Javid *et al. EURASIP Journal on Advances in Signal Processing* (2020) 2020:40
https://doi.org/10.1186/s13634-020-00695-2

EURASIP Journal on Advances in Signal Processing

RESEARCH **Open Access**

High-dimensional neural feature design for layer-wise reduction of training cost

Alireza M. Javid*, Arun Venkitaraman, Mikael Skoglund and Saikat Chatterjee

*Correspondence: almj@kth.se
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Brinellvägen 8, Stockholm, Sweden

Abstract

We design a rectified linear unit-based multilayer neural network by mapping the feature vectors to a higher dimensional space in every layer. We design the weight matrices in every layer to ensure a reduction of the training cost as the number of layers increases. Linear projection to the target in the higher dimensional space leads to a lower training cost if a convex cost is minimized. An ℓ_2-norm convex constraint is used in the minimization to reduce the generalization error and avoid overfitting. The regularization hyperparameters of the network are derived analytically to guarantee a monotonic decrement of the training cost, and therefore, it eliminates the need for cross-validation to find the regularization hyperparameter in each layer. We show that the proposed architecture is norm-preserving and provides an invertible feature vector and, therefore, can be used to reduce the training cost of any other learning method which employs linear projection to estimate the target.

Keywords: Rectified linear unit, Feature design, Neural network, Convex cost function

1 Introduction

Nonlinear mapping of low-dimensional signal to high-dimensional space is a traditional method for constructing useful feature vectors, specifically for classification problems. The intuition is that, by extending to a high dimension, the feature vectors of different classes become easily separable by a linear classifier. The drawback of performing classification in a higher dimensional space is the increased computational complexity. This issue can be handled by a well-known method called “kernel trick” in which the complexity depends only on the inner products in the high-dimensional space. Support vector machine (SVM) [1] and kernel PCA (KPCA) [2] are examples of creating high-dimensional features by employing the kernel trick. The choice of the kernel function is a critical aspect that can affect the classification performance in the higher dimensional space. A popular kernel is the radial basis function (RBF) kernel or Gaussian kernel, and its good performance is justified by its ability to map the feature vector to a very high, infinite, dimensional space [3]. In this manuscript, we design a high-dimensional feature using an artificial neural network (ANN) architecture to achieve a better classification performance by increasing the number of layers. The architecture uses the rectified linear unit

(ReLU) activation, predetermined orthonormal matrices, and a fixed structured matrix. We refer to this as high-dimensional neural feature (HNF) throughout the manuscript.

Neural networks and deep learning architectures have received overwhelming attention over the last decade [4]. Appropriately trained neural networks have been shown to outperform the traditional methods in different applications, for example, in classification and regression tasks [5, 6]. By the continually increasing computational power, the field of machine learning is being enriched with active research pushing classification performance to higher levels for several challenging datasets [7–9]. However, very little is known regarding how many numbers of neurons and layers are required in a network to achieve better performance. Usually, some rule-of-thumb methods are used for determining the number of neurons and layers in an ANN, or an exhaustive search is employed which is extremely time-consuming [10]. In particular, the technical issue—guaranteeing performance improvement with increasing the number of layers—is not straightforward in traditional neural network architectures, e.g., deep neural network (DNN) [11], convolutional neural network (CNN) [12], and recurrent neural network (RNN) [13]. We endeavor to address this technical issue by mapping the feature vectors to a higher dimensional space using predefined weight matrices.

There exist several works employing predefined weight matrices that do not need to be learned. Scattering convolution network [14] is a famous example of these approaches which employs wavelet-based scattering transform to design the weight matrices. Random matrices have also been widely used as a mean for reducing the computational complexity of neural networks while achieving comparable performance as with fully learned networks [15–18]. In the case of the simple, yet effective, extreme learning machine (ELM), the first layer of the network is assigned randomly chosen weights and the learning takes place only at the end layer [19–22]. It has also been shown recently that a similar performance to fully learned networks may be achieved by training a network with most of the weights assigned randomly and only a small fraction of them being updated throughout the layers [23]. It has been shown that networks with Gaussian random weights provide a distance-preserving embedding of the input data [16]. The recent work [18] designs a deep neural network architecture called progressive learning network (PLN) which guarantees the reduction of the training cost with increasing the number of layers. In PLN, every layer is comprised of a predefined random part and a projection part which is trained individually using a convex cost function. These approaches indicate that randomness has much potential in terms of high performance at low computational complexity. We design a multilayer neural network using predefined orthonormal matrices, e.g., random orthonormal matrix and DCT matrix, to ensure reducing the training cost as the number of layers increases.

1.1 Our contributions

Motivated by the prior use of fixed matrices, we design the HNF architecture using an appropriate combination of ReLU, random matrices, and fixed matrices. We use predefined weight matrices in every layer of the network, and therefore, the architecture does not suffer from the infamous vanishing gradient problem. We theoretically show that the output of each layer provides a richer representation compared to the previous layers if a convex cost is minimized to estimate the target. We use an ℓ_2-norm convex constraint to reduce the generalization error and avoid overfitting to the training data. We analytically derive the regularization hyperparameter to ensure the decrement of the training cost in each layer. Therefore, there is no need for cross-validation to find the optimum regularization hyperparameters of the network. We show that the proposed HNF is norm-preserving and invertible and, therefore, can be used to improve the performance of other learning methods that use linear projection to estimate the target. Finally, we show the classification performance of the proposed HNF against ELM and state-of-the-art results. Note that a preliminary version of this manuscript has been submitted to ICASSP 2020 recently.

1.2 Notations

We use the following notations unless otherwise noted: We use bold capital letters, e.g., $\mathbf{W}$, to denote matrices and bold lowercase letters, e.g., $\mathbf{x}$, to denote vectors. We use calligraphic letter $\mathcal{M}$ to denote a set and $\mathcal{M}^c$ to denote compliment set. The cardinality of a set $\mathcal{M}$ is denoted by $|\mathcal{M}|$. For a scalar $x \in \mathbb{R}$, let us denote its sign and magnitude as $s(x) \in \{-1, +1\}$ and $|x|$, respectively, and write $x = s(x)|x|$. For a vector $\mathbf{x}$, we define the sign vector $\mathbf{s}(\mathbf{x})$ and magnitude vector $|\mathbf{x}|$ by the element-wise operation. We define $\mathbf{g}(\cdot)$ as a nonlinear function comprised of a stack of element-wise ReLU activation functions. A vector $\mathbf{x}$ has non-negative part $\mathbf{x}^+$ and non-positive part $\mathbf{x}^-$ such that $\mathbf{x} = \mathbf{x}^+ + \mathbf{x}^-$ and $\mathbf{g}(\mathbf{x}) = \mathbf{x}^+$. We use $\|\cdot\|$ and $\|\cdot\|_F$ to denote ℓ_2-norm and Frobenius norm, respectively. For example, it can be seen that $\|\mathbf{x}\|^2 = \|\mathbf{x}^+\|^2 + \|\mathbf{x}^-\|^2$.

2 Proposed method

In this section, we illustrate the motivation to design a high-dimensional feature vector by using ReLU activation function. We analyze the behavior of a single layer ReLU network to the input perturbation noise and show that by mapping the feature vectors to a higher dimension, we can increase the discrimination power of the ReLU network.

For an ANN, we wish to have noise robustness and discriminative power. We characterize this in the following definition.

Definition 1 (Noise Robustness and Point Discrimination) *Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two input vectors such that $\mathbf{x}_1 \neq \mathbf{x}_2$, and we have outputs of ANN $\tilde{\mathbf{t}}_1 = \mathbf{f}(\mathbf{x}_1)$ and $\tilde{\mathbf{t}}_2 = \mathbf{f}(\mathbf{x}_2)$. We can characterize a perturbation scenario with the perturbation noise Δ as $\mathbf{x}_2 = \mathbf{x}_1 + \Delta$. We wish that the proposed ANN holds the property*

$$c_1\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \leq \|\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2)\|^2 \leq c_2\|\mathbf{x}_1 - \mathbf{x}_2\|^2, \tag{1}$$

where $0 < c_1 \leq 1$ and $c_1 \leq c_2$.

Note that the lower bound provides point discrimination power and the upper bound provides noise robustness to the input.

2.1 Layer construction

We first concentrate on one block of ANN—this is called a layer in the neural network literature. The layer has an input vector $\mathbf{q} \in \mathbb{R}^{m\times 1}$ and the output vector $\mathbf{y} = \mathbf{g}(\mathbf{W}\mathbf{q})$. The dimension of $\mathbf{y}$ is the number of neurons in the layer. If we can guarantee that the layer of ANN provides noise robustness and point discrimination property, then the full

ANN comprising of multiple layers connected sequentially can be guaranteed to hold robustness and discriminative properties. We need to construct $\mathbf{W}$ in such a manner that the layer has noise robustness and discriminative power according to Definition 1.

2.2 ReLU activation and a limitation

We first show three essential properties of ReLU function, required to develop our main results. We then discuss one possible limitation of the ReLU function and propose a remedy to circumvent the problem.

Property 1 (Scaling) *ReLU function has a scaling property. If* $\mathbf{y} = \mathbf{g}(\mathbf{W}\mathbf{q})$, *then* $a\mathbf{y} = \mathbf{g}(\mathbf{W}(a\mathbf{q}))$ *for a scalar* $a \geq 0$.

Property 2 (Sparsity) *ReLU function provides sparse output vector* $\mathbf{y}$ *such that* $\|\mathbf{y}\|_0 \leq dim(\mathbf{y})$.

Property 3 (Noise Robustness) *Let us consider* $\mathbf{z} = \mathbf{W}\mathbf{q}$. *For two vectors* $\mathbf{q}_1$ *and* $\mathbf{q}_2$, *we define corresponding vectors* $\mathbf{z}_1 = \mathbf{W}\mathbf{q}_1$ *and* $\mathbf{z}_2 = \mathbf{W}\mathbf{q}_2$, *and output vectors* $\mathbf{y}_1 = \mathbf{g}(\mathbf{z}_1) = \mathbf{g}(\mathbf{W}\mathbf{q}_1)$ *and* $\mathbf{y}_2 = \mathbf{g}(\mathbf{z}_2) = \mathbf{g}(\mathbf{W}\mathbf{q}_2)$. *Now, we have the following relation*

$$0 \leq \|\mathbf{y}_1 - \mathbf{y}_2\|^2 = \|\mathbf{g}(\mathbf{z}_1) - \mathbf{g}(\mathbf{z}_2)\|^2 \leq \|\mathbf{z}_1 - \mathbf{z}_2\|^2. \tag{2}$$

The proof of Property 3 is shown in Appendix 1. The upper bound relation holds Lipschitz continuity that provides noise robustness. On the other hand, the lower bound being zero cannot maintain a minimum distance between two points $\mathbf{y}_1$ and $\mathbf{y}_2$. An example of extreme effect is that when $\mathbf{z}_1$ and $\mathbf{z}_2$ are non-positive vectors, we get $\|\mathbf{y}_1 - \mathbf{y}_2\|^2 = 0$. This may limit the capacity of the ReLU function for achieving a good discriminative power. A reason for the limitation "lower bound being zero" is due to the structure of the input matrix $\mathbf{W}$. We build an appropriate structure for the input matrix to circumvent the limitation.

We now engineer a remedy for this limitation. Let us consider $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}\mathbf{z}) = \mathbf{g}(\mathbf{V}\mathbf{W}\mathbf{q})$, where $\mathbf{z} = \mathbf{W}\mathbf{q} \in \mathbb{R}^n$ and $\mathbf{V}$ is a linear transform matrix. For two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, we have corresponding vectors $\mathbf{z}_1 = \mathbf{W}\mathbf{q}_1$ and $\mathbf{z}_2 = \mathbf{W}\mathbf{q}_2$, and output vectors $\bar{\mathbf{y}}_1 = \mathbf{g}(\mathbf{V}\mathbf{z}_1)$ and $\bar{\mathbf{y}}_2 = \mathbf{g}(\mathbf{V}\mathbf{z}_2)$. Our interest is to show that there exists a predefined matrix $\mathbf{V}$ for which we have both noise robustness and discriminative power properties.

Proposition 1 *Let us construct a* $\mathbf{V}$ *matrix as follows*

$$\mathbf{V} = \begin{bmatrix} \mathbf{I}_n \\ -\mathbf{I}_n \end{bmatrix} \triangleq \mathbf{V}_n. \tag{3}$$

For the output vectors $\bar{\mathbf{y}}_1 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_1) \in \mathbb{R}^{2n}$ *and* $\bar{\mathbf{y}}_2 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_2) \in \mathbb{R}^{2n}$, *we have* $\|\bar{\mathbf{y}}_1\|^2 = \|\mathbf{z}_1\|^2$ *and* $\|\bar{\mathbf{y}}_2\|^2 = \|\mathbf{z}_2\|^2$ *and*

$$0 < \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 \leq \|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 \leq \|\mathbf{z}_1 - \mathbf{z}_2\|^2. \tag{4}$$

The proof of the above proposition can be found in Appendix 2. Based on the above proposition, we can interpret the effect of noise passing through such layer. Let $\mathbf{z}_2 =$

$\mathbf{z}_1 + \Delta\mathbf{z}$, where $\Delta\mathbf{z}$ is a small perturbation noise. Note that $\Delta\mathbf{z} = \mathbf{z}_1 - \mathbf{z}_2 = \mathbf{W}[\,\mathbf{q}_1 - \mathbf{q}_2] = \mathbf{W}\,\Delta\mathbf{q}$. To investigate effect of perturbation noise, we now state our main assumption.

Assumption 1 *Given a small $\|\Delta\mathbf{z}\|^2$, the sign patterns of $\mathbf{z}_1$ and $\mathbf{z}_2$ do not differ significantly. On the other hand, for a large perturbation noise $\|\Delta\mathbf{z}\|^2$, the sign patterns of $\mathbf{z}_1$ and $\mathbf{z}_2$ vary significantly.*

The above assumption means that for a small $\|\Delta\mathbf{z}\|^2$, the set $\mathcal{M}(\mathbf{z}_1, \mathbf{z}_2) = \{i | s(z_1(i)) = s(z_2(i)) \neq 0\}$ is close to a full set and $\mathcal{M}^c(\mathbf{z}_1, \mathbf{z}_2)$ is close to an empty set. On the other hand, for a large $\|\Delta\mathbf{z}\|^2$, the set $\mathcal{M}(\mathbf{z}_1, \mathbf{z}_2) = \{i | s(z_1(i)) = s(z_2(i)) \neq 0\}$ is close to an empty set and $\mathcal{M}^c(\mathbf{z}_1, \mathbf{z}_2)$ is close to a full set. Considering Assumption 1, we can present the following remark regarding the effect of noise in the layer.

Remark 1 (Effect of perturbation noise) *For a small perturbation noise $\|\Delta\mathbf{z}\|^2$, we have $\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 \approx \|\mathbf{z}_1 - \mathbf{z}_2\|^2$. On the other hand, a large perturbation noise is attenuated.*

This follows from the proof of Proposition 1, specifically Eqs. (28) and (29a). In fact, if $\mathcal{M}^c = \emptyset$, then $\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 = \|\mathbf{z}_1 - \mathbf{z}_2\|^2$. We interpret that a small perturbation noise passes through the single layer $\mathbf{g}(\mathbf{V}\mathbf{z})$ almost not attenuated. Let us construct an illustrative example. Assume that $\mathcal{M}^c(\mathbf{z}_1, \mathbf{z}_2)$ is a full set and $\forall i \in \mathcal{M}^c,\ |z_1(i)| = |z_2(i)|$. In that case, $\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 = 0.5\|\mathbf{z}_1 - \mathbf{z}_2\|^2$ and we can comment that the perturbation noise is attenuated.

3 High-dimensional neural feature

In this section, we employ the proposed weight matrix in (3) to construct a multilayer ANN. We show that by designing the weight matrices in every layer, it is possible to construct a network that provides noise robustness and point discrimination according to Definition 1.

Let us establish the relation between the input vector $\mathbf{q} \in \mathbb{R}^m$ and output vector $\bar{\mathbf{y}} \in \mathbb{R}^{2n}$. For two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, we have corresponding vectors $\mathbf{z}_1 = \mathbf{W}\mathbf{q}_1$ and $\mathbf{z}_2 = \mathbf{W}\mathbf{q}_2$, and output vectors $\bar{\mathbf{y}}_1 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_1) = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{q}_1)$ and $\bar{\mathbf{y}}_2 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_2) = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{q}_2)$. Our interest is to show that it is possible to construct a $\mathbf{W} \in \mathbb{R}^{n\times m}$ matrix for which we have both noise robustness and discriminative power properties. We can construct $\mathbf{W} \in \mathbb{R}^{n\times m}$ as orthonormal matrix, such that $n \geq m$ and $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_m$. In that case, we have $\|\mathbf{q}_1 - \mathbf{q}_2\|^2 = \|\mathbf{z}_1 - \mathbf{z}_2\|^2$ for any pair of $(\mathbf{q}_1, \mathbf{q}_2)$. By combining the this relation with Eq. (4), we conclude the following proposition.

Proposition 2 *Consider the single layer network $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{z}) = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{q})$ where $\mathbf{W} \in \mathbb{R}^{n\times m}$ is an orthonormal matrix, such that $n \geq m$ and $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_m$. Then, $\|\bar{\mathbf{y}}\|^2 = \|\mathbf{q}\|^2$, and for every two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, the following inequality holds*

$$\frac{1}{2}\|\mathbf{q}_1 - \mathbf{q}_2\|^2 \leq \|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 \leq \|\mathbf{q}_1 - \mathbf{q}_2\|^2. \tag{5}$$

The above proposition shows that by designing the weight matrix in a single layer network, it is possible to provide point discrimination and noise robustness according to Definition 1. Note that the weight matrix $\mathbf{W}$ can be any orthonormal matrix such as instances of random orthonormal matrix and DCT matrix. By considering the relation $\Delta\mathbf{z} = \mathbf{W}\,\Delta\mathbf{q}$, we can present a similar argument as in Remark 1. We interpret that a small perturbation noise $\|\Delta\mathbf{q}\|^2$ passes through the single layer $\mathbf{g}(\mathbf{VWq})$ almost not attenuated. This is stated in the following remark.

Remark 2 (Effect of perturbation noise) *For a small* $\|\Delta\mathbf{q}\|^2$*, we have* $\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 \approx \|\mathbf{q}_1 - \mathbf{q}_2\|^2$*. On the other hand, a large perturbation noise is attenuated.*

By directly using Proposition 1, we can present a similar bound in regard to the perturbation of the weight matrix $\mathbf{W}$ in a single layer construction. We can show that the perturbation norm in the output due to the perturbation to the weight matrix has an upper bound that is a scaled version of the input norm. The scaling parameter $\|\Delta\mathbf{W}\|_F^2$ is small for a small perturbation. The following remark illustrates this point in detail.

Remark 3 (Sensitivity to the weight matrix) *Let the weight matrix* $\mathbf{W}$ *be perturbed by* $\Delta\mathbf{W}$*. The effective weight matrix is* $\mathbf{W} + \Delta\mathbf{W}$*. For an input* $\mathbf{q}$ *and the respective outputs* $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{q})$ *and* $\bar{\mathbf{y}}_\Delta = \mathbf{g}(\mathbf{V}_n[\,\mathbf{W} + \Delta\mathbf{W}\,]\,\mathbf{q})$*, we have*

$$\|\bar{\mathbf{y}} - \bar{\mathbf{y}}_\Delta\|^2 \leq \|\Delta\mathbf{W}\|_F^2\|\mathbf{q}\|^2. \tag{6}$$

The proof can be found in Appendix 3.

3.1 Multilayer construction

A feedforward ANN is comprised of similar operational layers in a chain. Let us consider two layers in feedforward connection, e.g., l-th and $(l+1)$-th layers of an ANN. For the l-th layer, we use a superscript (l) to denote appropriate variables and parameters. Let the l-th layer has $m^{(l)}$ nodes. The input to the lth layer is $\mathbf{q}^{(l)} = \bar{\mathbf{y}}^{(l-1)}$. The output of l-th layer $\bar{\mathbf{y}}^{(l)} = \mathbf{g}\left(\mathbf{V}_{n^{(l)}}\mathbf{z}^{(l)}\right) = \mathbf{g}\left(\mathbf{V}_{n^{(l)}}\mathbf{W}^{(l)}\mathbf{q}^{(l)}\right)$ is next used as the input to the $(l+1)$-th layer, that means $\bar{\mathbf{y}}^{(l)} = \mathbf{q}^{(l+1)}$. Thus, the output of $(l+1)$-th layer is

$$\begin{aligned}\bar{\mathbf{y}}^{(l+1)} &= \mathbf{g}\left(\mathbf{V}_{n^{(l+1)}}\mathbf{z}^{(l+1)}\right)\\ &= \mathbf{g}\left(\mathbf{V}_{n^{(l+1)}}\mathbf{W}^{(l+1)}\mathbf{q}^{(l+1)}\right)\\ &= \mathbf{g}\left(\mathbf{V}_{n^{(l+1)}}\mathbf{W}^{(l+1)}\mathbf{g}\left(\mathbf{V}_{n^{(l)}}\mathbf{W}^{(l)}\mathbf{q}^{(l)}\right)\right)\end{aligned} \tag{7}$$

Now, for the two vectors $\mathbf{q}_1^{(l)}$ and $\mathbf{q}_2^{(l)}$, we have the following relations in l-layer based on Proposition 2

$$\frac{1}{2}\|\mathbf{q}_1^{(l)} - \mathbf{q}_2^{(l)}\|^2 \leq \|\bar{\mathbf{y}}_1^{(l)} - \bar{\mathbf{y}}_2^{(l)}\|^2 \leq \|\mathbf{q}_1^{(l)} - \mathbf{q}_2^{(l)}\|^2. \tag{8}$$

We present the above results as the following theorem to provide noise robustness and discrimination power properties of the proposed ANN and call it high-dimensional neural feature (HNF) afterwards.

Theorem 1 *The proposed HNF uses ReLU activation function and is constructed as follows:*

(a) *The HNF is comprised of L-layers where the l-th layer has the corresponding structure* $\bar{\mathbf{y}}^{(l)} = \mathbf{g}(\mathbf{V}_{n^{(l)}}\mathbf{W}^{(l)}\bar{\mathbf{y}}^{(l-1)})$*. The L-layers are in a chain. The input to the*

first layer is $\mathbf{q}^{(1)} = \mathbf{x}$*. The output of HNF is*

$$\bar{\mathbf{y}}^{(L)} = \mathbf{g}(\mathbf{V}_{n^{(L)}}\mathbf{W}^{(L)}\mathbf{g}(\ldots\mathbf{g}(\mathbf{V}_{n^{(1)}}\mathbf{W}^{(1)}\mathbf{x}))).$$

(b) *In the HNF,* $\mathbf{W}^{(l)} \in \mathbb{R}^{n^{(l)}\times m^{(l)}}$ *matrices are orthonormal matrices with appropriate sizes, that is* $n^{(l)} \geq m^{(l)}$ *and* $m^{(l)} = 2n^{(l-1)}$*.*

Then, $\|\bar{\mathbf{y}}^{(L)}\|^2 = \|\mathbf{x}\|^2$*, and the construted HNF provides noise robustness and discriminative power properties that are characterized by the following relation*

$$\frac{1}{2^L}\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \leq \|\bar{\mathbf{y}}_1^{(L)} - \bar{\mathbf{y}}_2^{(L)}\|^2 \leq \|\mathbf{x}_1 - \mathbf{x}_2\|^2, \tag{9}$$

where $\mathbf{x}_1 \in \mathbb{R}^{m^{(1)}}$ *and* $\mathbf{x}_2 \in \mathbb{R}^{m^{(1)}}$ *are two input vectors to the HNF and their corresponding outputs are* $\bar{\mathbf{y}}_1^{(L)}$ *and* $\bar{\mathbf{y}}_2^{(L)}$*, respectively.*

Note that a similar argument as in Remark 2 holds here as well. We interpret that a small perturbation noise passes through the multilayer structure almost not attenuated. On the other hand, a large perturbation noise is attenuated in every layer. Using Theorem 1, we follow similar arguments as in Remark 3 in regard to the perturbation of the weight matrices $\mathbf{W}^{(l)}$ in every layer of the HNF.

Remark 4 (Sensitivity to the weight matrix) *Consider a scenario where the weight matrix* $\mathbf{W}^{(l)}$ *is perturbed by* $\Delta\mathbf{W}^{(l)}$*. The effective weight matrix is* $\mathbf{W}^{(l)} + \Delta\mathbf{W}^{(l)}$*. We can show that*

$$\|\bar{\mathbf{y}}^{(L)} - \bar{\mathbf{y}}_{\Delta}^{(L)}\|^2 \leq \prod_{l=1}^{L}\|\Delta\mathbf{W}^{(L)}\|_F^2\|\mathbf{x}\|^2. \tag{10}$$

4 Reduction of training cost

In this section, we analyze the effectiveness of the weight matrix $\mathbf{V}$ in the sense of reducing the training cost. We show that the proposed HNF provides lower training costs as the number of layers increases. We also present how the proposed structure can be used to reduce the training cost of other learning methods which employ linear projection to the target.

Consider a dataset containing N samples of pairwise P-dimensional input data $\mathbf{x} \in \mathbb{R}^P$ and Q-dimensional target vector $\mathbf{t} \in \mathbb{R}^Q$ as $\mathcal{D} = \{(\mathbf{x}, \mathbf{t})\}$. Let us construct two single layer neural networks and compare effectiveness of their feature vectors. In one network, we construct the feature vector as $\mathbf{y} = \mathbf{g}(\mathbf{W}\mathbf{x})$, and in the other network, we build the feature vector $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{x})$. We use the same input vector $\mathbf{x}$, predetermined weight matrix $\mathbf{W} \in \mathbb{R}^{n\times P}$, and ReLU activation function $\mathbf{g}(\cdot)$ for both networks. However, in the second network, the effective weight matrix is $\mathbf{V}_n\mathbf{W}$ where $\mathbf{V}_n = \begin{bmatrix}\mathbf{I}_n \\ -\mathbf{I}_n\end{bmatrix} \in \mathbb{R}^{2n\times n}$ is fully predetermined. To predict the target, we use a linear projection of feature vector. Let the predicted target for the first network be $\mathbf{O}\mathbf{y}$, and the predicted target for the second network $\bar{\mathbf{O}}\bar{\mathbf{y}}$. Note that $\mathbf{O} \in \mathbb{R}^{Q\times n}$ and $\bar{\mathbf{O}} \in \mathbb{R}^{Q\times 2n}$. By using ℓ_2-norm regularization, we find optimal solutions for the following convex optimization problems.

$$\mathbf{O}^\star = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\mathbf{y}\|^2\right] \text{ s.t. } \|\mathbf{O}\|_F^2 \leq \epsilon, \tag{11a}$$

$$\bar{\mathbf{O}}^\star = \arg\min_{\bar{\mathbf{O}}} E\left[\|\mathbf{t} - \bar{\mathbf{O}}\bar{\mathbf{y}}\|^2\right] \text{ s.t. } \|\bar{\mathbf{O}}\|_F^2 \leq \epsilon, \tag{11b}$$

where the expectation operation is done by sample averaging over all N data points in the training dataset. The regularization parameter ϵ is the same for the two networks. By defining $\mathbf{z} \triangleq \mathbf{W}\mathbf{x}$, we have

$$\bar{\mathbf{y}} = \begin{bmatrix} \mathbf{z}^{+} \\ -\mathbf{z}^{-} \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ -\mathbf{z}^{-} \end{bmatrix}. \tag{12}$$

The above relation is due to the special structure of $\mathbf{V}_n$ and the use of ReLU activation $\mathbf{g}(\cdot)$. Note that the solution $\bar{\mathbf{O}}^{\star} = [\,\mathbf{O}^{\star}\ \mathbf{0}]$ exists in the feasible set of the minimization (11b), i.e., $\|[\,\mathbf{O}^{\star}\ \mathbf{0}]\|_F^2 \leq \epsilon$, where $\mathbf{0}$ is a zero matrix of size $Q \times n$. Therefore, we can show the optimal costs of the two networks have the following relation

$$E\left[\|\mathbf{t} - \bar{\mathbf{O}}^{\star}\bar{\mathbf{y}}\|^2\right] \leq E\left[\|\mathbf{t} - \mathbf{O}^{\star}\mathbf{y}\|^2\right], \tag{13}$$

where the equality happens when $\bar{\mathbf{O}}^{\star} = [\,\mathbf{O}^{\star}\ \mathbf{0}]$. Any other optimal solution of $\bar{\mathbf{O}}$ will lead to inequality relation due to the convexity of the cost. Therefore, we can conclude that the feature vector $\bar{\mathbf{y}}$ of the second network is richer than the feature vector $\mathbf{y}$ of the first network in the sense of reduced training cost. The proposed structure provides an additional property for the feature vector $\bar{\mathbf{y}}$ which we state in the following proposition. The proof idea of the proposition will be used in the next section to construct a multilayer structure, and therefore, we present the proof here.

Proposition 3 *For the feature vector* $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{W}\mathbf{x})$*, there exists an invertible mapping* $\mathbf{h}(\bar{\mathbf{y}}) = \mathbf{x}$ *when the weight matrix* $\mathbf{W}$ *is full-column rank.*

Proof We now state lossless flow property (LFP), as used in [17, 24]. A nonlinear function $\mathbf{g}(\cdot)$ holds the LFP if there exist two linear transformations $\mathbf{V}$ and $\mathbf{U}$ such that $\mathbf{U}\mathbf{g}(\mathbf{V}\mathbf{z}) = \mathbf{z}, \forall \mathbf{z} \in \mathbb{R}^n$. It is shown in [17] that ReLU holds LFP. In other words, if $\mathbf{V} \triangleq \mathbf{V}_n = \begin{bmatrix} \mathbf{I}_n \\ -\mathbf{I}_n \end{bmatrix} \in \mathbb{R}^{2n\times n}$ and $\mathbf{U} \triangleq \mathbf{U}_n = [\mathbf{I}_n\ \ -\mathbf{I}_n] \in \mathbb{R}^{n\times 2n}$, then $\mathbf{U}_n\mathbf{g}(\mathbf{V}_n\mathbf{z}) = \mathbf{z}$ holds for every $\mathbf{z}$ when $\mathbf{g}(\cdot)$ is ReLU. Letting $\mathbf{z} = \mathbf{W}\mathbf{x}$, we can easily find $\mathbf{x} = \mathbf{W}^{\dagger}\mathbf{z} = \mathbf{W}^{\dagger}\mathbf{U}_n\bar{\mathbf{y}}$, where $\dagger$ denotes pseudo-inverse when $\mathbf{W}$ is a full-column rank matrix. Therefore, the resulting inverse mapping $\mathbf{h}$ would be linear. □

4.1 Reduction of training cost with depth

In this section, we show that the proposed HNF provides lower training costs as the number of layers increases. Consider an L-layer feedforward network according to our proposed structure on the weight matrices as follows

$$\bar{\mathbf{y}}^{(L)} = \mathbf{g}(\mathbf{V}_{n^{(L)}}\mathbf{W}^{(L)}\mathbf{g}(\ldots\mathbf{g}(\mathbf{V}_{n^{(1)}}\mathbf{W}^{(1)}\mathbf{x}))). \tag{14}$$

Note that $2n^{(L)}$ is the number of neurons in the L-th layer of the network. The input-output relation in each layer is characterized by

$$\bar{\mathbf{y}}^{(1)} = \mathbf{g}(\mathbf{V}_{n^{(1)}}\mathbf{W}^{(1)}\mathbf{x}), \tag{15a}$$
$$\bar{\mathbf{y}}^{(l)} = \mathbf{g}(\mathbf{V}_{n^{(l)}}\mathbf{W}^{(l)}\bar{\mathbf{y}}^{(l-1)}),\ \ 2 \leq l \leq L, \tag{15b}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{n^{(1)}\times P}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{n^{(l)}\times m^{(l)}}$, and $m^{(l)} = 2n^{(l-1)}$ for $2 \leq l \leq L$. Let the predicted target using the l-th layer feature vector $\bar{\mathbf{y}}^{(l)}$ be $\mathbf{O}_l\bar{\mathbf{y}}^{(l)}$. We find optimal solutions for the following convex optimization problems

$$\mathbf{O}^{\star}_{l-1} = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\bar{\mathbf{y}}^{(l-1)}\|^2\right] \text{ s.t. } \|\mathbf{O}\|_F^2 \leq \epsilon_{l-1}, \tag{16a}$$

$$\mathbf{O}^{\star}_{l} = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\bar{\mathbf{y}}^{(l)}\|^2\right] \text{ s.t. } \|\mathbf{O}\|_F^2 \leq \epsilon_{l}. \tag{16b}$$

Let us define $\mathbf{z}^{(l)} \triangleq \mathbf{W}^{(l)}\mathbf{y}^{(l-1)}$. Assuming that weight matrices $\mathbf{W}^{(l)}$ are full-column rank, we can similarly derive $\mathbf{y}^{(l-1)} = [\mathbf{W}^{(l)}]^{\dagger}\mathbf{z}^{(l)}$. By using Proposition 3, we have $\mathbf{z}^{(l)} = \mathbf{U}_{n^{(l)}}\bar{\mathbf{y}}^{(l)}$, and then, we can write the following relations

$$\bar{\mathbf{y}}^{(l-1)} = [\mathbf{W}^{(l)}]^{\dagger}\mathbf{z}^{(l)} = [\mathbf{W}^{(l)}]^{\dagger}\mathbf{U}_{n^{(l)}}\bar{\mathbf{y}}^{(l)}, \tag{17}$$

where $\mathbf{U}_{n^{(l)}} = [\mathbf{I}_{n^{(l)}} \ -\mathbf{I}_{n^{(l)}}]$. If we choose $\mathbf{O}^{\star}_{l} = \mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\dagger}\mathbf{U}_{n^{(l)}}$, by using (17), we can easily see that $\mathbf{O}^{\star}_{l}\bar{\mathbf{y}}^{(l)} = \mathbf{O}^{\star}_{l-1}\bar{\mathbf{y}}^{(l-1)}$. Therefore, by including $\mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\dagger}\mathbf{U}_{n^{(l)}}$ in the feasible set of the minimization (16b), we can guarantee that the optimal cost of l-th layer would be lower or equal than that of layer $(l-1)$. In particular, by choosing $\epsilon_l = \|\mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\dagger}\mathbf{U}_{n^{(l)}}\|_F^2$, we can see that the optimal costs follow the relation

$$E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{l}\bar{\mathbf{y}}^{(l)}\|^2\right] \leq E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{l-1}\bar{\mathbf{y}}^{(l-1)}\|^2\right], \tag{18}$$

where the equality happens when we have $\mathbf{O}^{\star}_{l} = \mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\dagger}\mathbf{U}_{n^{(l)}}$. Any other optimal solution of $\mathbf{O}_l$ will lead to inequality relation due to the convexity of the cost. Therefore, we can conclude that the feature vector $\bar{\mathbf{y}}^{(l)}$ of an l-layer network is richer than the feature vector $\bar{\mathbf{y}}^{(l-1)}$ of an $(l-1)$-layer network in the sense of reduced training cost. Note that if we choose the weight matrix $\mathbf{W}^{(l)}$ to be orthonormal, then

$$\begin{aligned}\epsilon_l &= \|\mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\top}\mathbf{U}_{n^{(l)}}\|_F^2 \\ &= \operatorname{trace}\left[2\mathbf{I}_{n^{(l)}}\mathbf{W}^{(l)}[\mathbf{O}^{\star}_{l-1}]^{\top}\mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\top}\right] \\ &= 2\operatorname{trace}\left[\mathbf{W}^{(l)}[\mathbf{O}^{\star}_{l-1}]^{\top}\mathbf{O}^{\star}_{l-1}[\mathbf{W}^{(l)}]^{\top}\right] \\ &= 2\operatorname{trace}\left[[\mathbf{W}^{(l)}]^{\top}\mathbf{W}^{(l)}[\mathbf{O}^{\star}_{l-1}]^{\top}\mathbf{O}^{\star}_{l-1}\right] \\ &= 2\operatorname{trace}[[\mathbf{O}^{\star}_{l-1}]^{\top}\mathbf{O}^{\star}_{l-1}] = 2\|\mathbf{O}^{\star}_{l-1}\|_F^2,\end{aligned} \tag{19}$$

where we have used the fact that $\mathbf{U}_{n^{(l)}}[\mathbf{U}_{n^{(l)}}]^{\top} = 2\mathbf{I}_{n^{(l)}}$. As we have $\|\mathbf{O}_{l-1}\|_F^2 \leq \epsilon_{l-1}$, a sufficient condition to guarantee the cost relation (18) is to use the relation between regularization parameters as $\epsilon_l \geq 2\epsilon_{l-1}$. We can choose $\epsilon_l = 2\epsilon_{l-1} = 2^{l-1}\epsilon_1$. Note that the regularization parameter ϵ_1 in the first layer can also be determined analytically. Consider $\mathbf{O}^{\star}_{ls}$ to be the solution of the following least-square optimization

$$\mathbf{O}^{\star}_{\text{ls}} = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\mathbf{x}\|^2\right]. \tag{20}$$

Note that the above minimization has a closed-form solution. Similar to the argument in (18), by choosing $\epsilon_1 = \|\mathbf{O}^{\star}_{\text{ls}}[\mathbf{W}^{(1)}]^{\dagger}\mathbf{U}_{n^{(1)}}\|_F^2$, it can be easily seen that

$$E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{1}\bar{\mathbf{y}}^{(1)}\|^2\right] \leq E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{\text{ls}}\mathbf{x}\|^2\right], \tag{21}$$

where the equality happens only when we have $\mathbf{O}^{\star}_{1} = \mathbf{O}^{\star}_{\text{ls}}[\mathbf{W}^{(1)}]^{\dagger}\mathbf{U}_{n^{(1)}}$. Similar to Proposition 3, we can prove the following proposition regarding the invertibility of the feature vector at the l-th layer of the proposed structure.

Proposition 4 *For the feature vector $\bar{\mathbf{y}}^{(L)}$ in (14), there exists an invertible mapping function $\bar{\mathbf{h}}(\bar{\mathbf{y}}^{(L)}) = \mathbf{x}$ when the set of weight matrices $\{\mathbf{W}^{(l)}\}_{l=1}^{L}$ are full-column rank.*

Proof It can be proved by repeatedly using the lossless flow property (LFP) similar to Proposition 3. □

4.2 Reduction of training cost of ELM

Note that the feature vector $\bar{\mathbf{y}}^{(1)}$ in (15a) can be any feature vector that is used for linear projection to the target in any other learning method. In Section 4.1, we assume $\bar{\mathbf{y}}^{(1)}$ to be the feature vector constructed from $\mathbf{x}$ using the matrix $\mathbf{V}$, and therefore, the regularization parameter ϵ_1 is derived to guarantee performance improvement compared to least-square method as shown in (21). A potential extension would be to build the proposed HNF using the feature vector $\bar{\mathbf{y}}^{(1)}$ from other methods that employ linear projection to estimate the target. For example, the extreme learning machine (ELM) uses a linear projection of the nonlinear feature vector to predict the target [19]. In the following, we build the proposed HNF by employing the feature vector used in ELM to improve the performance.

Similar to Eq. (18), we can show that it is possible to improve the feature vector of ELM in the sense of training cost by using the proposed HNF. Consider $\bar{\mathbf{y}}^{(1)} = \mathbf{g}(\mathbf{W}^{(1)}\mathbf{x})$, to be feature vector used in ELM for linear projection to the target. In the ELM framework, $\mathbf{W}^{(1)} \in \mathbb{R}^{n^{(1)} \times P}$ is an instance of normal distribution, not necessarily full-column rank, and $\mathbf{g}(\cdot)$ can be any activation function, not necessarily ReLU. The optimal mapping to the target in ELM is found by solving the following minimization problem.

$$\mathbf{O}^{\star}_{\text{elm}} = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\bar{\mathbf{y}}^{(1)}\|^2\right]. \quad (22)$$

Note that this minimization problem has a closed-form solution. We construct the feature vector in the second layer of the HNF as

$$\bar{\mathbf{y}}^{(2)} = \mathbf{g}\left(\mathbf{V}_{n^{(2)}}\mathbf{W}^{(2)}\bar{\mathbf{y}}^{(1)}\right), \quad (23)$$

where $\mathbf{W}^{(2)} \in \mathbb{R}^{n^{(2)} \times m^{(2)}}$ and $m^{(2)} = n^{(1)}$. The optimal mapping to the target by using this feature vector can be found by solving

$$\mathbf{O}^{\star}_{2} = \arg\min_{\mathbf{O}} E\left[\|\mathbf{t} - \mathbf{O}\bar{\mathbf{y}}^{(2)}\|^2\right] \text{ s.t. } \|\mathbf{O}\|_F^2 \leq \epsilon_2, \quad (24)$$

where ϵ_2 is the regularization parameter. By choosing $\epsilon_2 = \|\mathbf{O}^{\star}_{\text{elm}}[\mathbf{W}^{(2)}]^{\dagger}\,\mathbf{U}_{n^{(2)}}\|_F^2$, we can see that the optimal costs follow the relation

$$E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{2}\bar{\mathbf{y}}^{(2)}\|^2\right] \leq E\left[\|\mathbf{t} - \mathbf{O}^{\star}_{\text{elm}}\bar{\mathbf{y}}^{(1)}\|^2\right], \quad (25)$$

where the equality happens when we have $\mathbf{O}^{\star}_{2} = \mathbf{O}^{\star}_{\text{elm}}[\mathbf{W}^{(2)}]^{\dagger}\,\mathbf{U}_{n^{(2)}}$. Otherwise, the inequality has to follow. Similarly, we can continue to add more layer to improve the performance. Specifically, for l-th layer of the HNF, we have $\bar{\mathbf{y}}^{(l)} = \mathbf{g}\left(\mathbf{V}_{n^{(l)}}\mathbf{W}^{(l)}\bar{\mathbf{y}}^{(l-1)}\right)$, and we can show that Eq. (18) holds here as well when the set of matrices $\{\mathbf{W}^{(l)}\}_{l=2}^{L}$ are full-column rank.

4.3 Practical considerations

The dimension of feature vector $\bar{\mathbf{y}}^{(l)}$ increases as the number of layers increases. For a multilayer feedforward network, if we use orthonormal matrix $\mathbf{W}^{(l)}$ for l-th layer, then each layer produces a feature vector that has at least twice the dimension of the input feature vector. At the L-th layer, we get the dimension 2^L times of the input data dimension. Note that $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{z})$ is norm-preserving by Proposition 1, that means $\|\bar{\mathbf{y}}\|^2 = \|\mathbf{z}\|^2$. Using this principle successively, the full network is also norm-preserving, that means $\|\bar{\mathbf{y}}^{(L)}\|^2 = \|\mathbf{x}\|^2$. Therefore, as the layer number increases, the amplitudes of scalars of the feature vector $\bar{\mathbf{y}}^{(L)}$ diminish at the rate of 2^L. We show that the proposed HNF does not require a large number of layers to improve the performance. This also answers the natural question that whether many layers are practically required for an ANN. Note that since the dimension of the feature vector $\bar{\mathbf{y}}^{(l)}$ is growing exponentially as 2^l, the proposed HNF is not suitable for cases where the input dimension is too large. One way to circumvent this issue is to employ the kernel trick [3] by using the feature vector $\bar{\mathbf{y}}^{(l)}$. We will address this solution in future works.

5 Results and discussion

In this section, we carry out experiments to validate the performance improvement and observe the effect of using the matrix $\mathbf{V}$ in the architecture of an HNF. We report our results for three popular datasets in the literature as in Table 1. Note that we only choose the datasets where the input dimension is not very large due to the computational complexities. Letter dataset [27] contains a 16-dimensional feature vector for each of the 26 English alphabets from A to Z. Shuttle dataset [28] belongs to the STATLOG project and contains a 9-dimensional feature vector that deals with the positioning of radiators in the space shuttles. MNIST dataset [29] contains gray-scale 28×28-pixel images of hand-written digits. Note that in all three datasets, the target vector $\mathbf{t}$ is one-hot vector of dimension Q (the number of classes). The optimization method used for solving the minimization problem (16b) is the alternating direction method of multipliers (ADMM) [30]. The number of iterations of ADMM is set to 100 in all the simulations.

We carry out two sets of experiments. First, we implement the proposed HNF with a fixed number of layers by using instances of random matrices for designing the weight matrix in every layer. In this setup, the weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{n^{(l)} \times m^{(l)}}$ is an instance of Gaussian distribution with appropriate size $n^{(l)} \geq m^{(l)}$ and entries drawn independently from $\mathcal{N}(0,1)$ to ensure being full-column rank. Second, we construct the proposed HNF by using discrete cosine transform (DCT), as an example of full-column rank weight matrix, instead of random matrices. In this scenario, we may need to apply zero-padding before DCT to build the weight matrix $\mathbf{W}^{(l)}$ with appropriate dimension. The step size in the ADMM algorithm is set accordingly in each of these experiments. Finally, we compare the performance and computational complextiy of HNF and backpropagation over the same-size network.

5.1 HNF using random matrix

In this subsection, we construct the proposed HNF by using instances of Gaussian distribution to design the weight matrix $\mathbf{W}^{(l)}$. In particular, the entries of the weight matrix are drawn independently from $\mathcal{N}(0,1)$. For simplicity, the number of nodes is chosen according to $n^{(l)} = m^{(l)}$ for $l \geq 2$ in all the experiment. The number of nodes in the first layer $2n^{(1)}$ is chosen for each dataset individually such that it satisfies $n^{(1)} \geq P$ for every dataset with input dimension P, as reported in Table 1. The step size in the ADMM algorithm is set to 10^{-7} in all the simulations in this subsection.

Table 1 Test classification accuracy of the proposed HNF for different datasets using random matrices

Dataset	Size of training data	Size of testing data	Input dimension (P)	Number of classes (Q)	Proposed HNF			ELM	Proposed HNF			State-of-the-art [reference]
					Accuracy	$n^{(1)}$	L	Accuracy	Accuracy	$n^{(1)}$	L	
Letter	13,333	6667	16	26	93.3	250	5	88.3	94.6	1000	3	95.8 [25]
Shuttle	43,500	14,500	9	7	99.3	250	5	99.0	99.6	1000	3	99.9 [19]
MNIST	60,000	10,000	784	10	97.1	1000	5	96.9	97.7	4000	3	99.7 [26]

We implement two different scenarios. First, we implement the proposed HNF with a fixed number of layers and show performance improvement throughout the layers. In this setup, the only hyperparameter that needs to be chosen is the number of nodes in the first layer $2n^{(1)}$. Note that the regularization parameter ϵ_1 is chosen such that it guarantees (21), and therefore eliminates the need for cross-validation in the first layer. Second, we build the proposed HNF by using the ELM feature vector in the first layer as in (23) and show the performance improvement throughout the layers. In this setup, the only hyperparameter that needs to be chosen is the number of nodes in the first layer $n^{(1)}$ which is the number of nodes of ELM to be exact. It has been shown that ELM performs better as the number of hidden neuron increases [24]; therefore, we choose a sufficiently large hidden neuron to make sure that ELM is performing at its best. Note that the regularization parameter ϵ_1 is chosen such that it guarantees (25), and therefore eliminates the need for cross-validation. Finally, we present the classification performance of the corresponding state-of-the-art results in Table 1.

The performance results of the proposed HNF with $L = 5$ layers are reported in Table 1. We report test classification accuracy as a measure to evaluate the performance. Note that the number of neurons $2n^{(1)}$ in the first layer of HNF is chosen appropriately for each dataset such that it satisfies $n^{(1)} \geq P$. For example, for MNIST dataset, we set $n^{(1)} = 1000 \geq P = 784$. The performance improvement in each layer of HNF is given in Fig. 1, where train and test classification accuracy is shown versus total number of nodes in the network $\sum_{l=1}^{L} 2n^{(l)}$. Note that the total number of nodes being zero corresponds to direct mapping of the input $\mathbf{x}$ to the target using least-squares according to (20). It can be seen that the proposed HNF provides a substantial improvement in performance with a small number of layers.

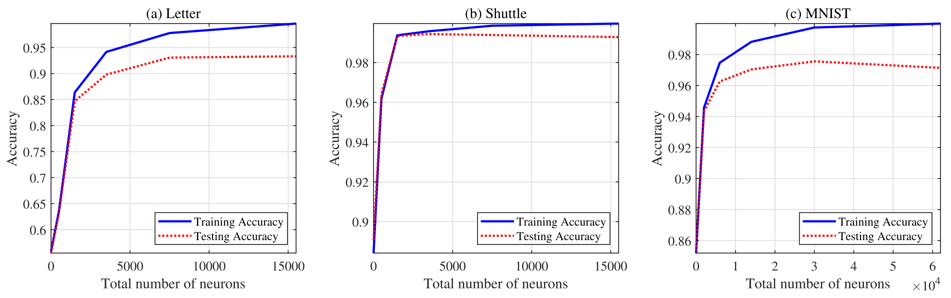

Fig. 1 Training and testing accuracy against size of one instance of HNF. Size of an L-layer HNF is represented by the number of random matrix-based nodes, counted as $\sum_{l=1}^{L} 2n^{(l)}$. Here, $L = 5$ for all three datasets. The number of nodes in the first layer ($2\,n^{(1)}$) is set according to Table 1 for each dataset

The corresponding performance for the case of using the ELM feature vector in the first layer of HNF is reported in Table 1. It can be seen that HNF provides a tangible improvement in performance compared to ELM. Note that the number of neurons in the first layer $n^{(1)}$ is, in fact, the same as the number of neurons used in ELM. We choose $n^{(1)}$ to get the best performance for ELM in every dataset individually. The number of layers in the network is set to $L = 3$ to avoid the increasing computational complexity. The performance improvement in each layer of HNF in this case is given in Fig. 2, where train and test classification accuracy is shown versus total number of nodes in the network $n^{(1)} + \sum_{l=2}^{L} 2n^{(l)}$. Note that the initial point corresponding to $n^{(1)}$ is in fact equal to the ELM performance reported in Table 1, which is derived according to (22).

Finally, we compare the performance of the proposed HNF with the state-of-the-art performance for these three datasets. We can see that the proposed HNF provides competitive performance compared to state-of-the-art results in the literature. It is worth mentioning that we have not used any pre-processing technique to improve the performance as in the state-of-the-art, but it can be done in future works.

5.2 HNF using DCT

In this subsection, we repeat the same experiments as in Section 5.1 by using DCT instead of the Gaussian weight matrix. The number of nodes in each layer of the network is chosen as in Section 5.1. We apply zero-padding before DCT in the first layer to build the weight matrix $\mathbf{W}^{(1)} \in \mathbb{R}^{n^{(1)} \times P}$ with appropriate dimension for each dataset. Note that $n^{(l)} = m^{(l)}$ for $l \geq 2$ in all the experiments, and therefore, there is no need to apply zero-padding in the next layers. The step size in the ADMM algorithm is set to 10^2 in all the simulations in this subsection.

We implement the same two scenarios. First, we implement the proposed HNF by using DCT and show performance improvement throughout the layers. Second, we build the proposed HNF by using the ELM feature vector in the first layer and DCT matrices in the next layers. Note that the regularization parameters ϵ_l for $l \geq 2$ are chosen according to (19). The choice of ϵ_1 is such that it guarantees (21) and (25) according to each scenario.

The performance results of the proposed HNF by using DCT matrices are reported in Table 2. Note that the number of neurons $n^{(1)}$ in the first layer and the number of layers

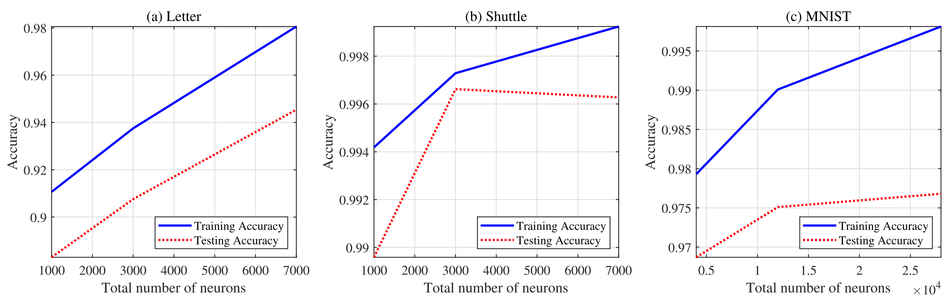

Fig. 2 Training and testing accuracy against size of one instance of HNF using ELM feature vector in the first layer. Size of an L-layer HNF is represented by the number of random matrix-based nodes, counted as $n^{(1)} + \sum_{l=2}^{L} 2n^{(l)}$. Here, $L = 3$ for all three datasets. The number of nodes in the first layer ($n^{(1)}$) is set according to Table 1 for each dataset

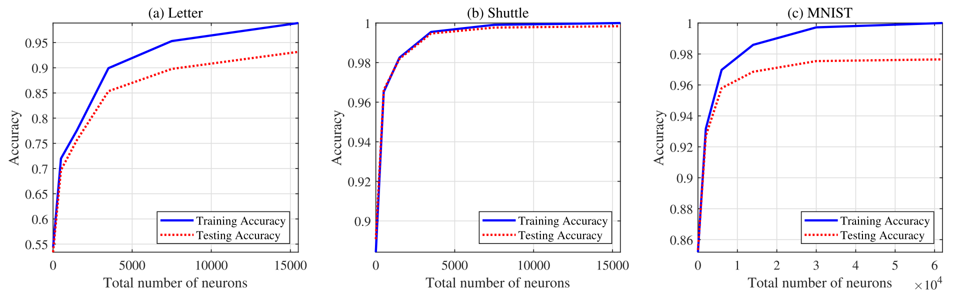

Fig. 3 Training and testing accuracy against size of HNF using DCT in every layer. Size of an L-layer HNF is represented by the number of DCT-based nodes, counted as $\sum_{l=1}^{L} 2n^{(l)}$. Here, $L = 5$ for all three datasets. The number of nodes in the first layer ($2\,n^{(1)}$) is set according to Table 2 for each dataset

are the same as Table 1. The performance improvement in each layer of HNF is given in Figs. 3 and 4. It can be seen that by using DCT in the proposed HNF, it is also possible to improve the performance with a few layers.

Finally, we compare the performance of the DCT-based HNF and that of the random matrix-based HNF as shown in Tables 1 and 2. We can see that using DCT as the weight matrix is as powerful as using random weights in these three datasets.

5.3 Computational complexity

Finally, we compare test classification accuracy and computational complexity of HNF with the backpropagation over the same learned HNF. We report training time of each method in seconds. We run our experiments on a server with multiprocessors and 256 GB RAM. The optimization method used for backpropagation is ADAM [31] from TensorFlow. The learning rate of ADAM is chosen via cross-validation, and the number of epochs is fixed to 1000 in all the experiments.

We construct HNF by using random weights and use the same number of layers and nodes as in Table 1. Note that we do not use ELM feature vector in the first layer for this experiments, although it is possible to use it in order to improve the performance. The results are shown in Table 3. As expected, backpropagation can improve the performance, except for Shuttle, at the cost of a significantly higher computational complexity. HNF, on the other hand, does not require cross-validation and only performs training at the last

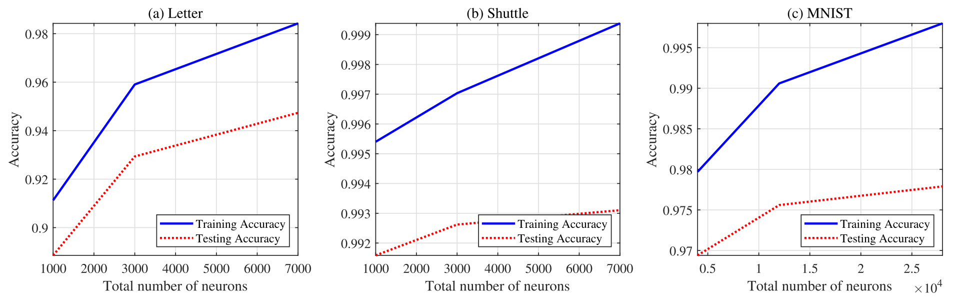

Fig. 4 Training and testing accuracy against size of HNF using ELM feature vector in the first layer and DCT in the next layers. Size of an L-layer HNF is represented by the number of nodes, counted as $n^{(1)} + \sum_{l=2}^{L} 2n^{(l)}$. Here, $L = 3$ for all three datasets. The number of nodes in the first layer ($n^{(1)}$) is set according to Table 2 for each dataset

Table 2 Test classification accuracy of the proposed HNF for different datasets using DCT

Dataset	Proposed HNF			ELM	Proposed HNF		
	Accuracy	$n^{(1)}$	L	Accuracy	Accuracy	$n^{(1)}$	L
Letter	93.2	250	5	88.3	94.7	1000	3
Shuttle	99.8	250	5	99.0	99.3	1000	3
MNIST	97.7	1000	5	96.9	97.8	4000	3

layer of the network, leading to a much faster training. Note that training time reported for backpropapation in Table 3 does not include cross-validation for the learning rate so that we can have a fair comparison with HNF.

At this point, we also provide the reported classification performance of scattering network on MNIST dataset for the sake of completeness. Scattering network with principal component analysis (PCA) [14] over a modulus of windowed Fourier transforms yields 98.2% test classification accuracy for a spatial support equal to 8. This result shows that scattering network can outperform HNF at the cost of a higher complexity of using several scattering integrals in each layer. Note that HNF only uses a random choice of a Gaussian distribution as the weight matrix in each layer. Besides, scattering network requires accurate choice of several hyperparameters such as the spatial support, number of filter banks, and type of the transforms, which can be crucial for the performance. For example, in our experiments, a scattering network with PCA over a modulus of 2D Morlet wavelets provides 94% accuracy, at best, for a spatial support of 28. The training on the our server lasted 1158 s to yield such an accuracy, which highlights the learning speed of HNF in Table 3. The same network with a spatial support of 14 gives a performance of 56.03%, showing the importance of a precise cross-validation.

6 Conclusion

We show that by using a combination of orthonormal matrices and ReLU activation functions, it is possible to guarantee a monotonically decreasing training cost as the number of layers increases. The proposed method can be used by employing any other loss function, such as cross-entropy loss, as long as a linear projection is used after the ReLU activation function. Note that the same principle applies if instead of random matrices, we use any other real orthonormal matrices. Discrete cosine transform (DCT), Haar transform, and Walsh-Hadamard transform are examples of this kind. The proposed HNF is a universal architecture in the sense that it can be applied to improve the performance of any other learning method which employs linear projection to predict the target. The norm-preserving and invertibility of the architecture make the proposed HNF suitable for other applications such as auto-encoder design.

Table 3 Training time and test classification accuracy of the proposed HNF versus backpropagation

Dataset	Proposed HNF		Backpropagation	
	Accuracy	Training time	Accuracy	Training time
Letter	93.42	47 s	95.03	4566 s
Shuttle	99.21	24 s	99.21	15,283 s
MNIST	97.14	108 s	98.30	20,433 s

Appendix 1. Proof of Property 3

Proof For scalars x_1 and x_2, we have $y_1 = g(x_1)$ and $y_2 = g(x_2)$. We have following relation

$$(y_1 - y_2)^2 = \begin{cases} (x_1 - x_2)^2 & \text{if } x_1 > 0, x_2 > 0 \\ x_1^2 & \text{if } x_1 > 0, x_2 < 0 \\ x_2^2 & \text{if } x_1 < 0, x_2 > 0 \\ 0 & \text{if } x_1 < 0, x_2 < 0. \end{cases} \tag{26}$$

Therefore, we find that ReLU function holds $0 \leq (y_1 - y_2)^2 \leq (x_1 - x_2)^2$. Considering the vectors $\mathbf{y}_1 = \mathbf{g}(\mathbf{z}_1) = \mathbf{g}(\mathbf{W}\mathbf{q}_1)$ and $\mathbf{y}_2 = \mathbf{g}(\mathbf{z}_2) = \mathbf{g}(\mathbf{W}\mathbf{q}_2)$, we have

$$\begin{aligned} 0 \leq \|\mathbf{y}_1 - \mathbf{y}_2\|^2 &= \sum_i (y_1(i) - y_2(i))^2 \\ \leq \sum_i (z_1(i) - z_2(i))^2 &= \|\mathbf{z}_1 - \mathbf{z}_2\|^2, \end{aligned} \tag{27}$$

where $y_1(i)$ is the i-th scalar element of $\mathbf{y}_1$ and $z_1(i)$ is the i-th scalar element of $\mathbf{z}_1$. □

Appendix 2. Proof of Proposition 1

Proof We have $\mathbf{z} = \mathbf{W}\mathbf{q} \in \mathbb{R}^n$ and $\bar{\mathbf{y}} = \mathbf{g}(\mathbf{V}_n\mathbf{z}) \in \mathbb{R}^{2n}$ where $\mathbf{V}_n = \begin{bmatrix} \mathbf{I}_n \\ -\mathbf{I}_n \end{bmatrix}$. For two vectors $\mathbf{q}_1$ and $\mathbf{q}_2$, we have corresponding vectors $\mathbf{z}_1 = \mathbf{W}\mathbf{q}_1$ and $\mathbf{z}_2 = \mathbf{W}\mathbf{q}_2$, and output vectors $\bar{\mathbf{y}}_1 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_1)$ and $\bar{\mathbf{y}}_2 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_2)$. Note that $\bar{\mathbf{y}}_1 = \begin{bmatrix} \mathbf{z}_1^+ \\ -\mathbf{z}_1^- \end{bmatrix}$, and therefore, $\|\bar{\mathbf{y}}_1\|^2 = \|\mathbf{z}_1^+\|^2 + \|\mathbf{z}_1^-\|^2 = \|\mathbf{z}_1\|^2$, by definition. Similarly, $\|\bar{\mathbf{y}}_2\|^2 = \|\mathbf{z}_2\|^2$. Let us define a set

$$\mathcal{M}(\mathbf{z}_1, \mathbf{z}_2) = \{i | s(z_1(i)) = s(z_2(i)) \neq 0\} \subseteq \{1, 2, \ldots, n\}.$$

Then, we have

$$\begin{aligned} \|\mathbf{z}_1 - \mathbf{z}_2\|^2 &= \sum_{i=1} (z_1(i) - z_2(i))^2 \\ &= \sum_i (s(z_1(i))|z_1(i)| - s(z_2(i))|z_2(i)|)^2 \\ &= \sum_{i \in \mathcal{M}(\mathbf{z}_1, \mathbf{z}_2)} (|z_1(i)| - |z_2(i)|)^2 \\ &+ \sum_{i \in \mathcal{M}^c(\mathbf{z}_1, \mathbf{z}_2)} (|z_1(i)| + |z_2(i)|)^2. \end{aligned} \tag{28}$$

We write $\mathbf{z}_1 = \mathbf{z}_1^+ + \mathbf{z}_1^- = \mathbf{s}\left(\mathbf{z}_1^+\right)|\mathbf{z}_1^+| + \mathbf{s}\left(\mathbf{z}_1^-\right)|\mathbf{z}_1^-|$. Then, after ReLU operation, we have $\bar{\mathbf{y}}_1 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_1) = \begin{bmatrix} |\mathbf{z}_1^+| \\ |\mathbf{z}_1^-| \end{bmatrix}$ and $\bar{\mathbf{y}}_2 = \mathbf{g}(\mathbf{V}_n\mathbf{z}_2) = \begin{bmatrix} |\mathbf{z}_2^+| \\ |\mathbf{z}_2^-| \end{bmatrix}$.

$$
\begin{aligned}
\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 &= \||\mathbf{z}_1^+| - |\mathbf{z}_2^+|\|^2 + \||\mathbf{z}_1^-| - |\mathbf{z}_2^-|)\|^2 \\
&= \sum_{i\in\mathcal{M}\left(|\mathbf{z}_1^+|,|\mathbf{z}_2^+|\right)} \left(|z_1^+(i)| - |z_2^+(i)|\right)^2 + \sum_{i\in\mathcal{M}^c\left(|\mathbf{z}_1^+|,|\mathbf{z}_2^+|\right)} \left(|z_1^+(i)| + |z_2^+(i)|\right)^2 \\
&+ \sum_{i\in\mathcal{M}\left(|\mathbf{z}_1^-|,|\mathbf{z}_2^-|\right)} \left(|z_1^-(i)| - |z_2^-(i)|\right)^2 + \sum_{i\in\mathcal{M}^c\left(|\mathbf{z}_1^-|,|\mathbf{z}_2^-|\right)} \left(|z_1^-(i)| + |z_2^-(i)|\right)^2 \\
&= \sum_{i\in\mathcal{M}(|\mathbf{z}_1^+|,|\mathbf{z}_2^+|)} \left(|z_1^+(i)| - |z_2^+(i)|\right)^2 + \sum_{i\in\mathcal{M}(|\mathbf{z}_1^-|,|\mathbf{z}_2^-|)} \left(|z_1^-(i)| - |z_2^-(i)|\right)^2 \\
&+ \sum_{i\in\mathcal{M}^c(|\mathbf{z}_1^+|,|\mathbf{z}_2^+|)} \left(|z_1^+(i)| + |z_2^+(i)|\right)^2 + \sum_{i\in\mathcal{M}^c(|\mathbf{z}_1^-|,|\mathbf{z}_2^-|)} \left(|z_1^-(i)| + |z_2^-(i)|\right)^2 \\
&= \sum_{i\in\mathcal{M}(\mathbf{z}_1,\mathbf{z}_2)} (|z_1(i)| - |z_2(i)|)^2 + \sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} |z_1(i)|^2 + |z_2(i)|^2. \\
&= \|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - 2\sum_{i\in\mathcal{M}(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i) \\
&= \|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - 2\sum_{i=1}^{n} \mathbf{z}_1(i)\mathbf{z}_2(i) + 2\sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i) \\
&= \|\mathbf{z}_1 - \mathbf{z}_2\|^2 + 2\sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i) \qquad (29a) \\
&= \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 + \frac{1}{2}(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - 2\sum_{i\in\mathcal{M}(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i) \\
&\quad + 2\sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i)) \\
&= \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 + \frac{1}{2}\left(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - 2\sum_{i\in\mathcal{M}(\mathbf{z}_1,\mathbf{z}_2)} |\mathbf{z}_1(i)||\mathbf{z}_2(i)|\right. \\
&\quad \left. - 2\sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} |\mathbf{z}_1(i)||\mathbf{z}_2(i)|\right) \\
&= \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 + \frac{1}{2}\left(\|\mathbf{z}_1\|^2 + \|\mathbf{z}_2\|^2 - 2\sum_{i=1}^{n} |\mathbf{z}_1(i)||\mathbf{z}_2(i)|\right) \\
&= \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 + \frac{1}{2}\||\mathbf{z}_1| - |\mathbf{z}_2|\|^2 \qquad (29b)
\end{aligned}
$$

With similar calculations as in (28), we can derive the relationships in Eqs. (29a) and (29b). Since the summation $\sum_{i\in\mathcal{M}^c(\mathbf{z}_1,\mathbf{z}_2)} \mathbf{z}_1(i)\mathbf{z}_2(i)$ is always non-positive, from (29a), we can see that

$$\|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2 \leq \|\mathbf{z}_1 - \mathbf{z}_2\|^2, \qquad (30)$$

where equality holds when $\mathcal{M}^c = \emptyset$, that means when sign patterns of $\mathbf{z}_1$ and $\mathbf{z}_2$ match exactly. From (29b), it can also be seen that

$$\frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2 \leq \|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\|^2, \qquad (31)$$

where equality holds when $|\mathbf{z}_1| = |\mathbf{z}_2|$. □

Appendix 3. Proof of Remark 3

Proof Consider $\mathbf{z} = \mathbf{W}\mathbf{q}$ and $\mathbf{z}_\Delta \triangleq [\,\mathbf{W} + \Delta\mathbf{W}]\,\mathbf{q} = \mathbf{W}\mathbf{q} + [\,\Delta\mathbf{W}]\,\mathbf{q} = \mathbf{z} + \Delta\mathbf{z}$. Based on Proposition 1, we can simply write

$$\begin{aligned} \|\bar{\mathbf{y}} - \bar{\mathbf{y}}_\Delta\|^2 &\le \|\Delta\mathbf{z}\|^2 = \|[\,\Delta\mathbf{W}]\ \mathbf{q}\|^2 \\ &\le \|\Delta\mathbf{W}\|_F^2 \|\mathbf{q}\|^2, \end{aligned} \tag{32}$$

where we have used Eq. (4). □

Abbreviations
ReLU: Rectified linear unit; SVM: Support vector machine; KPCA: Kernel principal component analysis; RBF: Radial basis function; ANN: Artificial neural network; HNF: High-dimensional neural feature; DNN: Deep neural network; CNN: Convolutional neural network; RNN: Recurrent neural network; ELM: Extreme leaning machine; PLN: Progressive learning network; DCT: Discrete cosine transform; ADMM: Alternating direction method of multipliers; PCA: Principal component analysis; ADAM: Adaptive moment estimation

Acknowledgements
We acknowledge the support of our KTH colleagues Amirreza Zamani and Hamid Ghourchian for proofreading and critical remarks.

Authors' contributions
AJ, AV, MS, and SC conceived and designed the study. AJ performed the experiments. AJ and SC wrote the paper. AJ, AV, and MS reviewed and edited the manuscript. All authors read and approved the manuscript.

Funding
The work is not funded by any private grant.

Availability of data and materials
All datasets used in the experiments are publicly available online. Please contact the corresponding author for simulation results.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 1 November 2019 Accepted: 3 August 2020
Published online: 10 September 2020

References
1. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
2. B. Schölkopf, A. Smola, K.-R. Muller, Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
3. C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. (Springer, Berlin, Heidelberg, 2006)
4. D. Yu, L. Deng, Deep learning and its applications to signal and information processing [exploratory dsp]. IEEE Signal Process. Mag. **28**(1), 145–154 (2011)
5. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge. Intl. J. Computer Vision. **115**(3), 211–252 (2015)
6. S. F. Dodge, L. J. Karam, A study and comparison of human and deep learning recognition performance under visual distortions. ArXiv e-prints (2017). 1705.02498. Accessed 1 Oct 2019
7. L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, in *Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28*. ed. by S. Dasgupta, D. McAllester, Regularization of neural networks using dropconnect (PMLR, Atlanta, 2013), pp. 1058–1066
8. D. Mishkin, J. Matas, in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. ed. by Y. Bengio, Y. LeCun, All you need is a good init, (2016). http://arxiv.org/abs/1511.06422
9. C.-Y. Lee, P. W. Gallagher, Z. Tu, in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 51*. ed. by A. Gretton, C. C. Robert, Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree (PMLR, Cadiz, 2016), pp. 464–472
10. A. J. Thomas, M. Petridis, S. D. Walters, S. M. Gheytassi, R. E. Morgan, in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, On predicting the optimal number of hidden nodes, (2015), pp. 565–570
11. C. Szegedy, A. Toshev, D. Erhan, in *Advances in Neural Information Processing Systems 26*. ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Deep neural networks for object detection, (2013), pp. 2553–2561
12. A. Krizhevsky, I. Sutskever, G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, Imagenet classification with deep convolutional neural networks (Curran Associates Inc., Red Hook, 2012), pp. 1097–1105

13. I. Sutskever, *Training Recurrent Neural Networks. PhD thesis*. (University of Toronto, Toronto, 2013). AAINS22066
14. J. Bruna, S. Mallat, Invariant scattering convolution networks. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1872–1886 (2013)
15. R. Vidal, J. Bruna, R. Giryes, S. Soatto, Mathematics of deep learning. ArXiv e-prints (2017). 1712.04741. Accessed 1 Oct 2019
16. R. Giryes, G. Sapiro, A. M. Bronstein, Deep neural networks with random Gaussian weights: a universal classification strategy? IEEE Trans. Signal Process. **64**(13), 3444–3457 (2016)
17. S. Chatterjee, A. M. Javid, S. K. Mostafa Sadeghi, P. P. Mitra, M. Skoglund, SSFN: self size-estimating feed-forward network and low complexity design. ArXiv e-prints (2019). 1905.07111. Accessed 1 Oct 2019
18. S. Chatterjee, A. M. Javid, M. Sadeghi, P. P. Mitra, M. Skoglund, Progressive learning for systematic design of large neural networks. ArXiv e-prints (2017). 1710.08177. Accessed 1 Oct 2019
19. G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification. J. Trans. Sys. Man Cyber. B. **42**(2), 513–529 (2012)
20. G. Huang, G.-B. Huang, S. Song, K. You, Trends in extreme learning machines: a review. Neural Netw. **61**(Supplement C), 32–48 (2015)
21. T. Hussain, S. M. Siniscalchi, C. C. Lee, S. S. Wang, Y. Tsao, W. H. Liao, Experimental study on extreme learning machine applications for speech enhancement. IEEE Access. **PP**(99), 1 (2017)
22. W. Zhu, J. Miao, L. Qing, G. Huang, in *2015 International Joint Conference on Neural Networks (IJCNN)*, Hierarchical extreme learning machine for unsupervised representation learning, (2015), pp. 1–8
23. A. Rosenfeld, J. K. Tsotsos, Intriguing properties of randomly weighted networks: generalizing while learning next to nothing. ArXiv e-prints (2018). 1802.00844. Accessed 1 Oct 2019
24. A. M. Javid, S. Chatterjee, M. Skoglund, in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, Mutual information preserving analysis of a single layer feedforward network, (2018), pp. 1–5
25. J. Tang, C. Deng, G. Huang, Extreme learning machine for multilayer perceptron. IEEE Trans. Neural Netw. Learn. Syst. **27**(4), 809–821 (2016)
26. L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, in *Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28*. ed. by S. Dasgupta, D. McAllester, Regularization of neural networks using dropconnect (PMLR, Atlanta, 2013), pp. 1058–1066
27. P. W. Frey, D. J. Slate, Letter recognition using Holland-style adaptive classifiers. Mach. Learn. **6**(2), 161–182 (1991). https://archive.ics.uci.edu/ml/datasets/letter+recognition
28. C. L. Blake, C. J. Merz, *UCI Repository of Machine Learning Databases*. (Dept. Inf. Comput. Sci., Univ., Irvine, 1998). http://www.ics.uci.edu/~mlearn/MLRepository.html
29. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, in *Proceedings of the IEEE*, Gradient-based learning applied to document recognition, (1998), pp. 2278–2324. Available: http://yann.lecun.com/exdb/mnist/
30. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
31. D. P. Kingma, J. Ba, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, ADAM: a method for stochastic optimization, (2015)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.