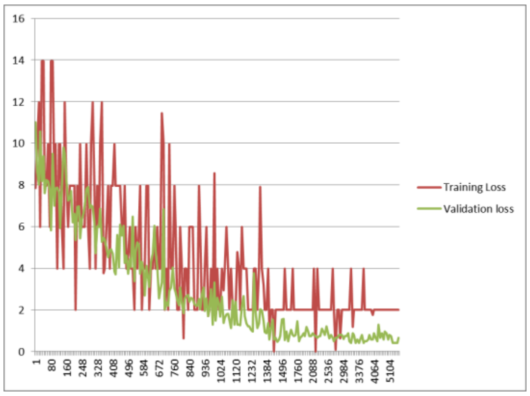


Figure 4.1(a): The training and validation loss (Y-axis) graph showed under each iteration (X-axis) of 100 epochs for the L1 convolution as presented in Table 4.1.

Remarks: The VL is much less than TL, which indicates a possible overfitting case

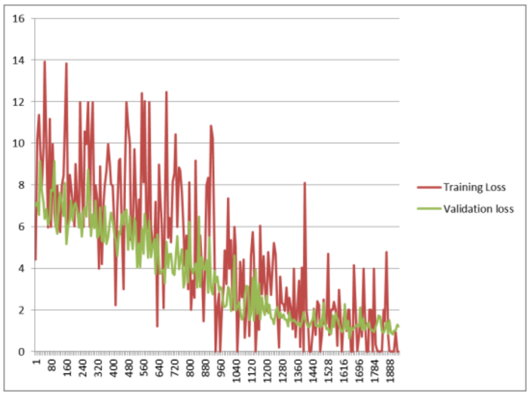


Figure 4.1(b): The training and validation loss (Y-axis) graph showed under each iteration (X-axis) of 100 epochs for the L2 convolution as presented in Table 4.1.

Remarks: The VL is less than TL, which indicates a possible overfitting case.

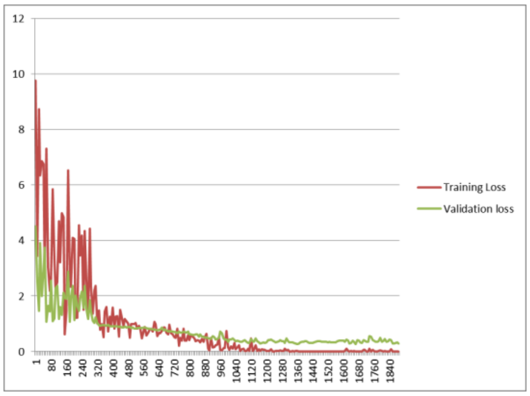


Figure 4.1(c): The training and validation loss (Y-axis) graph showed under each iteration (X-axis) of 100 epochs for the L3 convolution as presented in Table 4.1.

Remarks: The VL is higher than TL, which indicates a possible under- fitting case.

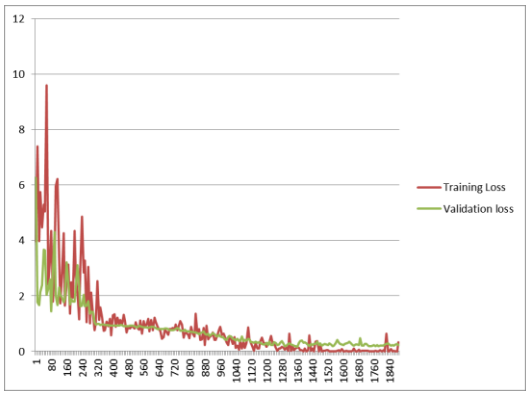


Figure 4.1(d): The training and validation loss (Y-axis) graph showed under each iteration (X-axis) of 100 epochs for the L4 convolution as presented in Table 4.1.

Remarks: The VL is slightly higher than TL, which indicates a possible optimal case.

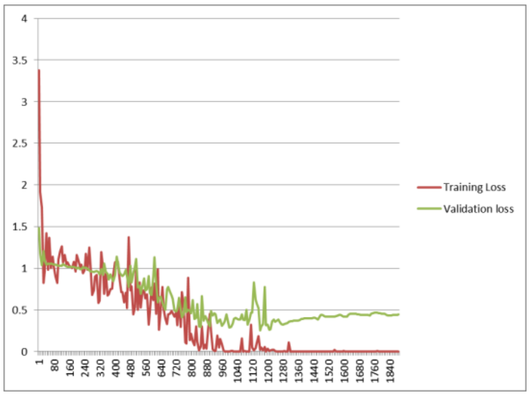


Figure 4.1(e): The training and validation loss (Y-axis) graph showed under each iteration (X-axis) of 100 epochs for the L5 convolution as presented in Table 4.1.

Remarks: The VL is much higher than TL, which indicates a possible under-fitting case.

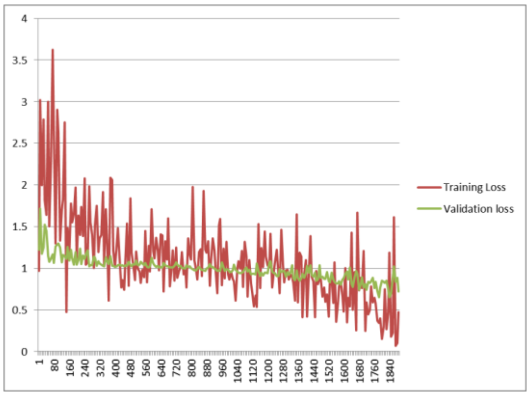


Figure 4.1(f): The training and validation loss graph (Y-axis) showed under each iteration (X-axis) of 100 epochs for the L6 convolution as presented in Table 4.1.

Remarks: The VL and TL both have higher values, which indicate a possible under-fitting case.

4.2.2 Test on different architectures

As discussed in section 2.4, the results using different architectures based on the reception area of convolving filter size i.e., the results from 4 architectures viz; U-net, converging, diverging, and equivalent is shown in Table 4.2.

Table 4.2: Test results using various types of architectures. The parameters are indexed as in Table 4.1. Here the ground matrix for testing all architectures model is $\begin{bmatrix} 63 & 0 & 0 \\ 0 & 91 & 0 \\ 0 & 0 & 142 \end{bmatrix}$.

Different Architecture	Parameters	Training Accuracy	GPU Training Time (min)	Validation Accuracy (%)	Testing Accuracy in the BASELINE_MRI (%)	Predicted Confusion Matrix (CM) for Testing
Encoder-decoder based (U-net) [68]	C[3*3*3 32,1] BN[32] R C[3*3*3*32 64,1] BN[64] M[2*2*2 2] C[3*3*3*64 64,1] BN[64] R C[3*3*3*64 128,1] M[2*2*2 2] C[3*3*3*128 128,1] BN[128] R C[3*3*3*128 256,1] R C[3*3*3*256 256,1] R C[3*3*3*256 256,1] R TC[2*2*2*512 512,2] CT C[3*3*3*768 256,1] R C[3*3*3*256 256,1] R TC[2*2*2*256 256,,2] CT C[3*3*3*384 128,1] R C[3*3*3*128 128,1] R TC[2*2*2*128 128,2] CT C[3*3*3*192 64,1] R C[3*3*3*64 64,1] R TC[2*2*2*64 64 ,2] FC[100*786432] R D FC[512*1000] R D R FC [3*512]S C	100	3988 (20 Epochs)	48.73	41.81	9 32 22 15 47 29 36 56 50
Converging	C[9*9*9 64,1] BN[64] R M[2*2*2 1] C[7*7*7*64 64,1] BN[64] R M[2*2*2 2] C[5*5*5 *64 64,1] BN[64] R M[2*2*2 3] C[3*3*3*64 64,1] BN[64] R M[2*2*2 4] FC[1728*1728] D FC[864*1728] D FC[100*864] D FC[3*100]S C	100	1429	94.92	94.59	60 1 2 0 88 3 5 5 132

L4 Diverging (divNet)	C[3*3*3 64,1] BN[64] R M[2*2*2 1] C[5*5*5*64 64,1] BN[64] R M[2*2*2 2] C[7*7*7 *64 64,1] BN[64] R M[2*2*2 3] C[9*9*9*64 64,1] BN[64] R M[2*2*2 4] FC[1728*1728] D FC[864*1728] D FC[100*864] D FC[3*100]S C	100	842	95.43	94.59	58 3 2 0 86 5 1 5 136
Equivalent	C[5*5*5 64,1] BN[64] R M[2*2*2 1] C[5*5*5*64 64,1] BN[64] R M[2*2*2 2] C[5*5*5 *64 64,1] BN[64] R M[2*2*2 3] C[5*5*5*64 64,1] BN[64] R M[2*2*2 4] FC[1728*1728] D FC[864*1728] D FC[100*864] D FC[3*100]S C	100	790	95.94	93.92	55 1 7 0 85 6 4 0 138

4.2.3 Test for different hyper-parameter settings

Hyper-parameters play crucial role to reach an optimal case for the top performance of the network so we experimented with several initialization techniques, activation functions, and optimization algorithms to find the best case as shown in Table 4.3.

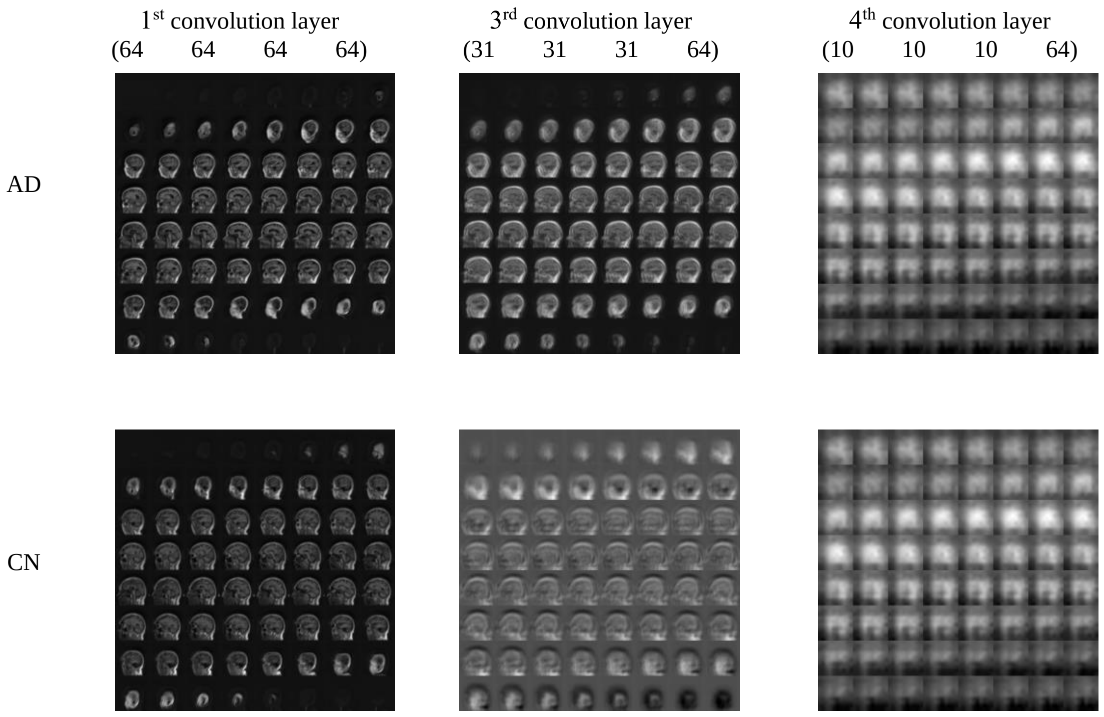
Table 4.3: Classification performance results for the BASELINE_MRI data; under a different hyper parameter setting that is investigated in the L4 diverging architecture as listed in Table 4.4.

Selected Architecture	Hyper parameter Description	Selected technique	GPU Training Time (min)	Training accuracy % (50%)	Validation accuracy % (20%)	Testing accuracy % (30%)
L4 Diverging	Initialization technique (Adam optimized, ReLU activated)	Xavier Glorot	842	100	95.43	94.59
		He	850	100	92.39	92.91
	Optimization (Glorot initialized, ReLU activated)	Adam	842	100	95.43	94.59
		SDG	844	100	93.908	92.91
	Activation (Adam, Glorot)	ReLU	842	100	95.43	94.59
		Tanh	850	100	94.42	92.23

	Leaky-ReLU	905	100	93.51	95.61

4.2.4 Figures for each architecture's convolutional transformation

Convolutional transformation is visualized using Pseudo-code 1; here we present Figure 2.4 for each class domain analysis, visualized using a single patient MRI scan. The number of features keeps on reducing from the former convolutional layer to the latter one. The result from the L4 diverging architecture network is presented in slice-view, scaled to 64×64 for better visualization.



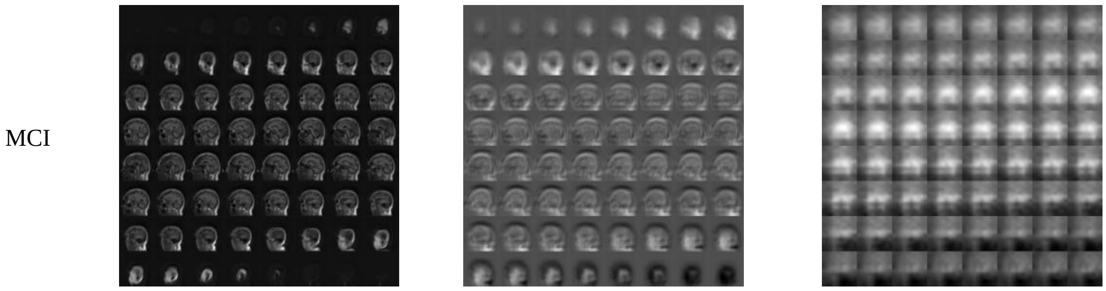


Figure 4.2: Convolution layer visualization of maximally activated feature using single MRI scan, original size resized to [64 64 64], using pseudocode 1, employed network is L4 diverging. Each convolution layer for a typical MRI of AD, CN and MCI category.

4.2.5 Test on different datasets

Although the selection of network architecture is finalized still the dataset size should be determined as it can heavily influence the network performance. So, we were concerned to know how the number of training material affects the testing accuracy. Hence, we completed few more experiments shown in Table 4.4. Demographic details and file type are justified in the Appendix.

Table 4.4: Results of the classification for the different dataset sizes using L4 diverging. This was tested on a variety of dataset sizes in MRI and/or PET imaging that ranges from small to large size datasets. The MRI1, MRI2 and PET1, PET2 type are detailed in the appendix.

Dataset type	AD MRI/PET count	CN MRI/PET count	MCI MRI/PET count	Included MRI/PET Type	Training accuracy (50%)	Training time (min)	Validation accuracy (20%)	Testing accuracy (30%)	Confusion Matrix (CM)	Ground Truth (GT)
BASELINE_MRI_SMALL	54	75	58	MRI1	100%	59	76.32%	74.55%	12 0 4 4 15 3 0 3 14	16 0 0 0 22 0 0 0 17
BASELINE_MRI	209	305	474	MRI2	100%	842	95.43%	94.59%	58 3 2 0 86 5 1 5 136	63 0 0 0 91 0 0 0 142
BASELINE_PET_SMALL	102	109	125	PET1	100%	99	66.67%	66.34%	22 6 3 6 20 7 5 7 25	31 0 0 0 33 0 0 0 37
BASELINE_PET_ALL	1165	1057	974	PET2	70-75%	3026	49.50%	50.21%	134 205 10 3 312 2 17 240 35	349 0 0 0 317 0 0 0 292

BASELINE_MRI+BASELINE_PET_SMALL	311	414	599	MRI2 +PET1	85-90%	1164	78.79 %	82.12%	55 1 37 0 94 30 0 3 177	93 0 0 0 124 0 0 0 180

4.2.6 Figures for each architecture's FCL t-SNE transformation

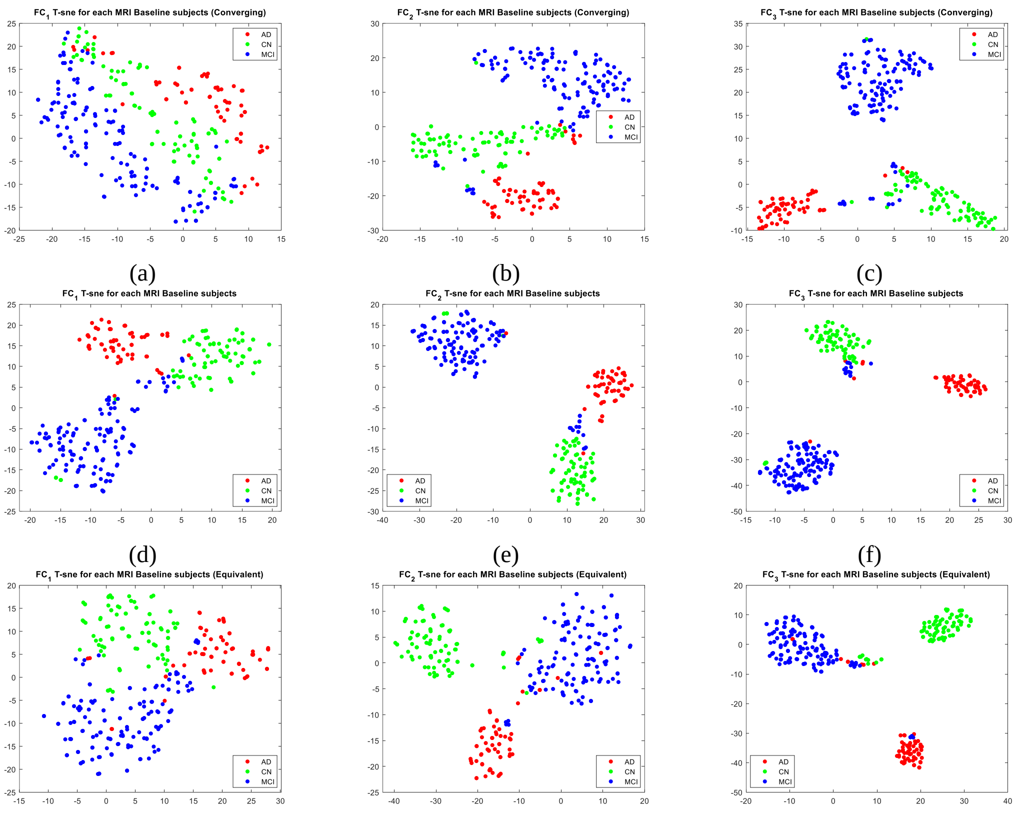
FC layers weights are visualized using T-SNE transformation as stated in Pseudo-code 2, the result of experiments from each architecture type is shown in Figure 4.3, where we have presented the class-wise representation of figures for the last three FCL used.

Architecture Type | FC1 | FC2 | FC3

Converging

Diverging

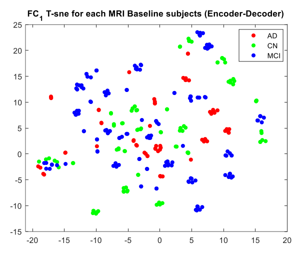
Equivalent



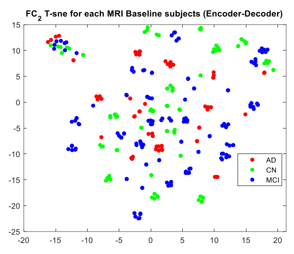
(a) (b) (c)

(d) (e) (f)

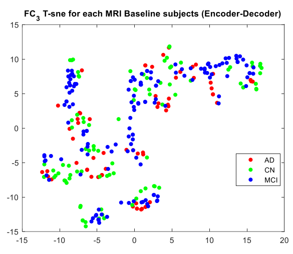
(g) (h) (i)



(j)



(k)



(l)

Figure 4.3: FCL feature visualization using t-SNE 2D feature projection for the different architectures during testing. The colored dots represent single MRI scan features from the test set in the first three FCL, namely FC1, FC2, and FC3.

4.3 3D CNN state-of-the-art comparison

Hosseini et al. [43] utilized a deeply supervised adaptable 3D CNN (DSA-3D-CNN) based on the autoencoder network for AD classification that explains feature maps for the various layers. The stated accuracy is 97.06% for the binary classification of the AD/NC MRI. The accuracy is from a 10-fold CV, which means that only one MRI in a batch of ten is used in testing, whereas the other nine are employed for training and validation. Hence, only 10% of the total image (i.e., 21 subjects) is used for testing [66]. Oh et al. [65] also performed 5-fold CV with a moderately sized dataset with an accuracy of around 84.5%. Evgin Foceri [62] and Gupta et al. [63] reported accuracies of 98.06% and 94.74% respectively, where they make use of data splitting and tested in 20% and 10% of the dataset respectively. Although the accuracy is higher, the SPR ratio is still high, which may produce a generalization error. Payan et al. [64] had an optimal performance for larger data size, with an accuracy of around 89.47% for three classes of AD/MCI and HC. However, here the testing ratio is only 10%, which may suggest the case of possible overfitting. They have trained 3D CNN using 5x5x5 patch-based thus not a whole MRI itself. Conversely, we tested using the whole MRI and PETs in different data sizes, splitting the data in 5:2:3 ratios for training, validation, and testing. Hence, the 30% untouched data when tested can give us a reliable result.

In Table 4.5, the term SPR is introduced, which indicates the use of multiple scans from a single patient, however not necessarily at the same time. As a result, multiple MRIs and PETs were acquired from a single patient for SPR greater than '1'; however, the image acquisition and preprocessing steps were different for each of the scans. A lower SPR value can bring variability in the dataset; therefore, the value of '1' indicates a single scan from a patient. This may eventually bring generality in the trained model; however, this can result in a low performance due to the constraint of the limited training material as in our case with the MRI scans, where the accuracy dropped to 74.55% φ from our best outcome of 94.5% (see Table 4.5) ξ. Later to check with the PET, we first trained it with a smaller database with scans from each unique patient (i.e., SPR=1); however, the results were poor. It was then tested with a larger PET database and a higher SPR. This also resulted in a low performanceς that led us to conclude that PET is not a good choice for image-based 3D-CNN classification. On further tests with PET+MRI as presented in the last row of table V, we found a moderate result that is merely due to higher true positives from the MRI scans than from the PET. Thorough experiments were performed with a different number of subjects to find the effect of data-size in both MRIs and PETs; hence we did not use the same number of patients.

Table 4.5: Comparison with other algorithms with 3D CNN based architecture.

Authors	Method	3D scans	# of patients	SPR	Testing Accuracy %
Evgin Goceri [62]	Sobolev Gradient based optimized CNN	Type: MRI CN: 568 AD: 570	CN: 130, AD: 185	4.36:3.08	98.06 (AD/CN)
Gupta et al. [63]	Sparse Auto encoder (SAE) based CNN	Type: MRI CN:1278 AD: 755 MCI: 2282	CN: 232, AD: 200, MCI: 411	5.50:3.75:5.55	94.74 (AD/MCI/NC)
Payan and Montana [64]	Sparse Auto encoder (SAE) patch-based 3D CNN	Type: MRI CN:755 AD:755 MCI:755	CN:755, AD:755 MCI:755	1:1:1	89.47 (AD/MCI/HC)

Hosseini Asl et al. [43]	Dsa-3d-CNN	Type: MRI -	CN:70, AD: 70	-	97.60 (AD vs. CN)
Oh et al. [65]	Inception auto encoder-based 3D CNN	Type: MRI -	NC:230 AD:198 sMCI: 101	-	84.5% (AD vs. NC) 74.07% (AD vs. sMCI)
Proposed divNet	Diverging CNN	Type: MRI CN:305 AD:209 MCI:474	CN:60 AD:65 MCI:87	5.08:3.21:5.44	94.59% (AD/CN/MCI) ξ
		Type: MRI CN:75 AD:54 MCI:58	CN:31 AD:28 MCI:48	2.41:1.92:1.20	74.55% (AD/CN/MCI) φ
		Type: PET CN: 109 AD: 102 MCI: 125	CN: 109 AD: 102 MCI: 125	1:1:1	66.6% (AD/CN/MCI)
		Type: PET CN: 1057 AD: 1165 MCI: 974	CN: 109 AD: 136 MCI: 337	9.69:8.56:2.89	50.21% (AD/CN/MCI) ζ
		Type: PET+MRI	CN: 414 AD: 311 MCI: 599	-	82.12% (AD/CN/MCI)

4.3.1 Performance-analysis and discussion

To study the proposed model performances listed in Table 4.5, we visualized the convolutional layer as well as the FCL with the help of Pseudo-code 1, 2, and 3. The convolution layers findings have been discussed earlier; here we will examine the FCL output. Figure 4.3 depicts the distribution of the features for the test image set, which consists of 296 scans that are separated layer-wise during classification from the first convolution to the last FCL. The classification performance of the converging and diverging architecture is the best out of the four selected architectures (Table 4.2). Even so on the basis of FCL patient-level visualization, as demonstrated in Figure 4.3, we see that the features for each class start to separate well in the diverging architecture than the converging one. From the first FCL FC1 to the third FCL FC3, the data visualization using t-SNE shows a better separation in the second case (i.e., diverging, see Figure 4.3). In figure 4.3 the feature starts to show a class-domain property from an FCL, and it is visualized by the start of the formation of the same-colored cluster. Based on the visual inspection, we determined that the diverging architecture-based features are better clustered and separated than the others, Figure (d)-(f). Meanwhile, there is poor separation in the case of the U-net-based architecture as shown in Figure (j)-(l). Here, the training environment and the training material used for training were all the same; the generated models are detailed in Table 4.2. The X-axis and Y-axis represent the values of the 1st dimension and 2nd dimension obtained from t-SNE 2D projection respectively. Figure 4.4 demonstrates the proposed divNet t-SNE feature visualization from the 1st convolution to the 4th convolution (i.e., from Figure (a) to Figure (d)). The features from similar groups start to segregate, and it can be distinctly visualized from the 1st FCL (i.e., FC1, Figure (e)). It continues until the last FCL (i.e., FC4), where only a few colored dots are found in the wrong cluster (Figure (h) near the green CN group and a few in the blue MCI group). This overlapped region may be due to the possible false positives or false negative predictions that are subjected to errors in the test set prediction. The X-axis and Y-axis represent the values of the 1st dimension and 2nd dimension obtained from the t-SNE 2D projection respectively. Similarly, based on the final FCL graph plotted as separate color curves for each cohort domain against the real weights of the final 100 parameters from the trained network without any projection (see Figure 4.6), shows a better demarcation between each colored graph than the 512 parameters from the U-net architecture. Afterward, we moved back to the training curve of these three networks to finalize the best performance as shown in Figure 4.5. It was observed that the validation loss is significantly higher than the training loss in the converging and equivalent architecture. The training plot of the converging architecture has a validation loss that is much higher than the training loss. This may cause a poor performance, which is similar in the case with an equivalent architecture. However, the validation loss is quite reduced in the diverging architecture; thus, making it the optimal choice. Here, the training material and the training environment are identical in all three cases.This indicates that the network can still be optimized, which was achieved with a diverging architecture and proper hyper-parameter selection.

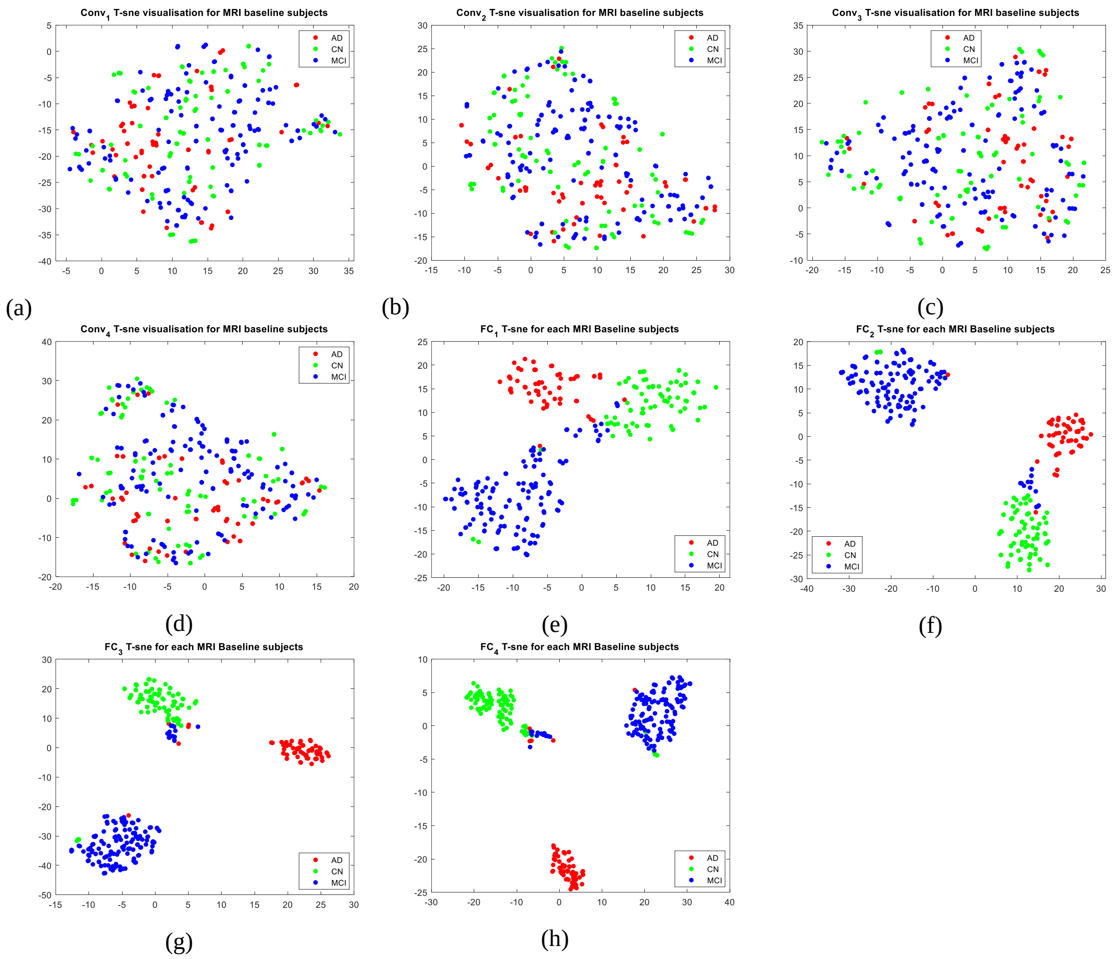
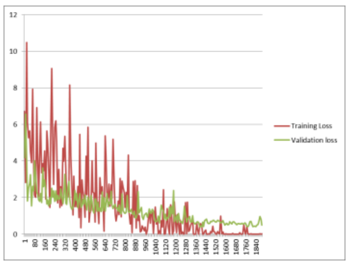


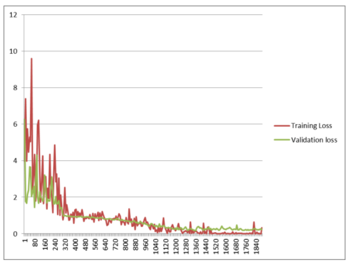
Figure 4.4: Feature visualization using t-SNE 2D projection for the L4 divNet for 296 test images from the BASELINE_MRI data. Each colored dot represents the feature of a single MRI of the indexed class.

Converging

Diverging

Equivalent





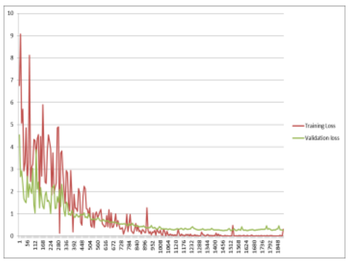
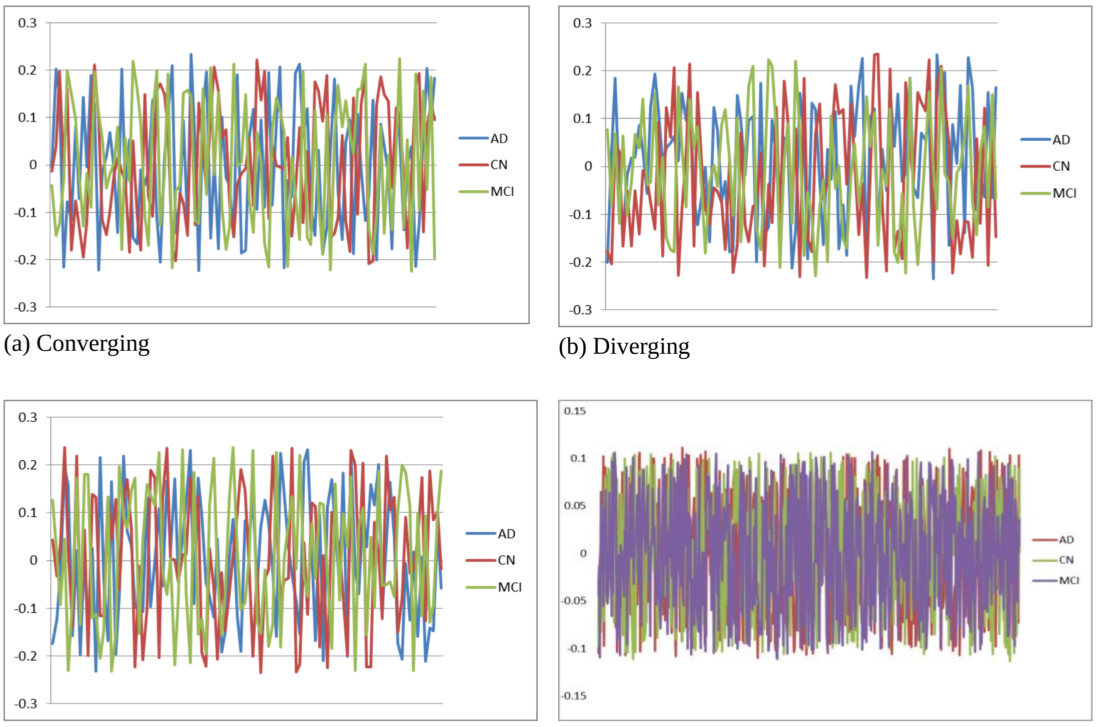


Figure 4.5: Training graph plotted against the training loss and validation loss in the Y-axis and the corresponding iteration number in the X-axis. By having more iteration numbers, the longer the epochs are.

4.3.2 Generalization and overfitting problem

If we study the recent architectures [43] [62] [63] [64] and their performance results, we discover that the reported precision and accuracy rate are very high, more than 90%. In MR-guided image acquisition, various technical specifications like acquisition instrument, contrast intensity, plane orientation, spatial positioning, correction method, registration template, and wrapping protocol can bring inconsistency in the MRI of the suspected class [70]. Hence, a neural network trained on one 'variety' of an MRI, may find it ambiguous to detect an MRI of the same target class, if this is acquired differently. This produces generalization error in the network. The generalization error is one of the primary challenges in medical imaging diagnosis. In this case, we have tested our network/model with other data from the ADNI, which we denoted as MRI_adapted. This is because it was partly adapted from [69] which differs in participants under the ADNI project. The MRI_adapted dataset was used only for testing of the generalization, which consists of 135 AD, 162 CN, and 134 MCI 3D scans; the testing results are presented in Figure 4.7. The other way to scrutinize could be with the visualization of the features. By extracting the better features, CNN will learn better. Similarly, overfitting is a contemporary part that comes with the generalization error. A non-generalized model learns 'too well' so that it only memorizes the training pattern that causes overfitting. Once we solve the overfitting problem, generality is also achieved.



(a) Converging

(b) Diverging

(c) Equivalent

(d) U-net

Figure 4.6: Final FCL weights values plotted in Y-axis directly for three target domains separately for each tested architecture using Pseudocode 3. X-axis extends from 0-100 for first 3 graphs (a), (b) and (c) whereas it extends from 0-512 in figure (d). The first three graphs have 100 parameters before producing the final three outputs for the SoftMax classifier whereas U-net has 512 parameters.

4.3.3 Conclusion for divNet

CNN like ANN, is a semi-supervised learning algorithm that doesn't require prior heavy feature engineering, because of its self-auto generic feature extraction property. Few researchers have been successful to develop optimization algorithm as [62], however, important contribution is the design of the better architectural unit itself [63] [64] [65]. Besides, the prevailing techniques are mainly 2D image-based methodology, therefore the 3D architecture-based concept is itself an initiative approach. This concluding section summarizes the key points that may be helpful for other researchers working with medical imaging in the same field with 3D CNN.

- The deep learning process heavily depends on the choice of training materials. Closely related images (training) can enhance the training performance; however, it can simultaneously 'spoil' the model due to overfitting. 'Good data' rather than 'big data' is required to generate a good network.
- Although our trained CNN is not deep enough to prototype a human brain structure, unlike reconstruction and segmentation models, it is however good enough to classify the MRIs, based on the segregated features learned in the convolutional layers, which is the actual aim of our study.
- MRI can be a better choice than PET for image-based CNN models. This may be due to its diverse pixel value of the MRI.
- Selection of hyper-parameters like initial-learn rate, learn-rate drop factor, the activation function, and the initialization algorithm can affect the training process, although it has little effect on its performance once the convergence is achieved.
- The architecture and depth heavily affect the performance of the model thus, it is very important to have a generalized cum optimized model. Regarding the selection of features, we are convinced that the diverging window or reception area in each layer will be more beneficial than the contemporarily used converging or equivalent reception area.
- 'Overfitting' and 'generalization' problems are the biggest challenges for deep learning models.
- Since we have proposed an optimized DL-based CNN for classification of AD, MCI, and NC using MRI/PET, it will assist the medical clinicians as an initial rapid test to identify the patient's condition using brain image scans only. Besides, MCI being an early stage of dementia means MCI identification will also help in the early prognosis of AD.

Based on our findings we hope this can be helpful in many ways to researchers working in the same field of MRI/PET classification. Our study here is limited in the ADNI dataset and may not act as universal CAD for AD detection yet more avenues are to be explored. The constantly developing deep learning methods can prove to make this process more optimal, robust, rapid, and automatic, with a minimum level of human supervision.

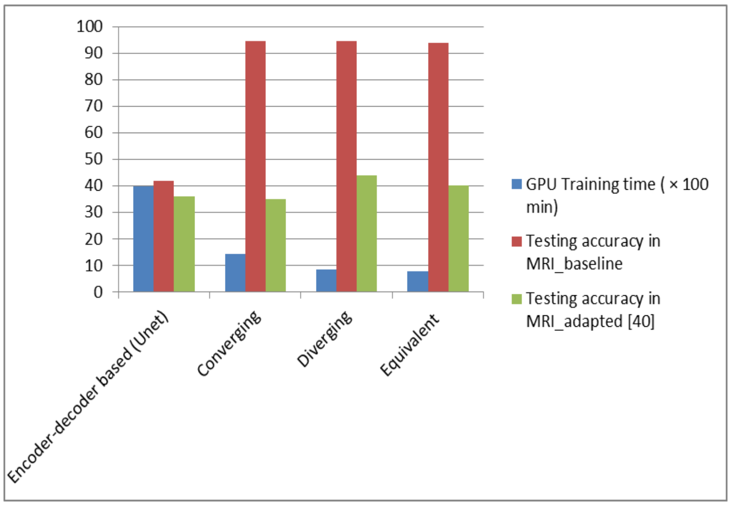


Figure 4.7: The generality test with an entirely different dataset that was not involved in training and was acquired from another ADNI project [69].

4.4 Experimental results using GAP

4.4.1 Classification performance and discussion

Numerous classification experiments were performed applying the base architecture with different normalization layers in the first 2 encoder part. Since the ultimate goal was to compare the classification result, benchmark 2D datasets were used for this purpose. The used datasets are CIFAR-10 [81], Caltech-102 [98], 5-animals, and MRI images from OASIS [99] for 2D classification purposes. Among these, 5-animals and MRI images are prepared privately and are made available in a public repository, whereas others are already publicly available. The 5-animals dataset consists of around 700 images per class of five animals viz, tiger, lion, dog, cat, and fox.

OASIS MRI consists of a total of 5220 images of four classes categorized based on the clinical dementia rating (CDR) level of participants. Details of participants are described in our previous work on 2D CNN [47] [100]. To test 3D CNN performance, 3D-MRI volumes obtained from the ADNI database (**http://adni.loni.usc.edu/**) were used. To compare the result on bulky and small training samples 2 types of datasets were prepared for 3D CNN, MRI_BASELINE the bulkier one with 988 MRI samples and MRI_SMALL smaller one with 187 MRI samples. These MRIs belong to one of the 3 classes viz AD, MCI, and NC.

Table 4.6: The base architecture is used for testing the proposed method against BN for classification. Please note that 2D and 3D architecture are different with different activation sizes. Note: here 'g1g2b3b4' architecture indicates 1st normalization GAP (g1), 2nd GAP (g2), and 3rd, 4th both BN as b3 and b4 respectively. Similarly, b1b2b3b4 means all BN and so on. The selection of hyperparameters and activation functions is based on our previous work [83]

S. N	Layers Name	Description -2D Base Architecture	2D Activation size	2D Learnable parameters	Description - 3D Base Architecture	3D Activation size	3D Learnable parameters
1	Input Image	227×227×3 images with 'zero center' normalization	227×227×3		64×64×64 images with 'zero center' normalization	64×64×64	
Encoder-1	Convolution	32 3×3 convolutions with stride [1 1] and padding 'same'	227×227×32	Weights- 3×3×32 Bias- 1×1×32	64 3×3×3×1 convolutions with stride [1 1 1] and padding 'same'	64×64×64 ×64	Weights- 3×3×3×64 Bias- 1×1×1×64
	Batch Normalization(b1) or GAP Normalization (g1)	Normalization or activated Normalization	227×227×32	Alpha- 1×1×32 Beta- 1×1×32 Gamma- 1×1×32 OR Offset- 1×1×32 Scale- 1×1×32	Normalization or activated Normalization	64×64×64 ×64	Alpha- 1×1×1×64 Beta- 1×1×1×64 Gamma- 1×1×1×64 OR Offset- 1×1×1×64 Scale- 1×1×1×64
	Leaky-ReLU	Leaky-	227×227×32		Leaky-ReLU with scale	64×64×64 ×64	

		ReLU with scale 0.01			0.01		
	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]	113×113×32		2×2×2 max pooling with stride [1 1 1] and padding [0 0 0; 0 0 0]	63×63×63 ×64	
Encoder-2	Convolution	64 5×5 convolutions with stride [1 1] and padding 'same'	113×113×64	Weights-5×5×32×64 Bias-1×1×64	64 5×5×5×64 convolutions with stride [1 1 1] and padding 'same'	63×63×63 ×64	Weights-5×5×5×64 ×64 Bias-1×1×1×64
	Batch Normalization (b2) or GAP Normalization (g2)	Normalization or activated Normalization	113×113×64	Alpha-1×1×64 Beta-1×1×64 Gamma-1×1×64 OR Offset-1×1×64 Scale-1×1×64	Normalization or activated Normalization	63×63×63 ×64	Alpha-1×1×1×64 Beta-1×1×1×64 Gamma-1×1×1×64 OR Offset-1×1×1×64 Scale-1×1×1×64
	Leaky-ReLU	Leaky-ReLU with scale 0.01	113×113×64		Leaky-ReLU with scale 0.01	63×63×63 ×64	
	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]	56×56×64		2×2×2 max pooling with stride [2 2 2] and padding [0 0 0; 0 0 0]	31×31×31 ×64	
Encoder-3	Convolution	64 7×7 convolutions with stride [1 1] and padding 'same'	56×56×64	Weights-7×7×64×64 Bias-1×1×64	64 7×7×7×64 convolutions with stride [1 1 1] and padding 'same'	31×31×31 ×64	Weights-7×7×7×64 ×64 Bias-1×1×1×64
	Batch Normalization (b3)	Batch normalization	56×56×64	Offset-1×1×64 Scale-1×1×64	Batch normalization with 64 channels	31×31×31 ×64	Offset-1×1×1×64 Scale-1×1×1×64
	Leaky-ReLU	Leaky-ReLU with scale 0.01	56×56×64		Leaky-ReLU with scale 0.01	31×31×31 ×64	

	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]	28×28×64		2×2×2 max pooling with stride [3 3 3] and padding [0 0 0; 0 0 0]	10×10×10×64	
Encoder-4	Convolution	128 9×9 convolutions with stride [1 1] and padding 'same'	28×28×128	Weights- 9×9×64×64 Bias- 1×1×64	64 9×9×9×64 convolutions with stride [1 1 1] and padding 'same'	10×10×10×64	Weights- 9×9×9×64×64 Bias- 1×1×1×64
	Batch Normalization (b4)	Batch normalization	28×28×128	Offset- 1×1×64 Scale- 1×1×64	Batch normalization with 64 channels	10×10×10×64	Offset- 1×1×1×64 Scale- 1×1×1×64
	Leaky-ReLU	Leaky-ReLU with scale 0.01	28×28×128		Leaky-ReLU with scale 0.01	10×10×10×64	
	Max Pooling	2×2 max pooling with stride [2 2] and padding [0 0 0 0]	14×14×128		2×2×2 max pooling with stride [4 4 4] and padding [0 0 0; 0 0 0]	3×3×3×64	
18	Fully Connected	1152 fully connected layer	1×1×1152	Weights- 1152×25088 Bias- 1152×1	1728 fully connected layer	1×1×1×1728	Weights- 1728×1728 Bias- 1728×1
19	Dropout	50% dropout	1×1×1152		50% dropout	1×1×1×1728	
20	Fully Connected	576 fully connected layer	1×1×576	Weights- 576×1152 Bias- 576×1	864 fully connected layer	1×1×1×864	Weights- 1728×864 Bias- 864×1
21	Dropout	50% dropout	1×1×576		50% dropout	1×1×1×864	
22	Fully Connected	N fully connected layer where ‘N’ is the number of trained class	1×1×N	Weights- N×576 Bias- N×1	3 fully connected layer	1×1×3	Weights- 3×864 Bias- 3×1
23	SoftMax	SoftMax	1×1×N		SoftMax	1×1×3	
24	Classification Output	Cross-entropy			Cross-entropy with 'AD' and 2 other classes	1×1×3	

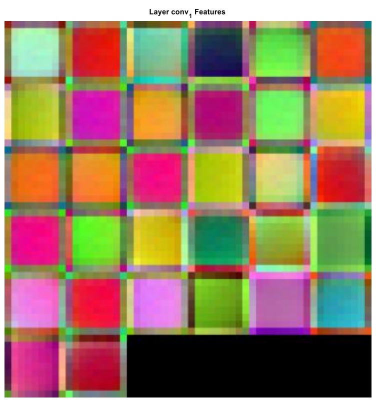
Table 4.7 shows the detailed condition for the experiment and the obtained results. Overall, it shows the use of the proposed layer for normalization gives results almost similar to that of batch normalization, and in few cases, the result is slightly better with higher training time. The result of the experiment is better in classification with double GAP layer as normalization layer in a bulkier dataset like MRI_BASELINE with overall test accuracy 93.58% against 91.89% using 100% batch normalization. Similarly, CIFAR-10 test accuracy improved from 75.11% to 75.21 % by replacing the GAP normalization layers with the BN layer in the first and second encoder. While the result is better with the replacement of the first BN layer with GAP in a smaller dataset like the 5-animals dataset (2D CNN accuracy: 62.92% vs. 58.48%), which supports the use of the proposed method for normalization and activation.

Table 4.7: Detailed experiment results using different normalization techniques in the same base architecture as shown in table 4.6. In the case of same dataset, the training, validation, and testing materials were identical, so the result could not be biased in any case. Accuracy represents the % of correctly classified samples during prediction, whereas average test recall and precision are calculated by taking the mean of class-wise recall and precision. 95% CI error represents the error with a 95% confidence score, the one with a score above 95% is only calculated for a min-error value and one with a score below 95% is only calculated for max-error value.

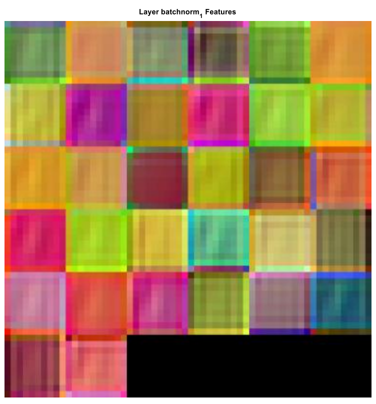
Dataset	Architecture	epoch	# of Training Image	# of Validation Image	# of Testing Image	Training Time (min)	Training Accuracy	Validation Accuracy	Testing Accuracy	Average Test recall	Average Test precision	95% CI error (min)	95% CI error (max)
Caltech-102	g1b2b3b4	100	5481	1824	1829	133	100	69.6	66.59	0.57	0.50	31.24	35.57
	g1g2b3b4					345	100	68.27	66.54	0.55	0.51	31.3	35.62
	b1b2b3b4					91	100	69.1	66.54	0.56	0.51	31.33	35.62
	Weightless AlexNet (WA)					84	100	60.36	59.69	0.43	0.42	38.16	42.65
5-animals	g1b2b3b4	100	1888	628	630	42	100	60.19	62.92	0.64	0.63	33.32	40.85
	g1g2b3b4					120	100	59.1	58.16	0.60	0.58	37.99	45.69

	b1b2b3 b4					30	100	57.1	58.48	0.59	0.59	37.68	45.37
	WA					24	90	48.09	42.47	0.41	0.42	53.67	61.38
CIFAR-10	g1b2b3 b4	100	40000	20000	10000	851	100	74.55	74.68	0.74	0.74	24.54	26.24
	g1g2b3 b4					2300	100	75.3	75.21	0.75	0.75	23.94	25.64
	b1b2b3 b4					390	100	74.68	75.11	0.76	0.75	24.02	25.72
	WA					360	100	75.14	74.76	0.75	0.75	24.39	26.09
OASIS MRI CDR (2D MRI)	g1b2b2 b4	50	2610	1044	1566	76	100	99.42	99.74	0.99	0.99	0.01	0.51
	g1g2b3 b4					180	100	99.67	99.62	0.99	0.99	0.08	0.69
	b1b2b3 b4					41	100	99.71	99.81	0.99	0.99	0.01	0.51
	WA					13	100	98.66	99.75	0.98	0.98	0.2	0.95
ADNI MRI_SMALL (3D CNN)	g1b2b3 b4	100	94	38	55	196	100	71.05	80.1	0.82	0.78	9.43	30.57
	g1g2b3 b4					551	100	73.36	81.82	0.82	0.80	7.99	27.38
	b1b2b3 b4					81	100	73.68	83.64	0.83	0.82	6.59	26.14
ADNI MRI_BASELINE (3D CNN)	g1b2b3 b4	100	495	197	296	670	100	94.92	92.23	0.94	0.90	4.72	10.82
	g1g2b3 b4					1780	100	94.42	93.58	0.93	0.92	3.63	9.21
	b1b2b3 b4					244	100	94.92	91.89	0.91	0.91	5.11	11.22

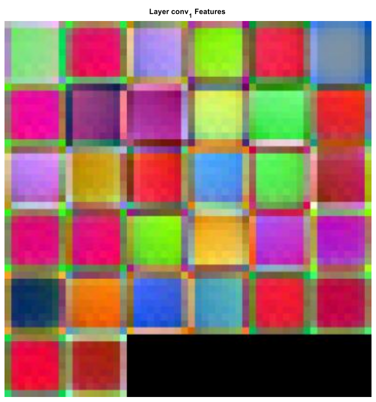
4.4.2 Feature visualization and analysis

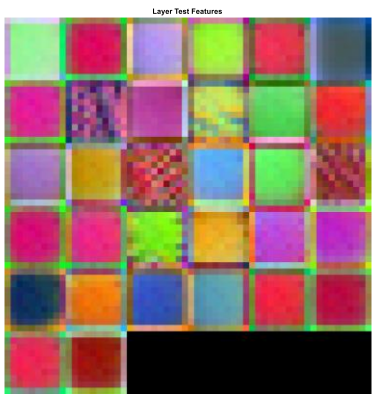


(a)



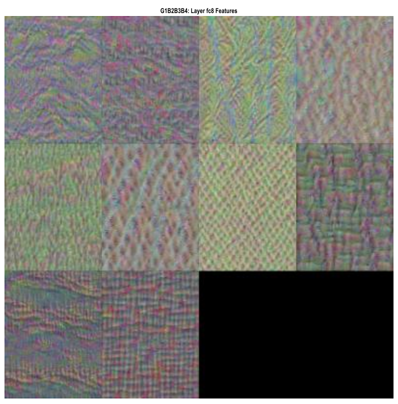
(b)

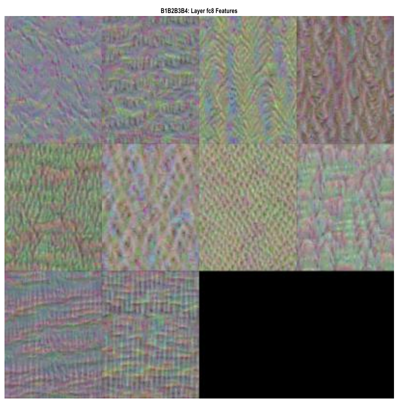




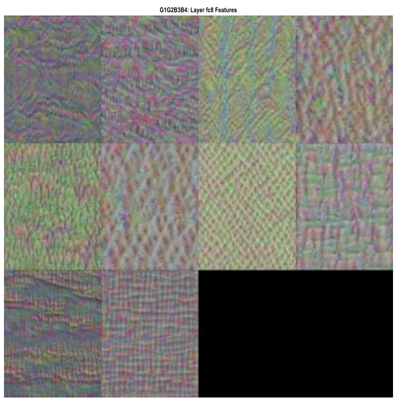
(b) (d)

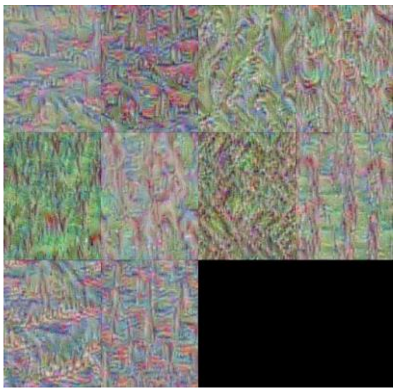
Figure 4.8: (a) First convolution layer visualization (b) First BN layer following a. visualization (c) First convolution layer visualization (d) First GAP layer following c. visualization. Check the difference in the output of batch and GAP w.r.t its respective convolution layer, the color is heavily changed in batch normalization due to insert of its batch properties, but Gaussian output remains the same, without any sharp change in filter color, instead, the color is slightly mixed up with similar color, hence a smoothing process is done here.





(a) (b)





(c) (d)

Figure 4.9: Class-wise feature learned by the trained networks of 10 classes {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck} produced using Deep dream [32] on CIFAR-10 scratch trained network under identical condition (a) using b1b2b3b4 base architecture (b) using g1b2b3b4 base architecture c) using g1g2b3b4 base architecture (d) AlexNet architecture. Please zoom in for a more detailed look. Please see the appendix for code implementation.

4.4.3 Correlation and generalization

Correlation measures the similarity between two signals or distribution. The correlation coefficient of two random variables measures the linear dependency between the input feature matrix (*X*) and output (*Z*) with *N* scalar observations as the Pearson correlation coefficient r:

$$r(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{X_i-\mu_X}{\sigma_X}\right)\left(\frac{Y_i-\mu_Y}{\sigma_Y}\right) = \frac{conv(X,Y)}{\sigma_X\sigma_Y} \ , \tag{3.11}$$

The correlation coefficient also calculates the covariance $'conv'$ between any two vector matrices. As BN is stated to reduce the internal covariant shift in its layer, we have measured the covariance between output and input using the Pearson correlation coefficient (r) using equation (3.11). Here in our case, *X* is the 3D feature matrix generated from the preceding convolution layer and *Z* is the output from the normalization layer with same dimensions as *X*. And the value of 'r' is a single value for all filters for a single image, hence correlating average characteristics of the activation layer. (Please note for the filter-wise response we have plotted the mean response plot as in figure 4.11.1 (b) to 4.11.4(b)). Perfect correlation conveys an identical result between input and output without any variance in which case the layer becomes useless, however very low correlation is also perilous, as it brings very high variability and shift between layer input and output, which makes the layer suspicious to mishaps like vanishing gradients. Hence, it is still unclear if a high correlation is good or not, which in our case, we expected a higher correlation value than the BN result, which turned out to be true.

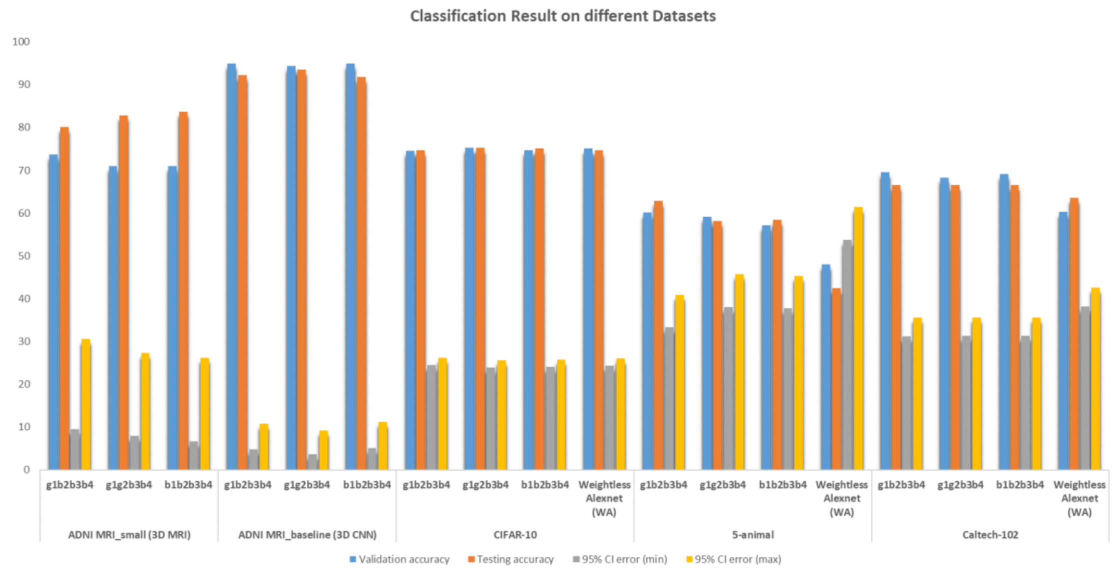
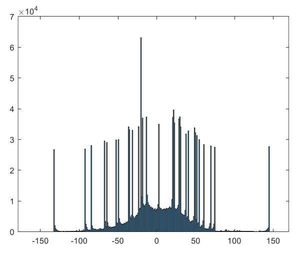
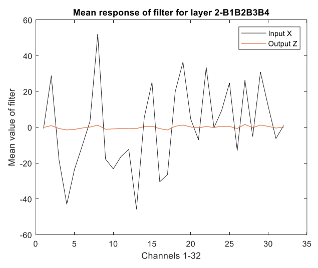


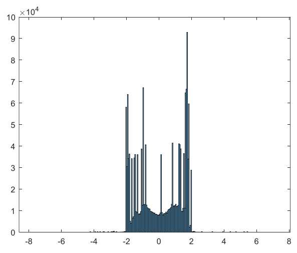
Figure 4.10: Classification result on different datasets for comparison along with validation accuracy, test accuracy, min 95% CI error, max 95% CI error as in Table 4.7. The validation accuracy and testing accuracy were calculated on the same set with identical training and testing conditions, to avoid any biases.



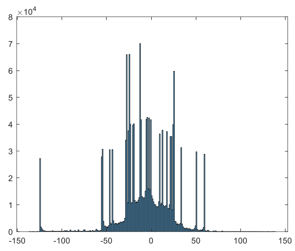
11.1 (a)



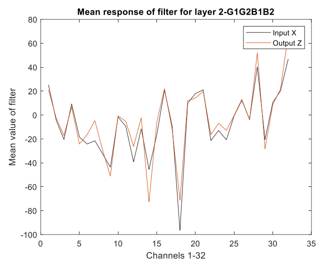
11.1 (b)



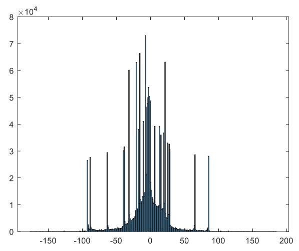
11.1(c)



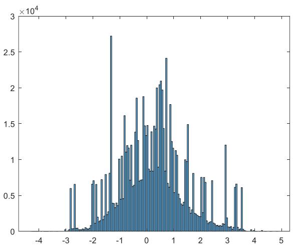
11.2 (a)



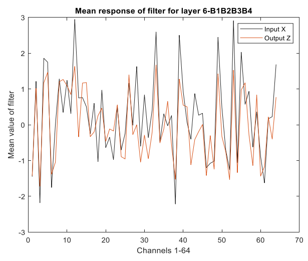
11.2 (b)



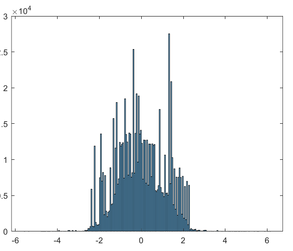
11.2 (c)



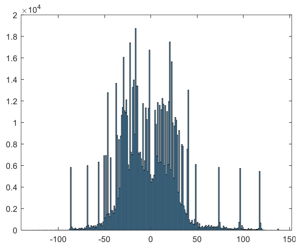
11.3 (a)



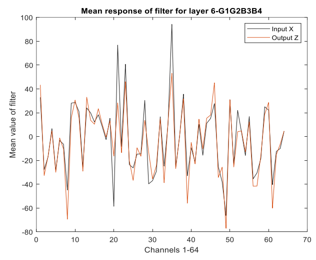
11.3 (b)



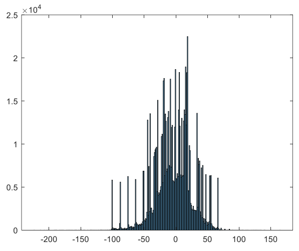
11.3 (c)



11.4 (a)



11.4 (b)



11.4 (c)

Figure 4.11: Layer-wise filter response visualization and histogram plot for a sample image from the 5-animals dataset during the testing phase. 11.1(a) Histogram of all filter input to layer b1 of b1b2b3b4. 11.1(b) Mean filter response of 32 filters in layer b1. 11.1(c) Histogram of all feature output from layer b1. 11.2(a) Histogram of all filter input to the layer g1 of g1g2b3b4. 11.2(b) Mean filter response of 32 filters in layer g1. 11.2(c) Histogram of all feature output from layer g1. 11.3(a) Histogram of all filter input to layer b2 of b1b2b3b4. 11.3(b) Mean filter response of 64 filters in layer b2. 11.3 (c) Histogram of all feature output from layer b2. 11.4(a) Histogram of all filter input to the layer g2 of g1g2b3b4. 11.4(b) Mean filter response of 64 filters in layer g2. 11.4(c) Histogram of all feature output from layer g2.

In Figure 4.11.1(c) and 4.11.3(c) the output of the batch normalization layer histogram shows the feature distribution is completely different from its input i.e., 4.11.1(a) and 11.3(a) respectively, the frequency of weight value was more in the mean range in input whereas later in output, the weight around mean is reduced. This drastic change in distribution is also shown in the mean response plot, 4.11.1(b) and 4.11.3(b) with correlation coefficients 90.46 and 84.73, respectively. However, the output pixel/weights distribution is not completely changed while using GAP normalization. The output histogram using GAP layers, 4.11.2(c) and 4.11.4(c) follows the input pattern of 4.11.2(a) and 4.11.4(a) respectively. Also, the mean plot response shows a very high correction with its input filter mean, and the correlation value is 94.6 and 91.3, respectively. Please see the appendix for code implementation.

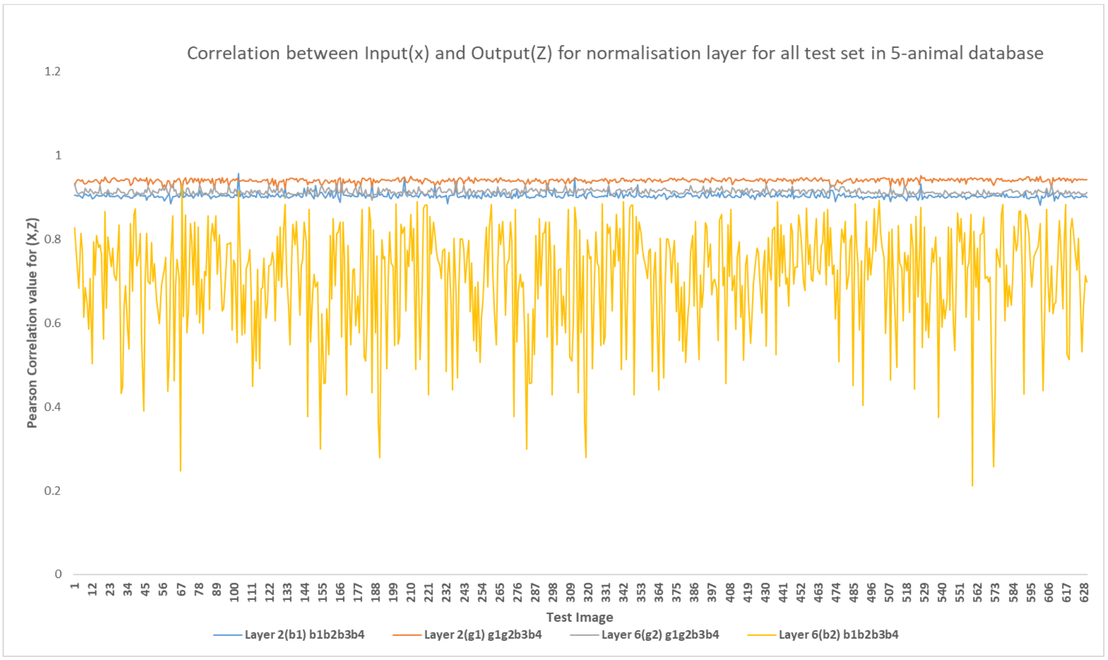


Figure 4.12: Correlation value plot between input and output in the normalization layer for all test images in the 5-animals dataset. Here in Layer 2, the BN (b1) layer produces a correlation value of around 90% for all test sets, whereas the GAP layer (g1) has a slightly higher correlation value than b1. Whereas in the second normalization layer i.e., Layer 6, the BN layer (b2) produces drastically low correlated output with its input, and in wide ranges for all test sets, i.e., ranging from 94% to as low as 22%, however, the output from g2 is not highly decorrelated with its input, hence in the range of around 90% correlation with its input. Input x is the output from the preceding convolution layer, and output Z is the output from the normalization layer. If the layer correlation

value comes out to be very low, it means the layer has decorrelated the feature. However perfect correlation is also useless.

Overfitting is one of the major adversity in ML that brings disparities in test performance and training performance within a trained network. This can be verified when the error on the training set is very less, but when an unseen similar data is predicted via the network the error is large. It means the network has 'memorized well' but has 'not learned well'. It is majorly the number of parameters that decide the fate of the network to overfitting. Keeping in mind that if the number of parameters in the network is much smaller than the total number of points in the training set, then there is little or no chance of overfitting, which simply means increasing the parameter of the training network increases the chance of overfitting. To check the generalization error, we computed the range of prediction error on 'N' test samples for a confidence score over 95% against the standard test error (STE) = 1 – accuracy. This test of error margin is also called the Wald test and represents the minimum and maximum error in the range of 95% confidence interval as shown in Table 4.7 and plotted in Figure 4.10 along with the accuracy graph. Besides, we plotted the T-SNE projection for test images and found out the errored distribution as shown in figure 4.13. In figure 4.13 the first column shows the 1728 activated features from the 1st FCL layer i.e., fc_1 is projected into 2 Principal components (PC) in x and y dimensions for all test sets. In mid column 864 activated features from 2nd FCL i.e., fc_2 layer are projected into 2 PCs in x and y dimension for all test sets. In the last column class-wise activated features from the 3rd FCL layer i.e., fc3 is projected into 2 Principal components in x and y dimension for all test sets. The distribution of colored dots is distinctly not separated in 1st FCL layers and later starts to separate in 2nd and 3rd FCL layers. The overlapped region with different colors is the region of error.

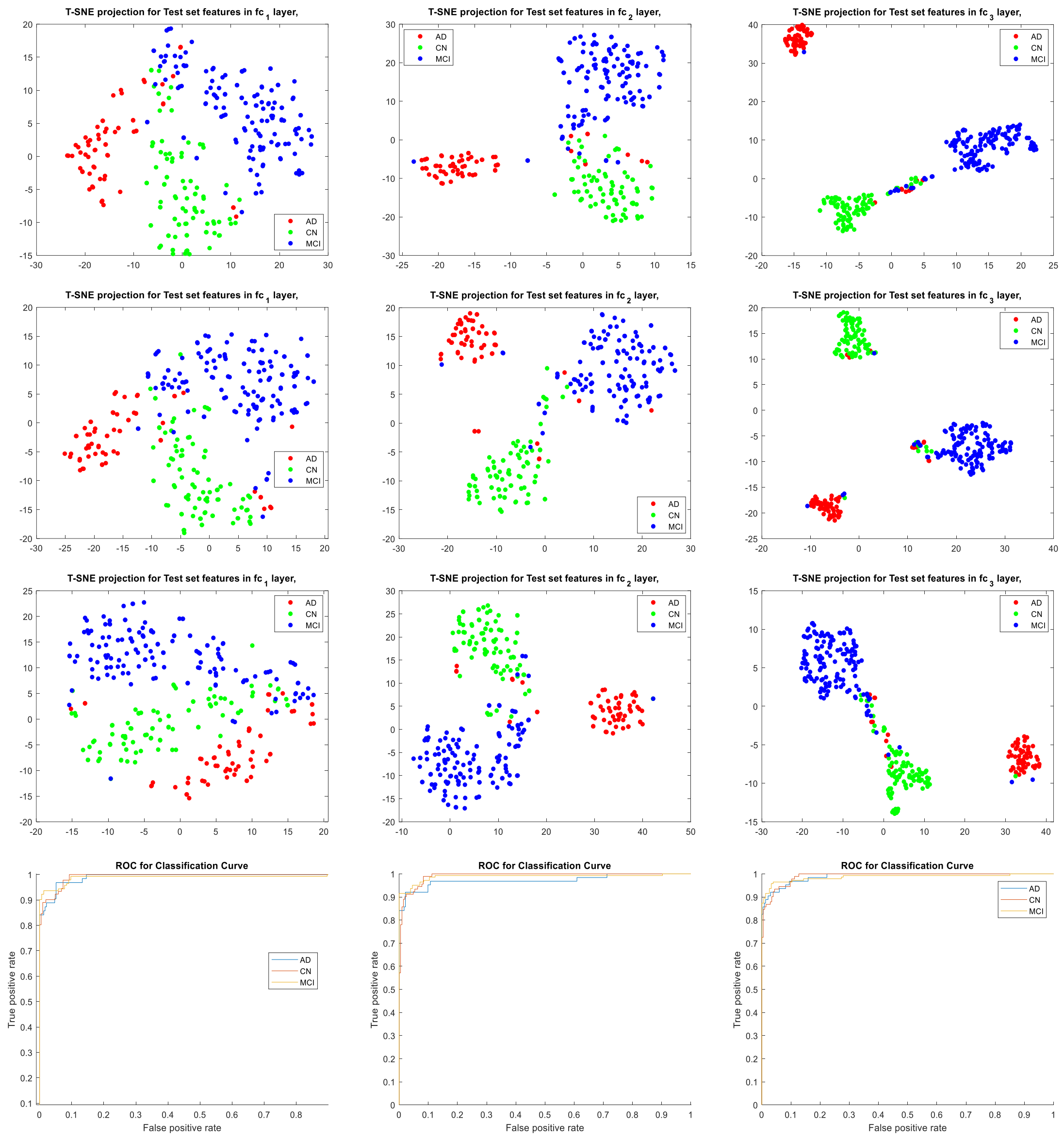
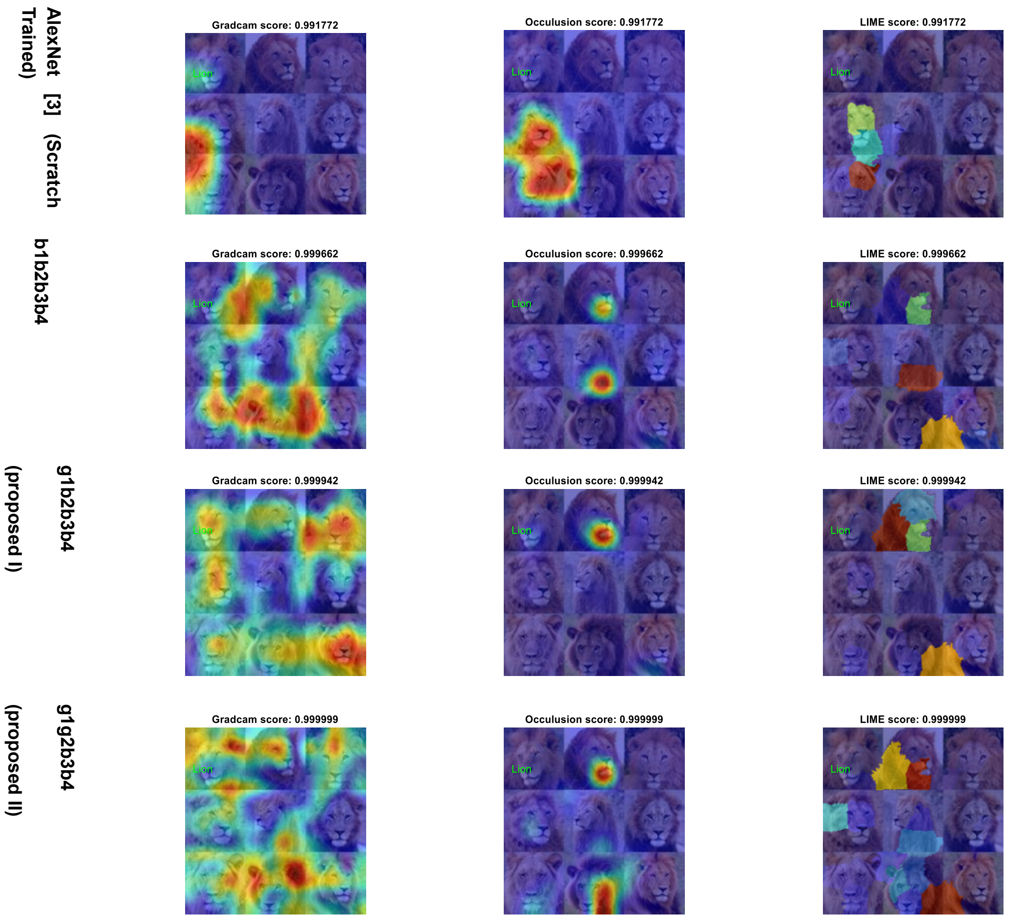
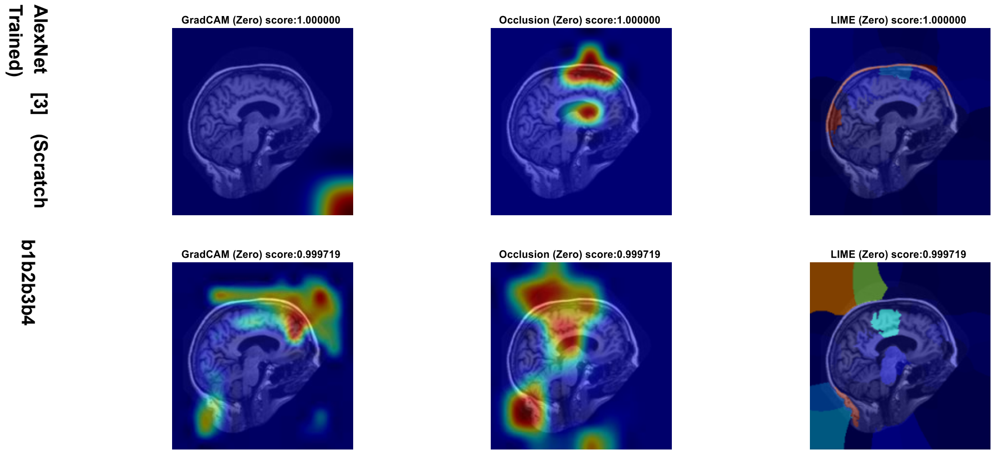


Figure 4.13: T-SNE projection of trained network for MRI_BASELINE 296 test subjects in different architecture 1st row: (a) 1st FCL features from b1b2b3b4 (b) 2nd FCL features from b1b2b3b4 trained network 4.13 (c) 3rd FCL features from b1b2b3b4 trained network. 2nd row: (a) 1st FCL features from g1b2b3b4 trained network (b) 2nd FCL features from g1b2b3b4 trained network 9.2(c) 3rd FCL features from g1b2b3b4 trained network. 3rd row: (a) 1st FCL features from g1g2b3b4 trained network (b) 2nd FCL features from g1g2b3b4 trained network (c) 3rd FCL features from g1g2b3b4 trained network. 4th row: ROC curve for all 3 class in single graph shown for MRI_BASELINE test set classification using (a) b1b2b3b4 network classification (b) g1b2b3b4 network classification (c) g1g2b3b4 network classification.

5- animals dataset



OASIS_MRI_CDR


AlexNet [3] (Scratch Trained)
GradCAM (Zero) score:1.000000
Occlusion (Zero) score:1.000000
LIME (Zero) score:1.000000
b1b2b3b4
GradCAM (Zero) score:0.999719
Occlusion (Zero) score:0.999719
LIME (Zero) score:0.999719

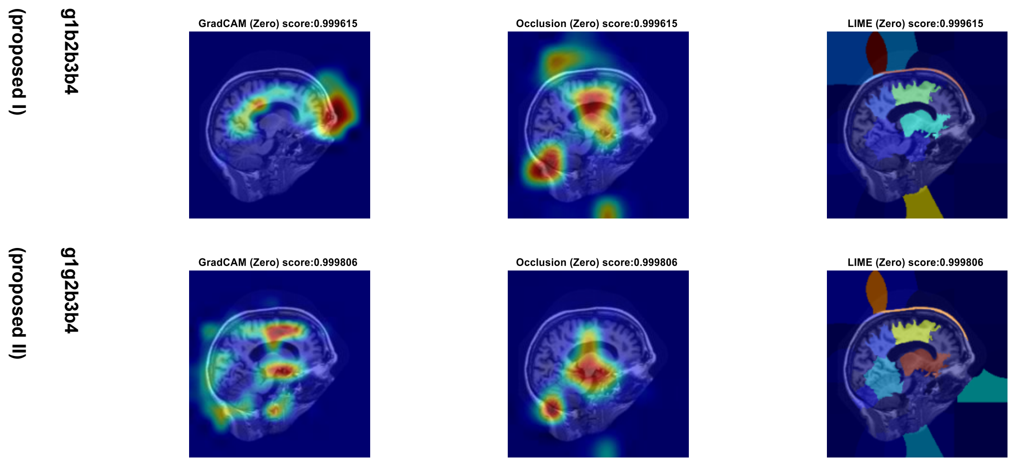

g1b2b3b4
(proposed I)
GradCAM (Zero) score:0.999615
Occlusion (Zero) score:0.999615
LIME (Zero) score:0.999615
g1g2b3b4
(proposed II)
GradCAM (Zero) score:0.999806
Occlusion (Zero) score:0.999806
LIME (Zero) score:0.999806

Figure 4.14: Comparison of feature detection heatmap using various visualization algorithms for natural images. The used techniques to generate heat maps on test images in successive order are LIME [104], Occlusion 105], and Grad-Cam[106]. Overall, AlexNet [74] has a narrow heat map area i.e., the region of influence for classification, and similarly, the heat map area of g1g2b3b4 and g1b2b3b4 is wider and more accurate than one using BN only i.e., b1b2b3b4. It signifies the better feature detection process done using GAP normalization.

4.4.4 Conclusion for GAP normalization

To conclude we have experimented with the proposed idea of the GAP layer as a normalization layer in 2D and 3D CNN. Multiple experiments show the use of GAP layers produces slightly better results in the case of the bulkier 3D dataset and lighter 2D dataset. We studied the phenomenon of overfitting via training and validation graph, normalization layer covariance via mean response plot and correlation plot, and feature representation via TSNE and histogram plots. To summarize we have listed the finding below:

- With b1b2b3b4 architecture, the training accuracy shoots higher quickly, indicating an overfitting condition as the validation accuracy during the mid-training was still too low causing a higher gap between training and validation accuracy. Hence, with the GAP layer, we delayed the faster convergence of weights during training (please see Figure 3.1).
- The weights of convolutional filters in early layers seem to change abruptly from convolution

to normalization process during BN, (please see Figure 4.8) this might suggest the feature property of the input image becomes highly uncorrelated with its input after normalization. Consequently, the correlation coefficient between input and output of BN is quite low. However, using GAP the filter weights are only slightly changed and the input-output correlation value is higher (Please see Figure 4.11 and 3.8), indicating less distortion of image property.

- As minibatch statistics are used for scaling in BN, the output is scaled with the minibatch mean, i.e., the minibatch properties of images are mixed, causing a sharper squeeze in its feature value (Please see histogram plot in Figure 4.11). This might have brought higher variability in BN output, as discussed in point 2. On the other hand, in GAP normalization the scaling mean coefficient is calculated from the activated channels from the same image i.e., equivalent to minibatch=1, so the image property from the same image is only mixed up, without spoiling its feature attributes.
- Scaling mean and variance are empirically calculated as in 3. for each input image, thus no need to pass the trained mean and variance value as in BN during the testing phase. Additionally, the BN’s error is higher for small batch size, due to imprecise batch statistics estimation.
- The activation region for decision-making is wider and accurate in most cases using g1g2b3b4 or g1b2b3b4 than the BN-based b1b2b3b4 and AlexNet (please see Figure 4.14).
- In most of the experiment, the result is slightly better (Table 4.7) and can also be visualized via T-SNE projection (please see Figure 4.13)

The proposed layer is itself not a replacement for batch normalization, since it is beneficial only if we use those layers in the first one or two convolution layers. Thus, it works as a good alternative for batch normalization in the early layers although not the ultimate layers. Moreover, another serious drawback in our idea is the use of the filter function itself, as it consumes a lot of time to perform a 3D filtering operation. Because of this the proposed method requires 2 (1 GAP) to 5 (2 GAPs) times more training time to generate a trained model. More the layers and training sample used; more will be the delay in training. To overcome this, if we can perform a weight-wise operation like convolution or BN in the layer itself without using an external filter function, it might reduce the operation time. This can be the future work. I hope this work will help the researcher working in the field of DNN to achieve better results in some specific application.

4.5 Experimental result using SGT activation

4.5.1 Classification performance and methods

The performance evaluation of the proposed function was done with the classification of three cohorts of MRIs clinically categorized as AD, CN, and MCI obtained from the ADNI website [148]. The demographic detail of the used MRIs is shown in table 4.8. Multiple scans from the same patients with different gradient wrapping and scale correction techniques were used to add heterogeneity and increase the number of experiment samples [150]. The detailed architecture used in the analysis is shown in table 4.9. The total dataset was divided into three parts viz train, validation, and test set in the ratio of 5:2:3 so that 495 MRIs were used in training, 197 MRIs for validation, and 296 MRIs were separated for testing the trained models.

Table 4.8: Participants' demographics and MRI counts.

Dataset type	AD participants	CN participants	MCI participants
Male/Female	29/36	22/38	54/33
Mean age	73.55/75.43	75.57/74.43	77.06/72.41
Total number of Participant	65	60	87
Number of MRI scans	209	305	474

Table 4.9: CNN baseline architecture used to train and classify the MRI 3D scans. Here, while analyzing the performances of different activation functions, layers containing SGT functions i.e., layer_gamma3d are replaced with other existing standard activation functions. Weights and bias values for convolution and FCL were initialized using the 'Glorot' initialization technique and for the proposed SGT layer, α and β values were randomly initialized between 0 to 1. The initial learning rate was set at 0.001 with learn drop factor of 0.95 after every 10 epochs and fully trained up to 80 epochs.

Layer no.	Layer Name	Layer Description	Output size	No of learnable Parameter
1	Image Input	64×64×64×1 images with 'zero-center' normalization	64×64×64×1	0
2	Convolution	64 3×3×3×1 convolutions with stride [1 1 1] and padding 'same'	64×64×64×64	Weights=1728 Bias= 64
3	Batch Normalization	Batch normalization with 64 channels	64×64×64×64	Offset = 64, Scale = 64
4	layer_gamma3d or ReLU/Leaky-ReLU/Tanh	Proposed SGT function with 2 learnable parameters for 64 channels	64×64×64×64	α= 64, β = 64 or 0
5	3-D Max Pooling	2×2×2 max pooling with stride [1 1 1] and padding [0 0 0; 0 0 0]	63×63×63×64	0
6	Convolution	64 5×5×5×64 convolutions with stride [1 1 1] and padding 'same'	63×63×63×64	Weights=512K Bias= 64
7	Batch Normalization	Batch normalization with 64 channels	63×63×63×64	Offset = 64, Scale = 64
8	layer_gamma3d or ReLU/Leaky-ReLU/Tanh	Proposed SGT function with 2 learnable parameters for 64 channels	63×63×63×64	α= 64, β = 64 or 0
9	3-D Max Pooling	2×2×2 max pooling with stride [2 2 2] and padding [0 0 0; 0 0 0]	31×31×31×64	0
10	Convolution	64 7×7×7×64 convolutions with stride [1 1 1] and padding 'same'	31×31×31×64	Weights=1.404M Bias= 64
11	Batch Normalization	Batch normalization with 64 channels	31×31×31×64	Offset = 64, Scale = 64
12	layer_gamma3d or ReLU/Leaky-ReLU/Tanh	Proposed SGT function with 2 learnable parameters for 64 channels	31×31×31×64	α= 64, β = 64 or 0
13	3-D Max Pooling	2×2×2 max pooling with stride [3 3 3] and padding [0 0 0; 0 0 0]	10×10×10×64	0
14	Convolution	64 9×9×9×64 convolutions with stride [1 1 1] and padding 'same'	10×10×10×64	Weights=2.985M Bias= 64
15	Batch Normalization	Batch normalization with 64 channels	10×10×10×64	Offset = 64, Scale = 64
16	layer_gamma3d or ReLU/Leaky-ReLU/Tanh	Proposed SGT function with 2 learnable parameters for 64 channels	10×10×10×64	α= 64, β = 64 or 0
17	3-D Max Pooling	2×2×2 max pooling with stride [4 4 4] and padding [0 0 0; 0 0 0]	3×3×3×64	0
18	Fully Connected	1728 fully connected layer	1×1×1×1728	Weights=2.98M Bias= 1728
19	Dropout	50% dropout	1×1×1×1728	0
20	Fully Connected	3 fully connected layer	1×1×1×3	Weights=5.18K Bias= 3

21	SoftMax	SoftMax function	1×1×1×3	0
22	Classification Output	Cross-entropy with 'AD', 'CN' and 'MCI' labels	1×1×1×3	0

Table 4.10: Results for multi-class MRI classification using CNN architecture as in Table 4.9.

Type of Activation function	Activation Function Name	Final Validation Accuracy (%)	Test Accuracy (%)	Final Validation Loss	Cohen's kappa Score	Precision (class-wise [AD CN MCI])	Predicted Confusion Matrix	True Confusion Matrix
Using Standard Activation functions	Tanh	90.862	92.57	0.5338	0.897	[0.8889 0.9120 0.9507]	56 1 6 0 83 8 1 6 135	
	ReLU	87.817	91.22	1.0425	0.8603	[0.9048 0.9011 0.9225]	57 3 3 1 82 8 1 10 131	
	Leaky-ReLU	90.355	93.92	0.8201	0.9029	[0.9206 0.9121 0.9648]	58 2 3 1 83 7 1 4 137	
Using all or partially SGT function	gamma2	90.862	92.91	0.8777	0.887	[0.9206 0.8901 0.9577]	58 0 5 5 81 5 1 5 136	63 0 0 0 91 0 0 0 142
	gamma2_alt	91.370	92.91	0.7587	0.886	[0.8730 0.9231 0.9577]	55 0 8 0 84 7 1 5 136	
	gamma4_adam	92.893	92.57	0.5683	0.8818	[0.8730 0.9451 0.9366]	55 6 2 0 86 5 1 8 133	
	gamma4_sgdm	92.893	93.24	0.3086	0.892	[0.9048 0.9121 0.9577]	57 2 4 1 83 7 1 5 136	

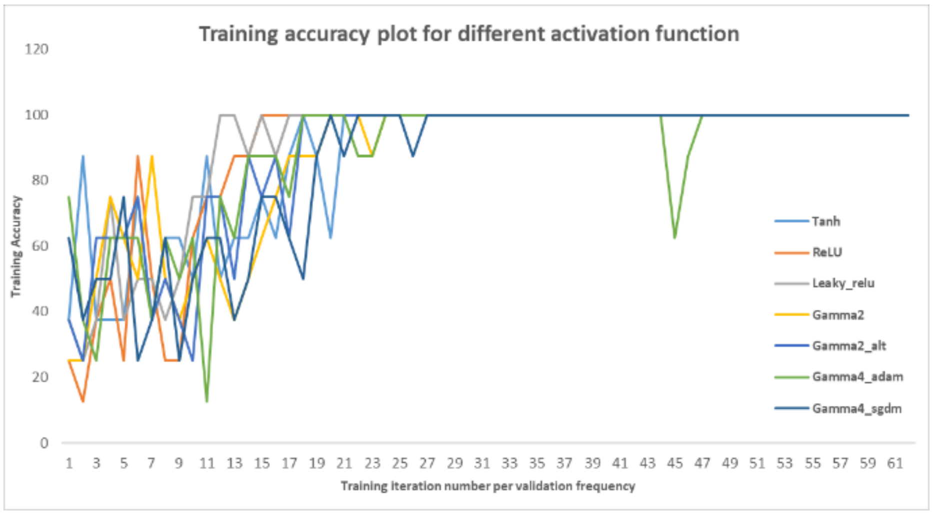


Figure 4.15: (a) Training accuracy plot for MRI classification using baseline CNN models with different activation functions.

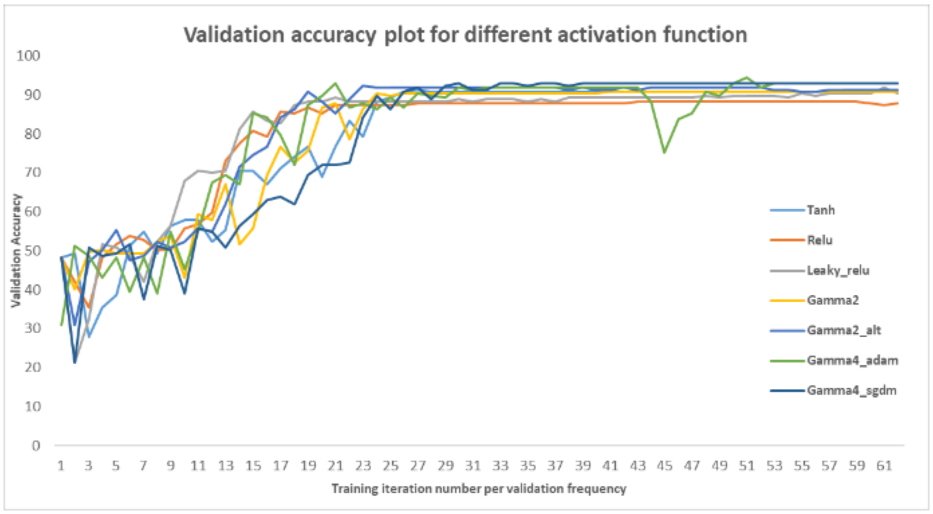


Figure 4.15: (b) Validation accuracy plot for MRI classification using baseline CNN models with different activation functions.

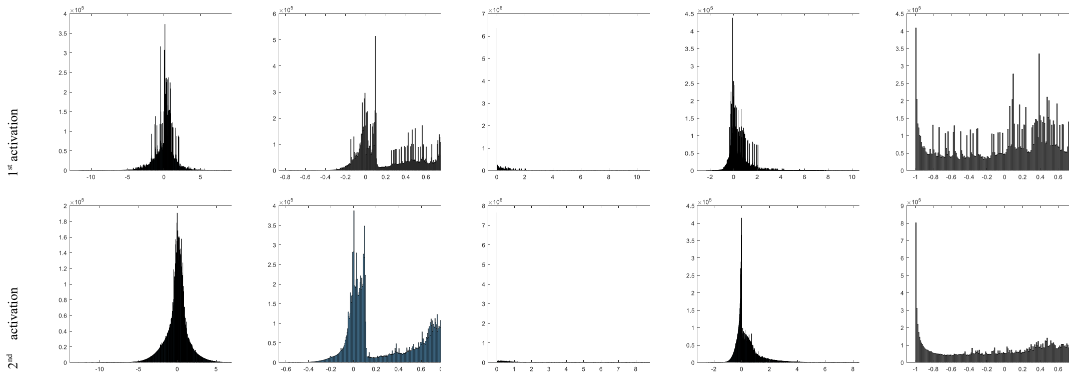
From Table 4.10 it is observed that all the version of the network using SGT activation (i.e., gamma2, gamma2_alt, gamma4_adam, gamma4_sgdm) has higher validation accuracy, precision, and Cohen's kappa score than the other activation schemes. These classification performance parameters measure the reliability and correctness of the work, e.g., accuracy measures the number of correct prediction against the true predictions whereas precision measures how close the measured valued are to the true values. Similarly, Cohen's kappa score is like accuracy except that it is more robust and measures how much better the model is performing over the performance of a model that randomly predicts according to the frequency of each class, best suited for multiclass imbalanced dataset (please see Appendix III for all formulas). All calculated in terms of percentage, and higher the better. Here gamma2 means the first two activations are SGT and other ReLU, gamma2_alt means first and third is SGT and other ReLU, gamma4_adam uses all four activation layers as SGT with Adam optimizer while gamma4_sgdm also uses four SGT activation layers but the optimizer is Stochastic Gradient Descent with Momentum (SGDM). The validation set is the test set used during training to calculate the accuracy of prediction at different epochs, hence it helps to know how well the network is learning. Figure 4.15: (b) shows the validation accuracy calculated at different epochs along with its training accuracy in 4.15(a). It can be clearly noticed that the SGT activated network (gamma4_sgdm, gamma4_adam) reaches higher validation accuracy than other activation schemes in the final stages of training. The final validation accuracy reported in table 4.10 is the accuracy on the validation set at the 80th epoch or the final epoch. Similarly, the test set is the set that is completely unseen for the trained model and the higher performance in the test set means the network is well generalized and has good performance for unseen data. To get an unbiased result, the experimental environment along with all the hyperparameters and participating MRIs were always kept identical for all networks irrespective of the choice of activation functions. During test set classification, Leaky-ReLU performed the best with around 0.5% higher test accuracy than that of gamm4_sgdm. Still, the test accuracy of all SGT activated networks was higher than the ReLU and tanh by 2% and 1% respectively, which indicates that the proposed SGT activation scheme outperforms the traditional ReLU activation by a clear margin.

4.6 Discussion and analysis for SGT activation

4.6.1 Histogram analysis and asymmetric distribution

Weights of each layer's input (X_l) or output (Z_l) as in equations (5.2) or (5.5) is plotted against its frequency in the histograms. The normalized output values from BN are zero-mean with almost normal distribution, therefore it is not a good idea to throw away all the negative valued parameters/weights using activation functions like ReLU or sigmoid [129]. Though the flow of gradient is positive in ReLU, if a bunch of the weights is negative it causes dead ReLU with 'zero' derivative for negative weights, hence not every time ReLU is a wise choice. In cases like MRI, mostly with black background (low pixel value), it is better to use alternative activation function like Leaky-ReLU, GELU, SELU that provides non-zero gradients for negative weights ensuring the flow of gradient loss.

Figure 4.16 shows the input and output histogram plots through the SGT layer in comparison to ReLU versions. Here, please note that the input to the activation layer is the output from the batch normalization and the output of the activation layer is the input to the pooling layer. In figure 4.16, the input histogram of all activation layers has an almost symmetrical distribution which means most of the image pixel lies in the grey region after BN. Our goal of gamma correction is to reduce this grey zone and make the distinction between white (bright) and black (dark) regions. If we look at figure 4.16(b), the mid-grey region is very few in the case of output from the proposed SGT layer, whereas the output with ReLU has very high zeros and leaky-ReLU output still seems centered at zero, hence the clear skewness is seen in positive part. While the SGT layers' output data are decentralized in opposite edge regions unlike BN, and it seems like the combination of the output of tanh and Leaky-ReLU histogram. Additionally, this asymmetric feature distribution in the SGT layer supports the classification task due to the higher variance between the edge regions.



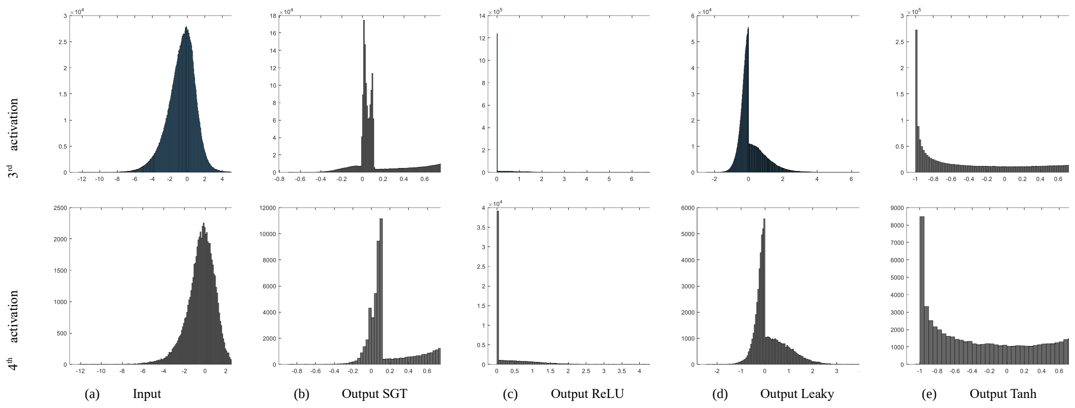


Figure 4.16: Histogram of the input features against output using various activation functions for a single MRI input plotted for different layers i.e., 4, 8, 12, and 16 (please see Table 4.9 for layers). Here, the histograms are combinedly produced using all the data values from 64 filters/channels. Generally, the combined histogram of all channels is similar to the single histogram of each channel (please see Figure 4.16_app for comparing the histogram plot of 19th filter out of 64 filters for same input MRI in Appendix section).

4.6.2 Channel wise activation

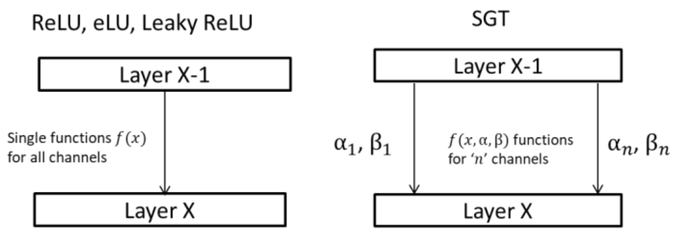


Figure 4.17: Conventional activation functions work in a constant way to all inputs whereas the proposed SGT function works differently for the different channels because of altering values of parameters α_n, β_n within the layers channel in respect to equation (5.2).

We were highly interested to see what value the SGT (layer_gamma3d in Table 4.9) parameters would learn at different activation layers. The stem plot for α and β values from all activation layers as in Figure 4.18 shows that for the first SGT activation layer (i.e., Layer 4) the values for α and β were mostly positive and only a few remained negative, also there were more β with value >1 than α. The range for the value of α and β lied between -0.4 to 1.4. Interestingly in the intermediate activation layers (i.e., Layer 8,12) and the final activation layer (i.e., Layer 16) none of the values for β remained negative while the values for α in most channels remained negative. This might imply that for feature value $x>0$, required positive gamma correction, and for negative feature value $x<0$, required negative gamma correction in the intermediate layer. In a more general statement, the gamma activation made brighter pixels look brighter and darker pixels look darker, which resulted in a more distinct intensity profile.

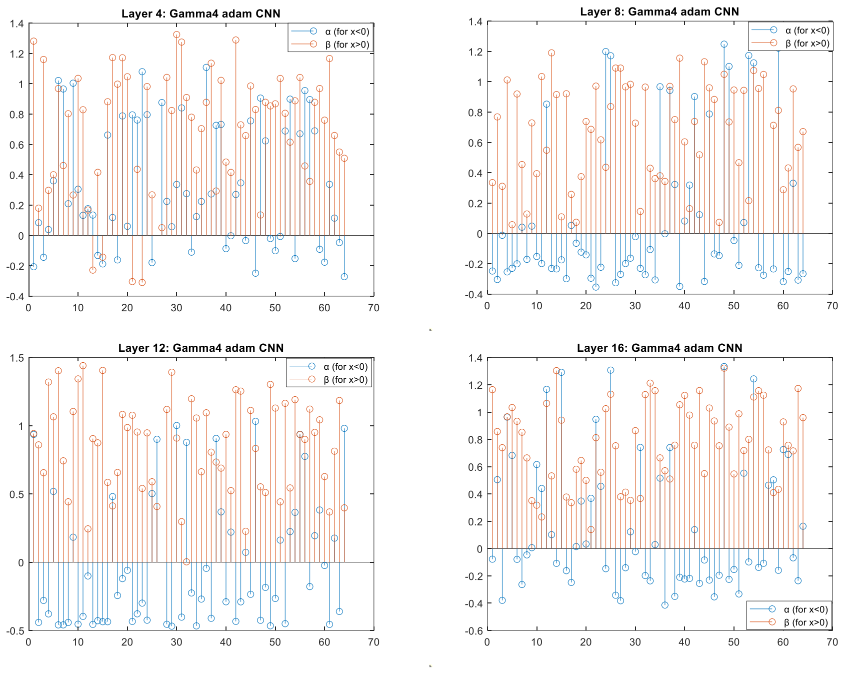


Figure 4.18: Pictorial representation of α and β values for a trained model at different layers for gamma4_adam network using Adam optimization. Here α and β are channel-wise learnable parameters in SGT layers, each corresponding to 64 channels

4.6.3 Analyzing weights and bias in the final FCL

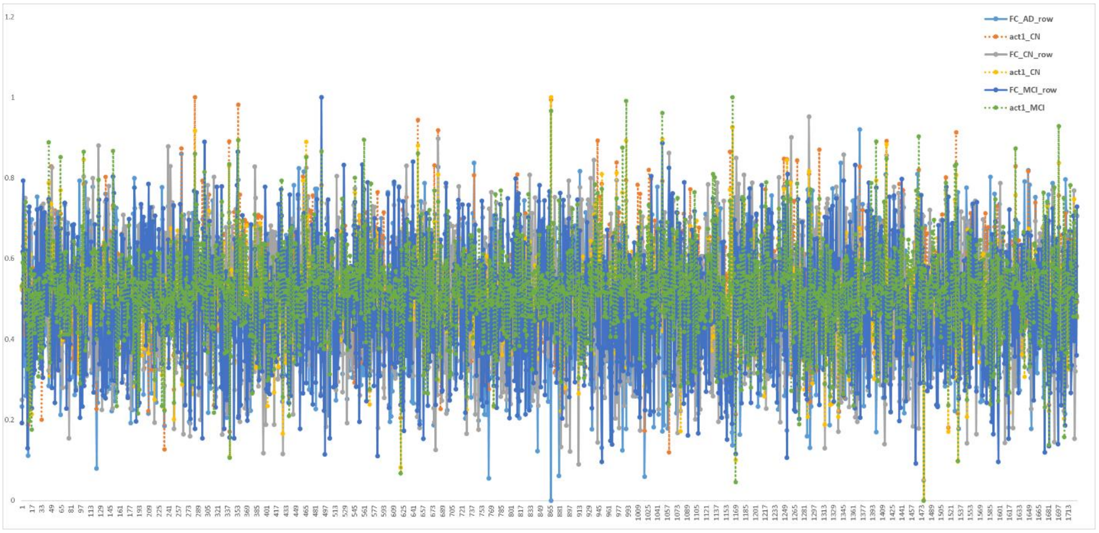


Figure 4.19: Final FCL 1728 weights plots of trained gamma4_adam network corresponding to each class label. Here FC_AD_row represents the final weights of the layer from the fully trained gamma4_adam network belonging to the AD class, similarly, FC_CN_row and FC_MCI_row represent for CN and MCI categories respectively. While the plots of act1_AD are the weights calculated for a typical AD categorized MRI, obtained using the trained model during the testing phase. So, are the weights calculated as act1_CN and act_MCI for a CN and MCI categorized MRI during testing respectively. This plot is to show how closely the test sample (act1_xx) follows its parent class characteristics (FC_xx_row). Furthermore, to evaluate this characteristic a correlation table is calculated as in table 4.11, where it is very clear that the test sample weights (act1_xx) have the highest correlation with its parent class (FC_xx_row) where xx represents the same class for both sample and parent. The same class high correlation between FC_xx_row and act_xx shows that the network is learning class-wise property precisely.

Table 4.11: Correlation matrix for weights as shown in Figure 4.19. The colored ones are the highest measured value for the sample-parent pair, higher being better.

	FC_AD_row	act1_AD	FC_CN_row	act1_CN	FC_MCI_row	act1_MCI
FC_AD_row	1	0.215182	0.39671942	-0.14342	0.341976354	-0.19525
act1_AD	0.215182223	1	0.04948163	0.786964	-0.187160957	0.630355
FC_CN_row	0.396719425	0.049482	1	0.242651	0.309621325	-0.00694
act1_CN	-0.143417277	0.786964	0.24265146	1	-0.009627914	0.85748
FC_MCI_row	0.341976354	-0.18716	0.30962132	-0.00963	1	0.275327
act1_MCI	-0.195245895	0.630355	-0.00693744	0.85748	0.275326654	1

FCL represents an MLP Feedforward network with learnable weights and bias but mostly without activation function when used in CNN[10][11]. In FCL all inputs are mapped to output unlike the convolutional layers which are used as a patch-based feature extractor, therefore weights and bias in FCL are highly responsible for predicting the result, and the weights themselves suggest which input has more effect (or gain) on output. Thus, the weight distribution pattern of FCL might indicate how a network behaves during the test phase. To interpret this, we plotted all trained weights of the final FCL (Input nodes=1728, output nodes= 3, connection= 5184) for all 3 classes as shown in figure 4.19. Later the correlation matrix is calculated as in Table 4.11, which shows a sample MRI's features (or weights) calculated from the FC layer is closely correlated with its parent class. For instance, the test sample CN MRI's FC weights i.e., act1_CN has correlation value [0.143417277 0.24265146 -0.009627914] with the trained network corresponding layer weights [FC_AD_row FC_CN_row FC_MCI_row]. As a result, the highest correlation value is 0.24265146 for FC_CN_row implies, the MRI test sample has a higher affinity for 'CN' class weights during classification besides, it supports the logic behind why the network predicts the test sample label as 'CN'.

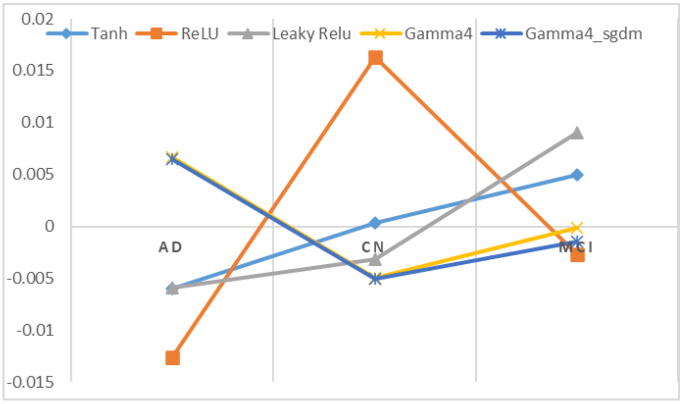
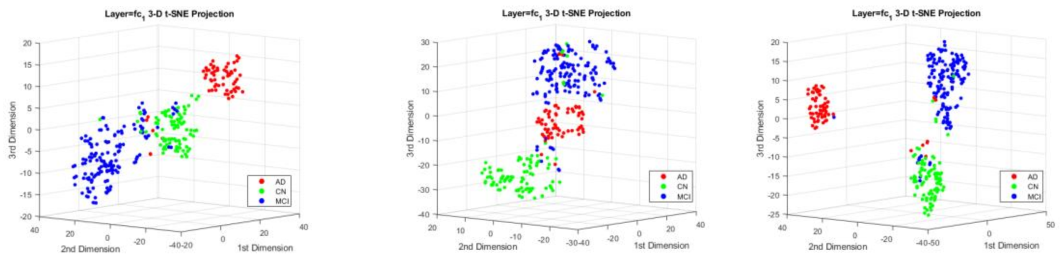


Figure 4.20: Bias value plot of final FCL layer from the baseline CNN model using different activation functions.

After weights analysis, we were also interested to analyze the bias value. Hence, the idea is to check how much network is biased to each class via calculated bias in the final FCL layer. The obtained bias value is from the last FCL, which goes into SoftMax for probability calculation. We know weights in the network directly influence the output value for input, whereas bias works as a regularization constant to make non-zero output when input/weights are zero and do not have a successive layer-wise influence on the output. Although it is difficult to exactly interpret the bias value theoretically, we assume the bias values close to each other cohort, can correlate how each other is numerically related. E.g., for Tanh trained CNN the obtained bias value is [AD CN MCI] = [-0.006021075 0.000316184 0.004943716], which means that AD (with negative value) is closely related to CN (small positive), being the difference of value between AD and CN greater than AD and MCI, which is against the general assumption that AD is closely related to MCI, both being a dementia condition. This might also indicate that the tanh network can easily differentiate between AD and MCI rather than AD and CN, which is not what it should be, the same is the case with Leaky-ReLU. Surprisingly this might be supportive for the classification task, as a higher difference in bias would make the network easier to calculate the class-wise probabilities scores. On the contrary, the proposed SGT networks (gamma4_adam and gamma4_sgdm) have a larger difference between AD and CN bias values, one being positive and the other being negative. While MCI is nearly 0 indicating a moderate status between AD and CN. The lower difference in MCI and CN bias values in the gamma4_adam network might suggest a higher difficulty in classification and generalization between CN and MCI, which supports the real scenario.

Figure 4.21 represents the 3D t-SNE projection for visualization of reduced features from the final FCL. The features into the FCL are originally from multiple channels later reduced into a single channel, so are considered flattened features. However, each MRI’s flatten feature needs to be reduced to a 2D or 3D dimension for proper visualization. The distinctive clustered distribution in the projection means the network is learning class discriminant properties with good fitness.



(a) ReLU activated CNN (b) Leaky-ReLU activated CNN (c) SGT activated CNN

Figure 4.21: 3D projection viewed at the same angle for the test set features reduced from 1728 dimension to 3 using the t-SNE algorithm. Here each color dot represents an MRI scan, hence a total of 296 dots for 296 test MRI. The non-linear feature distribution shows the requirement of complex boundaries for classifications. Here the figure from left to right is obtained as the result of t-SNE distribution using ReLU, Leaky-ReLU, and SGT activation separately in the same baseline 3D CNN model. Please see figure 21_app in the Appendix section for the 3D t-SNE projection of all individual layers in the gamm4_adam network.

4.6.4 Conclusion

DNN design and hyperparameter selection are task-specific with no single model or function that can work universally for all, however, after all the experiments and analysis we can conclude:

- A novel channel-wise dynamic activation function is introduced with superior performance than standard ReLU and tanh function in 3D CNN for MRI classification.
- We showed that the proposed activation function can diminish the negative gradient loss arising with the negative weights with less likelihood for vanishing or exploding gradient problem and also zero gradient problem unlike dead ReLU (please see derivative plots in Figure 4.2(c) and 4.2(d)) for shallower networks.
- The analysis performed in histograms (Figure 4.16), showed negative weights are produced in a quite large measure during convolution and batch normalization operation hence, the idea of utilizing negative weights to relatively contribute to the gradient loss proved meaningful with the proposed activation function.
- We tried to explore the pattern of weights and bias in the final FCL and how numerically they

might be related (Figure 4.19, 4.20, and Table 4.11) in regard to the classification task. This might be one of the few attempts in this field as weights can be optimized in numerous approaches and are difficult to analyze mathematically.

Our idea is quite simple as well as interesting so we hope, our work could be helpful and meaningful for other researchers working in deep learning. In the future, more modifications are required for superior performance than all other activation functions and to work universally in all kinds of the image dataset.

CHAPTER 5

Final Conclusion

5.1 Final conclusion and future works

This chapter is for the conclusion of the thesis work. We will make a very brief concluding remark here. Our work in the field of deep learning is an attempt to make the MRI classification task easier and more customized. Though medical imaging modalities alone cannot give a decisive answer for disease diagnosis, it can certainly assist clinicians and radiologists to make the final decision. And maybe in the very near future, AI-based methods can be practically used with higher accuracy than human raters, thanks to the hard work of deep learning researchers. Interestingly it is important to note that, the architectures and algorithms we design or use from the available libraries keep on developing, so neither a single architecture can last forever nor work universally. However, the concept that comes along with those architectures and algorithms are passed to the newer generation so that, it will certainly build the base for better CAD designs. We sincerely hope our work can also be a small but meaningful contribution to the field of medical image analysis using deep learning.

For future works, we consider the integration of the proposed layer into a single one with lesser parameters for a more simplified version. In doing this we need to properly design a single layer architecture for both the normalization and activation process, which in turn normalize the value and brings nonlinearity in the system with ‘activated’ features. Additionally, we can work on customizing layers in more complex DNN architectures like Transformers, GAN, LSTM, etc. in a similar way. For every architecture in DNN, weight update, optimization, and convergence are the most essential processes, therefore future algorithms should be designed to ease those processes. At the same time, our research should be directed toward making the algorithms simple rather than complex, because engineering is about making things simple not complex for any purposes. We hope we can do this in the future with more advancements in semiconductor devices.

5.2 Appendix

Appendix I: With list of files for MRI and PET types in Table I along with demographic details in Table II. Additionally, a high-quality image for Figure 2.2 is presented.

Appendix II: MATLAB implementation code is presented as C1, C2, C3. Confusion matrix of performance (as of Table 4.7) is also presented. The prepared dataset can be downloaded from https://drive.google.com/drive/folders/1G1fsK2VxaHkvtJJfvpiB3rMpiqCcdkB2?usp=sharing

Appendix III: Related figures with equations and implementation code.

References

[1] Tanabe, J.L., Amend, D., Schuff, N., DiSclafani, V., Ezekiel, F., Norman, D., Fein, G. and Weiner, M.W., 1997. Tissue segmentation of the brain in Alzheimer disease. American Journal of Neuroradiology, 18(1), pp.115-123.

[2] Alzheimer's Association, 2017. 2017 Alzheimer's disease facts and figures. Alzheimer's & Dementia, 13(4), pp.325-373.

[3] Huang, Y., Xu, J., Zhou, Y., Tong, T. and Zhuang, X., 2019. Diagnosis of Alzheimer's disease via multi-modality 3D convolutional neural network. Front Neurosci 13: 509, pp.1-12.

[4] Burton, E.J., McKeith, I.G., Burn, D.J., Williams, E.D. and O'Brien, J.T., 2004. Cerebral atrophy in Parkinson's disease with and without dementia: a comparison with Alzheimer's disease, dementia with Lewy bodies and controls. Brain, 127(4), pp.791-800.

[5] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), pp.541-551.

[6] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., Handwritten digit recognition with a back-propagation network, 1989. In Neural Information Processing Systems (NIPS) (pp. 396-404).

[7] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[9] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[10] Ren, S., He, K., Girshick, R. and Sun, J., 2017. Faster R-CNN. Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[11] Girshick, R., 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[12] Badrinarayanan, V., Kendall, A. and Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), pp.2481-2495.

[13] Kendall, A., Badrinarayanan, V. and Cipolla, R., 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:1511.02680.

[14] Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images.

[15] Mordvintsev, A., Olah, C. and Tyka, M., 2015. Inceptionism: Going deeper into neural networks.

[16] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2015. Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence, 38(1), pp.142-158.

[17] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[18] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A

large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). IEEE.

[19] Chen, J., Wang, Y., Wu, Y. and Cai, C., 2017, August. An ensemble of convolutional neural networks for image classification based on LSTM. In 2017 International Conference on Green Informatics (ICGI) (pp. 217-222). IEEE.

[20] Griffin, G., Holub, A. and Perona, P., 2007. Caltech-256 object category dataset.

[21] Khagi, B. and Kwon, G.R., 2018. Pixel-label-based segmentation of cross-sectional brain MRI using simplified SegNet architecture-based CNN. Journal of healthcare engineering, 2018.

[22] Rebsamen, M., Knecht, U., Reyes, M., Wiest, R., Meier, R. and McKinley, R., 2019. Divide and conquer: stratifying training data by tumor grade improves deep learning-based brain tumor segmentation. Frontiers in neuroscience, 13, p.1182.

[23] Tajbakhsh, N., Shin, J.Y., Gurudu, S.R., Hurst, R.T., Kendall, C.B., Gotway, M.B. and Liang, J., 2016. Convolutional neural networks for medical image analysis: Full training or fine tuning?. IEEE transactions on medical imaging, 35(5), pp.1299-1312.

[24] Castillo, D., Lakshminarayanan, V. and Rodríguez-Álvarez, M.J., 2021. MR images, brain lesions, and deep learning. Applied Sciences, 11(4), p.1675.

[25] Nair, T., Precup, D., Arnold, D.L. and Arbel, T., 2020. Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. Medical image analysis, 59, p.101557.

[26] Noor, M.B.T., Zenia, N.Z., Kaiser, M.S., Mahmud, M. and Mamun, S.A., 2019, December. Detecting neurodegenerative disease from MRI: a brief review on a deep learning perspective. In International conference on brain informatics (pp. 115-125). Springer, Cham.

[27] Ebrahimighahnavieh, M.A., Luo, S. and Chiong, R., 2020. Deep learning to detect Alzheimer's disease from neuroimaging: A systematic literature review. Computer methods and programs in biomedicine, 187, p.105242.

[28] Yamanakkanavar, N., Choi, J.Y. and Lee, B., 2020. MRI segmentation and classification of human brain using deep learning for diagnosis of Alzheimer's disease: a survey. Sensors, 20(11), p.3243.

[29] Rathore, S., Habes, M., Iftikhar, M.A., Shacklett, A. and Davatzikos, C., 2017. A review on neuroimaging-based classification studies and associated feature extraction methods for Alzheimer's disease and its prodromal stages. NeuroImage, 155, pp.530-548.

[30] Alzheimer's Association, 2017. 2017 Alzheimer's disease facts and figures. Alzheimer's & Dementia, 13(4), pp.325-373.

[31] Apostolova, L.G., 2016. Alzheimer disease. Continuum: Lifelong Learning in Neurology, 22(2 Dementia), p.419.

[32] Huang, Y., Xu, J., Zhou, Y., Tong, T., Zhuang, X. and Alzheimer's Disease Neuroimaging Initiative (ADNI), 2019. Diagnosis of Alzheimer's disease via multi-modality 3D convolutional neural network. Frontiers in neuroscience, 13, p.509.

[33] Burton, E.J., McKeith, I.G., Burn, D.J., Williams, E.D. and O'Brien, J.T., 2004. Cerebral atrophy in Parkinson's disease with and without dementia: a comparison with Alzheimer's disease, dementia with Lewy bodies and controls. Brain, 127(4), pp.791-800.

[34] Hubel, D.H. and Wiesel, T.N., 1959. Receptive fields of single neurons in the cat's striate cortex. The Journal of physiology, 148(3), p.574.

[35] Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference

Proceedings.

[36] https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34 accessed 11th February 2020.

[37] Cheng, D., Liu, M., Fu, J. and Wang, Y., 2017, July. Classification of MR brain images by combination of multi-CNNs for AD diagnosis. In Ninth international conference on digital image processing (ICDIP 2017) (Vol. 10420, pp. 875-879). SPIE.

[38] Liu, M., Cheng, D., Wang, K. and Wang, Y., 2018. Multi-modality cascaded convolutional neural networks for Alzheimer's disease diagnosis. Neuroinformatics, 16(3), pp.295-308.

[39] Lundervold, A.S. and Lundervold, A., 2019. An overview of deep learning in medical imaging focusing on MRI. Zeitschrift für Medizinische Physik, 29(2), pp.102-127.

[40] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[41] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. and Darrell, T., 2014, January. Decaf: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning (pp. 647-655). PMLR.

[42] Sharif Razavian, A., Azizpour, H., Sullivan, J. and Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).

[43] Hosseini-Asl, E., Keynton, R. and El-Baz, A., 2016, September. Alzheimer's disease diagnostics by adaptation of 3D convolutional network. In 2016 IEEE international conference on image processing (ICIP) (pp. 126-130). IEEE.

[44] Wang, S.H., Zhang, Y., Li, Y.J., Jia, W.J., Liu, F.Y., Yang, M.M. and Zhang, Y.D., 2018. Single slice based detection for Alzheimer's disease via wavelet entropy and multilayer

perceptron trained by biogeography-based optimization. Multimedia Tools and Applications, 77(9), pp.10393-10417.

[45] Yang, G., Zhang, Y., Yang, J., Ji, G., Dong, Z., Wang, S., Feng, C. and Wang, Q., 2016. Automated classification of brain images using wavelet-energy and biogeography-based optimization. Multimedia Tools and Applications, 75(23), pp.15601-15617.

[46] Jha, D., Kim, J.I. and Kwon, G.R., 2017. Diagnosis of Alzheimer's disease using dual-tree complex wavelet transform, PCA, and feed-forward neural network. Journal of healthcare engineering, 2017.

[47] Khagi, B., Kwon, G.R. and Lama, R., 2019. Comparative analysis of Alzheimer's disease classification by CDR level using CNN, feature selection, and machine-learning techniques. International Journal of Imaging Systems and Technology, 29(3), pp.297-310.

[48] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

[49] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017, February. Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-first AAAI conference on artificial intelligence.

[50] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[51] Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., Van Der Laak, J.A., Van Ginneken, B. and Sánchez, C.I., 2017. A survey on deep learning in medical image analysis. Medical image analysis, 42, pp.60-88.

[52] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing

systems, 28.

[53] He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[54] https://medium.com/finc-engineering/cnn-do-we-need-to-go-deeper-afe1041e263e accessed 11th February 2020.

[55] Targ, S., Almeida, D. and Lyman, K., 2016. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029.

[56] Karpathy, A., 2016. Connecting images and natural language (Doctoral dissertation, Stanford University).

[57] Pereira, S., Pinto, A., Alves, V. and Silva, C.A., 2016. Brain tumor segmentation using convolutional neural networks in MRI images. IEEE transactions on medical imaging, 35(5), pp.1240-1251.

[58] Sajjad, M., Khan, S., Muhammad, K., Wu, W., Ullah, A. and Baik, S.W., 2019. Multi-grade brain tumor classification using deep CNN with extensive data augmentation. Journal of computational science, 30, pp.174-182.

[59] https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-researchand-possibly-the-world/. Accessed January 8, 2019

[60] Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

[61] Malik, J., Devecioglu, O.C., Kiranyaz, S., Ince, T. and Gabbouj, M., 2021. Real-time patient-specific ecg classification by 1d self-operational neural networks. IEEE Transactions on Biomedical Engineering, 69(5), pp.1788-1801.

[62] Goceri, E., 2019. Diagnosis of Alzheimer's disease with Sobolev gradient-based optimization and 3D convolutional neural network. International journal for numerical methods in biomedical engineering, 35(7), p.e3225.

[63] Gupta, A., Ayhan, M. and Maida, A., 2013, May. Natural image bases to represent neuroimaging data. In International conference on machine learning (pp. 987-994). PMLR.

[64] Payan, A. and Montana, G., 2015. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks. arXiv preprint arXiv:1502.02506.

[65] Oh, K., Chung, Y.C., Kim, K.W., Kim, W.S. and Oh, I.S., 2019. Classification and visualization of Alzheimer's disease using volumetric convolutional neural network and transfer learning. Scientific Reports, 9(1), pp.1-16.

[66] https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation accessed on 11th February 2020

[67] Nguyen, A., Yosinski, J. and Clune, J., 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 427-436).

[68] Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[69] Cuingnet, R., Gerardin, E., Tessieras, J., Auzias, G., Lehéricy, S., Habert, M.O., Chupin, M., Benali, H., Colliot, O. and Alzheimer's Disease Neuroimaging Initiative, 2011. Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database. neuroimage, 56(2), pp.766-781.

[70] Baert, A.L., Günther, R.W. and von Schulthess, G.K., 2012. Interventional magnetic resonance imaging. Springer Science & Business Media.

[71] https://ida.loni.usc.edu/home/projectPage.jsp?project=ADNI accessed 11th February 2020.

[72] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), pp.541-551.

[73] LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. and Jackel, L., 1989. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems, 2.

[74] Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H. and Rui, Y., 2016, June. Visualizing and comparing AlexNet and VGG using deconvolutional layers. In Proceedings of the 33 rd International Conference on Machine Learning.

[75] Tang, P., Wang, H. and Kwong, S., 2017. G-MS2F: GoogLeNet based multi-stage feature fusion of deep CNN for scene recognition. Neurocomputing, 225, pp.188-197.

[76] Lin, H. and Jegelka, S., 2018. Resnet with one-neuron hidden layers is a universal approximator. Advances in neural information processing systems, 31.

[77] Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

[78] Jiang, H. and Learned-Miller, E., 2017, May. Face detection with the faster R-CNN. In 2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017) (pp. 650-657). IEEE.

[79] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), pp.834-848.

[80] Kendall, A. and Gal, Y., 2017. What uncertainties do we need in bayesian deep learning for computer vision?. Advances in neural information processing systems, 30.

[81] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

[82] Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[83] Khagi, B. and Kwon, G.R., 2020. 3D CNN design for the classification of Alzheimer's disease using brain MRI and PET. IEEE Access, 8, pp.217830-217847.

[84] Gülçehre, Ç. and Bengio, Y., 2016. Knowledge matters: Importance of prior information for optimization. The Journal of Machine Learning Research, 17(1), pp.226-257.

[85] Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

[86] Wiesler, S. and Ney, H., 2011. A convergence analysis of log-linear training. Advances in Neural Information Processing Systems, 24.

[87] He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).

[88] Prewitt, J.M., 1970. Object enhancement and extraction. Picture processing and Psychopictorics, 10(1), pp.15-19.

[89] Lewitt, R.M. and Matej, S., 2003. Overview of methods for image reconstruction from projections in emission computed tomography. Proceedings of the IEEE, 91(10), pp.1588-1611.

[90] Gonzalez, R.C. and Woods, R.E., 2002. Digital Image Processing, New Jersey.

[91] Sobel, I., 2014. History and definition of the Sobel operator. Retrieved from the World Wide Web, 1505.

[92] Zhang, C., Bengio, S., Hardt, M., Recht, B. and Vinyals, O., 2021. Understanding deep learning (still) requires rethinking generalization. Communications of the ACM, 64(3), pp.107-115.

[93] Robbins, H. and Monro, S., 1951. A stochastic approximation method. The annals of mathematical statistics, pp.400-407.

[94] ter Haar Romeny, B.M., 2003. The gaussian kernel. Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematics, pp.37-51.

[95] Lindeberg, T., 1990. Scale-space for discrete signals. IEEE transactions on pattern analysis and machine intelligence, 12(3), pp.234-254.

[96] Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT press.

[97] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), pp.1929-1958.

[98] Fei-Fei, L., Fergus, R. and Perona, P., 2004, June. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In 2004 conference on computer vision and pattern recognition workshop (pp. 178-178). IEEE.

[99] Marcus, D.S., Wang, T.H., Parker, J., Csernansky, J.G., Morris, J.C. and Buckner, R.L., 2007. Open Access Series of Imaging Studies (OASIS): cross-sectional MRI data in young, middle aged, nondemented, and demented older adults. Journal of cognitive neuroscience,

19(9), pp.1498-1507.

[100] Khagi, B., Lee, C.G. and Kwon, G.R., 2018, November. Alzheimer’s disease Classification from Brain MRI based on transfer learning from CNN. In 2018 11th biomedical engineering international conference (BMEiCON) (pp. 1-4). IEEE.

[101] Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2014. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[102] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2015. Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence, 38(1), pp.142-158.

[103] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H., 2015. Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.\

[104] Ribeiro, M.T., Singh, S. and Guestrin, C., 2016, August. " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[105] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[106] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. and Batra, D., 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

[107] Shrestha, A. and Mahmood, A., 2019. Review of deep learning algorithms and architectures. IEEE access, 7, pp.53040-53065.

[108] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L., 2021. Review of deep

learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8(1), pp.1-74.

[109] Voigtlaender, P., Luiten, J., Torr, P.H. and Leibe, B., 2020. Siam r-cnn: Visual tracking by re-detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 6578-6588).

[110] Wu, Y. and He, K., 2018. Group normalization. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

[111] Ba, J.L., Kiros, J.R. and Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

[112] Dai, Z. and Heckel, R., 2019. Channel normalization in convolutional neural network avoids vanishing gradients. arXiv preprint arXiv:1907.09539.

[113] Ulyanov, D., Vedaldi, A. and Lempitsky, V., 2016. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.

[114] Bengio, Y. and LeCun, Y., 2007. Scaling learning algorithms towards AI. Large-scale kernel machines, 34(5), pp.1-41.

[115] Glorot, X. and Bengio, Y., 2010, March. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.

[116] Ramachandran, P., Zoph, B. and Le, Q.V., 2017. Searching for activation functions. arXiv preprint arXiv:1710.05941.

[117] Farabet, C., Couprie, C., Najman, L. and LeCun, Y., 2012. Learning hierarchical features for scene labeling. IEEE transactions on pattern analysis and machine intelligence, 35(8), pp.1915-1929.

[118] Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., Van Der Laak, J.A., Van Ginneken, B. and Sánchez, C.I., 2017. A survey on deep learning in medical image analysis. Medical image analysis, 42, pp.60-88.

[119] Golilarz, N.A. and Demirel, H., 2017, September. Thresholding neural network (TNN) with smooth sigmoid based shrinkage (SSBS) function for image de-noising. In 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 67-71). IEEE.

[120] Gregor, K., Danihelka, I., Graves, A., Rezende, D. and Wierstra, D., 2015, June. Draw: A recurrent neural network for image generation. In International conference on machine learning (pp. 1462-1471). PMLR.

[121] Shen, D., Wu, G. and Suk, H.I., 2017. Deep learning in medical image analysis. Annual review of biomedical engineering, 19, p.221.

[122] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), pp.211-252.

[123] Allen-Zhu, Z. and Li, Y., 2019. What can resnet learn efficiently, going beyond kernels? Advances in Neural Information Processing Systems, 32.

[124] Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K., 2017. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).

[125] Klambauer, G., Unterthiner, T., Mayr, A. and Hochreiter, S., 2017. Self-normalizing neural networks. Advances in neural information processing systems, 30.

[126] Nair, V. and Hinton, G.E., 2010, January. Rectified linear units improve restricted boltzmann machines. In ICML.

[127]

[128] Maas, A.L., Hannun, A.Y. and Ng, A.Y., 2013, June. Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML (Vol. 30, No. 1, p. 3).

[129] Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853.

[130] Hendrycks, D. and Gimpel, K., 2016. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415.

[131] Clevert, D.A., Unterthiner, T. and Hochreiter, S., 2015. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[132] Anwar, S.M., Majid, M., Qayyum, A., Awais, M., Alnowami, M. and Khan, M.K., 2018. Medical image analysis using convolutional neural networks: a review. Journal of medical systems, 42(11), pp.1-13.

[133] Helaly, H.A., Badawy, M. and Haikal, A.Y., 2021. Deep learning approach for early detection of Alzheimer’s disease. Cognitive computation, pp.1-17.

[134] LeCun, Y.A., Bottou, L., Orr, G.B. and Müller, K.R., 2012. Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-48). Springer, Berlin, Heidelberg.

[135] Mohamad, G. and Asl, E.H., 2016. Alzheimer's disease diagnostics by a deeply supervised adaptable 3D convolutional network.

[136] Payan, A. and Montana, G., 2015. Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks. arXiv preprint arXiv:1502.02506.

[137] Abuhmed, T., El-Sappagh, S. and Alonso, J.M., 2021. Robust hybrid deep learning models for Alzheimer’s progression detection. Knowledge-Based Systems, 213, p.106688.

[138] Acharya, U.R., Oh, S.L., Hagiwara, Y., Tan, J.H. and Adeli, H., 2018. Deep convolutional neural network for the automated detection and diagnosis of seizure using EEG signals.

Computers in biology and medicine, 100, pp.270-278.

[139] Goceri, E., 2019. Diagnosis of Alzheimer's disease with Sobolev gradient-based optimization and 3D convolutional neural network. International journal for numerical methods in biomedical engineering, 35(7), p.e3225.

[140] Huang, Z., Du, X., Chen, L., Li, Y., Liu, M., Chou, Y. and Jin, L., 2020. Convolutional neural network based on complex networks for brain tumor image classification with a modified activation function. IEEE Access, 8, pp.89281-89290.

[141] Virtue, P., Stella, X.Y. and Lustig, M., 2017, September. Better than real: Complex-valued neural nets for MRI fingerprinting. In 2017 IEEE international conference on image processing (ICIP) (pp. 3953-3957). IEEE.

[142] Sharma, R., Goel, T., Tanveer, M., Dwivedi, S. and Murugan, R., 2021. FAF-DRVFL: Fuzzy activation function based deep random vector functional links network for early diagnosis of Alzheimer disease. Applied Soft Computing, 106, p.107371.

[143] Shin, H.C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D. and Summers, R.M., 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE transactions on medical imaging, 35(5), pp.1285-1298.

[144] McKesson, J.L., 2012. Learning Modern 3D Graphics Programming. Arcsynthesis. org, 17.

[145] Chen, C., Bai, W. and Rueckert, D., 2018, September. Multi-task learning for left atrial segmentation on GE-MRI. In International workshop on statistical atlases and computational models of the heart (pp. 292-301). Springer, Cham.

[146] Hong, J., Feng, Z., Wang, S.H., Peet, A., Zhang, Y.D., Sun, Y. and Yang, M., 2020. Brain age prediction of children using routine brain MR images via deep learning. Frontiers in Neurology, 11, p.584682.

[147] Zhang, Y.D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K. and Wang, S.H., 2019. Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. Multimedia Tools and Applications, 78(3), pp.3613-3632.

[148] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[149] Antiqueira, L., Rodrigues, F.A., van Wijk, B.C., Costa, L.D.F. and Daffertshofer, A., 2010. Estimating complex cortical networks via surface recordings—a critical note. Neuroimage, 53(2), pp.439-449.

Appendix

Appendix I

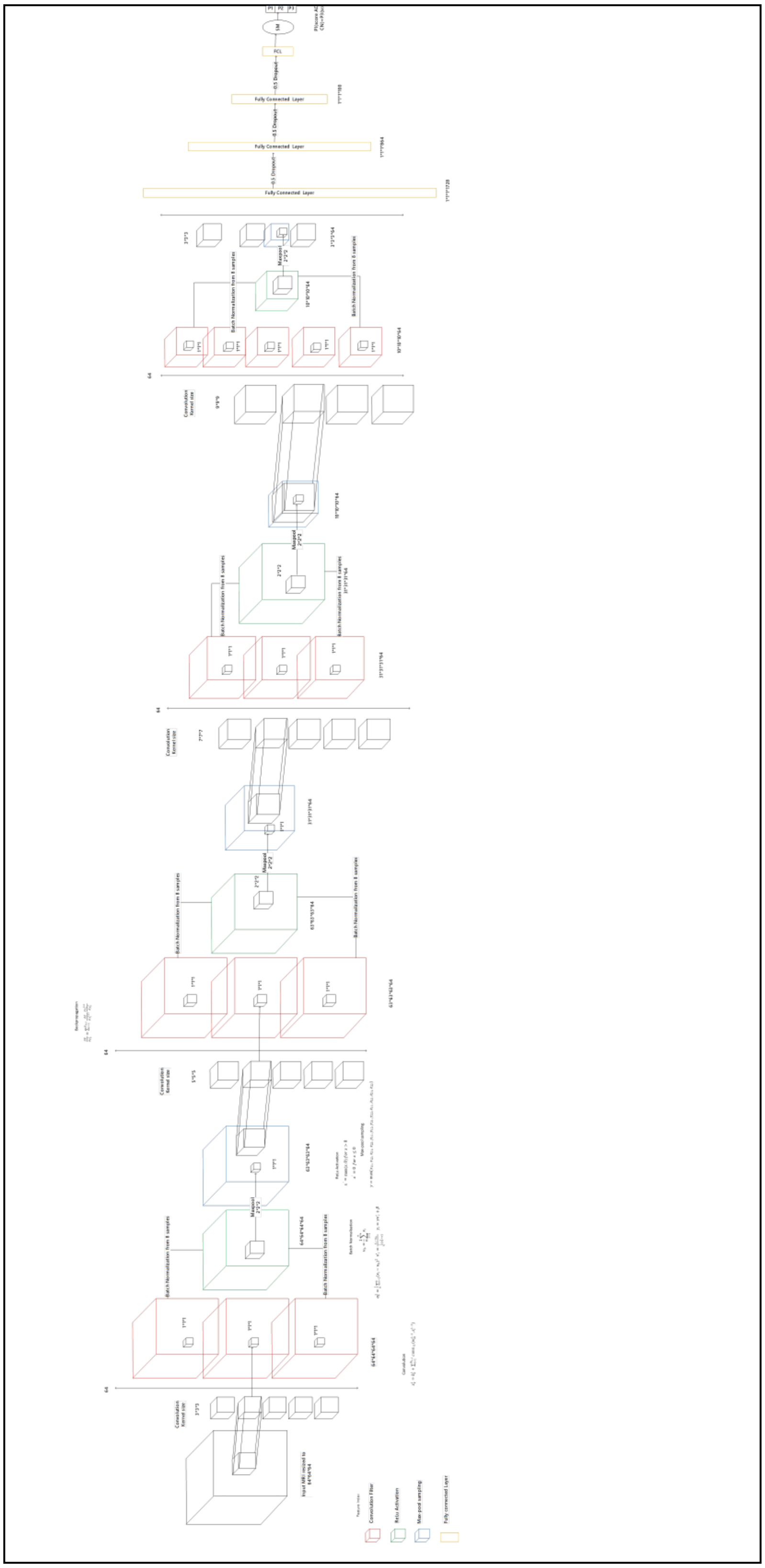
Table I: Demographics details

MRI/PET TYPE	Dataset type	AD participants	CN participants	MCI participants
MRI1	Male/Female	29/36	22/38	54/33
	Mean age	73.55/75.43	75.57/74.43	77.06/72.41
	BASELINE_MRI	65	60	87
MRI2	BASELINE_MRI_SMALL	28	31	48
PET1	BASELINE_PET_SMALL	102	109	337
PET2	BASELINE_PET_ALL	136	109	337

Table II: MRI and PET Types as discussed in Table 4.4

MRI2	MPR-R; GradWarp; B1 Correction; N3; Scaled
MPR; ; N3; Scaled	MPR; GradWarp; B1 Correction; N3; Scaled_2
MPR-R; ; N3	MPR-R; GradWarp; B1 Correction; N3; Scaled_2
MPR; ; N3; Scaled_2	MPR-R; GradWarp; N3; Scaled
MPR; ; N3	MRI1
MPR-R; ; N3; Scaled_2	MPR; GradWarp; B1 Correction; N3
MPR-R; ; N3; Scaled	MPR; GradWarp; B1 Correction; N3; Scaled
MPR; GradWarp; N3	MPR-R; GradWarp; B1 Correction; N3
MPR; GradWarp; N3; Scaled	MPR-R; GradWarp; B1 Correction; N3; Scaled
MPR; GradWarp	MPR; GradWarp; B1 Correction; N3; Scaled_2
MPR; GradWarp; N3; Scaled_2	MPR-R; GradWarp; B1 Correction; N3; Scaled_2
MPR-R; GradWarp; N3	PET2
MPR-R; GradWarp	Coreg, _Avg, _Standardized_Image_and_Voxel_Size
MPR; GradWarp; B1 Correction; N3	Coreg, _Avg, _Std_Img_and_Vox_Siz,_Uniform_Resolution
MPR; GradWarp; B1 Correction; N3; Scaled	Co-registered, _Averaged
MPR; GradWarp; B1 Correction	Co-registered_Dynamic
MPR-R; GradWarp; B1 Correction; N3	PET1
MPR-R; GradWarp; B1 Correction	Coreg, _Avg,_Standardized_Image_and_Voxel_Size

Figure 2.2: Proposed 3D CNN architecture for the MRI/PET classification on the basis of the diverging area of the reception, which is referred to as 'divNet'



Appendix II

C1: MATLAB code implementation for GAP layer used after convolutional layer as an alternative to Batch normalization layer.

```
encoder_1 = [
       convolution2dLayer([3
3],32,'Padding','same','WeightsInitializer','glorot');
       test_gauss(32,'g1',3); % batchNormalizationLayer for BN
       leakyReLULayer('Name','ReLU_s1');
maxPooling2dLayer(poolSize,'Stride',2);
]
```

The defination for test_gauss is as below:

```
classdef test_gauss < nnet.layer.Layer
    properties(Learnable)
        Alpha
        Beta
        Gamma
    end
    properties
    hsize
    end
    methods
        function layer = test_gaus(numChannels,name,hsize)
            layer.Name = name;
            layer.hsize=hsize;
            layer.Description = "Scale factor" +numChannels + "
channels";
            layer.Alpha = rand([1 1 numChannels]); %3D rand([1 1 1
numChannels]);
            layer.Beta = rand([1 1 numChannels]);
            layer.Gamma= rand([1 1 numChannels]);
        end
        function Z = predict(layer, X)
            X_mean = double(mean(X,3));    %
X(w1,w2,fiternumber,batchsize)
            X_std = std(X,0,3); % For 3D 4th dimension is used for mean
and std
            X_F=(X-X_mean)./X_std;
            h_size=layer.hsize;

            h_f = [h]; % h_f = convn(h,h,'same')for 2nd gap layer
            X_test=imfilter(X_F,h_f,'replicate');
            Z=layer.Alpha.*X+layer.Beta.*(X-X_test)+ layer.Gamma;
         end
        function [dLdX, dLdAlpha,dLdBeta,dLdGamma] = backward(layer, X,
~, dLdZ, ~)
              h_size=layer.hsize;
              h = fspecial3('Gaussian',h_size);
          X_mean = double(mean(X,3));    %X(w1,w2,fiternumber,
batchsize)
          X_std = std(X,0,3);
          X_F=(X-X_mean)./X_std;
          h_f = [h];
          X_test=imfilter(X_F,h_f,'replicate');
        dLdX=(layer.Alpha.*dLdZ+layer.Beta.*dLdZ);
        dLdAlpha = X.* dLdZ;
        dLdAlpha = sum(dLdAlpha,[1 2]); % for 3D dLdBeta =
sum(dLdBeta,[1 2 3]);
        dLdAlpha = sum(dLdAlpha,4); % for 3D dLdAlpha = sum(dLdAlpha,5);
        dLdBeta = (X-X_test).* dLdZ;
        dLdBeta = sum(dLdBeta,[1 2]);
        dLdBeta = sum(dLdBeta,4);
        dLdGamma= dLdZ;
        dLdGamma = sum(dLdGamma,[1 2]);
        dLdGamma = sum(dLdGamma,4);
        end
    end
end
```

C2: Code snippet for feature visualization and mean response as in Figure 4.11 and Figure 4.12.

To visualize the feature and get mean response of the proposed layer as in Figure 4.11.1 (b) to 4.11.4(b), where X and Z represent the input and output to the predict function, respectively.

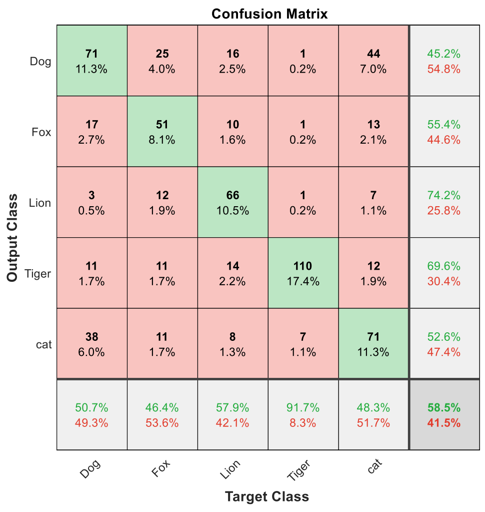
```
…
Load net,X,Z % Need to save net, X and Z layer as .mat file
previously.
feature=gather(X);
x(:)=gather(mean(X,[1 2]));
x=x';
z(:)=gather(mean(Z,[1 2]));
z=z';
plot(x)
hold on
plot(z)
hold off
Dsize=size(X,4); %for 3D CNN size(X,5)
for i=1:Dsize
A=gather(X(:,:,:,i)); %for 3D CNN 'i' in 5th dimension i.e minibatch
representing test samples
B=gather(Z(:,:,:,i));
R (:,:,i)= corrcoef(A,B);
```

```
plot(R)
end
...
```

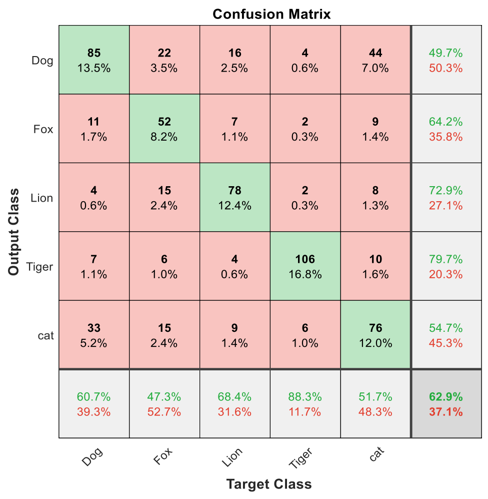
C3: Code snippet implementation for generating feature map using Deep dream as shown in Figure 3.5.

```
...
load net; % Load Pretrained Network
layer =22; % Last FC layer of the network
trained_net.Layers(end).ClassNames(channels)
% Generate detailed images that strongly activate these classes.
I = deepDreamImage(trained_net,layer,channels, ...
    'Verbose',false, ...
    'NumIterations',50);
figure
montage(I)
name = net.Layers(layer).Name;
title(['Layer ',name,' Features'])
```

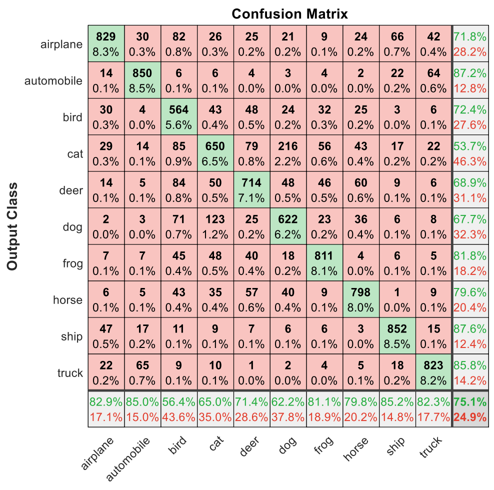
Confusion Matrix for dataset (architecture)



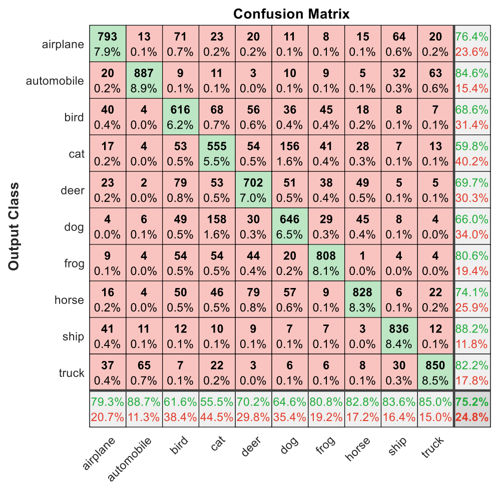
5-animals (b1b2b3b4)



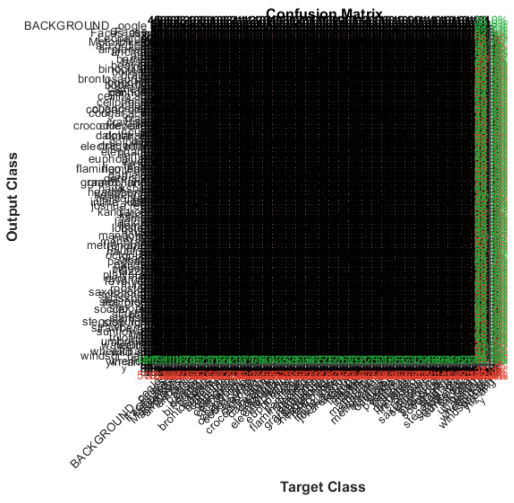
5-animals (g1b2b3b4)



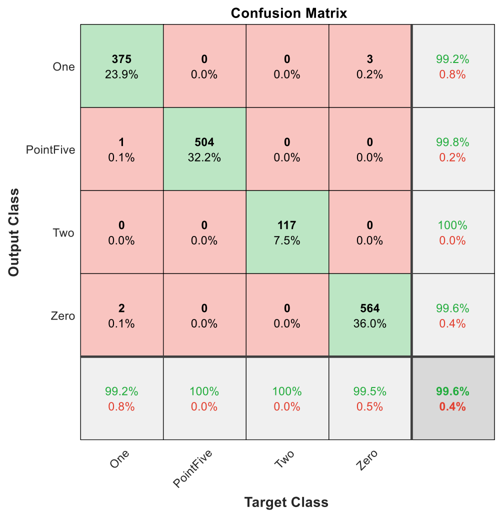
CIFAR-10 (b1b2b3b4)



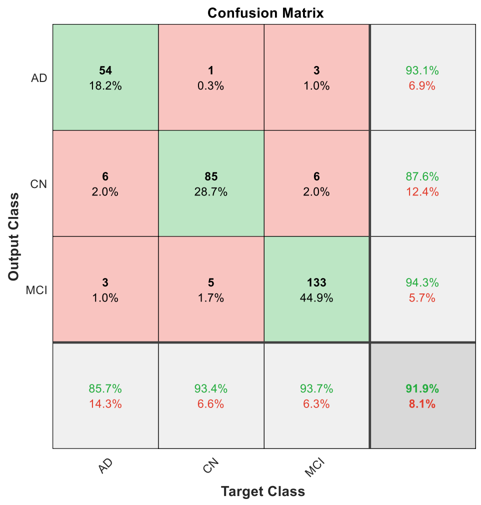
CIFAR-10 (g1g2b3b4)



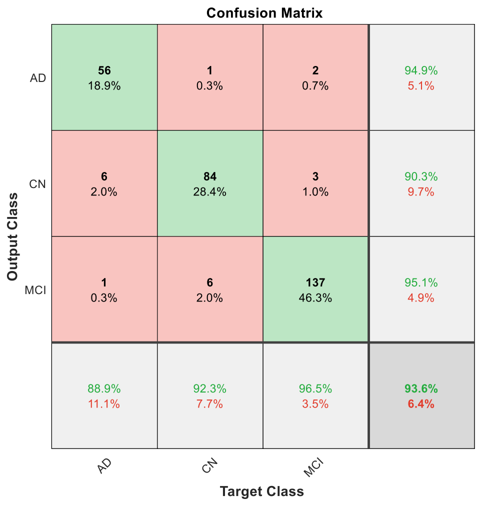
Caltech-102 (g1b2b3b4)



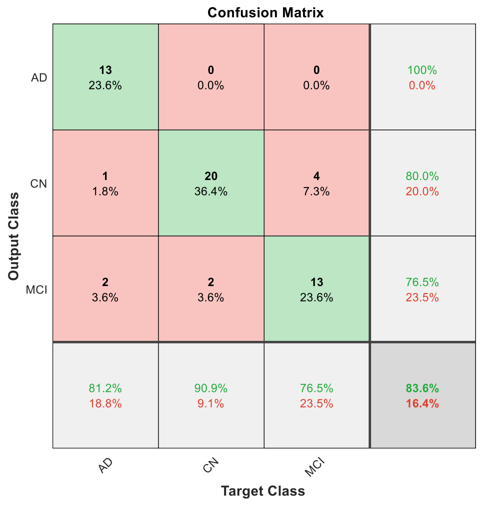
OASIS_MRI_CDR (g1b2b3b4)



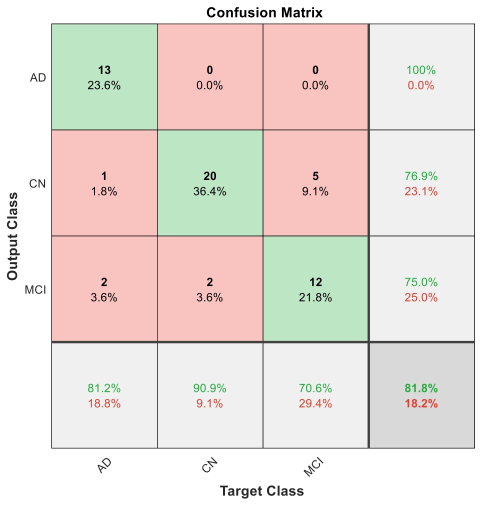
ADNI MRI_BASELINE (b1b2b3b4)



ADNI MRI_BASELINE (g1g2b3b4)



ADNI MRI_SMALL (b1b2b3b4)



ADNI MRI_SMALL (g1g2b3b4)

Appendix III

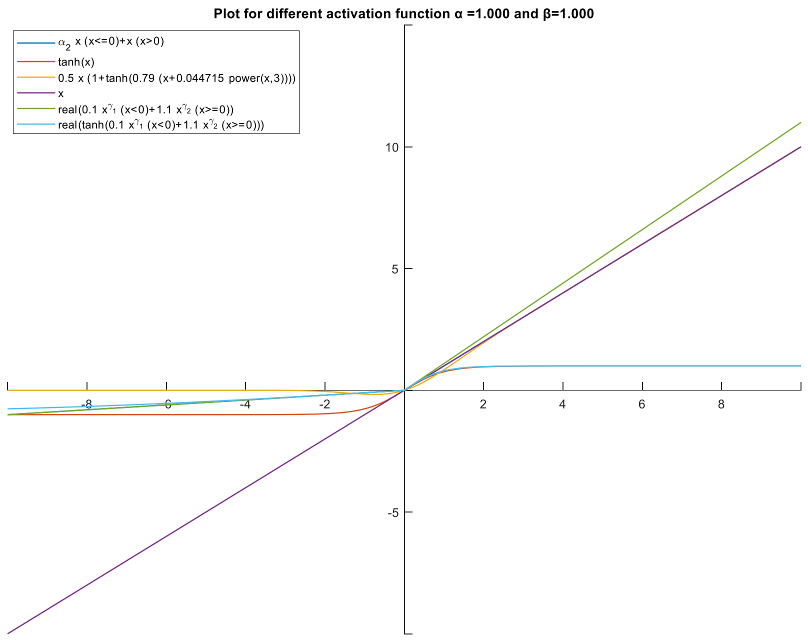


Figure 4.2(a)_app: Here the last sky blue graph represent the proposed-SGT function.

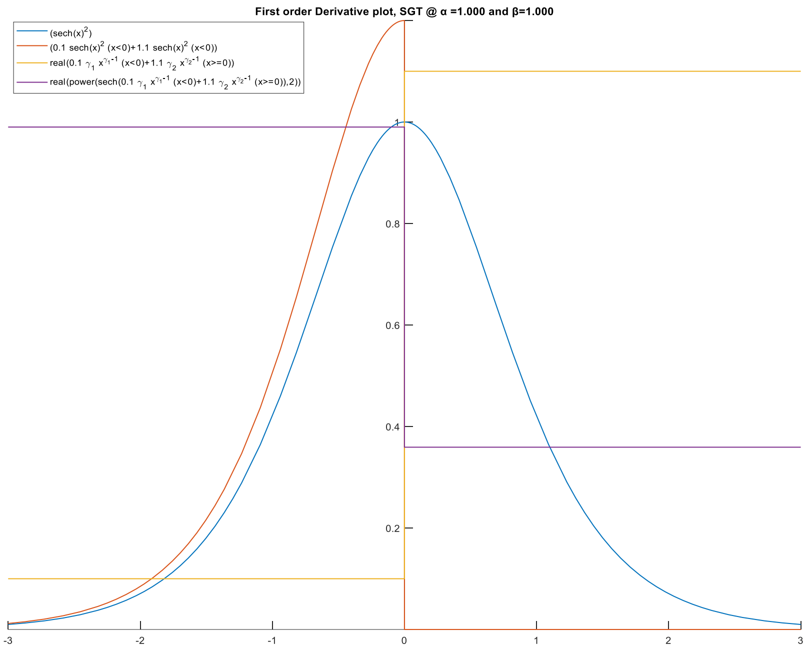
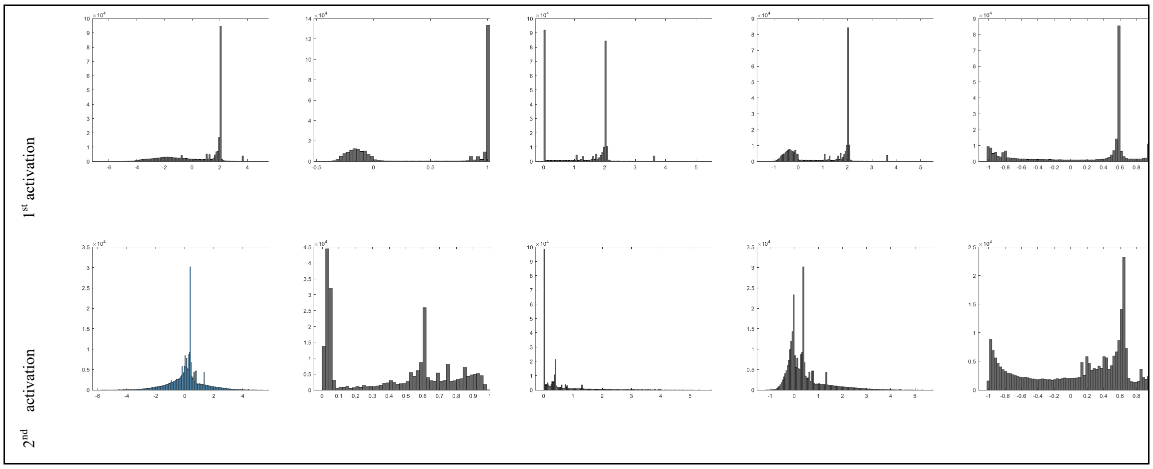


Figure 4.2(b)_app: Here the last purple curve represents the derivate of proposed-SGT function.



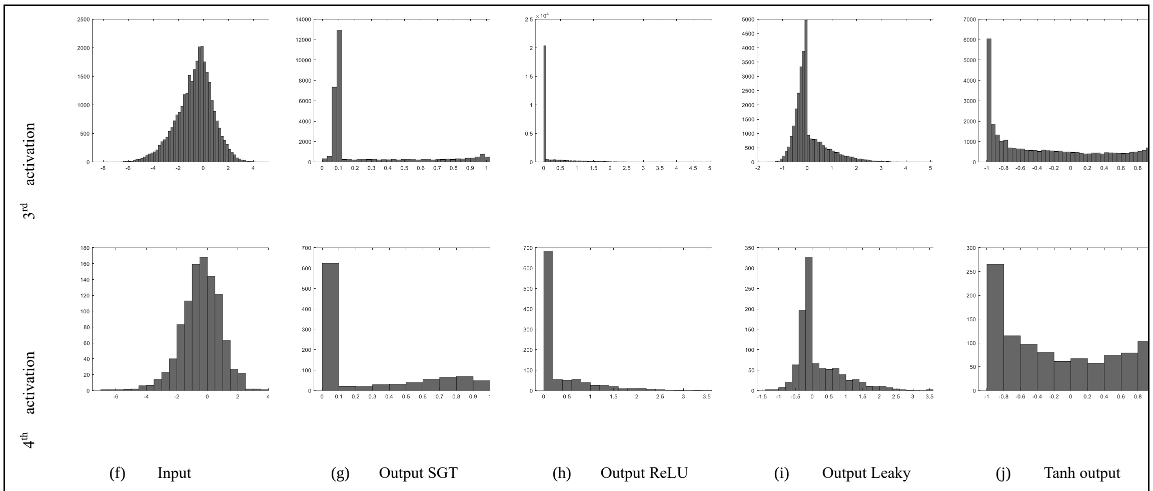
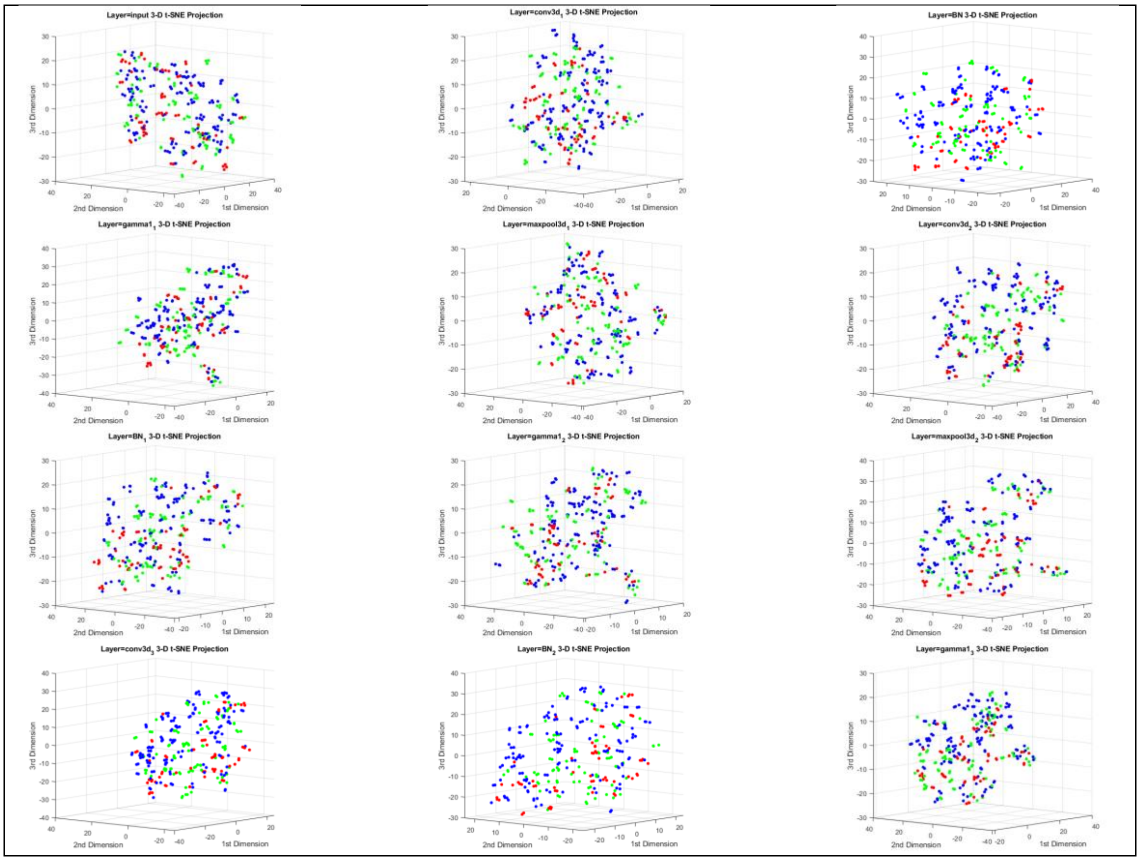


Figure 4.16_app: Histogram plot of 19th filter out of 64 filters for the input features against output using various activation functions for a single MRI input plotted for different layers i.e 4, 8, 12 and 16 (see Table 2 for layers and Figure 4 for all channels histogram)



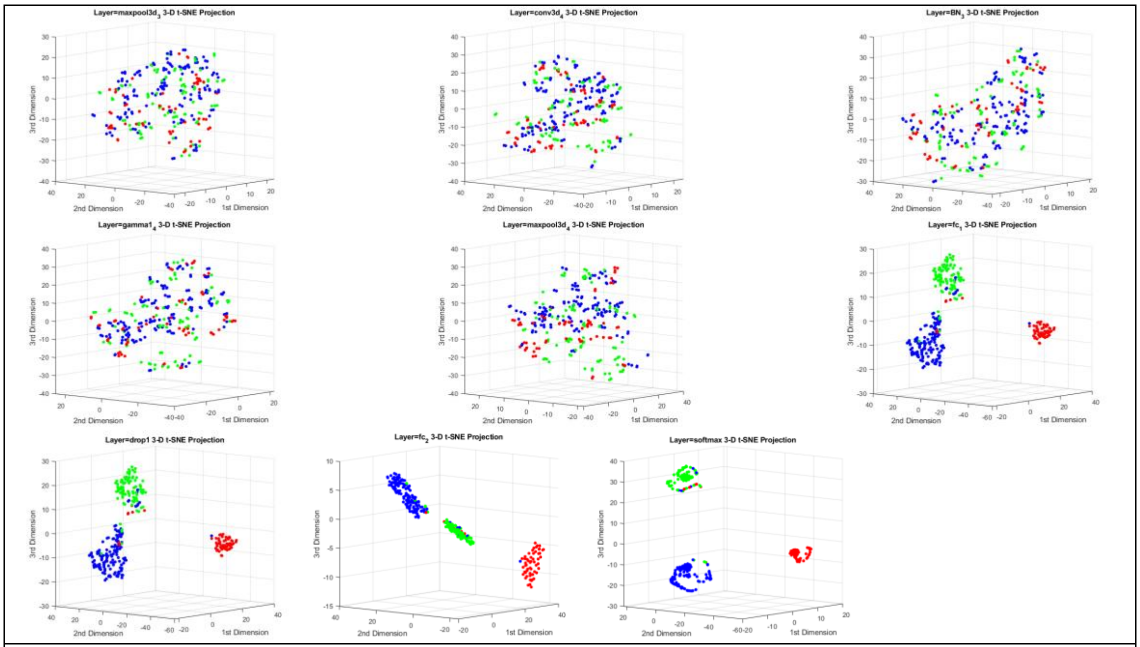


Figure 4.21_app: 3D t-SNE projection of all individual layers in gamm4 network. Here each red dots represents an AD subject MRI, blue dot represents MCI and green dot represents CN MRI.

Matlab code implementation for `layer_gamma3d` **layer as in Table 4.6**

```
classdef layer_gamma3d < nnet.layer.Layer
    properties (Learnable)
        Alpha
        Beta
    end

    methods
        function layer = layer_gamma3d(numchannel,name)
            layer.Name = name;
            layer.Description = " Proposed SGT layer with" +numchannel
+ " channels";
            layer.Alpha = rand([1 1 1 numchannel]);
            layer.Beta = rand([1 1 1 numchannel]);
        end

        function Z = predict(layer, X)
             X(isnan(X)) = 0.1;           %for NaN case
             layer.Alpha(isnan(layer.Alpha))= 0.1;
             layer.Beta(isnan(layer.Beta))= 0.1;
             layer.Beta=abs(layer.Beta);
             layer.Alpha=abs(layer.Alpha);
             X_F= 0.1.*(power(complex(X),complex(layer.Alpha,0)));
             check_X = 1.1*(power(complex(X),complex(layer.Beta,0)));
            X_F(X>0) = check_X(X>0);
            Z = tanh(real(X_F));
        end

        function [dLdX,dLdAlpha,dLdBeta] = backward(layer, X, ~, dLdZ,
~)
        X(isnan(X))= 0.001;                      %for NaN case
        dLdZ(isnan(dLdZ))= 0.001;
        layer.Alpha(isnan(layer.Alpha))= 0.001;
        layer.Beta(isnan(layer.Beta))= 0.001;
        layer.Beta=abs(layer.Beta);
        layer.Alpha=abs(layer.Alpha);
        X_loss  =  0.1.*layer.Alpha.*real(power(complex(X),(layer.Alpha-
1)));
        dLdX = power(sech(X_loss),2).*dLdZ;
        X_loss2   =   1.1*layer.Beta.*real(power(complex(X),(layer.Beta-
1)));
        check = power(sech(X_loss2),2).*dLdZ;
        dLdX(X>0) = check(X>0);
        dLdAlpha                                                       =
0.1.*real((log10(complex(X))).*(real(power(complex(X),complex(layer.A
lpha,0))).*(X<0))).*dLdZ;
        dLdAlpha = sum(dLdAlpha,[1 2 3]);
        dLdAlpha = sum(dLdAlpha,5);
        dLdBeta                                                        =
1.1*real((log10(complex(X))).*(real(power(complex(X),complex(layer.Be
ta,0))).*(X>0))).*dLdZ;
        dLdBeta = sum(dLdBeta,[1 2 3]);
        dLdBeta = sum(dLdBeta,5);
        end
    end
end
```

Outcome of the diagnostic test	**Condition (e.g. Disease) As determined by the Standard of Truth**		
	Positive	**Negative**	**Row Total**
Positive	TP	FP	TP+FP (Total number of subjects with positive test)
Negative	FN	TN	FN + TN (Total number of subjects with negative test)
Column total	TP+FN (Total number of subjects with given condition)	FP+TN (Total number of subjects without given condition)	N = TP+TN+FP+FN (Total number of subjects in study)

Sensitivity = TP/ (TP + FN) = (Number of true positive assessment)/ (Number of all positive assessment)
Specificity = TN/(TN + FP) = (Number of true negative assessment)/(Number of all negative assessment)
Accuracy = (TN + TP)/(TN+TP+FN+FP) = (Number of correct assessments)/Number of all assessment)

Cohen's Kappa score (**κ**)

$$\kappa = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FP) \times (FP + TN) + (TP + FN) \times (FN + TN)}$$

Acknowledgement

I would be failing in my endeavor if I did not convey my gratitude and appreciation to the number of individuals who have made valuable contribution directly and indirectly toward reaching this academic milestone.

First and foremost, I would like to express my special appreciation and sincere gratitude to my Ph.D. advisor, Prof. Dr. Goo-Rak Kwon, who has been a truly dedicated and tremendous mentor for encouraging my research, managing funds and allowing me to grow as a researcher. Under his supervision I had started to crawl, stand, walk, and run in the field of research. And one day, I hope I can fly high into the sky. Similarly, I want to thank all the professors and teachers I have met in Chosun University for sharing their knowledge and help me grow each day. Here, I would like to immensely thank my two seniors Dr. Debesh Jha and Kishor Singh, for their support to bring me into the academic research. Without their support, I would have been stuck in some boring office work in Nepal. My sincere gratitude to Dr. Ramesh Kumar Lama, Dr. Ji-In Kim, Uttam Khatri and all the past and present members of Digital media computing lab for their support and acquaintance. Also, throughout my stay from Day-1 in S.Korea I had a wonderful Nepalese community to spend holidays and festival, so that I really had some good times here, though missing home occasionally. Now I really feel that we hit the road when missing home and miss the sun when it starts to snow.

And above all, it’s the blessing of my parents, my lovely mom and dad, also love of my siblings without which I can achieve nothing in my life. Their words have empowered me to go through all the pain and hardship and become more stronger man. To love them and be a good son will be my never-ending job more than a Ph.D. or anything else.

-Bijen Khagi

August, 2022

List of Publications and Proceedings

Khagi, B. and Kwon, G.R., 2021. Convolutional Neural Network-Based Natural Image and MRI Classification Using Gaussian Activated Parametric (GAP) Layer. IEEE Access, 9, pp.96930-96947.

Khagi, B. and Kwon, G.R., 2020. 3D CNN design for the classification of Alzheimer's disease using brain MRI and PET. IEEE Access, 8, pp.217830-217847.

Khagi, B., Lee, K.H., Choi, K.Y., Lee, J.J., Kwon, G.R. and Yang, H.D., 2021. VBM-Based Alzheimer's Disease Detection from the Region of Interest of T1 MRI with Supportive Gaussian Smoothing and a Bayesian Regularized Neural Network. Applied Sciences, 11(13), p.6175.

Khagi, B. and Kwon, G.R., 2021. 3D CNN based Alzheimer's diseases classification using segmented Grey matter extracted from whole-brain MRI. JOIV: International Journal on Informatics Visualization, 5(2), pp.200-205.

Khagi, B. and Kwon, G.R., "Improving CNN result with image-smoothing-filter for medical MR images classification", Advances in Alzheimer's and Parkinson's Therapies An AT-AD/PD™ Focus Meeting; April 2- April 5, 2020, Vienna, Austria (ePoster)