Spring 2019

Classification of Parkinson's Disease Using MRI Data and Deep Learning Convolution Neural Networks

Roshni Saha

Classification of Parkinson's Disease Using MRI Data and Deep Learning Convolution Neural Networks

by

Roshni Saha

A creative component report submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

Master of Science

Major: Information Systems

Program of Study Committee:
Dr. Sree Nilakanta, Major Professor
Dr. Vivekananda Roy, Minor Professor
Dr. Auriel A Willette, Committee Member
Mohammadmahdi Moqri, Committee Member

The student author, whose presentation was approved by the program of study committee, is solely responsible for the content of this report. The Graduate College will ensure this report is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

ACKNOWLEDGMENTS

I would like to thank my Major Professor, Dr. Sree Nilakanta for his extensive academic guidance and patience during my research. His insights and knowledge of the subject kept me motivated and steered me in the right direction. I would also like to express immense gratitude towards my Minor Professor, Dr. Vivekananda Roy for his valuable suggestions and continuous support. I offer my sincere appreciation for the learning opportunities provided by my committee members Dr. Auriel A Willette and Professor Mohammadmahdi Moqri.

Also, I would like to take this opportunity to express my profound gratitude to my parents who have unconditionally devoted their love and supported my career goals. It is because of my parents that I could come to Iowa State University to pursue my master's degree.

TABLE OF CONTENTS

FIGURES AND TABLES

GLOSSARY OF TERMS

Term	Description
PD	Parkinson's Disease
PPMI	Parkinson's Progression Markers Initiative
ANN	Artificial Neural Network
CNN	Convolution Neural Network
MRI	Magnetic Resonance Imaging
MCI	Mild Cognitive Impairment
fMRI	Functional Magnetic Resonance Imaging
ADNI	Alzheimer's Disease Neuroimaging Initiative
AD	Alzheimer's Disease
NMV	Net Magnetization Vector
ReLu	Rectified Linear Unit
NIFTI	Neuroimaging Informatics Technology Initiative
MMSE	Mini Mental State Examination
PDMCI	Parkinson's Disease with Mild Cognitive Impairment
PDD	Parkinson's Disease and Dementia
ROI	Region of Interest
PET	Positron emission tomography

ABSTRACT

Machine Learning techniques and algorithms play a significant role in pattern recognition in biomedical sciences. These techniques have been assisting researchers in classification of medical images and prediction of models to have a comprehensive understanding of complex medical problems. Deep learning is the subfield of machine learning algorithms which deals with the structure and the function of the brain known as artificial neural network(ANN). In this paper, Convolution Neural Network has been applied for the classification of neural brain images to discriminate or identify Parkinson's Disease (PD) affected brains from normal healthy brains. It is challenging to classify complex clinical data to detect diseases like PD or to estimate the stage of the disease. Convolution neural network is a machine learning algorithm which learns from rich data sources such as MRI, fMRI and likewise and develops predictive models for accurate image classification. Using CNN, we successfully classified MRI data of PD patients from normal controls and attained an accuracy of 97.63% without batch normalization and 97.91% with batch normalization respectively. This endeavor suggests that CNN has the potential to extract the most discriminative features of these complex clinical data and hence the same architecture has the capacity to be effectively used in our research to perform other medical image classification or more complicated systems.

INTRODUCTION

Parkinson's disease (PD) is a neurodegenerative disease first described by James Parkinson. It is one of the common causes of neurological illness among older adults affecting about 1% of the population over 60 years old. PD is mainly characterized by motor disorder (and other neuropsychiatric symptoms) and impairment in cognitive function and is diagnosed by symptoms such as slowed movements (bradykinesia), muscle rigidity, and tremor (at rest). There are many other associated signs of PD, including expressionless face, quiet speech, cramped handwriting, shuffling gait, trouble getting out of a chair, and difficulty swallowing.

Recent clinical, pathological and community-based studies suggest that cognitive decline is marked by a frontal-subcortical impairment progressing to dementia when posterior and cortical deficits are present during middle to late stages of PD.

Dementia is a chronic or persistent disorder and decline of mental functions caused by brain disease or injury and characterized by memory disorders, personality changes, and impaired reasoning and it affects around 40% of PD patients. However, dementia should be differentiated from mild cognitive impairment (MCI) in order to enable earlier therapeutic intervention to prevent cognitive decline in PD. The dementia of PD is called "subcortical" because of the location of affected brain areas, and subcortical dementias have somewhat different clinical symptoms than a "cortical" dementia like AD. Neuroanatomic biomarkers provided by MRI helps in the early diagnosis of this condition.

STATE OF ART

The Deep Learning Convolution Neural Networks (CNN) defines the state of art for widespread applications such as image recognition problem, segmentation and retrieval. This new paradigm has produced fantastic results in recent years in analyzing the content of images, speech and videos. The current state-of-the-art Convolutional Neural Networks achieve accuracies that surpass human-level performance. Our aim of this work is to adapt CNN network to develop trained predictive models that will efficiently classify Parkinson's Disease affected neural images from healthy normal neural images. CNN uses supervised learning algorithm.

PROBLEM STATEMENT

In most areas of clinical diagnosis, prevention is better than cure. About 1% of the population over 60 years old are affected by Parkinson's Disease (PD). Hence, early diagnosis of the PD is essential for better treatment. Diagnosis of PD requires very careful medical assessments. Neuroanatomic biomarkers provided by magnetic resonance imaging (MRI) helps in the early diagnosis of this condition in PD affected patients or to estimate the stage of the disease. The significance of classifying this kind of clinical medical data is to potentially develop a predictive model to classify or recognize PD affected subjects from normal controls. In this work, we are using the magnetic resonance imaging (MRI) scans and the Convolution neural networks (CNN) to develop the prediction models.

LITERATURE REVIEW

A lot of research has been prevalent for a long time in the medical field on neuroimaging, clinical, cognitive, and biomarker datasets for normal aging and Parkinson's Disease (PD). Population aging has become prevalent all over the globe in both developed and developing countries. Neurogenerative diseases such as PD have high prevalence in all over the world affecting about 1% of the population over 60 years old. The significance of classifying this kind of clinical medical data is to potentially develop a predictive model to classify or recognize PD affected subjects from normal controls or to estimate the stage of the disease.

According to previous studies and the research paper "Medical Image Classification with Convolution Neural Network", in image classification problems, one of the primary challenges is that it is critical to achieve good classification. This mostly is due to the descriptive and discriminative power that the feature vectors extracted from the images contain. These feature vectors extracted from the image data are used to train the classification model. For many years, researchers have studied Artificial Neural Network (ANN) to solve complex image classification problems. In recent times, the deep learning technique, convolutional neural networks (CNN) have shown excellent results for analyzing the image content in image classification. One of the benefits of these neural networks is that they can be generalized to solve different problems using the same design(1).

In this paper (1), the author has demonstrated the superior performance of CNN in solving critical image classification problems in applications such as traffic sign detection and have surpassed human capability in benchmarking tests. However, it is challenging to achieve high accuracy in

classification due to the high visual variation within the same class and the high similarity between different classes. In this noteworthy work, a single convolutional layer architecture was used to reduce the number of parameters in the CNN model to avoid the over-fitting problem.

The paper "Classification of Alzheimer's Disease Using fMRI Data and Deep Learning Convolutional Neural Networks" talks about the challenges of feature selection and reduction in image classification. The paper demonstrates about the challenges of selecting the most discriminative features required for building the classification model. In this paper, some of the Convolutional Neural Network (CNN) architectures have been discussed which successfully classified functional MRI data of Alzheimer's subjects from normal control subjects (3).

The paper "A Bayesian network decision model for supporting the diagnosis of dementia, Alzheimer's disease and mild cognitive impairment" proposes a Bayesian network decision model for diagnosis of dementia, AD and mild cognitive decline (MCI) as these networks can represent the causality and uncertainty in clinical medical images (2).

Another paper of relevance in Bayesian structure learning is "Predicting dementia development in Parkinson's disease using Bayesian network classifiers". This paper proposes the use of a multivariate filter-based Naïve Bayes model as the best classifier to distinguish between dementia in PD patients from healthy brains and shows the highest cross-validated sensitivity (4).

This paper (4) presents how researchers have studied surrogate, neuro anatomic biomarkers provided by magnetic resonance imaging(MRI) and comprehended that the MRI scans help in

early diagnosis of the disease. In this article, four classification models(Naive Bayes, multivariate filter-based Naive Bayes, filter selective Naive Bayes and support vector machines, SVM) have been applied to evaluate their capacity to discriminate between patients with Parkinson's disease(PDCI), PDMCI and PDD (4).

This idea of using the magnetic resonance imaging(MRI) scans and the use of Convolution neural networks (CNN) motivated our algorithm to use several particular convolution neural network architectures with and without batch normalization, over subset of variables, rather than all the variables at once in order to build a good image classification model. We have also applied the LeNet-5 model which has been designed and handwritten digit and machine-printed character recognition. However, the best model selected for this work is the model where dropout algorithm and batch normalization has been implemented.

DATA ACQUISITION AND PREPROCESSING

For this study, 30 PD patients and 24 elderly normal control subjects with an age range of 60-75 years were selected from the Parkinson's Progression Markers Initiative (PPMI) dataset. The PD patients and all normal participants were healthy and had no reported history of medical or neurological conditions. Scanning was performed on a Siemens Trio 3 Tesla MRI scanner. Anatomical scans were acquired with a 3D MPRAGE sequence. The pre-processing steps for the anatomical data involved the removal of nonbrain tissue from T1 anatomical images using the Brain Extraction Tool.

The Parkinson's Progression Markers Initiative (PPMI) is a landmark observational clinical study to comprehensively evaluate cohorts of significant interest using advanced imaging, biologic sampling and clinical and behavioral assessments to identify biomarkers of Parkinson's disease progression.

PPMI is taking place at clinical sites in the United States, Europe, Israel, and Australia. Data and samples acquired from study participants has helped in the development of a comprehensive Parkinson's database and biorepository, which is currently available to the scientific community to conduct field-changing research.

T1w MRI scans

T1w scans are T1 weighted sequences or images of the brain. They are also called "spin-lattice" relaxation time. It is one of the basic pulse sequences in MRI and relies upon the longitudinal relaxation of a tissue's net magnetization vector (NMV). The T1w sequence demonstrates the differences in the T1 relaxation times of tissues. T1 weighted sequences have short echo time

(TE) and repetition time (TR). In T1-weighted images, CSF and fluid have a low signal intensity than tissue and thus appear dark. Gray matter is darker than white matter here.

The preprocessed MRI 3D data in the PPMI dataset was both in NIFTI and DICOM format. **DICOM** is used to exchange and transmit medical images in standardized format, enabling the integration of medical imaging devices from multiple manufacturers. **NIfTI** is a new Analyze-style data format which facilitate inter-operation of functional MRI data analysis software packages. However, for this study, the data in the NIFTI format was extracted from the Parkinson's database and biorepository. These images were concatenated across z and t axes and then converted to a stack of 2D images where each 2D image represented each slice in PNG format. For the conversion we used the visualization tool MRIcro, Neuroimaging package Nibabel (http://nipy.org/nibabel/) and Python OpenCV (opencv.org). Next, images were labeled for binary classification of Parkinson's disease vs Normal data in order to be fed into Deep Learning platform.

The images were downloaded from the PPMI collection in NIFTI format and visualized using the MRIcro tool(http://people.cas.sc.edu/rorden/mricro/) and also the Neuroimaging package Nibabel (http://nipy.org/nibabel/) and Python OpenCV (opencv.org). Also, the patients' age, visit, sex, data acquiring data was also captured from the clinical data dump of PPMI.

The images were then aligned to the individual's high-resolution T1-weighted scans. The product of preprocessing step was 256x54 slices in which the last 5-10 slices of each image were removed as they contained no functional information. Finally, it was 6132 number of images for 30 PD affected subjects and 4416 number of images for 24 healthy control subjects.

MRIcro TOOL

MRIcro is an image viewing tool for medical images written by Chris Rorden. It allows Windows and Linux computers view medical images. It is a standalone program along with tools which functions like a software that allows neuroimagers to analyze MRI, fMRI and PET images. MRIcro allows efficient viewing and exporting of brain images. In addition, it allows neuropsychologists to identify regions of interest (ROIs, e.g. lesions). It can create Analyze format headers for exporting brain images to other platforms such as DICOM, .jpg and .png formats.

MRIcro also has the feature to draw three-dimensional regions of interest (ROIs). This is useful for illustrating disease affected regions of the brain that have significant damage. ROIs from different individuals can be overlapped on brain images that have been normalized to the same template and henceforth allows neuropsychologists to assess common areas of damage.

All ROIs are drawn on top of the MRI image, rather than directly on it enabling the brain images to be viewed with or without corresponding ROIs. MRIcro provides a number of tools for creating and viewing ROIs and are displayed as a series of buttons in the 'Region of interest' panel.

Converting medical images to Analyze format

When we convert medical images to Analyze format, you can select a series of 2D images that will be stacked and saved as a single unified 3D Analyze format file. Selecting 'Convert foreign to Analyze' from the 'Import' menu will create a new window that allows us to describe the images.

For the conversion, we went to the File menu of the MRIcro tool and chose the “Open Image [Analyze or VoxBo]” option to browse through the computer and open the appropriate nifti image. Then we used the “save as” command from the File menu to save it in the .png format. Each image

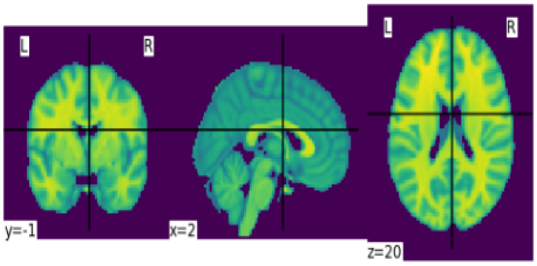
had 256 slices and all the slices were converted into individual 2D images in .png format. All the slices of the nifty format images were also combined and layered together to form a single 2D image for every nifty image. However, for the purpose deep learning and model building, individual slices were used. The png file that is created is 'anonymized' -and will contain no private details from the Analyze header (e.g. patient name, scan date, etc). The converted images were saved into two different folders, one containing the PD affected neural images and the other containing the control healthy brain images.

MRIcro can also convert Analyze format images to the DICOM format. The command 'Save as...' allows us to save and convert an image to DICOM. The *.img file that is created when we press 'Save DICOM' will be in the DICOM format. The DICOM file will be either 8-bit or 16-bit.

DATA VISUALIZATION

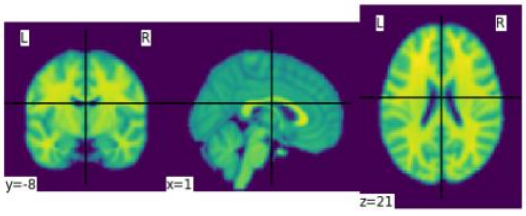
We have used the Neuroimaging package Nibabel (http://nipy.org/nibabel/) and Python OpenCV (opencv.org) to visualize the MRI scan images in the Nifti format. We have also used package "nilearn" to plot the nifti images. Screenshot of a sample PD affected neural image is shown below.

```
In [52]: MNI152_FILE_PATH = 'C:/Users/roshn/Desktop/PPMI_3604_MR_SAG_T1_3D_MPRAGE_.nii'
         from nilearn.datasets import MNI152_FILE_PATH

In [53]: print('Path to MNI152 template: %r' % MNI152_FILE_PATH)

         Path to MNI152 template: 'C:\\Users\\roshn\\Miniconda3\\envs\\tensorflow\\lib\\site-packages\\nilearn\\datasets\\data\\avg1
         52T1_brain.nii.gz'

In [54]: plotting.plot_img(MNI152_FILE_PATH)

Out[54]: <nilearn.plotting.displays.OrthoSlicer at 0x159a8292cf8>
```



Using the "image.smooth_img" function, we can smoothen the image. This is just for visualizing the image. However, it was not used for our work.

```
In [56]: plotting.plot_img(smooth_anat_img)

Out[56]: <nilearn.plotting.displays.OrthoSlicer at 0x159a519a4e0>
```



Using the “image.smooth_img” function, we can smoothen the image and also print the dimensions of the 3D Nifti image. Screenshot of the same has been captured below.

```
In [55]: from nilearn import image
         smooth_anat_img = image.smooth_img(MNI152_FILE_PATH, fwhm=3)

         # While we are giving a file name as input, the function returns
         # an in-memory object:
         print(smooth_anat_img)

<class 'nibabel.nifti1.Nifti1Image'>
data shape (91, 109, 91)
affine:
[[ -2.    0.    0.   90.]
 [  0.    2.    0. -126.]
 [  0.    0.    2.  -72.]
 [  0.    0.    0.    1.]]
metadata:
<class 'nibabel.nifti1.Nifti1Header'> object, endian='<'
sizeof_hdr      : 348
data_type       : b''
db_name         : b''
extents         : 0
session_error   : 0
regular         : b'r'
```

MRIcro VISUALIZATION

Below is a snapshot of visualization of the middle slice of a PD affected Nifti image in MRIcro.

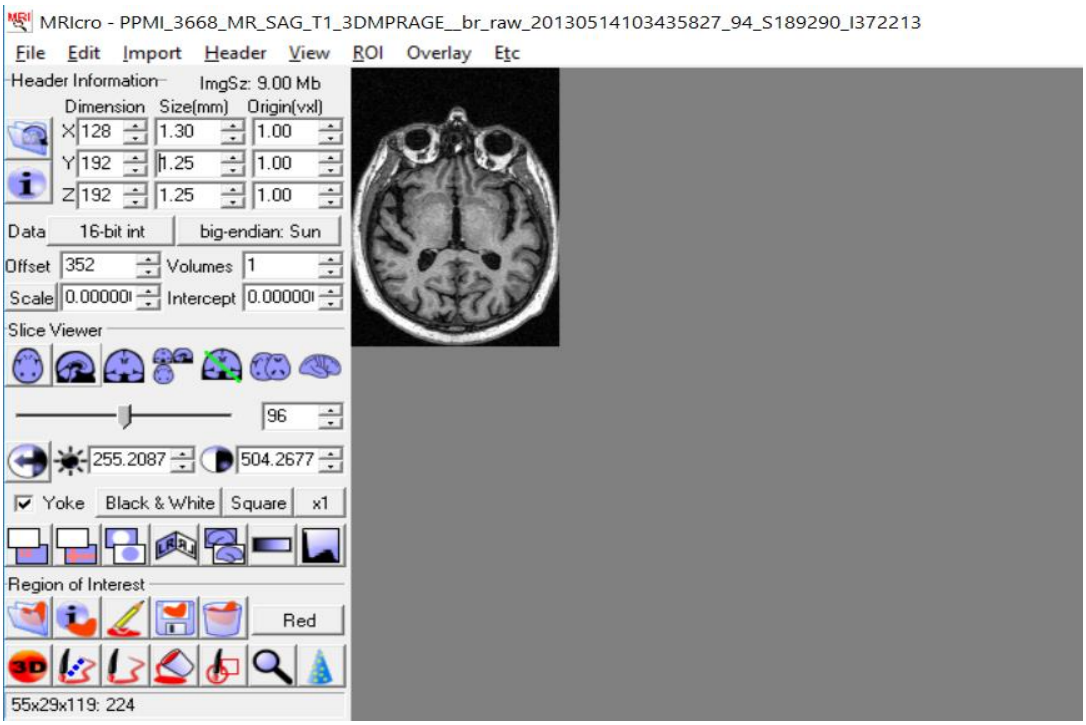


Fig 1. MR_SAG_T1_3DMPRAGE of a PD affected MRI scan

Deep Learning

Deep learning is a machine learning technique, developed based on complicated algorithms that learns high-level features and extract those abstractions directly from the data by using complicated neural network architecture. It is a modern branch of machine learning inspired by human brain.

Deep learning is a method that helps in reducing the dimensionality of the data or in feature reduction. It is usually used in problems associated with analog input data such as images of pixel data, text documents or files of audio or video data. It is called 'deep' because it learns the features at multiple levels of abstraction as part of the training in order to map the input directly to the output without depending on human-crafted features. Deep learning aims at learning feature hierarchies with features from higher levels formed by the composition of lower level features.

Convolution Neural network

Convolutional Neural Network (CNN), is a highly effective machine learning algorithm potentially used in a variety of applications such as handwritten digit recognition, visual recognition, and image classification. It is a special kind of multi-layer neural network which extracts visual patterns from pixel images with minimal preprocessing. The architecture of CNN has been designed in a way such that it utilizes spatial relationships to encode certain properties and reduce the number of hyper parameters and thus improves general feed-forward back propagation training. CNN models combine weights into smaller kernel filters to simplify the learning model.

The CNN network consists of various combinations of the convolution layers, max pooling layers, Normalization layers, fully connected layers or dense layers.

Convolution Layer

In CNN, small portions of the input image are treated as the lowest layer of the hierarchical structure. The first layer is the convolution layer which is made up of neurons having learnable weights and biases or they can be termed as a set of learnable filters. The **CONV layer** is the core building block in this network which takes the input image and creates feature maps from it. A filter is just a matrix of values, called weights, that are trained to detect specific features. Every filter is spatially small in size(also called the window size or the kernel size) but spans through the full input volume. This layer computes a dot product between the weights of the neurons and the local region or patch of the input volume that they are connected to. Each filter is then convolved across the width and height of the input volume with overlapping regions to produce a 2D activation map of the filter. Each and every unit in the output volume can be interpreted as an output of a neuron that looks into a small window in the input and shares parameters with neurons in the same activation map. These activation maps are then stacked for all fitters along the depth to form the full output volume.

Even small patches of the input image volume contain thousands of pixels resulting in large number of connection weight parameters to be trained. The weights are then shared across the entire input space and the ones that belong to the same layer shares the same weights. This weight sharing helps to reduce the total number of trainable parameters eventually creating a more effective model. The inputs can be original input volumes or information coming from other neurons that are fed in the successive CONV layers.

Max Pooling Layer

A pooling layer is usually inserted in between successive Conv layers. The function of the Pooling layer is to reduce or down sample the spatial size of the representation. This is done to reduce the huge amount of network hyper parameters and to control overfitting. The Pooling layer operates independently on every depth slice of the input and resizes it spatially, using the MAX Pooling operation. A max pooling function partitions the input data coming from the immediate predecessor conv layer into a set of overlapping windows and returns the maximum value for each subregion. In the pooling layer, the neighboring elements in the convolution layer output matrices are combined with a kernel size of say 2*2 (as I have used in the model). This combination of 4 neighboring elements are then replaced with the maximum of the input matrix to generate one element in the output matrix. During error back propagation process, the gradient signal is supposed to be routed back to only those neurons which contribute to the pooling output.

Normalization Layer

The layer within a series of Conv and max pooling layers is the **Normalization Layer or the Activation Layer.** This layer is used at the input for feature scaling, and in batch normalization at hidden layers. This layer scales the input so that the output has near to a zero mean and unit standard deviation, for efficient training. This layer applies an elementwise activation function, such as max (0,x) thresholding at 0. All the unique weights of the neurons are multiplied by the input and the total of them is then simulated through the activation function to check it is more than the threshold or less than that. If it is more than the threshold then it returns a 1 else 0. This happens for the rectified linear unit (ReLu) activation function. For the sigmoid activation function,

the function returns a value in between 0 and 1 (a fraction) after the comparison with the threshold. This layer does not change the size of the image.

Activation Function – ReLu

ReLu is rectified linear unit which is a very simple function for generating good experimentation results. A ReLu is simple defined as *f(max) = max(0, x)*, and the non-linear function is represented in the following graph. The function is 0 for negative values and it grows linearly for positive values.

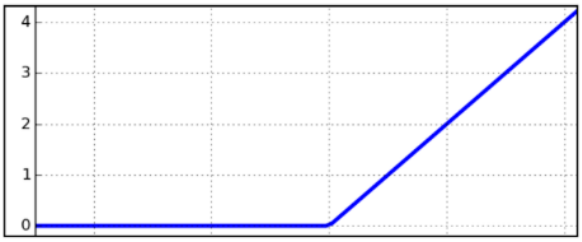


Fig 2. The ReLu Activation Function

Activation Function - Sigmoid

The sigmoid function is defined as follows:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

As represented by the following graph, it has small output changes in (0,1) when the input varies in

(-∞,∞). Mathematically the function is continuous. A typical sigmoid function is represented by the following graph:

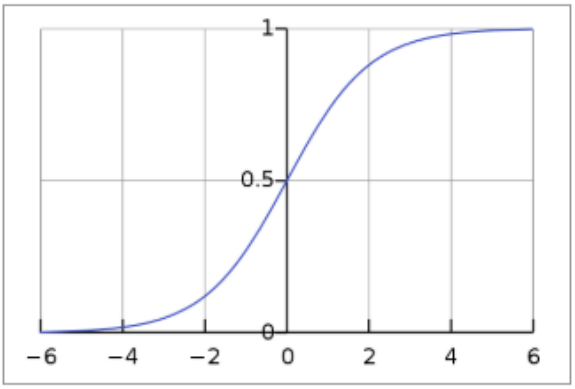


Fig 3. The Sigmoid Activation Function

The **SoftMax activation** function is usually used for probability distribution when there are multiple classes.

Flatten

Flatten is the layer just after the series of consecutive convolution and pooling layers. Convolution layers output 3-dimensional activation maps such that just the output is needed as to find if an image belongs to a particular class. This layer converts the 3D feature maps to 1D feature vectors.

Fully Connected Layer/Dense Layer

The final layer of the CNN network is the **Fully Connected Layer** which is a typical neural network in which every input is connected to every output by a learnable weight. The computation chain of a CNN ends in a fully connected network that integrates information across all locations in all the feature maps of the layers below. Each neuron in this layer will be connected to all the numbers in the previous volume. This layer is also called the Dense Layer and in here, every input is connected to every output by weight followed by a non-linear activation function. We have used the sigmoid activation function at this layer. This final fully connected layer typically has the same

number of output nodes as the number of classes. It computes the class scores, resulting in volume of number of classes.

Due to the weight sharing feature and the use of a subset of the weights of a dense layer, there are less weights in the dense layer. Due to the replication of weights in a CNN, a feature may be detected across the input data. If an image is shifted, the neuron detecting that feature is shifted in the same manner and amount. The complex architecture provides a level of invariance to shift, scale and rotation as the local receptive field allows the neuron access to elementary features such as oriented edges or corners. Below is an image of a Convolution Neural Network.

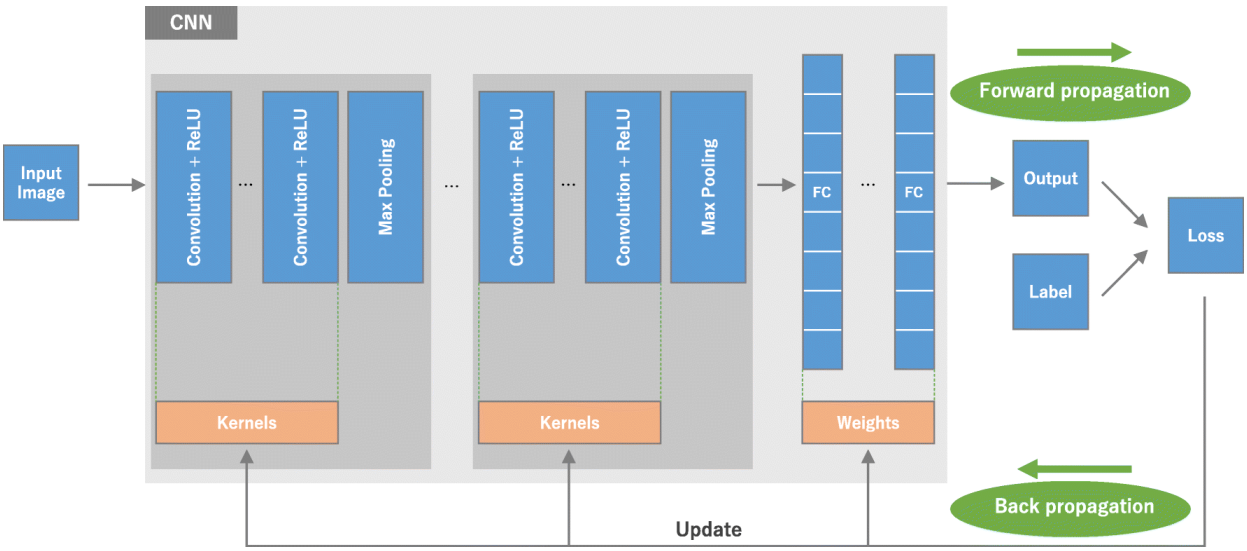


Fig.4 The Convolution Neural Network Architecture

Drop-out Algorithm

Neurons are randomly disabled in each layer of the CNN network. This is called the Drop-out algorithm which is applied while training the model in order to improve its performance. A drop-out map is initialized in each layer with the same size of the neurons in that layer. This is done to

mark the on and off state of the corresponding neuron at the beginning of each training iteration. During the training iteration, the neurons with off state are then removed from the network by disabling the activation signal forward propagation and error signal backward propagation of the neuron. During testing, all the neurons are in the turned-on state, however the activation signal is attenuated to the probability of average turn on rate during the training phase. The drop-out algorithm is applied to improve the model accuracy.

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools which makes the classification of images easy and efficient. Keras is utilized to produce deep models on smartphones (iOS and Android), on the web, on the Java Virtual Machine or on clusters of Graphic Processing Units. In this work, the Keras package was installed in anaconda Jupyter notebook for implementing Convolution Neural network models in Python. From the Keras package, we have imported the convolution neural network layers such as the Convolution layer, the Pooling layer,

Tensorflow

Tensorflow is an end-to-end open source platform for machine learning. It is an interface for expressing machine learning algorithms, and a platform for executing such algorithms. Tenserflow has a comprehensive, flexible ecosystem of tools, libraries and community resources that allows developers build and deploy Machine learning powered applications. We have installed Tensorflow in anaconda to support the Tensorflow environment in Jupyter using python.

Implementation

The packages that have been used in Python are:

- Nibabel, nilearn, cv2
- Pylab
- Matplotlib
- Keras
- Numpy, os,

After converting the PD affected patients' and control subjects' images from the nifty format to the .png format, we used a tenserflow background in Python to read the image files. The PD affected neural images were labelled as '0' whereas the control subjects' neural images were labelled as '1'. In total, we had 10,548 slices of images for 54 patients (30 PD and 24 control).

```
[104]: data=[]
labels=[]
PD_path =os.listdir("C:/Users/roshn/Desktop/Just/Just PD/")
for pars in PD_path:
    try:
        image=cv2.imread("C:/Users/roshn/Desktop/Just/Just PD/"+ pars)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((64, 64))
        #image_from_array = image_from_array.convert('L')
        data.append(np.array(size_image))
        labels.append(0)
    except Exception as e:
        print(e)

[105]: control_path=os.listdir("C:/Users/roshn/Desktop/Just/Just Control/")
for unef in control_path:
    try:
        image=cv2.imread("C:/Users/roshn/Desktop/Just/Just Control/"+ unef)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((64, 64))
        data.append(np.array(size_image))
        labels.append(1)
    except Exception as e:
        print(e)

[106]: print('Lenght of Data : ' + str(len(data)))
print('Lenght of Data : ' + str(len(labels)))

Lenght of Data : 10548
Lenght of Data : 10548
```

We divided the whole dataset of 10,548 images into 90% of training data containing 9493 images and 10% of test data containing 1055 images. 20% of the training dataset was used for validation during the learning or training phase of the model.

```
In [119]: print('Lenght of Data : ' + str(len(training_X)))
          print('Lenght of Data : ' + str(len(test_X)))

          Lenght of Data : 9493
          Lenght of Data : 1055
```

One hot Encoding

One hot encoding is a process by which categorical variables or class vector (integers) are converted into binary class matrix so that it can be provided to a Machine Learning algorithm to do a better job in prediction. The function "np_utils.to_categorical" from the Keras package was used for this purpose.

Baseline Model

Then we trained the model on the training data. The input data has the shape of 64*64*3. For the baseline model, we used three layers of the 2D convolution layer. The number of neurons or units allocated is usually 64 for the 1st conv layer, 32 for the 2nd conv layer and 16 for the 3rd conv layer. The kernel size or the window size or the filter that we selected for all the convolution layers is 3 * 3 which sweeps across the whole input layer with overlapping to create the feature map. The operations performed by this layer are linear matrix multiplications, but they go through an activation function at the output, which is usually a non-linear operation. We have used the ReLu activation function for all the conv layers. Each conv layer is then simulated through a max pooling layer having a total of 2-3 max pooling layers in the CNN network. The kernel size for the pooling layer is 2*2 which means it will output the maximum of the four neighboring elements from the input matrix. Then the next layer of the model is the flatten layer which converts the 3D matrix from the previous layers into a 1D matrix of vectors. The baseline model that we have used has 2 dense or fully connected layers. The first dense layer has 128 neurons or units and a ReLu activation function. The final and the second dense layer has 2 neurons or units each corresponds to each of the two output classes followed by a sigmoid activation function. Different models were

trained on the data and tested on the validation set of data for every epoch. The batch size was selected as 32 or 64 and number of epochs were 30.

Drop-out Algorithm

The Drop-out Algorithm refers to dropping out units (hidden and visible) in a neural network along with all its incoming and outgoing connections. At the beginning of each iteration, the neurons are marked on and off state. During the training iteration, the neurons with off state are then removed from the network. This algorithm is applied to improve the model accuracy and reduce overfitting. In this work, we saw that after applying this algorithm the performance of the model increased.

LeNet-5 Model

In this work, we dealt with a binary but very complicated binary classification of Parkinson's and Normal data. This led us to choose LeNet-5 architecture and adjust the architecture for MRI data. This is a special type of CNN architecture in which 2 layers of 2D convolution layers are used for training the model. The first layer contains 32 neurons or units and the second contains 64. The filter or the window size used in here is 3*3. The ReLu activation function has been used for all the two conv layers. Each conv 2D layer is then simulated through a max pooling layer having a total of 2 max pooling layers in the CNN network. The kernel size for the pooling layer is 2*2. It is followed by the Dropout layer (20%) and then a flatten layer which converts the 3D matrix from the previous layers into a 1D matrix. The LeNet-5 model has 2 dense or fully connected layers. The first dense layer has 128 neurons or units and a ReLu activation function. The second dense layer has 2 neurons or units followed by a sigmoid activation function. Applying this model, we achieved an accuracy of 96.99%.

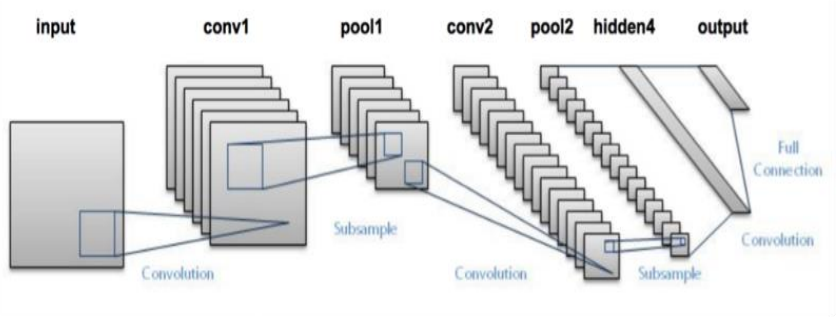


Fig.5 CNN Architecture of a LeNet-5 Model

Batch Normalization

Batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation to increase the stability of a neural network. It allows each layer of a network to learn by itself a little bit more independently of other layers. It reduces overfitting because it has slight regularization effects. We have used batch normalization to make the performance of the model better.

Model Compilation

The number of epochs was set to 30 and the batch size was usually 32 and sometimes 64. The optimizer used for the models is "Adam". The loss option was specified as 'Binary Cross-Entropy' because this is a binary classification problem. Adam is an optimization algorithm which is an extension to the stochastic gradient descent procedure to update network weights iterative based in training data. Adam is a recommended choice for binary image classification problem due to its bias-correction feature. A neural network does not attempt to maximize the accuracy of the model, rather it attempts to minimize the error every time it fits a model. We have applied the Binary Cross-Entropy Loss because it sets up a binary classification problem between $C'=2C'=2$ classes for every class in CC. The Binary Cross-Entropy Loss is a Sigmoid activation plus a Cross-

Entropy loss which is independent for each class. In other words, the loss computed for every CNN output vector component is not affected by other component values. The metric that we have specified to be viewed is 'accuracy'. However, we have also measured the loss function which gives the degree of error.

```
In [55]: model.compile(loss='binary_crossentropy',
                       optimizer='adam',
                       metrics=['accuracy'])
```

Results & Performance Metrics of the Models

In the training phase of the Baseline model, the loss and accuracy of the training and the validation data were measured. Then the models were tested using the test dataset. The figure below shows the comparison of the test accuracies of different models.

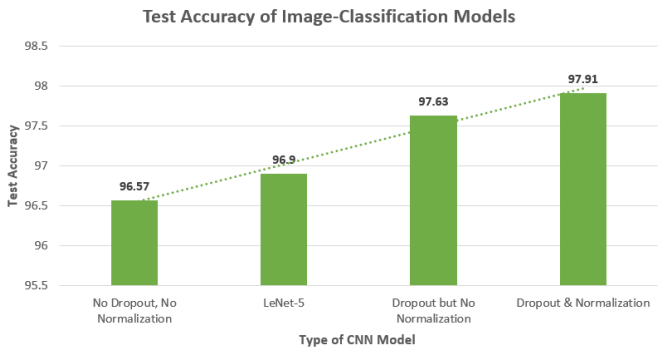


Fig.6 Comparison of different Models Test Metrics

The graph of the both accuracy and the loss function and the confusion matrix of the 2 best selected models are presented below.

Performance of the Model Without Batch Normalization

The training accuracy attained after the 30th epoch was **96.65%** and the test accuracy attained was **97.63%** for this model. The loss attained was **.07%.** Below are the plots for the model accuracy and the model loss.

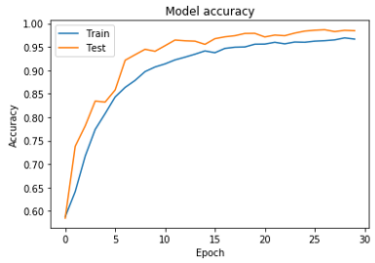


Fig.7 No Batch Normalization Model Accuracy of Train and Validation Set

We have used the **loss function** to measure how accurate our network was in identifying the PD symptoms from the input image. The plot of the model loss is given below:

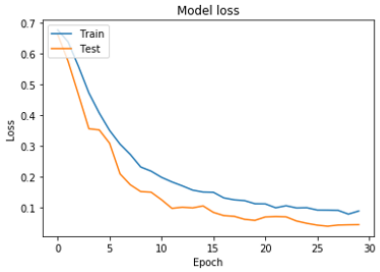
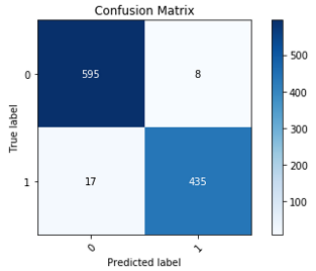


Fig.8 No Batch Normalization Model Loss of Train and Validation Set



	precision	recall	f1-score	support
0	0.97	0.99	0.98	603
1	0.98	0.96	0.97	452
micro avg	0.98	0.98	0.98	1055
macro avg	0.98	0.97	0.98	1055
weighted avg	0.98	0.98	0.98	1055

Fig.9 No Batch Normalization Model Confusion Matrix

Performance of the Model with Batch Normalization

The training accuracy attained after the 30th epoch was **95.43%** and the test accuracy attained was **97.91%** for this model. The loss attained was **.05%.** Below are the plots for the model accuracy and the model loss.

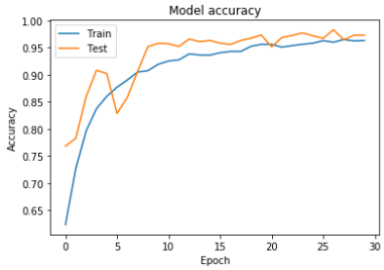


Fig.10 Batch Normalization Model Accuracy of Train and Validation Set

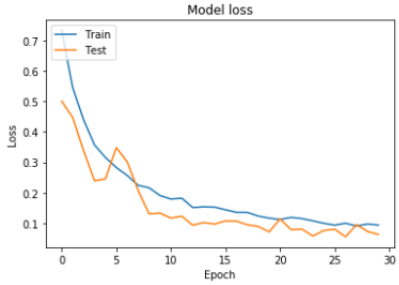
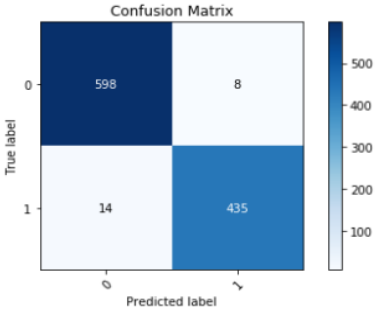


Fig.11 Batch Normalization Model Loss of Train and Validation Set



	precision	recall	f1-score	support
0	0.98	0.99	0.98	606
1	0.98	0.97	0.98	449
micro avg	0.98	0.98	0.98	1055
macro avg	0.98	0.98	0.98	1055
weighted avg	0.98	0.98	0.98	1055

Fig.12 Batch Normalization Model Confusion Matrix

Changes made to avoid contamination

In the above work, the input slices that were used for the training and the test dataset were from the same patients' MRI scan images. In this endeavor, we kept isolated the MRI scans 2D slices of 4 patients to be used for testing dataset. Among these four patients, two patients are affected by PD and the other two are healthy non-PD patients. Hence, we now have 9905 slices belonging to 50 patients (both PD and non-PD) to be used as the training and the validation dataset. We have 643 slices belonging to 4 patients (2 PD and 2 non-PD) to be used as the testing dataset.

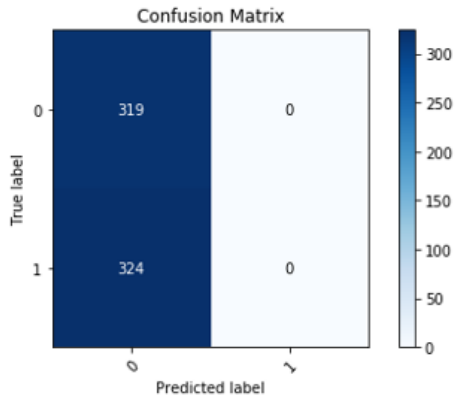
We applied our best two models as discussed in the earlier sections to this new dataset. The results are as below:

Performance of the Model with Batch Normalization

Training Accuracy: 96.59%

Test Accuracy: 49.6%

Confusion Matrix:

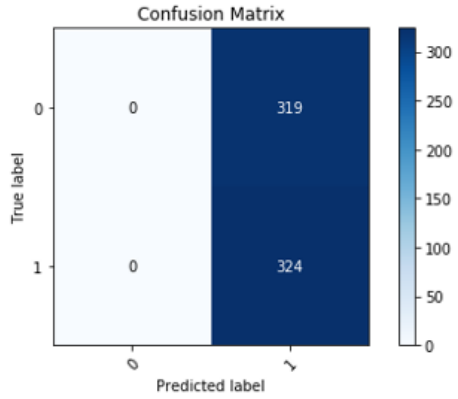


Performance of the Model Without Batch Normalization

Training Accuracy: 97.50%

Test Accuracy: 50.38%

Confusion Matrix:



From the above two model performances, we observed how the models have been overfitting before as for most of the patients' MRI scans, some of the image slices were used as the training dataset and some of them were used as the test dataset but both the sets belonged to the same patient. This phenomenon is called contamination of images. To avoid this phenomenon, we isolated 4 patients' all MRI scan slices as test dataset and the other 50 patients' data was used as the training dataset.

In light of the above discussion, it will be the best endeavor to increase the number of inputs images and optimize the model to increase the accuracy and reduce overfitting.

CONCLUSION AND FUTURE SCOPE

In this paper, we successfully classified the Parkinson's Disease data from the normal control with 97.91% accuracy using the CNN deep learning architecture with batch normalization and dropout algorithm. This model was trained on a large number of images. It is also possible to implement this method to predict different stages of the Parkinson's disease for different age groups and to study PD-related cognitive decline and dementia. We have applied different combinations of CNN models with parameter optimization and finally the best 2 models were with accuracies 97.63% without batch normalization and 97.91% with batch normalization. We can endeavor to optimize the model that has been built, to decrease the bias and the overfitting problem by tweaking the kernel sizes, the number of layers and the number of units or neurons in each layer and using the dropout algorithm. This deep learning-based solution opens future scope for medical image analysis and enables researchers and physicians to do feature selection and classification in order to potentially predict any new data.

REFERENCES

1. Medical Image Classification with Convolution Neural Network
   https://ieeexplore.ieee.org/abstract/document/7064414

2. A Bayesian network decision model for supporting the diagnosis of dementia, Alzheimer's disease and mild cognitive impairment
   https://www.sciencedirect.com/science/article/pii/S0010482514000961

3. Classification of Alzheimer's Disease Using fMRI Data and Deep Learning Convolution Neural Networks
   https://arxiv.org/pdf/1603.08631.pdf

4. Predicting dementia development in Parkinson's disease using Bayesian network classifiers
   https://www.sciencedirect.com/science/article/pii/S0925492712001254

5. A Study on the Image Detection Using Convolution Neural Networks and Tenser Flow
   https://ieeexplore.ieee.org/abstract/document/8597204

6. Convolutional Neural Networks from the ground up
   https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1

7. www.github.com

8. Large-scale Video Classification with Convolutional Neural Networks

   https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.pdf

9. https://nilearn.github.io/user_guide.html

10. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, SoftMax Loss, Logistic Loss, Focal Loss and all those confusing names

    https://gombru.github.io/2018/05/23/cross_entropy_loss/

11. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

12. CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more

https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

13. https://pythonprogramming.net/convolutional-neural-network-deep-learning-python-tensorflow-keras/

14. Convolutional neural networks: an overview and application in radiology

https://link.springer.com/article/10.1007/s13244-018-0639-9

15. https://nilearn.github.io/auto_examples/plot_nilearn_101.html

16. [Book] Deep Learning with Keras by Antonio Gulli, Sujit Pal

https://books.google.com/books?hl=en&lr=&id=20EwDwAAQBAJ&oi=fnd&pg=PP1&dq=theory+of+tensorflow+and+keras&ots=lHdz8icSW5&sig=yipkOX33pyXOLW0f6-7-tKxz_sc#v=onepage&q&f=false

17. https://www.youtube.com/watch?v=2-Ol7ZB0MmU

18. https://machinelearningmastery.com/what-is-deep-learning/

19. Deep Convolutional Neural Networks for Hyperspectral Image Classification

https://www.hindawi.com/journals/js/2015/258619/