Discriminative Concept Learning Network: Reveal High-level Differential Concepts from Shallow Architecture

Qiao Wang[1,2,3†], Sylvia Young[1,4,5], Aaron Harwood[1] and Cheng Soon Ong[6]
[1]Computing and Information Systems, [2]Centre for Epidemiology and Biostatistics, [3]Centre for Neural Engineering
The University of Melbourne, Parkville, VIC 3010, Australia.
[4]Centre for Diabetes Research, The Harry Perkins Institute of Medical Research, [5]Centre for Medical Research
The University of Western Australia, WA 6009, Australia.
[6]NICTA Canberra Research Lab, Tower A, 7 London Circuit, Canberra ACT 2601, Australia.
[†]Email: qwan@unimelb.edu.au

***Abstract*—A desired capability of deep learning is to understand the high-level, class-specific features via hierarchical features learning. However the training of deep architectures is costly comparing to simple shallow models. Bringing the high-level feature understanding into a simple shallow architecture remains an open question.**

We proposed a supervised learning algorithm[1], enabling binary classification along with an intrinsic ability of learning high-level discriminative concepts via a shallow neural network architecture. The physical architecture of the network has one hidden layer (also serving as the output layer) responsible for the classification and an input layer directly identifies the informative features that constitute the high-level differential concepts between the two classes.

Compared to other shallow classifiers, we demonstrate its practicability in real world classification problems. We also illustrate the human-understandable, discriminative concepts learned from the two image recognition exercises. Lastly, we show how it is useful in validating the disease-associated genetic variants in human genome as a real diagnostic genomics application.

I. Introduction

The artificial neural network has been a workhorse of machine learning since its inception [1], and it has had a resurgence in popularity in recent years due to the success of deep architectures [2], [3]. Deep neural networks have performed very well in computer vision [4]. A visual image can be described at different abstraction levels such as pixels, edges, object parts, objects and so on. However the shallow architectures are considered only able to capture low level features and simple in-variances such as "edge" or "blob" [5]–[7].

To learn high level concepts and complex in-variances, such as the concept of human face, deep architectures are proposed to build hierarchical feature representations in which simple features are detected by lower layers and feed into higher layers for complex feature abstraction [5], [8]. The costs of learning such high level concepts via deep architecture include: 1) the design of multiple different-sized stacked layers; 2) fine-tuning a large number of hyperparameters; 3) long processing time; and 4) hundreds of machines required [5]. Therefore, the successful training of deep architectures remains a challenge.

In fact it turns out that shallow architectures can produce classification performances that are equally competitive [9]. Due to the lower model complexity and being easy to train, popular shallow classification tools [10]–[13] have achieved great success in a broad range of real world classification problems over the past few decades. However, how to bring the high-level concept understanding, considered as an asset of deep architectures, into a simple shallow model is a great challenge.

We proposed a shallow neural network model called Discriminative Concept Learning Network (DCLN) in the hope of preserving the shallow model's simplicity and enabling the understanding of high-level differential concepts at the same time. The idea is simple but important, the instances are taken by the input layer, mapped to the hidden layer and normalised as positive unit vectors. At the hidden layer using converted feature representations, the two centroids of the two classes are calculated respectively, instances at the hidden layer are gradually forced to move away from their opposite class centroid and become close to their own class centroid. This behaviour indicates that the structural differences of the features between the two classes can be gradually understood and amplified at the hidden layer. Once the training is finished, an unknown instance can be classified into a class by simply converting to the hidden layer and comparing the distances (inner products) to the two class centroids. The model scores each feature at the input layer for its relative contribution to the classification after the model training. These scores are empirically demonstrated as effective measurements of the discriminative features which constitute the high-level differential concepts between the two classes. Interestingly such differential concepts, reflected by these scores, are human-understandable as expected in our image recognition exercises.

Inspired from this high-level concept abstraction ability, we demonstrate how DCLN is applicable to searching for the disease-associated genetic variants of Type 1 Diabetes in Genome-Wide Association Studies.

[1]This paper is published in *Proceedings of International Joint Conference on Neural Networks 2015*. Link to the paper: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7280525

II. METHOD

In this section, we describe the proposed DCLN model in a binary classification setup. Given a dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$ of N individual training instances, we denote a generic input vector for an instance $\mathbf{x} = (1, x_1, \ldots, x_\alpha)^T$ as an $(\alpha+1)$-dimensional input vector, where 1 is an appended biased unit fixed for all instances, and y is a binary label, i.e., $y \in \{\mathcal{A}, \mathcal{B}\}$, and the superscript T as vector transpose.

Suppose there is a neural network with $\alpha+1$ input features and β hidden neurons, as illustrated in Figure 1(a). The activation function of each hidden neuron j is given by

$$Z_j = g(\mathbf{w}_j^T \mathbf{x}) = g(w_{0j} + \sum_{i=1}^{\alpha} x_i w_{ij}), \qquad j = 1, \ldots, \beta, \quad (1)$$

where $\mathbf{w}_j = (w_{0j}, w_{1j}, \ldots, w_{\alpha j})^T$ are model parameters and w_{0j} is the weight for the biased unit 1. The activation function $g(\cdot)$ takes the form of *logistic sigmoid* function that

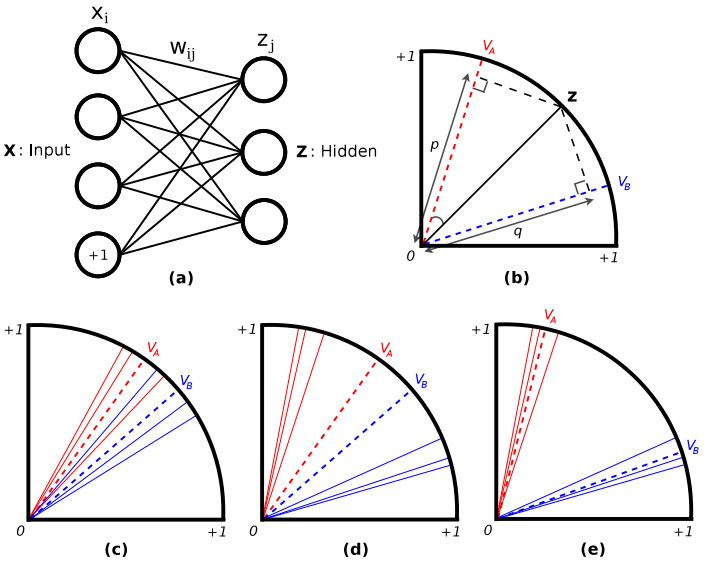
$$g(t) = \frac{1}{1 + e^{-\frac{1}{C}t}}, \quad (2)$$

where $0 < g(t) < 1$, constant $\frac{1}{C}$ controls the slope of the function [15]. By normalising each element Z_j we obtain z_j, such that

$$z_j = \frac{Z_j}{\sqrt{\sum_{k=1}^{\beta} Z_k^2}} = \frac{Z_j}{||\mathbf{Z}||_2^2}, \quad (3)$$

where $||\cdot||_2^2$ represents ℓ_2 norm of a vector, and we have $||\mathbf{z}||_2^2 = 1$ and $0 \leq z_j \leq 1$. The vector $\mathbf{z}$ composed of z_j is called a hidden neuron vector for instance $\mathbf{x}$. Because the ℓ_2 norm of vector $\mathbf{z}$ is equal to 1 and each z_j is greater than 0, the vector $\mathbf{z}$ can be illustrated as a link between the origin and a surface point of the *positive region* of a unit n-sphere in Figure 1(b). Note that an instance $\mathbf{x}$ in the dataset can now be represented by a hidden neuron vector $\mathbf{z}$: $\mathbf{x} \mapsto \mathbf{z}$.

Fig. 1. DCLN Structure and Principle



A key concept of the DCLN algorithm is the *centroid* of a class at the hidden layer, computable by using hidden neuron vectors $\mathbf{z}$'s. The general form of the two centroids of class $\mathcal{A}$ and $\mathcal{B}$ is defined as follows. $\forall \mathbf{x}$ in a class, the centroid $\mathbf{v}_y$ for class $y \in \{\mathcal{A}, \mathcal{B}\}$ is a β-dimensional vector containing elements v_{yj} $(j = 1, \cdots, \beta)$ which is given by

$$v_{yj} = \frac{\sum_{n \in y} z_{nj}}{\sqrt{\sum_{k=1}^{\beta} (\sum_{n \in y} z_{nk})^2}}, \qquad y \in \{\mathcal{A}, \mathcal{B}\},$$

where z_{nj} is the j^{th} element of the hidden neuron vector $\mathbf{z}$ for the n^{th} instance in the class y. The centroid can be illustrated as a vector representative of all the samples in each class at the hidden layer. Now that a centroid is in fact computed from hidden neuron vectors, it also holds the feature that $||\mathbf{v}||_2^2 = 1$ and $0 \leq \mathbf{v_j} \leq 1$. Here $\mathbf{v}$ is a generic representation of the centroids, $\mathbf{v}_\mathcal{A}$ or $\mathbf{v}_\mathcal{B}$.

Once the two centroids are obtained, each instance in the data can measure its distances to the two centroids by calculating the inner products respectively for the two classes:

$$p = \langle \mathbf{z}, \mathbf{v}_\mathcal{A} \rangle = ||\mathbf{z}|| \cdot ||\mathbf{v}_\mathcal{A}|| \cdot \cos\theta = 1 \cdot \cos\theta,$$

where θ is the angle between vector $\mathbf{z}$ and $\mathbf{v}_\mathcal{A}$. Geometrically speaking, in this scenario, p is the projection of $\mathbf{z}$ to $\mathbf{v}_\mathcal{A}$ shown in Figure 1(b). Similarly we have $q = \langle \mathbf{z}, \mathbf{v}_\mathcal{B} \rangle$ which is $\mathbf{z}$'s projection to $\mathbf{v}_\mathcal{B}$. Then the posterior probabilities of an instance belonging to class $\mathcal{A}$ and class $\mathcal{B}$ at the hidden layer are defined as:

$$P(y = \mathcal{A}|\mathbf{z}) := \frac{p}{p+q}, \qquad P(y = \mathcal{B}|\mathbf{z}) := \frac{q}{p+q} \quad (4)$$

which ensures that the two posterior probabilities sum to unity.

For the moment, consider a selected instance $\mathbf{x} \in \mathcal{A}$. Ostensibly its memberships to the two classes at the original input layer are: $P(y = \mathcal{A}|\mathbf{x} \in \mathcal{A}) = 1$ and $P(y = \mathcal{B}|\mathbf{x} \in \mathcal{A}) = 0$. However after mapping it to $\mathbf{z}$, its probabilities of belonging to class $\mathcal{A}$ and $\mathcal{B}$ at the hidden layer are redefined by equation 4. Due to the mapping $\mathbf{x} \mapsto \mathbf{z}$ being subject to a number of factors, such as the randomly initiated $\mathbf{w}^0$, its membership probability to class $\mathcal{A}$ is reduced below 1 at $\mathbf{z}$: $P(y = \mathcal{A}|\mathbf{z} \in \mathcal{A}) = \frac{p}{p+q} \leq 1$. To recover its membership probability back to 1 for class $\mathcal{A}$ at the hidden layer, the following must be satisfied:

$$\lim_{q/p \to 0} \frac{p}{p+q} = \lim_{q/p \to 0} \frac{1}{1 + \frac{q}{p}} = 1 = P(y = \mathcal{A}|\mathbf{x} \in \mathcal{A}). \quad (5)$$

Because the closeness is defined as the projection to the centroid, $q/p \to 0$ in equation 5 implies: 1) $0 < q \ll p < 1$, representing $\mathbf{z} \in \mathcal{A}$ is moving much closer to $\mathbf{v}_\mathcal{A}$ and furtherer away from $\mathbf{v}_\mathcal{B}$; or 2) $q = 0$ and $0 < p < 1$ means the $\mathbf{z} \in \mathcal{A}$ is orthogonal to $\mathbf{v}_\mathcal{B}$ with projection p to $\mathbf{v}_\mathcal{A}$. In both cases, the instance is moving away from $\mathbf{v}_\mathcal{B}$, and at least getting close or very close to the $\mathbf{v}_\mathcal{A}$ via the reconstruction of its class membership at the hidden layer. Symmetrically, we can derive the similar story for instance $\mathbf{z} \in \mathcal{B}$ that is moving away from $\mathbf{v}_\mathcal{A}$ and getting closer to $\mathbf{v}_\mathcal{B}$.

In summary, the *membership recovery* at the hidden layer is trying to force the instances moving away from their opposite class centroid and being around to their own class centroid (Figure 1(c)(d)). Such behaviour indicates that the learned data representation $\mathbf{z}$ is trying to capture and amplify the structural differences between the two classes. Once the two classes are well separated at the hidden layer, the binary classification

can be done by assigning an unknown instance to the nearest centroid.

Now the question is how to recover the instance membership by learning the optimal $\mathbf{w}^*$ that guides the mapping of $\mathbf{x} \mapsto \mathbf{z}$. Below we discuss two simple ways of learning $\mathbf{w}^*$ by either minimising *cross-entropy* (CE) or *squared error* (SE). We start with the cross-entropy method.

Firstly, we employ an indicator function for an instance $\mathbf{x}_n$ as below:

$$\gamma_n = \gamma(\mathbf{x}_n) = \begin{cases} 1, & \text{if } \mathbf{x}_n \in \text{class } \mathcal{A} \\ 0, & \text{otherwise.} \end{cases}$$

Assuming $\gamma_n|\mathbf{x}_n \sim \text{Bernoulli}(P(y_n = \mathcal{A}|\mathbf{z}))$, the sample likelihood is:

$$\mathcal{L}(\mathbf{w}|\mathcal{D}) = \prod_{n=1}^{N} P(y_n = A|\mathbf{z})^{\gamma_n} P(y_n = B|\mathbf{z})^{1-\gamma_n}, \quad (6)$$

where $P(y_n = B|\mathbf{z})^{1-\gamma_n} = (1 - P(y_n = A|\mathbf{z}))^{1-\gamma_n}$. For any instance in any case, one term must be equal to 1 in equation 6, either $P(y_n = A|\mathbf{z})^{\gamma_n}$ or $P(y_n = B|\mathbf{z})^{1-\gamma_n}$ because one indicator function must be 0, either γ_n or $1-\gamma_n$. In such a case, the likelihood function reaches its maximum value 1 when $\forall P(y_n = A|\mathbf{z})^{\gamma_n} \equiv 1$ for $\gamma_n = 1$, and $\forall P(y_n = B|\mathbf{z})^{1-\gamma_n} \equiv 1$ for $\gamma_n = 0$. This means, if we have a method to optimise the likelihood function to its maximum, the membership probability for the given instance will be recovered accordingly at the hidden layer.

To maximise the likelihood function, we convert it into the cross-entropy to minimise as the following:

$$\mathcal{E}(\mathbf{w}|\mathcal{D}) = -\sum_{n=1}^{N} \{\gamma_n \log P(y_n = \mathcal{A}|\mathbf{z}) + (1-\gamma_n)\log(1 - P(y_n = \mathcal{A}|\mathbf{z}))\}. \quad (7)$$

The commonly used Stochastic Gradient Descent (SGD) is employed here as the *online learning* approach of seeking the optimal model parameters $\mathbf{w}^*$. At a given *epoch*, update the $\mathbf{w}$, in the opposite direction of the gradient by a certain amount for every instance $\mathbf{x}$: $\Delta w_{ij} = -\eta \frac{\partial \mathcal{E}}{\partial w_{ij}}$, where η is the learning rate. The function reaches its local minimum when the derivatives are equal to 0. For each instance, Δw_{ij} is calculated by the chain rule of partial derivatives, as shown below:

$$\Delta w_{ij} = -\eta \frac{\partial \mathcal{E}(\mathbf{w}|\mathcal{D})}{\partial w_{ij}} = -\eta \frac{\partial \mathcal{E}(\mathbf{w}|\mathcal{D})}{\partial P(y = \mathcal{A}|\mathbf{z})} \cdot \frac{\partial P(y = \mathcal{A}|\mathbf{z})}{\partial z_j} \cdot \frac{\partial z_j}{\partial Z_j} \cdot \frac{\partial Z_j}{\partial w_{ij}}. \quad (8)$$

Hence we can now derive the four elementary components of equation 8 in a one-by-one manner. The first component is easy to be obtained. Simply by regarding equation 7 as a function of $P(y = \mathcal{A}|\mathbf{z})$, we have

$$-\eta \frac{\partial \mathcal{E}(\mathbf{w}|\mathcal{D})}{\partial P(y = \mathcal{A}|\mathbf{z})} = \eta\Big(\frac{\gamma_n}{P(y = \mathcal{A}|\mathbf{z})} - \frac{1-\gamma_n}{1 - P(y = \mathcal{A}|\mathbf{z})}\Big). \quad (9)$$

Recall from equation 4 that $P(y = \mathcal{A}|\mathbf{z})$ equals to $\frac{p}{p+q}$, where p represents the inner product of the instance and the centroid of class $\mathcal{A}$, and q the inner product of the instance and the centroid of class $\mathcal{B}$. In order to explicitly express the inner product as a function over z_j, we rewrite equation 4 into

$$P(y = \mathcal{A}|\mathbf{z}) = \frac{U(z_j)}{V(z_j)},$$

where

$$U(z_j) = p = \langle \mathbf{z}, \mathbf{v}_{\mathcal{A}} \rangle = \frac{\sum_{k=1}^{\beta}(z_k \sum_{n\in\mathcal{A}} z_{nk})}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{A}} z_{nk})^2}} \quad (10)$$

and

$$V(z_j) = p + q = \langle \mathbf{z}, \mathbf{v}_{\mathcal{A}} \rangle + \langle \mathbf{z}, \mathbf{v}_{\mathcal{B}} \rangle = \frac{\sum_{k=1}^{\beta}(z_k \sum_{n\in\mathcal{A}} z_{nk})}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{A}} z_{nk})^2}} + \frac{\sum_{k=1}^{\beta}(z_k \sum_{n\in\mathcal{B}} z_{nk})}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{B}} z_{nk})^2}}, \quad (11)$$

where z_j is derived by equation 3. With U and V, the $P(y = \mathcal{A}|\mathbf{z})$ is known, and hence equation 9 is solved.

The second term in equation 8 is calculated as the following:

$$\frac{\partial P(y = \mathcal{A}|\mathbf{z})}{\partial z_j} = \frac{\partial}{\partial z_j}\Big(\frac{U(z_j)}{V(z_j)}\Big) = \Big(\frac{U}{V}\Big)' = \frac{U'V - V'U}{V^2}, \quad (12)$$

where U' and V' are derivatives with respect to z_j for U and V respectively. Treating z_j as the only variable for both U and V, the derivatives have the following forms:

$$U' = \frac{dU(z_j)}{dz_j} = \frac{\sum_{n\in\mathcal{A}} z_{nj}}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{A}} z_{nk})^2}} = v_{\mathcal{A}j} \quad (13)$$

and

$$V' = \frac{dV(z_j)}{dz_j} = \frac{\sum_{n\in\mathcal{A}} z_{nj}}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{A}} z_{nk})^2}} + \frac{\sum_{n\in\mathcal{B}} z_{nj}}{\sqrt{\sum_{k=1}^{\beta}(\sum_{n\in\mathcal{B}} z_{nk})^2}} = v_{\mathcal{A}j} + v_{\mathcal{B}j}. \quad (14)$$

Therefore, the second partial derivative component in equation 8 is solved by substituting equations 10, 11, 13 and 14 into 12.

With respect to resolving the third component in equation 8 we directly present the solution of the derivative, then show the proof of it.

$$\frac{\partial z_j}{\partial Z_j} = (\sum_{k'=1}^{\beta} Z_{k'}^2)(\sum_{k=1}^{\beta} Z_k^2)^{-\frac{3}{2}} \quad \text{for } k' \neq j. \quad (15)$$

***Proof of equation 15**: We can rewrite equation 3 into*

$$z_j = 1/\sqrt{\frac{\sum_{k=1}^{\beta} Z_k^2}{Z_j^2}} = 1/\sqrt{1 + \frac{\sum_{k'=1}^{\beta} Z_{k'}^2}{Z_j^2}} \quad \textit{for } k' \neq j.$$

Now let

$$T = 1 + \frac{\sum_{k'=1}^{\beta} Z_{k'}^2}{Z_j^2}$$

and we can conveniently compute

$$\frac{dz_j}{dT} = -\frac{1}{2}T^{-\frac{3}{2}} = -\frac{1}{2}(1 + \frac{\sum_{k'=1}^{\beta} Z_{k'}^2}{Z_j^2})^{-\frac{3}{2}} \qquad (16)$$

as well as

$$\frac{dT}{dZ_j} = -2Z_j^{-3} \sum_{k'=1}^{\beta} Z_{k'}^2 \quad \text{for } k^{'} \neq j. \qquad (17)$$

Equations 16 and 17 consequently give us the result that

$$\frac{\partial z_j}{\partial Z_j} = \frac{dz_j}{dT} \cdot \frac{dT}{dZ_j} = Z_j^{-3}(\sum_{k'=1}^{\beta} Z_{k'}^2)(\frac{\sum_{k=1}^{\beta} Z_k^2}{Z_j^2})^{-\frac{3}{2}} = (\sum_{k'=1}^{\beta} Z_{k'}^2)(\sum_{k=1}^{\beta} Z_k^2)^{-\frac{3}{2}} \quad \text{for } k^{'} \neq j.$$

■

The last component $\frac{\partial Z_j}{\partial w_{ij}}$ in equation 8 is dependent on the input units. In general it takes the form of

$$\frac{\partial Z_j}{\partial w_{ij}} = \frac{1}{C} \cdot x_i \cdot g(\sum_{i=1}^{\alpha} x_i w_{ij})(1 - g(\sum_{i=1}^{\alpha} x_i w_{ij})) = \frac{1}{C} x_i Z_j (1 - Z_j). \qquad (18)$$

So far we have explicitly derived all components required by equation 8. Therefore, we need to update $\mathbf{w}$ for every instance using equation 8 to minimise the cross-entropy described by equation 7.

Next, we explore an alternative way of learning $\mathbf{w}^*$ by minimising squared error as mentioned previously. Squared error generally takes the following form:

$$\mathcal{E}(\mathbf{w}|\mathcal{D}) = \frac{1}{2}\sum_{n=1}^{N} ||\gamma_n - P(y_n = \mathcal{A}|\mathbf{z})||^2 = \frac{1}{2}\sum_{n=1}^{N} ||1 - \gamma_n - P(y_n = \mathcal{B}|\mathbf{z})||^2. \qquad (19)$$

Because all squared terms in equation 19 are equal or greater than 0, the squared error function reaches its minimum value 0 when $\forall P(y_n = A|\mathbf{z}) \equiv 1$ for $\gamma_n = 1$, and $\forall P(y_n = B|\mathbf{z}) \equiv 1$ for $\gamma_n = 0$. Therefore, the minimum of squared error function also leads to the membership recovery at the hidden layer.

Similarly, as was done with the cross-entropy method, SGD is also used here searching for the minimum, and $\mathbf{w}$ is updated for all instances. The chain rule of partial derivatives is almost the same as equation 8 except for the first term which is now in the form of:

$$-\eta \frac{\partial \mathcal{E}(\mathbf{w}|\mathcal{D})}{\partial P(y = \mathcal{A}|\mathbf{z})} = \eta(\gamma_n - P(y_n = \mathcal{A}|\mathbf{z})).$$

The remaining three terms in equation 8 are exactly the same as the ones used in cross-entropy method (equations 12, 15 and 18).

III. Algorithm for Training the DCLN Model

To update $\mathbf{w}$ with equation 8 under the current neural network setting, DCLN firstly calculates the two centroids at the hidden layer via the randomly initiated weights $\mathbf{w}^0$, Figure 1(c). Two centroids are temporarily fixed for the current *epoch* (going through every instance in the data exactly once). $\mathbf{w}$ is updated with $\mathbf{w}^n \leftarrow \mathbf{w}^{n-1} + \Delta\mathbf{w}$ for every instance within the current epoch one by one in a random order by equation 8. This will recover the membership probability for every instance to some extent as we discussed previously, Figure 1(d). After that, the two centroids are re-calculated, Figure 1(e), and the above procedure is repeated until some pre-defined number of epochs is reached. The DCLN training algorithm is detailed in Algorithm 1.

Algorithm 1 Training DCLN with Cross-Entropy (CE) and Squared Error (SE) Minimisation

Require: Input data $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, $y \in \{\mathcal{A}, \mathcal{B}\}$.
Set optimisation method, either CE or SE.
Set hyperparameters: hidden layer dimension β; learning rate η; sigmoid slope factor C; maximum $epoch$.
Randomly initiate $\mathbf{w}$: $w_{ij} \in [-0.5, 0.5]$, where $i \in [0, \alpha]$ for input layer and $j \in [1, \beta]$ for hidden layer.
$\mathbf{w}^*$, $\mathbf{v}_{\mathcal{A}}^*$ and $\mathbf{v}_{\mathcal{B}}^* \leftarrow \varnothing$; $min \leftarrow 1$.
repeat
  for each instance $\mathbf{x}_n$, where $n \in [1, N]$ **do**
    $\mathbf{Z}_n$: $Z_{nj} \leftarrow 1/(1 + e^{-\frac{1}{C}(\mathbf{w}_j^T \mathbf{x}_n)})$ for $j \in [1, \beta]$.
    $\mathbf{z}_n$: $z_{nj} \leftarrow Z_{nj}/\sqrt{\sum_{k=1}^{\beta} Z_{nk}^2}$ for $j \in [1, \beta]$.
  end for
  for $y \in \{\mathcal{A}, \mathcal{B}\}$ **do**
    $\mathbf{v}_y$: $v_{yj} \leftarrow (\sum_{n \in y} z_{nj})/\sqrt{\sum_{k=1}^{\beta}(\sum_{n \in y} z_{nk})^2}$, for $j \in [1, \beta]$.
  end for
  if $\langle \mathbf{v}_{\mathcal{A}}, \mathbf{v}_{\mathcal{B}} \rangle < min$ **then**
    $\mathbf{w}^* \leftarrow \mathbf{w}$; $\mathbf{v}_{\mathcal{A}}^* \leftarrow \mathbf{v}_{\mathcal{A}}$; $\mathbf{v}_{\mathcal{B}}^* \leftarrow \mathbf{v}_{\mathcal{B}}$; $min \leftarrow \langle \mathbf{v}_{\mathcal{A}}, \mathbf{v}_{\mathcal{B}} \rangle$.
  end if
  for every instance $\mathbf{x}_n$ in random order **do**
    $U \leftarrow \langle \mathbf{z}_n, \mathbf{v}_{\mathcal{A}} \rangle$; $V \leftarrow \langle \mathbf{z}_n, \mathbf{v}_{\mathcal{A}} + \mathbf{v}_{\mathcal{B}} \rangle$.
    for $j \in [1, \beta]$ **do**
      $U^{'} \leftarrow v_{\mathcal{A}j}$; $V^{'} \leftarrow v_{\mathcal{A}j} + v_{\mathcal{B}j}$.
      for $i \in [0, \alpha]$ **do**
        $\Delta\mathbf{w} \leftarrow (U^{'}V - V^{'}U)/V^2 \cdot (\sum_{k'=1}^{\beta} Z_{nk'}^2) \cdot (\sum_{k=1}^{\beta} Z_{nk}^2)^{-\frac{3}{2}} \cdot \frac{1}{C} \cdot x_{ni} \cdot Z_{nj}(1 - Z_{nj})$, for $k^{'} \neq j$.
        if $\mathbf{x}_n \in \mathcal{A}$ **then**
          $w_{ij}^{(CE)} \leftarrow w_{ij}^{(CE)} + \eta \cdot V/U \cdot \Delta\mathbf{w}$.
          $w_{ij}^{(SE)} \leftarrow w_{ij}^{(SE)} + \eta \cdot (1 - U/V) \cdot \Delta\mathbf{w}$.
        else
          $w_{ij}^{(CE)} \leftarrow w_{ij}^{(CE)} - \eta/(1 - U/V) \cdot \Delta\mathbf{w}$.
          $w_{ij}^{(SE)} \leftarrow w_{ij}^{(SE)} - \eta \cdot U/V \cdot \Delta\mathbf{w}$.
        end if
      end for
    end for
  end for
until epoch=h.
output $\mathbf{w}^*$, $\mathbf{v}_{\mathcal{A}}^*$ and $\mathbf{v}_{\mathcal{B}}^*$.

One thing to note here is that for very imbalanced data, assuming $n_1 = 10$ instances for $\mathcal{A}$ and $n_2 = 1000$ instances

Fig. 2. Non-Linearly Separable Synthetic Data

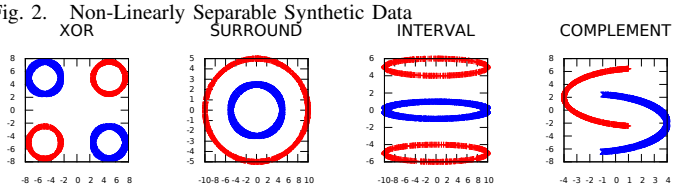


Fig. 3. $\langle \mathbf{v}_{\mathcal{A}}, \mathbf{v}_{\mathcal{B}} \rangle$ for 100 Epochs on Synthetic Data

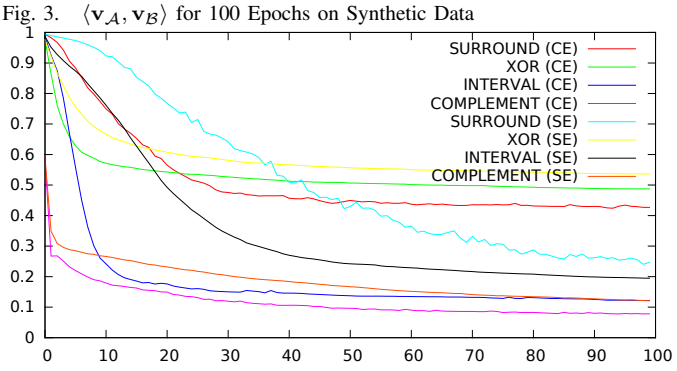
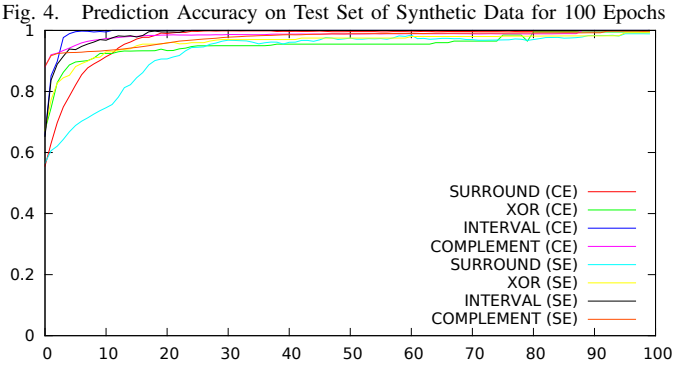


Fig. 4. Prediction Accuracy on Test Set of Synthetic Data for 100 Epochs



for $\mathcal{B}$, instances in class $\mathcal{A}$ have much lower chances of getting into the update of $\mathbf{w}$. The update is dominated by instances in class $\mathcal{B}$, and hence the update is biased to the movement of $\mathbf{v}_{\mathcal{B}}$. To balance this, we can simply manipulate the learning rate by using an "adjusted" learning rate η^*, instead of η, differently to the two classes. Here, in Algorithm 1, we may use $\eta^*_{\mathcal{B}} = \eta \times n_1/n_2$ for $\mathbf{x} \in \mathcal{B}$ if $n_1 < n_2$ when updating $\mathbf{w}$. And symmetrically, use $\eta^*_{\mathcal{A}} = \eta \times n_2/n_1$ for $\mathbf{x} \in \mathcal{A}$ if $n_2 < n_1$. This is a simple idea of *slowing down* the update for the class with too many samples. The learning rate is one of the most important parameters and deserves more investigation in future.

Generally it is expected to see that the two centroids, as the class representatives, are moving away from each other as the training proceeds. During training, the two centroids $\mathbf{v}^*_{\mathcal{A}}$ and $\mathbf{v}^*_{\mathcal{B}}$ with maximum distance (i.e. minimum inner product between each other) are recorded along with the corresponding $\mathbf{w}^*$. Once the DCLN is trained, we can map an unknown instance $\mathbf{x} \mapsto \mathbf{z}$ via equations 1, 2 and 3 using calculated $\mathbf{w}^*$, and then classify the instance by simply assigning it to the nearest class centroid at the hidden layer, i.e., comparing $\langle \mathbf{z}, \mathbf{v}^*_{\mathcal{A}} \rangle$ and $\langle \mathbf{z}, \mathbf{v}^*_{\mathcal{B}} \rangle$.

To test the model's classification ability and to explore whether the two centroids are moving apart from each other

TABLE I. List of Data Used in Benchmarking The Classification Performance

UCI Data	Description	Class $\mathcal{A}$	Class $\mathcal{B}$
Bupa	Liver disorders	200	145
Ionosphere	Radar Returns from the Ionosphere	225	126
Sonar	Connectionist Bench, Mines VS. Rocks	111	97
Wpbc*	Breast Cancer Wisconsin, Prognostic	151	47
Wdbc*	Breast Cancer Wisconsin, Diagnostic	357	212
SPECT	SPECT Heart Images	212	55
SPECTF	SPECTF Heart Images	212	55

*The first column of data is sample ID which has been removed before processing.

in practice, we created four *non-linearly separable* synthetic data shown in Figure 2. For each data, 2000 instances (points) are created separately for the two classes. Each point has two features: horizontal and vertical axis coordinates. Points in red belongs to class $\mathcal{A}$ and points in blue are in class $\mathcal{B}$. For each class, we randomly select 1000 instances for training and use the rest 1000 instances for testing.

We apply DCLN with both CE and SE optimisations to each data. For each trial, the DCLN is set for 100 epochs with hyperparameters $\beta = 50$, $\eta = 0.1$ and $C = 0.1$. The inner product $\langle \mathbf{v}_{\mathcal{A}}, \mathbf{v}_{\mathcal{B}} \rangle$ and the prediction accuracy on the test set are recorded at each epoch. A total of 50 trials were done. As the training proceeds, the average inner product and the classification accuracy at each epoch are illustrated in Figure 3 and Figure 4 respectively. It is evident that all data experienced the expected decline of $\langle \mathbf{v}_{\mathcal{A}}, \mathbf{v}_{\mathcal{B}} \rangle$ which clearly indicates that the two centroids are moving in the opposite way as we expected. And also the classification accuracy on the test sets quickly reach the maximum 1 for nearly all data. The variances between these curves may be subject to different factors. Some synthetic data may be more sensitive to a specific hyperparameter setting, and the proper representations for some data may also affect the convergence speed. For simplicity, we just used a fixed parameter setting in all cases to demonstrate the idea at this stage.

IV. Classification with Real World Datasets

The previous discussion demonstrates the effectiveness and the expected behaviour of DCLN on several simple synthetic data. Now we examine its practicability in real world classification problems. Below we apply DCLN to 7 real world binary classification datasets obtained from UCI Machine Learning Repository [16] with sample size details listed in Table I. The raw feature values are used as input in all experiments.

We benchmark DCLN against two popular shallow models: Support Vector Machine (SVM) [10], [11] with popular LIBSVM library [13], and Extreme Learning Machine (ELM) [12]. We downloaded the LIBSVM 3.18 from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and the MATLAB version of ELM from http://www.ntu.edu.sg/home/egbhuang/elm_codes.html.

To report the best classification result for each model on each data, selection of hyperparameters for each model is obtained via the following experiment setting: A total of 50 trials have been done for the experiment. For each trial, we randomly choose 75% samples (sampling without repeating) as the training set, and use the remaining 25% as the test set. For SVM model, three widely used kernels such as Linear, Radial Basis Function (RBF) and Polynomial kernels are tested in the experiment. For ELM, various numbers of neurons are

TABLE II. MODEL COMPARISONS FOR CLASSIFICATION PERFORMANCE

		SVM			ELM	DCLN	
Data	Metric	Linear	RBF	Poly	Neuron	CE	SE
Bupa	**Bal-Accuracy**	0.67	0.60	0.66	0.62	0.67	**0.69**
	Accuracy	0.69	0.65	0.67	0.63	0.69	0.70
Ionosphere	**Bal-Accuracy**	0.85	**0.93**	0.81	0.83	0.90	0.90
	Accuracy	0.87	0.94	0.86	0.87	0.92	0.92
Sonar	**Bal-Accuracy**	0.76	**0.85**	0.50	0.76	0.81	0.81
	Accuracy	0.76	0.85	0.53	0.76	0.82	0.82
Wpbc	**Bal-Accuracy**	0.64	0.50	0.57	0.54	0.62	**0.65**
	Accuracy	0.79	0.77	0.66	0.76	0.64	0.61
Wdbc	**Bal-Accuracy**	**0.94**	0.50	0.84	0.85	0.90	0.90
	Accuracy	0.95	0.63	0.87	0.87	0.90	0.91
SPECT*	**Bal-Accuracy**	0.78	0.77	0.65	0.73	**0.79**	0.77
	Accuracy	0.76	0.74	0.36	0.70	0.76	0.77
SPECTF*	**Bal-Accuracy**	0.61	0.65	0.58	0.60	0.72	**0.77**
	Accuracy	0.72	0.47	0.73	0.51	0.70	0.65

*The 75:25 split for training and testing sets doesn't apply. The source of data has provided the training and testing sets which are used directly in the experiment.

tested. For DCLN, we assess both the CE and SE minimisation methods. The details of hyperparameters tuning and DCLN sample code are in [14]. Once the best hyperparameters setting is determined in the training data, we predict on the test data and report the average Balanced Accuracy (Bal-Accuracy) and Accuracy in Table II. As can be seen from Table I, it is obvious that the samples are very imbalanced between the two classes in most cases, and hence the Bal-Accuracy is the preferred metric in comparing the classification performance.

As illustrated in Table II, DCLN received four top Bal-Accuracy, while SVM is ranked highest in the other three by Bal-Accuracy. For Bupa, DCLN$_{SE}$ reported the highest Bal-Accuracy 0.69, followed by SVM$_{Linear}$'s 0.67. The Accuracy results are similar between the DCLN$_{SE}$ and SVM$_{Linear}$, with 0.70 and 0.69 respectively. For Ionosphere, SVM$_{RBF}$ reported the best Bal-Accuracy 0.93, followed closely by DCLN's 0.90. The accuracy between the two are also close, with 0.94 and 0.92 respectively. Similarly, for Sonar, SVM$_{RBF}$ received the top Bal-Accuracy and Accuracy at 0.85, followed closely by DCLN's 0.81 and 0.82 for Bal-Accuracy and Accuracy. For Wpbc, DCLN$_{SE}$ received the highest Bal-Accuracy 0.65, followed by SVM$_{Linear}$'s 0.64, but SVM$_{Linear}$ reported a high Accuracy 0.79 in this scenario. For Wdbc, SVM$_{Linear}$ reported the highest Bal-Accuracy 0.94, followed by DCLN$_{SE}$'s 0.90. In terms of the Accuracy, SVM$_{Linear}$ is higher than DCLN$_{SE}$ by 4%. For SPECT, DCLN$_{CE}$ reported the best Bal-Accuracy 0.79, followed by SVM$_{Linear}$'s 0.78. On the other hand, DCLN$_{SE}$ received the top 0.77 in Accuracy, followed by SVM$_{Linear}$'s 0.76. Lastly, for SPECTF, DCLN$_{SE}$ reported highest Bal-Accuracy 0.77, followed by SVM$_{RBF}$'s 0.65 in this case.

In summary, the differences between Bal-Accuracy and Accuracy are relatively small, and DCLN is stable and quite competitive to other shallow classifiers.

V. HIGH-LEVEL DISCRIMINATIVE CONCEPTS LEARNING

Here we reveal another intrinsic capability of DCLN. That is the input layer by itself can directly tell the high-level discriminative features that constitute the differential concepts between the two classes.

Recall from previous discussion, we only have one input layer and one hidden layer. There exists a mapping $\mathbf{x} \mapsto \mathbf{z}$ via $\mathbf{w}$, and the classification is entirely decided by $\mathbf{z}$. An element z_j receives its value directly from the normalised $g(\mathbf{w}_j^T\mathbf{x})$ (Equations 1, 2 and 3). Therefore, each feature x_i contributes to a hidden neuron j by w_{ij} at the input layer, and its contribution to the vector $\mathbf{z}$ (classification deciding factor) at the input layer is characterised by $\sum_{j=1}^{\beta} w_{ij}$ which becomes a natural measurement of the relative contribution of x_i to the classification. We denote $\mathbf{\Omega}$ to this contribution: $\mathbf{\Omega} := \{\Omega_i = \sum_{j=1}^{\beta} w_{ij}\}$ for features $\{x_i\}$ of an instance $\mathbf{x}$.

On the other hand, as we demonstrated previously, $\mathbf{v}_{\mathcal{A}}$ and $\mathbf{v}_{\mathcal{B}}$ are moving away from each other. This indicates the structural differences between the two classes are learned and amplified gradually at the layer $\mathbf{z}$. That means there must be a $\mathbf{w}$ guiding the mapping $\mathbf{x} \mapsto \mathbf{z}$ that is responsible for this difference amplification. Through $\mathbf{w}$, the structural differences at the input layer can be aware, transformed and gradually amplified at the layer $\mathbf{z}$. Since the structural differences in $\mathbf{x}$ can be aware through $\mathbf{w}$, and the natural measurement of the relative contribution for the features at the input layer is quantified by $\mathbf{\Omega}$ which is made up of $\mathbf{w}$, then a natural hypothesis would be: The contribution $\mathbf{\Omega}$ can quantify the structural differences for the features at the input layer, and reflect what features are relatively important in separating the two classes via such quantification.

We empirically demonstrate that the $\mathbf{\Omega}$ quantifies such structural differences between the two classes directly from the features at the input layer. Interestingly, such quantification reveals the human-understandable, high-level discriminative concepts between the two classes in the following two image recognition exercises:

We downloaded the MNIST Handwritten Digit Database [17] containing 60,000 training samples and 10,000 testing samples for the 10 handwritten digit classes $0-9$, each in $28 \times 28 = 784$ pixels (features). Examples are shown in Figure 5. For every pair of digit classes, we train DCLN using instances of the two digit classes with parameters $\beta = 10, \eta = 1, epochs = 200$ and $C = 10000$.

Once the training is finished, we receive the optimal $\mathbf{w}^*$ and we calculate $\Omega_i = \sum_{j=1}^{\beta} w_{ij}^*$ for every feature x_i. After that we plot in Figure 6 with calculated $\mathbf{\Omega}$ values to see if any visually meaningful concepts between the two classes can be reflected by $\mathbf{\Omega}$ at the original input layer. As can be seen from Figure 6, the top-left corner is the $\mathbf{\Omega}$ plot for digit class 0 (with 5923 training samples) and class 1 (with 6742 training samples) at the input layer. In this plot, the overall shape looks like the *"Layered Union"* of 0 and 1, in which the digit 1 is floating on top of the digit 0. If we only look at the points with $\mathbf{\Omega}$ lower than -0.001, then visually the informative features are clearly separated from the background. Similar things can be seen in other digit class pairs in Figure 6. These plots suggest that from 784 pixels (features), $\mathbf{\Omega}$ is able to visually identify the informative features that constitute the differential concepts of the two classes. The selection of $\mathbf{\Omega}$ spectra is key to isolating the differential concepts from the noises, and deserve further investigations.

Fig. 5. The MNIST Handwritten Digits

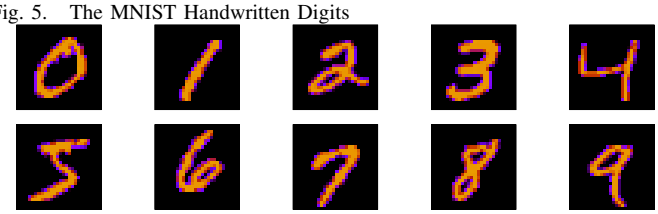


Fig. 6. Discriminative Concepts Learned for Pairs of MNIST Digit Classes

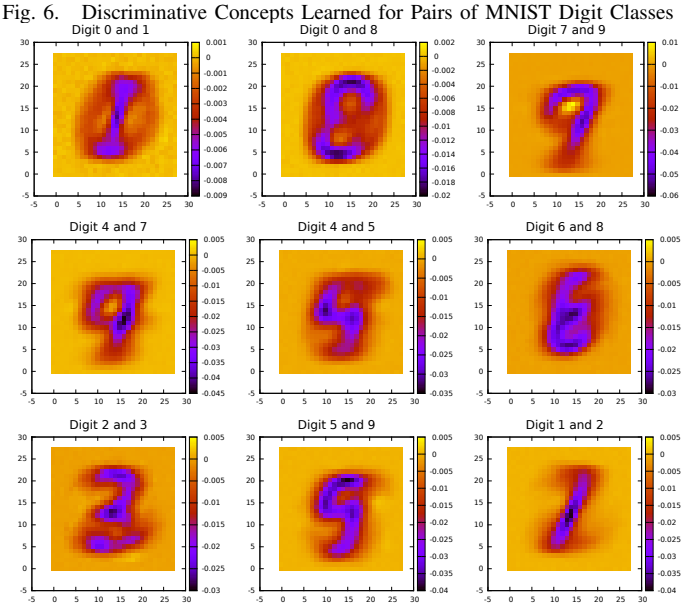
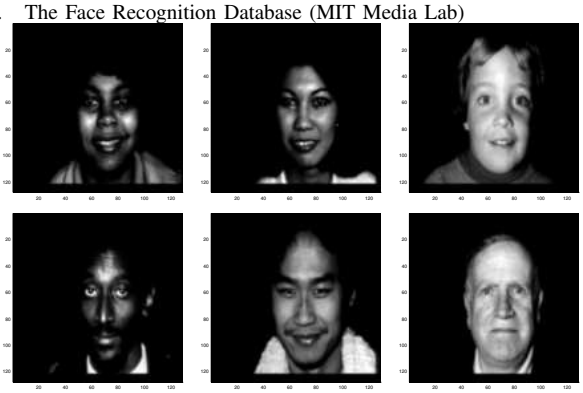
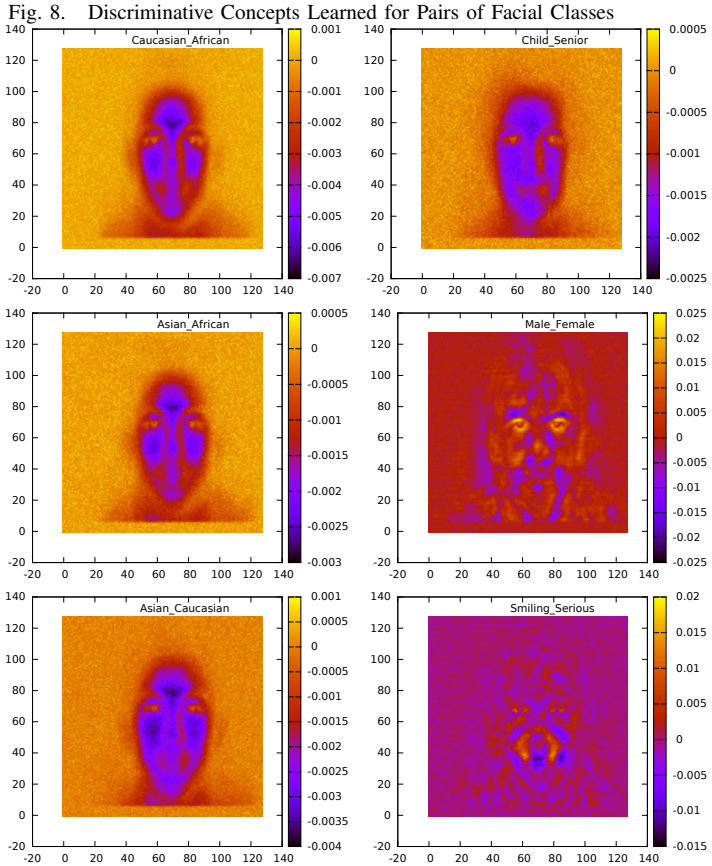


Fig. 7. The Face Recognition Database (MIT Media Lab)



To provide further support, we apply it to a more complicated exercise using Face Recognition data [18] developed by the Media Lab of Massachusetts Institute of Technology (MIT). This data contains about 4000 human facial images with 2000 training samples and 2000 testing samples, each in 128×128 grayscale pixels (features). Images are categorised into different facial classes such as Female/Male, Child/Senior, Smiling/Serious and African/Asian/Caucasian. Some examples are illustrated in Figure 7. Similar like what we did with MNIST experiment, we plot the $\mathbf{\Omega}$ for each of the following facial class pairs: 1) Training DCLN on class pairs of Male/Female and Smiling/Serious with $\beta = 10, \eta = 10, epochs = 100$ and $C = 1000$; 2) Training on class pairs of Asian/African and Child/Senior with $\beta = 20, \eta = 1, epochs = 300$ and $C = 10000$; 3) Training on class pairs of Caucasian/African and Asian/Caucasian with $\beta = 30, \eta = 10, epochs = 100$ and $C = 10000$. (NB: here we calculated the adjusted learning rate η^* from η, and use it in the algorithm as we have discussed previously.)

Fig. 8. Discriminative Concepts Learned for Pairs of Facial Classes



The $\mathbf{\Omega}$ plots are obtained after the training, and illustrated in Figure 8. Intuitively, $\mathbf{\Omega}$ captured the high-level discriminative concepts for each of the facial class pairs. For pairs of ethnic groups, the subtle facial differences in the areas of forehead, eyes, cheeks, nose and jaw are highlighted. The pairs of Caucasian/African and Asian/African are quite similar except for the slight differences in the area of the cheeks and forehead. However the pair of Asian/Caucasian shows large facial differences than the other two ethnic pairs. For Male/Female, on the other hand, the key areas such as eye, beard and eyebrow are identified. Such areas are also key to humans to tell the genders. For Smiling/Serious, as in our common sense, the mouth area stands out as we expected. Compared to the general concept of human face learned by the "grandmother neuron" in deep architecture [5], DCLN is able to illustrate the clear discriminative concepts using only $\mathbf{\Omega}$ values directly at the input layer. The learned concepts are naturally the key to the binary classification.

So far we have visually demonstrated the role of $\mathbf{\Omega}$ in revealing the high-level differential concepts in binary classifications. The true predictive power of the identified informative features needs to be further verified in real world applications. To give a simple example, we show how $\mathbf{\Omega}$ is useful

in searching for the informative biomarkers in *case-control* genomics data as a real diagnostic genomics application in the next section.

VI. Discriminative Biomarkers Identification in Genome-Wide Association Studies

In human DNA, the genetic variants between the individuals are called the Single Nucleotide Polymorphism (SNPs). As an example, the A-T base pair at a given genome location in a population may differ from the C-G base pair in another population at the same genome location. To explain how SNP variants are affecting the disease risk, Genome-Wide Association Studies (GWAS) is employed to assay millions of SNPs for multiple individuals at the same time [19]. In this case, GWAS data contains thousands of individuals, in which the people carrying the disease are termed *cases*, while the healthy people are called *controls*. Individuals are genotyped at million genome locations. At a genome location for a given individual, one of the three possible categorical genotypes 0, 1 and 2 representing the *dominant homozygous (AA)*, *heterozygous (Aa)*, and *recessive homozygous (aa)* respectively, is recorded for the SNP at that location. Therefore, GWAS is a high dimensional data with millions of SNPs, and thousands of samples either in case or in control group.

It is broadly believed that SNPs do not individually affect the disease [20], but rather interacting with each other with non-negligible synergistic effects [21]. Extensive efforts have been made to exhaustively examine all SNP pairs in GWAS data and select the pairs with strong joint effects [22]–[27]. In a nutshell, existing methods take the case-control GWAS data, analyse every SNP pair by a predefined statistical test, and generate a list of surviving pairs ranked by the test. The reported SNP pairs are expected to be the key features to differentiating between case and control [26], and an interactions network can be built and visualised from that point. However the validation for their segregation ability is largely missing. As a simple case study, we demonstrate how DCLN is applied to validate the discriminative power for the SNP pairs reported by the widely used χ^2 test [29]–[31], and how a discriminative interactions network can be built on top. We test the idea on Type 1 Diabetes GWAS (Wellcome Trust Case Control Consortium data [28]) with 449,471 SNPs, 1963 cases and 2938 controls.

Firstly, we randomly select 75% samples as the training set (sampling without repeating) and use the remaining 25% samples for testing. The raw categorical genotype values are used as input. We run a fast GPU screening tool GWIS_{FI} [27] with χ^2 statistics to score every SNP pairs on the training set. We then focused on 17 unique SNPs that constitute the top 20 SNP pairs reported by GWIS_{FI}. We remove all SNPs other than these 17 SNPs from the training and testing sets, and generate the new training and testing *sub sets* containing only these 17 SNPs. We run DCLN 50 times with parameters $\beta = 50$, $\eta = 100$, $epochs = 3000$ and $C = 100$ on the training *sub set*. For each feature x_i calculate the average value of $\mathbf{\Omega}_i$ over the 50 runs, which leads to $\mathbb{E}[\mathbf{\Omega}_i] = \{-41.3, -82.9, -4.9, -121.3, -73.1, -4.1, -110.7, -75.5, -115.9, -128.4, -65.2, -99.9, -139.9, -72.8, -100.9, -91.8, -5.5\}$ for $\{x_1, ..., x_{17}\}$. Assuming the discriminative information can be reflected from $\mathbf{\Omega}$, then the question is what $\mathbf{\Omega}$ spectra can separate the discriminative features from the noises. For the purpose of simplest demonstration, we simplify the $\mathbf{\Omega}$ spectra to a single cut-off -75 and generate *mini sets* from *sub set* with *mini set* A containing all x_i for $\mathbb{E}[\mathbf{\Omega}_i] <$ cutoff , and *mini set* B containing x_i for $\mathbb{E}[\mathbf{\Omega}_i] >$ cutoff. In this case, *mini set* A contains 10 SNPs, and *mini set* B contains 7 SNPs. So far, we don't know whether and which *mini set* contains the discriminative features. To test it, we do the following experiment.

Fig. 9. DCLN Suggested SNP Interactions Network for T1D WTCCC

We use LIBSVM as an external verification tool. For the first step, the SVM's Linear, RBF and Polynomial kernels with default settings are trained on the training *sub set* and test on the testing *sub set* with following performance: Linear kernel (Bal-Accuracy 0.658, AUC 0.734); RBF kernel (Bal-Accuracy 0.676, AUC 0.745); Polynomial of degree 2 (Bal-Accuracy 0.666, AUC 0.750). If one *mini set* contains most of the discriminative SNPs and the other contains most of the noise features, then the classification performance on one *mini set* should be at least close to the performance on the *sub set*, and the performance for the other *mini set* should be close to a random guess. Therefore, we apply SVM to both *mini sets* obtaining the following results. For *mini set* A we have Linear kernel (Bal-Accuracy 0.651, AUC 0.713); RBF kernel (Bal-Accuracy 0.647, AUC 0.732); Polynomial of degree 2 (Bal-Accuracy 0.657, AUC 0.725). For *mini set* B we have Linear kernel (Bal-Accuracy 0.514, AUC 0.558); RBF kernel (Bal-Accuracy 0.559, AUC 0.553); Polynomial of degree 2 (Bal-Accuracy 0.521, AUC 0.573). From these performance numbers, it is obvious that the performance on *mini set* A is very close to the *sub set*, and hence owns most of the informative features. On the other hand, the performance on *mini set* B is close to the random guess, and indicates that *mini set* B contains most of the noises. Finally the discriminative SNP interactions network for the top 20 pairs made by χ^2 test is illustrated in Figure 9, in which the SNPs in red are 10 informative SNPs of *mini set* A, suggested by DCLN.

VII. Conclusion

In this paper, we demonstrated that the proposed DCLN model served as both an effective binary classifier and a learner for high-level discriminative concepts. From the illustrated discriminative concepts, it is clear that shallow model is also able to tell the high-level concepts without relying on multi-layer learning.

The other advantage of DCLN is its lower computational complexity, requiring only $\mathbf{w}$ to be updated between the input and hidden layer during the training. In our facial recognition exercises, only about an hour is required to train the DCLN using a low-end laptop comparing to hundreds of machines required usually by a typical deep learning model.

This enables the possibilities in large-scale applications such as high-dimensional genomics analysis. The algorithm can be further accelerated using multicore computing techniques such as general-purpose computing on graphics processing units (GPGPU) to suit such applications.

ACKNOWLEDGEMENT

We thank Adam Kowalczyk, John Hopper, Miroslaw Kapuscinski, Christopher Leckie, Sheng Wang and XiaoLan Yu for their help with the manuscript and preparing the data.

This study makes use of data generated by the Wellcome Trust Case-Control Consortium. A full list of the investigators who contributed to the generation of the data is available from www.wtccc.org.uk.

This work has received the Google Ph.D. travel prize from Google Australia.

REFERENCES

[1] Haykin, S., *Neural Networks: A Comprehensive Foundation*, MacMillan, 1994.

[2] Arel, I., Rose, D.C., and Karnowski, T.P., *Deep machine learning–A new frontier in artificial intelligence research*, IEEE Comput. Intell. Mag. **5**(4), 13–18, 2010.

[3] Bengio, Y., *Learning deep architectures for AI*, Foundat. and Trends Mach. Learn. **2**(1), 1–127, 2009.

[4] Farabet, C., Couprie, C., Najman, L., and LeCun, Y., *Learning Hierarchical Features for Scene Labeling*, Pattern Analysis and Machine Intelligence, IEEE Transactions on. **35**(8), 1915–1929, 2013.

[5] Le, Q., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G., Dean, J., Ng, A., *Building high-level features using large scale unsupervised learning*, ICML, 2012.

[6] Olshausen, B.A., and Field, D.J., *Emergence of simple-cell receptive field properties by learning a sparse code for natural images*, Nature, **381**, 607–609, 1996.

[7] Lee, H., Battle, A., Raina, R., and Ng, A.Y., *Efficient sparse coding algorithms*, Proc. Adv. Neural Inf. Process. Syst. 801–808, 2007.

[8] Lee, H., Grosse, R., Ranganath, R., and Ng, A., *Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations*, ICML, 2009.

[9] Ba, J., and Caruana, R., *Do deep nets really need to be deep?*, Proc. Adv. Neural Inf. Process. Syst. 2654–2662, 2014.

[10] Cortes, C., and Vapnik, V., *Support-vector network*, Mach.Learn. **20**, 273–297, 1995.

[11] Vapnik, V., *Statistical learning theory*, Wiley, 1998.

[12] Huang, G. B., Zhu, Q. Y., and Siew, C. K., *Extreme learning machine: Theory and applications*, Neurocomputing, **70**, 489–501, 2006.

[13] Chang, C. C., and Lin, C. J., *LIBSVM: A library for support vector machines*, ACM Transactions on Intelligent Systems and Technology, **2**(3), 27:1–27:27, 2011.

[14] Wang, Q., Young, S., Harwood, A. and Ong, C.S, *Discriminative Concept Learning Network: Sample Source Code and Supplement*, [Online], https://sourceforge.net/projects/dcln/.

[15] Han, J., and Moraga C., *The influence of the sigmoid function parameters on the speed of backpropagation learning*, From Natural to Artificial Neural Computation, pp. 195–201. Springer, Berlin, 1995.

[16] Bache, K., and Lichman, M., *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml, Irvine, CA: University of California, School of Information and Computer Science, 2013.

[17] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, **86**(11), 2278–2324, 1998.

[18] Szummer, M., and Kapoor, A., *Face Recognition Project*, [Online] Updated on Oct 31 2002, http://courses.media.mit.edu/2004fall/mas622j/04.projects/faces/ (Accessed: Aug 6 2014).

[19] Hirschhorn, J.N., and Daly, M.J., *Genome-wide association studies for common diseases and complex traits*, Nat Rev Genet, **6**(2), 95–108, 2005.

[20] Culverhouse, R., Suarez, B.K., Lin, J., and Reich, T., *A perspective on epistasis: limits of models displaying no main effect*, Am. J. Hum. Genet. **70**, 461–471, 2002.

[21] Marchini, J., Donnelly, P., and Cardon, L.R., *Genome-wide strategies for detecting multiple loci that influence complex diseases*, Nat Genet. **37**(4), 413–417, 2005.

[22] Hu, X., Liu, Q., Zhang, Z., Li, Z., Wang, S., He, L., and Shi, Y., *SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder*, Cell Research, **20**(7), 854–857, 2010.

[23] Kam-Thong, T., Czamara, D., Tsuda, K., Borgwardt, K., Lewis, C.M., Erhardt-Lehmann, A., Hemmer, B., Rieckmann, P., Daake, M., Weber, F., Wolf, C., Ziegler, A., Pütz, B., Holsboer, F., Schölkopf, B., and Müller-Myhsok, B., *EPIBLASTER-fast exhaustive two-locus epistasis detection strategy using graphical processing units*, European Journal of Human Genetics, **19**, 465–471, 2011.

[24] Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M.A., Bender, D., Maller, J., Sklar, P., de Bakker, P.I., Daly, M.J., and Sham, P.C., *PLINK: a tool set for whole-genome association and population-based linkage analyses*, American journal of human genetics, **81**(3), 559–575, 2007.

[25] Canela-Xandri, O., Juli, A., Gelp, J. L., and Marsal, S., *Unveiling case-control relationships in designing a simple and powerful method for detecting gene-gene interactions*, Genetic Epidemiology, **36**(7), 710–716, 2012.

[26] Goudey, B., Rawlinson, D., Wang, Q., Shi, F., Ferra, H., Campbell, R., Stern, L., Inouye, M., Ong, C.S., and Kowalczyk, A., *GWIS - model-free, fast and exhaustive search for epistatic interactions in case-control gwas*, BMC Genomics, **14**(Suppl 3), S10, 2013.

[27] Wang, Q., Fan, S., Kowalczyk Andrew, Campbell., R.M., Goudey, B., Rawlinson, D., Harwood, A., Herman, F., and Kowalczyk Adam., *GWISFI: A universal GPU interface for exhaustive search of pairwise interactions in case-control GWAS in minutes*, Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on, 403–409. doi:10.1109/BIBM.2014.6999192.

[28] The Wellcome Trust Case-Control Consortium., *Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls*, Nature, **447**(7145), 661–678, 2007.

[29] Agresti, A., *Categorical Data Analysis*, Wiley series in probability and statistics, Wiley Interscience, 2 edition, 2002.

[30] Wang, Z., Wang, Y., Tan, K.-L. L., Wong, L., and Agrawal, D., *eCEO: an efficient Cloud Epistasis cOmputing model in genome-wide association study*, Bioinformatics (Oxford, England), **27**(8), 1045–1051, 2011.

[31] Chen, L., Yu, G., Miller, D. J., Song, L., Langefeld, C., Herrington, D., Liu, Y., and Wang, Y., *A Ground Truth Based Comparative Study on Detecting Epistatic SNPs*, Proceedings. IEEE International Conference on Bioinformatics and Biomedicine, **1-4**, 26–31, 2009.