Majlesi Journal of Electrical Engineering
Vol. 18, No. 1, March 2024

Parkinson's Disease Classification based on Enhanced Ensemble Learning and Brain MRI

Anitha Rani Palakayala[1], Kuppusamy P[1]
1- School of Computer Science and Engineering, VIT-AP University, Guntur, AP, India.
Email: anitha.palakayala@gmail.com (Corresponding author), drpkscse@gmail.com

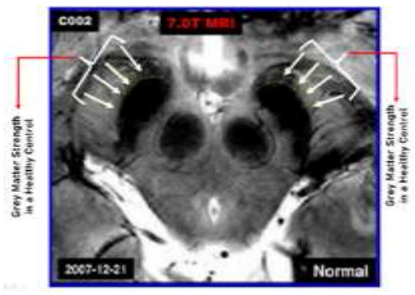
ABSTRACT:
Parkinson's Disease (PD) is a neurological disorder that causes progressive loss of brain cells. Despite the fact that there is no known cure for this neurodegenerative disease at present, early diagnosis and treatment may improve the quality of life. Magnetic Resonance Imaging (MRI) detects structural changes related to dopamine deficiency in PD. To categorize MRI scans as Healthy Control (HC) or PD, this study proposes an ensemble of Deep Convolution Neural Network (DCNN) models. Initially, we have used DCNN models using augmentation and transfer learning, to classify as PD or HC. In the next stage, we applied a classifier fusion ensemble approach to enhance the overall result of the classification model. The proposed model is trained using the data collected from PPMI database, while assessed on custom dataset which is created using the data collected from Lalitha Super Specialty Hospital (LSSH). Finally, it was observed that the developed ensemble model produced an outstanding performance by plotting an overall accuracy of 99%, while the transfer learned EfficientNet B1 DCNN model stood in the second position, achieving a remarkable accuracy of 98%. This study serves as a significant step forward, providing valuable insights for researchers and clinicians engaged in the domain of clinical image assessment using deep learning techniques.

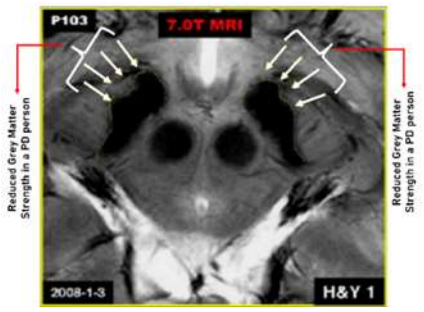
KEYWORDS: Deep Learning, Ensemble Learning, Neurodegeneration, Parkinson's Disease, Transfer Learning.

1. INTRODUCTION

Parkinson's disease (PD) is a progressive neurodegenerative disorder, characterized by the preliminary harm caused in the substantia nigra region of the brain. It occurs when the dopamine-generating neurons become damaged or die, because of various factors such as genetics, environmental factors and lifestyle choices [1]. People with PD begin to experience tremors, and difficulty in speaking, writing, walking, or completing other simple tasks. There are several other symptoms like sweating, soreness, drooling of saliva, pill rolling, anxiety, hysteria and sleep interference. The PD diagnosis can be made using neuroimaging techniques like Magnetic Resonance Imaging (MRI), Single-Photon Emission Computed Tomography (SPECT), and Positron Emission Tomography (PET). These techniques identify the structural and functional alterations that occur in the brain, as the disease progresses. Substantial reduction in grey matter strength is quite common in people with PD, compared to those of Healthy Controls (HC) and it is evident from the bilateral nigral hyperintensity in the substantia nigra (directional arrows) regions of Fig. 1, 2. These methods are able to discover the problem only when 80% of the neurons have degenerated and the person exhibits peculiar indications like tremor, finger tapping, pill rolling, hallucinations, drooling of saliva, posture bent, anxiety, difficulty in speaking, writing, and walking. A dependable, economical, and congenial Computer-Aided Diagnostic (CAD) method is necessary to identify PD and observe its advancement using MRI, which could be valuable for medical professionals. Deep learning algorithms produce contemporary results when several tasks related to computer vision are implemented [5]. Even though DCNNs require a significant portion of data for training the network and powerful GPUs to speed up the learning process, they demonstrated remarkable outcomes in a wide scope of diagnostic imaging applications.

Paper type: Research paper
https://doi.org/10.30486/mjee.2024.2006443.1360
Received: 28 November 2023; revised: 24 December 2023; accepted: 5 February 2024; published: 1 March 2024
How to cite this paper: A. R.Palakayala, Kuppusamy P, **"Parkinson's Disease Classification based on Enhanced Ensemble Learning and Brain MRI"**, *Majlesi Journal of Electrical Engineering*, Vol. 18, No. 1, pp. 283-309, 2024.


C002
7.0T MRI
Grey Matter Strength in a Healthy Control
Grey Matter Strength in a Healthy Control
2007-12-21
Normal
Fig. 1. MRI of H.C [52].


P103
7.0T MRI
Reduced Grey Matter Strength in a PD person
Reduced Grey Matter Strength in a PD person
2008-1-3
H&Y 1
Fig. 2. MRI of PD [52]

Ensemble learning [58, 59] combines the predictions from multiple models, maximizing the accuracy and robustness by leveraging their unique strengths. It reduces errors, improves generalization, enhances stability, and boosts prediction effectiveness. Therefore, in this work, an image-based classification system using DCNN, transfer learning and ensemble approach for detecting PD and HC is developed on a GPU and tested on a custom data set created from the data obtained from LSSH. The study has several compelling advantages, as follows:

1. The early detection of PD can be accomplished through image-based computer vision techniques, eliminating the need for extensive clinical tests. This approach offers the advantages of reducing the associated costs and saving valuable time in the diagnostic process.
2. Facilitating real-time clinical comparisons in the PD diagnosis process by incorporating a cutting-edge real-time custom dataset created from the data obtained from LSSH, for the testing phase.
3. A meticulous comparison of the proposed approach conducted with state-of-the-art methodologies positions this study at the forefront in the field of research.

1.1. Motivation

The quality of life of a PD person is negatively impacted by both the movement and non-movement indications connected to the process of aging [2, 62]. At present, there are no established blood or laboratory tests that can detect PD and its advancement. In the initial stages of PD, it is essential to take the appropriate preventative measures, which can halt or slow down the progression of the disease. The doctors look at a patient's medical history and perform a cognitive evaluation using Unified Parkinson's Disease Rating Scale (UPDRS) or Hoehn and Yahr (H&Y) scale, for initial diagnosis [3]. Skilled therapists play a vital role in the effectiveness of therapeutic tests, but their decisions may be subjective. Obtaining all the required information is a lengthy process that requires the involvement of many individuals.

1.2. Contributions

The major contributions made by this research for improving prediction accuracy are as follows:

- Data augmentation technique helps to overcome the problem of imbalanced class distribution in the dataset. This approach is superior to the restricted availability of MRIs of PD patients and a noticeable improvement in overall performance can be observed.
- The proposed DCNN model, based on whole-brain analysis, utilizes spatial structures autonomously, avoiding the need for hand-crafted features, thus making the design as
- A remarkable increase in performance can be observed when ensemble learning with classifier fusion is applied on the trained DCNN models, compared to individual models.
- Accuracy enhancement can be observed when the preprocessing operations are applied on images, followed by

DCNN model training, rather than training the raw images directly.

- Implementing efficient DCNN models, namely EfficientNet B1, ResNet-50, ResNet152V2 and MobileNetV2 using transfer learning and conducting a thorough investigation and comparative analysis to analyze the efficiency of the proposed study, compared to the state-of-the-art works.
- Facilitating real-time clinical comparisons by incorporating a real-time custom dataset created from the data obtained from LSSH, for the testing phase and also by interpreting the developed model.
- Providing a comprehensive perspective on classification and analysis applications within the domain of PD diagnosis, intended to benefit PD diagnosis researchers and developers.

2. RELATED WORKS

An early-stage diagnosis approach for PD was suggested by [6], and it involved employing a Joint Function Sample Selection (JFSS) technique to select the most beneficial features to develop a model. For evaluation purposes, fabricated and openly available MRI brain scan files of PD are utilized in order to display a high precision ranking. [7] designed a therapeutic assistance program that makes use of an Artificial Neural Network (ANN). Measurements of DCNN gathered from brain detriments were used to estimate PD, for the purpose of this investigation. DCNN le-Net with transfer learning was used by Sivaranjini et al., [8] to classify MRI Brain scans of persons with PD and HC using PPMI dataset. An accuracy of 88.9% was achieved with this method. Lei et al. [9] came up with the idea of a sparse feature selection model and reported that it had an accuracy of about 80%. To identify distinguishing characteristics, Salvatore et al. [10] looked at MRIs of people with healthy brains, people with PD and people supranuclear palsy. Next, they conducted a Principal Components Analysis (PCA) to identify the pertinent characteristics, and then they fed those features into a SVM classifier for data classification. Experiments to classify PD were carried out by Brahim et al. [11] employing characteristics based on structure and superficial connections, as well as an SVM classifier. An accuracy of 92.6 was achieved from their work. Transfer learning concept and the InceptionV3 model were incorporated by Quan et al. [12], in their experiment of identifying PD. They obtained 98% score of accuracy. From the database of PPMI, Arman et al. [13] selected 64 PD cases, with 38 males and 26 females making up the total of 64. The non-imaging and imaging factors that were utilised to predict motor results were combined using Random Forest (RF) analysis, comprised of 5000 trees. By utilizing this technology, a more accurate early prediction of PD was made possible. The inclusion of radiomic characteristics to traditional measures resulted in a substantial improvement in the prediction of outcome, which in turn resulted in a reduction in the absolute error of predicting MDS-UPDRS-III. The connection of Single Nucleotide Polymorphisms (SNP) with imaging traits increased the statistical power of the study and allowed for the examination of a wider variety for the study that was published by Choi et al. [14]. A DCNN framework along with PD Net model was used of SNP. Data were gathered from PPMI and Seoul National University Hospital (SNUH) cohort, on 431 people diagnosed with PD, 193 HC and 77 patients diagnosed with Scans Without Evidence of Dopaminergic Deficit (SWEDD). Computerized quantification of the DAT Binding Ratio (BR) was carried out as a usual approach for perceptible analysis on the SPECT data. Prashanth et al. [15] used data from the PPMI database. All of the individuals with PD are considered to be in the early stage (Hoehn and Yahr (H&Y) stage 1 or 2), and all of the SWEDD subjects, who are newly identified PD patients based on clinical symptoms but have normal dopaminergic imaging, exhibit early-stage PD symptoms. Binary classification was implemented, employing approaches such as SVM, boosted trees, random forests, and naive Bayes. The classification was early PD versus normal or SWEDD. Using data from the PPMI database, Rumman et al. [16] analyzed 200 SPECT images. Image processing and ANN methodologies were successful in achieving an accuracy of 94%. This method requires a comparatively tiny sum of computing resources, and it can assist a physician in shortening the lengthy procedure of diagnosing PD. Data obtained from the PPMI were utilized in the study of Ortiz et al [17]. A total of 269 DaTSCAN images, were analyzed. DCNN's LeNet-5 and AlexNet designs were used in the process of determining the individuals with PD and HC. Integral Normalization and feature Extraction (INE) Iso-surfaces were used in this investigation. Iso-surfaces resulted in a reduction in the amount of data by 95% while maintaining the highest quality of the information. Chakraborty et al. [18] obtained 3T T1-weighted MRI images from the PPMI database of a total of 406 participants. The DCNN model was utilised in order to train the network. Bayesian Sequential Model-based Optimization (SMBO) method is utilised for the purpose of selecting the ideal collection of hyperparameters, with the goal of preserving the generalizability of the overall model and achieving the highest possible objective score. Magesh and colleagues [19] collected 642 DATSCANs that have been labelled as PD or non-PD and have been trained on a DCNN (VGG-16) via transfer learning. The utilization of intensity normalization resulted in a precision of 95.2%. The model's goals were to make an early diagnosis for PD quicker and more intuitive. Using the Alex-Net Architecture Model with transfer learning and Generative Adversarial Networks (GAN) based data augmentation, Kaur et al. [20] achieved an accuracy of 89.23%. When combined with transfer learned classifier, the data-augmentation strategy is shown to result in an improvement of up to 3% accuracy. Mohammed et al. [21] used AlexNet architecture and 10-fold cross-validation and evaluated model performance. The network was able to achieve

99.34% accuracy. The fact that SPECT imaging technology is not readily available in most hospitals due to the fact that it is nuclear in nature and has a low resolution makes it difficult to visualize the basal ganglia in people with PD is perhaps, the primary drawback of this study. Pianpanit et al. [22] utilised four distinct DCNN architectures. 159 HC and 448 images of people with dementia were obtained from PPMI. The SHapley Additive exPlanation (SHAP) interpretation approach performed significantly better than the other methods when it came to differentiating between PD and HC individuals and delivered the highest quality heatmap at the location of the uptake depletion. Chien et al. [23] collected 205 DAT-SPECT PD images from one database and 52 images from another. An active contour segmentation to train the network, and a Transfer learned AlexNet for learning and classification, were implemented. PD and non-PD groups were differentiated using the Specific Binding Ratio (SBR) and the ASymmetry Index (ASI) with machine-learning and deep learning approaches. Dhanalakshmi et al. [24] received an aggregate of 1390 DaTscan images from PPMI. The DenseNet-121 model was employed exploring the region of interest (ROI), extracted from the images. The Contour Edge Detection algorithm was utilised to identify the borders of the objects in the images.

Table 1 provides a review of the existing research works, for PD detection. The Previous research [6, 7, 9, 12] has examined the progression of PD depending on the selected ROI, which arises the issue of being prone to knowledge loss, limited knowledge and segmentation mistakes, which can be a significant obstacle.

Table 1. Study of Current Methodologies for PD Classification.

Ref / author	Data Base	Count	Type of data	Method	Accuracy (%)
[6] Adeli et al.	PPMI	274 PD, 170 HC	MRI / Selected ROI	Joint Feature Sample Selection	81.9
[7] Amoroso et al.	PPMI	82 PD, 100 HC		ANN	83
[8] Sivaranjini et al.	PPMI	123 PD, 56 HC, 29 SWEDD		SVM classifier	86.37
[12] Quan et al.	PPMI	449 PD, 210 HC		Deep CNN using transfer learning	96.4
[14] Choi et al.	PPMI, SNUH	PPMI: 379 PD, 170 HC SNUH: 72 PD, 10 PT	SPECT / Selected ROI	Deep CNN with PD Net	PPMI-96 SNUH-98.8
[15] Prashanth et al.	PPMI	427 PD, 208 HC, 80 SWEDD		SVM, boosted trees, RF, naive Bayes	97.29
[16] Rumman et al.	PPMI	100 PD, 100 HC		Image processing and ANN	94
[22] Pianpanit et al.	PPMI	448 PD, 159 HC		Deep PD Net	96
[23] Chien et al.	PPMI	105 PD, 100 HC		TL AlexNet	86
[24] Dhanalakshmi et al.	PPMI	1160PD, 230 HC		DenseNet-121	99.2
[13] Arman et al.	PPMI	64 PD	DaTscan/ Selected ROI	RF analysis for prediction.	P<0.001
[17] Ortiz et al.	PPMI	158 PD, 111 HC		LeNet and AlexNet	95.1
[19] Magesh et al.	PPMI	430 PD, 212 HC		VGG16 using TL	95.2
[42] Castillo et al.	PPMI	168 PD, 194 HC, 26 SWEDD		Ensemble classification	64.5
[63] Aditi et al.	PPMI	70 Mild PD, 70 Moderate PD, 70 Severe PD, 70 HC	Hand Drawings	Deep Learning (YOLO v8)	94
[10] Salvatore et al.	PPMI	56 PD, 28 HC	MR I/ Whole Brian	SVM classifier	85.8
[11] Brahim et al.	PPMI	158PD,111 HC		Histogram Equalizer with PCA, SVM	92.6
[18] Chakraborty et al.	PPMI	203 PD, 203 HC		CNN with Bayesian SMBO method	95.29
[20] Kaur et al.	PPMI	67 PD, 85 HC		TL AlexNet + data augmentation	89.23
Proposed Approach	**PPMI, LSSH**	**PPMI: 40 PD, 30 HC** **LSSH: 20 PD, 10 HC**		**EL with TL + DCNN + augmentation**	**99**

***** EL-Ensemble Learning; TL-Transfer Learning; aug-augmentation; PT-Parkinsonian Tremor; RF-Random Forest;

3. PROPOSED SYSTEM

Transfer learning [26], data augmentation [27], fine-tuning [28] and ensemble learning [29,58,59] are some examples of processes that are often used to decrease the gap as much as possible between training and validation mistakes. This research overcomes the discussed limitations by making use of the subject's whole brain in the study rather than using segments, thus keeping all the essential details intact. As a novel aspect, an Optimal Weighted Priority Mean (OWPM) ensemble method using classifier fusion mechanism was implemented in this work, for the very first time. The entire process flow of the suggested approach can be observed from Fig. 3, 4. The proposed method uses a neural network, to which brain MR Images are fed as input. In the initial part of the process, the images are pre-processed to remove some of the background noises. In the next stage, data augmentation techniques are carried out to handle the issue of compact size of the dataset as well the data imbalance, besides improving the efficacy of the suggested method. In the next phase, transfer learned methodology associated with EfficientNetB1, ResNet-50, ResNet152V2, MobileNetV2 are considered and some of the rear layers are transformed to equal the number of classifications, suitable for this application. This is done in preparation for the fourth step, which is the actual classification of the images. In the next step of the process, the suggested model's performance is assessed using test MR scans of HC and people with PD. Finally, the ensemble method is applied to the developed DCNN models and performance analysis is analyzed and compared.

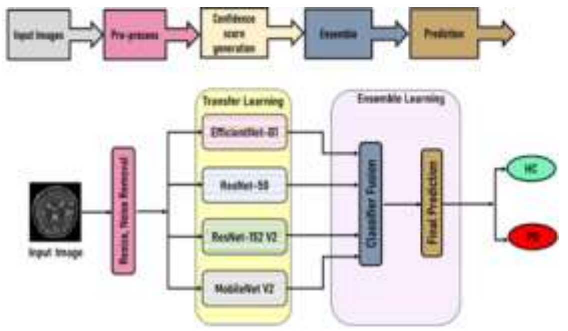


Fig. 3. The process flow of the suggested approach.

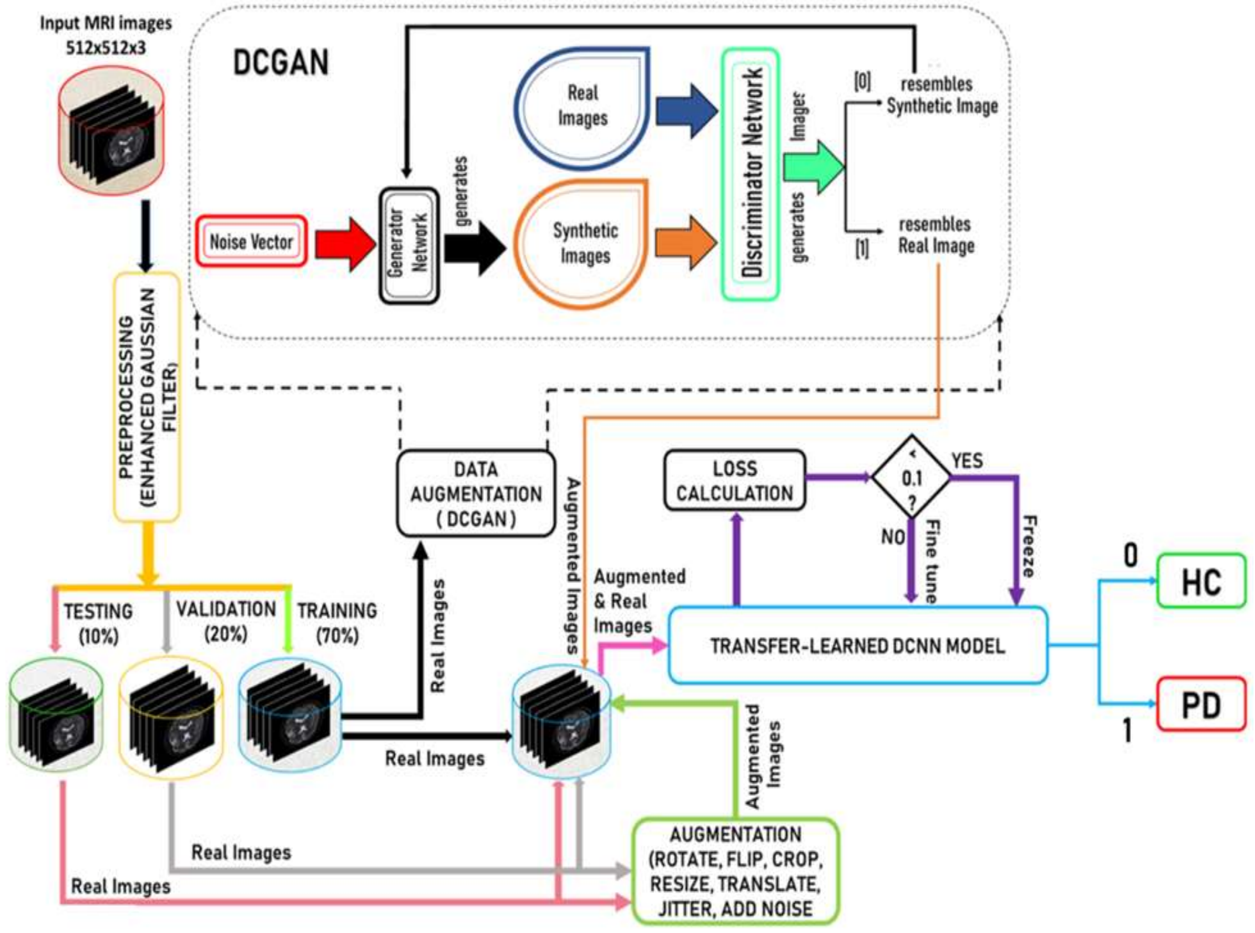


Fig. 4. Transfer-learned DCNN training/testing process.

3.1. Materials and Methods

The data for the proposed approach was sourced from the esteemed PPMI database (www.ppmi-info.org/data), which stands as an international landmark and multicenter endeavor aimed at researching biomarkers underlying the progression of PD. 45 images of PD and 25 images of HC were collected from the PPMI, contributing to a total of 70 images. These images belong to the persons, who are aged between 40 and 50years. These images were used for training the classification models. The data for testing was collected from LSSH. A total of 30 images, of which 15 belongs to PD and 15 belongs to HC categories were obtained, satisfying our criteria that the persons are aged between 40 and 50years of age. These collected images underwent interpretation by a panel of two experienced nuclear medicine physicians. Table 2 can be observed to find these details. Visual assessment of the images based on the density of DopAmine Transporter (DAT) led to the classification of subjects into a group with reduced DAT density (PD) and another with normal DAT density (HC). The MRI scans selected for this study were meticulously chosen, adhering to specific imaging protocols outlined in Table 3. The training data is augmented using GAN an testing data is augmented using different augmentation mechanisms (rotation, flipping, resizing, etc).).

Table 2. Data Split.

Category /Dataset	Real Data			Augmented Data		
	HC	PD	Total	HC	PD	Total
Training / PPMI	25	45	70	4000	7200	11200
Validation / LSSH	10	10	20	100	100	200
Test / LSSH	5	5	10	50	50	100
Total	40	60	100	4090	7290	11500

Table 3. Specifications of the acquired MRI scans.

Image Parameters	PPMI	LSSH
Modality/Weighting	MRI / T1	MRI / T1
Research Group	HC, PD	HC, PD
Visit	Initial, Preliminary	Initial, Preliminary
Dimensions (pixels)	224 x 224 x 178	224 x 224 x 178
Interslice Gap	1.0 mm	1.0 mm
Slice Thickness	1.0 mm	1.0 mm
Spacing	1.0 × 1.0 × 1.0 mm	1.0 × 1.0 × 1.0 mm
Acquisition Plane / Type	Axial / 3D	Axial / 3D
Gender*	PD: M-32, F-13	PD: M-12, F-03
	HC: M-16, F-09	PD: M-09, F-06
Age	Between 40-50 years	Between 40-50 years
Disease Stage	H&Y stage-I / Stage-II	H&Y stage-I / Stage-II
*M-Male; F-Female;		

3.2. Image Preprocessing

The original datasets were of size 512 □ 512 in Digital Imaging and Communications in Medicine (DICOM) format, which are converted to 3D Neuroimaging InFormatics Technology Initiative (NIfTI) format using the dcm2nii tool in MRICron software (http://www.mccauslandcenter.sc.edu/mricro/mricron/). Preprocessing involved two main steps: skull stripping using the Brain Extraction Tool (BET) (Smith et al., 2004; Jenkinson et al., 2012) and normalization to Montreal Neurological Institute (MNI) space through co-registration with the MNI template (MNI152 T1MPRAGE-1mm brain) in FSL 4.1 software (Grabner et al., 2006). The preprocessed images were subsequently brought into MATLAB through the utilization of the 'Tools for NIfTI and ANALYZE Image' toolbox(http://www.mathworks.com/matlabcentral/fileexchange/8797). The final whole-brain volumes were limited to 224 × 224 × 178 voxels, which ensures that all the brain images used in the study have the same spatial dimensions as needed for this study, making it easier to analyze and compare them in subsequent data processing and analysis steps. As the obtained MRI datasets contain a large amount of noise, an enhanced gaussian filter, (modified version of the traditional Gaussian filter) was employed to remove it. It is a high-pass filter which is added to the Gaussian kernel, that amplifies the high-frequency elements of the image while suppressing the low-frequency noise. This kernel is used to preserve the edges, fine details in the image. Reduction in noise and an increase in analytical sensitivity are the outcomes of the filter [45] and hence suits best for the medical image processing [46]. The effect of the preprocessing can be observed from Fig. 5. The pseudocode is displayed in Algorithm-1.

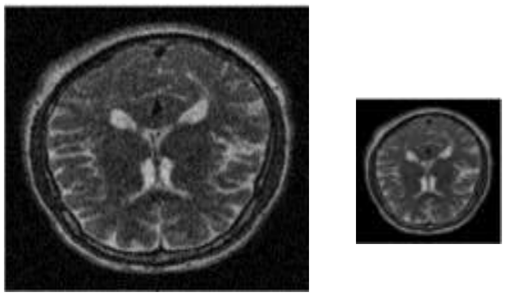


Fig. 5. (left to right) before and after preprocessing.

Algorithm-1: Enhanced Gaussian Filter

```
1.Input: Image of size $224x224$ with noise; An empty matrix of size nxn to represent Gaussian Filter kernel $GF_{nxn}$ ; the standard deviation (σ) for the Gaussian distribution; predefined threshold parameter (T);
2.Output: Enhanced Gaussian Filtered image of size $224x224$
3.Begin
  3.1 for each position in kernel:
      3.1.1 x = row -n / 2
      3.1.2 y = col - n / 2
      3.1.3 $\text{kernel}[\text{position}] = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
      3.1.4 Height, width = dimensions of the image (i.e., 224,224)
      3.1.5 border = n // 2
      3.1.6 for i = 1 to height:
      3.1.7     for j = 1 to width:
      3.1.8         window = empty matrix (n,n)
      3.1.9 for each position in window:
            3.1.9.1 $\text{window}[\text{position}] = \text{image}\,[\text{i} + \text{row offset} - \text{border}, \text{j} + \text{col}_{\text{offset}} - \text{border}]$
            3.1.9.2 sum = 0
            3.1.9.3 for each value in window:
            3.1.9.4     sum += value
            3.1.9.5     $\text{local}_{\text{mean}} = \frac{\text{sum}}{\text{total elements in (window)}}$
            3.1.9.6     deviation_sum = 0
            3.1.9.7 for each value in window:
            3.1.9.8     $\text{deviation} = \text{value} - \text{local\_mean}$
            3.1.9.9 $\text{deviation}_{\text{sum}} += \text{deviation}^2$
            3.1.9.10        $\text{local}_{\text{std}_{\text{dev}}} = \frac{\sqrt{(\text{deviation}_{\text{sum}})}}{\text{total elements in (window)}}$
            3.1.9.11        $\text{Wt}_{\text{ixj}} = 1 - \frac{\text{local}_{\text{std}_{\text{dev}}}}{\text{threshold}}$
            3.1.9.12        enhanced_kernel = empty_matrix (n,n)
            3.1.9.13        for each position in enhanced_kernel:
            3.1.9.14
                $\text{enhanced}_{\text{kernel[position]}} = \text{kernel}[\text{position}] * \text{weight}$
            3.1.9.15        $\text{EGF}_{\text{ixj}} = \text{GF}_{\text{ixj}} * \text{Wt}_{\text{ixj}}, \forall\, 1 \le i \le n,, \forall\, 1 \le j \le n$
            3.1.9.16            sum = 0
```

```
                3.1.9.17          for each value in enhanced_kernel:
                3.1.9.18          $sum += value$
                3.1.9.19          $enhanced\_kernel = \frac{enhanced\_kernel}{sum}$;
                                  $convolved\_value = 0$;
                3.1.9.20          for each value, weight in window, enhanced_kernel
                3.1.9.21          $convolved\_value += value * weight$
                3.1.9.22          $result[i, j] = convolved\_value$
                3.1.9.23          $filtered\_image = result$
        3.1.10  End
3.2 End
```

3.3. GANs

GANs used to be considered complex and advanced a few years ago. But they have become a common and widely adopted tool in various fields, now-a-days. They create realistic images and have various applications in image manipulation techniques like image synthesis, augmentation [34], super resolution [30], image-to-image translation [31], image blending and [32], image inpainting. Deep Convolutional Generative Adversarial Networks (DCGAN) are an extension of GANs, that are designed for producing high-quality, realistic images [27]. The distinguishing feature of DCGANs is the incorporation of convolutional layers in both the generator and discriminator networks. This architectural choice enables the networks to effectively capture spatial patterns and dependencies within the data. As a result, DCGANs consistently demonstrate proficiency in generating detailed and visually coherent images across diverse domains. DCGANs make use of two adversarial networks (G(z) and D(x)), of which generator generates a photorealistic image G(z) with the intention of fooling the discriminator, while the discriminator sends back the images that resemble unrealistic (D(x)) back to the generator network i.e., the generator's job is to find the lowest possible value for the cost function V (D, G), while the discriminator's job is to find the highest possible value [33, 34]. As there are very limited labelled images available, the process of analyzing medical images requires the use of an important technique called image augmentation [34]. The DCGAN-based data augmentation [35] was utilized to generate fresh images. A generator network with an architecture that is comprised of seven layers has been chosen, and this design is depicted in Fig. 6. The data that is supplied to the generator network is a vector that has been selected at random from a set of hundreds of invariant values that are arrayed between 0 and 1. The well-connected layer, which consists of 128 □ 128 □ 64 = 1048576 values, is immediately supplied by the input data and is consequently modified to achieve 64 □ 64 □ 128 measure, by passing through the convolution layer. A Leaky ReLU activation function is appended to each of the layers, along with batch normalization application. The process is repeated for the consecutive six convolution layers of different sizes of inputs and outputs, to obtain a final block of size 1 □ 1 □ 512. These 512 units are fed as input to the deconvolution layer, to produce a block of 2 □ 2 □ 1024. A ReLU activation function is applied to every layer, along with the application of dropout and batch normalization. The process is repeated for the successive two Deconvolution layers of different sizes of inputs and outputs, to obtain an output block of size 8□8□1024. It is passed through a deconvolution layer, equipped with ReLU activation function and batch normalization. The process is repeated for the consecutive three convolution layers of different sizes of inputs and outputs, to obtain a final block of size128 □ 128 □ 128. This block is passed through a deconvolution layer supported with ReLU activation function and Tanh functions to obtain a synthetic image of size 224 □ 224 □ 3. In addition to the real image samples, fake samples obtained from the generator are fed as input to the first convolution layer of discriminator network, which is of size 128 □ 128 □ 64. Leaky ReLU activation function is applied here and this processed block is fed as input to the convolution layer, where Leaky ReLU along with batch normalization is applied.

This process is repeated for two times with different sizes of blocks to obtain a block of size 31□31□512. This block of data is passed through a convolution layer along with sigmoid activation function, thus resulting in generating a block of size 30*30*1. Sigmoid activation function is applied on this block finally, to produce the final output as ‘0’ or ‘1’ to represent a fake image or real one respectively. If the resultant image resembles fake, it is fed back to the generator, for proper training. Otherwise, added to training set. The iterative process of the DCGAN training is outlined in Algorithm-2 (Appendix). Random numbers are used to determine the values of the weight and bias parameters. All weights were initialized from a zero-centered Normal distribution with standard deviation 0.02. In the LeakyReLU, the slope of the leak was set to 0.2 in all models. Adam optimizer was used with a suggested learning rate of 0.001. Additionally, when β1 was assigned with a value of 0.9, it resulted in training oscillation and instability while reducing it to 0.6 helped stabilize training. Assuming total number of batches as batchcount, batch size as batchsize, total number of epochs as epochcount, total number of images as imgcount, the total number of images generated during training (imggen-count) can be calculated as follows:

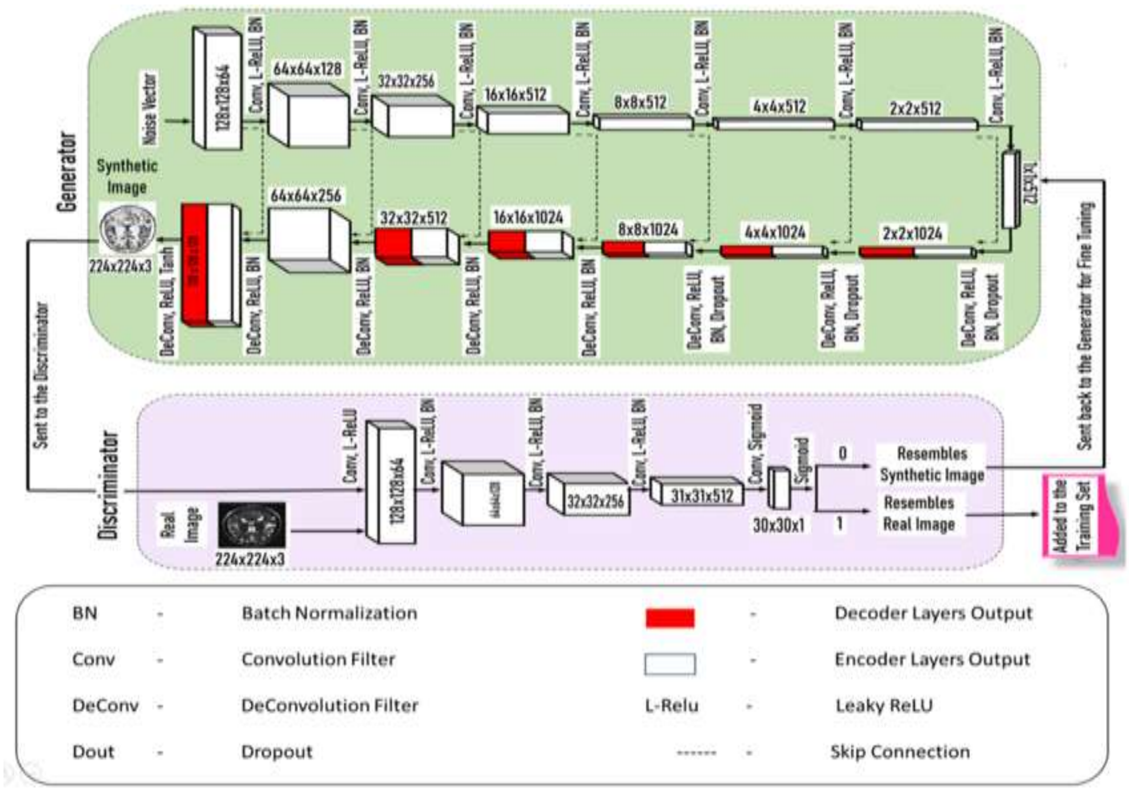


Fig. 6. DCGAN architecture.

First, calculate the number of batches per epoch. Since we have 70 input samples for training, and a batch size of 10, the number of batches per epoch would be:

$$(\mathbf{batch_{count}}) = \frac{(\mathbf{img_{count}})}{\mathbf{batch_{size}}} = \frac{\mathbf{70}}{\mathbf{10}} = \mathbf{7} \tag{1}$$

Now, calculate the total number of images generated during training with 160 epochs as follows:

$$\mathbf{image_{gen-count}} = \mathbf{batch_{size}} * (\mathbf{batch_{count}}) * (\mathbf{epoch_{count}}) \tag{2}$$

$$\mathbf{image_{gen-count}} = \mathbf{10 * 7 * 160 = 11,200\ images} \tag{3}$$

Therefore, a total of 11,200 images (4000 HC images and 7200 PD images) are generated during data augmentation over 160 epochs. These images are saved to a local storage location on the computer and are then organized into folders based on their class labels, to maintain a clear structure. The images are finally fed as input to the DCNN models. The GAN network goes through 'n' number of rounds, and the loss matrix is calculated at the end of each iteration. This loss function is used to measure the performance of both the generator and the discriminator. It is a combination of two terms: the generator loss and the discriminator loss.

Generator Loss: It measures how well the generator is able to produce fake samples, similar to the real.
Discriminator Loss: It measures how well the discriminator is able to vary between real, fake samples.

During training, the generator and discriminator are updated alternately to minimize this loss function. Let 'G' be the generator function that takes in a random noise vector 'z' and outputs a generated sample 'x', and let 'D' be the discriminator function that takes in a sample ‘x’ and outputs a probability score ‘p’ indicating the probability that ‘x’ is a real sample. The mathematical equation for the overall loss calculation (L) in a GAN, can be represented as

$$\mathbf{L = L_G + L_D} \quad (4)$$

Where ‘LG’ is the generator loss, which is calculated using the cross-entropy loss between the discriminator's output on the generated samples and a target vector of all 1’s as

$$\mathbf{L_G = -\log\left(D\left(G(z)\right)\right)} \quad (5)$$

and ‘LD’ is the discriminator loss, which is calculated as the sum of the cross-entropy loss between the discriminator's output on the real samples and a target vector of all 1’s, and the cross-entropy loss between the discriminator's output on generated samples and a target vector of all 0’s as:

$$\mathbf{L_D = -\log\left(D(x)\right) - \log\left(1 - D\left(G(z)\right)\right)} \quad (6)$$

The entire weight and bias values are updated, accordingly. This process continues until both networks become stable..

3.4. Optimizers

Stochastic Gradient Descent (SGD): SGD is an optimization algorithm used to determine the model parameters that best match the predicted and the actual outputs. It calculates the gradient using just a random small part of the observations instead of all of them. The major advantage of SGD is that the update steps are performed very quickly that leads to reach the minimum in a very small amount of time.

Adam: Adaptive Moment Estimation (Adam) is an optimization technique for gradient descent. It takes a smaller amount of memory and is more efficient. In this study, the GAN learning rate is configured to be 0.001, the number of iterations to be 160, and batch size as 10. To reach equilibrium, it is necessary for both the Generator and the Discriminator networks to engage in concurrent or the adversarial learning. The variation in the loss function can be observed from Fig. 7, 8. Adam was proved to be better, compared to SGD, for image classification tasks [43, 46, 47]. The loss function transition of the Generator Network gets reduced over time while it gets increased, with adversarial network. This suggests that the modelling level and bias against their most recent data has variable change levels, which is beneficial for the model. During the first round of iteration, the produced images exhibited a high degree of inconsistency. It started to decrease after 150 iterations, and became stable for good, after a total of 160 iterations. When it comes to the replicating periods depicted in Fig. 8, it is easy to demonstrate that the value shift of the adversarial network's loss function starts to climb from roughly 0.65 onwards and continues, until it reaches a maximum. Fig. 9 (x1, y1) demonstrates that the images have become distorted and faulty after 80th iteration, while Fig. 9 (x2, y2) demonstrates that contours and textures have started to take shape after the 120th iteration. When looking at Fig. 9 (x3, y3), it is easy to see that after 150 iterations, the individualized characteristics have been improved and have become, to some extent, more improved and have become, to some extent, more comparable to the actual images. This is evidenced by the fact that they are now more like the genuine images. Succeeding 160 iterations, the suggested strategy is said to be balanced. PSNR is a measurement instrument that is used to determine image quality and its equation can be represented as

$$\mathbf{Peak\ Signal - to - Noise\ Ratio = \frac{10\log\left(Count - 1\right)^2}{MSE}} \quad (7)$$

Where ‘Count’ represents the number of maximum intensity levels and MSE represents Mean Squared Error. If the PSNR number is higher, then the image will have less blurring and other distortions. PSNR>=y, where y=30, makes image, superior to the standard. The objective was not accomplished until more than 150 cycles were carried out and can be seen from Table 4. It can be observed that as the number of iterations increases, the overall image quality that is produced by the study also improves. A high value of PSNR and stability in the images is observed, as the total number of repetitions move towards 160. It has been found that the greatest potential outcomes may be accomplished when the convolution kernel for feature extraction is set to 3□3, and the iteration number for DCNN is set to 160.

3.5. DCNN

DCNN is a type of deep learning algorithm commonly used for image recognition and processing tasks. The key feature is their ability to automatically learn hierarchical representations of features in images. DCNNs have revolutionized the field of image classification by providing state-of-the-art performance on a wide range of datasets. The role of DCNNs in PD detection is to automatically learn features that may include changes in brain morphology or activity that are associated with the disease. Once trained, the DCNN can be used to classify new medical images as either HC or PD. Different characteristics related to various DCNN models that support image classification tasks are presented in Table 5. It can be observed that Alexnet, GoogLeNet and VGGNet were not considered in this study because of the memory requirements, architecture complexity and efficiency. EfficientNet B1, ResNet-50, ResNet152V2, and MobileNetV2 are all popular and widely used DCNN architectures in image classification tasks. They are chosen for the proposed study because of their image size, number of trainable parameters, and associated merits, demerits.

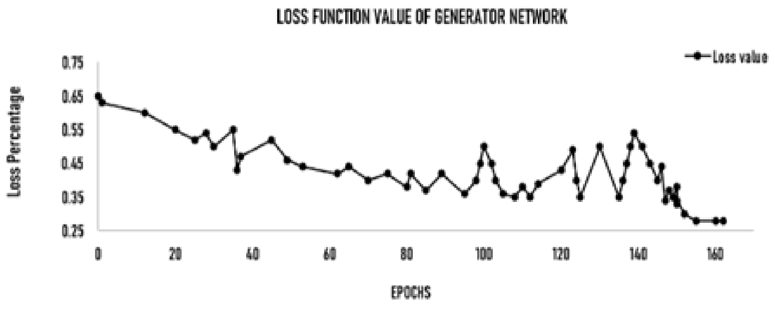


Fig. 7. Loss function of Generator network.

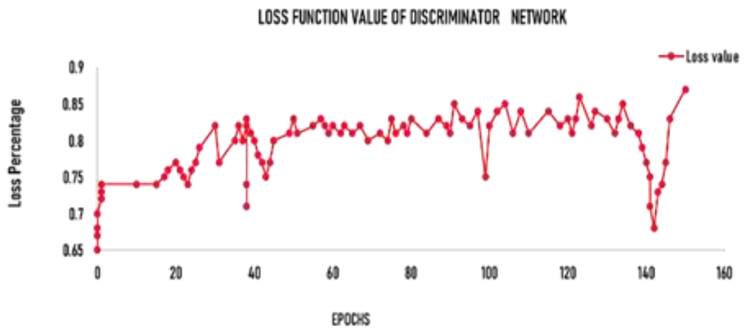


Fig. 8. Loss function of Discriminator network.

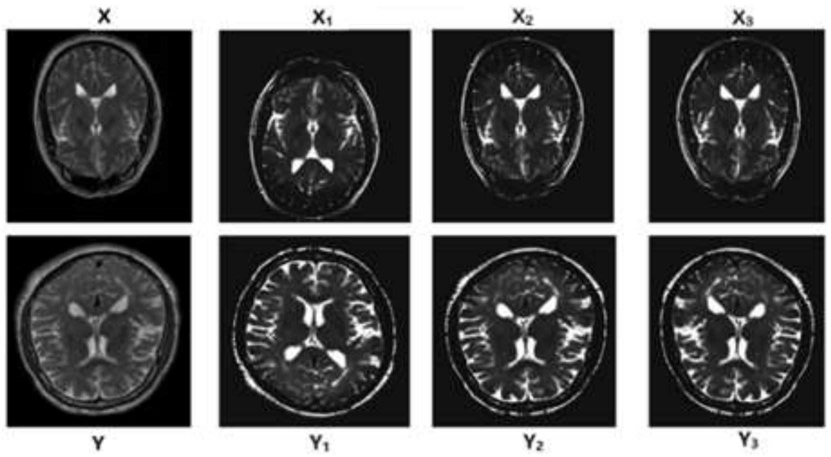


Fig. 9. (x, y) Real images of PD, HC; (x1, y1), (x2, y2), (x3, y3): Synthetic images of PD, HC after 80, 120, 160 iterations respectively.

Table 4. Iterations Vs PSNR Values.

Epochs	1	40	80	120	160
PSNR value (dB)	0	19.61	24.75	32.86	35.18

Table 5. Characteristics of DCNN Models Using Images.

DCNN Model	Image Size	#parameters (millions)	Merits	Demerits
AlexNet	224 × 224	62.3	Introduced ReLU activation and dropout.	High computational, memory requirements; Prone to overfitting on small datasets.
VGGNet	224 × 224	138	Uniform architecture, Strong performance on binary classification problems, Introduced Inception module for feature combination.	Extremely deep with a high number of parameters and not the most parameter efficient architecture.
GoogLeNet	224 × 224	5	High accuracy with computational efficiency	Complex architecture; challenging to train.
ResNet-50	224 × 224	23.5	Residual connections mitigate vanishing gradient problem, Effective for binary tasks, due to skip connections.	Deeper variants can be computationally expensive; It may require substantial data for training deep variants.
ResNet152V2	224 × 224	60.4		
EfficientNet B1	224 × 224	6	Achieves high accuracy with few parameters	Low performance on very small datasets.
MobileNetV2	224 × 224	4.2	Lite weight, highly computational efficient and low memory footprint.	Limited model capacity compared to larger architectures.

3.6. Transfer Learning

Instead of copying the complete network, including all of the fully connected layers, it is more common practice to merely copy the weights that are included within the convolutional layers. The pre-trained DCNN models (EfficientNetB1, ResNet-50, ResNet152V2 and MobileNetV2) which were trained on ImageNet dataset with 1000 classes were chosen for the classification of brain MRI with two distinct classes, HC and PD, using transfer learning mechanism. Additionally, DCNN would require a significant quantity of MRI to generate upgraded weights and produce new ones. As a result, shifting weights from the pre-trained model into the DCNN will cause it to produce output that is desirable and will speed up the rate at which it converges [20]. The number of output neurons in pre-trained version using ImageNet data set is 1000, but it is only ‘2’ to represent HC and PD classes as per the stated problem. Fig. 10 shows the transfer-learned DCNN architecture.

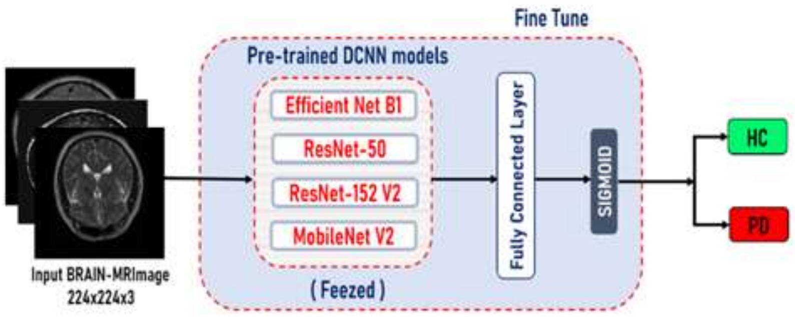


Fig. 10. Transfer Learned DCNN.

3.7. Ensemble Learning

Ensemble techniques are powerful methodologies that leverage a diverse range of models and combine their predictions to create a robust and reliable final model. Instead of relying solely on a single model and hoping for the best accuracy, ensemble learning takes advantage of combining multiple competing models to improve deep learning outcomes. This approach proves to be more effective in predicting future outcomes compared to using a solitary model alone [40, 58, 59]. The proposed ensemble model can be observed from Fig. 3. The dataset under study is experimented with three different ensemble mechanisms of bagging technique [59], that combine the strengths of multiple classifiers, compensating for their individual weaknesses and biases. They are (a) average-voting and (b) majority-voting (c)

OWPM method. It has been shown to improve the prediction accuracy, robustness, and enhance the generalization ability of the ensemble model, by taking into account the outcomes of the various classifiers and make the final choice.

3.7.1 Average Voting method

In In the average voting scheme, average function is applied over the predicted outcomes $[\mathbf{O_1}, \mathbf{O_2}, \mathbf{O_3}, \ldots \ldots \mathbf{O_n}]$, generated from different classifiers $(\mathbf{DCNN_1}, \mathbf{DCNN_2}, \ldots \ldots \mathbf{DCNN_n})$ of all the DCNN models, where 'n' is the total number of models. Let these outcomes belong to multiple classes $[\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}, \ldots \ldots \mathbf{C_j}]$, where 'j' is the total number of classes. Let '**Res**' be the predictive output of a sample 'x', to be generated using ensemble mechanism. Then, the average value is calculated as follows:

$$\mathbf{average} = \left(\frac{\textbf{sum of outcomes of classifiers}}{\textbf{total number of classifiers}}\right) \qquad (8)$$

$$\mathbf{average} = \left(\frac{\sum_{i=1}^{n} \mathbf{O_i}(x)}{\mathbf{n}}\right) \qquad (9)$$

In this study, there are only two classes that represent PD (=1) and HC (=0). 4 DCNN models were employed in the proposed study, that might generate the outcomes as either '0' or '1'.

.

$$\mathbf{Res} \begin{cases} \mathbf{1}, \text{ If average} > 0.5, \text{ indicating PD.} \\ \mathbf{0}, \text{ If average} <= 0.5, \text{ indicating HC.} \end{cases} \qquad (10)$$

In this scheme, the outcomes $[\mathbf{O_1}, \mathbf{O_2}, \mathbf{O_3}, \ldots \ldots \mathbf{O_n}]$, belonging to multiple classes $[\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}, \ldots \ldots \mathbf{C_j}]$, generated from different classifiers $(\mathbf{DCNN_1}, \mathbf{DCNN_2}, \ldots \ldots \mathbf{DCNN_n})$ are counted and the class with the highest count is considered as the resultant prediction, for the subject being analyzed using ensemble method. In this study, there are only two classes that represent PD (=1) and HC (=0). Let '$\mathbf{T_P}$' and '$\mathbf{T_N}$' represent the total number of outcomes from different classifiers resulting in '1' and '0' respectively. Let '**Res**' be the predictive output to be generated using ensemble mechanism and it can be calculated as:

$$\mathbf{Res} = \begin{cases} \mathbf{1}, \text{ If } (T_P \geq T_N), \text{ indicating PD} \\ \mathbf{0}, \text{ If } (T_P < T_N), \text{ indicating HC.} \end{cases} \qquad (11)$$

OWPM method in an ensemble mechanism, which is an extension of weighted majority voting scheme, that introduces "priority" as an additional layer of information about the relative importance of classifiers. The proposed method uses classifier fusion ensemble mechanism [57] which involves a slightly different mechanism for combining classifier outputs. In this scheme, let $(\mathbf{DCNN_1}, \mathbf{DCNN_2}, \ldots \ldots \mathbf{DCNN_n})$ represent various classifiers generating different outcomes $[\mathbf{O_1}, \mathbf{O_2}, \mathbf{O_3}, \ldots \ldots \mathbf{O_n}]$ belonging to multiple classes $[\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}, \ldots \ldots \mathbf{C_n}]$. These classifiers are assigned priorities $(\mathbf{P_1}, \mathbf{P_2}, \ldots \ldots \mathbf{P_n})$ and Weights $(\mathbf{W_1}, \mathbf{W_2}, \ldots \ldots \mathbf{W_n})$. The classification approach is as follows:

- Arrange the models in descending order of efficiency; The highest performance model is assigned with the highest priority (P=n) and largest weight (W=n); The model with the next highest performance is assigned with the next highest priority (P=n-1) and the next largest weight (W=n-1); This process is continued until the lowest performance model is assigned with the lowest priority (P=1) and the smallest weight (W=1).
- Let '**Res**' be the predictive output of a sample 'x', to be generated using ensemble mechanism.
- Calculate the Weighted Priority Mean as the ratio of sum of products of weights and priorities of models w.r.t sample 'x' along with their outcomes to the sum of their products of weights and priorities. i.e.,

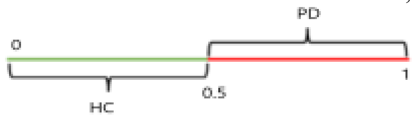
$$\textbf{Weighted Priority Mean} = \frac{\sum_{i=1}^{n} \mathbf{W_i P_i(x) * O_i(x)}}{\sum_{i=1}^{n} \mathbf{W_i P_i(x)}} \qquad (12)$$

Where 'i' represents the classifier, and 'n' represents the total number of classifiers. In this study, there are only two classes that represent PD (=1) and HC (=0). 4 DCNN models were employed in the proposed study, that might generate the outcomes as either '0' or '1'.

$$\mathbf{Res} = \begin{cases} \mathbf{1}, \text{ If average} > 0.5, \text{ indicating PD.} \\ \mathbf{0}, \text{ If average} <= 0.5, \text{ indicating HC.} \end{cases}$$

(13)

The pseudocode is presented in Algorithm-3. The final classification of a subject in a case where equal number of votes results for both the classes may need more data for the model to train on, or, an external intervention by an expert medical practitioner. The subject was assigned to either of the classes 'HC' or 'PD', in this study.



Algorithm-3: The OWPM Algorithm

```
1. Input: The trained and analyzed DCNN models ($DCNN_1, DCNN_2, \dots \dots DCNN_n$), utilized in this study; the models efficiencies ($E_1, E_2, E_3, \dots \dots E_n$); the models outcomes [$O_1, O_2, O_3, \dots \dots O_n$]; total number of models 'n'; the prediction for the subject with unknown label 'Res';
2. Output: Prediction for the subject with unknown label as PD (or) HC, using the OWPM algorithm.
3. Begin
   3.1 Arrange the models ($\mathbf{DCNN_1}, \mathbf{DCNN_2}, \dots \dots \mathbf{DCNN_n}$) in descending order of their efficiency and assign priorities ($\mathbf{P_1}, \mathbf{P_2}, \dots \dots \mathbf{P_n}$) to these models and Weights ($\mathbf{W_1}, \mathbf{W_2}, \dots \dots \mathbf{W_n}$) in such a way that the highest performance model ($\mathbf{DCNN_i}, say$), will be assigned the highest priority 'n' ($\mathbf{P_i} = \boldsymbol{n}$) and the highest weight ($\mathbf{W_i} = \boldsymbol{n}$), the model with the next highest performance ($\mathbf{DCNN_x}, say$),will be assigned priority 'n-1' ($\mathbf{P_x} = \boldsymbol{n} - \mathbf{1}$) and highest weight of 'n-1' ($\mathbf{W_x} = \mathbf{n} - \mathbf{1}$) and so on. This process is repeated until the model with the least performance will be assigned the lowest priority '1' and lowest weight '1'.
   3.2 As a subsequent step, calculate the weighted priority mean as the ratio of sum of the product of outcomes ($O_1, O_2, O_3, \dots \dots O_n$) and their associated weights ($\mathbf{W_1}, \mathbf{W_2}, \dots \dots \mathbf{W_n}$) and priorities ($\mathbf{P_1}, \mathbf{P_2}, \dots \dots \mathbf{P_n}$) to the sum of the products of weights and priorities, as follows:
       $\textbf{Weighted Priority Mean} = \frac{\textbf{sum of product of weights, priorities and models outcomes}}{\textbf{sum of the weights and priorities}}$
       $\textbf{Weighted Priority Mean} = \frac{\sum_{i=1}^{n} \mathbf{W_i P_i} * \mathbf{O_i}}{\sum_{i=1}^{n} \mathbf{W_i P_i}}$
4. If the value obtained in the previous step is greater than 0.5, Res=1 (PD); otherwise, Res=0 (HC);
5. End
```

Advantages of OWPM model over Weighted Majority Voting Schema:

- Weights focus on the contribution of a classifier's prediction to the final decision, while priorities focus on the relative ranking of efficiency among classifiers. Weights alone might not be sufficient, if there is a need to explicitly consider the order of efficiency among classifiers. Priorities alone might not capture the reliability of classifiers. The combination of weights and priorities allows the adaptability to problem-specific characteristics, providing a more versatile and fine-grained ensemble approach.
- Weights control the influence of each classifier based on its individual performance, while priorities introduce a global ranking that affects the overall contribution of each classifier. The use of both priorities and weights provides a comprehensive approach to ensemble learning, enabling a system to consider both the accuracy and order of efficiency among classifiers.

3.8. Training And Fitting

It is vital to employ a pre-trained model, also known as transfer learning, to increase the performance of image classification. This is because the pre-trained model's weights had previously been designed with characteristics that were pertinent to the large majority of computer vision problems. To facilitate transfer learning, a data set consisting of training, verification, and test information was constructed. Because the dataset that was used had images that were comparable to those that were used in ImageNet, it is conceivable to use the transfer-learning technique in this investigation. It would have been essential to retrain some of the layers in the model in the event that the images are not comparable to those that were used to train the models that are now available. Fine tuning can be accomplished by making appropriate modifications to the topmost layer that is already fully connected. All the tasks are carried out on a laptop equipped with an Intel ® Core (TM) i5-1135G7 microprocessor and 8 gigabytes of Random-Access Memory (RAM).

3.9. Output Layer Training

To enhance the model's performance, the original output layer was removed as the first step, followed by introducing a unique layer with two units in its place (PD and HC, as per the application) and a sigmoid activation function. During training, only the weights of this newly added output layer are adjusted, while the rest of the network remains frozen. This approach allows us to focus on fine-tuning specific aspects without disrupting the existing architecture. For optimization, Adam, a renowned optimizer in deep learning [38], was utilized. Selecting an appropriate initial learning rate for the DCNN model during fine-tuning can be challenging, as the learning rate may decrease throughout the process. To address this challenge, the Grid search optimization approach was employed, which proves substantially more efficient than random search. It enables us to methodically explore parameter combinations and select the most suitable configuration, ultimately improving the model's effectiveness [20]. Each classification problem is subjected to a series of tests, systematically varying the selected parameters such as the number of training epochs, the initial learning rate, the reduction factor, and the frequency. Through this iterative process, the optimal values for these parameters were determined, ensuring that the DCNN model is properly trained and achieves optimal performance on the specific classification task at hand. be challenging, as the learning rate may decrease throughout the process. To address this challenge, the Grid search optimization approach was employed, which proves substantially more efficient than random search. It enables us to methodically explore parameter combinations and select the most suitable configuration, ultimately improving the model's effectiveness [20]. Each classification problem is subjected to a series of tests, systematically varying the selected parameters such as the number of training epochs, the initial learning rate, the reduction factor, and the frequency. Through this iterative process, the optimal values for these parameters were determined, ensuring that the DCNN model is properly trained and achieves optimal performance on the specific classification task at hand.

3.10. Hyperparameter Tuning

Grid search method was employed for performing hyperparameter tuning. In this method, all of the potential combinations of the hyperparameters and their values are first laid out on a grid. Next, an algorithm is built for each combination of the grid's components. After making a note of the respective parameters' degrees of accuracy, an analysis of the validation data using those parameters was done. It took five steps to obtain the desired level of hyperparameter tuning. In the first stage of the process, the goal was to randomly assess how well each hyperparameter performed with reference to accuracy and loss, during both the training and validation phases. Next, 8 different experiments were conducted, out of which the best two models with reference to accuracy and loss function in the validation process are selected, for the classification of the test dataset. The initial weights assigned during training will be adjusted accordingly, during the process. Next stage was to investigate this disparity, and the two studies that performed the best overall in terms of accuracy underwent two further training cycles. The batch size for the best sample, which was obtained in the fourth stage, varied between 32, 64, 128, and 256 in the following stage. When either the algorithm reached its maximum epochs or testing phase's validation output started to decline over a set amount of time, the training procedure was stopped. Overfitting was avoided, as two different data sets were used for validation and testing, in this study. Additionally, Dropout method was employed during the training phase. The top level's goal is to lessen the average loss, while the bottom level's goal is to maximize the accuracy. Regular adjustments are made to hyper-parameters and the optimal ones are determined, based on the validation set. They can be observed from Table 6. The step-by-step classification procedure is outlined in Algorithm-4.

```
Algorithm-4: PD classification using pre-trained DCNN

Input:   Pretrained DCNN model (Lm), Augmented training MRI ($\mathbf{I}_{\mathbf{train}} = \mathbf{I}_{\mathbf{R}} + \mathbf{I}_{\mathbf{S}}$), Validation Images ($\mathbf{I}_{\mathbf{val}}$), Validation data $\mathbf{I}_{\mathbf{val}}$ ,
validation accuracy $\boldsymbol{\beta}$,
         iterations per stage $< \mathbf{Z} = \mathbf{Z}_{1,} \mathbf{Z}_2, \mathbf{Z}_3, \dots, \mathbf{Z}_{\mathbf{Z}} >$, Total no. of stages X, Training data per stage $\mathbf{I}_{\mathbf{train}} = \mathbf{I}^{1}_{\mathbf{train}}, \mathbf{I}^{2}_{\mathbf{train}}, \dots\dots\dots$
$\mathbf{I}^{\mathbf{Z}}_{\mathbf{train}} >$, hyper
         parameters $\mathbf{h}_{1:k}$ , Real MRI ($\mathbf{I}_{\mathbf{R}}$), Synthetic MRI ($\mathbf{I}_{\mathbf{s}}$)
Output: hyper parameters ($\mathbf{h}^{*}$), Predicted labels (0-for HC, 1-for PD patients)
Begin
         learning_rate = initial_learning_rate // Set the initial learning rate for Adam
         beta1 = 0.9 // Adam parameter
```

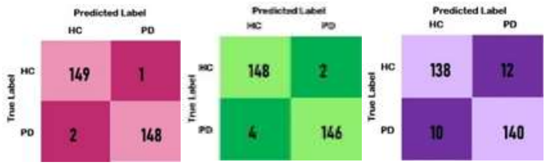
```
            beta2 = 0.999 // Adam parameter
            epsilon = $\frac{1}{10^8}$ // Adam parameter
            for 'i' iterations
                        for each $\mathbf{i} \in \mathbf{I_{train}}$
                              $\mathbf{DCNN} \leftarrow \mathbf{L_m(I)}$        // Initialization of any one of the DCNN models (EfficientNetB1, ResNet-50,
                                                                              ResNet152V2, MobileNetV2)
                              DCNN.fc.pop( )        // Removal of last fully connected layer
                              DCNN.dense(2, activation function)
                              for stage = 1 to X do
                                          for j=1 to k do
                                                $\boldsymbol{\beta_j}$ = evaluate $\boldsymbol{\beta(h_j, I^s_{train}, I_{val})}$
                                            end
                                            for i=k+1 to $Z_j$
                                                      g=grid-search ($h_j$, $\beta_i)_{i=1}^{j-i}$ ; $h_j$=max_args$_{h\in\alpha}$ a(h,g); $\beta_i$ = evaluate $\beta(h_i, I^s_{train}, I_{val})$;
                                               end for
                                 end for
                                 // Backpropagation with Adam
                                 loss = cross_entropy(I, I_labels) // Calculate the cross-entropy loss
                                 gradients = backpropagation(loss) // Compute gradients of the loss with respect to the model parameters
                                 m = initialize_1st_moment() // Initialize first moment vector
                                 v = initialize_2nd_moment() // Initialize second moment vector
                                 t = 0 // Time step counter
                                 for each parameter in model
                                          m = beta1 * m + (1 - beta1) * gradients // Update the first moment vector
                                          v = beta2 * v + (1 - beta2) * gradients^2 // Update the second moment vector
                                          t = t + 1
                                          m_hat = m / (1 - beta1^t) // Bias-corrected first moment estimate
                                          v_hat = v / (1 - beta2^t) // Bias-corrected second moment estimate
                                          parameter = parameter - learning_rate * m_hat / (sqrt(v_hat) + epsilon) // Update the model
parameters using Adam
                                 end for
                        end for
                end for
End Algorithm

validation:
for I iterations
      for I∈$I_{val}$
            loss←$\int_{cross\ entropy}$(R);
          updation of $R_m$ with Loss;
          DCNN $h_{1:k}$=best k configs ∈ ($h_{x1}$,$h_{x2}$,.....,$h_{xz}$)              //according to validation accuracy 'β'
      end for
      return    h* = max args h ∈ ($h_{x1}$,$h_{x2}$,.....,$h_{xz}$) $\beta_i$
end for
return h* // Return the hyperparameter configuration with the highest validation accuracy
```

Table 6. Grid search results over different values of hyper parameters.

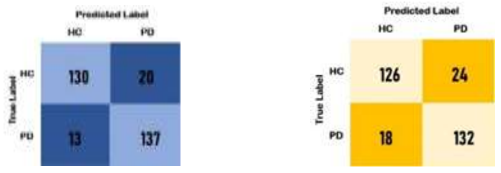
Optimizer			Adam	SGD	**Adam**	SGD	Adam	SGD	Adam	SGD
Learning rate			0.0001		**0.001**		0.01		0.1	
Loss (%)	EfficientNetB1	Training	0.35	0.45	**0.18**	0.3	0.32	0.48	0.4	0.5
Acc (%)			88.21	86.2	**98**	92	90.17	87.88	82.08	81.76
Loss (%)		Validation	0.38	0.47	**0.24**	0.4	0.45	0.53	0.51	0.58
Acc (%)			83.7	83.12	**93.33**	89.33	85.52	84.75	73.79	79.5
Loss (%)	ResNet-50	Training	0.35	0.46	**0.2**	0.32	0.33	0.48	0.43	0.52
Acc (%)			83.88	82.22	**92.6**	87.68	83.02	81.76	75.92	78.06
Loss (%)		Validation	0.42	0.5	**0.27**	0.43	0.49	0.55	0.52	0.6
Acc (%)			78.89	74.28	**82.33**	79.39	75.22	75.22	67.16	71.87
Loss (%)	ResNet152V2	Training	0.39	0.49	**0.23**	0.33	0.34	0.51	0.45	0.55
Acc (%)			78.97	79.56	**89**	85.54	80.89	80.33	73.01	74.17
Loss (%)		Validation	0.47	0.56	**0.33**	0.5	0.54	0.61	0.57	0.65
Acc (%)			69.69	66.46	**77.12**	73.42	70.68	68.19	62.71	63.83
Loss (%)	MobileNetV2	Training	0.42	0.52	**0.27**	0.35	0.36	0.55	0.48	0.58
Acc (%)			72.19	75.56	**86**	81.11	73.74	76.88	70.27	70.01
Loss (%)		Validation	0.55	0.63	**0.38**	0.58	0.6	0.68	0.63	0.72
Acc (%)			65.67	64.46	**73.33**	70.33	64.68	65.19	59.71	60.84

4. RESULTS

Appendixes, As a first step, the classifier is trained on the pre-trained networks. Next, transfer learning was used to improve the weights further, starting with backpropagation based on the most recent training datasets. The final model was applied for a total of 100 epochs, following the selection of the pertinent hyper-parameters. Both SGD and Adam were studied as optimization measures and the training was carried out for a total of 100 epochs. The parameters that gave the best performance of the model are a learning rate of 0.001, weight decay of 0.05, momentum factor of 0.8 and batch size of 64, with Adam being the best optimizer. After running through all of the earlier programmes, the model is saved with learned weights and is put through its paces during the testing procedure. It can be finally concluded that Adam is the best optimizer, with a learning rate of 0.001, batch size of 64 and total epochs of 100, to classify HC and PD patients. The confusion matrices can be observed from Fig. 11. Loss and Accuracy corresponding to SGD and Adam can be depicted from the Fig. 12(a), 12(b), 12(c) and 12(d) respectively. Table. 7 illustrates the ensemble model's performance as well as transfer learned DCNN models performance. The impact of data preprocessing and augmentation over the proposed DCNN models performance along with the computational complexity can also be observed.



(a)Ensemble method (b)fficientNetB1 (c) ResNet-50



(d) ResNet152V2 (e) MobileNetV2

Fig. 11. Confusion matrices of Ensemble, DCNN models.

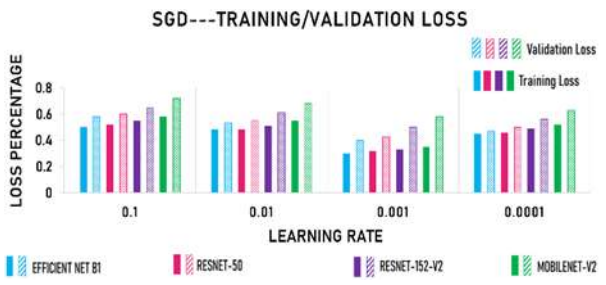


Fig. 12(a). Training / Validation loss using SGD

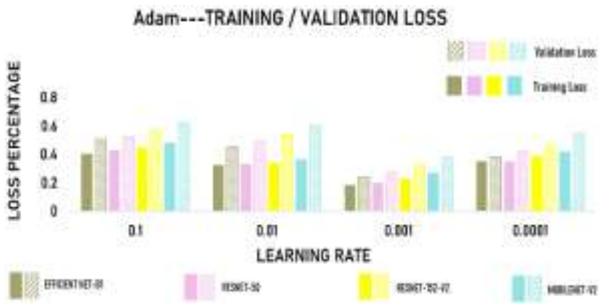


Fig. 12(b). Training / Validation loss using Adam

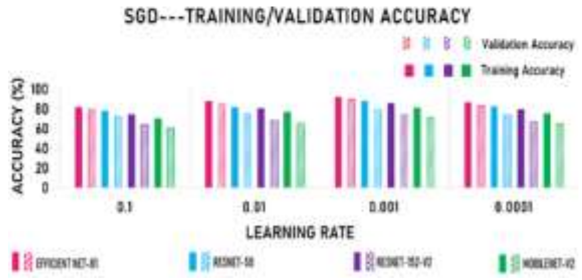


Fig. 12(c). Training / Validation accuracy using SGD

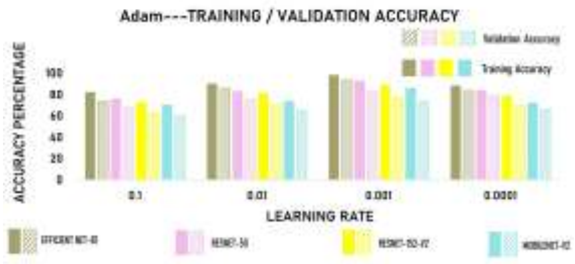


Fig. 12(d). Training / Validation accuracy using Adam

Matthews Correlation Coefficient (MCC) is a measure used to assess the quality of binary classifications. It is given by the formula:

$$\mathbf{MCC} = \frac{\mathbf{TP*TN-FP*FN}}{\sqrt{(\mathbf{TP+FP})(\mathbf{TP+FN})(\mathbf{TN+FP})(\mathbf{TN+FN})}} \quad (11)$$

Where TP is the number of True Positives, TN is the number of True Negatives, FP is the number of False Positives and FN is the number of False Negatives. The MCC value ranges from -1 to +1, where '+1' indicates a perfect prediction, '0' indicates no better than random prediction, and '-1' indicates total disagreement between prediction.

FPR stands for ‘False Positive Rate’. It is a metric that
represents the proportion of actual negative instances that are incorrectly predicted as positive by the model. It indicates how well the model identifies negative instances. A lower FPR indicates better performance in terms of minimizing false positives. It is calculated using the following formula:

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP+TN}} \qquad (12)$$

Where TN is the number of True Negatives and FP is the number of False Positives.

FNR stands for ‘False Negative Rate’. It is a metric that
represents the proportion of actual positive instances that are incorrectly predicted as negative by the model. It indicates how well the model identifies positive instances. A lower FNR indicates better performance in terms of minimizing false negatives. It is calculated using the following formula:

$$\mathbf{FNR} = \frac{\mathbf{FN}}{\mathbf{FN+TP}} \qquad (13)$$

Where FN is the number of False Negatives and TP is the number of True Positives.

Sensitivity is the proportion of actual positive instances that the model correctly predicts as positive. It quantifies the model's ability to "sensitively" detect or capture instances of the positive class. The values range from 0 to 1, where 1 indicates perfect sensitivity (all positive instances are correctly identified) and 0 indicates no sensitivity (no positive instances are correctly identified). It is calculated using the following formula:

$$\mathbf{Sensitivity} = \frac{\mathbf{TP}}{\mathbf{TP+FN}} \qquad (14)$$

Where FN is the number of False Negatives and TP is the number of True Positives

Specificity is the proportion of actual negative instances that the model correctly predicts as negative. It quantifies the model's ability to be "specific" in identifying instances of the negative class. The values range from 0 to 1, where 1 indicates perfect specificity (all negative instances are correctly identified as negative) and 0 indicates no specificity (no negative instances are correctly identified). It is calculated using the following formula:

$$\mathbf{Specificity} = \frac{\mathbf{TN}}{\mathbf{TN+FP}} \qquad (15)$$

Where FP is the number of False Positives and TN is the number of True Negatives

Balanced accuracy is a metric used to evaluate the performance of a classification model, especially when dealing with imbalanced datasets, where accuracy alone may be misleading. Balanced accuracy takes into account both sensitivity (true positive rate) and specificity (true negative rate) to provide a more comprehensive assessment. It ranges from 0 to 1, where 1 indicates perfect classification, and 0.5 indicates no better than random classification. The formula for balanced accuracy is:

$$\mathbf{Balanced\ Accuracy} = \frac{\mathbf{Sensitivity+Specificity}}{\mathbf{2}} \qquad (16)$$

Accuracy is the proportion of all instances that the model correctly predicts. i.e., It measures the overall correctness of a model by considering both true positives and true negatives. The values range from 0 to 1, where 1 indicates perfect accuracy (all predictions are correct), and 0 indicates no accuracy (all predictions are incorrect). It is calculated using the formula:

$$\mathbf{Accuracy} = \frac{\mathbf{TP+TN}}{\mathbf{total\ predictions}} \qquad (17)$$

Where TN is the number of True Negatives and TP is the number of True Positives.
Precision measures the accuracy of the positive predictions made by the model. A high precision indicates that the

model is making positive predictions with a low rate of false positives. It is calculated as:

$$\mathbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP+FP}} \qquad (18)$$

Where PN is the number of True Positives and FP is the number of False Positives.

Recall measures the ability of the model to capture all the positive instances. A high recall indicates that the model is effectively identifying a large proportion of the positive instances. It is calculated as:

$$\mathbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP+FN}} \qquad (19)$$

Where FN is the number of False Negatives and TP is the number of True Positives.

F-score is a metric that combines precision and recall into a single value, by taking their harmonic mean. The formula for the F1 score is:

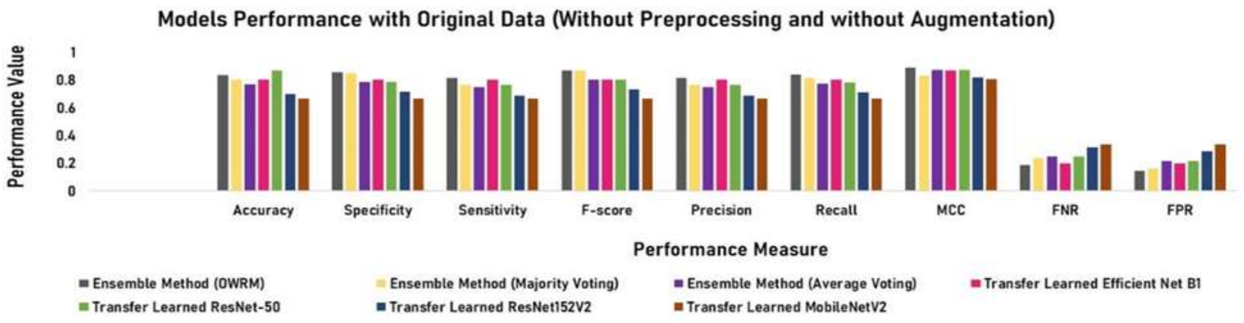
$$\mathbf{F-score} = \frac{\mathbf{2(precision)(recall)}}{\mathbf{precision + recall}} \qquad (20)$$

To determine the possibility that an image will be classified as either HC or PD, each image was examined while incorporating the final ensemble model into account. The impact of data preprocessing and augmentation on the performance of the proposed models can be observed from Fig. 13. When assessing the network's effectiveness, the Receiver Operating Characteristic (ROC) is used. Fig. 14 shows that the AUC calculated from the ROC curve has an estimated value of 0.9861, 0.9524, 0.7986, 0.7361 and 0.7083 for Ensemble Method, EfficientNet B1, ResNet-50, ResNet152V2 and MobileNetV2 respectively. This shows that Ensemble method results in a satisfying result of the trained model. The training and validation accuracies as well as losses, can be depicted from Fig. 15 and Fig. 16 respectively. Table 8 illustrates the proposed model's performance in comparison with state-of-the-art approaches. Clearly, the ensemble model outperforms all other published research works, confirming that the innovative deep-learning approach that was developed, has its advantages and is competitive. The purpose of this work is to suggest a system that uses ensemble learning using deep learning models to build imaging biomarkers that can predict early PD. Evaluating the efficiency of a deep learning model involves not only performance metrics but also the interpretability of its predictions, particularly in critical domains such as healthcare. The ability to explain the rationale behind specific predictions is predominant, in the context of implementing intelligent systems. For the interpretation of predictions generated by the developed DCNN model, A Gradient weighted Class Activation Mapping (Grad-CAM) was employed [60, 61]. It is an extended version of CAM that calculates the gradient directly using the backpropagation from each neuron in the last convolution layer feature map.

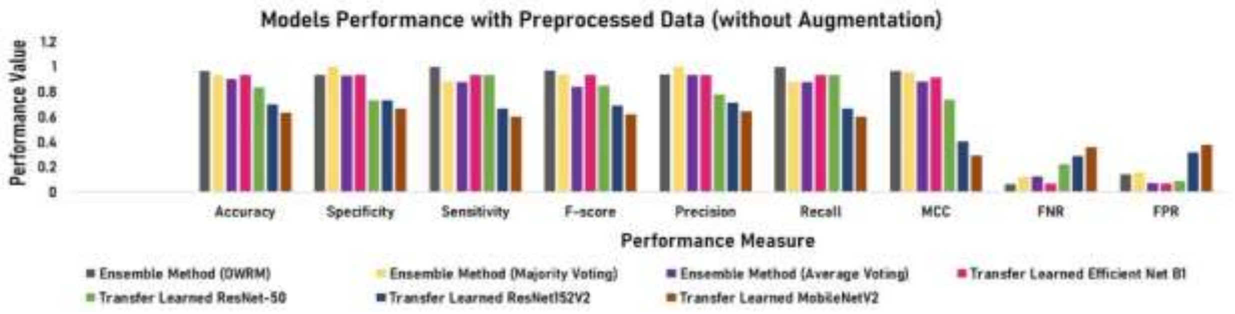
Table 7. Preprocessing and augmentation effect on DCNN and Ensemble models Performance.

	Performance Measure	**Ensemble Method**			**Transfer Learned Efficient Net B1**	**Computational Complexity (minutes)**	**Transfer Learned ResNet-50**	**Computational Complexity (minutes)**	**Transfer Learned ResNet152V2**	**Computational Complexity (minutes)**	**Transfer Learned MobileNetV2**	**Computational Complexity (minutes)**
		OWPM (Proposed Method)	**Majority Voting**	**Average Voting**								
Original data without preprocessing and without augmentation	Accuracy (%)	83.33	80	76.67	80.0	3.71 to 3.77	86.67	3.02 to 3.06	70.0	3.92 to 3.95	66.67	3.41 to 3.45
	Specificity (%)	85.71	84.62	78.57	80.0		78.57		71.43		66.67	
	Sensitivity (%)	81.25	76.47	75.0	80.0		76.47		68.75		66.67	
	Balanced Accuracy (%)	83.48	80.545	76.785	80		77.52		70.09		66.67	
	F-score (%)	86.67	86.67	80.0	80.0		80.0		73.33		66.67	
	Precision (%)	81.25	76.47	75.0	80.0		76.47		68.75		66.67	
	Recall (%)	83.83	81.24	77.42	80.0		78.29		70.99		66.67	

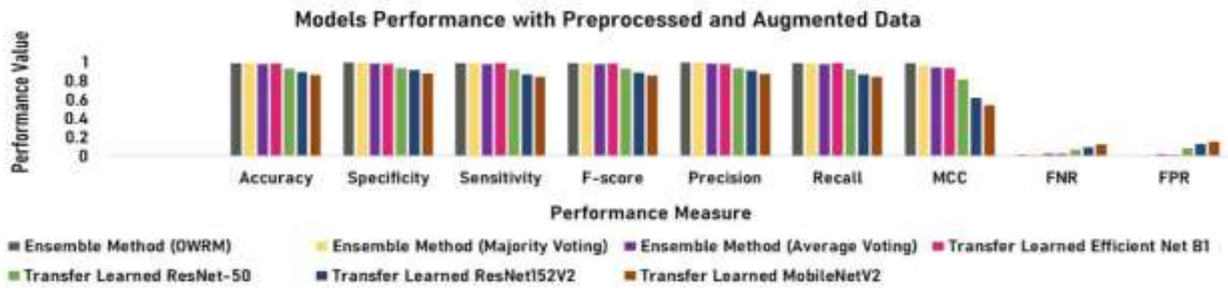
	FPR (%)	14.29	15.98	21.43	20.0		21.43		28.57		33.33	
	FNR (%)	18.75	23.53	25.0	20.0		25.0		31.25		33.33	
	MCC (%)	88.81	83.13	87.11	86.94		87.11		81.87		80.62	
Data pre-processed but not augmented	Accuracy (%)	96.66	93.33	90.0	93.33	1.93 to 1.97	83.33	1.36 to 1.4	70.0	1.86 to 1.91	63.33	1.64 to 1.67
	Specificity (%)	93.35	100	92.85	93.33		73.33		73.33		66.66	
	Sensitivity (%)	100	88.23	87.5	93.33		93.33		66.66		60.0	
	Balanced Accuracy (%)	96.67	94.11	90.17	93.33		83.33		69.995		63.33	
	F-score (%)	96.96	93.74	83.87	93.27		84.84		68.95		62.06	
	Precision (%)	93.75	100.0	93.33	93.33		77.77		71.42		64.28	
	Recall (%)	100	88.23	87.5	93.33		93.33		66.66		60.0	
	FPR (%)	14.29	15.38	07.14	06.67		08.33		31.25		37.5	
	FNR (%)	06.25	11.76	12.5	06.67		22.22		28.57		35.71	
	MCC (%)	96.82	95.16	88.39	91.36		73.51		40.54		29.37	
Pre-processed and augmented data	Accuracy (%)	99.0	98.33	97.6	98.0	311.35	92.6	220.29	89.0	299.38	86.0	263.36
	Specificity (%)	99.31	98.65	97.98	97.33		93.33		91.33		88.0	
	Sensitivity (%)	98.67	98.01	97.35	98.66		92.0		86.66		84.0	
	Balanced Accuracy (%)	98.99	98.33	97.66	97.99		92.665		88.99		86	
	F-score (%)	98.99	98.33	97.67	98.0		92.61		88.72		85.71	
	Precision (%)	99.33	98.66	98.0	97.36		93.24		90.9		87.5	
	Recall (%)	98.67	98.01	97.35	98.66		92.0		86.66		84.0	
	FPR (%)	00.66	01.33	02.02	01.33		07.89		12.74		15.38	
	FNR (%)	01.32	01.98	02.64	02.63		06.75		09.09		12.5	
	MCC (%)	98.64	95.6	93.72	93.21		81.12		61.93		54.05	



(a) Without data preprocessing and without augmentation



(b) with data preprocessing and without augmentation



(c) with data preprocessing and with augmentation

Fig. 13. Impact of data preprocessing and augmentation.

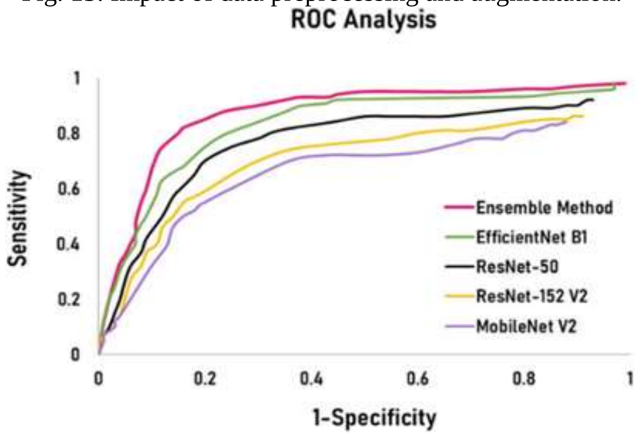


Fig. 14. ROC curve

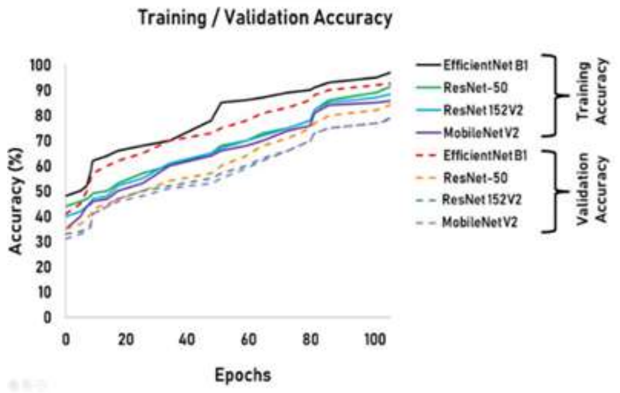


Fig. 15. Accuracy during training / Validation

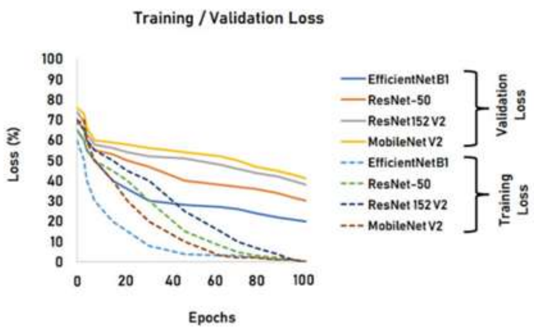


Fig. 16. Loss during training / Validation

CAM shows the score of importance of each part of the input by resizing 2D map that predicts class ‘C’ to the original input image. Then, it adds up these gradients within the ith feature map, to generate the weight of each map and predict class ‘C’. The Grad-CAM of class ‘C’ can be generated using the eq (24).

$$\mathbf{L^{C}_{Grad-CAM}} = \mathbf{ReLU}\left(\sum_{i} \mathbf{wt^{c}_{i} GAP^{i}}\right) \qquad (21)$$

Where $\mathbf{GAP^{i}}$ is the Global Average Pooling of $\mathbf{i^{th}}$ feature map and ‘**wt**’ is the weight of each map. The ReLU function is employed to eliminate negative contribution scores. This is because Grad-CAM focuses on considering only the input features that positively impact the prediction score for class 'C'. Grad-CAM can be applied to interpret any type of DCNN without making changes to the DCNN model, as it directly uses gradients from backpropagation. The flow of action can be observed from Fig. 17 [22]. From Fig. 18(b), it can be observed that the CAM highlights the regions of interest in MRI slices predicted as Parkinson’s Disease. Notably, the model demonstrates an enhanced focus on the Substantia Nigra pars compacta (SNc), a region significantly impacted by the loss of dopaminergic neurons. This interpretative approach, utilizing the Grad-CAM technique, enhances our understanding of the model's decision-making process.

Table 8. state-of-the-art-works comparison.

Reference	Methodology	Acc (%)
[10]	SVM classifier	85.8
Proposed	TL MobileNetV2 + data augmentation	86
Proposed	TL ResNet152V2 + data augmentation	89
[20]	TL AlexNet + data augmentation	89.23
[11]	Histogram Equalizer with PCA, SVM	92.6
Proposed	TL ResNet-50 + data augmentation	92.6
[18]	CNN + Bayesian SMBO optimization	95.29
Proposed	**TL EfficientNet B1 + data augmentation**	**98**
Proposed	**Ensemble Learning**	**99**
*TL-Transfer Learned; Acc-Accuracy		

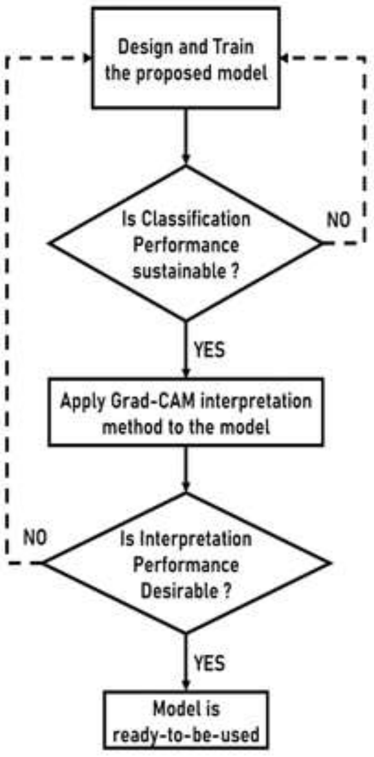

Design and Train the proposed model
Is Classification Performance sustainable ?
NO
YES
Apply Grad-CAM interpretation method to the model
NO
Is Interpretation Performance Desirable ?
YES
Model is ready-to-be-used

Fig. 17. Flowchart of Grad-CAM interpretation method.

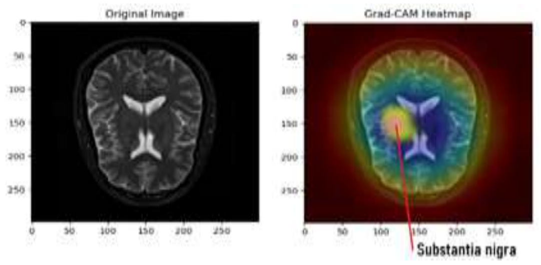


Fig. 18 (a). MR Image predicted as PD Fig. 18(b). Grad-CAM Heatmap predicted as PD.

5. DISCUSSION

In Results section, we demonstrated the performances of 4 DCNN models for binary classification based on the images obtained from PPMI and LSSH, and also compared them against the efficiency obtained using Ensemble learning in Table 6. According to this study, the training period for the network was on an average of five hours long and consisted of 100 epochs, which results in high computational complexity. The model achieved an accuracy of 99% and a balanced accuracy of 98.99%, which proves its high performance. Moreover, the interpretability model using Grad-CAM has proven its trust worthiness, by concentrating on the Substantia Nigra region to identify whether the image belongs to PD or HC category. Table 6 shows that the developed ensemble method using OWPM approach yields the best performance over all other models. Because the model achieved a greater level of classification accuracy when it was applied to the test set, it is clear that the model did not suffer from overfitting when it was applied to the training set. It is important to point out that the incorporation of synthetic images results in a significant reduction of the standard deviation of the outcomes in the validation of accuracy, sensitivity, specificity, as well as the Area Under the Curve (AUC). When compared to the results of classification carried out using only the original images, the ensemble model designed from preprocessed and augmented data, attains a greater accuracy of up to 15.67%. The early findings using various existing methodologies provided in Table 1 indicate an accuracy of 40%–90%, while the method developed offers an accuracy of 99%. MCC is used to assess the model's quality and having achieved 98.64% MCC, we can claim that our model stands as one of the best proposed models, in diagnosing early PD. Despite the high computational complexity, our model recommended for clinical use with its outstanding features.

6. CONCLUSION

PD ranks as the second most prevalent neurological disorder, next to Alzheimer's disease. In this study, the challenge of accurately classifying brain MRI as either PD or HC, is investigated by developing a comprehensive framework that combines data augmentation, transfer learning, ensemble learning and DCNN models. Leveraging the custom dataset created from the data obtained from LSSH, a comparative analysis was conducted against the clinical standard, which involves visual interpretation and quantification by experts besides facilitating real-time clinical comparisons in the PD diagnosis process. Remarkably, the proposed ensemble approach using OWPM method achieved exceptional accuracy of 99%, average recall of 0.986, average precision of 0.993, average specificity of 0.9931, f1-score of 0.9899, showcasing its potential as a reliable tool for discriminating PD from HCs, surpassing all the prominent architectures including EfficientNet B1, ResNet-50, ResNet152V2, and MobileNetV2. Impressively, the ensemble model yielded substantial improvement of 9.77%, compared to the state-of-the-art models. This remarkable advancement highlights the process of the approach in diagnosing the PD accurately, based on whole brain MRI. Furthermore, the ensemble model exhibited high differential fluency, as evidenced by an impressive AUC value of 0.9867 according to the ROC curve. This demonstrates the capability of the model to effectively distinguish between PD and HC cases. Moreover, the interpretation of the model over the MRI scans was also evaluated using Grad-CAM approach, and it was found that the model paid maximum attention to the substantia nigra region for predicting a particular MRI scan as PD. Nevertheless, this study has certain limitations that must be acknowledged. Specifically, patient characteristics such as different age groups, locations, dietary habits, and gender were not taken into consideration, which could impact the classification performance. Another limitation is being unable to identify the disease stage of a PD patient, as per the H&Y scale. A single clinical metric may not be sufficient to accurately track PD progression. Therefore, a combination of metrics or global statistical tests must be utilized (Huang et al., 2009). The training data set considered in this study suffers from class imbalance problem (25 HC and 45 PD). Efficient results in limited time may be possible, if data balance can be maintained. It is highly recommended to perform the research by considering specific subcortical structures for the detection of PD. Future research endeavors should aim to incorporate the discussed factors, in addition to the development of imaging biomarkers for PD and other related conditions. This study serves as a significant step forward, providing valuable insights for researchers and clinicians alike in the realm of clinical image evaluation. The objective identification of PD will become increasingly accessible and reliable for clinicians in the near future, with the ongoing advancements in deep learning techniques.

Data Availability. Data used in this study is available from PPMI public database and can be downloaded from the web site: www.ppmi-info.org/data.

Funding. There is no funding for this work.

Conflicts of interest. The authors declare no conflict of interest.

Ethics. The authors declare that the present research work has fulfilled all relevant ethical guidelines required by COPE.

REFERENCES

[1] W. Poewe, K. Seppi, et al., "**Parkinson Disease**," *Nat Rev Dis Primers*, vol. 3, p. 17013, 2017.
[2] K. R. Chaudhuri, A. H. Schapira, et al., "**non-motor symptoms of Parkinson's disease: Dopaminergic pathophysiology and treatment**," *Lancet Neurol*, vol. 8, no. 5, pp. 464-474, 2009.
[3] J. Xu, M. Zhang, "Use of magnetic resonance imaging and artificial intelligence in studies of diagnosis of PD," *ACS Chem Neurosci*, vol. 10, no. 1, pp. 445-449, 2019.
[4] S. Soltaninejad, I. Cheng, A. Basu, "**Towards the identification of PD using only T1 MR images**," *ArXiv*, abs/1806.07489, 2018.
[5] A. Naseer, M. Rani, "Refining Parkinson's neurological disorder identification through deep transfer learning," *Neural Comput Appl*, vol. 31, no. 5, pp. 1751-1762, 2019.
[6] E. Adeli, F. Shi, "Joint Feature-Sample Selection and Robust Diagnosis of Parkinson's Disease from MRI Data," *NeuroImage*, vol. 141, pp. 206-219, 2016.
[7] N. Amoroso, M. L. Rocca, A. Monaco, R. Bellotti, S. Tangaro, "**Complex networks reveal early MRI markers of Parkinson's disease**," *Med. Image Anal*, vol. 48, 12-24, 2018.

[8] S. Sivaranjini, C. M. Sujatha, "Deep learning-based diagnosis of Parkinson's disease using a convolutional neural network," *Multimed Tools Appl*, vol. 78, no. 1, 2019.

[9] H. Lei, Y. Zhao, Y. Wen, Q. Luo, Y. Cai, G. Liu, B. Lei, "**Sparse feature learning for multi-class Parkinson's disease classification**," *Technol. Health Care*, vol. 26, no. 1, pp. 193-203, 2018.

[10] C. Salvatore, A. Cerasa, I. Castiglioni, F. Gallivanone, A. Augimeri, M. Lopez, G. Arabia, M. Morelli, M. C. Gilardi, A. Quattrone, "**Machine learning on brain MRI data for differential diagnosis of Parkinson's disease and Progressive Supranuclear Palsy**," *J. Neurosci. Methods*, vol. 222, pp. 230-237, 2014.

[11] A. Brahim, L. Khedher, J.M. Górriz, J. Ramírez, H. Toumi, E. Lespessailles, R. Jennane, M. El Hassouni, "**A proposed computer-aided diagnosis system for Parkinson's disease classification using 123I-FP-CIT imaging**," *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2017, pp. 1-6.

[12] J. Quan, L. Xu, R. Xu, T. Tong, J. Su, "DaTscan SPECT Image Classification for Parkinson's Disease," *arXiv*, 2019.

[13] A. Rahmim, P. Huang, N. Shenkov, S. Fotouhi, E. Davoodi-Bojd, L. Lu, Z. Mari, H. Soltanian-Zadeh, V. Sossi, "**Improved prediction of outcome in Parkinson's disease using radiomics analysis of longitudinal DAT SPECT images**," *NeuroImage: Clinical*, vol. 16, pp. 539-544, 2017.

[14] H. Choi, S. Ha, H. J. Im, S. H. Paek, D. S. Lee, "Refining diagnosis of Parkinson's disease with deep learning-based interpretation of dopamine transporter imaging," *NeuroImage: Clinical*, vol. 16, pp. 586-594, 2017.

[15] R. Prashanth, S. Dutta Roy, P. K. Mandal, S. Ghosh, "**High-accuracy classification of Parkinson's disease through shape analysis and surface fitting in 123I-Ioflupane SPECT imaging**," *IEEE journal of biomedical and health informatics*, vol. 21, pp. 794-802, 2017.

[16] M. Rumman, A. N. Tasneem, S. Farzana, M. I. Pavel, M. A. Alam, "**Early detection of Parkinson's disease using image processing and artificial neural network**," *7th International Conference on Informatics, Electronics & Vision and 2nd International Conference on Imaging, Vision & Pattern Recognition*, *IEEE*, 2018, pp. 256-261.

[17] A. Ortiz, J. Munilla, M. Martínez-Ibañez, J. M. Górriz, J. Ramírez, D. Salas-Gonzalez, "**Parkinson's disease detection using iso surfaces-based features and convolutional neural networks**," *Front. in neuro infor*, vol. 13, 2019.

[18] S. Chakraborty, S. Aich, H. C. Kim, "Detection of Parkinson's disease from 3T T1 weighted MRI scans using 3D convolutional neural network," *Diagnostics*, vol. 10, no. 6, p. 402, 2020.

[19] P. R. Magesh, R. D. Myloth, R. J. Tom, "An explainable machine learning model for early detection of Parkinson's disease using lime on datscan imagery," *Comput. Biol. Med*., vol. 126, p. 104041, 2020.

[20] S. Kaur, H. Aggarwal, R. Rani, "**Diagnosis of Parkinson's disease using deep CNN with transfer learning and data augmentation**," *Multimedia Tools and Appl*, vol. 80, no. 7, pp. 10113-10139, 2021.

[21] Farhan Mohammed, Xiangjian He, Yiguang Lin, "An easy-to-use deep-learning model for highly accurate diagnosis of Parkinson's disease using SPECT images," *Comput Med Imaging Graph*, 2021, 101810.

[22] Theerasarn Pianpanit, Sermkiat Lolak, Phattarapong Sawangjai, Thapanun Sudhawiyangkul, Theerawit Wilaiprasitporn, "**Parkinson's disease recognition using SPECT image and interpretable AI: A tutorial**," *IEEE Sensors Journal*, vol. 21, no. 20, pp. 22304-22316, 2021.

[23] Chung-Yao Chien, Szu-Wei Hsu, Tsung-Lin Lee, Pi-Shan Sung, Chou-Ching Lin, "Using artificial neural network to discriminate Parkinson's disease from Parkinsonisms by focusing on putamen of dopamine transporter SPECT images," *Biomedicines*, vol.9, no.1, 2020.

[24] S Dhanalakshmi, M Thakur, Kuresan Harisudha, K Wee Lai, Xiang Wu, "**Soft Attention Based DenseNet Model for Parkinson's Disease Classification Using SPECT Images**," *Frontiers in Aging Neu.sci*, vol. 532, 2022.

[25] S. Liu, G. Tian, Y. Xu, "A different scene classification model is combining Res-Net based transfer learning and data augmentation with a filte," *Neuro Comput J*, vol. 338, pp. 191–206, 2019.

[26] L. Shao, "**Transfer learning for visual categorization: a survey**," *IEEE Trans Neural Netw Learn Syst*, vol. 26, no. 5, pp. 1019–34, 2015.

[27] Radford, Alec, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint*, 2015, arXiv:1511.06434.

[28] M. Y. Shams, O. M. Elzeki, Mohamed Abd Elfattah, T. Medhat, Aboul Ella Hassanien, "**Why are GANs vital for deep neural networks? A case study on COVID-19 chest X-ray images**," *Springer*, 2020, pp. 147-162.

[29] Archana G, Afzal Hussain S, "A comparative study on performance of basic and ensemble classifiers with various datasets," *Applied Computer Science*, vol.19, 2023.

[30] C.Ledig, L.Theis, F.Huszár, J.Caballero, A.Cunningham, A.Acosta, "**Photo-realistic image super-resolution using a generative adversarial network**," *ArXiv Prepr*, 2016.

[31] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, "**Image-to-image translation with conditional adversarial networks**," *ArXiv Prepr*, 2017.

[32] H.Wu, S.Zheng, J.Zhang, K.Huang, "Gp-gan: Towards realistic high-resolution image blending," *ArXiv Prepr*, 2017.

[33] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, M. N. Do, "**Semantic image inpainting with deep generative models**," *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5485–5493.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, Y. Bengio, "**Generative adversarial nets**," *Advances in neural information processing systems*, 2014.

[35] T. Iqbal, H. Ali, "Generative adversarial network for medical images (MI-GAN)," *J Med Syst*, vol. 42, 2018.

[36] A. Mikołajczyk, M. Grochowski, "**Data augmentation for improving deep learning in image classification problem**," *2018 international interdisciplinary PhD workshop (IIPhDW), IEEE*, 2018.

[37] Y. Weng, H. Zhou, "Data augmentation computing model based on generative adversarial network," *IEEE Access*, 2019, 2917207.

[38] Y. J. Yoo, "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches," *Knowl Based Syst*, vol. 178, pp. 74–83, 2019.

[39] S. Kaur, H. Aggarwal, R. Rani, "**Diagnosis of PD using principal component analysis and deep learning**," *J Med Imaging Health Inf*, vol. 9, no. 3, pp. 602–609, 2019.

[40] E. Nazari, M. Aghemiri, A. Avan, A. Mehrabian, "Machine learning approaches for classification of colorectal cancer with and without feature selection method on microarray data," *Gene Reports*, vol. 25, p. 101419, 2021.

[41] S. Tewari, U. D. Dwivedi, "**A comparative study of heterogeneous ensemble methods for the identification of geological lithofacies**," *Journal of Petroleum Exploration and Production Technology*, vol. 10, 2020.

[42] D. Castillo-Barnes, J. Ramírez, F. Segovia, F. Martínez-Murcia, D. Salas-Gonzalez, J. Górriz, "Robust ensemble classification methodology for i123-ioflupane spect images and multiple heterogeneous biomarkers in the diagnosis of parkinson's disease," *Front. Neuroinformatics*, 2018.

[43] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint*, 1609.04747, 2016.

[44] R. Keys, "**Cubic convolution interpolation for digital image processing**," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153-1160, 1981.

[45] N. Kumar, M. Nachamai, "**Noise Removal and Filtering Techniques Used in Medical Images**," *Orient.J. Comp. Sci. and Technol*, vol. 10, no. 1, 2017.

[46] D. P. Kingma, J. Ba, "**Adam: A Method for Stochastic Optimization**," *International Conference on Learning Representations, California*, 2015.

[47] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht, "**The marginal value of adaptive gradient methods in machine learning**," *Advances in neural information processing systems*, vol. 30, 2017.

[48] L. Wang, Q. Zhang, H. Li, H. Zhang, "**SPECT molecular imaging in Parkinson's disease**", *BioMed Research International*, 2012.

[49] D. Huang, B. Sherman, R. Lempicki, "Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources", *Nat Protoc*, vol. 4, 44–57, 2009.

[50] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, H. Greenspan, "**GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification**", *Neurocomputing*, vol. 321, 2018.

[51] E. H. Houssein, Z. Abohashima, Elhoseny Mohamed, W. M. Mohamed, "**Hybrid quantum-classical convolutional neural network model for COVID-19 prediction using chest X-ray images**", *Journal of Computational Design and Engineering*, 2022, vol. 9, pp. 343–363.

[52] Zang Cho, "Review of Recent Advancement of Ultra High Field Magnetic Resonance Imaging: from Anatomy to Tractography", *Investigative Magnetic Resonance Imaging*, 2016, vol. 20, pp. 141.

[53] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. F. Beckmann, T. E. J. Behrens, H. Johansen-Berg, "**Advances in functional and structural MR image analysis and implementation as FSL**", *Neuroimage*, vol. 23, 2004.

[54] M. Jenkinson, C. F. Beckmann, T. E. Behrens, M. W. Woolrich, "**FSL**", *Neuroimage*, vol. 62, pp. 782–90, 2012.

[55] G.Grabner, A.L.Janke, M.M.Budge, D.Smith, J. Pruessner, "**Symmetric atlasing and model-based segmentation: an application to the hippocampus in older adults**," *Med Image Comput Comput Assist Interv*., 2006, vol. 9.

[56] https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758/

[57] W. Li, J. Hou, L. Yin, "**A classifier fusion method based on classifier accuracy**," *2014 International Conference on Mechatronics and Control (ICMC), Jinzhou, China*, pp. 2119-2122, d2016, oi: 10.1109/ICMC57404.2022.00020.

[58] Rokach, L., "**Ensemble Learning**", *World Scientific Publishing Co Pte Ltd*, 2019, 300.

[59] Dietterich, T. G., "**Ensemble methods in machine learning,"** *International workshop on multiple classifier systems*, pp. 1-15: Springer Berlin Heidelberg, 2000.

[60] Al-Khasawneh, M. A., A. Alzahrani, A. Alarood, "**Alzheimer's Disease Diagnosis Using MRI Images**," In *Data Analysis for Neurodegenerative Disorders*, pp. 195-212, Springer Nature Singapore, 2023.

[61] S. Wang and Y. Zhang, **"Grad-CAM: Understanding AI Models,"** *Comput. Mater. Contin*., vol.76, 2023.

[62] A. R. Palakayala and K. P, "**Survey of Parkinson's Disease Detection using Different Symptoms**," in *Algorithms, Computing and Mathematics Conference (ACM), Chennai*, India, 2022, pp. 77-82, doi: 10.1109/ACM57404.2022.00020.

[63] Aditi Govindu, Sushila Palwe, "Early detection of Parkinson's disease using machine learning.

APPENDIX

Algorithm-2: Generative Adversarial Network using ADAM optimizer

```
1.  Input: Real MRI of brain (Real_Imgs), Random noise value ('Z',0.0<=Z<=1.0),
    Generator Layers (Gen_Layers), Discriminator Layers (Dis_Layers), hyper parameters (hyp_param)
2.  Output: GAN
3.  Begin
    3.1   Model=setup(hyp_param)
          //Generator and Discriminator construction
    3.2       Gen_model=build_net(Gen_Layers);
    3.3       Dis_model=build_net(Dis_Layers);
```

```
            3.3.1 for x= 1 to model.Epoch
            3.3.2     train_img=random(num(Real_Imgs))
            3.3.3     for i= 1 to model.tot_batches
            3.3.4         tot_imgs=read(Real_Imgs, train_img);
            3.3.5         Gen_Model ⟵ train(tot_imgs, Z)
            3.3.6         Gen_Model ⟵ adamopt (Gen_Model, Dis_Model, imitation_Imgs)
            3.3.7         Dis_Model ⟵ train(tot_imgs, Z, imitation_Imgs);
            3.3.8         Dis_Model ⟵ adamopt (Gen_Model, Dis_Model, imitation_Imgs)
            3.3.9     End for
            3.3.10    End for
      3.4   GAN ⟵ {Gen_Model, Dis_Model}
  4.  End
# Algorithm adamopt
    Repeat:
        update 't', t:=t+1
        $g_t$=grad($\theta_{t-1}$)                                   // Obtain the gradients/derivatives 'g' w.r.t. 't
        $m_t=\beta_1 . m_{t-1} + (1-\beta_1) . g_t; v_t=\beta_2 . v_{t-1} + (1-\beta_2). g_t^2$   // Update the first moment '$m_t$' , second moment '$v_t$'
        $\widehat{m}_t = \frac{m_t}{(1-\beta_1^t)}$;   $\widehat{v}_t = \frac{v_t}{(1-\beta_2^t)}$          // Compute the bias corrected $m_t$ and bias corrected $v_t$
        $\theta_t = \theta_{t-1} - \alpha\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$                       // Update the parameter 'θ'
    until $\theta_t$ do not converge.
# Algorithm compute_generator_loss
    Function compute_generator_loss(Dis_Model, Fake_Imgs)
        Fake_predictions = Dis_Model.predict(Fake_Imgs)          // Compute the discriminator's prediction
for the fake images
        Generator_loss = compute_loss(Fake_predictions, target_labels='real')   // Calculate generator loss using the
predicted labels
        return Generator_loss
    End
# Algorithm compute_discriminator_loss
    Function compute_discriminator_loss(Dis_Model, Real_Imgs, Fake_Imgs)
        Real_predictions = Dis_Model.predict(Real_Imgs)
        Fake_predictions = Dis_Model.predict(Fake_Imgs)          //Compute discriminator's predictions for
real and fake images
        Real_loss = compute_loss(Real_predictions, target_labels='real')
        Fake_loss = compute_loss(Fake_predictions, target_labels='fake')   //Calculate the discriminator loss using the
predicted labels
        Discriminator_loss = Real_loss + Fake_loss               // Combine the losses to compute the overall
discriminator loss
        return Discriminator_loss
    End
4. End of the algorithm
```