Supplementary Information

Table of Contents

I. BRAIN IMAGING ANALYSIS

Image Processing Pipeline*:*

Image processing was performed to obtain total brain tissue volumes, regional brain tissue volumes, and cortical surface measures.

A) *Initial preprocessing steps*

1. Rigid body co-registration of both T1w and T2w data to a prior atlas template in pediatric MNI-space generated from 66 one-year old subjects from this study (see Kim et al[1] for more detail).
2. Correction of intensity inhomogeneity via N4[2]
3. Correction of geometric distortions for optimal processing of multi-site longitudinal data[3]

B) *Tissue segmentation*

Brain volumes were then obtained using a framework of atlas-moderated expectation-maximization that performs the following steps (see Kim et al[1] or more detail):

1. Co-registration of T2w data to T1w data

2. Deformable registration of a prior template and propagation of prior tissue probability maps for white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) from MNI space into individual T1w data. The template images are population averages computed at 6, 12 and 24 months of age[4]
3. For 12 month old data only: T1w and T2w intensities were locally adapted using a prior intensity growth map that quantifies the expected change in intensity from 1 to 2 years of age[1]
4. Expectation Maximization based tissue segmentation including parametric intensity inhomogeneity correction[1]
5. Brain masking was performed using the tissue segmentation as masking. If necessary manual correction of the brain masks were performed.

- Steps 1-5 were computed using the AutoSeg toolkit (http://www.nitrc.org/projects/autoseg/).
- ICV was defined as the sum of WM, GM, and CSF.
- Total brain tissue volume (TBV) was defined as the sum of WM and GM.
- Segmentation of 6 month old data did not yield reliable separation of WM and GM, and thus no individual analysis of either was performed here.

C) *Cortical surface measures*

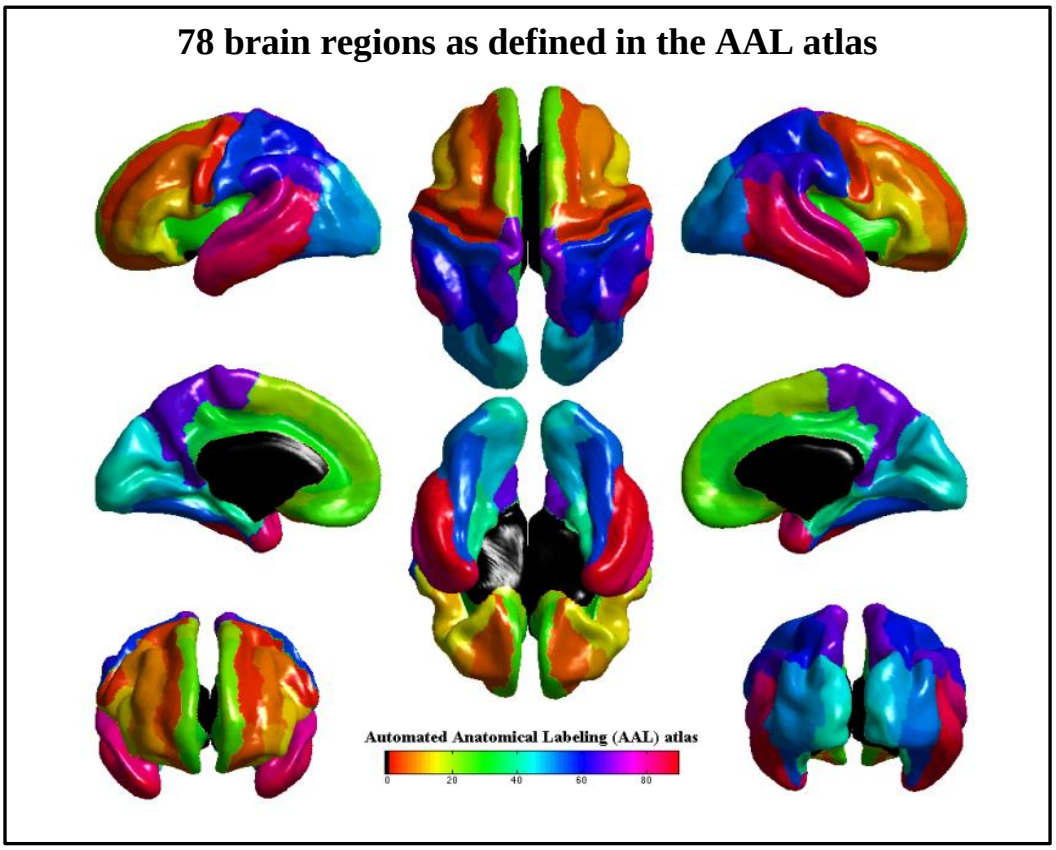
Cortical thickness (CT) and surface area (SA) measures for 12 and 24 month data were obtained via a CIVET workflow[5,6] adapted for this age using an age-corrected, automated, anatomical labeling (AAL) atlas[7]. CIVET includes shrink-wrap deformable surface evolution of WM, local Laplacian distance and local SA, mapping to spherical domain, co-registration using cortical sulcal features and extraction of regional measurements via a deformably co-registered fine-scale lobar parcellation[7]. SA was measured at the mid-cortical surface averaged from the computed white and pial surfaces. Both white and pial surfaces were visually QC'ed with surface cut overlay on the MRI images.

CIVET was applied as described above to 12 month and 24 month data following tissue segmentation with AutoSeg (step B).

For 6 month old datasets: As tissue segmentation (step B) for 6 month old subjects did not yield reliable white vs. gray matter segmentations, cortical surfaces at 6 months were determined longitudinally, only for subjects with MRI data at both 6 month and 12 month visits. Using ANTs[8] deformable diffeomorphic symmetric registration with normalized cross correlation (metric radius 2mm, Gaussian smoothing of 3 mm of the deformation map) of joint T1w and T2w data (both image sources were equally weighted), the pre-processed, brain masked MRI data of 12 month old subjects was registered to data from the same subject at age 6 months. This registration was applied to the cortical surfaces of the 12 month subjects to propagate them into the 6 month old space. Surfaces at 6 months were then visually QC'ed with surface cut

overlay on the MRI images. Local SA and CT measures at 6 month were finally extracted from these propagated surfaces.

Regional measures: Thickness measurements were averaged for all vertices within each of the 78 brain regions in the AAL label atlas[7]. The surface area measurements were summed over all vertices of each AAL region for a total regional surface area. These are illustrated below in Figure 1 (AAL is publically available http://www.bic.mni.mcgill.ca/ServicesSoftware/VisualGuides).



II. MACHINE LEARNING ANALYSIS

1. Rationale for non-linear classification

Children with autism show an abnormal brain growth trajectory that includes a period of early overgrowth that occurs within the first year of life. In particular, the total size of the brain is up to 10 percent larger in children with autism than typically developing children[9]. Thus the brain of a child with autism will typically show larger morphological differences such as intra-cranial cortical volume, cortical surface area, and cortical thickness. To estimate these growth trajectories, or *growth patterns*, linear models[9,10] have been applied, however these are primarily concerned with changes in total volume and not specific regions in the brain, or how the growth patterns in specific brain regions may differentiate typically developing children with autism. Also, as suggested by Giedd,[11] linear models may not accurately represent complex (i.e. non-

linear) growth patterns in the developing brain. As a result, classification methods that use linear models to recognize these complex brain growth patterns may not be the most suitable choice.

Recently, deep learning (DL)[12-14] has been used to estimate low dimension codes (LDC) that are capable of encoding latent, non-linear relationships in high dimension data. Unlike linear models such as PCA, kernel SVM, local linear embedding, and sparse coding, the general concept of deep learning is to learn highly compact hierarchical feature representations by inferring simple ones first and then progressively building up more complex ones from the previous level. To better understand these complex brain growth patterns within the first year of life, DL is a natural choice because the hierarchical deep architecture is able to infer complex non-linear relationships, and the trained network can quickly and efficiently compute the low dimension code for newly created brain growth data.

2. Methods and materials

2.1 High dimension feature vector

Each child in the training and test populations is represented by a *d=315* dimension feature vector that includes longitudinal cortical surface area (CSA), cortical thickness (CTH), intra-cranial volume (ICV) real-valued morphological measurements, and one binary sex feature (sex Male or Female). Specifically, features 1-78 are CSA measurements at 6 months, and features 79-156 are CSA measurements at 12 months. Each measurement corresponds to a brain region defined in the AAL atlas (i.e. 39 regions in the right hemisphere, and 39 regions in the left hemisphere as illustrated above). Likewise, features 157-234 are CTH measurements at 6 months, and features 235-312 are CTH measurements at 12 months. Lastly, features 313 and 314 correspond to ICV measurements at 6 and 12 months respectively, and feature 315 is the sex of the subject (1=Male, 0=Female).

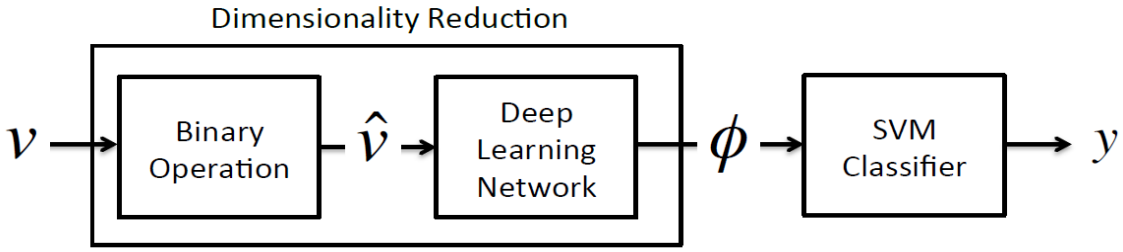
Lastly, because the infant brain develops so quickly within the first year of life, to adjust for differences in the actual age at acquisition for the 6 month and 12 month visit (i.e. the "6 month'' image acquisition of a subject does not exactly occurs at 6.0 months, but rather at, e.g., 5.7 or 6.2 months), all morphological measurements in the 315 dimensional feature vector are linearly age-normalized. Specifically, the 6-month CSA, CTH, and ICV measures are multiplied by $\Delta_{6M} = 6/a_{6M}$, where a_{6M} is the subject's actual age at "6 month" image acquisition. And the 12-month CSA, CTH, and ICV measures are multiplied by $\Delta_{12M} = 12/a_{12M}$, where a_{12M} is the subject's actual age at the "12 month" image acquisition.

2.2 Prediction Pipeline

To better recognize brain overgrowth patterns in autistic children within the first year of life the proposed prediction pipeline uses the 2-stage design illustrated in Figure 2 to recognize complex morphological patterns in different brain regions that are likely to differentiate children with autism from typically developing children. Specifically, our prediction pipeline includes a DL

based dimensionality reduction stage followed by a SVM classification stage. In general, the chosen two-stage design is a very common configuration (low dimension feature representation stage followed by a classification stage), and has been used in several state-of-the-art pipeline designs[16-20] that *only* use DL to approximate a new lower dimension feature representations, or LDC representations in our case, that are then used to train a separate classifier. To ensure the trained two-stage pipeline had optimal classification performance, a median DL network $\bar{N}$ is constructed using only those trained DL networks that resulted in a two-stage pipeline with PPV and accuracy scores $\geq 90\%$ as measured on the training data only. For more details about how the median network was constructed see **Median Network** section.

Figure 2. Proposed Two-stage prediction pipeline that includes a non-linear dimension reduction step followed by a SVM classification step



In general, the prediction pipeline is applied as follows (assuming that the prediction pipeline has been trained): First, given a high dimension feature vector v not included in the training data set, the real-valued vector is transformed into a binary feature vector $\hat{v}$ using a trained binary masking operation, then the low dimensional code (LDC) ϕ is estimated using a trained DL network. Next the diagnosis label y for ϕ is found using a trained SVM classifier. The technical details of each stage are outlined below.

The decision to implement a 2-stage prediction pipeline was primarily driven by two different design considerations: 1) creation of a self-contained non-linear dimensionality reduction algorithm that did not incorporate a classification mechanism, and 2) a flexible design where different classification algorithms, such as a support vector machines or distance weighted discrimination (DWD) classifiers, can be applied with little to no effort.

It is noteworthy that the dimensionality reduction stage in the proposed pipeline could be reconfigured to become a deep classification network.

Non-Linear Dimensionality Reduction

Given a $n\ x\ d$ training data matrix $A = \{\boldsymbol{a}_1, \boldsymbol{a}_1, \dots, \boldsymbol{a}_i, \dots \boldsymbol{a}_n\}$ of n subjects where row vector $\boldsymbol{a}_i = (a_{i1}, \dots, a_{id})$ represents the high dimension feature vector for subject i, and $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ is a n dimension vector that defines the binary group label for each subject in the training data set (i.e. the paired diagnosis information for row vector $\boldsymbol{a}_i$ is y_i, 0=HR-ASD and 1=HR-neg in our study), a binary operation $f: \mathbb{R}^d \rightarrow \{0,1\}^d$ is first applied to transform each

real-valued measurement $\boldsymbol{a}_{jd}$, into a binary one. Specifically, for each column vector $\boldsymbol{a}_j$, $j = 1, \ldots, d$ in A, an average median threshold $m_j = (m_j^{l_0} + m_j^{l_1})/2$ is calculated where $m_j^{l_0}$ is the median value of feature j and diagnosis label l_0, and $m_j^{l_1}$ is the median value of feature j and diagnosis label l_1. Values in $\boldsymbol{a}_j$ that are less than or equal to m_j are set to 0, and those that are greater than m_j are set to 1.

Even though more sophisticated binary operations, such as local binary patterns (LBP), exist they are typically used in DL networks that attempt to classify gray-level texture patterns representing local image patches[21,22]. In our case, our features are not image patches, so the LBP approach is not appropriate for application. Furthermore, it is very common to perform an unsupervised training procedure on individual two-layer greedy networks (such as restricted Boltzman machines and auto-encoders) using visible layer features that are binary and not real-valued[12,19]. In fact, there is evidence (suggested by Tijmen Tieleman), that when binary features are used in the visible layer this may reduce sampling noise that allows for fast learning[23]. Lastly, the strategy behind the binary operation chosen for this application is directly related to the overgrowth hypothesis. More specifically, we are more interested in something that is posed as a "binary" question. That is, is the value of this feature greater than a median value (possible overgrowth is true), or less than median value (possible overgrowth is false)? This approach allows us to train the DL network using simple binary patterns instead of complex real-valued ones; as a result, the binary operation may allow us to train the DL network faster (i.e. faster convergence, and as the reported results show, even though we may lose some information when the binary operation is applied, the prediction performance of the two-stage pipeline speaks for itself.

When the binary operation completes a new training population $\hat{A} = \{\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_1, \ldots, \hat{\boldsymbol{a}}_n\}$ is created where each binary high dimension feature vector now describes inter-subject brain region (cortical surface area and thickness) growth patterns, and inter-subject ICV growth patterns.

Next, the binary high dimension features in $\hat{A}$ are used to train a *deep network* that is implemented using the publically available deep learning Matlab toolbox called DeepLearn Toolbox*[1] In general, the deep network is trained by performing the two sequential steps listed below.

1. As illustrated in Figure 3(a) below, an unsupervised step is first performed that sequentially trains individual autoencoders (AE). In particular, an AE is a 2-layer bipartite graph that has a visible layer (input) and hidden layer (output). Initially, the edge

*[1] *https://github.com/rasmusbergpalm/DeepLearnToolbox*

weights (represented by a matrix W) in the bipartite graph are randomly chosen, and then iteratively refined[12] (including a set of bias values) using data step

$$\boldsymbol{h} = \sigma(W\boldsymbol{v} + \boldsymbol{\beta_d})$$

followed by a reconstruction step

$$\tilde{\boldsymbol{v}} = \sigma(W^t\boldsymbol{h} + \boldsymbol{\beta_r})$$

where vector $\boldsymbol{v}$ represents the nodes in the visible layer, vector $\boldsymbol{h}$ represents the nodes hidden layer, vectors $\boldsymbol{\beta_d}$ and $\boldsymbol{\beta_r}$ represent the bias values for the data and reconstruction steps, vector $\tilde{\boldsymbol{v}}$ represents the reconstructed nodes in the visible layer, and $\sigma(\cdot)$ is the sigmoid function. In general, the AE convergence criteria is:

- The maximum number of epochs is reached (that is set to 100), or
- The average mean square error (MSE), $MSE_e = 1/\delta \sum_{i=1}^{\delta}(\tilde{v}_i^e - \tilde{v}_i^{e-1})^2$ for the last 10 consecutive epoch (i.e., $e = 1,2 \ldots 10$) is less than 0.01, where $\tilde{v}_i^e$ is the reconstructed visible layer at epoch e, and $\tilde{v}_i^{e-1}$is reconstructed visible layer at epoch $e - 1$.

Figure 3. Deep network training

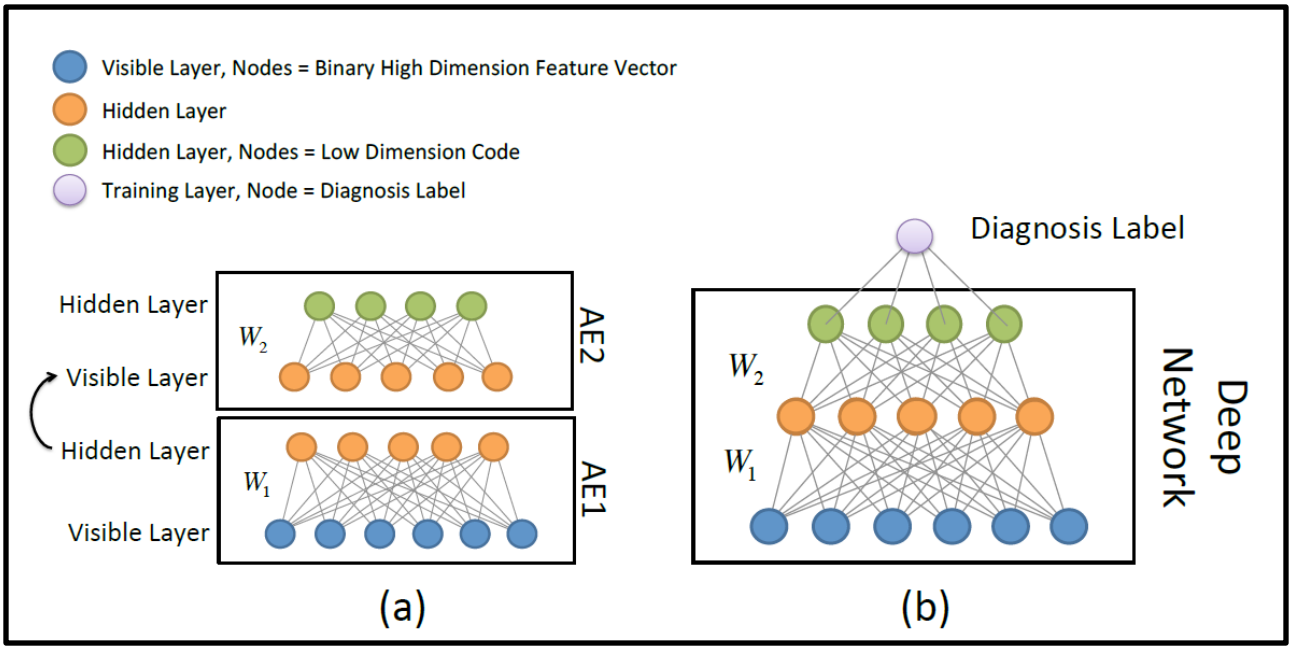
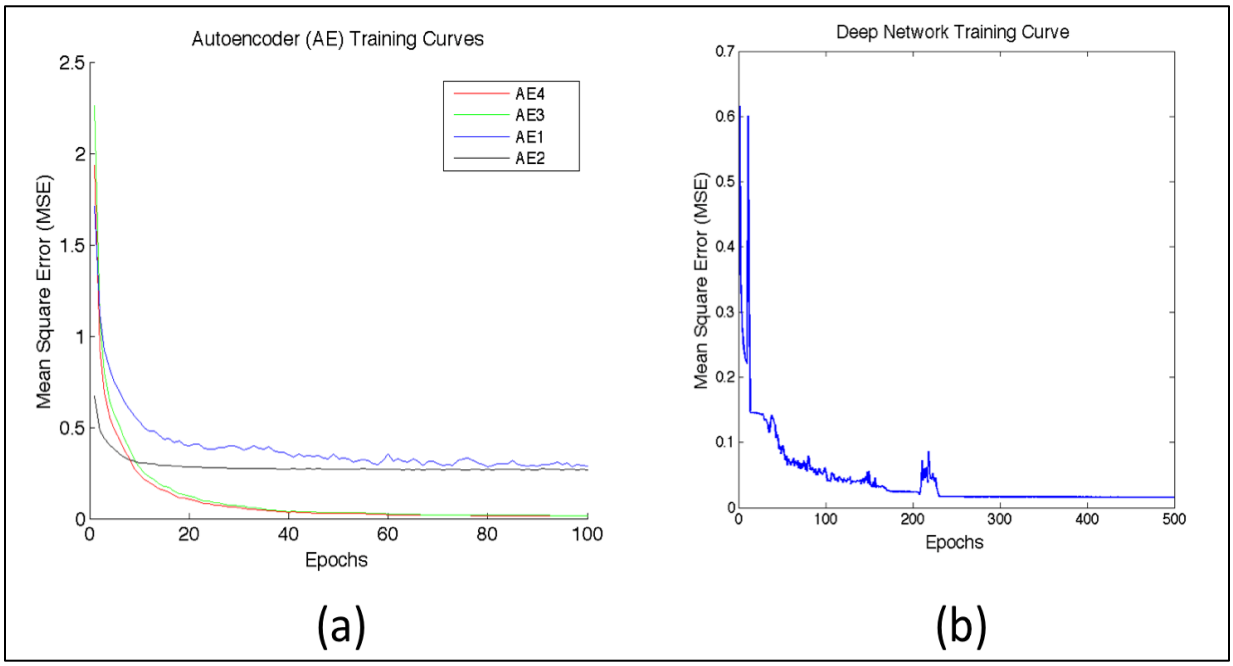


Figure 4 shows a typical training curve (4a) for each AE in the proposed 4-layer architecture. When the AE training step terminates and the hidden layer in current AE becomes the visible layer in the next AE, and the unsupervised process repeats itself for each AE in the deep network. In general, the goal of the unsupervised step is to

approximate edge weight and bias values that increase the likelihood of finding the global optimum, or at least a very good local minimum, during the supervised step.

Figure 4. Autoencoder and deep network training curves

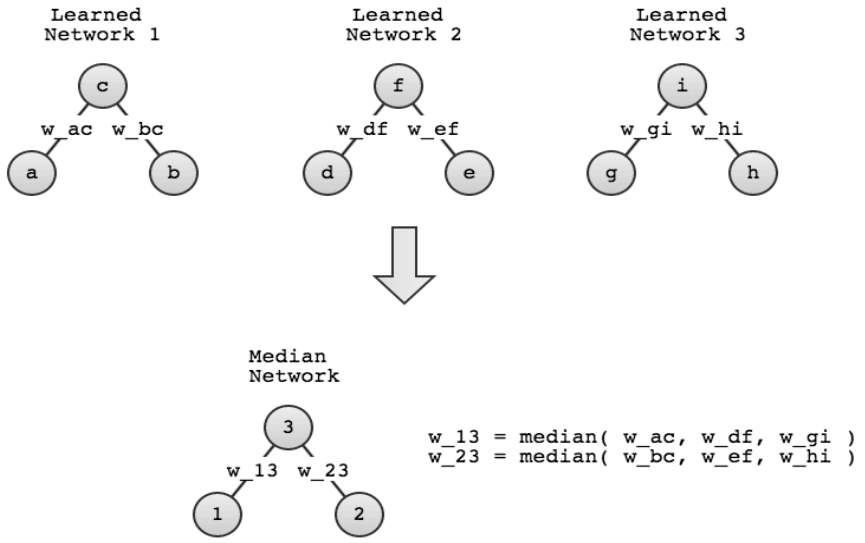


2. As illustrated in Figure 3(b), the supervised step stacks the initialized AEs (i.e. creating the deep network) and then adds one additional layer for the supervised training only (i.e. training label layer) that contains the binary diagnosis label for each binary high dimension feature vector in the training population. At this point the deep network is treated as a traditional feed forward neural network that uses back propagation to fine-tune the initial weight values in each AE (see Figure 4(b) for a typical training curve at this stage). Once the supervised step completes, the training label layer is removed, and the number of nodes in last hidden layer represents the final dimension of the output LDC.

Median Network

Since the initial edge weights in unsupervised step are randomly chosen, it is likely that the result of the training procedure depends on that random initialization. To mitigate this problem the deep network procedure was performed with 10,000 random initializations resulting in a set of fine-tuned networks $\{N_1, N_2, \ldots, N_{10,000}\}$ (all trained with the same training set). Using the edge weight matrices and bias values in each fine-tuned network in $\{N_1, N_2, \ldots, N_{10,000}\}$, an initial median network is constructed. The median calculation is illustrated in Figure 5 using a simple two-layer network that only defines three nodes (note: w_ac represents the edge weight that connects node a and node c).

Figure 5. Illustration that shows how the edge weights in the LD network were calculated



Once the initial median network is constructed the edge weight and bias values are further refined by one additional, final supervised training step. That is, the median weights become the initial weights (i.e. starting weights for back propagation optimization), thus giving us a potentially more robust starting point for the supervised training step that may lead to a fine-tuned deep network that is highly reproducible. Finally, the fully trained median deep network $\overline{N} = \left\{\{\overline{W}_1, \overline{\boldsymbol{\beta}}_1\}, \{\overline{W}_2, \overline{\boldsymbol{\beta}}_2\}, \ldots, \{\overline{W}_k, \overline{\boldsymbol{\beta}}_k\}\right\}$ is then used to estimate a LDC $\boldsymbol{\phi}_i$ for subject $\hat{\boldsymbol{a}}_i$.

The deep learning parameters momentum and learning rate were set to 0.7 and 1.25 respectively, and a four-layer architecture [315 100 10 2] was used, where the first layer, or input layer represented by a binary high dimension feature vector, defines 315 nodes, second hidden layer defines 100 nodes (approximately 70% feature reduction), the third hidden layer defines 10 nodes (90% feature reduction), and the last layer, or output LDC layer, defines 2 nodes (80% feature reduction). These parameters were chosen initially as suggested in UTML TR 2010–003, *A Practical Guide to Training Restricted Boltzmann Machines*[23], Geoffrey Hinton, and then experimentally refined using only the training data.

SVM Classifier

The LDCs $\{\boldsymbol{\phi}_i, i = 1, \ldots, n\}$ generated by the trained deep learning network along with the binary training labels $\{\boldsymbol{y}_i, i = 1, \ldots, n\}$ are then used to train a two-class SVM classifier that uses a

linear kernel. The SVM classifier was implemented using the SVM Matlab toolbox*[2] that is based on well-known LIBSVM library.

Once the SVM classifier is trained, the clinical outcome of a high dimension feature vector, say $\boldsymbol{v} = (v_1, \dots, v_d)$ not in the training data set can be predicted using the sequence of steps outlined below.

1. Estimate binary high dimension feature vector $\widehat{\boldsymbol{v}} = (v_1 m_1, v_2 m_2, \dots, v_d m_d)$, where $\{m_1, m_2, \dots, m_d\}$ is the learned median threshold values for each feature defined in the high dimension feature vector.
2. Estimate the LDC $\boldsymbol{\phi} = (\phi_1, \phi_2, \dots, \phi_\psi)$ for $\widehat{\boldsymbol{v}}$ using the median deep network $\bar{N}$, where ψ represents the number of nodes in the last hidden layer.
3. Calculate predicted diagnosis label

$$y = \sum_{i=0}^{\psi} \alpha_i \kappa(\boldsymbol{\delta}_i, \boldsymbol{\phi}) + b$$

where α are the weights, $\boldsymbol{\delta}$ are the support vectors, $\kappa(\cdot)$ is the inner product of the two vectors, and b is the bias that define the linear hyper-plane (decision boundary) learned by the SVM algorithm. The sign of the calculated prediction value (i.e. $y \geq 0$ or $y < 0$) determines which of the two diagnosis labels the subject has been assigned.

2.3 Using trained HR-ASD/HR-neg pipeline to predict LR subjects

To further validate the prediction accuracy, we trained an HR-ASD/HR-neg pipeline on all HR subjects and classified all low risk (LR) subjects (no LR subjects were included in the training data set). In general, for these subjects the diagnosis label predicted by the HR-ASD/HR-neg pipeline should be HR-neg.

To be precise, we first describe here the general process of computing SVM based diagnostic labels from a high dimension feature vector, $\boldsymbol{v} = (v_1, \dots, v_d)$ (not in the training data) using the sequence of steps outlined below.

1. Estimate binary high dimension feature vector $\widehat{\boldsymbol{v}} = (v_1 m_1, v_2 m_2, \dots, v_d m_d)$, where $\{m_1, m_2, \dots, m_d\}$ is the learned median threshold values for each feature defined in the high dimension feature vector.
2. Estimate the LDC $\boldsymbol{\phi} = (\phi_1, \phi_2, \dots, \phi_\psi)$ for $\widehat{\boldsymbol{v}}$ using the learned deep network $\bar{N}$, where ψ represents the number of nodes in the last hidden layer ($\psi = 2$ in our case).
3. Calculate predicted SVM diagnosis label

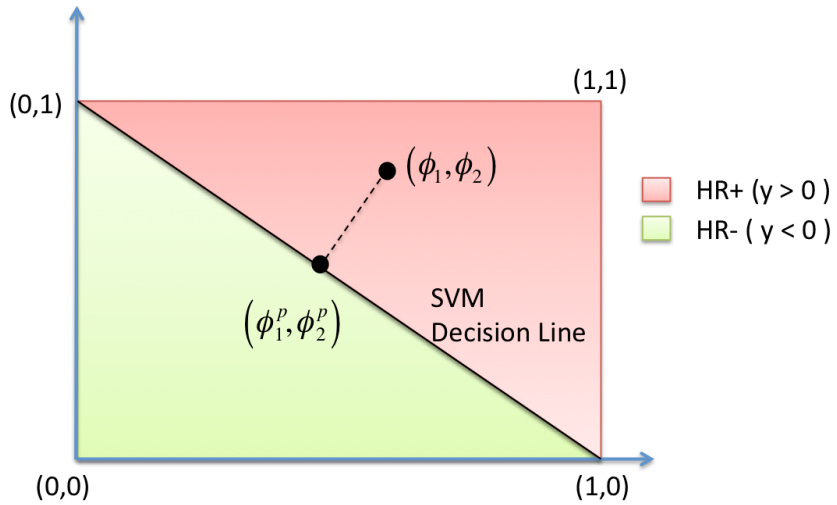
*[2] *http://www.mathworks.com/help/stats/support-vector-machines.html*

$$y = \sum_{i=0}^{\psi} \alpha_i \kappa(\boldsymbol{\delta}_i, \boldsymbol{\phi}) + b$$

where α are the weights, $\boldsymbol{\delta}$ are the support vectors, $\kappa(\cdot)$ is the inner product of the two vectors, and b is the bias that define the linear hyper-plane (decision boundary) learned by the SVM algorithm. The sign of the calculated prediction value (i.e. $y \geq 0$ or $y < 0$) determines which of the two diagnosis labels the subject has been assigned.

Using the learned SVM parameters $\{\boldsymbol{\alpha}, \boldsymbol{\delta}, b\}$ that describe the linear decision boundary, we can now assess the strength or weakness of the prediction by the trained classification pipeline via the distance *d* to the SVM decision boundary, as schematically illustrated in Figure 6 using an example 2D SVM decision space.

Figure 6. Example 2D SVM decision boundary that illustrates the procedure to compute a subject's distance from the decision boundary.



This distance *d* is computed using the following two steps:

- Step 1: Take LR LDC $\boldsymbol{\phi} = (\phi_1, \phi_2)$ and project $\boldsymbol{\phi}^p = (\phi_1^p, \phi_2^p)$ onto into SVM decision line.
- Step 2: Calculate the Euclidean distance $d(\boldsymbol{\phi}. \boldsymbol{\phi}^p)$ between two points using learned SVM decision parameters $\{\boldsymbol{\alpha}, \boldsymbol{\delta}, b\}$.

In general, the calculated distance can be interpreted as follows:

- A distance close to zero would indicate a weak/unsure classification
- A distance that is highly negative is a strong/safe classification for HR-neg
- A distance that is highly positive is a strong/safe classification for HR-ASD

This analysis included 84 LR- subjects. No LR-ASD subjects with both available 6 and 12mo surface area and cortical thickness data were available. Of the 84 LR subjects, 76 were classified (correctly) as HR-neg and 8 were classified (incorrectly) as HR-ASD. Some observations:

1. 90% (76/84) LR subjects were correctly classified as HR-neg
2. 10% (8/84) LR subjects were incorrectly classified as HR-ASD
3. 8% (7/84) LR subjects are (incorrectly) considered "safe" HR-ASD (at a distance higher than 0.15 from the decision boundary)
4. 7% (6/84) LR subjects are close to the decision boundary and thus are not considered clear decisions by our classification method (at a distance smaller than 0.15)
5. 84.5% (71/84) LR subjects are (correctly) considered safe HR-neg (at a distance higher than 0.15 from the decision boundary)
6. All of these subjects are not high-risk subjects and the prediction pipeline was purely trained for separating HR-neg from HR-ASD subjects.

2.4 Prediction pipeline cross-validation classification

The predictive power of the proposed two-stage HR-ASD/HR-neg pipeline is evaluated using a 10-fold cross-validation strategy. In particular, the HR-ASD and HR-neg subjects are first combined into one data set, and then partitioned into 10 different folds, where each fold contains high dimension feature vectors of randomly selected HR-ASD and HR-neg subjects. Furthermore, the ratio of HR-ASD to HR-neg subjects was maintained across each fold. The prediction pipeline (including the DL network generation) is fully re-trained in all its steps using the high dimension feature vector data in 9 of the 10 folds (i.e. a deep network was trained using 9 of the 10 folds) and then tested using the high dimension feature vector data in the remaining (or left out) fold. This iterative process terminates when each fold has been selected as the test one. Using the confusion matrix (TP=true positive, FP=false positive, FN=false negative, and TN=true negative) results in each test fold, the mean and standard error is reported for the specificity, sensitivity, positive predictive value, negative predictive value, and accuracy measures, where sensitivity (SEN) = TP/(TP+FN), specificity (SPE) = TN/(FP+TN), positive predictive value (PPV) = TP/(TP+FP), negative predictive value (NPV) = TN/(TN+FN), and accuracy (ACC) = (TP+TN)/(TP+FN+FP+TN). Table 3 in the main text shows the results of this cross-validation using our prediction pipeline.

Comparison with other classification methods

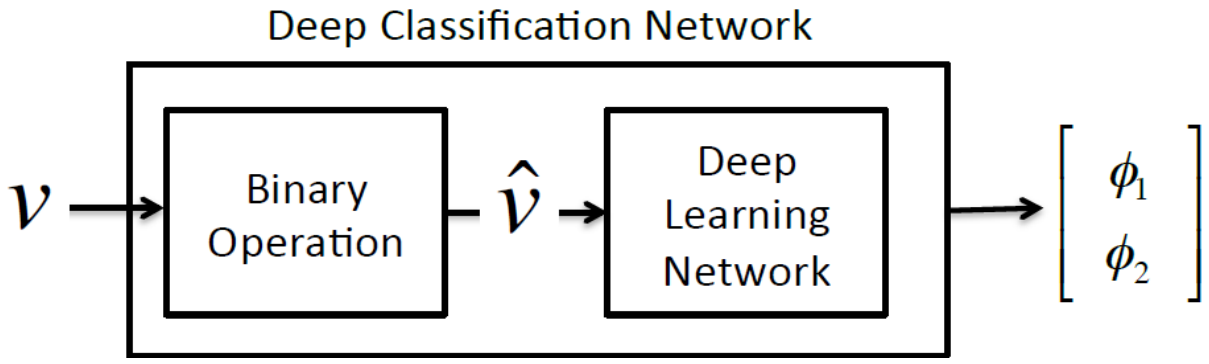
For comparison purposes, using a smaller portion of the dataset (N=133), we compared the proposed two-stage prediction pipeline approach with 3 other approaches: 1) a traditional DBN that does not include a separate classification step, 2) two-stage approach that uses a least squares linear regression algorithm (or sparse learning algorithm, SL) instead of a deep learning one, and 3) two-stage approach that uses principle component analysis (PCA) for dimensionality

reduction. Note: The smaller dataset only includes quality controlled imaging data up to August 2014, and was based purely on the imaging data available at the time of comparison analysis.

Deep classification network

It is noteworthy that the dimensionality reduction and the classification stages in the proposed pipeline could be combined into one stage creating a traditional deep classification network (DCN) as shown in Figure 7.

Figure 7. This figure demonstrates how the prediction pipeline is reconfigured to create a deep classification network. This network does not include a separate classification stage (unlike shown in Figure 2).



In this approach, the number of nodes in the last layer, i.e. ϕ_1 and ϕ_2 in the example DCN shown in Figure 2, represent the diagnosis label (such as HR-ASD and HR-neg). Specifically, after the deep network is trained (see **Non-linear dimension reduction** section) an unknown high dimension feature vector is input into the deep network, and if the subject is HR-ASD then $\phi_1 > \phi_2$, otherwise the subject is ASD-neg. In this approach, the binary operation, deep learning parameters, and four-layer architecture outlined in the non-linear dimension reduction section was performed, i.e. exactly without any modifications.

Sparse learning and SVM classifier

The elastic net algorithm[15] is used to find $\boldsymbol{x}$ a sparse weight vector that minimizes

$$\min_x \frac{1}{2}\|Ax - y\| + \frac{\rho}{2}\|x\|_2^2 + \lambda\|x\|_1$$

where $\lambda\|x\|_1$ is the l_1 regularization (sparsity) term, $\frac{\rho}{2}\|x\|_2^2$ is the l_2 regularization (over-fitting) term. For this approach no binary operation is performed on matrix *A*, instead values are normalized as follows: $A(i,j) = (a_{ij} - \mu_j)/\sigma_j$ where a_{ij} is the feature *j* for subject *i*, μ_j is the mean value for column vector *j* in matrix *A*, and σ_j is the standard deviation for column vector *j*

in matrix *A*. The above equation is optimized using the *LeastR* function in the Sparse Learning with Efficient Projections software package*[3]. After optimization, $\boldsymbol{x}$ has weight values in [0 1] where 0 indicate network nodes that do not contribute to the clinical outcome, and weight values greater than zero indicate network nodes that do contribute to the clinical outcome. In general, $\boldsymbol{x}$ is referred to as the *sparse representation* of training data set. Lastly, in our approach each weight value in $\boldsymbol{x}$ greater than zero is set to one, therefore the resulting sparse representation is a binary mask, that is, the network node is turned on (value of 1) or turned off (value of 0). A new $n\ x\ d$ sparse training matrix $\hat{A} = \{\hat{\boldsymbol{a}}_1, \hat{\boldsymbol{a}}_1, \dots, \hat{\boldsymbol{a}}_n\}$ is created, where row vector $\hat{a}_i = (a_{i1}x_1,\ a_{i2}x_2, \dots, a_{id}x_d)$. Lastly, the row vectors in the newly created sparse training matrix by the along with the binary training labels $\{\boldsymbol{y}_i, i = 1, \dots, n\}$ are then used to train a two-class linear kernel SVM classifier.

The sparse learning parameters λ and ρ were set to 0.5 and 1.0 respectively. In each fold, the sparse learning algorithm selected approximately 120 features from 315, which accounts for a 60% feature reduction. For all the reported results, a linear kernel SVM classifier was trained, and default parameters were used.

PCA and SVM classifier

In order to further compare our results to standard Principal Component Analysis (PCA), we performed the following analysis:

1. For PCA no binary operation is performed on matrix *A*, yet normalization is still necessary. Input values are normalized as follows: $A(i,j) = (a_{ij} - \mu_j)/\sigma_j$ where a_{ij} is the feature *j* for subject *i*, μ_j is the mean value for column vector *j* in matrix *A*, and σ_j is the standard deviation for column vector *j* in matrix *A*.
2. Because there are fewer observations than features PCA was computed via singular value decomposition (as is standard in this case), *[U,S,V]=svd(A)*, is performed on *A* that in turn finds a matrix of right singular vectors (*U*), a matrix of left singular vectors (*V*), and the singular value matrix (*S*).
3. Two different dimension reductions approaches are used: 1) Only include the largest eigenvalue (i.e. keep λ_1 and the remaining singular values are set to zero) and 2) the relative variance of each eigenvalue is computed and only eigenvalues greater than 1% of the total variation are kept, and all other eigenvalues are set to zero. For option (2), the largest 101-109 eigenvalues were selected in the 10-fold analysis.

Lastly, the PCA loads for the reduced set of eigenmodes are computed, and used along with the binary training labels $\{\boldsymbol{y}_i, i = 1, \dots, n\}$ to train a two-class linear kernel SVM classifier. For all

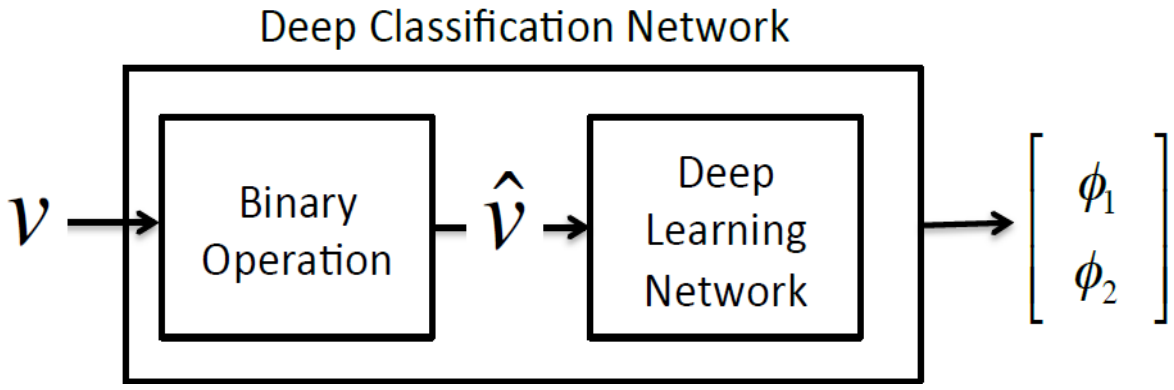
*[3] *http://www.public.asu.edu/~jye02/Software/SLEP*

the reported results, a linear kernel SVM classifier was trained, and default parameters were used.

Comparison results

Table 1 in Supplementary Information shows the results of the 10-fold cross-validation of HR-ASD vs HR-neg using the different methods discussed above. Specifically, DL+SVM is the proposed prediction pipeline; SL+SVM uses sparse learning (SL) for dimensionality reduction (i.e. feature selection) instead of DL; a deep classification network (DCN) that does not include a separate classification stage (see Figure 8), and the two proposed PCA+SVM classifications.

Figure 8. Pipeline prediction pipeline reconfigured to become a deep classification network, i.e., prediction pipeline that does not include a separate classification stage



The total number of subjects in the data set is 133 (27 HR-ASD subjects and 106 HR-neg subjects). The data set was partitioned into 10-folds where each fold has 12, 13, or 14 randomly selected subjects. Furthermore, the ratio of HR-ASD to HR-neg subjects in each fold is kept constant, or as similar as possible, to control the base rate and maintain the consistency of the learned decision boundary/space. Lastly, the binary diagnostic training labels used to train the prediction pipeline (i.e. deep network and SVM classifier) are 0=HR-ASD and 1=HR-neg.

The proposed prediction pipeline outperforms the other classification approaches as can be seen in Table 1 below.

Table 1. Results of 10-fold cross validation procedure on a reduced dataset (N=133).

	PPV		NPV		SENS		SPEC		ACC	
Prediction Model	**AVE**	**STD ERR**	**AVE**	**STD ERR**	**AVE**	**STD ERR**	**AVE**	**STD ERR**	**AVE**	**STD ERR**
DL+SVM	82%	3.22%	95%	0.93%	78%	3.87%	94%	0.97%	91%	0.92%
SL+SVM	48%	4.21%	92%	1.42%	77%	4.22%	75%	3.29%	75%	3.16%
DCN	62%	4.46%	90%	0.16%	62%	1.61%	89%	1.47%	83%	1.17%
PCA+SVM (Approach 1)	22%	3.75%	81%	1.20%	30%	4.66%	74%	1.84%	65%	1.87%
PCA+SVM (Approach 2)	19%	3.91%	80%	1.41%	27%	5.02%	72%	1.89%	63%	2.14%

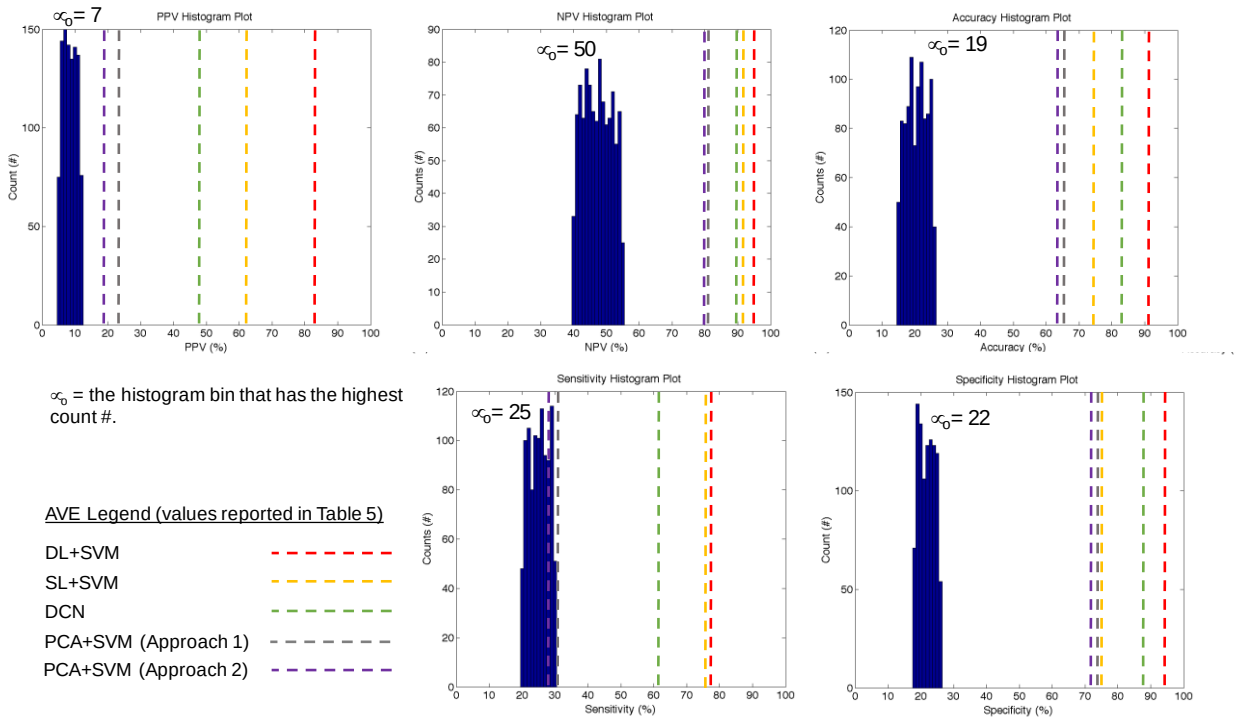
Key: DL = deep learning, SVM = support vector machine (classifier), SL = sparse learning, DCN = deep classification network, PCA – principal component analysis, PPV = positive predictive value, NPV = negative predictive value, Sens = sensitivity, Spec = specificity, ACC = accuracy

2.5 HR-ASD/HR-neg pipeline prediction analysis using random diagnosis labels

Furthermore, we performed permutation based random diagnosis evaluation to check whether our prediction pipeline results are optimistically biased. We applied the above described 10-fold cross validation to randomly scrambled diagnosis labels via standard permutation analysis (both the deep network and the SVM classifier were retrained at every permutation). The random label scrambling was achieved using the Matlab random permutation (*randperm*) function.

We employed 1,000 permutations, and the results for each measure (PPV, NPV, Accuracy, Sensitivity, and Specificity, average values of the 10-fold analysis) is visualized in the histogram plots shown in Figure 9.

Figure 9. Histogram plots of permutation analysis



As seen in in Figure 9, the range of values over the 1,000 runs for the: 1) PPV measure was between 5 and 12 percent, 2) the NPV measure was between 40 and 55 percent, 3) the Accuracy measure was between 15 and 26 percent, 4) the Sensitivity measure was between 20 and 30 percent, and 5) the Specificity measure was between 18 and 26 percent. These results suggest the reported performance of our HR-ASD/HR-neg pipeline is identifying a pattern of longitudinal

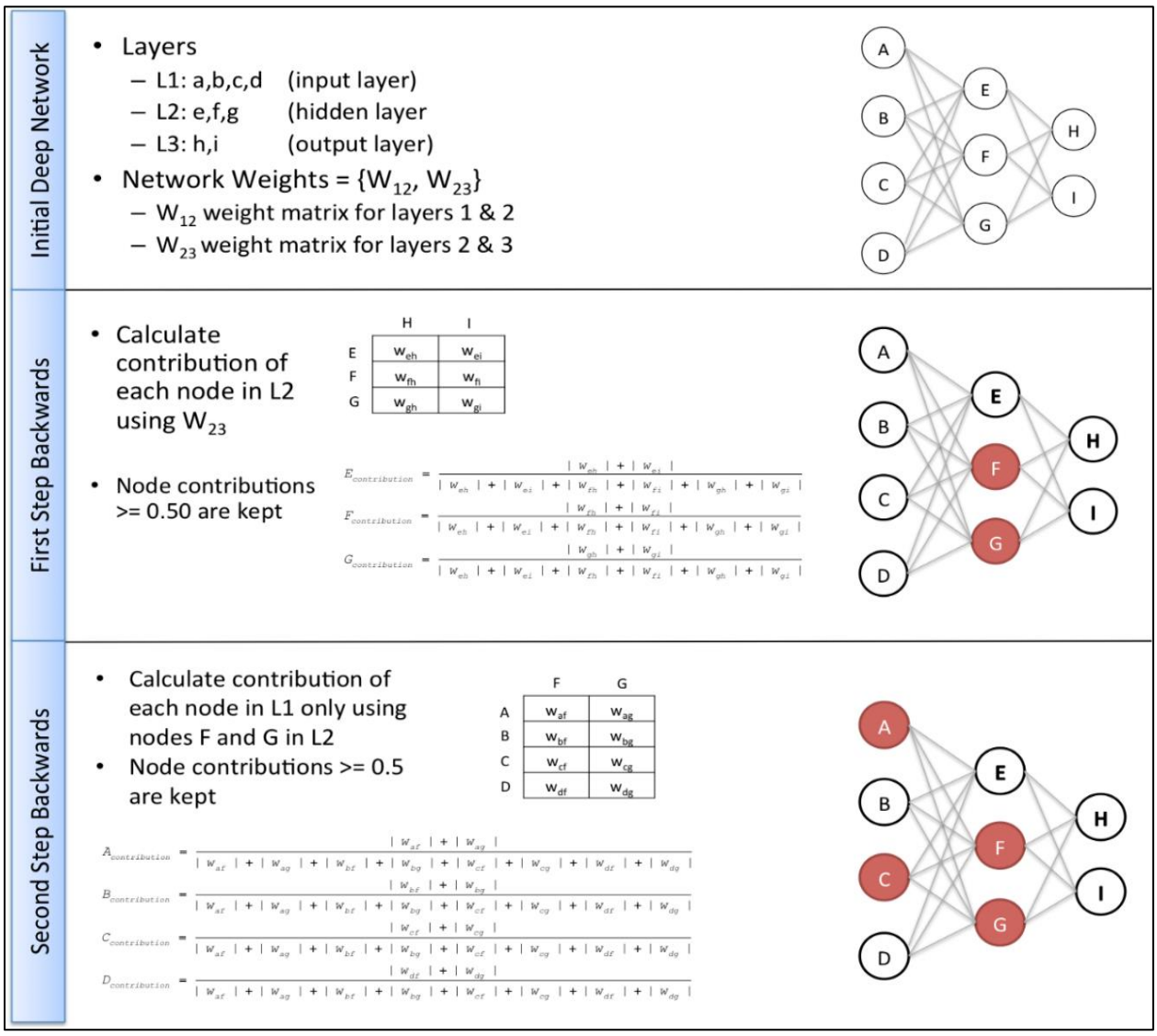
surface area, cortical thickness, ICV, and sex features that are expressed in HR-ASD subjects and that cannot be re-created using random labels, as not a single random permutation reached comparable classification performance. Furthermore, given the average (AVE) and standard error (STD ERR) values reported in Table 1 above for each performance measure (PPV, NPV, SENS, SPEC, ACC), and the u_o values shown in Figure 9, the resulting performance measure p-values for our method DL+SVM when test statistic $t_o = AVE - u_o/STD\ ERR$ is used would comfortably reject the null hypothesis H_o at a 0.0001 threshold. This strongly suggests the DL+SVM classification results are not due to chance.

2.6 Identifying high dimension features that contribute to prediction performance

Identifying which longitudinal CSA, CTH, ICV, and sex features defined in the high dimension feature vector have the greatest contribution (and the least contribution) to the LDC output by the fine-tuned DL network is a very challenging problem for deep non-linear dimensionality reduction approaches. To better understand how the input features are related to those defined in the LDC the following two approaches are used:

- **Approach 1 for our DL dimensionality reduction**: Using the weight matrices in the fully trained DL network $\bar{N} = \{\bar{W}_1, \bar{W}_2, \dots, \bar{W}_k\}$ we work backwards through the median DL network identifying only those nodes in the previous layer (e.g., $l-1$) that represent greater than 50% of the weight contribution layer l. Figure 10 illustrates this process, using a simple example 3-layer network.
- **Approach 2 for the linear sparse learning approach:** As shown in Table 1 above**,** since the sparse learning approach outperformed the PCA approach, we investigated also the features that were selected in that sparse learning. For this purpose we used an elastic net regularization, i.e. $\min_x \frac{1}{2}\|Ax - y\| + \frac{\rho}{2}\|x\|_2^2 + \lambda\|x\|_1$, the diagnosis labels $\boldsymbol{y}$, and the trained two-stage sparse learning pipeline as follows:
    - Sequentially adjust the λ parameter defined in the $l_1$regularization (sparsity) term between the values of 0.1 and 1.0 at increments of 0.1. For each increment, the learned sparse representation was applied to each high dimension feature vector in the test data set, and then the diagnosis labels for each sparse high dimension feature vector is predicted using trained two-stage pipeline and the PPV, NPV, and accuracy scores were stored.
    - The λ parameter that produced the highest PPV, NPV, and accuracy scores was then retrieved along with the learned sparse representation. In general, the selected features in the retrieved sparse representation provide insight about which high dimension longitudinal features contribute most to prediction performance.

Figure 10. Example procedure illustrating the identification of nodes that contribute most to the prediction performance.



The results in Table 2 below show the top 40 features found by Approach 1 (non-linear deep learning), and the results in Table 3 below show the top 40 features found by Approach 2 (linear sparse learning).

Table 2. Top 40 features contributing to the DL dimensionality reduction

	Measure	Month	Region	Description
1	CSA	6	24	Right Superior frontal gyrus: medial
2	CSA	6	58	Right Postcentral gyrus
3	CSA	6	23	Left Superior frontal gyrus: medial
4	CSA	6	57	Left Postcentral gyrus
5	ICV	6	N/A	N/A
6	CSA	6	61	Left Inferior parietal: but supramarginal and angular gyri
7	CSA	12	26	Right Superior frontal gyrus: medial orbital
8	CSA	12	55	Left Fusiform gyrus
9	CSA	12	10	Right Middle frontal gyrus orbital part
10	CSA	6	62	Right Inferior parietal: but supramarginal and angular gyri
11	CSA	12	47	Left lingual gyrus
12	CSA	12	21	Left Olfactory Cortex
13	CTH	12	67	Left Precuneus
14	CSA	12	89	Left Inferior temporal gyrus
15	ICV	12	N/A	N/A
16	CTH	6	36	Right Posterior cingulate gyrus
17	CSA	12	14	Right Inferior frontal gyrus: triangular part
18	CTH	6	23	Left Superior frontal gyrus: medial
19	CTH	12	90	Right Inferior temporal gyrus
20	CSA	12	33	Left Median cingulate and paracingulate gyri
21	CSA	12	61	Left Inferior parietal: but supramarginal and angular gyri
22	CSA	12	6	Right Superior frontal gyrus: orbital part
23	GA	12	18	Right Rolandic operculum
24	GA	12	22	Right Olfactory Cortex
25	GA	12	68	Right Precuneus
26	CTH	6	11	Left Inferior frontal gyrus: opercular part
27	GA	12	84	Right Temporal pole: superior temporal gyrus
28	CSA	12	90	Right Inferior temporal gyrus
29	CTH	6	79	Left Heschl gyrus
30	CSA	12	27	Left Gyrus Rectus
31	CTH	6	80	Right Heschl gyrus
32	CSA	6	47	Left Lingual gyrus
33	CSA	6	34	Right Median cingulate and paracingulate gyri
34	CSA	12	25	Left Superior frontal gyrus: medial orbital
35	CSA	6	55	Left Fusiform gyrus
36	CTH	12	39	Left Parahippocampal gyrus
37	CTH	12	65	Left Angular gyrus
38	CTH	6	35	Left Posterior cingulate gyrus
39	CSA	12	34	Right Median cingulate and paracingulate gyri
40	CTH	12	81	Left Superior temporal gyrus

Table 3. Top 40 features contributing to the linear sparse learning classification

	Measure	Month	Region	Description
1	CSA	6	24	Right Superior frontal gyrus: medial
2	CSA	6	23	Left Superior frontal gyrus: medial
3	CSA	6	58	Right Postcentral .gyfus
4	CSA	6	57	Left Posteentral gyrus
5	CSA	6	61	Left Inferior parietal: but supramarginal and angular gyri
6	CSA	6	62	Right Inferior parietal; but supramarginal and angular gyri
7	CTH	6	27	Left Gyrus Rectus
8	CSA	6	34	Right Median cingulate and paracingulate gyri
9	ICV	6	N/A	N/A
10	CSA	6	55	Left Fusiform gyrus
11	CSA	6	59	Left Superior parietal gyrus
12	CSA	6	47	Left Lingual gyrus
13	CTH	6	83	Left Temporal pole: superior temporal gyrus
14	CSA	6	40	Right Parahippocampal gyrus
15	CSA	12	56	Right Fusiform gyrus
16	CTH	6	43	Left Calcarine fissure and surrounding cortex
17	CTH	6	34	Right Median cingulate and paracingulate gyri
18	CSA	12	34	Right Median cingulate and paracingulate gyri
19	CSA	6	49	Left Superior occipital gyrus
20	CTH	6	82	Right Superior temporal gyrus
21	CSA	6	4	Right Superior frontal gyrus; dorsolateral
22	CTH	6	51	Left Middle occipital gyrus
23	CTH	6	84	Right temporal pole: superior temporal gyrus
24	CSA	6	89	Left Inferior temporal gyrus
25	CSA	6	48	Right Lingual gyrus
26	ICV	12	N/A	N/A
27	CSA	6	68	Right Precuneus
28	CSA	12	68	Right Precuneus
29	CSA	6	28	Right Gyrus Rectus
30	CSA	6	39	Left Parahippocampal gyrus
31	CTH	6	56	Right fusiform gyrus
32	CTH	6	86.	Right Middle temporal gyrus
33	CTH	6	39	Left Parahippocampal gyrus
34	CTH	6	90	Right Inferior temporal gyrus
35	CTH	6	6	Right Superior frontal gyrus: orbital part
36	CSA	6	1	Left Precentral gyrus
37	CTH	6	80	Right Heschl gyrus
38	CTH	6	22	Right Olfactory cortex
39	CSA	12	18	Right Rolandic operculum
40	CSA	6	83	Left Temporal pole: superior temporal gyrus

Comparing these results from the linear sparse learning classification (Extended Data Figure 2) to the cortical regions showing significant expansion in surface from 6 to 12 months in HR-ASD (see Main Text Figure 3), the left inferior temporal gyrus is observed in the contribution analysis as well as in the significant expansion map. In contrast, the cuneus and middle occipital gyrus regions observed in the significant surface area expansion map are not part of the top 40 contributing features in the DL dimensionality reduction. As mentioned above, identifying the most relevant features in as DL dimensionality reduction is difficult. Any current method to deduce such most contributing features will only partially capture the nature of the DL dimensionality reduction. Thus the absence of the cuneus and middle occipital gyrus features from this particular contribution analysis does not consequently mean that those features are not relevant for the DL dimensionality reduction.

References

1. Kim, S. H., et al. Adaptive prior probability and spatial temporal intensity change estimation for segmentation of the one-year-old human brain. *J Neurosci Methods*. **212**(1), 43–55 (2013).
2. Tustison, N. J., et al. N4ITK: improved N3 bias correction. *IEEE Transactions on Medical Imaging*. **29**(6), 1310–1320 (2010).
3. Fonov, V., et al. Improved precision in the measurement of longitudinal global and regional volumetric changes via a novel MRI gradient distortion characterization and correction technique. *Computer Vision* - Accv 2006, Pt I, 6326, 324–333. (2010).
4. Fonov, V., et al. Unbiased average age-appropriate atlases for pediatric studies. *NeuroImage*. **54**(1), 313–327 (2011).
5. Shaw, P. et al. Development of cortical surface area and gyrification in attention-deficit/hyperactivity disorder. *Biol Psychiatry*. **72**(3), 191-197 (2012).
6. Shaw, P. et al. Neurodevelopmental trajectories of the human cerebral cortex. *J Neurosci*. **28**(14), 3586-94 (2008).
7. Tzourio-Mazoyer, N. et al. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *NeuroImage*. **15**(1), 273-289 (2002).
8. Avants, B. B., et al. A reproducible evaluation of ANTs similarity metric performance in brain image registration. *NeuroImage*. **54**(3), 2033–2044. (2011).
9. Hazlett, H.C, et al. Early brain overgrowth in autism associated with an increase in cortical surface area before age 2 years. *Arch Gen Psychiatry*. **68**(5): 467-76 (2011).
10. Fishbaugh, J., Durrleman, S., Piven, J., Gerig, G. A framework for longitudinal data analysis via shape regression. Proc. SPIE 8314, Medical Imaging 2012: Image Processing.

11. Giedd, J.N., et al. Brain development during childhood and adolescence: a longitudinal MRI study. *Nature Neuroscience*. **2**: 861-863 (1999).
12. Hinton, G.E., Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science.* **313** (5786): 504-507 (2006).
13. Hoo-Chang Shin, M. R. Orton, D. J. Collins, S. J. Doran, M. O. Leach, "Stacked Autoencoders for Unsupervised Feature Learning and Multiple Organ Detection in a Pilot Study Using 4D Patient Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8):1930-1943 (2013).
14. Dan C. Ciresan, Alessandro Giusti, Luca M. Gambardella, Jurgen Schmidhuber, "Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks", MICCAI 2013, Nagoya, Japan.
15. Zou, H., & Hastie, T. Regularization and variable selection via the elastic net. *J of the Royal Statistical Society. Series B, Statistical Methodology*. **67**(2), 301–320. (2005).
16. Li, Train, Thung, Ji, Shen, Li, "Robust deep learning for improved classification of AD/MCI patients", Machine Learning in Medical Imaging (MLMI) Workshop, Lecture Notes in Computer Science, Vol 8679, pp 240-247, 2014.
17. Suk, Shen, "Deep learning-based feature representation for AD/MCI classification" , Medical Image Computing and Computer Assisted Intervention (MICCAI), Lecture Notes in Computer Science, Vol 8150, pp 583-590, 2013.
18. Lee, Lagman, Pham, Ng, "Supervised feature learning for audio classification using convolutional deep belief networks", Advances in neural information processing systems (NIPS), 2009.
19. Lee, Gross, Ranganath, Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations", 26th Annual International conference on Machine Learning (ICML), pp 609-616, 2009.
20. Hao, Raiko, llin, Karhunen, "Gated Boltzmann Machine in Texture Modeling", International Conference on Artificial Neural Networks and Machine Learning, (ICANN), Lecture Notes in Computer Science, Vol 7553, pp 124-131, 2012.
21. Huang, Lee, Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2518-2525, 2012.
22. Gan, Li, Zhai, Liu, "Deep self-taught learning for facial beauty prediction", Neurocomputing, Vol. 144, pp. 295-303, 2014.
23. Hinton, G., "*A Practical Guide to Training Restricted Boltzmann Machines"*, UTML TR 2010–003.