Published in final edited form as:

Proc SPIE Int Soc Opt Eng. 2017 October ; 10572: . doi:10.1117/12.2294537.

Deep Learning based Classification of FDG-PET Data for Alzheimers Disease Categories

Shibani Singh^a, Anant Srivastava^a, Liang Mi^a, Richard J Caselli^b, Kewei Chen^c, Dhruman Goradia^c, Eric M. Reiman^c, and Yalin Wang^a

^aSchool of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA

^bDepartment of Neurology, Mayo Clinic Arizona, Scottsdale, AZ, USA

^cBanner Alzheimer's Institute, Phoenix, AZ, USA

Abstract

Fluorodeoxyglucose (FDG) positron emission tomography (PET) measures the decline in the regional cerebral metabolic rate for glucose, offering a reliable metabolic biomarker even on presymptomatic Alzheimer's disease (AD) patients. PET scans provide functional information that is unique and unavailable using other types of imaging. However, the computational efficacy of FDG-PET data alone, for the classification of various Alzheimers Diagnostic categories, has not been well studied. This motivates us to correctly discriminate various AD Diagnostic categories using FDG-PET data. Deep learning has improved state-of-the-art classification accuracies in the areas of speech, signal, image, video, text mining and recognition. We propose novel methods that involve probabilistic principal component analysis on max-pooled data and mean-pooled data for dimensionality reduction, and multilayer feed forward neural network which performs binary classification. Our experimental dataset consists of baseline data of subjects including 186 cognitively unimpaired (CU) subjects, 336 mild cognitive impairment (MCI) subjects with 158 Late MCI and 178 Early MCI, and 146 AD patients from Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset. We measured F1-measure, precision, recall, negative and positive predictive values with a 10-fold cross validation scheme. Our results indicate that our designed classifiers achieve competitive results while max pooling achieves better classification performance compared to mean-pooled features. Our deep model based research may advance FDG-PET analysis by demonstrating their potential as an effective imaging biomarker of AD.

Keywords

Deep Learning; Multilayer Perceptrons; Alzheimers; Neural Networks; Cross Validation; Dimensionality Reduction; PET

1. Introduction

In the study of Alzheimers disease (AD), neuroimaging based measures have shown high sensitivity in tracking changes over time and thus were proposed as possible biomarkers to evaluate AD burden and progression and response to interventions. In addition to the pathological amyloid and tau imaging measurements for AD, fluorodeoxyglucose (FDG) positron emission tomography (PET) characterizes the cerebral glucose hypometabolism related to AD and AD risk, offering a reliable metabolic biomarker even at its presymptomatic stage. Fig. 1 visualizes the neural activity in normalized PET scans of AD and normal subjects. We see that the central image in both cases displays loss of functionality for AD patients as compared to normal. There has been growing interest to study FDG-PET for AD and AD risk and particularly to identify and predict mild cognitive impairment (MCI). Although numerous analysis tools have been developed, much of the prior work (e.g.¹), has relied on voxel-wise analysis corrected by multiple comparisons to discover group-wise differences and the general trend in data. However, there are a number of issues in extending the group analysis framework to compute AD risk on individual basis. For example, prior work has showed that the statistically significant pixels obtained in group difference studies do not necessarily carry strong statistical power for predictions.² To develop an effective precision medicine, one needs some system which may be able to measure subtle difference and make robust prediction/classification on an individual basis. Thus far, it is still challenging to build FDG-PET imaging diagnosis and prognosis systems because of the tremendous difficulty to optimally integrate global functional image information.

In the study of Alzheimers disease (AD), neuroimaging based measures have shown high sensitivity in tracking changes over time and thus were proposed as possible biomarkers to evaluate AD burden, progression and response to interventions. In addition to the pathological amyloid and tau imaging measurements for AD, fluorodeoxyglucose (FDG) positron emission tomography (PET) characterizes the cerebral glucose hypometabolism related to AD and AD risk, offering a reliable metabolic biomarker even at its presymptomatic stage. Fig. 1 visualizes the neural activity in normalized PET scans of AD and normal subjects. We see that the central image in both cases displays loss of functionality for AD patients as compared to normal. There has been growing interest to study FDG-PET for AD and AD risk and particularly to identify and predict mild cognitive impairment (MCI). Although numerous analysis tools have been developed, much of the prior work (e.g.¹), has relied on voxel-wise analysis corrected by multiple comparisons to discover group-wise differences and the general trend in data. However, there are a number of issues in extending the group analysis framework to compute AD risk on individual basis. For example, prior work has showed that the statistically significant pixels obtained in group difference studies do not necessarily carry strong statistical power for predictions.² To develop an effective precision medicine, one needs some system which may be able to measure subtle difference and make robust prediction/classification on an individual basis. Thus far, it is still challenging to build FDG-PET imaging diagnosis and prognosis systems because of the tremendous difficulty to optimally integrate global functional image information.

Recently deep learning has helped achieve state-of-the-art classification results in myriad classification problems in the areas of signal, speech, text, image processing and medical imaging.³ Deep learning based feature representation using auto-encoders was recently used to achieve high accuracies using MRI and PET data.⁴ Deep learning has also been used for classification using MRI and PET data.⁵ Classification has been improvised using a combination of multiple imaging modalities to improvise on neuroimaging biomarkers, requiring less labeled data.⁶ The advance in deep learning research inspires us to develop novel deep learning methods to advance the FDG-PET analysis research which may facilitate their use in preclinical and clinical AD treatment development.

In this work, we propose a novel method that involves dimensionality reduction using Probabilistic Principal Component Analysis on max-pooled data and mean-pooled data, and a Multilayer Feed Forward Neural Network (also known as Multilayer Perceptron(MLP)) which performs binary classification. Fig 2 shows the pipeline of our system. We validated our algorithm in the Alzheimers Disease Neuroimaging Initiative (ADNI) baseline dataset (N= 668) consisting of baseline data of subjects including 186 healthy control (CU), 336 Mild Cognitive Impairment (MCI) with 158 Late MCI and 178 Early MCI, and 146 AD. The FDG-PET images were processed using SPM,⁷ for alignment, segmentation and normalization. We measured F1-measure, precision, recall, negative and positive predictive values with a 10-fold cross validation. Our results indicate that our designed classifiers achieve competitive results while max pooling results into better classification performance compared to results on mean pooled features.

Our work has three main contributions. First, we propose a coherent and efficient deep learning framework that well explores the possibility of FDG-PET for AD diagnosis. Secondly we evaluated our work in a relatively large dataset and achieved competitive results. Thirdly we exhibit the effective increase in classification performance with the addition of demographic variables (Age, Gender, APOE 1, APOE 2, and FAQ score) to our max-pooled (intensity) data. We conduct thorough comparison experiments by comparing our work to other state-of-the-art FDG-PET analysis methods. Our work may inspire more deep learning based work on FDG-PET analysis and advance preclinical AD research.

2. Data and Methods

We work on FDG-PET data from the ADNI-2 dataset. This dataset contains FDG-PET data that has been manually labeled into diagnostic categories by an expert. The baseline data of patients includes 186 healthy control (CU), 336 Mild Cognitive Impairment (MCI) with 158 Late MCI and 178 Early MCI, and 146 AD.

2.1 Data and Processing

The size of each FDG-PET image is $79 \times 95 \times 79$. Table 1 shows the age distribution for our subjects. We normalize the data to linearly align all the images into a common space using the software toolkit Statistical Parametric Mapping.⁷ The normalized FDG-PET images are of size $79 \times 95 \times 79$. Each value is a voxel intensity value. We use the intensity values for the whole brain in our experiments. Each voxel is a feature and hence the feature dimensionality is $592895(f_{dim})$ per image data sample. Since the number of data samples is

much less than the number of features ($n \ll f_{dim}$), we use dimensionality reduction techniques to reduce f_{dim} . This is discussed in the next section. We then use a multilayer perceptron classifier to perform binary classification. A gene called APOE can influence the risk for the more common late-onset type of Alzheimer's. There are three types of the APOE gene, called alleles: APOE2, E3 and E4. The E2 allele is the rarest form of APOE and carrying even one copy appears to reduce the risk of developing Alzheimer's by up to 40%. APOE3 is the most common allele and doesn't seem to influence risk. The APOE4 allele, present in approximately 20% of people, increases the risk for Alzheimer's and lowers the age of onset. The National Institutes of Health recommends genetic testing for APOE status to advance drug research in clinical trials. APOE4 is just one of many risk factors for dementia and its influence can vary across age, gender, race, and nationality^{8,9}.

We use this information to further enhance our classification performance. We also use the age, gender and FAQ (Functional Activities Questionnaire) scores for each subject. FAQ is an informant-based measure of functional abilities. Informants provide performance ratings of the target person on ten complex higher-order activities. We have the FAQ scores, APOE1 and APOE2 values for each of our subjects.

2.1.1 Extent of Linear Separation—The term “classification” can also be described as finding a clear separation between different classes. This can be thought of as separating two points in a 2D plane by a line. Similarly, if the training examples in an n-dimensional are linearly separable, we can easily separate them by constructing an (n-1)-dimensional plane. We use linear SVM, which is based on the implementation LIBLINEAR¹⁰ to judge the linear separability in the maxpooled data. Based on our judgement, we then configure the multilayer perceptron to increase the linear separability in data representation. Table 2 shows the extent of Linear Separability between each pair of classes. We see that the linear separability is relatively low for EMCI/CU and least for LMCI/EMCI. These classes also have a smaller temporal gap in disease progression.

2.1.2 Sparse Representation of Data—This non-linearity of data can be visualized as a dense representation of the training examples. We need to sparsely represent the data in order to be able to distinguish between different classes. Sparse representations are useful in the following ways (for details please refer to Glorot et al.¹¹):

- **Information disentangling:** To disentangle the factors explaining the variations in the data. Lets say we have a densely populated region with data points (having dense representations, represented in small dimensions) from various classes. Learning a model on this representation does not guarantee correct classification for points that slightly vary in feature values, from the points the model has trained on. If a representation is both sparse and robust to small changes in feature values, the non-zero features will be conserved throughout training.
- **Efficient variable size representation:** Due to the varying amount of information contained in every input, the number of active input neurons will vary. This may help control dimensionality of the representation for every input.

- Linear Separability: Sparse representations induce linear separability of data due to high dimensionality of sparse representations.

We will see further that Rectified Linear Units when applied to neurons in the multilayer perceptron are able to help the network learn sparse representations of the data.

2.2 Feature Selection using Maxpooling

Feature selection, variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features of all the available descriptors of data. Y-Lan Boureau et. al.¹² show that maxpooling has better performance than other pooling operations. Pooling is widely used to reduce the number of features, to help boost classification performance. Since our dataset consists of 668 data samples with 592895 features each, the number of samples is much less compared to the number of features. Learning based on this representation does not help better classification. We therefore perform max-pooling to reduce the number of features by a large count, for a sample count in hundreds. While the features are pooled, we also make sure to keep consistent overlapping between the patches. So for every two consecutive patches we have three maxpooled values. Overlapping is necessary to store relational information in the features. Max-pooling with overlapping patches of size $(10 \times 10 \times 10)$ reduces our feature dimensionality to 4050 per data sample. We perform maxpooling on the 3-dimensional PET images to make it 2 dimensional.

We then run Linear Support Vector Machine (SVM) on our data to test its linear separability. Running linear SVM as shown previously by Asa Ben-Hur et. al.¹³ displays the extent of linear separability between the various categories on maxpooled data. Running Linear SVM on a binary classification problem shows how linearly separable classes can be classified using a linear classifier like SVM. The hyper-plane (line in 2-d) is the classifiers decision boundary. A point is classified according to which side of the hyper-plane it falls on, which is determined by the sign of the discriminant function.

Running Linear SVM on maxpooled data gives us results (measured as f1-score) as shown in Table 2. From the table we see that AD and CU are to a large extent linearly separable (with a few incorrect classifications), and LMCI EMCI are not linearly separable, because the f1-score is 0.62. The patch size was varied for AD/CU classification from $5 \times 5 \times 5$ to $15 \times 15 \times 15$. We compare performance based on patch size for AD and CU subject data. For this experiment we used max-pooled data with age, gender, apoe1, apoe2 and FAQ score for all data samples. In max-pooling 3-dimensional patches of size $p_{size} \times p_{size} \times p_{size}$ are extracted uniformly from the 3-dimensional intensity value data. We vary the patch sizes in the range $p_{size} = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$. For the experimentation without demographic features and just max-pooled values, the number of hidden layers is given by $n_{hidden} = 4$ and the number of neurons in each hidden layer are $n_{neurons} = [1000, 500, 100, 10]$. For the experimentation with demographic features in addition to max-pooled values as the input, the number of hidden layers is given by $n_{hidden} = 7$ and the number of neurons in each hidden layer are $n_{neurons} = [1000, 800, 600, 400, 200, 100, 10]$. We compare both max-pooled data with demographics and performance on just max-pooled data. Table 3

We then select two patch sizes ($p_{size} = 9$, $p_{size} = 10$) to experiment with, on Max-pooled data along with 4 demographic features. The performance of both patch sizes is comparable and almost similar. We proceed with a patch size of $10 \times 10 \times 10$ for all the following experiments.

2.3 Dimensionality Reduction using Probabilistic Principal Component Analysis

In machine learning, dimensionality reduction or dimension reduction is the process of reducing the number of random variables under consideration, via obtaining a set of principal variables. Dimensionality reduction is the introduction of new feature space where the original features are represented. The new space is of lower dimension than the original space.

M.E. Tipping¹⁴ proposed a closed form solution to estimate Maximum Likelihood for Probabilistic Principal Component Analysis (PPCA). Principal Component Analysis (PCA) is widely used to transform data into reduced dimensionality. PCA maximizes the variance of projected data(x), which is represented in a lower dimensional space using a set of orthonormal vectors W . PPCA is the following latent variable model:

$$\begin{aligned} z &\sim \mathcal{N}(0, I); \\ x &\sim \mathcal{N}(Wz + \mu, \sigma^2 I), \end{aligned}$$

where $x \in \mathbb{R}^p$ is one observation and $z \in \mathbb{R}^q$ is a latent variable vector, usually $q \ll p$.

The error covariance structure in PPCA is $\sigma^2 I$. The Maximum Likelihood solution for PPCA is obtained as:

$$W_{MLE} = U_q (\Lambda_q - \sigma_{MLE}^2 I)^{1/2} R,$$

where U_q is a matrix of q leading principal directions (eigen values of the covariance matrix), Λ_q is a diagonal matrix of corresponding eigenvalues.

$\sigma_{MLE}^2 = \frac{1}{d - q} \sum_{j=q+1}^d \lambda_j$ represents the variance lost in the projection and R is an arbitrary $q \times q$ rotation matrix (corresponding to rotations in the latent space).

Principal Components are generally selected to reduce reconstruction error or to maximize variance. We select the approach of variance maximization to choose the appropriate number of principal components for each of our binary experiments. Table 4 shows our choice of the number of Principal Components chosen using PPCA.

Fig. 3(a) shows that PPCA, even with 3 components separates AD from Normal to a great extent, whereas in Fig. 3(b) we see that 3 of the 4050 maxpooled features are closely represented with possible overlaps for AD and Normal. The cumulative variance displayed in Fig. 3(c) is low ($\sim 35\%$), whereas for 250 components PPCA has a high cumulative

variance ($\sim 97\%$) shown in Fig. 3(d). We further need to reduce our feature dimensionality from 4050 to a count in hundreds, as training a neural net with features close to the number of samples will give us a model that can perform better classifications. Hence we use PPCA to reduce our 4050 max-pooled/mean-pooled features to features in the range 250 to 300. This range of feature dimensionality count gives the best variance from PPCA, as shown in Fig. 3(d).

2.4 Multilayer Perceptron

A perceptron produces a single binary output given several binary inputs x_1, x_2 , and so on. Fig 4 below shows a schematic of Rosenblatt's Perceptron with m inputs $x_1, x_2, x_3 \dots x_m$.

Rosenblatt* proposed a simple way to compute the output. He assigned weights $w_1, w_2, \dots$, to inputs signifying the importance of inputs in the determination of the output. In the modern sense, the perceptron is an algorithm for learning a binary classifier: a function that maps its input x (a real valued vector) to an output value $f(x)$ (a single binary value):

$$f(x) = \begin{cases} 1 & w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product $\sum_{i=0}^m w_i x_i$, where m is the number of inputs to the perceptron and b is the bias. The bias shifts the decision boundary away from the origin and does not depend on any input value. A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP networks are typically used in supervised learning problems. This means that there is a training set of input-output pairs and the network must learn to model the dependency between them. The supervised learning problem of the MLP can be solved with the backpropagation algorithm. The algorithm consists of two main steps, forward propagation and back propagation. In the forward pass, the predicted outputs corresponding to the given inputs are evaluated, by applying a set of weights to the input data. For the first forward propagation, the set of weights is selected randomly. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. Back propagation measures the margin of error of the output and adjusts the weights accordingly to decrease the error.

A one hidden layer MLP can be represented using the function: $f: R^D \rightarrow R^L$ where D is the size of input vector x and L is the size of the output vector $f(x)$.

*https://en.wikipedia.org/wiki/Frank_Rosenblatt

Where, $f(x) = G(b^{(2)} + W^{(2)}(s(b^{(2)} + W^{(1)}x)))$ with bias vectors $b^{(1)}, b^{(2)}$; weight matrices $W^{(1)}, W^{(2)}$ and activation functions G and s .

2.4.1 Activation Function—We use the activation function Rectified Linear Unit for the activation units in the MLP. The rectifier activation function allows a network to easily obtain sparse representations, hence inducing the sparsity effect on networks. Figure 6 shows how the 1st hidden layer has certain deactivated neurons because the output from their respective ReLu activation functions is zeroed out. Which means they do not contribute as inputs to the second hidden layer, and only 2 of the 6 neurons in the 1st hidden layer contribute as inputs to the 2nd hidden layer. Similarly sparsity is induced with ReLu in the 2nd hidden layer.

Experimental results show engaging training behavior of this activation function, especially for deep architectures,¹⁵ i.e., where the number of hidden layers in the neural network is 3 or more. Which means that training proceeds better when operating neurons in a network are either off or operating mostly in a linear regime. MLP using a backpropagation algorithm is the standard algorithm for any supervised learning pattern recognition process and the subject of ongoing research in computational neuroscience and parallel distributed processing. They are useful in research in terms of their ability to solve problems stochastically, which often allows one to get approximate solutions for extremely complex problems like classification.

After reducing the feature dimensionality for each sample, we pass these as inputs to our MLP. We tried various configurations to obtain the best performing models. We vary the number of hidden layers and the number of neurons in each layer for every classification experiment. One hidden layer MLPs are known to have good classification accuracies with x

number of neurons where $x = \frac{\text{inp} + \text{out}}{2}$, where *inp* is the number of inputs and *out* is the number of outputs. The activation function for each neuron is Linear Rectification, which given an input y , returns $f(y) = \max(0, y)$.

The learning rate for MLP is set to 0.001, and the loss minimization (Gradient Descent Optimization) is performed using the Adam(Adaptive Moment Estimation) Optimizer.¹⁶

2.4.2 Backpropagation—Backpropagation is a method of training artificial Neural Networks used in conjunction with an optimization method (such as gradient descent). Backpropagation calculates a gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function. Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as autoencoders.

Backpropagation is used for computing the error δ' and the gradient of the cost function. The Backpropagation algorithm is as follows:

1. Input x : Set a^1 (activation) for the input layer

2. Feedforward: For each $l = 2, 3, \dots, L$, compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$
3. Output Error: (δ^l)
Compute the vector $\delta^l = \nabla_a C \odot \sigma'(z^l)$
4. Backpropagate the Error: For each $l = L-1, L-2, \dots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. Output: Gradient of the cost function is given by: $\frac{\delta C}{\delta w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\delta C}{\delta b_j^l} = \delta_j^l$

The loss minimization problem can be given by:

$$W_{\min} \left\{ L(W) := \frac{1}{m} \sum_{i=1}^m \ell(W; x_i, y_i) + \lambda r(W) \right\}$$

where, $\{x_i, y_i\}_{i=1}^m$ are training instances for (x_i) and corresponding labels; (y_i)

W - network parameters to learn; $\ell(W; x_i, y_i)$ - loss of network parameterized by W w.r.t (x_i, y_i) ; $r(W)$ - regularization function. (e.g. $\|W\|_2^2$); $\lambda < 0$ - regularization weight.

Optimization methods must be, **First-order** - update based on objective value and gradient only; and **Stochastic** - update based on subset of training examples.

$$L_t(W) := \frac{1}{b} \sum_{j=1}^b \ell(W; x_i, y_i) + \lambda r(W)$$

$(x_i, y_i)_{j=1}^b$ - random mini-batch chosen at iteration t .

The update rule for Adam Optimization is:

$$W_t = W_{t-1} - \alpha \frac{\hat{M}}{\sqrt{\hat{R}_t + \epsilon}}$$

where,

$$M_t = \beta_1 M_{t-1} + (1 - \beta_1) L_t(W_{t-1}) \text{ (1}^{st} \text{ moment estimate);}$$

$$R_t = \beta_2 R_{t-1} + (1 - \beta_2) L_t(W_{t-1})^2 \text{ (2}^{nd} \text{ moment estimate);}$$

$$\hat{M}_t = M_t / (1 - (\beta_1)^t) \text{ (1}^{st} \text{ moment bias correction);}$$

$$\hat{R}_t = R_t / (1 - (\beta_2)^t) \text{ (2}^{nd} \text{ moment bias correction).}$$

The Hyper-parameters are: $\alpha > 0$ - learning rate (choice : 0.001);

$\beta_1 \in [0, 1)$ - 1st moment decay rate (choice : 0.9);

$\beta_2 \in [0, 1)$ - 2nd moment decay rate (choice : 0.999);

$\epsilon > 0$ - numerical term (typical choice: 10^{-8}).

Adam adaptively selects a separate learning rate for each parameter. Parameters that would ordinarily receive smaller or less frequent updates receive larger updates with Adam (the reverse is also true). This speeds learning in cases where the appropriate learning rates vary across parameters.

2.5 Finding an Optimal Configuration

We further experiment to find an optimal deep neural network configuration for the MLP architecture. Our experimentation is as shown in the algorithm, which is purely trying to estimate in one fixed direction. According to this algorithm, we estimate a one hidden layer MLP that gives the maximum f_1 score when neurons are varied from 5 to 1000. We then fix the number of neurons in the first layer to the one that gives us the best f_1 score, and vary the number of neurons in the second layer from 5 to 1000, and estimate the number of neurons that give us the best f_1 score. We keep doing this for iterations counting the number of hidden layers given as input. Since this is purely based on an assumption that fixing the previous neural network configurations and varying the new hidden layer configuration will lead to better results, this may not be true. Also we only vary the number of neurons from 5 to 1000 at intervals of 5, increasing the upper and lower bound for this experiment may lead to different results. To investigate this approach, we use it for the classification of 7(excluding CU MCI) of the binary classification experiments. The results are as shown in Appendix A A This helps us reach better accuracies than a random brute force approach. A better approach would be to try all permutations and combinations for each hidden layer, but this approach is computationally expensive.

Algorithm Pseudo Code for Estimating an Optimal Configuration for n hidden layers

```

arr_sizes=emptyList([])
n_iter= n
for i ← [1,2,...,n_iter] do
    for numNeurons = 5; numNeurons+= 5; numNeurons 1000 do
        for k_fold in 10 fold Cross Validation do
            models←MLPClassifier(hiddenLayerSizes=arr_sizes.append(numNeurons), activation='ReLU',
            solver='Adam', maxIterations=1000)
            {Now train on k-1 folds and test on kth fold}
            model.fit([1_fold, 2_fold, ..., k-1_fold], trainLabels)
            {Add all confusion matrices for k-folds}
            confusionMatrix←confusionMatrix+confMat(testLabels, model.predict(kth fold))
        end for
        {Store the number of neurons for which we have the maximum  $f_1$  score}
        if confusionMatrix. $f_1$  > max  $f_1$  then
            max  $f_1$  ← confusionMatrix. $f_1$ 
        end if
    end for
end for

```

```

        bestNumNeurons ← numNeurons
    end if
end for
arrsizes[i] ← bestNumNeurons
end for

```

3. Results

Figure 7 shows the ROC and AUCs for each of our experiments with demographic data and without demographic data. In this section, we perform experiments to validate our proposed method on the ADNI2 dataset for evaluating its performance. We use FDG-PET baseline scans from the ADNI2 dataset. We use the software toolkit Statistical Parametric Mapping (SPM)⁷ to linearly align all the images into a common space. We measure the classification accuracy using f1-measure. To evaluate the performance of our method, we perform 10-fold cross validation.

Our experiments show that the proposed system is promising for AD diagnosis research. Whether or not this approach provides more statistical power than those afforded by other classification work requires careful validation for each application. We anticipate that this work will inspire more work that builds deep learning based systems for FDG-PET data analysis. We compare our results with the best achieved results as of yet using FDG-PET¹⁷ images only, our results show an increase by 4.76% in the accuracy for AD Normal.

Comparison with other Classification Algorithms

We compare our MLP results with other machine learning algorithms. This comparison has been done on maxpooled data combined with age, gender, APOE 1, APOE 2, and FAQ score information using 10 fold cross validation for each of the methods.

Comparison with Other Dimensionality Reduction Algorithms

Table 6 shows comparison of PPCA as a dimensionality reduction technique to other prominently used techniques. This experimentation was done on max-pooled data along with demographic features (age, gender, apoe and FAQ scores), and 10 fold cross validation to evaluate performance. We see that PPCA outperforms the other techniques. These accuracies are achieved by trying random combinations of neural networks.

Effect of Demographic Features

We also include age and gender of the subjects as additional features to our maxpooled data. Our data matrix, input to the neural network is in this case of size $n \times 4052$ where n is the number of samples (training + testing). n is 332 in the case of CU AD binary classification experiment (186 CU samples, 146 AD samples). We see in Table 7 the improvisations with the addition of age and gender (assigned values 0 for female and 1 for male) to maxpooled data. Our results clearly display a difference with the addition of just two demographic features, which means an addition of other features from ADNI subjects (such as MMSE score) will further help improve prediction results.

We also append AD/CU beta positive/negative values as features in addition to the demographic features along with maxpooled PET intensity values. The results are as represented in Table 8.

We follow the algorithm described in methods, and our best results for the classification of FDG-PET data with and without demographics is shown in Table 9. These results are obtained for varying configurations of MLP, each of which has been discussed in this section and the previous sections.

We further add beta positive/negative for AD and CU comparison experiment along with demographic features. The experiment is performed on max-pooled data from which 300 probabilistic principal components are extracted. We see in Methods how the number of principal components is selected for each experiment.

Comparison of Max-pooled data with Mean-pooled data

We performed Binary Classification experiments on 2 types of datasets (one with max-pooling and the other mean-pooling). We perform max-pooling on our data and mean-pooling on our data. We then compare the two datasets based on classification performance achieved by using a Multilayer Perceptron(MLP) classifier. Table 10 shows that max-pooling helps achieve better performance in majority of the binary classification experiments.

4. Conclusion and Future Work

We present a deep learning framework for AD clinical group classification using FDG-PET imaging data. Our findings support that deep model may be useful to improve clinical diagnosis and prognosis of AD. In future, we will refine our research and apply our framework to further study cognitive score and treatment effect prediction problems.

Acknowledgments

The research was supported in part by NIH (R21AG049216, RF1AG051710 and U54EB020403) and NSF (DMS-1413417 and IIS-1421165).

Appendix A. Configuring A 5-Hidden-Layer MLP

We studied the relationship between the depths of the deep models and performance in a variety of experiments. The results are summarized in Table 11, 12, 13, 14, 15, 16, 17.

References

1. Reiman EM, Caselli RJ, Chen K, Alexander GE, Bandy D, Frost J. Declining brain activity in cognitively normal apolipoprotein e $\epsilon 4$ heterozygotes: a foundation for using positron emission tomography to efficiently test treatments to prevent alzheimer's disease. *Proceedings of the National Academy of Sciences*. 2001; 98(6):3334–3339.
2. Sun D, van Erp TG, Thompson PM, Bearden CE, Daley M, Kushan L, Hardt ME, Nuechterlein KH, Toga AW, Cannon TD. Elucidating a magnetic resonance imaging-based neuroanatomic biomarker for psychosis: classification analysis using probabilistic brain atlas and machine learning algorithms. *Biological psychiatry*. 2009; 66(11):1055–1060. [PubMed: 19729150]

3. Hazlett HC, Gu H, Munsell BC, Kim SH, Styner M, Wolff JJ, Elison JT, Swanson MR, Zhu H, Botteron KN, et al. Early brain development in infants at high risk for autism spectrum disorder. *Nature*. 2017; 542(7641):348–351. [PubMed: 28202961]
4. Suk, H.I., Shen, D. Deep learning-based feature representation for AD/MCI classification. *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer. 2013. p. 583-590.
5. Li F, Tran L, Thung KH, Ji S, Shen D, Li J. A robust deep model for improved classification of AD/MCI patients. *IEEE journal of biomedical and health informatics*. 2015; 19(5):1610–1616. [PubMed: 25955998]
6. Liu S, Liu S, Cai W, Che H, Pujol S, Kikinis R, Feng D, Fulham MJ, et al. Multimodal neuroimaging feature learning for multiclass diagnosis of Alzheimer's disease. *IEEE Transactions on Biomedical Engineering*. 2015; 62(4):1132–1140. [PubMed: 25423647]
7. Penny, W.D., Friston, K.J., Ashburner, J.T., Kiebel, S.J., Nichols, T.E. *Statistical parametric mapping: the analysis of functional brain images*. Academic press; 2011.
8. Farrer LA, Cupples LA, Haines JL, Hyman B, Kukull WA, Mayeux R, Myers RH, Pericak-Vance MA, Risch N, Van Duijn CM. Effects of age, sex, and ethnicity on the association between apolipoprotein e genotype and alzheimer disease: a meta-analysis. *Jama*. 1997; 278(16):1349–1356. [PubMed: 9343467]
9. Liu CC, Kanekiyo T, Xu H, Bu G. Apolipoprotein E and Alzheimer disease: risk, mechanisms and therapy. *Nature Reviews Neurology*. 2013; 9(2):106–118. [PubMed: 23296339]
10. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. Liblinear: A library for large linear classification. *Journal of machine learning research*. 2008; 9(Aug):1871–1874.
11. Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. *Aistats*. 2011; 15(106):275.
12. Boureau, Y.L., Ponce, J., LeCun, Y. A theoretical analysis of feature pooling in visual recognition; *Proceedings of the 27th international conference on machine learning (ICML-10)*; 2010. p. 111-118.
13. Ben-Hur A, Weston J. A users guide to support vector machines. *Data mining techniques for the life sciences*. 2010:223–239.
14. Tipping ME, Bishop C. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*. 1999; 21:611–622.
15. Bengio Y, et al. Learning deep architectures for ai. *Foundations and trends in Machine Learning*. 2009; 2(1):1–127.
16. Diederik, P., Kingma, J.B. Adam: A method for stochastic optimization; *The International Conference on Learning Representations (ICLR)*; 2015.
17. Illán I, Górriz JM, Ramírez J, Salas-Gonzalez D, López M, Segovia F, Chaves R, Gómez-Rio M, Puntónet CG, Initiative ADN, et al. 18 F-FDG PET imaging analysis for computer aided Alzheimers diagnosis. *Information Sciences*. 2011; 181(4):903–916.

Figure 1.
Normalized PET image slices for CU and AD subjects.

Figure 2.

Classification Pipeline. From the left to the right: pre-processed PET data is the initial input data; PET images are normalized by ; Max-pooling/Mean-pooling for two classification pipelines performed on each subject's data to reduce the dimension of features from $79 \times 95 \times 79$ to 4050×1 ; probabilistic Principal Component Analysis (PCA) is applied to further reduce the number of features from 4050 to 250 – 300; new features with reduced dimensions per image is passed to train a Multilayer Feed-forward Neural Network; the neural network assigns class labels as a binary classification problems.

Figure 3. PPCA on ADNI2 subject samples for AD and Normal subjects. (a) 3 component PPCA for AD and Normal shows a good separation of AD and Normal subjects (b) Displaying the first 3 dimensions out of 4050 of the maxpooled data, (c) Cumulative Variance for 3 component PCA (d) Cumulative Variance for up to 250 features

Figure 4. Schematic of Rosenblatt's perceptron

Figure 5. Multilayer Perceptron with 1 hidden layer

Figure 6. Rectified Linear Units Inducing Sparsity in a Neural Network, adopted from Bengio et.al.¹⁵

Figure 7. ROC for Multilayer Perceptron Classifier. The figure on the left is without the addition of demographic features, and on the right is with the addition of demographic features

Table 1**Age Distribution for Subjects**

Category	Age $\pm$ SD	Age Range	Males	Females
AD	74.74 $\pm$ 8.16	56 ~ 90	85	61
MCI	71.88 $\pm$ 7.34	55 ~ 91	186	150
LMCI	72.50 $\pm$ 7.51	55 ~ 91	84	74
EMCI	71.34 $\pm$ 7.20	55 ~ 88	102	76
CU	73.56 $\pm$ 6.25	56 ~ 89	89	97

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 2

Linear SVM, an estimate of linear separability

Linear SVM	AD CU	AD MCI	CU MCI	EMCI AD	LMCI AD	EMCI CU	LMCI EMCI	LMCI CU
F1-Score	0.9178	0.8433	0.7082	0.8138	0.6882	0.6377	0.6260	0.6507

Table 3

Performance Comparison: Patch Size vs F1-Accuracy

Patch Size (n)	Without F1	Demographics	Precision	Recall	With F1	Demographics	Precision	Recall
5	0.9312	0.9462		0.9167	0.9474	0.9677		0.9278
6	0.9175	0.9570		0.8812	0.9708	0.9839		0.9581
7	0.9044	0.9409		0.8706	0.9710	0.9892		0.9534
8	0.9271	0.9570		0.8990	0.9735	0.9892		0.9583
9	0.9231	0.9677		0.8824	0.9737	0.9946		9536
10	0.9255	0.8867		0.9677	0.9735	0.9839		0.9632
11	0.9251	0.9624		0.8905	0.9661	0.9946		0.9391
12	0.9299	0.9624		0.8995	0.9661	0.9946		0.9391
13	0.8144	0.7822		0.8495	0.9561	0.9946		0.9204
14	0.9199	0.9570		0.8856	0.9634	0.9892		0.9388
15	0.9133	0.9624		0.8689	0.9609	0.9892		0.9340

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 4

#Principal Components Chosen for Each Experiment.

Experiment	AD		CU		AD		AD		CU		CU		EMCI		EMCI	
	CU	MCI	AD	MCI	CU	MCI	AD	EMCI	LMCI	AD	LMCI	CU	LMCI	EMCI	LMCI	
#PCs	300	400	400	300	500	500	300	300	300	300	320	320	340	340	310	310

Table 5
Performance Comparison: MLP vs Other Machine Learning Algorithms

Method	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
PPCA+MLP	F1	0.9734	0.8954	0.7830	0.8621	0.7790	0.8325	0.72	0.656
	Prec	0.9839	0.9167	0.8214	0.8562	0.7603	0.9086	0.7742	0.6910
	Recall	0.9632	0.875	0.7479	0.8681	0.7986	0.7682	0.6729	0.6244
PPCA + Linear SVM	F1	0.9558	0.8781	0.7625	0.8522	0.7279	0.7413	0.6598	0.6136
	Prec	0.9892	0.8899	0.75	0.8493	0.7329	0.7473	0.6882	0.6067
	Recall	0.9246	0.8667	0.7754	0.8552	0.7230	0.7354	0.6337	0.6207
PPCA + SGD Classifier	F1	0.9551	0.8846	0.7463	0.8255	0.6957	0.7287	0.6773	0.5977
	Prec	0.9731	0.8899	0.7440	0.8425	0.7123	0.7366	0.6828	0.5843
	Recall	0.9378	0.8794	0.7485	0.8092	0.6797	0.7211	0.6720	0.6118
PPCA+GNB Classifier	F1	0.9080	0.8113	0.7098	0.7451	0.6351	0.6841	0.6067	0.6011
	Prec	0.8495	0.8125	0.7024	0.7808	0.6438	0.7043	0.6344	0.6180
	Recall	0.9753	0.8101	0.7173	0.7125	0.6267	0.6650	0.5813	0.5851

Table 6
Performance Comparison: PPCA vs Other Dimensionality Reduction Algorithms

Method	Measure	AD/ CU	AD/ MCI	CU/ MCI	AD/ EMCI	AD/ LMCI	CU/ LMCI	CU/ EMCI	LMCI /EMCI
PPCA + MLP	F1	0.9734	0.8954	0.7830	0.8621	0.7790	0.8325	0.72	0.656
	Prec	0.9839	0.9167	0.8214	0.8562	0.7603	0.9086	0.7742	0.6910
	Recall	0.9632	0.875	0.7479	0.8681	0.7986	0.7682	0.6729	0.6244
Truncated SVD + MLP	F1	0.9526	0.9053	0.7734	0.7473	0.7596	0.79	0.6667	0.6062
	Prec	0.9731	0.9673	0.7619	0.7192	0.7466	0.8495	0.7097	0.6573
	Recall	0.9330	0.8508	0.7853	0.7778	0.7730	0.7383	0.6286	0.5625
Kernel PCA + MLP	F1	0.9659	0.8937	0.7598	0.8489	0.7622	0.8082	0.735	0.64
	Prec	0.9859	0.9137	0.7530	0.8082	0.7466	0.8495	0.7903	0.6742
	Recall	0.9436	0.8746	0.7667	0.8940	0.7786	0.7707	0.6869	0.6091

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 7
Performance Comparison: Maxpooled data vs (Maxpooled+Age+Gender)

MLP	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
Maxpooled Data	F1	0.9275	0.8612	0.7527	0.8112	0.7230	0.6976	0.6253	0.6844
	Prec	0.9624	0.8958	0.8155	0.7945	0.7328	0.7258	0.6505	0.7247
	Recall	0.895	0.8292	0.6990	0.8286	0.7133	0.6716	0.6020	0.6482
Maxpooled + Age + Gender	F1	0.9326	0.8632	0.7531	0.8211	0.7569	0.7413	0.6064	0.6813
	Prec	0.9677	0.9018	0.8125	0.8014	0.7466	0.8011	0.6129	0.6966
	Recall	0.9	0.8278	0.7018	0.8417	0.7676	0.6898	0.6	0.6667
Maxpooled + Age + Gender + APOE + FAQ	F1	0.9734	0.8954	0.7830	0.8621	0.7790	0.8325	0.72	0.656
	Prec	0.9839	0.9167	0.8214	0.8562	0.7603	0.9086	0.7742	0.6910
	Recall	0.9632	0.875	0.7479	0.8681	0.7986	0.7682	0.6729	0.6244

Score

Table 8
Addition of AD/CU beta positive/negative with demographic features

Measure	1 HL	2 HL	3 HL	4 HL	5 HL
HL Config	(820)	(820,150)	(820,150,905)	(820,150,905,70)	(820,150,905,70,15)
F_1 score	0.9520	0.9544	0.9558	0.9522	0.9548
Precision	0.9982	0.9844	0.9712	0.9608	0.9642
Recall	0.9098	0.9262	0.9408	0.9438	0.9455

Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

Table 9

Summary of Best Results

Data	Measure	AD / CU	AD / MCI	CU / MCI	AD / EMCI	AD / LMCI	CU / LMCI	CU / EMCI	LMCI /EMCI
No Demo	F_1 score	0.9430	0.8743	0.7527	0.8747	0.7706	0.6976	0.6388	0.6844
Demo	F_1 score	0.9814	0.9125	0.7858	0.9036	0.8288	0.8325	0.72	0.656

Table 10
Classification comparison for Max-Pooled and Mean-Pooled Data

Performance Comparison	Pooling	AD / CU		AD / MCI		CU / MCI		AD / LMCI		CU / LMCI		AD / EMCI		CU / EMCI		LMCI / EMCI	
		Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean
F-1 score	Max	0.92	0.86	0.86	0.75	0.82	0.72	0.72	0.63	0.63	0.64	0.63	0.57	0.61	0.61	0.61	0.61
	Mean	0.93	0.86	0.71	0.85	0.73	0.73	0.73	0.63	0.63	0.64	0.63	0.57	0.61	0.61	0.61	0.61
Precision	Max	0.97	0.90	0.82	0.80	0.80	0.73	0.73	0.62	0.62	0.64	0.62	0.59	0.59	0.59	0.59	0.59
	Mean	0.96	0.89	0.73	0.87	0.87	0.77	0.77	0.62	0.62	0.64	0.62	0.59	0.59	0.59	0.59	0.59
Recall	Max	0.87	0.83	0.70	0.84	0.84	0.71	0.71	0.67	0.67	0.64	0.67	0.60	0.60	0.64	0.64	0.64
	Mean	0.90	0.83	0.70	0.82	0.82	0.71	0.71	0.65	0.65	0.63	0.65	0.55	0.55	0.63	0.63	0.63
NPV	Max	0.82	0.57	0.37	0.88	0.88	0.73	0.73	0.58	0.58	0.60	0.73	0.55	0.55	0.60	0.60	0.60
	Mean	0.87	0.58	0.42	0.77	0.77	0.66	0.66	0.72	0.72	0.70	0.66	0.54	0.54	0.70	0.70	0.70
PPV	Max	0.97	0.90	0.82	0.80	0.80	0.73	0.73	0.73	0.73	0.64	0.73	0.65	0.65	0.64	0.64	0.64
	Mean	0.96	0.89	0.73	0.87	0.87	0.77	0.77	0.62	0.62	0.59	0.77	0.60	0.60	0.59	0.59	0.59

Table 11
Estimating an Optimal Configuration for AD vs. CU Classification

#HL	AD vs CU		AD vs CU with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(700)	0.9430	(525)	0.9563
2	(700,555)	0.9415	(525,880)	0.9737
3	(700,555,305)	0.9403	(525,880,880)	0.9761
4	(700,555,305,25)	0.9368	(525,880,880,255)	0.9812
5	(700,555,305,25,10)	0.9393	(525,880,880,255,775)	0.9814

Table 12
Estimating an Optimal Configuration for AD vs. MCI Classification

#HL	AD vs MCI		AD vs MCI with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(85)	0.8684	(160)	0.9086
2	(85,120)	0.8727	(160,270)	0.9125
3	(85,120,110)	0.8743	(160,270,405)	0.9083
4	(85,120,110,625)	0.8734	(160,270,405,350)	0.9086
5	(85,120,110,625,120)	0.8677	(160,270,405,350,215)	0.9140

Table 13
Estimating an Optimal Configuration for EMCI vs. AD Classification

#HL	EMCI vs AD		EMCI vs AD with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(755)	0.8696	(80)	0.9003
2	(755,55)	0.8736	(80,190)	0.9011
3	(755,55,625)	0.8747	(80,190,380)	0.9036
4	(755,55,625,15)	0.8641	(80,190,380,425)	0.9000
5	(755,55,625,15,585)	0.8644	(80,190,380,425,550)	0.8950

Table 14
Estimating an Optimal Configuration for LMCI vs. AD Classification

#HL	LMCI vs AD		LMCI vs AD with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(380)	0.7561	(215)	0.8288
2	(380,660)	0.7706	(215,105)	0.8193
3	(380,660,70)	0.7688	(215,105,150)	0.8098
4	(380,660,70,55)	0.7679	(215,105,150,600)	0.8086
5	(380,660,70,55,535)	0.7580	(215,105,150,600,260)	0.8086

Table 15
Estimating an Optimal Configuration for LMCI vs. CU Classification

#HL	LMCI vs CU		LMCI vs CU with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(915)	0.6512	(560)	0.7324
2	(915,20)	0.6536	(560,490)	0.7774
3	(915,20,690)	0.6539	(560,490,35)	0.7747
4	(915,20,690,170)	0.6471	(560,490,35,40)	0.7735
5	(915,20,690,170,15)	0.6507	(560,490,35,40,75)	0.7671

Table 16
Estimating an Optimal Configuration for EMCI vs. CU Classification

#HL	EMCI vs CU		EMCI vs CU with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(5)	0.6044	(930)	0.6564
2	(5,25)	0.6192	(930,860)	0.6866
3	(5,25,5)	0.6388	(930,860,130)	0.6961
4	(5,25,5,185)	0.6287	(930,860,130,385)	0.7015
5	(5,25,5,185,5)	0.6293	(930,860,130,385,750)	0.6859

Table 17
Estimating an Optimal Configuration for EMCI vs. LMCI Classification

#HL	EMCI vs LMCI		EMCI vs LMCI with Demo	
	Config	F_1 Score	Config	F_1 Score
1	(525)	0.6258	(125)	0.6214
2	(525,360)	0.6275	(125,585)	0.6205
3	(525,360,285)	0.6032	(125,585,5)	0.6467
4	(525,260,285,250)	0.5828	(125,585,5,430)	0.6519
5	(525,360,285,250,750)	0.6024	(125,585,5,125)	0.6103