Two Dimensional Stochastic Configuration Networks for Image Data Analytics

Ming Li, *Member, IEEE*, Dianhui Wang*, *Senior Member, IEEE*

***Abstract*—Stochastic configuration networks (SCNs) as a class of randomized learner model have been successfully employed in data analytics due to its universal approximation capability and fast modelling property. The technical essence lies in stochastically configuring the hidden nodes (or basis functions) based on a supervisory mechanism rather than data-independent randomization as usually adopted for building randomized neural networks. Given image data modelling tasks, the use of one-dimensional SCNs potentially demolishes the spatial information of images, and may result in undesirable performance. This paper extends the original SCNs to a two-dimensional version, termed 2DSCNs, for fast building randomized learners with matrix inputs. Some theoretical analyses on the goodness of 2DSCNs against SCNs, including the complexity of the random parameter space and the superiority of generalization, are presented. Empirical results over one regression example, four benchmark handwritten digit classification tasks, two human face recognition datasets, as well as one natural image database, demonstrate that the proposed 2DSCNs perform favourably and show good potential for image data analytics.**

***Index Terms*—2D stochastic configuration networks, Randomized algorithms, Image data analytics.**

I. Introduction

With the rising wave of deep learning, neural networks, by means of their universal approximation capability and well-developed learning techniques, have achieved great success in data analytics [1]. Usually, the input layer of a fully connected neural network (FCNN) is fed with vector inputs, rather than two-dimensional matrices such as images or higher dimensional tensors like videos or light fields [2]–[4]. Technically, the vectorization operation makes the dot product (between the inputs and hidden weights) computationally feasible but inevitably induces two drawbacks: (i) the dimensionality curse issue when the number of training samples is limited; (ii) the loss of spatial information of the original multi-dimensional input. Although convolutional neural networks (CNNs) have brought about some breakthroughs in image data modelling, by means of their good potential in abstract feature extraction, power in local connectivity and parameter sharing, etc. [5], the development of FCNNs with matrix inputs (or multidimensional inputs in general) is of great importance in terms of both theoretical and algorithmic viewpoints. In [6], Gao et al. first nominated the term matrix neural networks (MatNet) and extended the conventional back-prorogation (BP) algorithm [7] to a general version that is capable of dealing with 2D inputs. The empirical results in [6] and their parallel work [8] demonstrate the advantages of MatNet for image data modelling. Obviously, MatNet may still suffer some intrinsic drawbacks of gradient descent-based approaches such as local minimum and low convergence rate. This indicates an urgent need to develop fast learning techniques to build FCNNs with 2D inputs, as an immediate motivation of this work.

Randomized learning techniques have demonstrated their great potential in fast building neural network models and algorithms with less computational cost [9]. In particular, random vector functional-link (RVFL) networks developed in the early 90s [10], [11] and stochastic configuration networks (SCNs) proposed recently [12] are two representatives of the randomized learner models. Technically, the success of SCNs and their extensions [13]–[17] in fast building universal approximators with randomness has been extensively demonstrated on data analytics. Generally, the very heart of the SCN framework lies in the supervisory (data-dependent) mechanism used to stochastically (and incrementally) configure the input weights and biases from an appropriate 'support range'. In the presence of multi-dimensional input, especially 2D input (e.g. images), both RVFL networks and SCNs require a regular vectorization operation before feeding the given input signal into the neural network model. The authors of [18] made a first attempt on two dimensional randomized learner models by developing RVFL networks with matrix inputs (termed 2DRVFL) with applications in image data modelling. Although some advantages of the 2D model are experimentally demonstrated, the concerned methodology/framework still suffers from the drawbacks of RVFL networks [19], [20].

This paper develops two dimensional SCNs on the basis of our previous SCN framework [12], aiming to fast build 2D randomized learner models that are capable of resolving data anlytics with matrix inputs. We first provide a detailed algorithmic implementation for 2DSCN, followed by a convergence analysis with special reference to the universal approximation theorem of SCNs. Then, some technical differences between 2DSCN and SCN are presented, highlighting in various aspects, such as the support range for random parameters, the complexity for parameter space and data structure preservation. Interestingly, our work is the first to consider a potential relationship between 2DSCN and CNN in problem-solving, that is, the computations involved in 2DSCN in some sense can be viewed as equivalent to the 'convolution'

M. Li is with the School of Information Technology in Education, South China Normal University, Guangzhou, China, and also with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia. (e-mail: ming.li.ltu@gmail.com).

D. Wang is with the Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia, and is also with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, Liaoning 110819, China. (e-mail: dh.wang@latrobe.edu.au).

* Corresponding author

and 'pooling' tricks performed in the CNN structure. Later, some technical issues around why randomized learner models produced by the 2DSCN algorithm are more prone to have a better generalization ability are investigated in-depth. In particular, some solid results from statistical learning theory are revisited with our special interpretation, for the purpose of a qualitative analysis of learner models' generalization power and useful insights into certain very influential factors. Furthermore, we provide an intuitive sense that 2DSCN may exhibit a similar philosophy as that in DropConnect [21] to effectively alleviate over-fitting. Importantly, to make a reasonable and practical judgement on the generalization ability of an obtained randomized learner model, we focus our efforts on developing a nearly sharp estimation of the model's test error upper bound, thereby one can effectively predict the generalization performance. An extensive experimental study on both regression and classification problems demonstrates the remarkable advantages of 2DSCN on image data modelling, compared to some existing shallow randomized learning techniques as well as some classic deep CNNs. Overall, our main contributions can be summarized as three-fold:

- From the algorithmic perspective, we extend our original SCN framework [12] to a 2D version, and the proposed 2DSCN algorithm can more effectively deal with data modelling tasks with matrix inputs, compared with some existing randomized learning techniques;
- Theoretically, the universal approximation property of 2DSCN-based learner models is verified and some technical differences between 1D and 2D randomized learner models are investigated in terms of various perspectives. Importantly, we provide an upper bound for the test error of a given randomized learner model, and demonstrate in theory how the hidden layer output matrix (computationally associated with the training inputs, the hidden input weights and biases, the number of hidden nodes, etc.) and the output weights can affect the randomized learner model's generalization power.
- For practical applications, the merits of the developed 2DSCN on image data analytics are illustrated on various benchmarks, including the rotation angle prediction for handwritten digits, handwritten digit classification, human face recognition, and CIFAR-10 image classification. Different from deep CNN models that seem dominant in deep learning area, 2DSCN, which belongs to the class of shallow randomized learner models, can provide an alternative method of image data modelling.

The remainder of this paper is organized as follows. Section II overviews some related work, including the 2D random vector functional link (RVFL) networks and our original SCN framework. Section III details the proposed 2D stochastic configuration networks (2DSCNs) with algorithmic descriptions, technical highlights, and some theoretical explanations, aiming to distinguish 2DSCN from the other randomized learning techniques. Section IV presents the experimental study in terms of both image-based regression and classification problems, and Section V concludes this paper with further remarks.

II. Related Work

A. 2DRVFL Networks

2DRVFL networks with matrix inputs were empirically studied in [18]. Technically, it can be viewed as a trivial extension of the original RVFL networks in computation by employing two sets of input weights acting as a matrix transformation over the left and right sides of the inputs. Here we start directly with the problem formulation for 2DRVFL, rather than revisit the basics of RVFL networks. Given N training instances (x_i, t_i) sampled from an unknown function, with inputs $x_i \in \mathbb{R}^{d_1 \times d_2}$, outputs $t_i \in \mathbb{R}^m$, training a 2DRVFL learner model with L hidden nodes is equivalent to solving a linear least squares (LS) problem ($w.r.t$ the output weights), i.e,

$$\min_{\beta_1,\ldots,\beta_j} \sum_{i=1}^{N} \| \sum_{j=1}^{L} \beta_j \phi(u_j^{\mathrm{T}} x_i v_j + b_j) - t_i \|^2,$$

where u_j, v_j, b_j are randomly assigned from $[-\lambda, \lambda]^{d_1}$, $[-\lambda, \lambda]^{d_2}$, $[-\lambda, \lambda]$, respectively and remain fixed. $g(\cdot)$ is the activation function.

The above LS problem can be represented by a matrix form, i.e.,

$$\beta^* = \arg\min_{\beta} \|H\beta - T\|_F^2 \tag{1}$$

where

$$H = \begin{pmatrix} g(u_1^{\mathrm{T}} x_1 v_1 + b_1) & \cdots & g(u_L^{\mathrm{T}} x_1 v_L + b_L) \\ \vdots & \cdots & \vdots \\ g(u_1^{\mathrm{T}} x_N v_1 + b_1) & \cdots & g(u_L^{\mathrm{T}} x_N v_L + b_L) \end{pmatrix}$$

is the hidden layer output matrix, $T = [t_1, t_2, \ldots, t_N]^{\mathrm{T}}$, $\beta = [\beta_1, \beta_2, \ldots, \beta_L]^{\mathrm{T}}$. A closed form solution can be obtained by using the pseudo-inverse method, i.e., $\beta^* = H^{\dagger} T$.

Remark 1. Although RVFL networks (with either vector or matrix inputs) allow fast building a model by randomly assigning input weights and biases, some key technical issues are still unresolved. Theoretically, the approximation error for this kind of randomized learner model is bounded in the statistical sense, which means preferable approximation performance is not guaranteed for every random assignment of the hidden parameters [11]. Furthermore, it has been proved that in the absence of such additional conditions, one may observe exponential growth in the number of terms needed to approximate a non-linear map, and/or the resulting learner model will be extremely sensitive to the parameters [19]. From an algorithmic perspective, all these theoretical predictions do not address the learning algorithm or implementation issues for the randomized learner. The practical usage of this kind of randomized model encounters one key technical difficulty, that is, how to find an appropriate range for randomly assigning hidden parameters with considerable confidence to ensure the universal approximation property. So far, the most accurate (and trivial) way for implementing RVFL networks should employ trial-and-error/rule-of-thumb for parameter setting, that is to say, one needs to perform various setting of λ before obtaining an acceptable learner model. This trick sounds practical but it still has potential drawbacks due to uncertainty

caused by the randomness, as theoretically and empirically studied in [20]. We also note that one can try out different random selection ranges for u_j and v_j in 2DRVFL, such as $u_j \in [-\lambda_1, \lambda_1]^{d_1}$ and $v_j \in [-\lambda_2, \lambda_2]^{d_2}$, but may need more grid-searching in algorithm implementation to find out the 'best' collection $\{\lambda_1^*, \lambda_2^*\}$.

B. SCN framework

Our recent work [12] is the first to touch on the foundation of building a universal approximator with random basis functions. More precisely, a new type of randomized learner model, termed stochastic configuration networks (SCNs), is developed by implanting a 'data-dependent' supervisory mechanism to the random assignment of input weights and biases.

Let $\Gamma := \{g_1, g_2, g_3 ...\}$ represent a set of real-valued functions, and span(Γ) stands for the associated function space spanned by Γ. $L_2(K)$ denotes the space of all Lebesgue measurable functions $f = [f_1, f_2, \ldots, f_m] : \mathbb{R}^d \rightarrow \mathbb{R}^m$ defined on $K \subset \mathbb{R}^d$, with the L_2 norm defined as

$$\|f\| := \left(\sum_{q=1}^{m} \int_D |f_q(x)|^2 dx \right)^{1/2} < \infty. \tag{2}$$

Given another vector-valued function $\phi = [\phi_1, \phi_2, \ldots, \phi_m] : \mathbb{R}^d \rightarrow \mathbb{R}^m$, the inner product of ϕ and f is defined as

$$\langle f, \phi \rangle := \sum_{q=1}^{m} \langle f_q, \phi_q \rangle = \sum_{q=1}^{m} \int_K f_q(x) \phi_q(x) dx. \tag{3}$$

Note that this definition becomes a trivial case when $m = 1$, corresponding to a real-valued function defined on a compact set.

Before revising the universal approximation theory behind SCNs, we recall the problem formulation as follows. For a target function $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$, suppose that we have already built a neural network learner model with only one hidden layer and $L-1$ hidden nodes, i.e, $f_{L-1}(x) = \sum_{j=1}^{L-1} \beta_j g_j(w_j^{\mathrm{T}} x + b_j)$ ($L = 1, 2, \ldots$, $f_0 = 0$), with $\beta_j = [\beta_{j,1}, \ldots, \beta_{j,m}]^{\mathrm{T}}$, and residual error $e_{L-1} = f - f_{L-1} = [e_{L-1,1}, \ldots, e_{L-1,m}]$ far from an acceptable accuracy level, our SCN framework can successfully offer a fast solution to incrementally add β_L, g_L (w_L and b_L) leading to $f_L = f_{L-1} + \beta_L g_L$ until the residual error $e_L = f - f_L$ falls into an expected tolerance ϵ. The following Theorem 1 restates the universal approximation property of SCNs, corresponding to Theorem 7 in [12].

Theorem 1. Suppose that span(Γ) is dense in L_2 space and $\forall g \in \Gamma$, $0 < \|g\| < b_g$ for some $b_g \in \mathbb{R}^+$. Given $0 < r < 1$ and a nonnegative real number sequence $\{\mu_L\}$ with $\lim_{L \rightarrow +\infty} \mu_L = 0$, $\mu_L \leq (1 - r)$, for $L = 1, 2, \ldots$, denoted by

$$\delta_L = \sum_{q=1}^{m} \delta_{L,q}, \delta_{L,q} = (1 - r - \mu_L) \|e_{L-1,q}\|^2, q = 1, 2, \ldots, m, \tag{4}$$

if the random basis function g_L is generated to satisfy the following inequalities:

$$\langle e_{L-1,q}, g_L \rangle^2 \geq b_g^2 \delta_{L,q}, q = 1, 2, \ldots, m, \tag{5}$$

and the output weights are evaluated by

$$[\beta_1, \beta_2, \ldots, \beta_L] = \arg\min_{\beta} \|f - \sum_{j=1}^{L} \beta_j g_j\|, \tag{6}$$

it holds that $\lim_{L \rightarrow +\infty} \|f - f_L\| = 0$, where $f_L = \sum_{j=1}^{L} \beta_j g_j$, $\beta_j = [\beta_{j,1}, \ldots, \beta_{j,m}]^{\mathrm{T}}$.

Remark 2. We would like to highlight that SCNs outperforms several existing randomized learning techniques (e.g. RVFL networks) that employ a totally data-independent randomization in the training process, and demonstrate considerable advantages in building fast learner models with a sound learning and generalization ability. It implies a good potential for dealing with online stream and/or big data analytics. Recently, some extensions of SCNs were proposed from various viewpoints. In [15], we have generalized our SCNs to a deep version, termed as DeepSCNs, with both theoretical analysis and algorithm implementation. It has been empirically illustrated that DeepSCNs can be constructed efficiently (much faster than other deep neural networks) and share many significant features, such as learning representation and a consistency property between learning and generalization. Furthermore, in [13], [16], we built robust SCNs for the purpose of uncertain data modelling. This series of work to some extent exhibits the effectiveness of the SCN framework and showcases an advisable and useful way to study/implant randomness in neural networks.

III. 2D Stochastic Configuration Networks

This section details our proposal for two dimensional stochastic configuration networks (2DSCN). First, based on our original SCN framework, we can straightforwardly present the algorithm description for 2DSCN, followed by theoretically verifying the convergence property. Then, a comparison around some technical points between these two methods is discussed. Afterwards, a theoretical analysis is provided as to why randomized learner models with 2D inputs have a good potential for inducing better generalization.

A. Algorithm Implementation

On the basis of the SCN framework, the problem of building 2DSCN can be formulated as follows. Given a target function $f : \mathbb{R}^{d_1 \times d_2} \rightarrow \mathbb{R}^m$, suppose that a 2DSCN with $L-1$ hidden nodes has already been constructed, that is, $f_{L-1}(x) = \sum_{j=1}^{L-1} \beta_j g_j(u_j^{\mathrm{T}} x v_j + b_j)$ ($L = 1, 2, \ldots$, $f_0 = 0$), where $g(\cdot)$ represents the activation function, $u_j \in \mathbb{R}^{d_1}$, $v_j \in \mathbb{R}^{d_1}$ stand for the collection of input weights (to be stochastically configured with certain constraints), $\beta_j = [\beta_{j,1}, \ldots, \beta_{j,m}]^{\mathrm{T}}$ are the output weights. With the current residual error denoted by $e_{L-1} = f - f_{L-1} = [e_{L-1,1}, \ldots, e_{L-1,m}]$, which as supposed does not reach a pre-defined tolerance level, our objective is to quickly generate a new hidden node g_L (in lieu of u_L, v_L, and b_L) so that the resulting model f_L has an improved residual error after evaluating all the output weights $\beta_1, \beta_2, \ldots, \beta_L$ based on a linear least squares problem.

Suppose we have a training dataset with inputs $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in \mathbb{R}^{d_1 \times d_2}$ and its corresponding outputs $T = \{t_1, t_2, \ldots, t_N\}$, where $t_i = [t_{i,1}, \ldots, t_{i,m}]^{\mathrm{T}} \in$

$\mathbb{R}^m$, $i = 1, \ldots, N$, sampled based on a target function $f : \mathbb{R}^{d_1 \times d_2} \rightarrow \mathbb{R}^m$. Denoted by $e_{L-1} := e_{L-1}(X) = [e_{L-1,1}(X), e_{L-1,2}(X), \ldots, e_{L-1,m}(X)]^{\mathrm{T}} \in \mathbb{R}^{N \times m}$ as the corresponding residual error vector before adding the L-th new hidden node, where $e_{L-1,q} := e_{L-1,q}(X) = [e_{L-1,q}(x_1), \ldots, e_{L-1,q}(x_N)] \in \mathbb{R}^N$ with $q = 1, 2, \ldots, m$. With N two-dimensional inputs $\{x_1, x_2 \ldots, x_N\}$, the L-th hidden node activation can be expressed as

$$h_L := h_L(X) = [g_L(u_L^{\mathrm{T}} x_1 v_L + b_L), \ldots, g_L(u_L^{\mathrm{T}} x_N v_L + b_L)]^{\mathrm{T}}, \tag{7}$$

where $u_L \in \mathbb{R}^{d_1}$ and $v_L \in \mathbb{R}^{d_2}$ are input weights, b_L is the bias.

Denote a set of temporal variables $\xi_{L,q}, q = 1, 2, ..., m$ as follows:

$$\xi_{L,q} = \frac{\left(e_{L-1,q}^{\mathrm{T}} \; h_L\right)^2}{h_L^{\mathrm{T}} h_L} - (1 - r) e_{L-1,q}^{\mathrm{T}} e_{L-1,q}. \tag{8}$$

Based on Theorem 1, it is natural to think about the inequality constrain for building 2DSCN by letting $\sum_q^m \xi_{L,q} \geq 0$.

After successfully adding the L-th hidden node (g_L), the current hidden layer output matrix can be expressed as $H_L = [h_1, h_2, \ldots, h_L]$. Then, the output weights are evaluated by solving a least squares problem, i.e.,

$$\beta^* = \arg\min_{\beta} \|H_L \beta - T\|_F^2 = H_L^{\dagger} T, \tag{9}$$

where $H_L^{\dagger}$ is the Moore-Penrose generalized inverse [22] and $\|\cdot\|_F$ represents the Frobenius norm.

Algorithm 1: 2D Stochastic Configuration Networks (2DSCN)

Input : Training inputs $X = \{x_1, x_2, \ldots, x_N\}$, $x_i \in R^{d_1 \times d_2}$, outputs $T = \{t_1, t_2, \ldots, t_N\}$, $t_i \in R^m$; The maximum number of hidden nodes L_{max}; The expected error tolerance ϵ; The maximum times of random configuration T_{max}; Two sets of scalars $\Upsilon = \{\lambda_1, \ldots, \lambda_{end}\}$ and $\mathcal{R} = \{r_1, \ldots, r_{end}\}$

Output: A 2DSCN model

1 **Initialization**: $e_0 := [t_1^{\mathrm{T}}, t_2^{\mathrm{T}}, \ldots, t_N^{\mathrm{T}}]^{\mathrm{T}}$, $\Omega, W := [\ ]$;
2 **while** $L \leq L_{max}$ *and* $\|e_0\|_F > \epsilon$ **do**
3 **for** $\lambda \in \Upsilon$ **do**
4 **for** $r \in \mathcal{R}$ **do**
5 **for** $k = 1, 2 \ldots, T_{max}$ **do**
6 Randomly assign u_L, v_L, b_L from $[-\lambda, \lambda]^{d_1}$, $[-\lambda, \lambda]^{d_2}$, $[-\lambda, \lambda]$, respectively;
7 Calculate h_L by Eq. (7), and $\xi_{L,q}$ by Eq. (8);
8 **if** $min\{\xi_{L,1}, \xi_{L,2}, ..., \xi_{L,m}\} \geq 0$ **then**
9 **Save** w_L and b_L in W, ξ_L=$\sum_{q=1}^m \xi_{L,q}$ in Ω;
10 **else**
11 go back to **Step 5**;
12 **end**
13 **end**
14 **if** W *is not empty* **then**
15 Find (u_L^*, v_L^*, b_L^*) maximizing ξ_L in Ω, and set the hidden output matrix $H_L = [h_1^*, h_2^*, \ldots, h_L^*]$;
16 **Break** (go to **Step 23**);
17 **else**
18 **Continue**: go to **Step 5**;
19 **end**
20 **end**
21 **end**
22 Calculate $\beta^* = [\beta_1^*, \beta_2^*, \ldots, \beta_L^*]^{\mathrm{T}}$ based on Eq. (9);
23 Calculate $e_L = H_L \beta^* - T$ and renew $e_0 := e_L$, $L := L + 1$;
24 **end**
25 **Return**: $L^*, \beta^*, u^*, v^*, b^*$.

B. Convergence Analysis

The key to verify the convergence of Algorithm 1 is to analyze the universal approximation property of 2DSCN. Recall the proof of Theorem 1 (Theorem 7 in [12]), one can observe that it is the inequality constraints that dominate the whole deduction, rather than the form of input weights (either vector or matrix). In fact, it still holds that $\|e_L\|$ is monotonically decreasing and convergent, $\|e_L\|^2 \leq r\|e_{L-1}\|^2$ for a given $r \in (0, 1)$. Therefore, $\lim_{L \rightarrow +\infty} \|e_L\| = 0$.

We remark that the r value varies during the whole incremental process and the same approach intuitively applies to verify the convergence. Also, it sounds logical to set r as a sequence with monotonically increasing values, because it will become more difficult to meet the inequality condition after a considerable amount of hidden nodes are successfully configured. To some extent, this user-determined (and problem-dependent) parameter affects the algorithm's convergence speed. In particular, one can set r sequence (monotonically increasing) with an initial value quite close to one, which can ease the configuration phase when adding one hidden node as the inequality condition can be easily satisfied. Alternatively, a user can start with a relatively small value (but not too small), which however requires more configuration trials at one single step to find suitable input weights and biases that fit the inequality condition. This can lead to a huge computational burden or even more unnecessary fails during the configuration phase. Since the convergence property is guaranteed theoretically, one can think about some practical guidelines for setting the r sequence with reference to their practical task. Based on our experience, the first trick, i.e., initializing r with a value close to one and then monotonically increasing (progressively approaching one), is more feasible in algorithm implementation. Additionally, Lines 14-19 use the same trick performed in SCN (Remark 7, in particular) [12] to make the structure of 2DSCN more compact. In practice, larger $\xi_{L,q}$ can contribute to a faster decrease of the residual error, i.e., u_L^*, v_L^*, b_L^* maximizing ξ_L in Ω can result in the best candidate to form the hidden layer matrix.

C. Comparison with SCNs

1) Support Range for Random Parameters: Technically, 2DSCN still inherits the essence of our original SCN framework, that is, stochastically configuring basis functions in light of a supervisory mechanism (see Theorem 1). This kind of data-dependent randomization can effectively and efficiently locate the 'support range', where one can randomly generate hidden nodes with insurance for building universal approximators. Despite this common character, the differences between support ranges induced by these two methods should be highlighted. Computationally, it holds that

$$\begin{aligned} u^{\mathrm{T}} x v &= \mathrm{Tr}(u^{\mathrm{T}} x v) = \mathrm{Tr}(x v u^{\mathrm{T}}) \\ &= \mathrm{Tr}((uv)^{\mathrm{T}} x) = (vec(uv^{\mathrm{T}}))^{\mathrm{T}} vec(x), \end{aligned}$$

where Tr denotes the matrix trace, $u \in \mathbb{R}^{d_1}$, $v \in \mathbb{R}^{d_2}$, $vec(\cdot) \in \mathbb{R}^{d_1 d_2}$ stands for vectorization of a given 2D array.

We observe that although $(vec(uv^{\mathrm{T}}))^{\mathrm{T}}vec(x)$ can be viewed as a regular dot product (between the hidden weight vector and input) computation performed in SCN, the resulting $(d_1 d_2)$-dimensional vector $vec(uv^{\mathrm{T}})$ may exhibit a different distribution, in contrast to a random $(d_1 d_2)$-dimensional vector from SCN-induced support range.

We should also note that there is no special requirements for the initial distribution of u and v performed in the algorithm implementation. For instance, one can set two different range parameter sets $\Upsilon_u = \{\lambda_1^u, \ldots, \lambda_{end}^u\}$ and $\Upsilon_v = \{\lambda_1^v, \ldots, \lambda_{end}^v\}$ respectively in their experimental setup. If so, in algorithm design, one more loop is needed for searching the appropriate λ^v from Υ_v when λ^u is chosen and fixed, or vice versa. Since the universal approximation capability is always guaranteed, this complex manipulation sounds not computationally efficient in practical implementation . For simplicity, we use the same random range setting for u and v, i.e., merely Υ, as noted in step 3 of the above Algorithm 1.

In practice, u or v, which can be viewed as row/column-direction hidden weight, has its own support range, which relies on their initially employed distribution (Υ_u or Υ_v) and the inequality constraint for hidden node configuration.

2) Parameter Space: Despite the fact that neural networks can universally approximate complex multivariate functions, they still suffer from difficulties on high-dimensional problems where the number of input features is much larger than the number of observations. Avoiding high-dimensional inputs and seeking useful input features to alleviate overfitting are important and essential for the majority of machine learning techniques. It is clear that one 2DSCN model with L hidden nodes has L d_1-dimensional weights and L d_2-dimensional input weights, L biases (scalar), L m-dimensional output weights, that is, $L \times (d_1 + d_2 + 2)$ parameters in total; whist the SCN model with the same structure has L $(d_1 \times d_2)$-dimensional input weights and the same amount of biases and output weights, i.e., $L \times (d_1 d_2 + 1 + m)$ parameters altogether. Technically, in SCN, it can impose a high dimensional parameter space that may cause potential difficulties in meeting the stochastic configuration inequality (5), especially when the number of training samples is far lower than the dimensionality of the input weights. Furthermore, for a relatively large L, a huge memory is needed for saving $L \times (d_1 d_2 + 1 + m)$ parameters in computation. On the contrary, 2DSCN can effectively ease the high-dimensional issue and to some extent economize physical memory in practice.

3) Data Structure Preservation: It sounds logical that 2D-SCN has several advantages in preserving the spatial details of the given input images as it cares about the 2D-neighborhood information (the order in which pixels appear) of the input rather than the simple vectorization operation performed in SCN. This argument has been raised and is commonly accepted in literature, however, there is no sufficient scientific evidence verifying why and how the vectorization trick affects the structural information of the 2D inputs. In this part, we examine the resemblance between 2DSCN methodology and convolutional neural networks (CNNs) in terms of computational perspective. A schematic diagram is plotted in Fig. 1 and corresponding explanations are as follows.

Recall Eq. (10), the left low-dimensional vector u^{T} is acting as a 'filter' to extract some random features from the 2D input x. In other words, each column of x is now considered as a block, i.e., image $x = [x_1, x_2, \ldots, x_{d_2}]$ is supposed to be represented by d_2 block-pixels, then $u^{\mathrm{T}}x$ can be viewed as a 'convolution' operation between the 'filter' u^{T} (of size $1 \times d_1$) and the input x along the vertical direction, leading to a feature map $[u^{\mathrm{T}}x_1, u^{\mathrm{T}}x_2, \ldots, u^{\mathrm{T}}x_{d_2}]$. Then, a 'pooling' operation, conducted by calculating a weighted sum of the obtained feature map, is used to aggregate feature information. As a conjecture, 2DSCN might have some technical merits in common with CNNs for image data analytics. More theoretical and/or empirical research on this judgment is left for our future study.

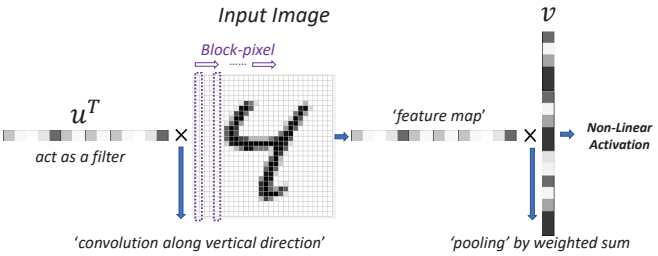


Fig. 1. Schematic diagram of computational equivalence between 2DSCN and CNN.

D. Superiority in Generalization

In this section, we investigate in-depth why 2DSCN (and 2DRVFL) potentially leads to a better generalization performance than SCN (and RVFL). Four supportive theories (ST1 to ST4) are presented to explain our intuitive prediction, that is, the stochastically configured input weights and biases of 2DSCN to a great extent are more prone to result in lower generalization error.

ST1: Learning Less-Overlapping Representations. Typically, elements of a weight vector have one-to-one correspondence with observed features and a weight vector is oftentimes interpreted by examining the top observed features that correspond to the largest weights in this vector. In [23], the authors proposed a non-overlapness promoting regularization learning framework to improve interpretability and help alleviate overfitting. It imposes a structural constraint over the weight vectors, thus can effectively shrink the complexity of the function class induced by the neural network models and improve the generalization performance on unseen data. Assuming that a model is parameterized by L vectors $\mathcal{W} = \{\bar{w}_i\}_{i=1}^L$, [23] proposed a hybrid regularizer consisting of an orthogonality-promoting term and sparsity-promoting term, denoted by

$$\Omega(\mathcal{W}) = \mathrm{tr}(M) - \log\det(M) + \gamma \sum_{i=1}^{L} \|\bar{w}_i\|_{l_1},$$

where M is the Gram matrix associated with $\mathcal{W}$, i.e., $M_{i,j} = \bar{w}_i^{\mathrm{T}}\bar{w}_j$, γ is a tradeoff parameter between these two regularizers.

The first term $\mathrm{tr}(M) - \log\det(M)$ controls the level of near-orthogonality over the weight vectors from $\mathcal{W}$, while the second term $\sum_{i=1}^{L} \|\bar{w}_i\|_{l_1}$ encourages $w_i \in \mathcal{W}$ to have more elements close to zero. It is empirically verified that this hybrid form of regularizer can contribute to learner models with a better generalization performance [23].

Hence, in light of our research, we can roughly explain why 2D models (2DSCN and 2DRVFL) can outperform 1D models (SCN and RVFL), and simultaneously, why SCN-based models are better than RVFL-based ones: (i) generally, there is no big difference between SCN and 2DSCN on the near-orthogonal level of $\mathcal{W}$, however, random weights in 2DSCN can have a higher level of sparsity than that in SCN, hence leading to a smaller $\sum_{i=1}^{L} \|\bar{w}_i\|_{l_1}$. This deduction can also be used to distinguish 2DRVFL from RVFL as well; (ii) given a similar level of sparsity in $\mathcal{W}$, SCN-based models are more prone to having a lower near-orthogonal level than RVFL-based ones, therefore, indicating a smaller $\mathrm{tr}(M) - \log\det(M)$. For further justifications of these intuitive arguments, we present analogous theories regarding the near-orthogonality of weight vectors in the following section (ST2) and provide some statistical results at the end.

ST2: Weight Vector Angular Constraints to Promote Diversity. The authors of [24], [25] showed the empirical effectiveness and explained in theory when and why a low generalization error can be achieved by adjusting the diversity of hidden nodes in neural networks. Theoretically, increasing the diversity of hidden nodes in a neural network model can reduce the estimation error but increase the approximation error, which implies that a low generalization error can be achieved when the diversity level is set appropriately. Specifically, the near-orthogonality of the weight vectors (e.g, input weights and biases) can be used to characterize the diversity of hidden nodes in a neural network model, and a regularizer with weight vector angular constraints can be used to alleviate overfitting. To highlight the impact of near-orthogonality (of the weight vectors) on the generalization error, we will reformulate the two main theoretical results addressed in [25]. Prior to this, some notations and preliminaries on statistical learning theory are revisited.

Consider the hypothesis set

$$\mathcal{F} := \{x \mapsto \sum_{j=1}^{L} \beta_j g(\bar{w}_j^{\mathrm{T}} x) \;\big|\; \|\beta\|_2 \le B, \|\bar{w}_j\|_2 \le C, \forall i \neq j, |\bar{w}_i^{\mathrm{T}} \bar{w}_j| \le \tau \|\bar{w}_i\|_2 \|\bar{w}_j\|_2\}.$$

where β stands for the output weight and $g(t) = 1/(1+e^{-t})$ is the sigmoid activation function. Given the training samples $\{(x_i, y_i)\}_{i=1}^{N}$ generated independently from an unknown distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$, the generalization error of $f \in \mathcal{F}$ is defined as $R(f) = \mathbb{E}_{\mathcal{P}_{\mathcal{X}\mathcal{Y}}}[\frac{1}{2}(f(x) - y)^2]$. As $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$ is not available, one can only consider minimizing the empirical risk $\hat{R}(f) = \frac{1}{2N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$ in lieu of $R(f)$. Let $f^* \in \arg\min_{f\in\mathcal{F}} R(f)$ be the true risk minimizer and $\hat{f} \in \arg\min_{f\in\mathcal{F}} \hat{R}(f)$ be the empirical risk minimizer. Then, the generalization error $R(\hat{f}) := R(\hat{f}) - R(f^*) + R(f^*)$ (of the empirical risk minimizer $\hat{f}$) can be estimated by bounding the estimation error $R(\hat{f}) - R(f^*)$ and the approximation error $R(f^*)$, respectively. The following Theorem 2 and Theorem 3 show these two estimations in relation to the factor τ.

Theorem 2 [25] (Estimation Error). With a probability of at least $1 - \delta$, the estimation upper bound of estimation error decreases as τ becomes smaller, i.e.,

$$R(\hat{f}) - R(f^*) \le \frac{\gamma^2\sqrt{2\ln(4/\delta)} + \gamma B(2C + 4|g(0)|)\sqrt{m}}{\sqrt{N}},$$

where $\gamma = 1 + BC\sqrt{(m-1)\tau + 1}/4 + \sqrt{m}B|g(0)|$.

Suppose the target function $G = \mathbb{E}[y|x]$ satisfies a certain smoothness condition given by $\int \|\omega\|_2 |\tilde{G}(\omega)| d\omega \le B/2$, where $\tilde{G}(\omega)$ represents the Fourier transformation of G. Then, the approximation error, which reflects the power of the hypothesis set $\mathcal{F}$ for approximating G, is expressed as follows.

Theorem 3 [25] (Approximation Error). A smaller τ contributes to a larger upper bound of the approximation error, that is, let $C > 1$ and $L \le 2(\lfloor \frac{\pi/2-\theta}{\theta} \rfloor + 1)$, where $\theta = \arccos(\pi)$, then there exists $f \in \mathcal{F}$ such that

$$\|f - G\|_2 \le B(\frac{1}{\sqrt{m}} + \frac{1 + 2\ln C}{C}) + 2\sqrt{L}BC\sin(\frac{\min(2L\theta, \pi)}{2}).$$

Based on Theorem 2 and Theorem 3, we can come to a conclusion, that is, a larger upper bound of the generalization error can be caused by the case when the weight vectors are highly near-orthogonal with each other (τ is extremely small) or in a situation when τ is close or equal to 1 (e.g., there exist two weight vectors that are linearly dependent). Therefore, given the two obtained (randomized) learner models with roughly the same training performance, the one equipped with hidden weight vectors of high near-orthogonality is likely to result in worse generalization. On the other hand, our previous work [20] reveals a key pitfall of RVFL networks, that is, all high-dimensional data-independent random features are nearly orthogonal to each other with a probability of one. Fortunately, the supervisory mechanism used in the SCN framework imposes an implicit relationship between each weight vector and can effectively reduce the probability of near-orthogonality. With all these clues, we can roughly explain why the learner models produced by SCN and 2DSCN are more prone to result in a better generalization performance than RVFL and 2DRVFL.

ST3: Vague Relationship between 2DSCN and DropConnect framework. To effectively alleviate over-fitting and improve the generalization performance, Dropout has been proposed for regularizing fully connected layers within neural networks by randomly setting a subset of activations to zero during training [26], [27]. DropConnect proposed by Wan et al. [21] is an extension of Dropout in which each connection, instead of each output unit, can be dropped with a certain probability. Technically, DropConnect can be viewed as similar to Dropout because they both perform dynamic sparsity within the learner model during the training phase, however, they differ in that the sparsity-based concerns are imposed on the hidden input weights, rather than on the output vectors of a layer. This means the fully connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen at random during the training stage. Importantly, as noted in [21], the mechanism employed in DropConnect

is not equivalent to randomly assigning a sparse hidden input weights matrix (and remain fixed) during the training process, which indirectly invalidates the effectiveness of the RVFL and 2DRVFL methods even when they use sparse weights in the hidden layer.

Intuitively, our proposed 2DSCN could be thought of as being related to DropConnect, in terms of the following points:

- the supervisory mechanism used in 2DSCN aims at incrementally configuring the weight vectors until convergence to a universal approximator, which is equivalent to the training objective of DropConnect;
- once the random weight vectors in 2DSCN have many small elements close to zero, their functionality is similar to the sparsity mechanism imposed in DropConnect on the hidden weights;
- on the basis of the above two clues, the incremental process performed in 2DSCN can be viewed as being similar to the proceeding dynamic sparsity within the learner model during the training phase as used in DropConnect.

We would like to highlight that the original SCN does not have this kind of vague relationship with DropConnect, unless certain weights sparsity regularizer is concerned in the training process. In contrast, 2DSCN involves more weight vectors with small values, which indeed can be viewed as a considerable degree of sparsity and has a good potential to inherit some of the merits of DropConnect and its parallel methodology.

ST4: Novel Estimation of Test Error. Various statistical convergence rates for neural networks have been established when some constraints on the weights are concerned [28], [29]. Empirically, small weights together with a small training error can lead to significant improvements in generalization. All these investigations lend scientific support to heuristic techniques like weight decay and early stopping. The reason behind this is that producing over-fitted mappings requires high curvature and hence large weights, while keeping the weights small during training can contribute to smooth mappings. Technically, the regularization learning framework, introducing various types of weight penalties such as L_2 weight decay [30], [31], Lasso [32], Kullback-Leibler (KL)-divergence penalty [33], etc., shares a similar philosophy to help prevent overfitting.

A comprehensive overview of existing theories/techniques concerning learner models's generalization capability is out of the focus of this paper. Instead, we revisit the theoretical results presented in [34], and illustrate mathematically how the output weights' magnitudes affect the randomized learner models' generalization power. For a better understanding and consistent problem formulation, we restate their main result with reference to our previous notations used in ***ST2***, that is,

Theorem 4 [34]. Consider the hypothesis set $\mathcal{F}_p := \{f(x) = \int \alpha(w)g(w;w)dw \big| |\alpha(w)| \leq Bp(w)\}$ with certain distribution p and function g satisfying $\sup_{x,w} |g(x;w)| \leq 1$, and given a training data set with N input-output pairs drawn iid from some distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$, a randomized learner model $\hat{f}(x) = \sum_{j=1}^{L} g(x_i;w_i)$ can be obtained by randomly assigning $w_i, w_2, \ldots, w_L$ from the distribution p and solving the empirical risk minimization problem [1] $\min_\beta \frac{1}{N}\sum_{i=1}^{N}(\hat{f}(x_i) - y_i)^2$ subject to $\|\beta\|_\infty \leq B/L$. Then, with a probability of at least $1-2\delta$, the upper bound for the generalization error of $\hat{f}$ can be estimated by

$$R[\hat{f}] \leq \min_{f\in\mathcal{F}_p} R[f] + \mathcal{O}\left(\left(\frac{1}{\sqrt{N}} + \frac{1}{\sqrt{L}}\right)2B\sqrt{\log\frac{1}{\delta}}\right)$$

Theoretically, the upper bound in Theorem 3 implies that randomized learner models with good training results and small output weights can probably lead to preferable generalization performance, in terms of the probability perspective. However, this cannot be used directly to bound the practical test error to evaluate the randomized learner models's generalization performance. More numerical estimation for the test error (resulting from algorithm realization in practice) is required to better characterize the generalization capability as well as the associated impacting factors.

As one of our main contributions in this work, a novel upper bound estimation for the test error is presented in terms of computational perspective. To facilitate our theoretical investigation, we view the hidden layer matrix H as a matrix-valued function of matrix variable, i.e., $H : \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times L}$, denoted by (see [35] for basic fundamentals on matrix calculus)

$$H := H(X) = \begin{pmatrix} g(w_1^{\mathrm{T}}x_1 + b_1) & \cdots & g(w_L^{\mathrm{T}}x_1 + b_L) \\ \vdots & \ldots & \vdots \\ g(w_1^{\mathrm{T}}x_N + b_1) & \cdots & g(w_L^{\mathrm{T}}x_N + b_L) \end{pmatrix} \tag{10}$$

with the argument X represented by

$$X = (x_1, x_2, \ldots, x_N)^{\mathrm{T}} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ldots & \vdots \\ x_{N,1} & \cdots & x_{N,d} \end{pmatrix} \tag{11}$$

Suppose that H is differentiable and has continuous first-order gradient ∇H, defined by a quartix which belongs to $\mathbb{R}^{N\times L\times N\times d}$, i.e.,

$$\nabla H(X) = \begin{pmatrix} \nabla H_{1,1}(X) & \cdots & \nabla H_{1,L}(X) \\ \vdots & \ldots & \vdots \\ \nabla H_{N,1}(X) & \cdots & \nabla H_{N,L}(X) \end{pmatrix},$$

where for $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, L$

$$\nabla H_{i,j}(X) = \begin{pmatrix} \frac{\partial g(w_j^{\mathrm{T}}x_i + b_j)}{\partial x_{1,1}} & \cdots & \frac{\partial g(w_j^{\mathrm{T}}x_i + b_j)}{\partial x_{1,d}} \\ \vdots & \ldots & \vdots \\ \frac{\partial g(w_j^{\mathrm{T}}x_i + b_j)}{\partial x_{N,1}} & \cdots & \frac{\partial g(w_j^{\mathrm{T}}x_i + b_j)}{\partial x_{N,d}} \end{pmatrix}.$$

Then, the first directional derivative in a given direction $Z \in \mathbb{R}^{N\times d}$ can be represented by

$$\overset{\to Z}{d}H(X) := \begin{pmatrix} \mathrm{tr}\Big(\nabla H_{1,1}(X)^{\mathrm{T}}Z\Big) & \cdots & \mathrm{tr}\Big(\nabla H_{1,L}(X)^{\mathrm{T}}Z\Big) \\ \vdots & \ldots & \vdots \\ \mathrm{tr}\Big(\nabla H_{N,1}(X)^{\mathrm{T}}Z\Big) & \cdots & \mathrm{tr}\Big(\nabla H_{N,L}(X)^{\mathrm{T}}Z\Big) \end{pmatrix}$$

[1] *In [34], a general form of cost function is concerned. Here we specify a quadratic loss function and its associated Lipschitz constant has no impacts on the final estimation.

It is logical to think that the test sample matrix $\tilde{X}$ can be represented by imposing sufficiently small random noises into the training sample matrix X, i.e., $\tilde{X} := X + \epsilon Z$, where $Z \in \mathbb{R}^{N\times L}$ is a random matrix and ϵ is sufficiently small. Then, we can take the first-order Taylor series expansion about X [35], i.e.,

$$H(\tilde{X}) := H(X + \epsilon Z) = H(X) + \epsilon \overrightarrow{d}^{Z} H(X) + o(\epsilon^2)$$

Therefore, the test error can be estimated by

$$\begin{aligned} & \|H(\tilde{X})\beta - Y\|_F \\ = \; & \|(H(X) + \epsilon \overrightarrow{d}^{Z} H(X) + o(\epsilon^2))\beta - Y\|_F \\ \leq \; & \|H(X)\beta - Y\|_F + \epsilon \|\overrightarrow{d}^{Z} H(X)\|_F \|\beta\|_F \\ & + o(\epsilon^2)\|\beta\|_F \end{aligned} \tag{12}$$

where $\|\cdot\|_F$ stands for the Frobenius norm, $\|H(X)\beta - Y\|_F$ represents the training error.

Basically, this rough estimation implies two points that should be highlighted:

(i) The upper bound for the test error can be viewed as an increasing function of $\|\beta\|_F$, which means that learner models with smaller output weight values are more prone to generalize preferably on unseen data. This is consistent with the philosophy behind the regularization learning framework, that is, imposing a penalty term to control the output weights magnitudes during the training process.

(ii) We can further investigate how the input weights and biases affect the value of $\|\overrightarrow{d}^{Z} H(X)\|_F$. In particular, we use the sigmoid function in the following deduction, i.e., $g(t) = 1/(1+e^{-t})$ and $g'(t) = g(t)(1-g(t))$. Mathematically, the (i,j)-th element ($i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, L$) inside $\overrightarrow{d}^{Z} H(X)$ can be expressed as

$$\begin{aligned} & \mathrm{tr}\Big(\nabla H_{i,j}(X)^{\mathrm{T}} Z\Big) \\ := \; & \sum_{i'=1}^{N}\sum_{k'=1}^{d} \frac{\partial g(w_j^{\mathrm{T}} x_i + b_j)}{\partial x_{i',k'}} Z_{i',k'} \\ = \; & -g(w_j^{\mathrm{T}} x_i + b_j)(1 - g(w_j^{\mathrm{T}} x_i + b_j)) \sum_{k=1}^{d} w_{j,k} Z_{i,k}. \end{aligned}$$

Then, a rough upper bound for $\|\overrightarrow{d}^{Z} H(X)\|_F$ can be obtained, that is,

$$\begin{aligned} & \|\overrightarrow{d}^{Z} H(X)\|_F \\ \leq \; & \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{L}(g_{ij}(1-g_{ij}))^2 (\sum_{k=1}^{d} w_{j,k}^2)(\sum_{k=1}^{d} Z_{i,k}^2)} \\ \leq \; & \max_{1\leq i\leq N} \|Z_i\|_2 \cdot \|H \circ (O - H) \circ \ddot{W}\|_F, \end{aligned}$$

where we use abbreviation g_{ij} for $g(w_j^{\mathrm{T}} x_i + b_j)$), and Cauchy-Schwarz inequality in the first inequality. Z_i stands for the i-th row vector of the matrix Z. H is defined in (10), $O \in \mathbb{R}^{N\times L}$ is a matrix of ones (every element is equal to one), and $\ddot{W} \in \mathbb{R}^{N\times L}$ is formulated by copying the row vector $(\|w_1\|_2, \|w_2\|_2, \ldots, \|w_L\|_2)$ N-times, '$\circ$' stands for the Hadamard (entrywise) product among the matrixes.

So far, we can summarize the above theoretical result in the following Theorem 5. Readers can refer to some of the aforementioned notations in the context.

Theorem 5. Given training input $X \in \mathbb{R}^{N\times d}$ and output $Y \in \mathbb{R}^{N\times m}$, suppose a randomized neural network model with L hidden nodes is build, corresponding to the hidden layer output matrix (on the training data) $H \in \mathbb{R}^{N\times L}$, the output weight matrix β, and the training error $\|H\beta - Y\|_F$. Let $\tilde{X} := X + \epsilon Z$ be the test (unseen) input data matrix, where $Z \in \mathbb{R}^{N\times L}$ is a random matrix, ϵ is sufficiently small, $\tilde{H}$ stand for the associated hidden layer output matrix, then, the test error can be bounded by

$$\begin{aligned} & \|\tilde{H}\beta - Y\|_F \\ \leq \; & \|H\beta - Y\|_F + \epsilon \max_{1\leq i\leq N} \|Z_i\|_2 \cdot \|H \circ (O - H) \circ \ddot{W}\|_F \|\beta\|_F \\ & + o(\epsilon^2)\|\beta\|_F. \end{aligned} \tag{13}$$

Remark 3. We would like to highlight a trick concerned in the previous deduction for Theorem 5. We preserve the bundle of computational units '$(g_{ij}(1-g_{ij}))^2\|w_j\|_2^2$' rather than to roughly estimating the whole term by '$(\frac{1}{4})^2\|w_j\|_2^2$', which consequently can result in a very blunt bound $\frac{1}{4}\|W\|_F\|Z\|_F$ for $\|\overrightarrow{d}^{Z} H(X)\|_F$. Unfortunately, upper bound $\frac{1}{4}\|W\|_F\|Z\|_F$ sounds meaningless because it does not consider the saturation property of the sigmoid function, and may cause the misleading thought that 'larger input weights can destroy the generalization capability'. In contrast, our proposed upper bound (13) is nearly sharp and can provide valuable information to identify the role of input weights (and biases) and training samples on the learner models's generalization power. It is a bundle of computational units $\|H \circ (O - H) \circ \ddot{W}\|_F$ rather than merely the $\|W\|_F$ that acts as a suitable indicator for predicting the generalization performance. Furthermore, it should be noted that, input weights (and biases) with small values but enforcing the $g(\cdot) \approx 1$ or $g(\cdot) \approx 0$ (corresponding to the saturation range of the sigmoid function), are more likely to result in a small value of $\|H \circ (O - H) \circ \ddot{W}\|_F$ and consequently bring a small generalization error bound.

On the other hand, the right side of Eq. (13) has a strong resemblance to the regularized learning target by viewing $\epsilon \max_{1\leq i\leq N}\|Z_i\|_2 \cdot \|H \circ (O-H) \circ \ddot{W}\|_F$ as the regularization factor $\sigma > 0$, that is, $\|H\beta - Y\|_F + \sigma\|\beta\|_F$, considered as a whole to effectively alleviate over-fitting.

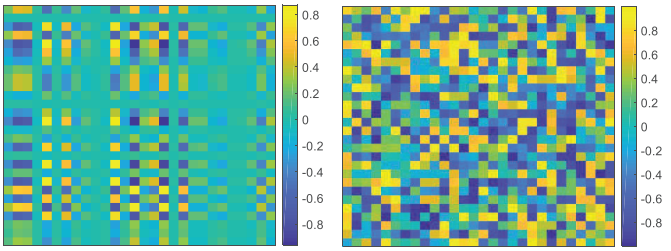
Why 2D randomized models are equipped with more small weights? Since small weights to some extent can probably have a certain positive influence on enhancing a learner model's generalization ability, one major issue still left unclarified is whether or not 2D randomized learner models possess this advantage. For this purpose and before ending this section, we provide a statistical verification on the frequency when sufficiently small weights occur in 1D and 2D randomized models, aiming to further support the superiority of 2D randomized models.

Given the distribution $\mathcal{P}$ (either uniform or gaussian), we investigate the statistical differences among the following three

strategies for randomly assigning parameters:

- M1: Randomly assign $w = [w_1, w_2, \ldots, w_d]^{\mathrm{T}}$ from $\mathcal{P}$;
- M2: Randomly assign $z_1 = [z_{1,1}, z_{1,2}, \ldots, z_{1,d}]^{\mathrm{T}}$, $z_2 = [z_{2,1}, z_{2,2}, \ldots, z_{2,d}]^{\mathrm{T}}$ from $\mathcal{P}$, then calculate their Hadamard (entrywise) product $w^{(1D-P)} = z_1 \circ z_2$;
- M3: Randomly assign $u = [u_1, u_2, \ldots, u_{d_1}]^{\mathrm{T}}$, $v = [v_1, v_2, \ldots, v_{d_2}]^{\mathrm{T}}$, with $d_1 d_2 = d$, then calculate uv^{T} and let $w^{2D} := vec(uv^{\mathrm{T}})$.

A simple and vivid demonstration for the distribution of the random weights induced by M1 and M3 is provided in Fig. 2, in which it can be clearly seen that w^{2D} have more small values (near zero) than w. Based on our empirical experience, similar plotting (display) between M2 and M3 looks visually indistinguishable. More statistical results are helpful for making a reasonable distinction among M1∼M3.



(a) w_i^{2D} (b) w_i

Fig. 2. Simple illustration for the distribution of 1D and 2D random weights: (a) Values of w_i^{2D} generated by M3 with $d_1 = d_2 = 28$; (b) Values of w_i generated by M1 with $d = 784$. Both w_i^{2D} and w_i are reshaped into 28×28 for visualization.

Our primary objective is to find how frequently these strategies contribute to a high dimensional random vector (w, $w^{(1D-P)}$, or w^{2D}) with a considerable number of elements whose values are close to zero. Here we present the theoretical results to answer this question and then provide an empirical verification using statistics. Specifically, from a probability perspective, we conclude that **M3** are more prone to get a random vector with more elements close to 0, as mathematically expressed by

$$\begin{aligned} & P\left(\frac{\#\left\{i \middle| |w_i^{2D}| \le \epsilon\right\}}{d} \ge p\right) \\ \ge\ & P\left(\frac{\#\left\{i \middle| |w_i^{(1D-P)}| \le \epsilon\right\}}{d} \ge p\right) \\ \ge\ & P\left(\frac{\#\left\{i \middle| |w_i| \le \epsilon\right\}}{d} \ge p\right), \end{aligned} \tag{14}$$

where ϵ is a small value close to 0, as a reference factor for locating small elements of the random vector. $\#\{\cdot\}$ stands for the cardinality, which of course is equal to the number of elements we are interested in counting, i.e, whose absolute values are equal to/lower than ϵ. On the other hand, p is a threshold by which we can study the normalized percentage ($\#\{\cdot\}/d \in [0, 1]$) when elements of interest occur in that random vector (w, $w^{(1D-P)}$, or w^{2D}).

Instead of providing a rigorous mathematical proof, here we focus on an empirical study to verify the inequalities (14) using statistics. In particular, we set $d_1 = d_2 = 28$, $d = 784$, $p = 8\%, 10\%, 12\%, 15\%$, $\epsilon = 0.01, 0.03, 0.05, 0.1$, respectively, and study two options for $\mathcal{P}$, i.e., uniform (Case 1) and Gaussian distribution (Case 2), then run 100,000 independent numerical simulations to approximate those three probability values compared in (14), in terms of each set of (p, ϵ) for both distribution cases. In the following, we give some theoretical description and the statistical results for Case 1 and 2.

Case 1. Given two independent uniform random variables $z_1 \sim U[-1, 1]$ and $z_1 \sim U[-1, 1]$, the probability density function (p.d.f) of their product $z = z_1 z_2$ can be expressed by

$$p(x) = \begin{cases} \frac{-1}{2} \ln x, & 0 < z \le 1 \\ \frac{-1}{2} \ln(-x), & -1 \le z < 0 \end{cases}$$

In the simulation, we conduct 100,000 independent trials for randomly assigning w, $w^{(1D-P)}$, and w^{2D} according to M1 M3, respectively. Then, we can count the number of times (denoted by M) when the condition $\#\{\cdot\}/d \ge p$ is satisfied, followed by roughly estimating the true probability in (14) with $\tilde{P} = M/100000$. In Table I, we list the corresponding $\tilde{P}$ values (arranged in order $w^{2D}/w^{(1D-P)}/w$) for the cases with different settings of ϵ (e.g., $0.001, 0.005, 0.01$) and p (e.g., $8\%, 10\%, 12\%, 15\%$), demonstrating that 2D models have more opportunities to have small input weights during the training process.

TABLE I
STATISTICAL VERIFICATION FOR CASE 1 UNIFORM DISTRIBUTION

Case 1	$\epsilon = 0.001$	$\epsilon = 0.005$	$\epsilon = 0.01$
$p = 8\%$	0.0047 / 0 / 0	0.1338 / 2.0e-5 / 0	0.4022 / 0.2838 / 0
$p = 10\%$	1.1e-4 / 0 / 0	0.0160 / 0 / 0	0.1131 / 0 / 0
$p = 12\%$	3.0e-5 / 0 / 0	0.0045 / 0 / 0	0.0497 / 0 / 0
$p = 15\%$	0 / 0 / 0	6.7e-4 / 0 / 0	0.0126 / 0 / 0

Case 2. Given two independent normal random variables $z_1 \sim N(0, \sigma_1^2)$ and $z_2 \sim N(0, \sigma_2^2)$, the probability density function (p.d.f) of their product $z = z_1 z_2$ can be expressed by

$$p(x) := \frac{1}{\pi \sigma_1 \sigma_2} K_0\left(\frac{|x|}{\sigma_1 \sigma_2}\right), x \in \mathbb{R},$$

where $K_0(\cdot)$ is a modified Bessel function of the second kind of order zero [36], as given by

$$K_0(x) = \int_0^\infty e^{-x \cosh(t)} dt$$

Similarly, we present the associated $\tilde{P}$ values for $w^{2D}/w^{(1D-P)}/w$ respectively in Table II, in which the records also show that 2D randomized models are more prone to be equipped with small weights.

IV. PERFORMANCE EVALUATION

In this section, we demonstrate the advantages of 2DSCNs in image data modelling tasks, compared with some baseline/randomized learning methods namely SCN, RVFL and

TABLE II
STATISTICAL VERIFICATION FOR CASE 2 GAUSSIAN DISTRIBUTION

Case 2	$\epsilon = 0.001$	$\epsilon = 0.005$	$\epsilon = 0.01$
$p = 8\%$	0.0011 / 0 / 0	0.0445 / 0 / 0	0.1781 / 4.6e-4 / 0
$p = 10\%$	1.0e-5 / 0 / 0	0.0025 / 0 / 0	0.0248 / 0 / 0
$p = 12\%$	0 / 0 / 0	3.8e-4 / 0 / 0	0.0071 / 0 / 0
$p = 15\%$	0 / 0 / 0	2.0e-5 / 0 / 0	9.2e-4 / 0 / 0

2DRVFL networks, as well as some classic (pretrained) deep CNN models.

A. Regression: Rotation Angle Predication for Handwritten Digits

We first demonstrate the merits of the proposed 2DSCN by predicting the angles of rotation of handwritten digits. In particular, the Neural Network Toolbox in MATLAB R2017b provides a collection of synthetic handwritten digits, which contains 5000 training and 5000 test images of digits with corresponding angles of rotation. Each image represents a rotated digit in grayscale and of normalized size (28×28). Baseline approaches such as RVFL and SCN deploy one-dimensional input (reshaping the image into a vector) in modelling, while 2DRVFL and 2DSCN can directly deal with image-based inputs in problem-solving. It should be noted that, w in RVFL, u (and v) in 2DRVFL are randomly assigned from $[-1, 1]^{784}$ and $[-1, 1]^{28}$, respectively, while the biases for them are randomly assigned from $[-1, 1]$. For SCN and 2DSCN, we take $T_{max} = 5$, $\lambda = \{1, 5, 15, 30, 50, 100, 150, 200, 250\}$, $r = \{1 - 10^{-j}\}_{j=2}^{7}$ in the algorithm implementation (see Section 3 for the functionality of these parameters).

Two types of evaluation metrics are used in the performance comparison: (i) the percentage of predictions within an acceptable error margin (PPA); and (ii) the root-mean-square error (RMSE) of the predicted and actual angles of rotation. In particular, a user-defined threshold θ (in degrees) is needed to measure the PPA values, that is, calculating the error between the predicted and actual angles of rotation and then counting the number of predictions within an acceptable error margin θ from the true angles. Mathematically, the PPA value within threshold θ can be obtained by

$$PPA = \frac{\#\{|Prediction\ \ Error| < \theta\}}{Number\ \ of\ \ Sample\ \ Images}.$$

Table III shows both the training and test results for these four algorithms ($L = 1800$) in terms of PPA and RMSE, with threshold θ set to 15 and 20, respectively. It is observed that both SCN and 2DSCN outperform RVFL-based algorithms, and 2DSCN has the highest PPA and lowest RMSE values, which reflects a better learning and generalization capability. Specifically, given $\theta = 25$, the 2DSCN-based learner model contributes 100% (training) and 98.12% (test) PPA values, compared with 2DRVFL with 99.78% and 95.72%, respectively. Based on a simple calculation according to the PPA values and the number of training/test samples, we can immediately notice that 2DRVFL produces some training errors with absolute values larger than 25 degrees whereas all the training predictive errors of 2DSCN are under this threshold. Also, there are less than 100 test digits predicted by 2DSCN, however, more than 200 instances by 2DRVFL, with degree errors outside the interval $[-25, 25]$. It should be noted that all the results illustrated in Table III are the averaged values based on 50 independent trials. Based on a rule of thumb, their standard deviations have no significant difference, so we omit this information here without any confusion. Here we only consider randomized learner models with shallow structures. Some recent work studying the manifold regularized deep learning architecture or hierarchical convolutional features [37]–[39] can also offer alternative methods for this image-based regression task.

TABLE III
PERFORMANCE COMPARISON ON TRAINING (TR) AND TEST (TE)

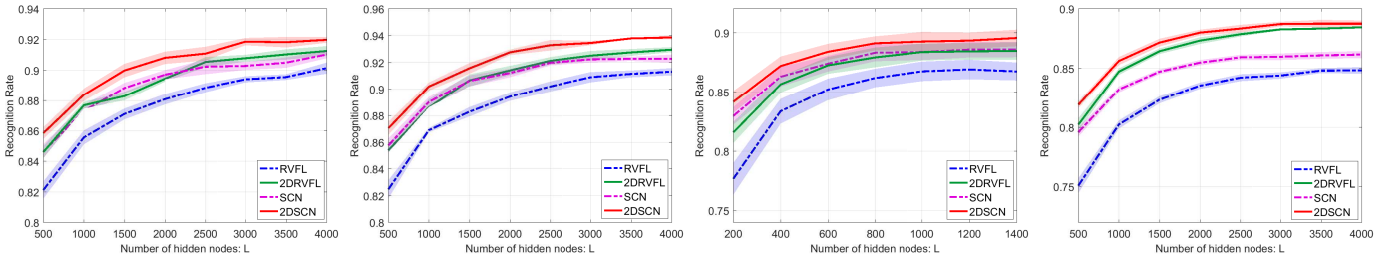
Algorithms	PPA (%), $\theta = 15$		PPA (%), $\theta = 25$		RMSE	
	Tr	Te	Tr	Te	Tr	Te
RVFL	95.15	79.97	99.76	95.64	7.51	12.04
SCN	99.84	87.19	99.98	98.08	4.57	10.08
2DRVFL	95.59	81.05	99.78	95.72	7.32	11.91
2DSCN	**99.88**	**87.55**	**100**	**98.12**	**4.42**	**9.96**

B. Classification: Handwritten Digit Recognition

In this part, we compare our proposed 2DSCN algorithm with the other three randomized approaches on the image classification problem. Four benchmark handwritten digits databases are employed in the comparison. Parameter setting for these four algorithms ($w, b, u, v, \lambda, T_{max}, r$) is the same as the configuration used in the previous regression task. In particular, Bangla, Devnagari, and Oriya handwritten databases provided by the ISI (Indian Statistical Institute, Kolkata)[2], and CVL which is usually used for pattern recognition competitions[3], are employed in our experimental study as a benchmark resource for optical character recognition. Readers can refer to our previous work [12] for a more description of these datasets as well as their training and test division.

1) Results and Discussion: Fig. 3 compares the performance of the testing in terms of different settings of L, in which the mean and standard deviation values of recognition rate are based on 10 independent trials (more trails can provide more convincing results but this is not necessary based on our experience, because the standard deviation values are relatively small and stable at a certain level). It is apparent that 2DSCN outperforms the others while RVFL has the worst results for all four datasets. Interestingly, the test results of 2DSCN are much better than that of SCN. For Bangla, Devnagari, and Oriya, 2DRVFL occasionally results in slightly higher recognition rates in testing. It becomes much clearer in the subplots for the CVL dataset, i.e., 2DRVFL works more favorably than SCN but it is still worse than 2DSCN, which to some extent, lends strong support to our theoretical investigation of the superiority of randomized learners with 2D inputs.

[2]http://www.isical.ac.in/ ujjwal/download/database.html
[3]http://www.caa.tuwien.ac.at/cvl/category/research/cvl-databases/

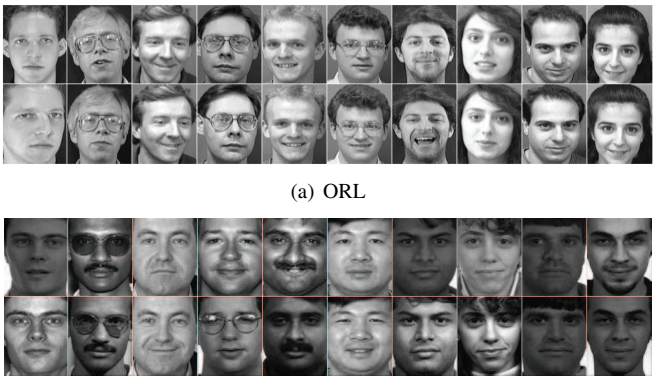


(a) Bangla:Test (b) Devnagari:Test (c) Oriya:Test (d) CVL:Test

Fig. 3. Performance comparison among 2DSCN, 2DRVFL, SCN, and RVFL on Bangla, Devnagari, Oriya, and CVL database.

C. Case Study: Human Face Recognition

We further demonstrate the advantage of 2DSCN over the other three randomized learner models on human face recognition tasks, where the input dimensionality is far more larger than that of the handwritten digit problems previously addressed.



(a) ORL

(b) FERET

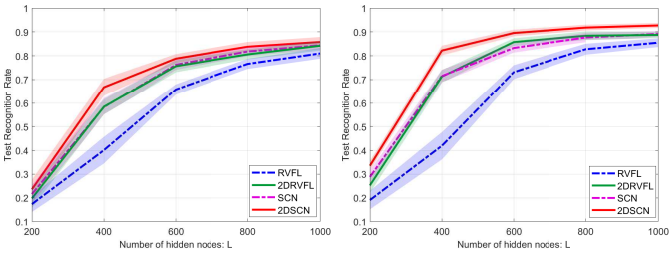
Fig. 4. Samples of face images from the ORL and FERET database. Ten subjects with two expressions for each are displayed.

1) Databases: ORL [40]: The Olivetti (ORL, now AT&T) database contains ten 112×92 pixel gray scale images of 40 different individuals. Some of the images were taken at different times, with various lighting conditions, different types of facial expressions (open/closed eyes, smiling/not smiling) and varying facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

FERET [41]: The Facial Recognition Technology (FERET) database contains a total of 14,126 gray scale images for 1199 subjects, which were collected over several sessions spanning more than three years. For some individuals, over two years had elapsed between their first and last sittings, with some subjects being photographed multiple times. In our experimental study, we choose 72 subjects with 6 frontal images of size 112×92 per person.

From these two databases, we randomly select half of the images for each subject as the training samples and the other half for testing.

2) Results and Discussion: In our experiments, we employ the same parameter settings for these four algorithms (see part A for details), and conduct the performance comparison with various setups for the number of hidden nodes, i.e., $L = 200, 400, 600, 800, 1000$, respectively. For each dataset, 50 independent trials are performed on each L for all four algorithms. Fig. 5 shows their test recognition rates with both mean and standard deviation values. Similar to what we have previously demonstrated, 2DSCN clearly outperforms the other three algorithms in terms of generalization. On the other hand, we should note that the training recognition rate for all these algorithms in all cases is 1, which means that each method with the current parameter settings exhibits sufficient learning capability, however, there is significant discrepancy in generalization ability. Furthermore, as shown in Fig. 5, there is no large difference between the performance of 2DRVFL and that of SCN, and both achieve better results than RVFL. It is fair to say that 2D models have more advantages than 1D models in both training and testing, similar to what was observed in Parts A and B. Based on Theorem 5 in Section III,



(a) ORL (b) FERET

Fig. 5. Comparison of the test recognition rate of 2DSCN, 2DRVFL, SCN, and RVFL on ORL and FERET.

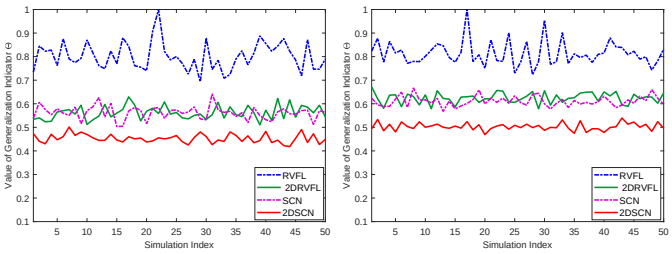
we can numerically estimate the test error upper bound for the learner model produced by each of these four algorithms. Note that the upper bound in Eq. (13) comprises three parts, that is, training error $\|H\beta - Y\|_F$, $\epsilon \max_{1\le i\le N} \|Z_i\|_2 \cdot \|H \circ (O - H) \circ \ddot{W}\|_F \|\beta\|_F$, and the sufficiently small term $o(\epsilon^2)\|\beta\|_F$, respectively. Since their training errors stay at the same level (training recognition rate equals to 1 for each case) and the fact that $\|\beta\|_F$ also occurs in the second part, we only need to consider the second term but without the common factor $\max_{1\le i\le N} \|Z_i\|_2$ in our empirical examination. In particular, for each method, we consider the case of $L = 600$ and calculate the value of $\|H \circ (O - H) \circ \ddot{W}\|_F \cdot \|\beta\|_F$ for the 50 resulting learner models based on independent trials, followed by a normalization operation by dividing the corresponding

figure by the maximum value of all the $50 \times 4 = 200$ records. We call this numerical (predictive) upper bound the generalization indicator $\Theta := \{\Theta_i\}_{i=1}^{200}$, denoted by (refer to Section III for some notations)

$$\Theta_i = \frac{\{\|H \circ (O - H) \circ \ddot{W}\|_F \cdot \|\beta\|_F\}_i}{\max\{\|H \circ (O - H) \circ \ddot{W}\|_F \cdot \|\beta\|_F\}_{i=1}^{200}},$$

where the index i corresponds to the i-th record among the total 200 records.

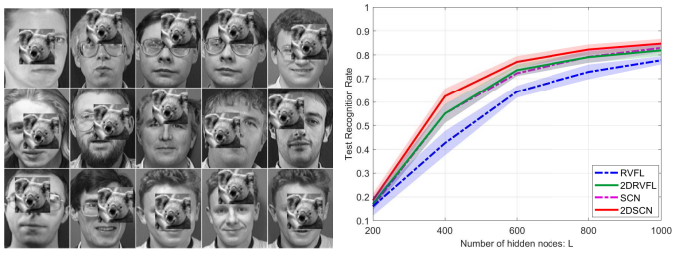
Fig. 6 plots all these 200 records for the four algorithms (50 records for each), in which we can clearly observe that 2DSCN exhibits a lowest test error upper bound than the other three methods while RVFL has the highest results. Interestingly, this is consistent with the comparison of their real test recognition rate shown in Fig. 5, therefore, this verifies the effectiveness and practicability of our Theorem 5. Based on our experience, similar results can be obtained with the other option of L setting, and as such, learner models built by the 2DSCN algorithm have the smallest predictive test error upper bound estimation. As for the space limitation, more statistical results and analysis are left for our future work.



(a) ORL (b) FERET

Fig. 6. Test error upper bound comparison for 2DSCN, 2DRVFL, SCN, and RVFL on ORL and FERET.

3) Robustness Illustration: To further investigate the superiority of 2DSCN over the other methods, we randomly select 50 images from the ORL training set and artificially simulate contiguous occlusion by replacing a randomly located square block of each chosen image with an unrelated image (e.g., *Koala*). Fifteen of these corrupted images are displayed in Fig. 7(a). Our objective is to compare the test performance of these four algorithms and study their capability to perform robustly in training, that is, to what extent can they alleviate the impacts of random block occlusion attached in the training images. All details for the experiment setup remain the same as those used for the original (clean) ORL dataset. As can be seen in Fig. 7(b), 2DSCN still performs the best in generalization, and at the same time, RVFL has the lowest test recognition rates. This observation can to some extent implies that 2D models may still exhibit their merits for problem-solving in relation to robust image data modelling, on the basis of their underlying advantages as shown throughout this paper. Due to the space limitation, we do not conduct in-depth comparisons and analysis on this task, although various methods address weakly supervised face classification/detection [42]. This is beyond the scope of our study.



(a) (b)

Fig. 7. (a): Sample images from ORL with random block occlusion; (b): Test recognition rate comparison for 2DSCN, 2DRVFL, SCN, and RVFL on corrupted ORL.

D. Comparison with Classic Deep CNN Models

In this section, we compare the 2DSCN model with 11 classic deep CNN models: VGG16 (with depth 16 and around 138 million parameters) and VGG19 (with depth 19 and around 144 million parameters) [43], AlexNet [44] (with depth 8 and around 61 million parameters), ResNet-18 (with depth 18 and around 11.7 million parameters), ResNet-50 (with depth 50 and around 25.6 million parameters), ResNet-101 (with depth 101 and around 44.6 million parameters) [45], GooLeNet (with depth 22 and around 7 million parameters) [46], SqueezeNet (with depth 18 and around 1.24 million parameters) [47], DenseNet-201 (with depth 201 and around 20 million parameters) [48], Inception-v3 (with depth 48 and around 23.9 million parameters) [49], and Inception-ResNet-v2 (with depth 164 and around 55.9 million parameters) [50], on the CIFAR-10 dataset[4] which has 60000 32×32 colour images in 10 classes, with 6000 images per class. We should clarify that we only use the pretrained modules of these models as the starting point to build a deep classifier on CIFAR-10, that is, transfer learning, which is typically much faster and easier than training a network from scratch. The pretrained networks are built on more than a million images and have already learned to extract powerful and informative features from natural images. The training images are a subset of the ImageNet database[5], which is used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [51]. For simplicity, we only consider four categories in CIFAR-10, i.e., 'deer', 'dog', 'frog', 'cat', and randomly select 500 sample images per class for training (2000 images in total) and the whole set of test images in these four classes for testing (4000 images in total). These deep CNN models can be imported from MATLAB2018b Deep Learning Toolbox by using the Add-On Explorer. In AlexNet, VGG16 and VGG19, the last layer with learnable weights is a fully connected layer, therefore, we have to replace this fully connected layer by specifying 'numClasses==4' to fit the classification task on CIFAR-10. In some networks such as SqueezeNet, the last lacerable layer is a 1-by-1 convolutional layer instead. In such cases, we modify the convolutional layer with a new convolutional layer with the number of filters equal to 4. Furthermore, we specify the training options with 'InitialLearnRate==0.001', 'Max-

[4]CIFAR-10. https://www.cs.toronto.edu/ kriz/cifar.html
[5]ImageNet. http://www.image-net.org

Epochs==20', 'MiniBatchSize==60', "ExecutionEnvironment=='cpu"'. For 2DSCN, we set the number of hidden nodes as 850, which is selected based on sufficient empirical trials with $L = 500 : 50 : 1000$, and use a similar hyper-parameter setting as in the previous simulations.

TABLE IV
COMPARISON WITH CLASSIC DEEP CNN MODELS ON CIFAR10

Model	Accuracy (%)		Elapsed Time (s)	
	Training	Test	Training	Test
VGG16 [43]	90.81	89.08	31,721	978
VGG19 [43]	91.17	90.03	38,025	1,267
AlexNet [44]	88.29	86.00	2,460	76
ResNet-18 [45]	88.88	86.25	6,088	175
ResNet-50 [45]	89.00	87.33	18,523	448
ResNet-101 [45]	90.27	88.90	30,669	720
GoogLeNet [46]	90.47	89.20	7,287	236
SqueezeNet [47]	85.75	84.60	4,427	146
DenseNet-201 [48]	91.73	90.65	68955	1,124
Inception-v3 [49]	84.88	83.98	25,888	736
Inception-ResNet-v2 [50]	85.54	84.70	106,795	1,424
2DSCN	88.20	85.63	313	16

Table IV demonstrates the training and test results for these deep CNN models and our 2DSCN model. It is clear that the accuracy of 2DSCN can match most of the deep CNN models whilst it has the least time cost in both training and testing. It is worth mentioning that 2DSCN appears inferior to some models in terms of accuracy, however, this is excusable in view of two aspects: (i) Convolution operations can benefit the abstract feature extraction, which contributes to a preferable classifier. 2DSCN is not equipped with the convolutional layer, which it seems relatively unfair to put it in the lineup of deep CNN models; (ii) 2DSCN only has one hidden layer rather than a very deep structure. Furthermore, some tricks like batch-normalization, pooling and dropout, as contributors for improving generalization capability in deep CNN models, are not included in the current design of 2DSCN. In spite of these points, 2DSCN has shown good potential in image modeling with acceptable effectiveness and sufficient efficiency. Before ending this section, we should clarify again that our main focus in this paper is the extension of our previous SCN framework [12] to a 2D version, rather than building deep CNN ones. In our future work, we will devote more effort to enhancing 2DSCN with convolutional layers and a deep architecture.

V. CONCLUSIONS

This paper develops two dimensional stochastic configuration networks (2DSCNs), which extend the original SCN framework for data analytics with matrix inputs. Compared to existing randomized learning techniques, the proposed algorithm maintains all the advantages of the original learning techniques for SCNs, such as fast modelling, universal approximation property, and sound generalization power. Some associations and differences between 2DSCNs and SCNs are theoretically investigated and empirically justified. Our main technical contribution in this paper lies in an interpretation of the reasons behind the improved generalization of 2DSCNs against the original SCNs for image data modelling. Compared to the performance obtained from SCNs, RVFLs and 2DRVFLs, we conclude that 2DSCNs outperform in terms of both learning and generalization.

There are many interesting studies left for future work. The framework and the associated theoretical analysis are generic and general enough to adopt deep machinery. Therefore, one can make efforts towards building a deep 2DSCN to maintain both the superiority of DeepSCN [15] and the advantages of 2DSCN. Alternatively, one can employ the 2DSCN-based autoencoder and then stack the configured autoencoders in a greedy layerwise fashion to build a deep network. It is also of practical importance to employ the proposed 2DSCN in stream image data modelling or robust data modelling [13], [16], and it is meaningful to further enhance 2DSCN by considering the regularization learning framework or refined stochastic configuration inequality with sparsity constraints. Various extensions of the 2DSCN framework to versions with convolutional layers or sparse learning representation or (deep) kernel learning or tensor representation (or multi-channel) for dealing with color images (or videos), are expected.

ACKNOWLEDGMENT

The authors would like to thank the editors and reviewers for their suggestions on improving this paper. This work was supported by the National Natural Science Foundation of China (No. 61802132, No. 61890924), and the China Postdoctoral Science Foundation Grant (No. 2018M630959).

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[2] N. Qi, Y. Shi, X. Sun, J. Wang, and B. Yin, "Two dimensional synthesis sparse model," in *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1–6, 2013.
[3] N. Qi, Y. Shi, X. Sun, J. Wang, and W. Ding, "Two dimensional analysis sparse model," in *Proceedings of the 20th IEEE International Conference on Image Processing*, pp. 310–314, 2013.
[4] N. Qi, Y. Shi, X. Sun, J. Wang, B. Yin, and J. Gao, "Multi-dimensional sparse models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 1, pp. 163–178, 2018.
[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[6] J. Gao, Y. Guo, and Z. Wang, "Matrix neural networks," *arXiv preprint arXiv:1601.03805*, 2016.
[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by backpropagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
[8] S. Liu, Y. Sun, Y. Hu, J. Gao, F. Ju, and B. Yin, "Matrix variate RBM model with gaussian distributions," in *Proceedings of International Joint Conference on Neural Networks*, pp. 808–815, 2017.
[9] S. Scardapane and D. Wang, "Randomness in neural networks: An overview," *WIREs Data Mining and Knowledge Discovery*, vol. e1200. doi: 10.1002/widm.1200, 2017.
[10] Y. H. Pao, G. H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.
[11] B. Igelnik and Y. H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.
[12] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Transactions on Cybernetics*, vol. 47, no. 10, pp. 3466–3479, 2017.
[13] D. Wang and M. Li, "Robust stochastic configuration networks with kernel density estimation for uncertain data regression," *Information Sciences*, vol. 412, pp. 210–222, 2017.

[14] D. Wang and C. Cui, "Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics," *Information Sciences*, vol. 417, pp. 55–71, 2017.
[15] D. Wang and M. Li, "Deep stochastic configuration networks with universal approximation property," In *Proceedings of International Joint Conference on Neural Networks*, pp. 1–8, 2018.
[16] M. Li, C. Huang, and D. Wang, "Robust stochastic configuration networks with maximum correntropy criterion for uncertain data regression," *Information Sciences*, vol. 473, pp. 73–86, 2019.
[17] M. Pratama, and D. Wang, "Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams," *Information Sciences*, vol. 495, pp. 150–174, 2019.
[18] J. Lu, J. Zhao, and F. Cao, "Extended feed forward neural networks with random weights for face recognition," *Neurocomputing*, vol. 136, pp. 96–102, 2014.
[19] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, "Approximation with random bases: Pro et contra," *Information Sciences*, vol. 364, pp. 129–145, 2016.
[20] M. Li and D. Wang, "Insights into randomized algorithms for neural networks: Practical issues and common pitfalls," *Information Sciences*, vol. 382–383, pp. 170–178, 2016.
[21] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, pp. 1058–1066, 2013.
[22] P. Lancaster and M. Tismenetsky, *The Theory of Matrices: With Applications*. Elsevier, 1985.
[23] P. Xie, H. Zhang, and E. P. Xing, "Learning less-overlapping representations," *arXiv preprint arXiv:1711.09300*, 2017.
[24] P. Xie, Y. Deng, and E. Xing, "On the generalization error bounds of neural networks under diversity-inducing mutual angular regularization," *arXiv preprint arXiv:1511.07110*, 2015.
[25] P. Xie, Y. Deng, Y. Zhou, A. Kumar, Y. Yu, J. Zou, and E. P. Xing, "Learning latent space models with angular constraints," in *Proceedings of the 34th International Conference on Machine Learning*, pp. 3799–3810, 2017.
[26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[28] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.
[29] K. Zhong, Z. Song, P. Jain, P. L. Bartlett, and I. S. Dhillon, "Recovery guarantees for one-hidden-layer neural networks," *arXiv preprint arXiv:1706.03175*, 2017.
[30] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
[31] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, no. 26, pp. 314–319, 1985.
[32] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
[33] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning*, pp. 265–272, 2011.
[34] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems*, pp. 1313–1320, 2009.
[35] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing, 2010.
[36] M. Springer and W. Thompson, "The distribution of products of Beta, Gamma and Gaussian random variables," *SIAM Journal on Applied Mathematics*, vol. 18, no. 4, pp. 721–737, 1970.
[37] Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 10, pp. 2222–2233, 2015.
[38] X. Lu, X. Zheng, and Y. Yuan, "Remote sensing scene classification by unsupervised representation learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 5148–5157, 2017.
[39] X. Lu, Y. Chen, and X. Li, "Hierarchical recurrent neural hashing for image retrieval with hierarchical convolutional features," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 106–120, 2018.
[40] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," In *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, pp. 138–142, 1994.
[41] P. J. Phillips, H. Wechsler, J. Huang, and P. J. Rauss, "The FERET database and evaluation procedure for face-recognition algorithms," *Image and Vision Computing*, vol. 16, no. 5, pp. 295–306, 1998.
[42] Q. Huang, C. K. Jia, X. Zhang, and Y. Ye, "Learning discriminative subspace models for weakly supervised face detection," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2956–2964, 2017.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.
[46] C. Szegedy, W.Liu, Y. Jia, P. Sermanet, S. Reed, and et al., "Going deeper with convolutions," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.
[47] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.
[48] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2261–2269, 2017.
[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.
[50] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," In *Proceedings of the 31st AAAI Conference on Artificial Intelligence* , pp. 4278-4284, 2017.
[51] O. Russakovsky, J.Deng, H. Su, J. Krause, S. Satheesh, and et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

Ming Li is doing a postdoctoral fellowship in the Department of Information Technology in Education at South China Normal University, China. Before this, he received his PhD degree from the Department of Computer Science and IT at La Trobe University, Australia, in 2017. Before that, he was awarded a Master degree in Applied Mathematics from China Jiliang University, China, in 2013, and a Bachelor degree in Information and Computing Sciences from Shandong Normal University, China, in 2010. His research interests include deep learning, fast and randomized learning algorithms, robust modelling techniques, educational data analytics, applied and computational harmonic analysis.

Dianhui Wang (M'03-SM'05) was awarded a Ph.D. from Northeastern University, Shenyang, China, in 1995. From 1995 to 2001, he worked as a Postdoctoral Fellow with Nanyang Technological University, Singapore, and a Researcher with The Hong Kong Polytechnic University, Hong Kong, China. He joined La Trobe University in July 2001 and is currently a Reader and Associate Professor with the Department of Computer Science and Information Technology, La Trobe University, Australia. He is a visiting Professor at The State Key Laboratory of Synthetical Automation of Process Industries, Northeastern University, China. His current research focuses on industrial big data oriented machine learning theory and applications, specifically on Deep Stochastic Configuration Networks (http://www.deepscn.com/) for data analytics in process industries, intelligent sensing systems and power engineering.

Dr Wang is a Senior Member of IEEE, and serving as an Associate Editor for IEEE Transactions On Neural Networks and Learning Systems, IEEE Transactions On Cybernetics, Information Sciences, and WIREs Data Ming and Knowledge Discovery.