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 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 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, four benchmark handwritten digits classification, and two human face recognition datasets 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

Along 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-propagation (BP) algorithm [7] to a general version that is capable for dealing with 2D inputs. Empirical results in [6] and their parallel work [8] demonstrate some 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. That indicates a urgent demand for developing 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, RVFL networks randomly assign the input weights and biases from a fixed distribution (range) that is totally data-independent, and optimize merely the output weights by solving a linear least squares problem. This trivial idea sounds computationally efficient, however, the obtained learner models may not have universal approximation capability when an inappropriate distribution (range) is used for the random assignment. This drawback indeed makes RVFL networks less practical in data modelling problems because more human intervention and/or empirical knowledge is required for problem-solving. Fortunately, SCNs, as state-of-the art randomized leaner models, were developed with rigorous theoretical fundamentals and advanced algorithm implementations [12]. The success of SCNs and their
contributions can be summarized in three-fold: verified by the statistical simulation results. Overall, our main demonstrated remarkable advantages of 2DSCN on image data performance. Extensive experimental study on both regression bound, thereby one can effectively predict the generalization ability of an obtained randomized learner model, we make efforts towards developing RVFL networks as highlighted above.

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 for resolving data analytics 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 with highlights in various aspects, such as the support range for random parameters, the complexity for parameter space, data structure preservation. Among that and interestingly, our work is the first to think about a potential relationship between 2DSCN and CNN in problem-solving, that is, computations involved in 2DSCN in some sense can be viewed as equivalent to the ‘convolution’ and ‘pooling’ tricks performed in CNN structure. Later, some technical issues around why randomized learner models produced by 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 qualitative analysis on learner models’ generalization power and useful insights on certain very influential factors. Besides, we provide an intuitive sense that 2DSCN may exhibit similar philosophy as concerned in DropConnect [16] for effectively alleviating over-fitting. Importantly, to make a reasonable and practical judgement on the generalization ability of an obtained randomized learner model, we make efforts towards developing a nearly sharp estimation about the model’s test error upper bound, thereby one can effectively predict the generalization performance. Extensive experimental study on both regression and classification problems (with matrix inputs setup) have demonstrated remarkable advantages of 2DSCN on image data modelling, compared to some existing randomized learning techniques. Also, our theoretical analysis has been successfully verified by the statistical simulation results. Overall, our main contributions can be summarized in three-fold:

- From the algorithmic perspective, we extend our original SCN framework to 2D version, and the proposed 2DSCN algorithm can 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 algorithm on image data analytics have been illustrated on various benchmark tasks, such as the rotation angles predication for handwritten digits, handwritten digits classification, and human face recognition. The extensive experimental study conducted in this paper can lend some empirical support to end-users who would like to employ FCNNs rather than CNNs in image data modeling.

The remainder of this paper is organized as follows. Section II provides 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 description, technical highlights, and some theoretical explanation, aiming to distinguish 2DSCN from the other randomized learning techniques. Section IV presents experimental study in terms of both image-based regression and classification problems, and Section V concludes this paper with further remarks and expectation.

II. RELATED WORK

This section reviews two types of randomized learner models with highlights in their technical discrepancy. First, 2DRVFL networks as an extension of RVFL networks can deal with matrix inputs but could not guarantee universal approximation in data modelling, which causes some infeasibility as well as uncertainty for problem-solving; SCNs as an advanced universal approximator have demonstrated their effectiveness and efficiency in data analytics with vector inputs as usually done. Basically, the brief examination of these two methodology motivates us to think about the formulation of 2DSCN and also its potential advantages in image data modelling problems, as to be detailed in the following section.

A. 2DRVFL Networks

2DRVFL networks with matrix inputs has been empirically studied in [15]. Technically, it can be viewed as a trivial extension of the original RVFL networks in computation via employing two sets of input weights acting as matrix transformation over the left and right sides of 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.,

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

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. $\phi(\cdot)$ is the activation function.

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

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

where

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

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

**Remark 1.** Although RVFL networks (with either vector or matrix inputs) allows fast building a model by randomly assigning input weights and biases, some key technical issues are still unresolved. Theoretically, approximation error for this kind of randomized learner model are bounded in the statistical sense, which means preferable approximation performance is not guaranteed for every random assignment of the hidden parameters [17]. Besides, it has been proved that in the absence of such additional conditions, one may observe exponential growth of the number of terms needed to approximate a non-linear map, and/or the resulting learner model will be extreme sensitivity to the parameters [17].

From the algorithmic perspective, all these theoretical predictions do not address the learning algorithm or implementation issues for the randomized learner. Practical usage of this kind of randomized model encounter 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 perform various setting of $\lambda$ before getting an acceptable learner model. This trick sounds practical but still has potential drawbacks due to uncertainty causes by the randomness, as theoretically and empirically studied in [18]. We also note that one can try out different random selection range 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 the foundation of building 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. Readers who are interested in a complete roadmap of this novel work can refer to [12]. Here we just briefly revisit the essence and highlight some technical points.

Let $\Gamma := \{g_1, g_2, g_3, \ldots\}$ represent a set of real-valued functions, and span($\Gamma$) stands for the associated function space spanned by $\Gamma$. $L_2(K)$ denote 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

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

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

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

Note that this definition becomes the 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 $L$ hidden nodes, i.e., $f_{L-1}(x) = \sum_{j=1}^{L-1} \beta_j g_j(u_j^T x + b_j)$ ($L = 1, 2, \ldots$, $f_0 = 0$), with $\beta_j = [\beta_{j1}, \ldots, \beta_{j,m}]^T$, and residual error $e_{L-1} = f - f_{L-1} = [e_{L-1,1}, \ldots, e_{L-1,m}]$ far away from an acceptable accuracy level, our SCN framework can successfully offer a fast solution to incrementally add $\beta_L$, $g_L$ ($w_L$ 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 $\epsilon$. The following Theorem 1 restates the universal approximation property of SCNs, corresponding to Theorem 7 in [12].

**Theorem 1.** Suppose that span($\Gamma$) is dense in $L_2$ space and $\forall q \in \Gamma$, $0 < ||g|| < b_q$ for some $b_q \in \mathbb{R}^+$. 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

$$\delta_L = \sum_{q=1}^{m} \delta_{L,q}$$

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

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

and the output weights are evaluated by

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

it holds that $\lim_{L \rightarrow +\infty} ||f - f_L|| = 0$, where $f_L = \sum_{j=1}^{L} \beta_j g_j$, $\beta_j = [\beta_{j1}, \ldots, \beta_{j,m}]^T$.

Basically, the algorithmic procedures for building SCNs can be summarized as repeating the following sessions with $L = 1, 2, 3 \ldots$ until the given training error tolerance is reached:
Stochastically configure a new hidden node \( g_L \) (i.e., find out random \( w_L \) and \( b_L \) from support range) based on the inequality (5):

- Evaluate \( \beta \) by solving the linear least squares problem expressed in Eq. (6).
- Calculate the current training error \( e_L \) and check the termination condition is met or not.

**Remark 2.** We would like to highlight that SCNs outperforms some existing randomized learning techniques (e.g., RVFL networks) that employ a totally data-independent randomization in training process, and consider considerable advantages in building fast learner models with 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 are proposed towards various viewpoints. In [19], an ensemble version of SCNs with heterogeneous features was developed with applications in large-scale data analytics. In [14], 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 great features, such as learning representation and consistency property between learning and generalization. Besides in [13], we built robust SCNs for the purpose of uncertain data modelling. This series of work to some extent exhibits the effectiveness of SCN framework and have displayed an advisable and useful way on studying/implanting 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, comparison around some technical points between these two methods are discussed. Afterwards, a theoretical analysis why randomized learner models with 2D inputs have a good potential for inducing better generalization is provided.

#### A. Algorithm Implementation

On the basis of 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, \( f_{L-1}(x) = \sum_{j=1}^{L-1} \beta_j g_j(u_j^T x v_j + b_j) \) \((L = 1, 2, \ldots, f_0 = 0)\), 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 constrains), \( \beta_j = [\beta_{j,1}, \ldots, \beta_{j,m}]^T \) 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 fast generate a new hidden node \( g_L \) (in lieu of \( u_L, v_L, \) and \( b_L \)) so that the resulted model \( f_L \) has 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 \( t_i = [t_{i,1}, \ldots, t_{i,m}]^T \in \mathbb{R}^m \), \( i = 1, \ldots, N \), sampled based on a target function \( f : \mathbb{R}^{d_1 \times d_2} \rightarrow \mathbb{R}^m \). Denoted by \( e_{L-1} = [e_{L-1,1}(X), e_{L-1,2}(X), \ldots, e_{L-1,m}(X)]^T \in \mathbb{R}^{N \times m} \) as the corresponding residual error vector before adding the \( L \)-th new hidden node, where \( e_{L-1,q} := e_{L-1,q}(x_i) = [e_{L-1,1,q}(x_i), \ldots, e_{L-1,q}(x_i)] \in \mathbb{R}^N \) for \( q = 1, 2, \ldots, m \).

With \( N \) two dimensional inputs \( \{x_1, x_2, \ldots, x_N\} \), the \( L \)-th hidden node activation can be expressed as 
\[
h_L := h_L(X) = [g_L(u_1^T x v_1 + b_1), \ldots, g_L(u_L^T x v_L + b_L)]^T,
\]
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 \( \xi_{L,q}, q = 1, 2, \ldots, m \) as follows:
\[
\xi_{L,q} = \frac{(e_{L-1,q}^T h_L)^2}{h_L^T h_L} - (1 - r)e_{L-1,q}^T e_{L-1,q}.
\]

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

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.,
\[
\beta = \arg \min_{\beta} \| H_L \beta - T \|^2_F = H_L^+ T,
\]
where \( H_L^+ \) is the Moore-Penrose generalized inverse [20] and \( \cdot \| \cdot \|_F \) represents the Frobenius norm.

#### 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 constrains 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, \( \lim_{L \to +\infty} \| e_{L} \| = 0 \).

We remark that \( r \) value is varying 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 considerable amount of hidden nodes are successfully configured. To some extent, this user-determined (and problem-dependent) parameter affects the algorithm convergence speed. In particular, one can set \( r \) sequence (monotonically increasing) with 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, user can start with a relatively small value (but cannot be too small), which however requires more configuration trials at one single step to find suitable input weights and biases that fit the inequality condition. It can lead to huge computational burden...
Algorithm 1: 2DSCN

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

Output: A 2DSCN model

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

or even more unnecessary fails during the configuration phase. Since the convergence property is guaranteed theoretically, one can think about some practical guideline for setting \( r \) sequence with reference to their practical task. Based on our experience, the first trick, i.e., initializing \( r \) with value close to one and then monotonically increase (progressively approaching one), offers more feasibility in algorithm implementation. Later in Section IV, we will recall this note in our experimental setup.

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 way 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, differences between support ranges induced by these two methods should be highlighted. Computationally, it holds that

\[ u^T x v = \text{Tr}(u^T x v) = \text{Tr}(x v u^T) = \text{Tr}(u v^T x) = (v e c(u v^T))^T v e c(x), \]

where \( \text{Tr} \) means the matrix trace, \( u \in \mathbb{R}^{d_1} \), \( v \in \mathbb{R}^{d_2} \), \( v e c(\cdot) \in \mathbb{R}^{d_1 \times d_2} \) stands for vectorization of a given 2D array. We observe that although \((v e c(u v^T))^T v e c(x)\) can be viewed as a regular dot product (between the hidden weight vector and input) computation performed in SCN, the resulted \((d_1 d_2)\)-dimensional vector \( v e c(u v^T) \) may exhibit 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 \( \Upsilon_u = \{\lambda_1^u, \ldots, \lambda_{\text{end}}^u\} \) and \( \Upsilon_v = \{\lambda_1^v, \ldots, \lambda_{\text{end}}^v\} \) respectively in their experimental setup. If so, in algorithm design, one more loop is need for searching appropriate \( \lambda^v \) from \( \Upsilon_v \) when \( \lambda^u \) is chosen and fixed, or vice versa. Since universal approximation capability is always guaranteed, this complex manipulation sounds not computationally efficient in practical implementation . For simplicity, we just use the same random range setting for \( u \) and \( v \), i.e., merely \( \Upsilon \), as noticed 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 \( \Upsilon_u \) or \( \Upsilon_v \) and the inequality constrain for hidden node configuration. Regarding discrepancies between 2DSCN and SCN in random parameter distribution, we will elaborate more details at the end of this section.

2) Parameter Space: Despite that neural networks can universally approximate complex multivariate functions, they still suffers from difficulties on high-dimensional problems where the number of input features is much larger than the number of observations. Generally, a huge number of training observations is required for training/building an acceptable approximator, as normally performed in deep learning community. Empirically, problems with very limited number of training samples but of very high dimension usually need further technical concerns in algorithm development, like feature selection or learning representation with sparsity (e.g., Lasso). To avoid high-dimensional inputs and seek useful input features for the alleviation of overfitting are important and essential for the majority of machine learning techniques.

It is clear that one 2DSCN model with \( L \) \((d_1 + d_2)\)-dimensional weights and \( L \) \((d_1 + d_2)\)-dimensional input weights, \( L \) biases (scalar), \( L \) \(m\)-dimensional output weights, that is, \( L \times (d_1 L m) + 2 \) parameters in total; whilst 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 to meet the stochastic configuration inequality \((\text{\ref{eq:16}})\), especially when the number of training samples is far lower than the dimensionality of the input weights. Besides, for a relatively large \( L \), huge memory is needed for saving \( L \times (d_1 \times d_2 + 1 + m) \) 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 2DSCN has some advantages in preserving the spatial details of the given input images, due to that it cares about the 2D-neighborhood information (the order in which pixels appear) of the input rather than a simple vectorization operation.
performed in SCN. This argument has been raised and commonly accepted in literature, however, there is no sufficient scientific evidence verifying why and how the vectorization trick affect the structural information of the 2D inputs. In this part, we aim at examining the resemblance between 2DSCN methodology and convolutional neural networks (CNNs) in terms of computational perspective. A schematic diagram is plotted in Fig. 2 and corresponding explanations are as follows.

Recall Eq. (10), the left low-dimensional vector $u^T$ is acting as a ‘filter’ used in extracting 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 $u^T x$ 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 $[u^T x_1, u^T x_2, \ldots, u^T x_{d_2}]$. 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 are left for our future study.

![Fig. 2. Schematic diagram for computational equivalence between 2DSCN and CNN.](image)

\[\text{Input Image} \quad u^T \quad \text{convolution along vertical direction} \quad \text{pooling} \quad \text{Non-Linear Activation} \quad \text{output} \]

\[\text{‘feature map’} \quad \text{‘filter’} \quad \text{‘pooling’ by weighted sum} \quad \text{Activation} \]

\[\{\bar{w}_i\}_{i=1}^L, [21]\] proposed a hybrid regularizer consisting of a orthogonality-promoting term and sparsity-promoting term, denoted by

\[\Omega(\mathcal{W}) = \text{tr}(M) - \log \det(M) + \gamma \sum_{i=1}^L \|\bar{w}_i\|_1,\]

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

Theoretically, the first term $\text{tr}(M) - \log \det(M)$ controls the level of near-orthogonal over the weight vectors from $\mathcal{W}$, while the second term $\sum_{i=1}^L \|\bar{w}_i\|_1$ encourage $w_i \in \mathcal{W}$ to have more elements close to zero. It is empirically verified that this hybrid form of regularizer can contribute to learner models with better generalization performance [21].

Back to our thesis, 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 higher level of sparsity than that in SCN, hence leading to a smaller $\sum_{i=1}^L \|\bar{w}_i\|_1$. This deduction can also be used to differ 2DRVFL from RVFL as well; (ii) Given similar level of sparsity in $\mathcal{W}$, SCN-based models are more prone to have a lower near-orthogonal level than RVFL-based ones, therefore, indicating a smaller $\text{tr}(M) - \log \det(M)$. For further justifications on these intuitive arguments, we present analogous theories regarding the near-orthogonality of weight vectors in the following part (see ST2 below) and demonstrate some statistical results at the end of this section.

**ST2: Weight Vector Angular Constraints for Diversity Promoting.** Authors in [22], [23] have shown empirical effectiveness and explained in theory when and why a low generalization error can be achieved via adjusting the diversity of hidden nodes in neural networks. Theoretically, increasing the diversity of hidden nodes in a neural network model would reduce estimation error but increase approximation error, which implies that a low generalization error can be achieved when the diversity level is set appropriately. Specifically, 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 two main theoretical results addressed in [23]. Before that, some notations and preliminaries on statistical learning theory are revisited.

Consider the hypothesis set

\[\mathcal{F} := \{x \mapsto \sum_{j=1}^L \beta_j g(\bar{w}_j^T x) \mid \|\beta\|_2 \leq B, \|\bar{w}_j\|_2 \leq C, \forall i \neq j, |\bar{w}_i^T \bar{w}_j| \leq \tau \|\bar{w}_i\|_2 \|\bar{w}_j\|_2\},\]

where $\beta$ stands for the output weight $g(t) = 1/(1 + e^{-t})$ is the sigmoid activation function. Given training samples
\{(x_i, y_i)\}_{i=1}^{N} \) generated independently from an unknown distribution \( \mathcal{P}_{XY} \). The generalization error of \( f \in \mathcal{F} \) is defined as \( R(f) = \mathbb{E}_{\mathcal{P}_{XY}}[\frac{1}{2}(f(x) - y)^2] \). As \( \mathcal{P}_{XY} \) is not available, one can only consider minimizing the empirical risk \( \hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2 \) 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 \( R(\hat{f}) - R(f^*) \) 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 \( \tau \).

**Theorem 2** [23] (Estimation Error). With probability at least \( 1 - \delta \), the estimation upper bound of estimation error decreases as \( \tau \) becomes smaller, i.e.,
\[
R(\hat{f}) - R(f^*) \leq \frac{2 \sqrt{2 \ln(4/\delta)} + \gamma B(2C + 4|g(0)|) \sqrt{m}}{\sqrt{N}},
\]
where \( \gamma = 1 + BC \cdot \sqrt{(m-1)\tau} + 1/4 + \sqrt{m}B|g(0)| \).

Suppose the target function \( G = E[y|x] \) satisfy certain smoothness condition given by \( f \parallel \omega \parallel_2 |G(\omega)|d\omega \leq B/2 \), where \( G(\omega) \) 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** [23] (Approximation Error). A smaller \( \tau \) contributes to a larger upper bound of approximation error, that is, let \( C > 1 \) and \( L \leq 2(\frac{\pi(2-\theta)}{2\theta}) + 1 \), where \( \theta = \arccos(\pi) \), then there exists \( f \in \mathcal{F} \) such that
\[
\|f - G\|_2 \leq B(\frac{1}{\sqrt{m}} + \frac{1 + 2 \ln C}{C} + 2\sqrt{LBC} \min\{2\frac{L\theta}{\pi}, \tau\}).
\]

Based on Theorem 2 and Theorem 3, we can come to a conclusion, that is, a larger upper bound of generalization error can be caused by the case when the weight vectors are highly near-orthogonal with each other (\( \tau \) is extremely small) or the situation that \( \tau \) is close or equal to 1 (e.g., there exist two weight vectors that are linearly dependent). Therefore, given 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 [18] reveals a key pitfall of RVFL networks that all high-dimensional data-independent random features are nearly orthogonal to each other with probability one. Fortunately, the supervisory mechanism used in 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 leaner models produced by SCN and 2DSCN are more prone to result in a better generalization performance than RVFL and 2DRVFL. It would be interesting to organize rigourous theoretical analysis and extensive empirical study on differing the SCN framework from RVFL networks from this point of view. Also, it is meaningful to think about weight vector angular constraints in the development of SCNs, for the purpose of the enhancement of generalization. To avoid losing keynote for this work, we leave these useful explorations to our future research.

**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 [24], [25]. DropConnect proposed by Wan et al. [16] is the extension of Dropout in which each connection, instead of each output unit, can be dropped with 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, differs in that the sparsity-based concerns are imposed on the hidden input weights, rather than on the output vectors of a layer. That 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 [16], 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 RVFL and 2DRVFL method even when they use sparse weights in the hidden layer.

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

- 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 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 similar to 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 considerable degree of sparsity, have a good potential to inherit some merits of DropConnect and its parallel methodology. Fig. [3] highlights the characteristics of DropConnect, and provides a vivid demonstration of our logic why 2DSCN differs from SCN in exhibiting sparsity among the hidden input weights.

**ST4: Novel Estimation of Test Error.** Various statistical convergence rates for neural networks have been established when some constrains on the weights are concerned [26]–[29]. Empirically, small weights together with small training error can lead to significant improvements in generalization. All these investigations lend scientific supports to the heuristic techniques like weight decay and early stopping. The reason behind 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 penalty such as L2 weight de-
A comprehensive overview of existing theories/techniques concerning learner models’s generalization capability is out of our focus in this paper. Instead, we revisit the theoretical result presented in [34], and illustrate mathematically how the output weights magnitudes affect 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 [72], that is,

**Theorem 4** [34]. Consider the hypothesis set $\mathcal{F}_p := \{ f : f = \int \alpha(w)g(w;w)dw | |\alpha(w)| \leq Bp(w) \}$ with certain distribution $p$ and function $g$ satisfying $\sup_{w,w} g(x; w) \leq 1$, and given a training data set with $N$ input-output pairs drawn iid from some distribution $P_{XY}$, a randomized learner model $f(x) = \sum_{j=1}^{L} g(x; w_i)$ can be obtained by randomly assigning $w_1, w_2, \ldots, w_L$ from the distribution $p$ and solving the empirical risk minimization problem $\min_{\beta} \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - y_i)^2$ subject to $\|\beta\|_\infty \leq B/L$. Then, with probability at least $1 - 2\delta$, the upper bound for the generalization error of $f$ can be estimated by

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

Theoretically, the upper bound in Theorem 3 implies that randomized learner models with good training result and small output weights can probably lead to preferable generalization performance, in terms of probability perspective. However, this cannot be used directly to bound the practical test error for evaluating the randomized learner models’s generalization performance. More numerical estimation for the test error (resulted 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., $H : \mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times L}$, denoted by (see [35] for basic fundamentals on matrix calculus)

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

with the argument $X$ represented by

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

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

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

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

$$\nabla H_{i,j}(X) = \begin{pmatrix} \frac{\partial g(w_1^T x_i + b_1)}{\partial x_{1,1}} & \cdots & \frac{\partial g(w_L^T x_i + b_L)}{\partial x_{1,d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial g(w_1^T x_i + b_1)}{\partial x_{N,1}} & \cdots & \frac{\partial g(w_L^T x_i + b_L)}{\partial x_{N,d}} \end{pmatrix}.$$
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 $\|d \hat{H}(X)\|_F^2$. In particular, we use 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 $d \hat{H}(X)$ can be expressed as

$$\text{tr}(\nabla H_{i,j}(X)^T Z)$$

where $Z$ is sufficiently small, $\nabla$ stand for the associated hidden layer output matrix, then, the test error can be bounded by

$$\|\hat{H} \beta - Y\|_F^2 \leq \|H \beta - Y\|_F^2 + \epsilon \max_{1 \leq i \leq N} \|Z_i\|_2^2 \cdot \|H \circ (O - H) \circ \hat{W}\|_F \|\beta\|_F + o(\epsilon^2)\|\beta\|_F.$$  \hfill (13)

**Remark 3.** We would like to highlight a trick concerned in the previous deduction for Theorem 5. Indeed, we have considered to preserve the bundle of computational units `$g((1 - g_i)j)w_j^2$ rather than to roughly estimate the whole term by `$((1 - g_i)j)w_j^2$`, which consequently can result in a very blunt bound $\frac{1}{2}\|W\|_F \|Z\|_F$ for $\|d \hat{H}(X)\|_F$. Unfortunately, upper bound $\frac{1}{2}\|W\|_F \|Z\|_F$ sounds meaningless because it does not consider the saturation property of sigmoid function, and may cause some misleading 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’ generalization power. It is the bundle of computational units $\|H \circ (O - H) \circ \hat{W}\|_F$ rather than merely the $\|W\|_F$ that acts as a suitable indicator for predicting the generalization performance. Besides, 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 sigmoid function), are more likely to result in a small value of $\|H \circ (O - H) \circ \hat{W}\|_F$ 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 $\epsilon \max_{1 \leq i \leq N} \|Z_i\|_2^2 \cdot \|H \circ (O - H) \circ \hat{W}\|_F$ as the regularization factor $\sigma > 0$, that is, $\|H \beta - Y\|_F^2 + \sigma\|\beta\|_F$, 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 certain positive influence on enhancing a learner model’s generalization ability, one major issue still left unclarified is that whether or not 2D randomized learner models possess this advantage. For that purpose and before ending this section, we would like to 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 distribution $P$ (either uniform or gaussian), we investigate the statistical differences among the following three strategies for randomly assign parameters:

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

A simple and vivid demonstration for the distribution of the random weights induced by M1 and M3 is provided in Fig. 4 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\(\sim\)M3.

![Fig. 4. Simple illustration for the distribution of 1D and 2D random weights: (a) Values of \( w_{2D} \) generated by M3 with \( d_1 = d_2 = 28 \); (b) Values of \( w_i \) generated by M1 with \( d = 784 \). Both \( w_{2D} \) and \( w_i \) are reshaped into \( 28 \times 28 \) for visualization.](image)

Our primary objective is to find out how frequently these strategies contribute to a high dimensional random vector \((w, w^{(1D-P)}, \text{or} w^{2D})\) with considerable amount of elements whose values are close to zero. Here we present a theoretical result for answering this question and then provide an empirical verification in 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

\[
P \left( \frac{\# \{ i : |w_i^{2D}| \leq \epsilon \} }{d} \geq p \right) \geq P \left( \frac{\# \{ i : |w_i^{(1D-P)}| \leq \epsilon \} }{d} \geq p \right) \geq P \left( \frac{\# \{ i : |w_i| \leq \epsilon \} }{d} \geq p \right), \tag{14} \]

where \( \epsilon \) 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 equal/lower than \( \epsilon \). On the other hand, \( p \) is a threshold by which we can study the normalized percentage \( \# \{ \cdot \} / d \in [0, 1] \) when interested elements occur in that random vector \((w, w^{(1D-P)}, \text{or} w^{2D})\).

Instead of making efforts to give a rigorous mathematical proof, here we focus on a empirical study to verify the inequalities (14) in 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 \((p, \epsilon)\) 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 \( z_2 \sim U[-1, 1] \), the probability density function (p.d.f) of their product \( z = z_1 z_2 \) can be expressed by

\[
p(x) = \begin{cases} \frac{1}{2} \ln x, & 0 < x \leq 1 \\ \frac{1}{2} \ln(-x), & -1 \leq x < 0 \end{cases}
\]

In the simulation, we conduct 100,000 independent trials for randomly assigning \( w, w^{(1D-P)}, \text{and} w^{2D} \) according to M1 M3, respectively. Later we can count the number of times (denoted by \( M \)) when the condition \#\{\cdot\} / d \geq p \) are satisfied, followed by roughly estimating the true probability in (14) with \( \hat{P} = M / 100000 \). In Table I we list the corresponding \( \hat{P} \) values (arranged in order \( w^{2D} / w^{(1D-P)} / w \)) for the cases with different settings of \( \epsilon \) (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.

| 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 \( \hat{P} \) 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.

| 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 | 0.0074/0/0 |
| \( p = 15\% \) | 0/0/0 | 2.0e-5/0/0 | 9.2e-4/0/0 |

IV. Performance Evaluation

In this section, we demonstrate the advantages of 2DSCNs in image data modelling tasks, compared with some baseline/randomized learning methods including SCN, RVFL and...
2DSCN networks. Both regression and classification problems with image inputs are examined in the simulation. Datasets description, experimental setup, results and discussions are detailed for each task, as introduced in the following.

A. Regression: Rotation Angles Predication for Handwritten Digits

We first demonstrate the merits of the proposed 2DSCN by predicting the angles of rotation of handwritten digits. In particular, 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). For example, 16 random samples are displayed in Fig. 5. Apparently, baseline approaches such as RVFL and SCN deploy one dimensional input (reshaping the image into a vector) in modelling, while 2DSCN and 2DRVFL 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] and [−1, 1] respectively, while the biases for them are randomly assigned from [−1, 1]. For SCN and 2DSCN, we take $T_{max} = 5$, $\lambda = \{1, 5, 15, 30, 50, 100, 150, 200, 250\}$, $r = \{1 - 10^{-j}\}_{j=2}$ in the algorithm implementation (see Section 3 for the functionality of these parameters).

![Fig. 5. Samples of Handwritten Digits with Angles of Rotation.](image)

Two types of evaluation metrics are used in 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 $\theta$ (in degrees) is needed to measure 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 $\theta$ from the true angles. Mathematically, the PPA value within threshold $\theta$ can be obtained by

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

Table III shows both training and test results for these four algorithms ($L = 1800$) in terms of PPA and RMSE, with threshold $\theta$ set to 15 and 20, respectively. It is observed that both SCN and 2DSCN outperform RVFL-based algorithms, and 2DSCN shows the highest PPA and lowest RMSE values, which reflects a better learning and generalization capability. Specifically, given $\theta = 25$, 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 simple calculation according to PPA values and the number of training/test samples, we can immediately notice that 2DSCN produces some training errors with absolute values larger than 25 degrees while 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 led 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. To fully demonstrate the difference between 2DSCN and 2DRVFL on residual distribution for test images (for each digit class),
we plot the corresponding box-and-whisker diagram in Fig. 6. It is obvious that 2DSCN works much more favorably than 2DRVFL for almost each digit class as there are less records with abnormal degree error marked in red plus symbol. 2DSCN shows better stability on prediction for every digit as there are less errors outside the interval $[-40, 40]$ than that of 2DRVFL. One can easily observe that this finding also corresponds what we have presented in Table III. In particular, the learner model produced by 2DRVFL is more prone to result in larger predictive errors (with absolute values close/higher than 50) for digit ‘4’ and ‘7’. Obviously, 2DSCN can contribute to a slightly better rotation correction than SCN for each setting of the learner model architecture, while they both outperform RVFL and 2DRVFL.

### B. Classification: Handwritten Digits Recognition

In this part, we compare our proposed 2DSCN algorithm with the other three randomized approaches on image classification problem. Four benchmark handwritten digits databases are employed in the comparison. Parameter setting for these four algorithms $(w, b, u, v, \lambda, T_{\text{max}}, r)$ are 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\(^3\)) are employed in our experimental study as a benchmark resource for pattern recognition competitions\(^4\). CVL usually used for pattern recognition competitions\(^5\) are employed in our experimental study as a benchmark resource for optical character recognition. Readers can refer to our previous work\(^12\) for more descriptions about these datasets. Here we summarize the basic information about these four databases in Table IV. It should be noted that all the images of these four databases have been converted to grayscale and normalized size $(28\times28)$ by applying the similar preprocessing procedures used in MNIST database\(^6\), that is, scaled to 20x20 by preserving the aspect ratio and then the centroid of the image is placed on the center of a standard plane of size $28\times28$. Some sample images are shown in Fig. 8.

**TABLE IV**

| Name   | Number of Images | Training | Test  |
|--------|------------------|----------|-------|
| Bangla | 23392            | 19392    | 4000  |
| Devnagari | 22556          | 18794    | 3762  |
| Oriya  | 5970             | 4970     | 1000  |
| CVL    | 35780            | 14000    | 21780 |

1) Results and Discussion: Fig. 5 displays performance comparison for both training and test in terms of different setting of $L$, in which the mean and standard deviation values of recognition rate are based on 10 independent trials (More trails can indicate more convinced results but not necessary based on our experience, because the standard deviation values are relatively small and stable at certain level). It is apparent that 2DSCN outperforms the others while RVFL has the worst results in all these four datasets. Interestingly, the test results of 2DSCN are much better than that of SCN, even when their training recognition rate is relatively close. Similar finding can be noticed among the comparison between 2DRVFL and SCN. For example, for Bangla, Devnagari, and Oriya, 2DRVFL occasionally can result in slightly higher recognition rates in testing even the corresponding training recognition rate is lower/quite close to that of SCN. It becomes much more clear in the subplots for CVL dataset, i.e., 2DRVFL works more favorably than SCN but still worse than 2DSCN, which to some extent lends strong support for our theoretical investigation on the superiority of randomized learners with 2D inputs.

2) Results and Discussion: In our experiments, we employ the same parameter setting for these four algorithms (see part A for details), and conduct the performance comparison...  

C. Case Study: Human Face Recognition

In this section, 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 addressed before. Followed by a brief description of the used databases, we present the performance comparison for these four methods. Later, as a verification for Theorem 5, we calculate the corresponding test error upper bounds for the randomized models produced by those four algorithms, and visualize their differences for both datasets. In addition, we compare the capabilities of those four algorithms in dealing with training images corrupted by certain level of random noises, aiming to show their robustness in problem-solving. Generally, our experimental results can strongly support the superiority of 2DSCN algorithm on image data modelling problems.

1) Databases: Two benchmark datasets for human face recognition problem are employed in our experiments, as we introduce in the following. ORL\(^37\): The Olivetti (ORL, now AT&T) database contains ten $112 \times 92$ pixel gray scale images of 40 distinct subjects (individuals). Among those subjects, some 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). Specifically, 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\(^38\): 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 over three years. In particular, 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 $112 \times 92$ per person.

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

2) Results and Discussion: In our experiments, we employ the same parameter setting for these four algorithms (see part A for details), and conduct the performance comparison...
Fig. 8. Samples of handwritten digits from Bangla, Devnagari, Oriya, and CVL database.

Fig. 9. Performance comparison among 2DSCN, 2DRVFL, SCN, and RVFL on Bangla, Devnagari, Oriya, and CVL database.

Fig. 10. Samples of face images from ORL, FERET database. Ten subjects with two expressions for each are displayed.

with various setup 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. 11 shows their test recognition rates with both mean and standard deviation values. Clearly and similar to what we have demonstrated before, 2DSCN well outperforms the other three algorithms in 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 setting exhibit sufficient learning capability, however, has significant discrepancy in generalization ability. Besides, as shown in 11, there is no big difference between the performance of 2DRVFL and that of SCN, and both of them contribute to a better result than RVFL. Now, it is fair to say that 2D models have more advantages than 1D models in both training and test, such as what we have observed in previous Part A and B. Based on Theorem 5 in Section III, 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 contain three parts, that is, training error $\|H\beta - Y\|_F$, $\epsilon \max_{1 \leq i \leq N} \|Z_i\|_2 \cdot \|H \circ (O - H) \circ \bar{W}\|_F$, and the sufficiently small term $o(\epsilon^2)\|\beta\|_F$, respectively. Since
their training errors stay in 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 $\max_{1 \leq i \leq N} \|Z_i\|_2$ in our empirical examination. In particular, for each method, we consider the case of $L = 600$ and calculate the value of $\max_i \left\{ \left\| H \circ (O - H) \circ \hat{W} \right\|_F \cdot \|\beta\|_F \right\}$ for 50 resulted leaner models based on independent trials, followed by a normalization operation via 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)

$$\Theta_i = \frac{\max_i \left\{ \left\| H \circ (O - H) \circ \hat{W} \right\|_F \cdot \|\beta\|_F \right\}}{\max_{1 \leq i \leq N} \|Z_i\|_2}$$

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

Fig. 12 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. Apparently and interestingly, this is consistent with their real test recognition rate comparison shown in Fig. 11, therefore, verify 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 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.

3) Robustness Illustration: To further investigate the superiority of 2DSCN over the other methods, we randomly select 50 images from 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). 30 of these corrupted images are display in Fig. 13. Our objective is to compare the test performance of these four algorithms and study their capability in performing robustness in training, that is, to what extent they can alleviate the impacts of random block occlusion attached in the training images. All details for experimental setup remain the same as what we have used for the original (clean) ORL dataset.

As can be seen in Fig. 14, 2DSCN still performs best in generalization, at the same time, RVFL has the lowest test recognition rates. This observation can to some extent imply that 2D models may still exhibit their merits for problem-solving of robust image data modelling, on the basis of their underlying advantages as shown throughout this paper. We expect more interesting work employing/extending the current 2DSCN framework to deal with robust image recognition tasks or similar applications in parallel. By saying that, we have to admit that the technical issue around is out of our main focus in this work and more interesting follow-up researches are desirable in the future.

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 on 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, and have great potential for real world applications.

There are many interesting studies left for future work. For example, a trivial extension of the current 2DSCN to robust version can be realized by an immediate combination of this work and our previous research [13]. It is also important to point out that, while this work focuses on shallow neural networks, the framework and the associated theoretical analysis are generic and general enough to adopt the deep machinery, therefore, one can make efforts towards building a deep 2DSCN to maintain both the superiority of DeepSCN [14] and the advantages of 2DSCN. It is also of practical importance to employ the proposed 2DSCN in stream image data modelling. Other than that, it would be interesting to further enhance 2DSCN by considering the regularization learning framework, or refined stochastic configuration inequality with sparsity constraints.

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. IEEE, 2013, pp. 1–6.
[3] N. Qi, Y. Shi, X. Sun, J. Wang, and W. Ding, “Two dimensional analysis sparse model,” in Proceedings of the 20th International Conference on Image Processing. IEEE, 2013, pp. 310–314.
[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. IEEE, 2017, pp. 808–815.
[9] S. Scardapani 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. Igelhik 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. 5, 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 M. Li, “Deep stochastic configuration networks with universal approximation property,” arXiv:1702.05639, to appear inIJCNN 2018.