PARKINSON'S, ALZHEIMER'S DISEASE DIAGNOSIS USING FDG-PET IMAGES WITH NEURAL NETWORKS

Alok Yathiraj

S4128427

Dept. of Biomedical Engineering

University of Groningen

The Netherlands

Period: 08/04/2020 - 31/08/2020

Internship

Supervisor: Rick van Veen, PhD student, University of Groningen

Mentor: Marcel Greuter, Faculty of Medical Sciences, UMCG

Contents

Abstract:

Parkinson's (PD) and Alzheimer's (AD) disease are two of the most common neurological diseases worldwide. PD is caused due to the death of dopamine-producing neurons in the midbrain leading to motor disabilities as dopamine is an important neurotransmitter that controls voluntary and unconscious movement. Whereas AD is caused by the buildup of abnormal proteins in and around the brain cells, this then leads to the decrease of neurotransmitters which send and receive signals between brain cells.

The diagnosis of these two diseases is especially hard, currently there are no standard tests or biomarkers in use for detection at early stages. In the case of PD, physicians make a subjective score based on visual observations. For AD the diagnosis is based on cognitive tests. The diagnosis for both these diseases are subjective, making them unreliable as results could vary clinic to clinic. With each of these diseases having different treatment plans it is imperative to diagnose them accurately at an early stage. It has been found that some areas of the brain degenerate and others become more active for each of these neurological diseases, this change in activity can be visualised using [^{18}F]-fluoro-deoxyglucose positron emission tomography (FDG-PET) which depicts the metabolic activity of the brain. It has also been found that using Scaled Subprofile Modelling/Principal Component Analysis (SSM/PCA) on the FDG-PET images helps by converting the data into a space where disease specific patterns of covariation in the brain can be seen easily. This makes it easier to distinguish between different patterns seen in the FDG-PET images.

The goal of this paper is to analyse the effectiveness of using feed forward neural networks and the influence of using two different activation functions to classify AD from PD patients, using FDG-PET images that have been pre-processed using SSM/PCA as the input features. Additionally, performance of the network to classify AD from healthy controls (HC) and PD from HC is also tested with the parameters of each network chosen such that the highest accuracy can be achieved.

1. Introduction

It is estimated that around 50 million people have Alzheimer's or related dementia [1] and more than 10 million people have Parkinson's worldwide [2]. Additionally, only one in four people with AD are diagnosed and only 4% of people with PD are diagnosed with PD before the age of 50.

The symptoms seen during the early stages of AD include memory impairment, having problems with language and executing complex tasks. As the disease progresses, patients struggle with simple daily tasks as well. The symptoms seen during the early stages of PD include cognitive impairment, problems with motor function such as tremors at rest, shuffling gait. Later stages of PD can lead to the development of dementia as well [3]. These two neurological disorders have overlapping symptoms, especially in the early stages, which makes it hard to diagnose the right disease. It is important to differentiate these diseases as it helps choose the right form of treatment. Currently, the diagnosis of these diseases are subjective and based on the clinical symptoms shown by the patient.

[^{18}F]-fluoro-deoxyglucose positron emission tomography (FDG-PET) is used to obtain three dimensional images of the brain, which depicts the metabolic activity in different regions. Making use of a combination of FDG-PET images and machine learning not only helps distinguish and diagnose PD and AD at an earlier stage but also reduces the subjectivity of the diagnosis, making a more universal diagnosis. Previous research in the field uses Support Vector Machine (SVM), a family of machine leaning algorithms for the purpose of classification. It is known to be accurate in high dimensional spaces, i.e. it works by mapping non-linear data to a high dimensional space where the data is linearly separable. It was found that the SVM algorithm gives a mean accuracy of 94.5% for the classification of AD patients from healthy controls using three dimensional T1 weighted MR images [4]. Other research using SVM to classify PD patients from control subjects using MR images found accuracy of up to 97.5% [5]. Other research uses Learning Vector Quantization (LVQ) as the machine learning algorithm for the binary classification of Control subjects, AD and PD patients using data from FDG PET scans. It was found that LVQ shows an average accuracy of 95% when classifying PD and AD subjects using data from within a study centre [6]. LVQ creates an interpretable model, allowing visual interpretation of what the algorithm has learnt in the original voxel/brain space, hence making it more user friendly [7].

This project focusses on using Scaled Subprofile Modelling/Principal Component Analysis (SSM/PCA) for feature extraction from FDG PET images of the brain. Research shows that the SSM/PCA method can construct specific covariance patterns per disease. Of which the expression of these patterns can be calculated in new/unseen patients. [8]. The expression scores obtained from SSM/PCA are then used as input features for a feedforward neural network for the binary classification of AD, PD, and control subjects (HC).

2. Data

For this study [^{18}F]-fluoro deoxyglucose - positron emission tomography (FDG-PET) images of the brain are used. FDG is a radioactive tracer which can be used to measure the metabolic activity of the brain, as it follows the same path as the only source of energy for the brain would, i.e., glucose. Functionally, FDG acts the same as glucose, except it emits a positron when consumed, which can be detected by the PET scanner. In the case of neurodegenerative brain diseases, certain areas of the brain degenerate which causes patterns of metabolic activity displayed on the FDG PET images. The data used includes the FDG PET images of Parkinson's and Alzheimer's patients, along with healthy control subjects from five studies at the University Medical Centre Groningen (UMCG). Namely, TEUNE, MD5, KOK, REESINK and SKM studies.

Source	Study ID	No. of PD patients	No. of AD patients	No. of HC subjects
UMCG	**TEUNE**	**19**	**0**	**17**
	MD5	**54**	**38**	**0**
	KOK	**0**	**0**	**16**
	REESINK	**0**	**0**	**12**
	SKM	**0**	**0**	**19**

Table 1: Number of FDG Pet images available from each study at UMCG for PD, AD and HC subjects

3. Methods

3.1 Feature Extraction

Subject scores are calculated from the FDG PET images and are used as the input features for the neural network. This is done using the Scaled Subprofile Modelling/Principal Component Analysis (SSM/PCA) method [8]. The first step is to reduce the dimensionality of the data by converting the 3D image into a flattened (1D) array of voxels. This is then put through a threshold mask (threshold of 35% whole brain maximum) to remove noise and out of brain voxels. The next step is a log transform and the subtraction of subject mean from each subject's voxel values. This is then subtracted by the group mean profile, which is the voxel-wise mean of the healthy control subjects from the TEUNE study [9] resulting in the subject residual profiles. This step scales the data around the mean of the healthy control's FDG PET scans. This results in an output matrix with both positive and negative values, positive indicating a greater brain activity and negative indicating a lower brain activity than a healthy subject in those specific regions of the brain.

The next step is to use principal component analysis (PCA) on the subject residual profiles, this helps with dimension reduction. PCA converts the subject residual profiles into clusters of correlated variables (Subject Scores) using principle components (PCs) [10]. PCs are orthogonal uncorrelated variables, which define the new space in which the subject scores exist. The number of PCs generated is equal to the number of subjects in the reference group and these PCs are arranged in descending order of variance, this results in the first few PCs having most of the variance. Here the TEUNE group (table 1.) is used as the reference group, which consists of 36 PD and HC subjects, hence 36 PCs are generated, resulting in dimension reduction. Subject scores were derived from the MD5 group for the AD and PD subjects, and the HC subject scores were obtained from the KOK, REESINK and SKM studies. As the reference group (TEUNE) consists of only PD and HC subjects, the subject scores are specific to the PD disease pattern. This enables distinguishing various neurodegenerative diseases such as Parkinson's and Alzheimer's easier. The subject scores are then fed into the neural network as features for the classification of these diseases. Fig.1 shows the leading two subject scores plotted against each other of AD and PD subjects. It can be seen that SSM/PCA alone is not enough to distinguish the two diseases, but it does help reduce dimensionality of the FDG-PET images, making it easier to use as input features for a feed forward neural network.

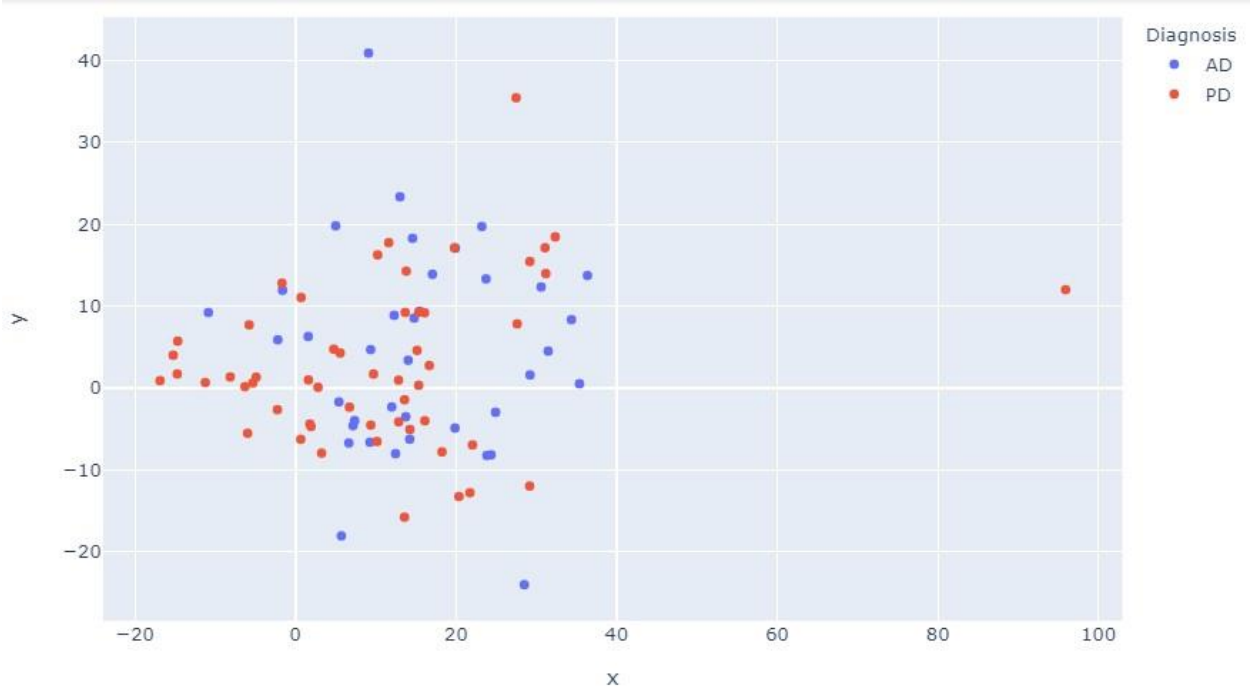

Figure 1: Graphical representation of the subject scores, with the x axis representing the first PC with highest variance and the y axis representing the second PC with the second highest variance.

3.2 Neural Network Design

Artificial neural networks (ANN) aim to mimic the learning abilities of the human brain which is made up of billions of neurons connected by synapses. At a primitive level, neurons are essentially computational units that take inputs through their dendrites as electrical inputs which are called "spikes". These inputs are then channelled to outputs through the axons. In our model, the dendrites are the input features, and the output is the result of our hypothesis function. Thus, an artificial neural network consists of interconnected processing units. The weights of the network are determined through supervised learning.

3.2.1 Neural Network Architecture

In this case the ANN is used to determine whether the FDG-PET images represent Parkinson's or Alzheimer's subjects. A one-hidden-layer feedforward neural network that consists of one hidden layer, one output layer and one input layer is used. The subject scores obtained from the SSM/ PCA method are used as the input features, each FDG-PET image generates 36 features as discussed in section 3.1. The last feature is excluded from the classification process as its

eigenvalue is 0 and does not help with classification. Hence, the input layer has 35 nodes. The output layer has 2 nodes, one indicating the probability of Parkinson's and the other the probability of Alzheimer's. The number of nodes in the hidden layer was chosen based on the ANN obtaining the least error, further discussed in section 3.3.1. The architecture of the feed forward neural network is shown in Fig. 2 below.

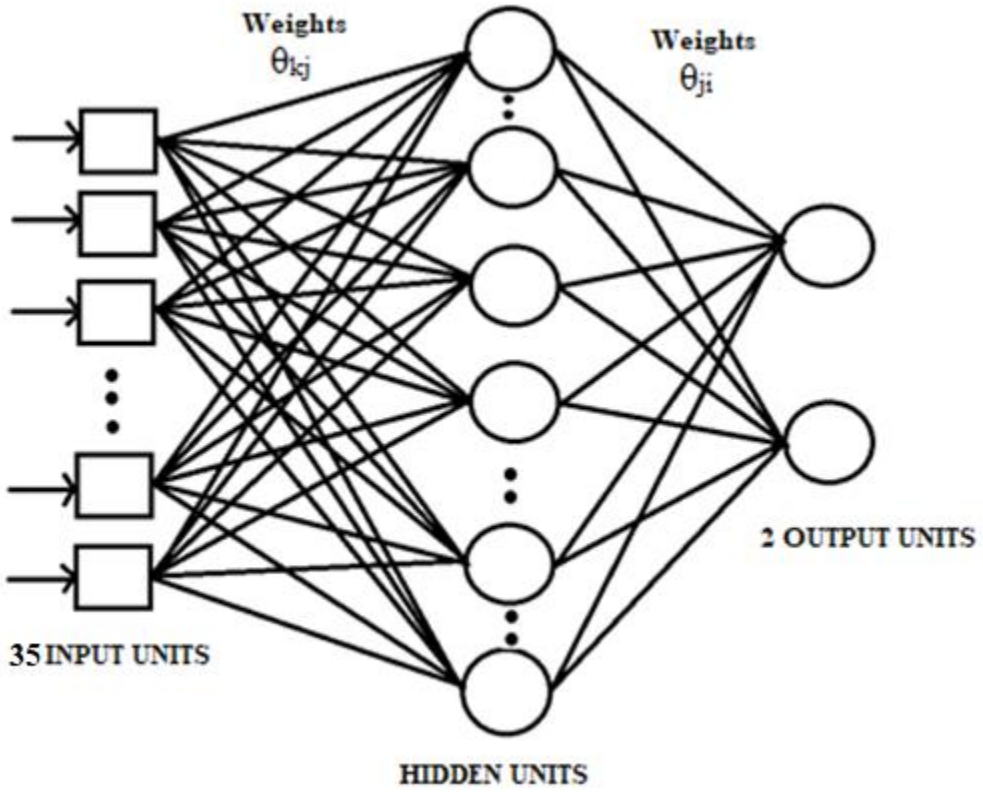

Figure 2: Architecture of neural network with one hidden layer used

The input nodes in layer 1, also known as the "input layer" are fed into every node in layer 2, i.e. the "hidden layer". The hidden layer then outputs the hypothesis function, also known as the "output layer". The value obtained in each node in the hidden and output layer is dependent on three variables, the inputs, weights and the biases and is given by:

$$y_j = \sum_{k=1}^{92}(\theta_{kj} * X_k) + Bias\ ,$$

where the weights are represented by θ_{kj} and θ_{ji} (Fig.2), θ_{kj} represents the weight from the k^{th} node in the input layer to the j^{th} node in the hidden layer and θ_{ji} represents the weight from the j^{th} node in the hidden layer to the i^{th} node in the output layer, y_j represents the output to the j^{th} node in the hidden layer and X_k represents the k^{th} node in the input layer. These weights and biases are adjusted repeatedly until the error between the predicted output and the actual output is minimized using back propagation of error, which is discussed further in section 3.2.3.

3.2.2 Activation Function

Before the output "y" is used as the input for the next layer, it is altered using an activation function: f(x). This activation function is part of each node and is used to determine whether the node should be activated or not depending on the input to the node. Applying an activation function in the network introduces non-linearity to the output, enabling the network to learn more complex tasks. Therefore, the final output of each node in the hidden layer is given by:

$$Y_j = f(y_j) = f\left(\sum_{k=1}^{92}(\theta_{kj} * X_k) + Bias\right).$$

Similarly, application of the weights and biases along with the activation function is repeated between the hidden layer and the output layer as seen in Fig.2. There are various activation functions that can be used, for this project the Rectifier Linear unit (ReLu) function and the Sigmoid function were used at the hidden layer in the neural network separately to check which function would yield better results. The activation function used between the hidden layer and the output layer is the SoftMax function as it provides the probability of the data representing a particular class of output, which is useful when classifying.

3.2.2.a ReLu Function:

Mathematically the ReLu function is defined as,

$$ReLu(x) = \begin{cases} x, & if\ x > 0 \\ 0, & if\ x \leq 0 \end{cases}$$

Where, x is the input. This is a linear function that leaves positive inputs untouched and outputs a 0 for all negative inputs.

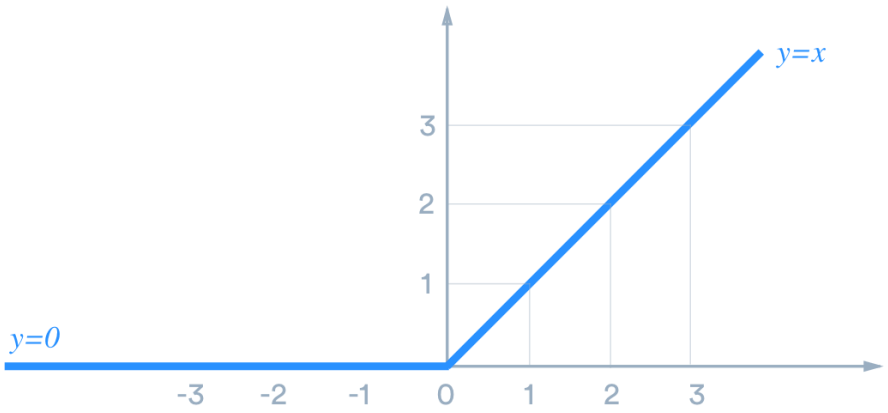

Figure 3: Graphical representation of the ReLu function

3.2.2.b Sigmoid Function:

Mathematically, the sigmoid (σ) function is given by,

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

The way that the sigmoid function works is that when the input given to it is greater than or equal to zero, its output is greater than or equal to 0.5. It is a logistic function, hence the output is always scaled between 0 and 1. The function has an s-shaped graph as shown in Fig. 4.

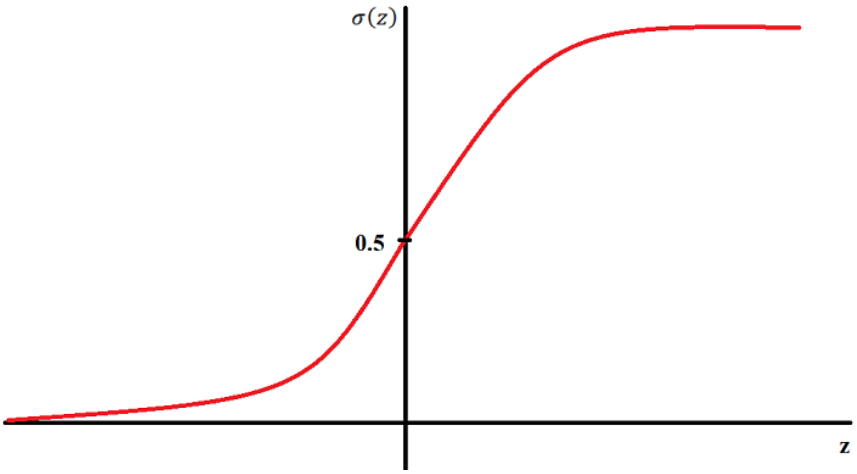

Figure 4: Graphical representation of sigmoid function

3.2.2.c Advantages and disadvantages of the ReLu and sigmoid activation functions:

Activation Function	Pros	Cons
ReLu	• Computationally inexpensive • Avoids vanishing gradient problem [11]. Which occurs when a large range of input is compressed into a small range such as in the sigmoid function. This leads to the gradient/ derivative of the activation function becoming a small value, which in turn results in the updating of the weights per epoch being very small during back propagation (section 3.2.3)	• Dead ReLu problem, where the weights and biases of certain nodes do not get updated due to the input being negative, hence causing the output to be zero every time. This results in negative inputs not being used effectively to train the model. • Cannot be used as the activation function for the output layer as the sum of outputs of each class is not equal to one, i.e. it does not give the probabilities of each class
Sigmoid	• No dead ReLu problem. • It can be used for models that predict probability more efficiently as the outputs lie between 0 and 1.	• Vanishing gradient problem • Computationally expensive due to the exponential operation. • Cannot be used as the activation function for the output layer as the sum of outputs of each class is not equal to one, i.e. it does not give the probabilities of each class

Table 2: Pros and Cons of the ReLu and Sigmoid activation functions.

3.2.2.d SoftMax Function:

While using a neural network to classify, SoftMax activation function is used at the output units to interpret the output values as posterior probabilities. The mathematical representation of the function is given below.

$$S(Y_i) = \frac{e^{Y_i}}{\sum_j e^{Y_j}}$$

Where, Y_i represents the input to the i^{th} output node. The SoftMax function gives output such that the sum of the outputs from all output layer nodes is 1, i.e., the probability of each class is the final output.

3.2.3 Back Propagation of error

Back propagation [12] is a term used in neural networks for minimizing the cost function (J(θ)) (Appendix) of the neural network, which is done by adjusting the weights (θ) in the network. The gradient/ derivative of the cost function with respect to the weights is calculated for each node. This value shows how quickly the cost changes when weights are adjusted. This allows us to identify the nodes at which modification of weights leads to the greatest reduction of error. Further details on the back propagation algorithm are provided in the appendix.

3.3 Division of Data

The data used for the feedforward Neural Network is the MD5 data from the UMCG (Table.1), which consists of 54 PD and 38 AD subject's FDG-PET data. The TEUNE data is not used as it is used to construct the transformation of the data during the SSM/ PCA feature extraction and using this data again to train the ANN would result an inflated performance. These 92 subject's data are divided into 2 groups after randomization of the order of the data: training (82), and test (10) groups. The training group is used to train the neural network, using backpropagation method, the weights and number of nodes in the hidden layer are altered so that the data is classified accurately. However, this could result in overfitting, where the classification is too specific to the training data. To counter this, a 10 times repeated 10-fold stratified cross validation is performed. The following parameters are optimized based on the accuracy the network shows after this process: the number of hidden nodes, the learning rate, the number of loops of back projection done to adjust the weights of the network and the activation function used. The range of the parameters used to find the optimum parameters is, number of hidden nodes tried between 20 to 80, learning rate chosen between 10^{-3}, 10^{-4} and 10^{-5}, number of back projection loops chosen between 200 to 1000 in steps of hundred. The test group remains completely left out of this process, i.e. no

parameters of the model are adjusted based on the test group. This helps simulating a real world situation and the accuracy can be calculated based on new unseen data.

3.3.1 N times Repeated K-Fold Cross Validation

For this neural network to have a practical application it needs to be tested with unseen data, i.e. data that is not used to train the model. The accuracy calculated from data already known to the model does not give an idea of how well the model will work with new subjects in a real world environment. Using unseen data to validate the model helps to check if the model is over-fitting, under-fitting or generalizing well.

Stratified K-fold cross validation works by dividing a dataset into k folds, each fold having approximately equal number of subjects with the ratio of AD and PD subjects in each fold being approximately equal to the original ratio between AD and PD patients. With N-times repeated K-fold stratified cross validation, this process is repeated N times, so as to give a new set of K folds from the dataset each time. In this case 10 times repeated 10 fold stratified cross validation was used, the training set (82 subjects) is divided into 10 folds. 8 folds having 8 subjects each and 2 having 9 subjects each. With approximately equal ratios of PD: AD subjects in each fold. Nine of these folds are then used as the training set and the remaining one as the validation set. The accuracy of the model having a fixed number of nodes in the hidden layer is then calculated. This is then repeated such that every fold is used as a validation set with the remaining folds as the training set, hence generating an array of 10 accuracies. This process is repeated 10 times resulting in an array of 100 accuracies from which the average accuracy and standard deviation is calculated.

This 10 times repeated 10 fold cross validation is repeated several times with the number of nodes in the hidden layer, the learning rate, the number of loops of back projection to adjust the weights of the model being changed each time and the activation function used. These four parameters are then chosen such that the average accuracy is the highest and the standard deviation is the least.

4. Results

A one layer feed forward neural network that consists of one hidden layer, one output layer with 2 output units and one input layer with 35 input units that are fed with the 35 features extracted

using SSM/PCA is used. Accuracy, sensitivity, and specificity of the neural network were calculated for the training, validation, and test data sets. Due to the 10 times repeated 10 fold cross validation training and validation resulted in 100 values of accuracy, sensitivity and specificity, the average and standard deviation of these are given below.

4.1 AD vs PD

Table 3 shows that the sigmoid function achieves better accuracy, sensitivity and specificity when compared to the ReLu activation function in both the training and validation groups. The two functions show equal results in the test group.

	Accuracy		Sensitivity		Specificity	
	ReLu	Sigmoid	ReLu	Sigmoid	ReLu	Sigmoid
Training	91% (0.03)	**94% (0.02)**	89% (0.05)	**93% (0.03)**	92% (0.04)	**95% (0.02)**
Validation	72% (0.15)	**77% (0.14)**	64% (0.27)	**70% (0.25)**	77% (0.18)	**83% (0.17)**
Test	70%	70%	50%	50%	83%	83%

Table 3: Training, Validation and Test results of the neural network. Average values shown with standard deviation in brackets. Higher scores in bold face.

Activation Function	**No. of Nodes**	**No. of back projection iterations**
ReLu	66	200
Sigmoid	63	600

Table 4: Neural network parameters that achieved highest accuracy.

4.2 HC vs AD

Table 5 shows that the ReLu function achieves better results in the training group but does not perform as well as the sigmoid function in the validation and test groups. This indicates that the ReLu function is overfitting.

	Accuracy		Sensitivity		Specificity	
	ReLu	Sigmoid	ReLu	Sigmoid	ReLu	Sigmoid
Training	**99.9% (0.001)**	98.8% (0.01)	**100% (0)**	97.8% (0.02)	**99.9% (0.002)**	99.6% (0.01)
Validation	92.4% (0.09)	**93.5% (0.08)**	90.5% (0.16)	**92.5% (0.14)**	94% (0.11)	94% (0.11)
Test	90%	**100%**	80%	**100%**	100%	100%

Table 5: Training, Validation and Test results of the neural network. Average values shown with standard deviation in brackets. Higher scores in bold face.

Activation Function	**No. of Nodes**	**No. of back projection iterations**
ReLu	65	300
Sigmoid	69	500

Table 6: Neural network parameters that achieved highest accuracy.

4.3 HC vs PD

Table 7 shows that the ReLu activation function is overfitting again with better results in the training group, but worse in the validation group when compared to the sigmoid function.

	Accuracy		Sensitivity		Specificity	
	ReLu	Sigmoid	ReLu	Sigmoid	ReLu	Sigmoid
Training	**99.7% (0.006)**	97.9% (0.01)	**99.5% (0.01)**	96.3% (0.02)	**99.9% (0.002)**	99.7% (0.008)
Validation	83% (0.12)	**84% (0.1)**	82% (0.2)	**82.4% (0.16)**	84% (0.18)	**86% (0.16)**
Test	100%	100%	100%	100%	100%	100%

Table 7: Training, Validation and Test results of the neural network. Average values shown with standard deviation in brackets. Higher scores in bold face.

Activation Function	**No. of Nodes**	**No. of back projection iterations**
ReLu	66	500
Sigmoid	68	900

Table 8: Neural network parameters that achieved highest accuracy

5. Discussion

From table 4, 6 and 8 we see that the ReLu activation function required lesser iterations of back projection to train the single layer feed forward neural network when compared to the sigmoid function. This is due to the vanishing gradient problem seen with the sigmoid function mentioned in section 3.2.2.c. The combination of greater iterations and it being more complex resulted in the sigmoid function having a greater computation time. However, due to the dead ReLu problem as mentioned in section 3.2.2.c where negative inputs do not contribute to the model, the ReLu function was found to be less effective in classifying the validation and test groups when compared to the sigmoid activation function. This might have had a bigger influence as the data was cantered around the HC subject, hence the areas of the brain with lesser activity than seen in HC subjects did not influence the model It was also found that the training results were higher when using the ReLu activation function when classifying AD and PD patients from HC subjects (Table 5, 7), yet the validation and test results were lower than the model using sigmoid activation function, indicating over fitting. With respect to the number of nodes, there was no drastic change in accuracies seen if the number of nodes were between 55-70.

Overall, the neural network using the sigmoid function performed better than the ReLu function. The greater training time was also not significant due to the amount of data used to train the network and the reduced dimensionality achieved by the SSM/ PCA. Some of the limitations of this study would be that ROC curves were not used to visualize the results. The use of a larger database would be very useful. Additionally, the classes within the data are unbalanced and were kept unbalanced by using stratified k fold cross validation, these could be balanced out by using synthetic minority oversampling technique (SMOTE) to increase the minority class by randomly replicating them.

6. Conclusion

Neural networks are able to distinguish (with limitations discussed in section 5) between HC, AD and PD well. Furthermore, the sigmoid function is the better choice for activation function compared to the ReLu. Comparing it to previous work done with LVQ in [6] performance can still be improved with respect to problem AD vs PD. The performance of the neural network with HC vs PD is equal to that of the LVQ. With HC vs AD, the LVQ model performs slightly better. As neural networks are more complex, a larger database would result in more accurate classification.

Further research can be done using different techniques such as adding more hidden layers to the network, using other activation functions or using other types of neural networks such as convolutional neural network (CNN). Using CNN on the FDG-PET images directly without the use of SSM/ PCA might be beneficial as the main advantage of CNN is its ability to identify important features in the data without human supervision. Additionally, the network performs well on the HC vs AD classification even though the reference group during the SSM/ PCA consisted of only HC and PD subjects, making the subject scores more specific to PD patterns. With a larger database using AD subjects in the reference group would be possible, which might lead to better results.

7. Bibliography

[1] "alzheimers.net," [Online]. Available: https://www.alzheimers.net/resources/alzheimers-statistics/.

[2] "parkinson.org," [Online]. Available: https://www.parkinson.org/Understanding-Parkinsons/Statistics.

[3] S. Risacher and A. Saykin, "Neuroimaging biomarkers of neurodegenerative diseases and dementia," *Seminars in Neurology,* vol. 33, no. 4, pp. 386-416, 2013.

[4] B. Magnin, L. Mesrob, S. Kinkingnehun, M. P ´ el´ egrini-Issac, O. Colliot, M. Sarazin, B. Dubois,, ´ S. Lehericy and H. Benali, "Support vector machine-based classification of Alzheimer's disease from ´ whole-brain anatomical MRI," *Neuroradiology,* vol. 51, no. 2, p. 73–83, 2009.

[5] S. Haller, S. Badoud, D. Nguyen, V. Garibotto, K.O. Lovblad and P.R Burkhard, "Individual Detection of Patients with Parkinson Disease using Support Vector Machine Analysis of Diffusion

Tensor Imaging Data: Initial Results," *American Journal of Neuroradiology,* vol. 33, no. 11, pp. 2123-2128, 2012.

[6] R. van Veen, L. Talavera Martinez, R. V. Kogan, S. Meles, D. Mudali, J. Roerdink, F. Massa, M. Grazzini, J. Obeso and . M. Rodriguez-Oroz, "Machine Learning Based Analysis of FDG-PET Image Data for the Diagnosis of Neurodegenerative Diseases," *Applications of Intelligent Systems,* vol. 310, pp. 280-289, 2018.

[7] R. v. Veen, V. Gurvits, R. V. Kogan, S. K. Meles, G.-J. de Vries, R. J. Renken, M. C. Rodriguez-Oroz, R. Rodriguez-Rojas, D. Arnaldi, S. Raffa, B. M. de Jong, K. L. Leenders and M. Biehl, "An Application of Generalized Matrix Learning Vector Quantization in Neuroimaging," *Computer Methods and Programs in Biomedicine,* 2020.

[8] Spetsieris PG and Eidelberg D, "Scaled subprofile modeling of resting state imaging data in Parkinson's disease: methodological issues," *Neuroimage,* vol. 54, no. 4, pp. 2899-2914, 2011.

[9] Laura K. Teune, Anna L. Bartels, Bauke M. de Jong, Antoon T. M. Willemsen, Silvia A. Eshuis, Jeroen J. de Vries, Joost C. H. van Oostrom and Klaus L. Leenders, "Typical Cerebral Metabolic Patterns in Neurodegenerative Brain Diseases".

[10] Lever, J., Krzywinski, M. and Altman, N., "Points of Significance: Principal component analysis.," *Nature Methods,* vol. 14, no. 7, pp. 641-642, 2017.

[11] Hong Hui Tan and King Hann Lim, "Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization," in *7th International Conference on Smart Computing & Communications (ICSCC)*, 2019.

[12] M. Nielsen, "Neural Networks and Deep Learning," 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap2.html#:~:text=Plan%20of%20attack%3A%20Backpropagation%20is,gradient%20of%20the%20cost%20function.&text=An%20equation%20for%20the%20error,%E2%80%B2(zLj).. [Accessed 2020].

8. Appendix

Back propagation of error [12]**:**

The cost function (J(θ)) is given by:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y(i)_k log(h_\Theta(x(i))_k + (1 - y(i)_k)\, log(1 - (h_\Theta(x(i))_k)] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\Theta(l)_j, i\right)^2,$$

where,

- L- Number of layers in the network
- S_l- number of units in layer *l*
- k- number of output nodes in the neural network
- $h_\Theta(x)_k$ is denoted as the hypothesis that results in the k^{th} output.
- λ is the regularization parameter.
- y is the given output value.
- m is the number of training samples.

The nested summations account for our multiple output nodes.

Back Propagation Algorithm:

Given training set $\{(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}).....(x^{(t)},y^{(t)})\}$

- Set $\Delta_{i,j}^{(l)} := 0$ for all *l*,i,j where Δ is the gradient, *l* is the layer going from *l*= 2,3…L.
- For training example t going from 1 to t, where t is the number of samples,
  Set $a_t^{(1)} := x^{(t)}$, where $a_t^{(1)}$ is the t^{th} node value at layer *l* =1.
- Implement forward propagation to compute $a^{(l)}$ for *l* = 2,3..L.

- Using $y^{(t)}$, compute $\delta^{(L)} = a^{(L)} - y^{(t)}$, where L is our total number of layers and $a^{(L)}$ is the vector of outputs of the activation units for the last layer. So, our "error values" for the last layer are simply the differences of our actual results in the last layer and the correct outputs in y. To get the delta values of the layers before the last layer, we can use an equation that steps us back from right to left.
- Compute $\delta^{(L-1)}, \delta^{(L-2)} \ldots, \delta^{(2)}$ using:

  $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) .* a^{(l)} .* (1 - a^{(l)})$

  The delta values of layer *l* are calculated by multiplying the delta values in the next layer with the theta matrix of layer *l*. Then, element-wise multiplication is performed with a function called σ', or g-prime, which is the derivative of the activation function σ evaluated with the input values given by $z^{(l)}$.

  The g-prime derivative terms can be written as:

  $\sigma'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$
- Now,

  $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

  Hence, we update our new Δ matrix.

  $D_{i,j}^{(l)} := \frac{1\left(\Delta_{i,j} + \lambda\theta_{i,j}^{(l)}\right)}{m}$, if $j \neq 0$

  $D_{i,j}^{(l)} := \frac{1(\Delta_{i,j})}{m}$, if $j = 0$

  The matrix D, is an accumulator that is used for summation of all values as we go along to obtain the partial derivative.

  Thus, $\frac{\delta}{\delta\Theta_{i,j}^{(l)}} J(\Theta) = D_{i,j}^{(l)}$.
- Update rule:

  $\Theta_j := \Theta_j - \alpha D_{i,j}^{(l)}$, where α is the learning rate. All Θ values are simultaneously updated.