SCIENTIFIC REPORTS

OPEN

Cell Identity Codes: Understanding Cell Identity from Gene Expression Profiles using Deep Neural Networks

Farzad Abdolhosseini[1], Behrooz Azarkhalili[2], Abbas Maazallahi[1], Aryan Kamal[1], Seyed Abolfazl Motahari[1], Ali Sharifi-Zarchi [ID][1] & Hamidreza Chitsaz[3]

Understanding cell identity is an important task in many biomedical areas. Expression patterns of specific marker genes have been used to characterize some limited cell types, but exclusive markers are not available for many cell types. A second approach is to use machine learning to discriminate cell types based on the whole gene expression profiles (GEPs). The accuracies of simple classification algorithms such as linear discriminators or support vector machines are limited due to the complexity of biological systems. We used deep neural networks to analyze 1040 GEPs from 16 different human tissues and cell types. After comparing different architectures, we identified a specific structure of deep autoencoders that can encode a GEP into a vector of 30 numeric values, which we call the *cell identity code* (CIC). The original GEP can be reproduced from the CIC with an accuracy comparable to technical replicates of the same experiment. Although we use an unsupervised approach to train the autoencoder, we show different values of the CIC are connected to different biological aspects of the cell, such as different pathways or biological processes. This network can use CIC to reproduce the GEP of the cell types it has never seen during the training. It also can resist some noise in the measurement of the GEP. Furthermore, we introduce *classifier autoencoder*, an architecture that can accurately identify cell type based on the GEP or the CIC.

Accurate identification of cell types has received significant attention due to several applications in research and clinics. For instance, one major goal of regenerative medicine is to differentiate pluripotent stem cells, such as embryonic stem cells (ESCs) or induced pluripotent stem cells (iPSCs), to some specific tissue required for a patient, such as neurons or cardiomyocytes[1]. After treating the stem cells with a differentiation protocol, an important question is whether the produced cells have obtained true identity of the desired cell type[2].

There are some specific marker genes identified whose expression patterns can be used to characterize different cell types. Some examples are the expression of Insulin, which confirms the β-cell identity; pluripotency markers such as Nanog which are used to identify pluripotent stem cells; and membrane markers that label several types of cancer cells. The application of marker genes, however, is limited to specific cell types. For many cell types, no markers are identified so far that can accurately characterize them. Furthermore, the expression of many marker genes is not exclusive to one particular cell type. For instance, Oct4 that is used in many studies as a pluripotency marker[3] is also expressed in non-pluripotent cell types such as adult stem cells[4]. One important possible solution is to recognize the cell identity using the whole-genome gene expression profile (GEP), rather than a small subset of marker genes[5,6].

Despite its importance, there have been limited efforts to systematically recognize cell identity through the GEP. Classical differential expression analysis and clustering methods are used to discriminate a limited number of cells. An extension of the nearest centroid classification method was suggested for classification of cancer types[7]. This method and empirical Bayes analysis[8] were used to classify 88 normal and prostate cancer samples into different sub-types[9]. Linear algebraic and statistical methods such as non-negative matrix factorization,

[1]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. [2]Royan Institute for Stem Cell Biology and Technology, ACECR, Tehran, Iran. [3]Department of Computer Science, Colorado State University, Fort Collins, CO, USA. Farzad Abdolhosseini and Behrooz Azarkhalili contributed equally. Correspondence and requests for materials should be addressed to A.S.-Z. (email: asharifi@sharif.ir) or H.C. (email: chitsaz@chitsazlab.org)

Kullback-Leibler divergence and non-negative least squares were employed in an unsupervised method for cell type identification. We applied it to 77 samples for discriminating three cell types[10]. Although shown to be successful for a few cell types, whether these methods can be extended to analyze tens of different cell types needs further investigation.

Although there are many models developed so far to classify a few different cell types, there are few generic models to classify an arbitrarily large number of cell types with good accuracy, one of which is CellNet[11]. It is an online tool that can analyze Microarray gene expression profiles of mouse or human and assign them to each of 16 human or 20 mouse tissues or cell types. CellNet requires a minimum 60 samples of each tissue to derive the global and tissue-specific gene regulatory networks. The subnetworks of the union of these networks are then identified using a community detection algorithm called Infomap[12], from which the tissue-specific subnetworks are identified using Gene Set Enrichment Analysis (GSEA)[13]. Expression levels of the genes belonging to the tissue-specific subnetworks are then used to train Random Forest binary classifiers for each tissue. As a limitation, it is a cumbersome task to hand-pick an equal number of samples from each cell types that have the same Microarray platform and undergo perturbations, in order to derive the gene regulatory networks. Furthermore, deriving a highly accurate gene regulatory network by only using GEPs is challenging.

A major obstacle to classical machine learning methods such as Support Vector Machine (SVM) or Random Forest is the presence of hundreds, or even thousands of genes, which are differentially expressed among different cell types, without causal connection to the cell identity. Their expression change can be even more significant than the master regulators of the cell identity due to a number of reasons, such as differences among batches, culture conditions, strains, or treatment protocols of different cell types. As a result, a simple classifier might highlight a group of genes as discriminators of the cell identity for one training dataset, which fail to correctly predict the identities of another dataset.

Recently, deep learning has made breakthroughs in different scientific areas such as games[14,15], speech[16], face[17], image and text[18] recognition, robotics[19], and web search[20]. It has also been used in several bioinformatics applications including predicting protein binding sites in DNA and RNA[21], DNA replication initiation and termination zones[22], protein secondary structure[23] and folding[24], residue-residue and protein-protein interaction[25], non-coding DNA function prediction[26] and inferring expression of target from landmark genes[27].

Deep autoencoders are a group of deep neural networks that were originally developed to learn dimensionality reduction. They are usually given some data as the input, and they are expected to generate the same data as the output. The data needs to pass an internal layer that has significantly fewer neurons than the dimensionality of the data, hence it learns to encode and decode the data with the minimum loss. They have received considerable attention due to several reasons. They are usually unsupervised and can be trained using unlabelled data. They can learn internal patterns of the data, such as correlations among different variables, in order to reduce its dimensionality. Furthermore, there are several extensions of deep autoencoders that can perform additional tasks such as denoising the data[28].

Due to the internal patterns of the GEPs such as coexpression of large clusters of genes we hypothesized the dimensionality of GEPs can be significantly reduced using deep autoencoders. By doing so, we can transform the GEP into a smaller number of features that can reproduce the whole GEP. We also hypothesized that these features might reflect different aspects of the cell biology. Furthermore, we speculated the cellular identity can be revealed by these features. Here we employ and extend deep autoencoders to address these hypotheses.

Results and Discussion

We obtained normalized expression profiles of 20184 genes in 1040 samples of 16 different human tissues and cell types from different datasets of NCBI GEO, which were collected and preprocessed in CellNet[11]. There was an equal number of 65 samples per cell type or tissue with at least 10 different biological perturbations to ensure the diversity of the samples from each tissue. Presence of different datasets and perturbations was important to prevent overfitting.

Figure 1a shows a general architecture of an autoencoder, consisting of an encoder part that converts a given gene expression profile (GEP) to a code layer shown in red, and then decodes it back to reproduce the original GEP. Hereafter we call the output of the encoder part the Cell Identity Code (CIC), and the output of the decoder part as the Reproduced Expression Profile (REP).

Our first task was to determine a particular architecture of autoencoders that can accurately reproduce GEPs. For this purpose, we tested 10 different architectures with distinct layer configurations and activation functions. After selecting the outperforming architecture, we tested different sizes of the CIC layer and selected a particular architecture with 30 neurons in the CIC layer, as shown in Fig. 1b. See methods for more details. The results of the comparison between different neural network architectures and between different sizes of CICs are presented in subsections 3.3 and 3.4.

Accuracy of Cell Identity Codes. We asked whether a 20184-dimensional GEP vector can be accurately compressed in a 30-dimensional CIC, without losing the data. To address this question, we trained the selected network architecture (Fig. 1b) by 75% of randomly selected samples ($n = 780$) as the training dataset. Then we measured its performance on the remaining 25% of the GEPs ($n = 260$), as the test dataset. Each test sample was encoded to a CIC and then decoded to a reproduced gene expression profile (REP). The distance between each REP and the original GEP was measured by Mean Square Error (MSE). We observed a stable decreasing MSE trend during the training.

To ensure the robustness of the results and lack of overfitting, we also performed a 10-fold cross-validation with balanced sampling. As shown in Fig. 2a, the average MSE of the test dataset is significantly decreased during the training from 0.3 (epoch 1) to 0.11 (epoch 66). The average MSE value plateaus at 0.11 for the rest of the training. The error bars depict the standard errors of the measured MSEs among cross-validation runs. The
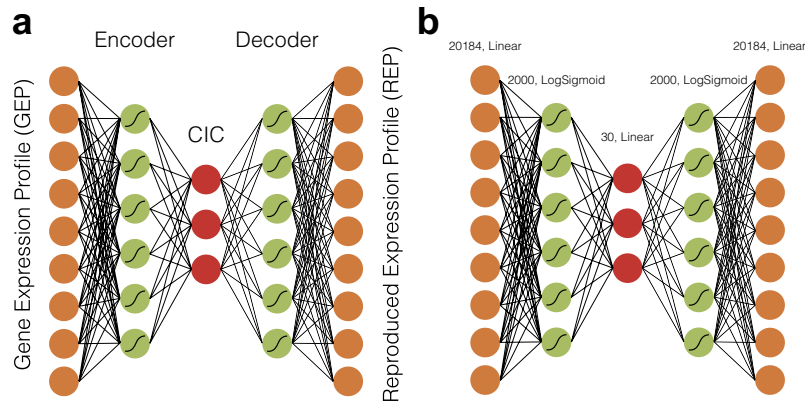
Figure 1. The architecture of autoencoder. (**a**) We call the output of the red layer in the middle Cell Identity Code (CIC). Each layer is fully connected to the next layer, and the edges are not shown. (**b**) The network that outperformed the other networks of our study. The number of neurons in each layer and the activation functions are shown above each layer.

maximum standard error during the training is 0.0035, which is much smaller than the average MSEs (maximum ratio <3%), which shows the measured average MSEs are robust, and there is no sign of overfitting to a particular portion of the data.

While the average MSEs showed a small average distance between each pair of GEP-REP, we questioned whether this holds for each individual pair. To answer that, for each GEP in the test data, we counted the number of other GEPs that were closer than the paired REP. There was an average number of about 16 samples per tissue in the test data, many of which were from biological or technical replicates. Hence if a REP was too different from its GEP, it was likely that many other test samples were closer to the GEP. Figure 2b shows the results, in which the Spearman correlation coefficient is used as the measure of similarity between a pair of GEP-REP, or between two GEPs. For 88.4% of the test cases (230 out of 260), the paired REP was ranked the first, which means the REP has been closer than any other test sample. For 22 cases (8.4%) the REP was ranked the second, and for 8 test cases (3%) it was ranked the third. We also used MSE as a measure of distance and counted the number of test-cases for which the REP was the least distant than any other test sample (Fig. 2c). For 226 test cases (87%) the REP was ranked the first, for 21 cases (8%) it was ranked the second, and for 13 cases (5%) it was ranked the third. This showed the results are similar, regardless of the similarity or distance metric. Most of the samples for which the REP was ranked the third were embryonic stem cells (ESCs).

We also performed the reverse experiment: for each test-case REP, we sorted all test-case GEPs according to their similarity (measured by either higher correlation or lower MSE) and identified the ranking of the original GEP that matched the REP. For 100% of the test-cases, the correct GEP was ranked the first. It means for each REP, its original GEP is more similar than any other test sample.

To further illustrate this, three examples are provided in Fig. 2d. In each column, the red scatter plot compares two GEPs denoted as samples 1 and 2, and the blue scatter plot compares the same sample 1 GEP with its REP. The colon and neuron samples (the left and middle columns) are carefully selected from the same study with an exactly matching region of the body, from two different human subjects. In both examples, the GEP of sample 1 is closer to its REP, than another sample from the same type and study. This is shown as higher correlation and lower MSE in blue scatter plots than the red ones. The ESC samples in the right column are replicates of the same cell line in the same study. There is a negligible distance between the MSE and correlation values of the blue versus the red scatter plots. It is important to note that ESCs are not primary tissues, rather they are *in-vitro* isogenic cell lines that are cultured in equal culture conditions.

To further scrutinize this, we identified for each of $n = 260$ test-cases the closest other GEP based on Spearman correlation coefficient (Fig. 2e) and MSE (Fig. 2f). Then we measured the similarity (correlation) or the distance (MSE) between the paired GEPs, and also between each GEP and its REP. As shown, the Spearman correlation values are significantly higher for REP-GEP than the GEP-GEP pairs (Wilcoxon p-value $< 10^{-15}$). Also, the MSE values are significantly lower between REP-GEP pairs rather than GEP-GEP (Wilcoxon p-value $< 10^{-15}$).

Taken together, these experiments show the expression profiles are reproduced from the CICs with an accuracy comparable to profiling a technical replicate of the same sample.

Next, we compared the performance of our method against other widely-used dimensionality reduction algorithms. For that purpose, we transformed GEPs to 30-dimensional spaces using ordinary PCA, Non-negative Matrix Factorization (NMF), Polynomial-kernel PCA, Cosine-kernel PCA and Radial Basis Function (RBF)-kernel PCA. For each method, we used the inverse transformation to reproduce 20184-dimensional REP vectors and measured the MSE between each pair of GEP-REP. As shown in Fig. 2g, our CIC vectors outperform the other dimensionality reduction algorithms.

Universality of Cell Identity Codes. The next question was whether the autoencoder that is trained to generate the CICs based on some specific training cell types can also generate accurate cell identity for the other cell types, i.e. it can generalize to unseen cell types without retraining.
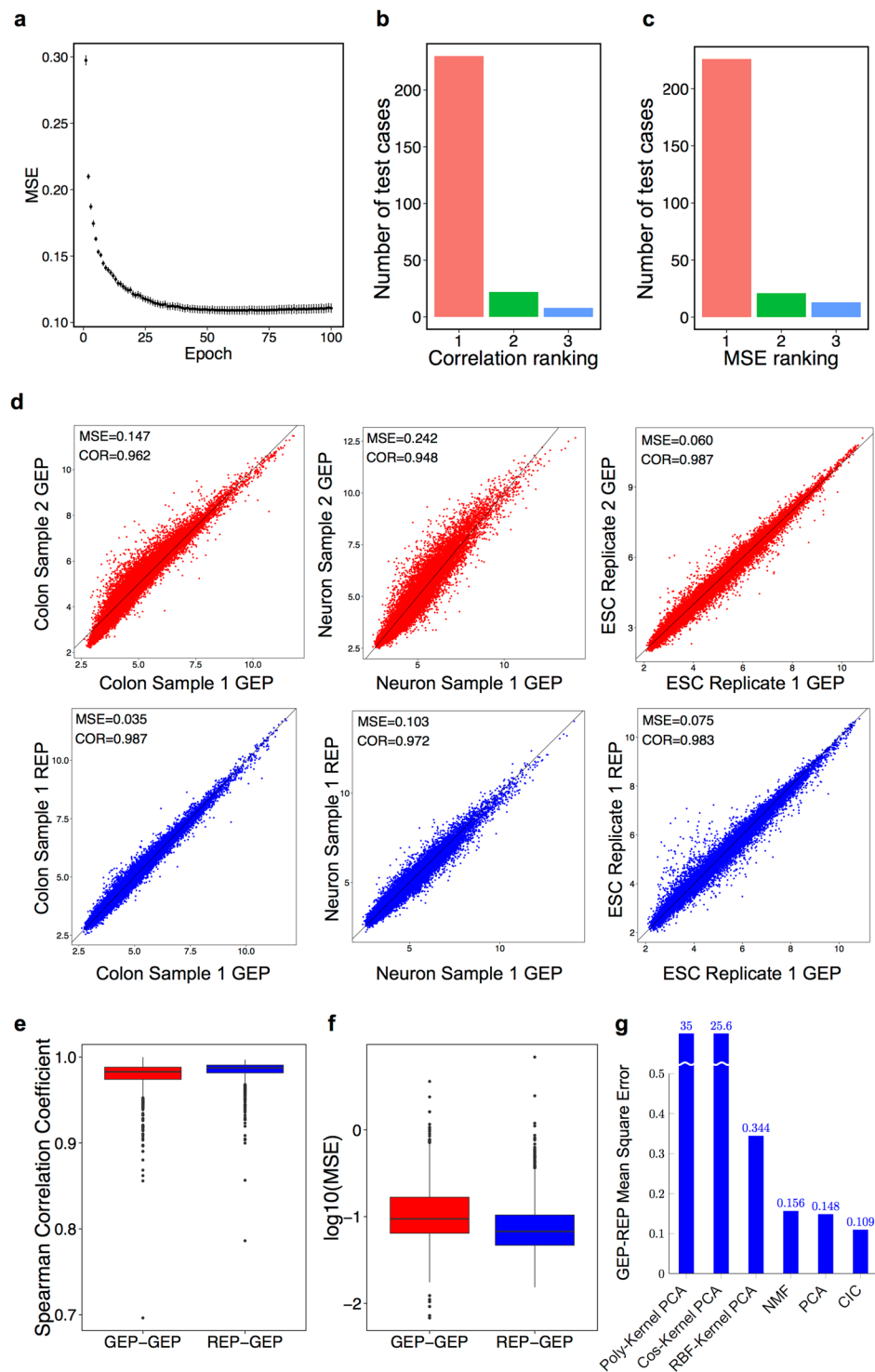
Figure 2. Comparison of the reproduced gene expression profiles (REPs) that are computationally made from cell identity codes (CICs) and the original gene expression profiles (GEPs). (**a**) The measured mean squared error (MSE) over the test samples after each epoch of training. Each dot represents the mean value for 10 measurements during 10-fold cross-validation, and error bars show standard errors. (**b**) The ranking of similarity of each REP to the original GEP, in comparison with all other test-cases. The similarity is measured by Spearman correlation coefficient. Rank 1 shows the number of test cases that their REPs have been the most similar to their GEPs than any other test-case GEPs. (**c**) A similar ranking, but the similarity is measured by the MSE value, the lower the MSE value the higher the similarity. (**d**) Three examples of test-cases from Colon (left column), Neuron (middle column), and ESC (right column). In each column, the first row (red scatter-plots) compare two GEPs of very similar biological samples obtained from the same experiment. The second row shows one of those GEPs, along with its REP. (**e**) Boxplots that show the quantiles and median of the distribution of Spearman correlation coefficient between GEPs of each test sample and the closest test sample

to it (GEP-GEP), and also between each test sample GEP and its REP. (**f**) The same comparison for MSE in $\log_{10}$ scale. (**g**) Comparison of the MSE distance between original GEP and reproduced expression profile by using different algorithms. MSE = Mean squared error, COR = Spearman correlation coefficient, GEP = Gene expression profile, REP = Reproduced gene expression profile, CIC = Cell identity code, ESC = Embryonic stem cells, PCA = Principal components analysis, NMF = Non-negative matrix factorization, RBF = Radial basis function, Cos = Cosine.

To answer this question we reserved all 65 B-cell samples for test and used 975 samples of all other 15 tissue types to train. After training an autoencoder on this dataset, we generated cell identity codes (CICs) for the test cases, and subsequently reproduced the expression profiles of B-cells using CICs. In all 65 samples, the reproduced gene expression profiles were closer to the original gene expression profiles of each B-cell sample than to any of the other 975 gene expression profiles of the training dataset. Both correlation and MSE results were significantly better for REPs than for the samples of other tissues.

This observation was surprising since there were other groups of blood cell types such as T-cells or macrophages available in the dataset. If the autoencoder had not generalized, it would have produced cell identity codes for B-cells that were similar to the cell identity codes of T-cells or macrophages. As a result, the reproduced gene expression profiles of those cell identity codes would be closer to the GEP of T-cells or macrophages available in the training dataset, rather than those of the original B-cell sample.

We also questioned whether the REPs have a B-cell biological identity. For that purpose, we performed a differential expression analysis between B-cell reproduced gene expression profiles and T-cell original gene expression profiles. We selected the genes that were significantly enriched ($\log_2$ fold-change >1 and Benjamini-Hochberg adjusted p-value ≤ 0.05) in B-cell REPs. This analysis identified 380 of such unique genes, that were used for a pathway analysis using Enrichr[29]. B-cell receptor signaling pathway of Homo sapiens was ranked the highest according to a Z-score criterion, with an adjusted p-value of 0.026.

This was a striking observation since the B-cell samples were never used in training and the autoencoder did not have any record of the B-cell gene expression patterns. When very well discriminated from the T-cells, we were confident we could obtain the same or even better results for comparison of the B-cell REPs with the GEPs of the other training cells.

To ensure this observation was not limited to the B-cells, we performed the same experiment for each of the primary tissue samples available in our dataset; see Fig. 3. For each cell type, we excluded all of its samples from the training and tested the autoencoder with those samples. Although we trained autoencoders in a few epochs (less than 3 epochs for all tissues excepting B-cell and heart), both higher Correlation medians and lower MSE medians in almost all tissues show the expression profiles can be produced from cell identity codes with high accuracy even if the target tissue is not used in training of the algorithm.

Cell Type Classification. We then asked if cell identity codes can be used for characterization of the samples. For this purpose, we used the CICs produced by training on 75% of all samples. Then we used two widely used classification algorithms Random Forest (RF) and Support Vector Machines (SVM) to determine the tissue or type of the cells, from the 16 available tissue/cell types, using cell identity codes. Both RF and SVM were trained on the same 75% training samples that were used for training the autoencoder.

To compare the accuracy of the results, we also used Principal Component Analysis (PCA) to reduce the dimension of the original gene expression profiles from 20184 to 30 dimensions, equal to the dimension of the cell identity codes. Conjunction of PCA with classification algorithms is widely used in many different scientific areas, particularly to prevent overfitting by reducing the number of parameters to be learned from the data after PCA-based dimensionality reduction[30]. Both classification algorithms were separately trained on the PCA transformation of the same training set. Then, all trained classifiers were applied to the test samples.

Out of 260 test cases, 25 and 27 samples were misclassified on the PCA transformation of the GEPs by Random Forest and SVM, respectively. Using cell identity codes, the number of misclassifications by Random Forest was slightly reduced to 16. SVM produced the same number of errors (Supplementary Fig. S7).

Remarkably, there could be an unlimited number of CICs generated from the same gene expression profile depending on the parameters and weights of the neural network, with comparable accuracies of reproducing gene expression profile. All of those cell identity codes, however, would not be necessarily suitable for classification of cell types. We expected some CICs to have particular patterns for each cell type, while other CICs of the same sample could not be easily classified.

In order to guide the training process of autoencoder to identify those neural network parameters that could produce more easily classifiable cell identity codes, we incorporated some additional layers to the network architecture; see Fig. 4. The new architecture, which we call classifier autoencoder, consists of two encoder and output sequential subnetworks, which were connected through the 30 neurons in the middle - i.e. the cell identity code neurons. The output consists of two branches, a decoder and a classifier. The classifier contained at most one hidden layer followed by a non-linear SoftMax layer. Therefore, the final code should also have the capacity to predict cell type by using a simple model (linear or neural network with only one hidden layer). Training was performed using a weighted average of two criteria functions, an MSE criterion for the decoder and a cross-entropy criterion for the classifier output.

This new architecture was used in the same way that we trained the simple autoencoder on the training set. Its performance on the test set outperformed both PCA and simple autoenoder. There were 10 misclassifications (3.8% error) by both SVM and Random Forest methods. The 60% error improvement in classifier autoencoder, in comparison with PCA, was an evidence for the capabilities of cell identity codes in determining the cell identity (Supplementary Fig. S7).
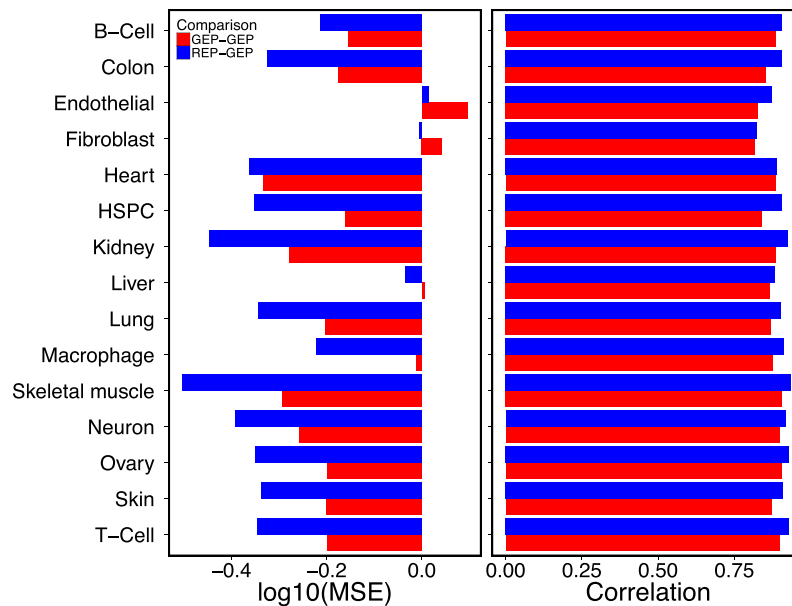
Figure 3. Cell identity codes can be generated for the cell types and tissues that are not used in training. The median of MSE and Spearman correlation coefficients are provided for test results. See text for more details.

CIC Represents Aspects of Cell Biology. An important question is whether biological pathways, processes, and other important aspects of cell biology can be learned from the expression data using cell identity codes. In other words, does each CIC component represent a different part of the cellular machinery? By CIC component, we mean each of the 30 numeric values in a cell identity code vector.

To answer this question, we sought to determine which subset of genes are most influenced by changing the i-th component of CIC ($1 \leq i \leq 30$). As an example, if the first CIC component represents the cell division process, then it should have the maximum effect on the reproduced expression values of the genes that are involved in cell division.

Let $C_i(x)$ be the i-th component of CIC for a given GEP x. We call $\overline{C}_i$ and σ_i as the mean and standard deviation of $C_i(x)$ for all available samples x, respectively. We used the trained classifier autoencoder to compute the values $\overline{C}_i$ and σ_i for all values $i (1 \leq i \geq 30)$. Then we produced a REP by feeding the $\overline{C} = (\overline{C}_1, \overline{C}_2, \cdots, \overline{C}_{30})$ as the input of the decoder network. We call the output of the decoder network for this particular input as the baseline REP. We also fed the decoder network 30 additional inputs with a value of $(\overline{C}_1, \overline{C}_2, \cdots, \overline{C}_{i-1}, \overline{C}_i + 2\sigma_i, \overline{C}_{i+1}, \cdots, \overline{C}_{30})$. It means that for round i, we only changed the i-th component of $\overline{C}$ by adding $2\sigma_i$ to $\overline{C}_i$. Then we compared the output of decoder network to see the reproduced expression values of which genes are most increased, in comparison with the baseline REP. For each CIC component, we determined 100 genes with the highest absolute change in REP after increasing value of the component. We took an equal number of genes for each CIC component to prevent bias in statistical analyses. There were a few cases, that the same gene was present in sets of two or more CIC components.

To analyze the pathway and Gene Ontology (GO) of these 30 gene sets, we used ToppCluster, an online tool for enrichment analysis of multiple gene sets[31]. The gene set enrichment p-values were adjusted using the Bonferroni method. For each gene set, all biological processes, cellular compartments, and pathways that were significantly enriched were analyzed; see Fig. 5 for cellular compartments, Fig. 6 for biological processes, and Fig. 7 for pathways. A value of 0.05 was used as the cutoff for the adjusted p-values of pathways. To have fewer nodes for better visualization, we used a slightly more stringent cutoff of 0.01 in cellular compartments and biological processes analyses.

For instance, the CIC component 11 (the lower-left side in Fig. 5) is linked to cellular compartments such as spindle, kinetochore, condensed chromosome, centromere, and microtubule. In Fig. 6 lower-right side, the same component is connected to biological processes such as mitotic cell cycle, mitotic nuclear division, metaphase/anaphase transition, chromosome segregation, nuclear division, and cell division. The same component is connected to cell cycle, M-phase, resolution of sister chromatids, separation of sister chromatids, G1/S transition, mitotic G1-G1/S phases, and cell-cycle pathways; see Fig. 7 upper-left side. It is evident that the CIC component 11 represents the genes, pathways, compartments, and processes involved in cell division.

While there is an obvious consistency in pathways, compartments, and processes that are linked to cell identity code components, each component is connected to different areas of cellular life. Among cellular compartments, we can see the extracellular space (CIC component 6), lateral plasma membrane (CIC component 25), cytosolic part (CIC component 7), ribosome (CIC component 21), and the nucleolus (CIC component 29). There are a few cases that two or more CIC components are linked to the same compartment (e.g. components 4, 12, 14, 15 and 27 to extracellular space).

Vast collections of biological processes and pathways are also represented by different CIC components. Likewise, most of them are linked to one CIC component. This shows important aspects of cellular biology are
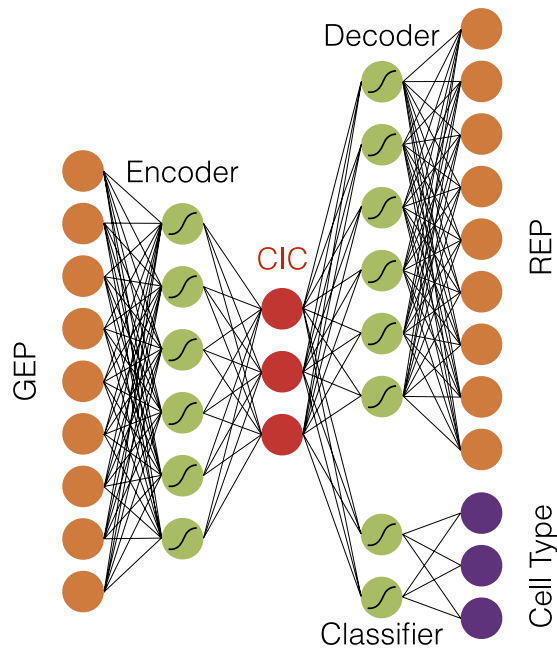
Figure 4. The schematic architecture of a classifier autoencoder. A gene expression profile (GEP) is encoded into a cell identity code (CIC, the red nodes). The CIC is connected to two parts: the decoder part which reproduces the expression profiles (REP) and also the classifier that identifies the cell type.

learned by the classifier autoencoders in an unsupervised approach (i.e. without providing any training data about GO or pathways). Our enrichment analysis with ToppCluster showed significant connections between CIC components and additional areas of cell biology such as molecular functions, protein domains, microRNAs, human phenotypes, disorders, and drugs. See Supplementary Figs S1–S6 and Table S1. The outcome of these analyses seem biologically relevant; for instance, CIC component 18 is associated with immunological synapse in Fig. 5, T-cell aggregation in Fig. 6, T-cell receptor signaling in Fig. 7, and a group of immunity disorders such as Lupus Erythematosus and AIDS; see Supplementary Fig. S5.

The stringent choice of Bonferroni p-value adjustment and 0.01 as the cutoff was to reduce the number of nodes for visualization. Using Benjamini-Hochberg false discovery rate p-value adjustment and a threshold of 0.05, which are usually used for GO and pathway analysis, significantly increases the number of pathways, processes and functions that are influenced by CIC components (2115 biological processes, 303 cellular compartments and 484 pathways). Additional networks and the list of 30 gene sets that are influenced the most by CIC components are provided as supplementary information.

CIC Can Resist Noise. We also questioned how resistant can our model be against noise. This noise can come from the measurement methods such as Microarray or RNA-seq, stochasticity of the gene expression process in the cells, environmental changes or other factors.

For that purpose, we first normalized the expression levels of each gene in all samples between 0 and 1, to control the noise level among all the genes. We performed 5-fold cross-validation. In each round, 80% of randomly selected samples were used as the training set and the remaining 20% as the test set. We ensured each sample was used in the test dataset exactly once. Before each round of cross-validation, all weights and bias parameters of the network were restarted to random values to ensure the training dataset of the previous round is unseen in the new round. The noise was generated from a Gaussian distribution $\mathcal{N} \sim (\mu = 0, \sigma = 0.1)$. During the training, the noise was added to the input layer, but the original GEPs without noise were expected to be reproduced in the output layer. We trained the network for 500 rounds, due to the altered structure of training.

The resulting classification accuracies for the test dataset were between 98.6% to 100% in five rounds of cross-validation, with an average accuracy of 99.4% and a standard error of 0.002. These results confirmed the CIC codes can be used to accurately classify the cell types even in presence of some noise.

Methods

Microarray Data. The preprocessed microarray expression profiles consisting of 20184 unique genes in 1040 biological samples from 16 human tissues or cell types (65 samples per tissue) were obtained from the CellNet package[11]. The cell types included Embryonic Stem Cell (ESC), ovary, skin, neuron, hematopoietic stem and progenitor cells (HSPC), macrophage, B-cell, T-cell, endothelial fibroblast, skeletal muscle, heart, kidney, lung, liver, and colon. All of the data were selected from the NCBI Gene Expression Omnibus (GEO)[32] and shared a common microarray platform (Affymetrix HG133 plus 2). There were at least 10 different conditions per cell type/tissue, that was essential for generalization of the learning process and to ensure that our network is not learning only a particular cell type of specific laboratory.

Figure 5. Cellular compartments that are enriched by each cell identity code (CIC) component. The red numbers show the 30 CIC components. For each CIC neuron, we identified a cluster of top 100 genes that their reproduced expression levels had maximum absolute influence after altering the output of the neuron. Then we identified cellular components that are significantly associated with the cluster of each neuron, which are the additional nodes in the plot. Each link in the plot shows a connection between a CIC component and a cellular compartment that is significantly associated with the genes altered by that CIC component.

The raw expression profiles were log$_2$ transformed and quantile normalized before our analysis. Differential expression analysis was performed using the linear regression and Bayesian analysis of the R/Bioconductor package limma[33].

Deep Neural Networks. There are many dense clusters in gene regulatory networks and the members of each cluster are usually co-expressed, which made us hypothesize that the information of gene expression profiles (GEPs) can be compressed in a significantly lower dimension, such that the whole GEP can be reproduced from the lower dimension data. To test this hypothesis, we decided to employ autoencoders for dimensionality reduction of the GEPs.

Autoencoders are a class of deep neural networks that have been applied to important applications such as denoising[34] or dimensionality reduction and have been shown to outperform other dimensionality reduction algorithms such as PCA for particular classes of data including images[35]. Furthermore, autoencoders can extract useful features from the data and significantly reduce the computational cost of the downstream analysis by replacing the whole data with a vector of smaller dimension.

A simple form of an autoencoder consists of one hidden layer that is fully connected to both input and output layers. The training data is fed to both input and output layers, and the aim is to reduce the dimensionality of the data in the hidden layer. This form of autoencoder has been used to analyze bacteria[36] and human gene expression data[37]. The deep architectures of autoencoders contain multiple layers between input and output. This type of autoencoder has been used to analyze yeast transcriptome profiles[38].

Figure 6. Biological processes that are enriched by each cell identity code (CIC) component. More details are provided in the caption of Fig. 5.

Comparison of Autoencoder Architectures.

The first methodological challenge was to design a particular architecture of the deep autoencoders that could accurately reconstruct the gene expression profiles after training. To address this challenge, we created 10 different architectures of autoencoders, each consisting of an encoder and a decoder part that was attached serially (Fig. 1a). They contained different sizes of neurons per layer, and various activation functions.
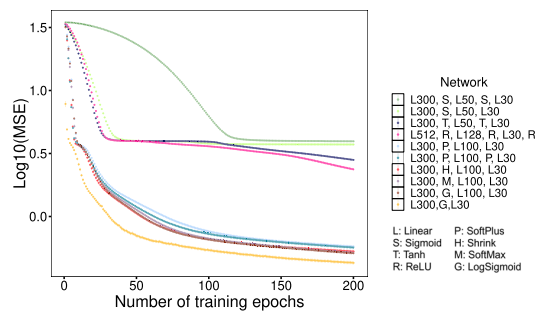
For this experiment, we selected 1000 genes with the highest variance of expression among all 1040 samples to speed up the training time. The encoder part of each neural network contained 1000 input neurons and 30 code neurons. Symmetrically, the decoder part started with 30 code neurons and ended with 1000 output neurons.

The encoder parts included 3 to 5 layers of linear and non-linear neurons (Fig. 8). A subset of 780 samples (75%) was randomly selected as the training set, and the remaining 260 samples (25%) were used as the test set. The training phase of each neural network included 200 epochs, with 100 iterations per epoch (total 20,000 iterations). In each iteration, all training samples were fed into the neural network after a random shuffle, and the neural network parameters were updated by a stochastic gradient descent (SGD) algorithm to minimize the mean squared error (MSE) loss function.

As shown in Fig. 8, some architectures had lower loss values than the others on the test samples. The second and third best architectures had a similar 1000 to 300 (1000:300) fully connected linear layer, followed by a SoftMax or LogSigmoid function applied to each of the 300 neurons, followed by 300:100 and 100:30 linear layers. The best network had the same architecture as the second best, except that the last two linear layers were merged into a 300:30 fully connected layer. Similar results were expectable from this pair of architectures since the product of the weight matrices for serial linear layers can result in the weight matrix of a single linear layer.

Figure 7. Pathways that are enriched by each cell identity code (CIC) component. More details are provided in the caption of Fig. 5.

It is also not surprising to see LogSigmoid and SoftMax have close results, as these functions can be transformed to each other. While LogSigmoid is mainly used for introducing non-linearity and binary classification, the main application of SoftMax is multi-class classification. We could expect the LogSigmoid CICs to have multiple active outputs, while each SoftMax CICs was expected to have one active output.

To ensure the stability of the results, we performed 10-fold cross-validation. Error-bars show the standard errors of the MSE values. Standard errors were generally very small that showed measured MSEs are stable and the models are not overfitted to a particular portion of the training data.

Comparison of Cell Identity Code Dimensionality. The second methodological challenge was to determine the best size of code to reduce the dimensionality of the GEPs without loss of data. For this purpose, we created a set of 10 autoencoders similar to the optimal architecture in the previous experiment, but different sizes of the code layer. In this experiment, we used the complete expression profiles of 20184 genes. All autoencoders included a 20184:2000 fully connected linear layer, a LogSigmoid layer, and then the 2000 neurons were fully connected linearly to a code layer of size 10 to 100 neurons. Again, 75% of data was randomly selected for training, and the remaining 25% was used as the test data.

The performance of the networks on the test samples are shown in Fig. 9. Once training is completed, there is a small gap between the networks with 10 and 20 code-layer neurons and the other networks. In a trade-off between the size of the CIC and the accuracy of the results, we selected 30 as the size of cell identity code for our analyses.

Figure 8. Comparison of 10 different architectures of the deep neural networks in encoding the expression levels of 1000 genes into the cell identity code (CIC) of size 30 and decoding it back to the original values. The numbers following letter L determine the number of neurons after a linear layer, e.g. "L300, G, L30" is a linear layer that fully connects the input 1000 neurons to 300 neurons, followed by a layer that applies a LogSigmoid function to each of the 300 neurons, and followed by a second linear layer that fully connects them to 30 neurons of the CIC. The structure of decoder parts are not shown in this figure but are symmetric to the encoders. Each dot indicates the average MSE values (log10 scale) for 10-fold cross-validation, and the error bars show standard errors.
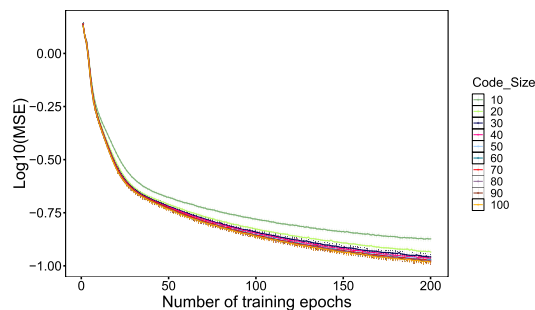
Figure 9. Similar structures of autoencoders, having different cell identity code (CIC) sizes are compared in learning the gene expression profiles (GEPs). Each dot indicates the average MSE values (log10 scale) for 10-fold cross-validation, and the error bars show standard errors.

One further observation was the lower MSE levels in this experiment, in comparison with the previous experiment on 1000 genes with the highest expression variances. In the presence of many genes with subtle changes in expression among different samples, a lower error rate was reasonable here.

Again, we performed 10-fold cross-validation without stratification. The standard errors, depicted as the error bars, are quite small that means the robustness of our results.

Architecture, training and testing the networks.

We trained an autoencoder with the optimal architecture that we determined previously to compress the gene expression profiles of 20184 genes in 780 training samples. In each iteration, a randomly selected training sample was fed into the neural network, and all of the weights were updated by an SGD algorithm. The learning rate was reduced during iterations of the same epoch, but restarted to the original value by each new epoch.

We used the sequential network architecture in all of our scenarios. Training was performed using the stochastic gradient descent algorithm with mean squared error (MSE) criterion. The learning rate of 0.01 did not work very well for many cases; hence, we used the learning rates between 0.001 and 0.003. There were a number of epochs in each training procedure. Each epoch consisted of several iterations, and training data were fed into the network in each iteration. The network parameters were saved after each epoch and used for the next epoch, but the decayed learning rate restarted after each epoch. For most cases, we used 200 epochs and 100 iterations per epoch.

Transfer and criterion functions.

To have non-linear layers, we used several typical transfer functions mentioned above. The formulas for the functions are as follows:

$$ReLU(x) = \max(0, x) \quad Sigmoid(x) = \frac{1}{1 + e^{-x}} \quad LogSigmoid(x) = \log\left(\frac{1}{1 + e^{-x}}\right)$$

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad SoftPlus(x) = \log(1 + e^x) \quad SoftShrink(x) = \begin{cases} x - \lambda, & \text{if } x > \lambda \\ x + \lambda, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases}$$

The soft-max function is applied to a list of values and returns a list of the same size with values in the range [0, 1] summing to one, therefore resembling a discrete probability distribution:

$$SoftMax_i(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

We also used cross-entropy error as the criterion for the classifier. Assuming we know the correct class c (e.g. cell-type), the cross-entropy loss can be calculated on the output of a soft-max function as follows:

$$CE\_Loss(x, \ c) = -\log\left(\frac{e^{x_c}}{\sum_j e^{x_j}}\right) = -x_c + \ \log\left(\sum_j e^{x_j}\right)$$

More details about different types of activation functions can be found on "Activation function" page of Wikipedia.

Cross-validation. During the work, we had to ensure the results were robust and the training was not overfitted towards a particular portion of the data. For this purpose, we performed additional experiments using 10-fold cross-validation. In our normal experiments, we randomly selected 75% of the samples as the training dataset and the remaining 25% as the test dataset. In 10-fold cross-validations, however, we randomly partitioned all of the samples among 10 groups of equal size. In each round of cross-validation, one group was taken as the test dataset, and the other 9 groups were used as training dataset. Each round of training was started from the scratch, i.e. the network parameters such as weights and biases were restarted to the random initial values. By this way, we ensured the test group is unseen after training the network with the other 9 groups. The test results of all of 10 rounds, such as MSE or correlation values, were merged together by calculating the mean value and standard error.

Gene Set Enrichment Analysis. We used ToppCluster multi gene-list enrichment analysis online application to determine GO terms, pathways, diseases, drugs, domains, and microRNAs associated with the 30 gene lists associated with the cell identity code components[31]. Nominal *p*-values were adjusted using Bonferroni or Benjamini-Hochberg methods, with 0.01 or 0.05 as *p*-values. While all of the different settings are considered as statistically significant, we increased stringency for some cases to keep the number of nodes suitable for visualization. A complete list of enriched terms can be determined using Supplementary Table S1.

Visualization. Both pre- and post-processing of the data and visualization of results were achieved by custom scripts in the R statistical language. We used several R/Bioconductor packages including ggplot2, parallel, data. table, and plyr. Networks were visualized using Gephi[39].

Implementation. We used the script language Lua with the package Torch7 to implement deep neural networks. To increase efficiency, we used Graphical Processing Unit (GPU) through the library CUDA for some of the training procedures. In each training procedure, the whole data was read from tabular text files and all samples were permuted using a fixed random seed. A random subset of 75% of all samples was used to train the networks, and the remaining 25% to test. The data was transformed into Torch Tensor for CPU, and Cuda Tensor for GPU training/testing.

We used several Torch7 packages, including nn, torch, cutorch, cunn and cudnn. Several neuron types were used in our analyses including the linear fully connected layers, rectified linear units (ReLUs), sigmoid, logarithmic sigmoid, hyperbolic tangent, soft-max, soft-plus, and shrink.

We also developed the same architectures in TensorFlow, and did not observe a change in the results due to platform change.

In order to ensure that there is no overfitting in our learning process we experimented with adding L1 and L2 regularization as well as dropout. However, as far as the results show, no meaningful change was observed between the models trained with and without regularization.

Hardware. We used a Linux server running Fedora 24 version 4.7.5–200. It contained 4 AMD Opteron(tm) 6386 SE processors, with 64 total cores running at 2.8 GHz and 512 GBytes of main memory. Training of neural networks was performed using the Cuda driver on an NVIDIA Tesla K20c GPU running at 706 MHz with 5120 MB memory.

Future Works

The present work can be extended in several ways. Batch normalization can be employed to limit the variation of values in different genes and datasets. Weight normalization can help to control the variations of weights in each layer. Moreover, more efficacious activation functions such as ReLu, PeakyReLu, Swish, etc. can be used to avoid some issues such as vanishing moments and saturation which can dramatically decrease the network's performance. Finally, hyperparameter tuning algorithms such as grid or random search can be used to evaluate different networks and chose the best one among all.

Data Availability

The full source codes and pre-processed data are available upon request.

References

1. Murry, C. E. & Keller, G. Differentiation of embryonic stem cells to clinically relevant populations: lessons from embryonic development. *Cell* **132**, 661–680 (2008).
2. Volarevic, V. *et al*. Human stem cell research and regenerative medicine-present and future. *British Medical Bulletin* **99**, 155–168 (2011).
3. Leitch, H. G. *et al*. Naive pluripotency is associated with global DNA hypomethylation. *Nature Structural & Molecular Biology* **20**, 311–316 (2013).
4. Liedtke, S., Stephan, M. & Kögler, G. Oct4 expression revisited: potential pitfalls for data misinterpretation in stem cell research. *Biological chemistry* **389**, 845–850 (2008).
5. Kuo, C. J. *et al*. A transcriptional hierarchy involved in mammalian cell-type specification. *Nature* **355**, 457–461 (1992).
6. Vidarsson, H., Hyllner, J. & Sartipy, P. Differentiation of Human Embryonic Stem Cells to Cardiomyocytes for *In Vitro* and *In Vivo* Applications. *Stem Cell Reviews and Reports* **6**, 108–120 (2010).
7. Tibshirani, R., Hastie, T., Narasimhan, B. & Chu, G. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences of the United States of America* **99**, 6567–6572 (2002).
8. Efron, B., Tibshirani, R., Storey, J. D. & Tusher, V. Empirical Bayes Analysis of a Microarray Experiment. *Journal of the American Statistical Association* **96**, 1151–1160 (2011).
9. Stuart, R. O. *et al*. In silico dissection of cell-type-associated patterns of gene expression in prostate cancer. *Proceedings of the National Academy of Sciences of the United States of America* **101**, 615–620 (2004).
10. Zuckerman, N. S., Noam, Y., Goldsmith, A. J. & Lee, P. P. A self-directed method for cell-type identification and separation of gene expression microarrays. *PLoS Computational Biology* **9**, e1003189 (2013).
11. Cahan, P. *et al*. CellNet: Network Biology Applied to Stem Cell Engineering. *Cell* **158**, 903–915 (2014).
12. Bohlin, L., Edler, D., Lancichinetti, A. & Rosvall, M. Community Detection and Visualization of Networks with the Map Equation Framework. In *Measuring Scholarly Impact*, 3-34 (Springer International Publishing, Cham, 2014).
13. Subramanian, A. *et al*. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America* **102**, 15545–15550 (2005).
14. Mnih, V. *et al*. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
15. Silver, D. *et al*. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
16. Deng, L., Hinton, G. & Kingsbury, B. New types of deep neural network learning for speech recognition and related applications: An overview. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013).
17. Sun, Y., Wang, X. & Tang, X. Deep learning face representation from predicting 10,000 classes. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014).
18. Srivastava, N. & Salakhutdinov, R. R. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research* **15**, 2949–2980 (2014).
19. Lenz, I., Lee, H. & Saxena, A. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* **34**, 705–724 (2015).
20. Huang, P. S. *et al*. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information and Knowledge Management* (2013).
21. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology* **33**, 831–838 (2015).
22. Liu, F. *et al*. De novo Identification of replication-timing domains in the human genome by deep learning. *Bioinformatics* **32**, btv643–649 (2015).
23. Heffernan, R. *et al*. Improving prediction of secondary structure, local backbone angles, and solvent accessible surface area of proteins by iterative deep learning. *Scientific Reports* **5**, 11476 (2015).
24. Jo, T., Hou, J., Eickholt, J. & Cheng, J. Improving Protein Fold Recognition by Deep LearningNetworks. *Scientific Reports* **5**, 17573 (2015).
25. Du, T., Liao, L., Wu, C. H. & Sun, B. Prediction of residue-residue contact matrix for protein-protein interaction with Fisher score features and deep learning. *Methods (San Diego, Calif.)* (2016).
26. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research* **44**, e107–e107 (2016).
27. Chen, Y., Li, Y., Narayan, R., Subramanian, A. & Xie, X. Gene expression inference with deep learning. *Bioinformatics* **32**, 1832–1839 (2016).
28. Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, 1096–1103, https://doi.org/10.1145/1390156.1390294 (ACM, New York, NY, USA, 2008).
29. Chen, E. Y. *et al*. Enrichr: interactive and collaborative HTML5 gene list enrichment analysis tool. *BMC Bioinformatics* **14**, 128 (2013).
30. Subasi, A. & Gursoy, M. I. Eeg signal classification using pca, ica, lda and support vector machines. *Expert systems with applications* **37**, 8659–8666 (2010).
31. Kaimal, V., Bardes, E. E., Tabar, S. C., Jegga, A. G. & Aronow, B. J. ToppCluster: a multiple gene list feature analyzer for comparative enrichment clustering and network-based dissection of biological systems. *Nucleic Acids Research* **38**, W96–102 (2010).
32. Barrett, T. *et al*. NCBI GEO: archive for functional genomics data sets-update. *Nucleic Acids Research* **41**, D991–5 (2013).
33. Ritchie, M. E. *et al*. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* **43**, e47–e47 (2015).
34. Vincent, P., Larochelle, H. & Bengio, Y. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008).
35. Hinton, G. E. Reducing the Dimensionality of Data with Neural Networks. *Science* **313**, 504–507 (2006).
36. Tan, J., Hammond, J. H., Hogan, D. A. & Greene, C. S. ADAGE-Based Integration of Publicly Available Pseudomonas aeruginosa Gene Expression Data with Denoising Autoencoders Illuminates Microbe-Host Interactions. *mSystems* **1**, e00025–15 (2016).
37. Tan, J., UNG, M., Cheng, C. & Greene, C. S. Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 132–143 (2015).
38. Chen, L., Cai, C., Chen, V. & Lu, X. Learning a hierarchical representation of the yeast transcriptomic machinery using an autoencoder model. *BMC Bioinformatics* **17**(Suppl 1), 9 (2016).
39. Bastian, M., Heymann, S. & Jacomy, M. Gephi: An open source software for exploring and manipulating networks, http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154 (2009).

Acknowledgements

Author Contributions

H.C. and A.S.Z. conceived the project. F.A. and B.A. developed the deep learning framework, generated the results, and created the figures. A.S.Z. and H.C. wrote the manuscript. F.A., B.A., H.C. and A.S.Z. revised the manuscript. H.C. provided the computation facility. S.A.M. provided some methodological ideas. A.M. and A.K. were involved in early data analysis.

Additional Information

Supplementary information accompanies this paper at https://doi.org/10.1038/s41598-019-38798-y.

Competing Interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.