XNAS: A Regressive/Progressive NAS for Deep Learning

S. Y. KUNG, Princeton University

Deep learning has achieved great and broad breakthroughs in many real-world applications. In particular, the task of training the network parameters has been masterly handled by back-propagation learning. However, the pursuit on optimal network structures remains largely an art of trial and error. This prompts some urgency to explore an architecture engineering process, collectively known as Neural Architecture Search (NAS). In general, NAS is a design software system for automating the search of effective neural architecture. This article proposes an X-learning NAS (XNAS) to automatically train a network's structure and parameters. Our theoretical footing is built upon the subspace and correlation analyses between the input layer, hidden layer, and output layer. The design strategy hinges upon the underlying principle that the network should be coerced to learn how to structurally improvethe input/output correlation successively (i.e., layer by layer). It embraces both Progressive NAS (PNAS) and Regressive NAS (RNAS). For unsupervised RNAS, Principal Component Analysis (PCA) is a classic tool for subspace analyses. By further incorporating teacher's guidance, PCA can be extended to Regression Component Analysis (RCA) to facilitate supervised NAS design. This allows the machine to extract components most critical to the targeted learning objective. We shall further extend the subspace analysis from multi-layer perceptrons to convolutional neural networks, via introduction of Convolutional-PCA (CPCA) or, more simply, Deep-PCA (DPCA). The supervised variant of DPCA will be named *Deep-RCA* (DRCA). The subspace analyses allow us to compute optimal eigenvectors (respectively, eigen-filters) and principal components (respectively, eigen-channels) for optimal NAS design of multi-layer perceptrons (respectively, convolutional neural networks). Based on the theoretical analysis, an *X-learning* paradigm is developed to jointly learn the structure and parameters of learning models. The objective is to reduce the network complexity while retaining (and sometimes improving) the performance. With carefully pre-selected baseline models, X-learning has shown great successes in numerous classification-type and/or regression-type applications. We have applied X-learning to the ImageNet datasets for classification and DIV2K for image enhancements. By applying X-learning to two types of baseline models, MobileNet and ResNet, both the low-power and high-performance application categories can be supported. Our simulations confirm that X-learning is by and large very competitive relative to the state-of-the-art approaches.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning;**

Additional Key Words and Phrases: Deep learning, X-learning, neural architectural search (NAS), regression/progression NAS, discriminant information (DI), regression component analysis (RCA), Kernel-LSE (KLSE), Deep-PCA (DPCA)

ACM Reference format:
S. Y. Kung. 2022. XNAS: A Regressive/Progressive NAS for Deep Learning. *ACM Trans. Sensor Netw.* 18, 4, Article 57 (November 2022), 32 pages.
https://doi.org/10.1145/3543669

Author's address: S. Y. Kung, E-Quad B230, Princeton University, Princeton, NJ 08544; email: kung@princeton.edu.

1550-4859/2022/11-ART57
https://doi.org/10.1145/3543669

1 INTRODUCTION

Deep learning networks provide a versatile platform to facilitate a broad spectrum of AI applications. However, it remains largely unresolved on various issues concerning the structural aspects of deep networks. To combat this problem, it has recently become trendy to explore **Neural Architecture Search (NAS)** [4, 5, 7, 12, 28, 47]. Along this vein, this article studies subspace analyses critical to **Regression NAS (RNAS)** and **Progressive NAS (PNAS)** for deep learning, which ultimately leads to a joint parameter-structure learning paradigms named **X-learning NAS (XNAS)**.

The organization and the logical flow of the article is as follows. Section 2 explains why the NAS design deep learning network can be perceived as a structural learning strategy designed to successively (i.e., layer by layer) enhance the desired input/output correlation. Therefore, our paramount objective hinges upon maximization of such correlation. To this end, Section 2 develops vital subspace analysis to support RNAS and PNAS designs for **Multi-Layer Perceptrons (MLPs)**. For that purpose, the algorithmic tools developed are **Principal Component Analysis (PCA)** and **Regression Component Analysis (RCA)** for unsupervised and supervised learning of MLPs, respectively. Section 3 further extends the analysis to cover **Convolutional Neural Network (CNN)** PNAS/RNAS, for which the corresponding algorithms are **Deep-PCA (DPCA)**/**Deep-RCA (DRCA)** for structural learning of CNNs. Based on the said analyses, Section 4 develops an *X-learning* paradigm to jointly train the structure and parameters of the network. Finally, Section 5 empirically demonstrates a broad spectrum of successful and competitive applications of XNAS.

2 SUBSPACE ANALYSIS FOR MLPS: RNAS AND PNAS

With reference to the U-shaped performance curve illustrated in Figure 1, there are two ways to up-grade the structure of networks: (1) PNAS: growing the baseline network (e.g., [9, 27]) or (2) RNAS: trim it into a slimmer model (e.g., [12, 14, 15, 24, 29, 44, 46]). We shall look into subspace analysis for PNAS and RNAS in deep learning. This section provides the theoretical footing for MLPs [33, 36, 37, 42].

2.1 Role of Least Square Error in Linear Correlation: The Higher the Correlation, the Lower the Least Square Error

In a nutshell, our learning objective is to see how the hidden layers might be gradually transformed (i.e., layer by layer) so that they become more and more aligned to the targeted output [21, 23, 41]. Mathematically, it means higher and higher correlation between the subspace spanned by the hidden layer(s) and the targeted (teacher) subspace at the output layer. This is pictorially illustrated by Figure 2, in which an originally low input-output correlation (cf. Figure 2(a)) is shown to be gradually transformed to a higher layer-output correlation in Figure 2(b).

Upon completion of the training phase, it is anticipated that the final layer will be ultimately trained to exhibit a high correlation with the targeted subspace, which will in turn assure high accuracy. Therefore, the optimization metric for parameter/structural learning must fully reflect such subspace correlation.

Least square error formulation: Without rank constraints on $\mathbf{H}^T$. The **Least Square Error (LSE)** has been a classic metric for correlation analysis in statistics and estimation theory. LSE offers an effective score to evaluate the correlation between a data input $\mathbf{x} \in \mathfrak{R}^M$ and the targeted output $y \in \mathfrak{R}^L$, where M and L denote the input and output dimensions. Let B denote the batch size, then the input and output data can be represented by their respective data matrices:

$$\mathbf{X} \in \mathfrak{R}^{M \times B} \text{ and } \mathbf{Y} \in \mathfrak{R}^{L \times B}.$$

The LSE problem is to find an optimal projection matrix $\mathbf{H}$ to minimize the LSE:

$$\mathcal{E} = \text{LSE} = \|\mathbf{Y} - \mathbf{H}^T\mathbf{X}]\|_F^2 = \text{trace}\left([\mathbf{Y} - \mathbf{H}^T\mathbf{X}]\ [\mathbf{Y} - \mathbf{H}^T\mathbf{X}]^T\right), \tag{1}$$

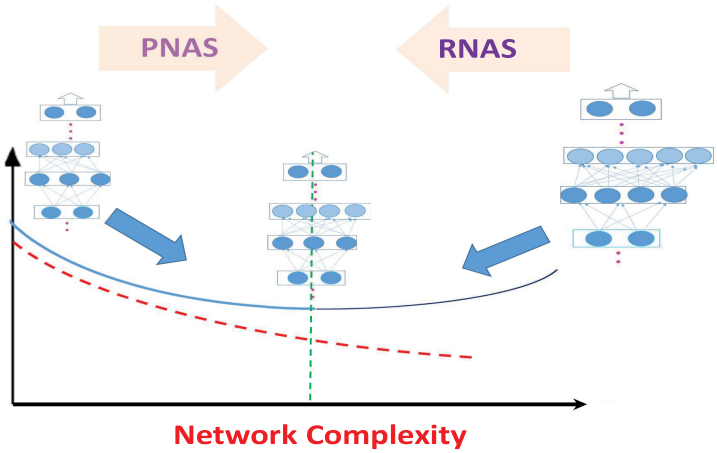
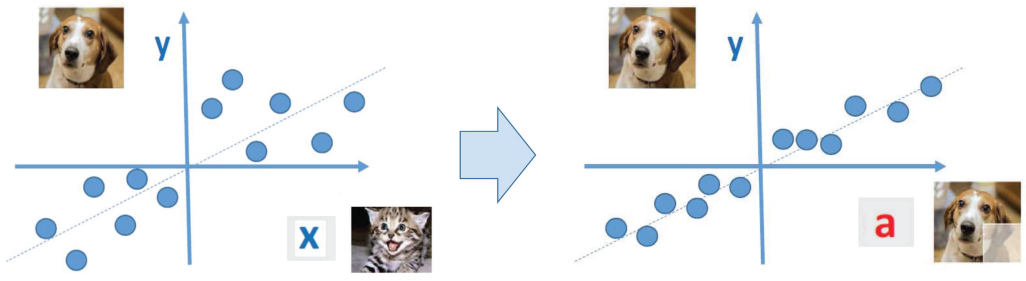


Fig. 1. A U-shaped structural learning curve: generalization error vs. the network size. Left-hand-side region: Networks need to be grown in size (i.e., PNAS). Right-hand-side region: Networks need to be pruned in size (i.e., RNAS).



(a) (b)

Fig. 2. Theoretically, our NAS design objective hinges upon maximization of a desired input/output and layer/output correlation, respectively denoted as $< \mathbf{x}, \mathbf{y} >$ and $< \mathbf{x}, \mathbf{y} >$. (a) Initially, the input/output correlation $< \mathbf{x}, \mathbf{y} >$ is generally unsatisfactory. (b) Gradually, the hidden layer will be trained to deliver a higher correlation $< \mathbf{x}, \mathbf{y} >$.

where the subscript denotes the Frobenius norm. The optimal estimate of the output is $\hat{\mathbf{Y}} = \mathbf{H}^T\mathbf{X}$, where

$$\mathbf{H}^T = \mathbf{Y}\mathbf{X}^T[\mathbf{X}\mathbf{X}^T]^{-1}. \tag{2}$$

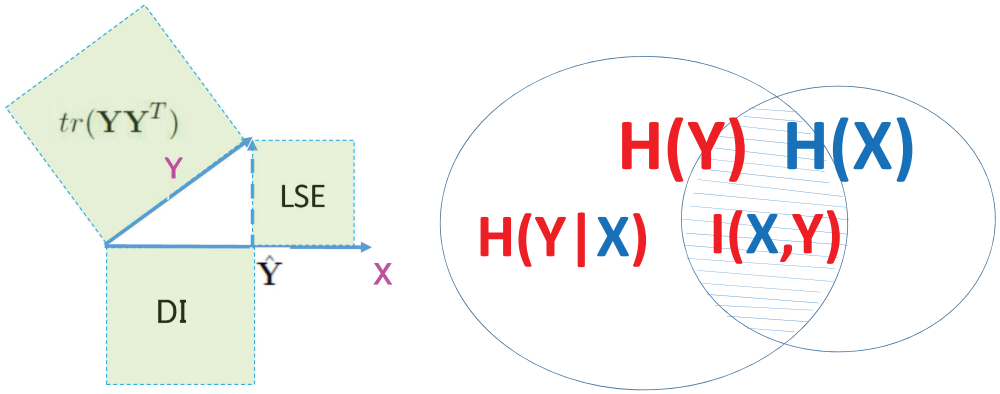
This optimal solution can them be plugged into Equation (1) to obtain

$$\mathcal{E} = \|\mathbf{Y} - \hat{\mathbf{Y}}]\|_F^2 = \|\mathbf{Y}\|_F^2 - \text{trace}\left([\mathbf{X}\mathbf{X}^T]^{-1}[\mathbf{X}\mathbf{Y}^T\ \mathbf{Y}\mathbf{X}^T]\right). \tag{3}$$

For notational convenience, we shall define **Regression Matrix (RM)** as follows:

$$\text{RM} = [\mathbf{X}\mathbf{X}^T]^{-1}(\mathbf{X}\mathbf{Y}^T\ \mathbf{Y}\mathbf{X}^T). \tag{4}$$

Conservation of the sum of LSE and discriminant information. The preceding trace norm is related to (1) Fisher's "discriminative ratio" in classification as well as (2) Shannon's "mutual information" in information theory [10, 39]. As such, we shall define the **Discriminant Information (DI)** as



(a) (b)

Fig. 3. Two conservation principles are intimately related, with $H(\mathbf{y})$, $I(\mathbf{x}|\mathbf{y})$, and $H(\mathbf{y}|\mathbf{x})$ corresponding respectively to $\|\mathbf{Y}\|_F^2$, DI, and LSE. (a) In estimation theory, the conservation between DI and LSE is $DI + LSE = \|\mathbf{Y}\|_F^2$. (b) Likewise, in information theory, $I(\mathbf{x}, \mathbf{y}) + H(\mathbf{y}|\mathbf{x}) = H(\mathbf{y})$.

follows [20–22]:

$$DI \equiv \text{trace (RM)}.$$

Equation (3) offers an important equality that

$$\|\mathbf{Y}\|_F^2 = LSE + DI. \tag{5}$$

This is illustrated by Figure 3(a). This suggests the conservation principle that for any observation $\mathbf{X}$, the sum of LSE and the $\mathbf{Y}$-relevant information embedded within $\mathbf{X}$ is always equal to the total "entropy" embedded in $\mathbf{Y}$. There are two closely related types of conservation principles depicted by Figure 3. Under some statistical assumptions, including normalized output-variance [21], they become interchangeable with each other. In this case, LSE and DI share the same role as $H(\mathbf{y}|\mathbf{x})$ and $I(\mathbf{x}, \mathbf{y})$, respectively.

Traditionally, LSE often serves as a primary optimization criterion in machine learning. Now, thanks to Equation (5), the LSE minimization problem can now be equivalently formulated as a DI maximization problem. Moreover, DI is intimately related to our subsequent subspace analysis (for structural learning) due to the fact that DI can be naturally subdivided into its eigen-components:

$$\text{DI} = \sum_{i=1}^{M} \lambda_i, \tag{6}$$

where $\{\lambda_i, i = 1, \ldots, M\}$ denote the eigenvalues of RM.

Equation (6) reflects a full DI corresponding to the full-rank regression analysis. In the situation that a condition of reduced rank is imposed, then it becomes a problem of subspace (regression) analysis. Moreover, it can be shown that the eigen-subdivision in DI plays a vital role in NAS structural learning. For NAS, it is vital to address key questions such as "which layer to prune/grow the model," "'how to prune/grow each layer," and "by how much." DI serves well as an appealing **Structural Optimization Metric (SOM)** for structural learning. More exactly, DI serves as an effective objective function guiding us to best augment/shrink a hidden layer. This will be elaborated further in our subsequent discussion.

2.2 Subspace Theory for RNAS

A main concern on deep learning via **Back-Propagation (BP)** has to do with the curse of dimensionality of both depth and width. This can be attributed to a phenomenon known as exploding/vanishing gradients with deep and/or wide networks. Dimension reduction is vital, if not imperative, because the curse may be somewhat mitigate when a hidden layer narrows. (Note that the *condition number* associated with a layer tends to deteriorate with increasing width [11]. Generically speaking, a high dimensionality in feature spaces (i.e., the number of neurons in a layer) usually means high computational complexity and power consumption in both the (offline) learning and (online) prediction phases. Moreover, it will become more vulnerable to data overfitting, which could further jeopardize the generalization performance. This offers a rather compelling motivation for us to consider the RNAS strategy.

2.2.1 Unsupervised PCA for Dimension Reduction. A popular subspace approach to dimension reduction is PCA, which can be used to effectively compress the original input data $\mathbf{X}$ and then later reconstruct it optimally. From RM's perspective, PCA represents a special situation that $\mathbf{X} = \mathbf{Y}$ in Equation (4). In this case, RM is degenerated into a symmetric covariance matrix:

$$\text{RM} = [\mathbf{X}\mathbf{X}^T]^{-1}\left(\mathbf{X}\mathbf{X}^T\ \mathbf{X}\mathbf{X}^T\right) = \mathbf{X}\mathbf{X}^T.$$

Note that PCA($\mathbf{X}$) is typically formed from the principal eigenvectors of $\mathbf{X}\mathbf{X}^T$.

2.2.2 Supervised RCA for Dimension Reduction. For the supervised scenarios, the subspace analysis for PCA needs to be extended to a notion termed *RCA*. RCA is the best subspace representation of $\mathbf{X}$, denoted by $\hat{\mathbf{Y}} = \mathbf{H}^T\mathbf{X}$, with $\mathbf{H}^T = \mathbf{F}\mathbf{U}$, which can be used to optimally reconstruct $\mathbf{Y}$. The RCA formulation minimizes the loss function (cf. Equation (1)):

$$\mathcal{L}(\mathbf{F}, \mathbf{U}) = \| \mathbf{Y} - \mathbf{F}\mathbf{U}\mathbf{X} \|_F^2, \tag{7}$$

where the subscript F denotes the Frobenius norm and $\mathbf{F} \in \Re^{L\times m}$ and $\mathbf{U} \in \Re^{m\times M}$ with $m \leq \min\{M, L\}$. The optimal RCA solution involves a two-stage process in **min-min** optimization:

$$\min_{\mathbf{F},\mathbf{U}} \mathcal{L}(\mathbf{F}, \mathbf{U}) = \min_{\mathbf{U}} \left[\min_{\mathbf{F}} \mathcal{L}(\mathbf{F}, \mathbf{U})\right]. \tag{8}$$

Note that its inner optimizer (for any fixed matrix $\mathbf{U}$) can be solved via the conventional LSE method (cf. Equation (2)). Thus, we have

$$\mathbf{F} = [\mathbf{U}\mathbf{X}\mathbf{X}^T\mathbf{U}^T]^{-1}\mathbf{U}\mathbf{X}\mathbf{Y}^T. \tag{9}$$

Plugging Equation (9) into Equation (7), we arrive at

$$\mathcal{L}(\mathbf{F}, \mathbf{U}) = \|\mathbf{Y}\|_F^2 - tr\left[\left(\mathbf{U}[\mathbf{X}\mathbf{X}^T]\mathbf{U}^T\right)^{-1}\left(\mathbf{U}\mathbf{X}\mathbf{Y}^T\ \mathbf{Y}\mathbf{X}^T\mathbf{U}^T\right)\right], \tag{10}$$

whose minimum can be attained by setting $\mathbf{U}$ as follows:

$$\mathbf{U} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]^T, \tag{11}$$

where $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ are the m principal eigenvectors of the RM, $[\mathbf{X}\mathbf{X}^T]^{-1}\left(\mathbf{X}\mathbf{Y}^T\ \mathbf{Y}\mathbf{X}^T\right)$ (cf. Equation (4)). Thus, we have the following RCA theorem.

Theorem 1 (RCA Theorem: RCA($\mathbf{X}$,$\mathbf{Y}$)). *The optimal RCA estimate can be expressed as* $\hat{\mathbf{Y}} = \mathbf{H}^T\mathbf{X}$, *where* $\mathbf{H}^T = \mathbf{F}\mathbf{U}$ *and*

- *According to Equation (11),* $\mathbf{U} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]^T$.
- *Thereafter, via Equation (9), we have* $\mathbf{F} = \mathbf{Y}\mathbf{X}^T\mathbf{U}^T[\mathbf{U}\mathbf{X}\mathbf{X}^T\mathbf{U}^T]^{-1}$.

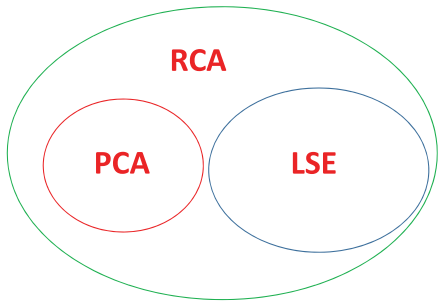


Fig. 4. Two vital special cases of RCA: (1) PCA (when $\mathbf{X} = \mathbf{Y}$) and (2) LSE (when $\mathbf{H}$ is fully ranked). Historically, Adrien-Marie Legendre and Carl Friedrich Gauss were both the recognized pioneers of the least squares method in the early 19th century. However, Gauss was also credited for systematically laying computational and theoretical foundation for LSE method. As to PCA, while the full phrase "Principal Component Analysis" was coined by Hoteling [49], its theoretical foundation PCA was first developed by Pearson [48].

Plugging Equations (9) and (11) into Equation (10), we have

$$LSE_{RCA} = \|\mathbf{Y}\|_F^2 - \sum_{i=1}^{m} \lambda_i. \tag{12}$$

Let us study a simple numerical example to help illustrate the RCA algorithm.

Example 1 (LSE, RCA). Let the input and output data matrices be given as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & 0.1 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 & 0 \end{bmatrix}$$

(a) *LSE solution*: The optimal LSE solution (with a full rank $m = 2$) is $\mathbf{H}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, which leads to an LSE of $\text{LSE}_{LSE} = 8$.

(b) *PCA solution*: The eigenvalues and eigenvectors of $\text{RM} = \begin{bmatrix} 1 & 1 \\ 1 & 10 \end{bmatrix}$ are

$$\lambda_1 = 10.11,\ \mathbf{v}_1 = [0.1212\ 0.9926\ ]^T;\ \text{and}\ \lambda_2 = 0.89, \mathbf{v}_2 = [-0.9823\ 0.1873]^T.$$

Thereafter, via Equations (9) and (11), the optimal rank-1 projection matrix can be obtained as

$$\mathbf{H}^T = \mathbf{FU} = \begin{bmatrix} 0.1204 & 0.9859 \\ 0.3254 & 2.6652 \end{bmatrix}.$$

This ultimately leads to the following estimation error: $\text{LSE}_{RCA} = \text{LSE}_{LSE} + \lambda_2 = 8 + 0.89 = 8.89$.

Figure 4 depicts two important special cases of RCA: (1) PCA (PCA is the same as RCA when $\mathbf{X} = \mathbf{Y}$) and (2) LSE (RCA becomes LSE when $\mathbf{H}$ is fully ranked). The latter leads to the conventional LSE of $\text{LSE}_{LSE} = \|\mathbf{Y}\|_F^2 - \sum_{i=1}^{M} \lambda_i$, since all of the eigen-components are fully accounted for.

2.3 Subspace Analysis for PNAS

Note that RCA can be further extended to ERCA for PNAS (Figure 5).

2.3.1 PCA Extraction for Unsupervised PNAS. For NAS, it is common practice that only a small number of neurons should be incremented in each step to assure safer convergence and better control on the hardware cost. As depicted in Figure 6,

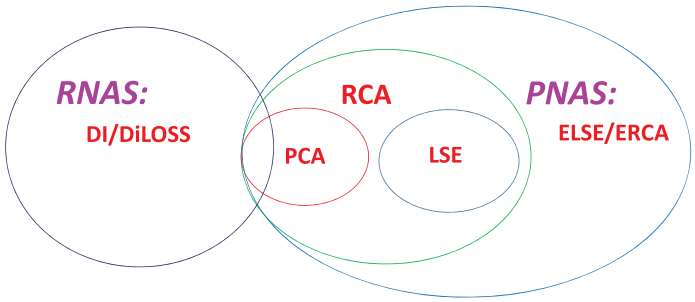


Fig. 5. Extension of RCA to (1: left-hand-side) DiLOSS neuron selection/pruning and (2: right-hand-side) ERCA for PNAS.

- Our PNAS approach proposes to relay and concatenate information in the lower layer to (1) the next layer or (2) directly to the top layer, just like U-Net or DenseNet [19, 35].
- Unlike U-Net or DenseNet, only critical and informative (principal) components are being relayed forward in the PNAS, thus effectively controlling unwanted growth of the width of hidden layers.

In the traditional PCA, the extracted components are expressed as $\mathbf{X}' = \text{PCA}(\mathbf{X})$. In information-theoretical denotation, they will be more conveniently expressed as

$$\mathbf{X}' = I_{CA}(\mathbf{X}),$$

where the subscript *CA* stands for component analysis and *I* stands for information or mutual information.

For PNAS design, there are more suitable components (than PCA) to be considered:

- *Unsupervised component extraction*: $I_{CA}(\mathbf{X}, \mathbf{A}^{\perp})$. To not duplicate information that might overlap with $\mathbf{A}$, we should extract $I_{CA}(\mathbf{X}, \mathbf{A}^{\perp})$ (cf. Figure 6(a)), where $\mathbf{A}^{\perp}$ denotes the complement subspace of span[$\mathbf{A}$].
- *Supervised component extraction*: $I_{CA}(\mathbf{X}, \mathbf{Y}, \mathbf{A}^{\perp})$. The objective of supervised learning for PNAS is to augment the original vector space (associated with the hidden layer) to best cover the vector space spanned by $\mathbf{Y}$. To this end, the teacher's value should be incorporated into the formula of the mutual information (i.e., $I_{CA}(\mathbf{X}, \mathbf{Y}, \mathbf{A}^{\perp})$) for our component analysis. See Figure 6(b).

Note that solutions for both formulations can be derived from the principal eigenvectors pertaining to their respective RM.

2.3.2 Extended PCA Extraction for Unsupervised PNAS. **Extended PCA (EPCA)** can be viewed as a generalized variant of PCA. From the information perspective, it can be viewed as the component analysis for $I_{CA}(\mathbf{X}, \mathbf{A}^{\perp})$. Let us denote the *input matrix* and *neuron data matrix* in a hidden layer by $\mathbf{X}$ and $\mathbf{A}$, respectively. EPCA takes into account the information already existing in the original hidden layer: $\mathbf{A}$. It would be wasteful if any resource is spent on the information that overlaps with $\mathbf{A}$. From the subspace's perspective, to circumvent the said redundancy, we first derive $\tilde{\mathbf{X}}(= \text{Res}(\mathbf{X}|\mathbf{A}))$—that is, the residue of $\mathbf{X}$ after removal the subspace spanned by $\mathbf{H}$. More precisely,

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{Z}\mathbf{A}, \qquad \text{where } \mathbf{Z} = \mathbf{X}\mathbf{A}^T[\mathbf{A}\mathbf{A}^T]^{-1}. \tag{13}$$

It follows that the component analysis for $ICA(\mathbf{X}, \mathbf{A}^{\perp})$ can be expressed as follows:

$$\mathbf{X}' = \text{PCA}(\tilde{\mathbf{X}}).$$

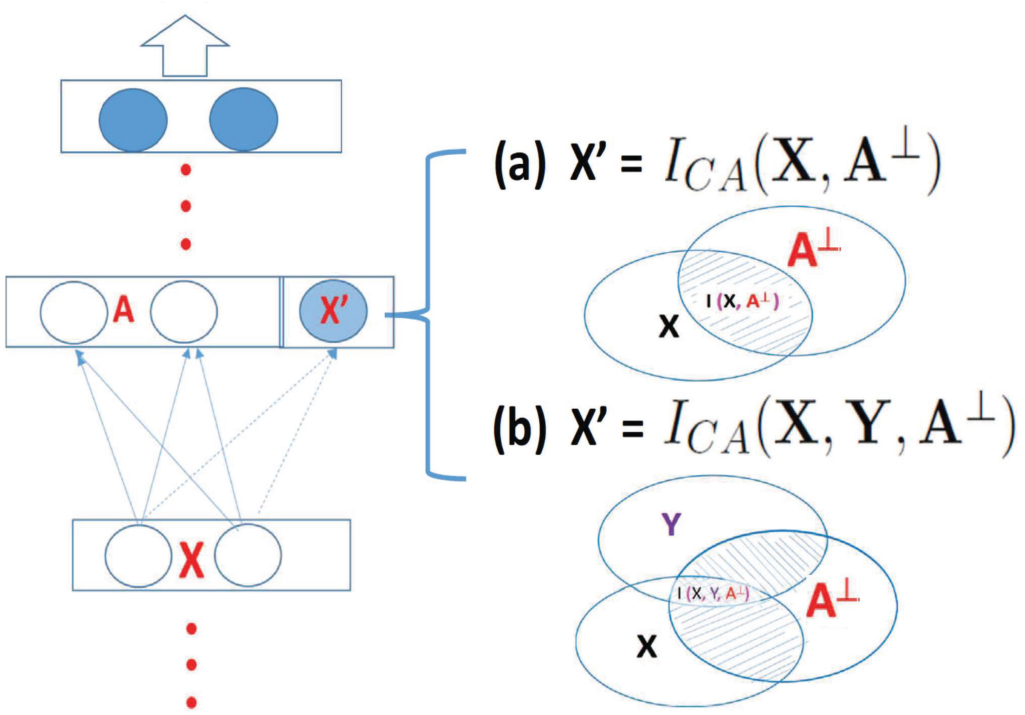


Fig. 6. Illustration of a concatenated hidden layer for PNAS structural learning. It concatenates the key information extracted from the in lower layer and appends it to either (1) the next adjacent layer or (2) directly to the top layer. Two kinds of critical components are under our consideration: (a) unsupervised EPCA (principal components of $I_{CA}(\mathbf{X}, \mathbf{A}^{\perp})$ are extracted) and (b) supervised ERCA (principal components of $I_{CA}(\mathbf{X}, \mathbf{Y}, \mathbf{A}^{\perp})$ are extracted).

LSE analysis. Let us denote an expanded matrix as follows:

$$\Phi \equiv \begin{bmatrix} \mathbf{A} \\ \mathbf{X}' \end{bmatrix}. \tag{14}$$

Corresponding to the expanded data matrix (from $\mathbf{X}'$ to Φ), we now want to find the optimal $\mathbf{W}$ to minimize $|\mathbf{Y} - \hat{\mathbf{Y}}]\|_F^2 = \|\mathbf{Y} - \mathbf{W}^T\Phi]\|_F^2$.

2.3.3 Extend RCA to ERCA for Supervised PNAS Design. The objective of supervised subspace learning for PNAS is to find the optimal component(s) of $\mathbf{X}$ to augment the hidden layer $\mathbf{A}$ so that the combined space can best cover $\mathbf{Y}$. Just like EPCA is used for unsupervised PNAS, ERCA is its counterpart for supervised PNAS. The ERCA algorithm consists of two stages:

- *Computation of the output residual matrix:* Let $\mathbf{Y}$ denote the *output matrix* (or *teacher matrix*) and $\mathbf{A}$ the *hidden-layer data matrix*. Let $\tilde{\mathbf{Y}}$ represent the *output residual matrix*:

$$\tilde{\mathbf{Y}} = \text{Res}(\mathbf{Y}|\mathbf{A}) = \mathbf{Y} - \mathbf{G}\mathbf{A}, \qquad \text{where } \mathbf{G} = \mathbf{Y}\mathbf{A}^T[\mathbf{A}\mathbf{A}^T]^{-1}, \tag{15}$$

  which is orthogonal to $\mathbf{A}$. Namely, $\tilde{\mathbf{Y}}$ eludes any information that has to do with $\mathbf{A}$.
- *Computation of the optimal component(s) to supplement* $\mathbf{A}$*:* For a more concise expression, let us denote

$$I(\mathbf{X},\ \mathbf{Y}, \mathbf{A}^{\perp}\ ) = I(\mathbf{X},\ I(\mathbf{Y}, \mathbf{A}^{\perp})\ ) = I(\mathbf{X}, \tilde{\mathbf{Y}}),$$

where $\tilde{\mathbf{Y}}$ represents the information embedded in $I(\mathbf{Y}, \mathbf{A}^{\perp})$. In short,

$$\text{ERCA} = I_{CA}(\mathbf{X}, \tilde{\mathbf{Y}}) = \text{RCA}(\ \mathbf{X},\ \tilde{\mathbf{Y}}\ ).$$

Theorem 2 (ERCA Algorithm). *The ERCA components can be derived via the RCA algorithm (cf. Theorem 1) as follows:*

$$\mathbf{X}' = \mathit{RCA}(\ \mathbf{X},\ \tilde{\mathbf{Y}}\ ). \tag{16}$$

Let the optimal LSE estimate of $\tilde{\mathbf{Y}}$ *be denoted as*

$$\hat{\tilde{\mathbf{Y}}} = \mathbf{H}^T\mathbf{X} = \mathbf{F}\,\mathbf{U}\mathbf{X}, \tag{17}$$

where $\mathbf{F} \in \Re^{L\times m}$ *and* $\mathbf{U} \in \Re^{m\times M}$, *with* $m \le min(L, M)$. *Similar to Equation (11), the optimal* $\mathbf{U}$ *can be formed as*

$$\mathbf{U} = [\mathbf{v}'_1 \cdots \mathbf{v}'_m]^T, \tag{18}$$

where $\mathbf{v}'_i$ *denotes the i-th eigenvector associated with*

$$RM_{ERCA} \equiv (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}\mathbf{X}^T).$$

The ERCA components can be expressed as $\mathbf{X}' = \mathbf{U}\mathbf{X}$. *Moreover, we have*

$$\mathbf{F}^T = (\mathbf{U}[\mathbf{X}\mathbf{X}^T]\mathbf{U}^T)^{-1}\mathbf{U}\mathbf{X}\tilde{\mathbf{Y}}^T. \tag{19}$$

LSE estimation. The optimal estimate of $\mathbf{Y}$ *can be derived as*

$$\hat{\mathbf{Y}} \underset{Equation\,(15)}{=} \mathbf{G}\mathbf{A} + \hat{\tilde{\mathbf{Y}}} \underset{Equation\,(17)}{=} \mathbf{G}\mathbf{A} + \mathbf{H}^T\mathbf{X} = \mathbf{G}\,[\mathbf{A}] + \mathbf{F}\,[\mathbf{U}\mathbf{X}]. \tag{20}$$

Error analysis. The estimate yields the following LSE:

$$LSE^{(m)}_{ERCA} = tr(\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T) - \sum_{i=1}^{m}\lambda'_i, \tag{21}$$

where λ'_i *denotes the i-th eigenvalue associated with* RM_{ERCA}. □

Example 2 (ELSE, EPCA, and ERCA). In addition to the known matrices $\mathbf{X}$ and $\mathbf{Y}$ already given in Example 1, now we also have

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

- *Extended LSE algorithm*: This can be viewed as an **Extended LSE (ELSE)** problem, where $\mathbf{Y}$ will be estimated from $\mathbf{X}$ and $\mathbf{A}$ via the classic LSE estimator. In this case, the combined rank of $\mathbf{X}$ and $\mathbf{A}$ is $2 + 2 = 4$. The ELSE solution can be derived as follows:

  $$\mathbf{G}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \mathbf{H}^T = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}.$$

  The optimal ELSE estimation of $\mathbf{Y}$ is

  $$\hat{\mathbf{Y}} = \mathbf{G}^T\mathbf{A} + \mathbf{H}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 3 & 2 & 1 & 0 \end{bmatrix}.$$

  Finally, the ELSE error amounts to $\text{LSE}_{ELSE} = 1$.

- *EPCA algorithm (unsupervised augmentation)*:

$$EPCA(\mathbf{X}\,|\mathbf{A},\ m=1) \text{ (i.e., total rank = 2 + } m = 3)$$

It can be shown that (details omitted) (via unsupervised EPCA) is $\mathbf{v}_1 = [-0.9910\ -0.1338\ ]^T$, leading to the following optimal (unsupervised) estimate of $\mathbf{Y}$:

$$\hat{\mathbf{Y}} \underset{Equation\ (20)}{=} \begin{bmatrix} 0.9701 & 1.1205 & 0.8795 & 1 & 0 \\ 0.1205 & 2.5150 & 2.4850 & 1 & 0 \end{bmatrix}. \tag{22}$$

This results in an LSE score of $\text{LSE}_{EPCA} = 1.5150$, which falls within the range of rank-3 scores prescribed bounds by Lemma 1. (See $\diamond$ in Figure 8 (shown later).)

- *ERCA algorithm (supervised augmentation)*:

$$ERCA(\mathbf{X},\mathbf{Y}\,|\mathbf{A},\ m=1) \text{ (i.e., total rank = 2 + } m = 3)$$

Via Equation (15), the *output residual matrix* is

$$\mathbf{G} = \begin{bmatrix} 1 & 1 \\ 2.5 & 1 \end{bmatrix}, \tag{23}$$

which leads to

$$\tilde{\mathbf{Y}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0.5 & -0.5 & 0 & 0 \end{bmatrix}.$$

Next, we construct the RM associated with ERCA based on $\mathbf{X}$ and $\tilde{\mathbf{Y}}$:

$$\text{RM} = (\mathbf{X}\mathbf{X}^T)^{-1}(\mathbf{X}\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}\mathbf{X}^T) = \begin{bmatrix} 1 & 0 \\ -0.0682 & 0.2500 \end{bmatrix},$$

with $\lambda_1' = 1$ and $\mathbf{v}_1 = [0.9959\ \ -0.0905\ ]^T$. The optimal augmenting component to $\mathbf{A}$ can thus be derived as

$$\mathbf{X}' = \mathbf{U}\mathbf{X} = \mathbf{v}_1^T\mathbf{X} = [0.9959\ 0\ 0\ 0\ 0\ ]$$

Error analysis. Via Equation (19), we obtain

$$\mathbf{H}^T = \mathbf{F}\mathbf{U} = \begin{bmatrix} 1.0041 & 0 \end{bmatrix} [0.9959\ -0.0905\ ] = \begin{bmatrix} 1 & -0.0909 \\ 0 & 0 \end{bmatrix}. \tag{24}$$

It follows that

$$\hat{\mathbf{Y}} \underset{Equation\ (20)}{=} \mathbf{G}\mathbf{A} + \mathbf{H}^T\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 2.5 & 2.5 & 1 & 0 \end{bmatrix}.$$

leading to $\text{LSE}_{ERCA} = 1.5$.
A numerical illustration of the ERCA PNAS design process is given in Figure 7.

2.4 Lower and Upper Bounds of LSE w.r.t. Ranks

Lemma 1 (RNAS/PNAS Bound Lemma: Ranges of LSE w.r.t. Ranks). *The subspace analysis theoretically reveals the ranges of LSE w.r.t. ranks are as follows:*

- *After reduction of the ranks of the X-layer from M down to m, the range of LSE is as follows:*

$$\textit{Range of LSE} \ = \left[tr(\mathbf{Y}\mathbf{Y}^T) - \sum_{i=1}^{m}\lambda_i,\ tr(\mathbf{Y}\mathbf{Y}^T) - \sum_{i=M+1-m}^{M}\lambda_M \right]. \tag{25}$$

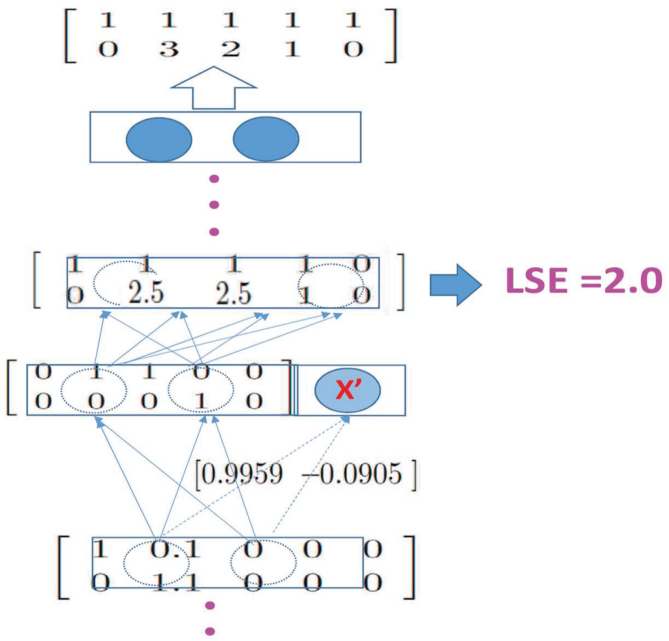


Fig. 7. Numerical Illustration of PNAS design for Example 2: augmented layer via supervised ERCA.

- *After augmentation of the ranks of the A-layer from M up to $M+m$, the range of LSE is as follows (proof omitted):*

$$\textit{Range of LSE} \;=\; \left[\, LSE_A \;-\; \sum_{i=1}^{m} \lambda'_i,\; LSE_A \;-\; \sum_{i=M+1-m}^{m} \lambda'_M \,\right]. \tag{26}$$

Let us revisit Example 2 to help illuminate the usage of Lemma 1.

Example 3. Lemma 1 can be once again verified via the following inequalities: Theoretically, we have (here, r denotes the rank of the space involved)

$$\mathcal{E}_{ELSE}(r=4) \le \mathcal{E}_{ERCA}(r=3) \le \mathcal{E}_{EPCA}(r=3) \le \mathcal{E}_{LSE}(r=2) \le \mathcal{E}_{RCA}(r=1) \le \mathcal{E}_{PCA}(r=1), \tag{27}$$

and, for this example, we have respectively

$$1 \;\le\; 1.5 \;\le\; 1.515 \;\le\; 2.5 \;\le\; 8.89 \;\le\; 17.5, \tag{28}$$

$$\mathcal{E}_{ELSE} = 1 \le \mathcal{E}_{ERCA} = 1.5 \le \mathcal{E}_{EPCA} = 1.515 \le \mathcal{E}_{LSE} = 2.5 \le \mathcal{E}_{RCA} = 8.89 \le \mathcal{E}_{PCA} = 17.5, \tag{29}$$

where the PCA's error 17.5 is based on the best estimate of $\mathbf{Y}$ via rank-1 PCA of $\mathbf{X}$. Finally, the inequalities and the lower/upper bounds of LSE are illustrated in Figure 8.

3 RNAS/PNAS DESIGNS FOR CNNS

CNN represents a new generation of neural network, which adopts learnable convolution operators so as to take into account the neighborhood sensitivity inherent in 1D and 2D signal/image processing applications [25]. Note that the matrix notations we used for the *net function* of a layer in MLP can be represented as $\mathbf{u}^{(l)} = \mathbf{W}^{(l)}\,\mathbf{a}^{(l-1)}$, where $\mathbf{W}$, $\mathbf{a}^{(l-1)}$ and $\mathbf{u}^{(l)}$ denote the weigh matrix, input channels, and net values pertaining to the l-th layer. WLOG, we shall place our focus on the first layer—that is, $l = 1$ (i.e., the $(l-1)$-th and l-th layers respectively represent the input

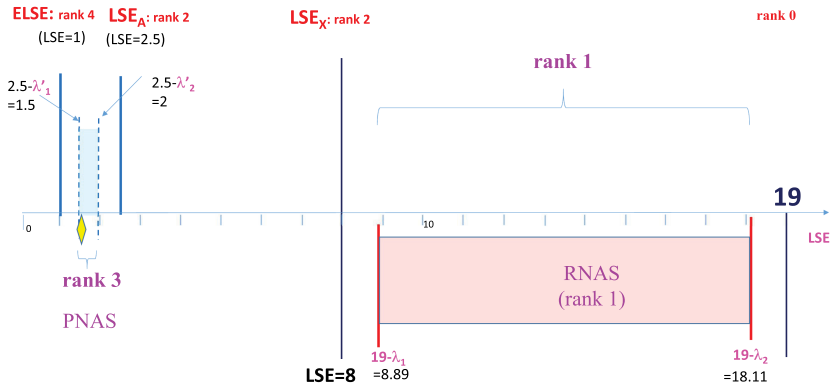


Fig. 8. Lower and upper bounds of the LSE scores with respect to different ranks. (Here, "◇" indicates LSE_{EPCA}.)

and first layers). In this case, the input vector is $\mathbf{a}^{(0)} = \mathbf{x}$. Note that a basic ConvNet in CNN is structurally similar to MLP, except that the multiplication-based operations in MLP are being substituted by convolution-based operations in CNN. More exactly, each link now performs a convolution operation:

$$\mathbf{u} = \mathbf{W} * \mathbf{x}, \tag{30}$$

where (and from now on) we further drop the layer indices for notational convenience.

3.1 Toeplitz-Matrix Representations for the Entire Layer of CNNs

Our approach to the unification between MLP and CNN is via observing that the convolution operators can be represented by a matrix-vector multiplication.

3.1.1 Single-Channel Toeplitz-Matrix Representations. For CNNs, the convolution in one individual link is

$$\sum_{\nu=0}^{\omega-1} w(\nu)x(\mu-\nu), \quad \text{where } \nu = 1, \ldots, d-\omega+1, \tag{31}$$

where ω denotes the length of the 1D kernel. Note that the output of the convolution can be represented by the following matrix-vector multiplication:

$$\mathbf{y} = \mathcal{X}^T \mathbf{w} = \mathcal{X}^T \begin{bmatrix} w(1) & w(2) & \ldots & w(\omega) \end{bmatrix}^T, \tag{32}$$

where $\mathcal{X}$ denotes its **Input Toeplitz Expanded Matrix (ITEM)**:

$$\mathcal{X} = \begin{bmatrix} x(1) & x(2) & \cdots & x(d-\omega+1) \\ x(2) & x(3) & \vdots & x(d-\omega+2) \\ \vdots & \vdots & \vdots & \vdots \\ x(\omega) & x(\omega+1) & \cdots & x(d) \end{bmatrix} \tag{33}$$

Example 4 (Equivalence of Two Matrix Representations). In this example, the weight vector and input vectors respectively are

$$\mathbf{w} = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}^T \text{ (i.e., } \omega = 3)$$

and

$$\mathbf{x} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}^T \text{ (i.e., } d = 6).$$

Pursuant to Equation (31), we have

$$\mathbf{x} * \mathbf{w} = \begin{bmatrix} 28 & 40 & 52 & 64 \end{bmatrix}^T.$$

However, the Toeplitz array is as follows:

$$\mathcal{X} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{bmatrix}.$$

It follows that the link-wise response can be derived by vector-matrix multiplication:

$$\mathbf{y} = \mathcal{X}^T \mathbf{w} = \begin{bmatrix} 28 & 40 & 52 & 64 \end{bmatrix}^T,$$

which is is equivalent to what derived via convolution in Equation (31).

3.1.2 Matrix Representation for an Entire CNN Layer. Let C_{in} and C_{out} respectively denote the number of channels into and out of the layer under our study. Collectively, for multi-channel CNN, an expanded matrix form, such a mapping performed by a ConvNet layer at any time instant, say t, can be collectively expressed as follows:

$$\begin{aligned} \mathcal{Y} &= \overrightarrow{\mathbf{W}} \, \mathbb{X}_{ITEM} \\ &= \begin{bmatrix} \overrightarrow{\mathbf{w}}_{1,1} & \overrightarrow{\mathbf{w}}_{1,2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{1,C_{in}} \\ \overrightarrow{\mathbf{w}}_{2,1} & \overrightarrow{\mathbf{w}}_{2,2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{2,C_{in}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \overrightarrow{\mathbf{w}}_{C_{out},1} & \overrightarrow{\mathbf{w}}_{C_{out},2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{C_{out},C_{in}} \end{bmatrix} \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \\ \vdots \\ \mathcal{X}_{C_{in}} \end{bmatrix}, \end{aligned} \tag{34}$$

where $\mathcal{Y}$ denotes the net value after the convolution and $\mathcal{X}_i$ denotes the ITEM matrix (cf. Equation (33)) associated with the i-channel, with $i = 1, \ldots, C_{in}$. This can be viewed as a basic system of linear equations for the Kernel-LSE (KLSE) formulation useful for robust deconvolution.

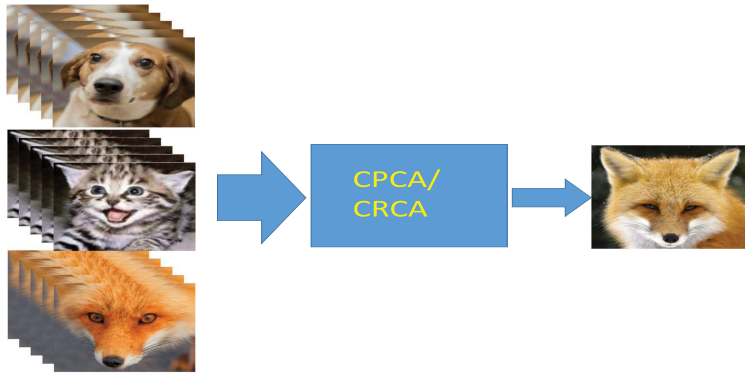
The matrix representation facilitates some sort of unification between MLPs and CNNs. More importantly, it leads to a new kind of component analysis instrumental for subspace analyses of CNNs. This will be elaborated in the subsequent discussion.

3.2 Deep PCA

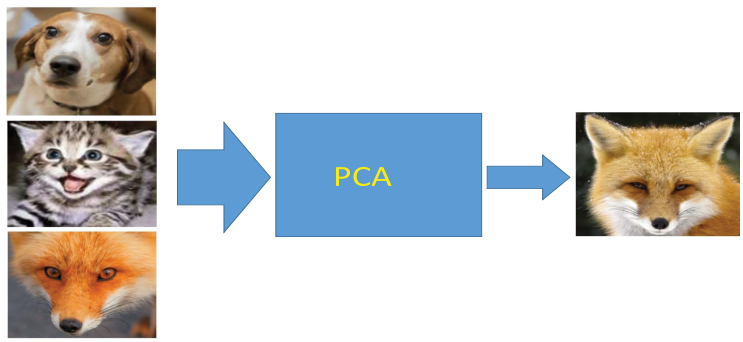
DPCA, also known as Convolutional-PCA (CPCA), combines three types of diversities:

- *Channel diversity*: Thechannel diversity (corresponding to the number of input channels $C_{in} \geq 1$) provides a vital ensemble statistics for various kinds of component analysis (e.g., PCA and RCA).
- *Kernel diversity*: The kernel diversity (represented by the window/kernel size $\kappa \geq 1$) is vital for CNN, as the diversified space may substantially expand the original input vector space.
- *Batch diversity*: DPCA also invokes the batch diversity parameterized by its batch size $B \geq 1$.

DPCA and eigen-channels. With reference to Figure 9(a), DPCA (or eigen-channel) represents a collective representation of multiple channels and batches of 1D or 2D feature maps, exemplified by 1D speeches/music or 2D images/faces. Supposing that m components are to be extracted, the



(a)



(b)

Fig. 9. (a) Eigen-channels are derived by DPCA via three varieties of diversities: kernel diversities (with $\kappa \geq 1$), channel diversity (with the number of input channels $C_{in} \geq 1$), and batch diversity (with the batch size $B \geq 1$). (b) In contrast, eigen-images are derived by PCA via the channel diversity without engaging any kernel diversity. As such, it can be perceived as a special case of eigen-channels with $\omega = 1$.

corresponding DPCA can be expressed as follows:

$$\begin{aligned} \mathbf{y}_{DPCA} &= \overrightarrow{\mathbf{W}}\, \mathbb{X}_{STEM} \\ &= \begin{bmatrix} \overrightarrow{\mathbf{w}}_{1,1} & \overrightarrow{\mathbf{w}}_{1,2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{1,C_{in}} \\ \overrightarrow{\mathbf{w}}_{2,1} & \overrightarrow{\mathbf{w}}_{2,2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{2,C_{in}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \overrightarrow{\mathbf{w}}_{m,1} & \overrightarrow{\mathbf{w}}_{m,2} & \cdots & \cdots & \overrightarrow{\mathbf{w}}_{m,C_{in}} \end{bmatrix} \begin{bmatrix} \mathcal{X}_1(1) & \mathcal{X}_1(2) & \cdots & \mathcal{X}_1(B) \\ \mathcal{X}_2(1) & \mathcal{X}_2(2) & \cdots & \mathcal{X}_2(B) \\ \vdots & \vdots & \vdots & \\ \mathcal{X}_{C_{in}}(1) & \mathcal{X}_{C_{in}}(2) & \cdots & \mathcal{X}_{C_{in}}(B) \end{bmatrix} \end{aligned} \tag{35}$$

The matrix $\mathbb{X}_{STEM}$, referred to as the **Space Time Expanded Matrix (STEM)** associated with 1D or 2D feature maps, has large dimensions both vertically (space-wise) and horizontally (time-wise):

$$\mathbb{X}_{STEM} \in \mathfrak{R}^{\kappa C_{in} \times D'B}$$

Note that 1D and 2D DPCAs involve different STEM-type "space-time" expansions:

- *1D STEM array representation*: For the 1D case (cf. Equations (33) and (34)), we have
  - The "space-dimension" is expanded from C_{in} to κC_{in}, where $\kappa = \omega$.
  - The "time-dimension" is expanded from B to $D'B$, where $D' = d - \omega + 1$.

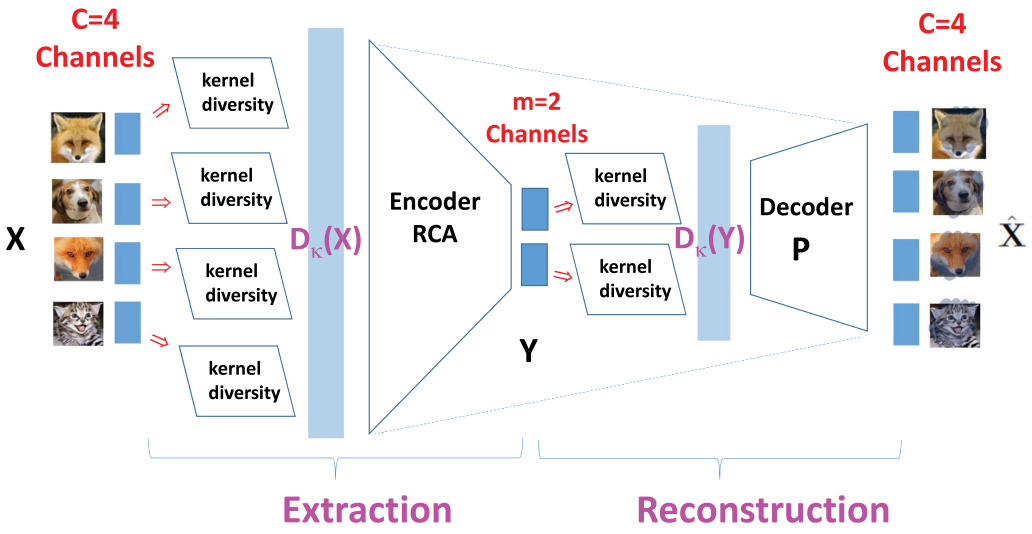


Fig. 10. For RCA-DPCA, the input diversity ($[\mathbf{D}_k\mathfrak{X}]\ ||_F$) is used as input data array. However, the reconstructed space and the LSE are represented/measured by the non-diversified space spanned by $\mathfrak{X}$.

- *2D STEM array representation*: For the 2D case (cf. [6]), we have
  - The "space-dimension" is expanded from C_{in} to κC_{in}, where $\kappa = \omega^2$.
  - The "time-dimension" is expanded from B to $D'B$, where $D' = (d - \omega + 1)^2$.

Two-stage optimization formulation for DPCA. The DPCA is derived from the STEM matrix given in Equation (35). To expand its spanning capability, we note that DPCA effectively invokes twice the diversification: one on the input feature maps (i.e., channels) and another on the extracted components (which by themselves are feature maps as well). The advantage of engaging kernel diversity is algebraically obvious because a STEM-diversified matrix always spans greater vector space than its non-diversified counterpart.

DPCA is based on the following two-stage optimization formulation:

- *Encoder or extraction phase*: Extraction of DPCA.
- *Decoder or reconstruction phase*: Reconstruction from DPCA.

3.2.1 Encoder Phase: Formulation for Extraction of DPCA. The extraction phase of DPCA is pictorially illustrated by Figure 10. Let the optimal estimate be denoted as $\mathfrak{Y} = \mathrm{W}[\mathbf{D}_k\mathfrak{X}]$, and we seek an optimal matrix $\mathbf{W}$, with $\mathbf{W} \in \Re^{C_{in} \times \kappa C_{in}}$ and $\mathrm{rank}(\mathbf{W}) = m \leq C_{in}$, such that

$$\min_{\mathbf{W}} ||\ \mathfrak{X} - \mathbf{W}\ [\mathbf{D}_k\mathfrak{X}]\ ||_F. \tag{36}$$

In this case, the goal is to make $\mathfrak{Y}$ approach $\mathfrak{X}$ as closely as possible. However, $\mathfrak{Y}$ is extracted from $\mathbf{D}_k\mathfrak{X}$ instead of $\mathfrak{X}$. Such an input-output asymmetry necessitates the adoption of the RCA algorithm instead of the conventional PCA. In fact, when ambiguity is a concern, it would be better to explicitly refer to DPCA of such kind as RCA-DPCA.

Alternatively, there may be situations that we would want the estimate, $\mathfrak{Y} = \mathbf{W}[\mathbf{D}_k\mathfrak{X}]$, where $\mathbf{W} \in \Re^{\kappa C_{in} \times \kappa C_{in}}$ and $\mathrm{rank}(\mathbf{W}) = m \leq C_{in}$, to best approach $\mathbf{D}_k\mathfrak{X}$ instead of $\mathfrak{X}$. In this case, the optimization formulation is modified to

$$\min_{\mathbf{W}} ||\ \mathbf{D}_k\mathfrak{X}\ -\ \mathrm{W}[\mathbf{D}_k\mathfrak{X}]\ ||_F \tag{37}$$

(i.e., the input and output are identical to each other). Such a symmetry enables adoption of the traditional PCA (instead of RCA), leading to a PCA-type algorithm named *PCA-DPCA*.

3.2.2 Decoder Phase: Formulation for Reconstruction from DPCA. In the reconstruction phase (cf. Figure 10), it is vital to harness the *kernel diversity* associated with the extracted components: $\mathfrak{Y}$. (Namely, the subspace used for reconstruction should be $\mathbf{D}_k\mathfrak{Y}$ instead of $\mathfrak{Y}$.) Furthermore, there exist two plausible formulations for reconstruction, each with its own justificator:

- *Reconstruction of non-diversified space* $\mathfrak{X}$: In this case, we treat DPCA as an end product. This corresponds to the scenario when the augmenting hidden layer (cf. Figure 6) happens to be the final layer and no more convolutions are anticipated. This calls for a typical LSE formulation to derive the best matrix $\mathbf{P} \in \Re^{C_{in} \times \kappa m}$ such that

$$\min_{\mathbf{P}} || \mathfrak{X} - \mathbf{P} [\mathbf{D}_k \mathfrak{Y}] ||_F. \tag{38}$$

  This type of reconstruction is pictorially illustrated by Figure 10.
- *Reconstruction of diversified space* $\mathbf{D}_k\mathfrak{X}$: In this case, the extracted DPCA is no longer treated as an end product. Therefore, some further convolutions and diversification will be expected. This corresponds to the scenario that the augmented hidden layer (cf. Figure 6) is positioned in the middle and supposed to continue on with further convolution-type processing. In this case, we will need to consider a different LSE formulation, seeking $\mathbf{P} \in \Re^{\kappa C_{in} \times \kappa m}$ for

$$\min_{\mathbf{P}} || \mathbf{D}_k \mathfrak{X} - \mathbf{P}[\mathbf{D}_k \mathfrak{Y}] ||_F. \tag{39}$$

3.3 Performance Comparison: DPCA (i.e., RCA-DPCA) vs. PCA-DPCA

Our simulations cover two types of component extraction methods, RCA-DPCA vs. PCA-DPCA, with comparison made over two types of reconstruction metrics[1]:

- *Reconstruction of non-diversified space* $\mathfrak{X}$: Note that RCA-DPCA is meant to best span $\mathfrak{X}$ according to Equation (36). As such, DPCA is theoretically poised to outperform its PCA-DPCA. Empirically, our simulation confirms that RCA-DPCA holds a noticeable advantage over its PCA-DPCA. In one experiment, we fix $m = 10$ and float $\kappa = 3, \ldots, 14$. As depicted in Figure 11(a), the RCA/PCA **Normalized DI (NDI)** ratio peaks at $\kappa = 10$, where the ratio is ≈ 2.1. We have also experimented with floating m; while fixing $\kappa = 10$, the NDI ratio peaks at $m = 2$ for a high ratio of 2.5 and gradually tapers off to 1.7 when $m = 20$.
- *Reconstruction of diversified space* $\mathbf{D}_k\mathfrak{X}$: It is illuminating to examine the following contrasting observations:
  - The PCA-DPCA is trained with diversification embedded in the target space (cf. Equation (37)). This first diversification is further advanced by yet a second diversification pertaining to $\mathbf{D}_k\mathfrak{Y}$ in Equation (39). All together, the whole process amounts to twice diversification, which exceeds what is really called for.
  - For RCA-DPCA, however, the optimization prescribed by Equation (36) would result in $\mathfrak{X} \approx \text{span}[\mathfrak{Y}]$ under the most idealistic scenario. This would in turn mean that $\mathbf{D}_k\mathfrak{X} \approx \text{span}[\mathbf{D}_k\mathfrak{Y}]$, which matches the diversification embedded in the targeted subspace prescribed by Equation (39).

  Theoretically, the preceding analysis is clearly in favor of RCA-DPCA, which is also empirically verified by the following simulation study. In the first experiment, we fix $m = 10$

[1]The NDI performance metric reported here represents the average statistics over 10 randomized datasets, based on 40 randomly correlated channels. Our comparison between RCA-DPCA and PCA-DPCA will be based on two types of experiments, respectively, with changing diversity ω and component number m.

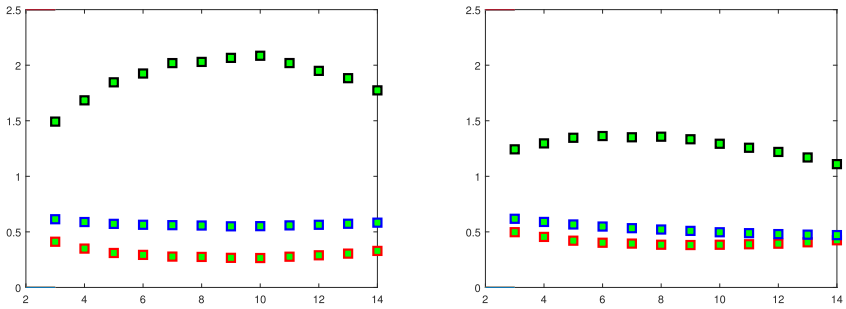


Fig. 11. RCA-type vs. PCA-type DPCAs with the number of components fixed at $m = 10$, and ω ranges from 3 to 14: on reconstruction of $\mathfrak{X}$ (a) and on reconstruction of $\mathbf{D}_k\mathfrak{X}$ (b). (The lowest curve: NDI of PCA-DPCA; middle curve: NDI of RCA-DPCA; highest curve: the NDI ratio: RCA-NDI/PCA-NDI.)

and float $\kappa = 3, \ldots, 14$. As depicted in Figure 11(b), the RCA/PCA NDI ratio peaks at $\kappa = 7$, where the ratio is ≈ 1.4. In the seccond experiment, with floating m and fixed $\kappa = 10$, the NDI ratio remains roughly constant at about 1.25 to 1.3 for the entire range: $m = 2 : 20$.

In conclusion, RCA-DPCA holds significant superiority over PCA-DPCA for both types of targeted reconstruction: either without diversification (i.e., $\mathfrak{X}$) or with diversification (i.e., $\mathbf{D}_k\mathfrak{X}$).

3.4 Comparison of PCA, RCA-DPCA, and PCA-DPCA

Channel diversity for eigen-images/eigenfaces. The development of eigen-images are primarily built upon the channel diversity, without invoking any kernel diversity. With reference to Figure 9(b), an eigen-image represents the collective statistics of multiple images. A set of m principal eigen-images is the result of optimally mapping C_{in} images to m representative images, where C_{in} is the number of input channels.

Let us now focus on the i-th eigen-images (i.e., the i-th PCA component), which is represented by

$$\mathfrak{Y}^{(i)} = \sum_{j=1}^{C_{in}} w_j^{(i)}\, \mathfrak{X}_j = \overrightarrow{\mathbf{w}}^{(i)}\mathfrak{X},\ i = 1, \ldots, m,$$

where $\mathfrak{X} \in \mathfrak{R}^{C_{in} \times D}$ denotes the matrix formed from the flattened vectors—each of dimension D—collected from the C_{in} channels. For the extraction of eigen-images, we adopt the following typical PCA formulation to derive an optimal matrix $\mathbf{W}$, with $\text{rank}(\mathbf{W}) = m \leq C_{in}$, such that

$$\min_{\mathbf{W}} ||\mathfrak{X} - \mathbf{W}^T\mathbf{W}\mathfrak{X}||_F, \text{ where } \mathbf{W} \in \mathfrak{R}^{m \times C_{in}}\ . \tag{40}$$

It is well known that the optimal solution for $\mathbf{W}$ can be formed from the m principal eigenvectors of $\mathfrak{X}\mathfrak{X}^T$.

Comparison of PCA, RCA-type DPCA, and PCA-type DPCA. The overall comparison among these is highlighted by the following table:

	PCA	**RCA-DPCA**	**PCA-DPCA**
Input Kernel Diversity	No	Yes	Yes
Output Kernel Diversity	No	No	Yes
Computational Complexity	$O(C_{in}^3)$	$O(C_{in}^3)$	$O(\kappa^3 C_{in}^3)$

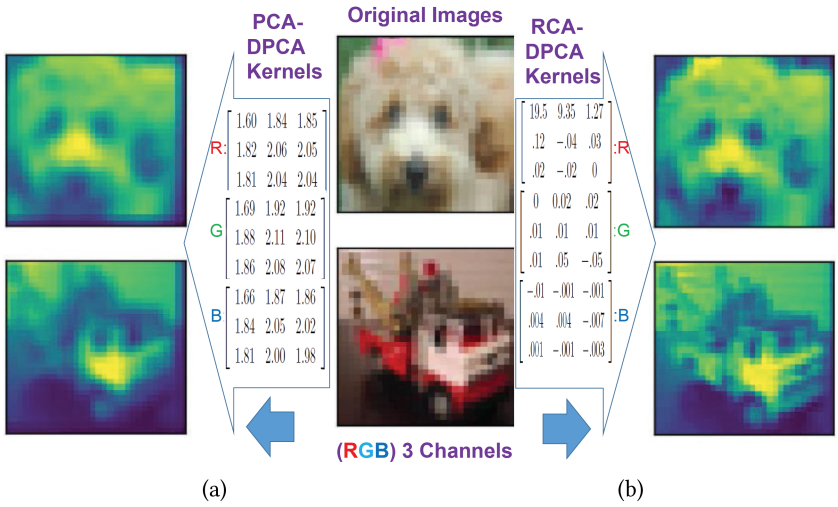


Fig. 12. The original color images are shown in the center. Also displayed are the (RGB) eigen-kernels and their respectively reconstructed images, associated with PCA-DPCA (a) and RCA-DPCA (b).

Theoretical comparison:

- Theoretically, RCA-DPCA outperforms the conventional PCA (used for extracting eigenfaces) because PCA is more restrictive since it is only a special case of RCA-DPCA, when $\kappa = \omega = 1$. As highlighted in the preceding table, PCA and RCA-DPCA actually share the same output target and the only difference is the use/lack of kernel diversity in their respective inputs. In this case, the more diversity, the better, as it can support greater spanning subspace. Thus, their mutual comparison tends to favor RCA-DPCA.
- However, RCA-DPCA and PCA-DPCA share the same input kernel diversity and differ only in the output kernel diversities. In reversal of fortune, more diversity is actually worse for this case. Consequently, the odd is in favor of RCA-DPCA. Note that the diversified output targeted by PCA-DPCA spreads out over a large subspace. Relatively speaking, RCA-DPCA places its focus on a concentrated subspace to facilitate the extraction of truly relevant components. As illustrated by Figure 12 (cf. Example 5), the image reconstructed by RCA-DPCA is visibly superior to PCA-DPCA.

Example 5 (Comparison Between PCA-Type DPCA and RCA-Type DPCA for Color Images). The dataset contains 64 color images, each with RGB channels. This means that the channel diversity is $C_{in} = 3$ and the batch size is $B = 64$. Visually, note the sharp contrast between the reconstructed images displayed in Figure 12(a) and (b) for PCA-DPCA and RCA-DPCA, respectively. Quantitatively, there exists a big gap between the two average PSNRs: 16.65 dB vs. 24.63 dB respectively for PCA-DPCA and RCA-DPCA.

Computational complexity: RCA-DPCA offers computational saving of $O(\kappa^3)$. PCA-DPCA demands a huge computational complexity amounting to $O(\kappa^3 C_{in}^3)$. In contrast, the computational complexity of RCA-DPCA can be drastically reduced to $O(C_{in}^3)$ since it follows the smaller of the two matrix dimensions. Numerically, for the experiment in Example 5, RCA-DPCA is reportedly faster than PCA-DPCA by around 26 folds.

3.5 DPCA for CNN's NAS Design

Just like PCA/EPCA are meant for structural design of MLP, DPCA/EDPCA are good for structural design of CNNs. For the latter, the STEM representation plays a vital role in the derivation of DPCA/EDPCA. Therefore, with the STEM representation, the subspace-based design can then be made amenable to RNAS and PNAS deigns of CNNs. More elaborately:

- For MLPs, optimal PCA (or EPCA) weights $\mathbf{w}$ and components $\mathbf{y}$ are extracted from data matrix $\mathbf{X}$—that is,

$$\mathbf{X} \Rightarrow (\mathbf{w} = \text{eigenvector}, \mathbf{y} = \text{eigen-component}).$$

- For CNNs, optimal DPCA (or EDPCA) kernels $\mathbf{w}$ and channels $\mathbf{y}$ are extracted from the STEM array $\mathbb{X}$—that is,

$$\mathbb{X} \Rightarrow (\mathbf{w} = \text{eigen-kernel}, \mathbf{y} = \text{eigen-channel}).$$

Note further that DPCA is heavily dependent on the order of the convolution kernels (i.e., ω), As such, it is sometimes necessarily denoted as DPCA(ω) for sake of clarity. Moreover, DPCA can be computed from the *m principal eigenvectors* of a STEM array formed by cascading $\mathcal{X}_i(t),\ t = 1, \ldots, B,\ i = 1, \ldots, C_{in}$ over both time and channels. This is now further illuminated via a numerical derivation of an exemplar STEM array in Equation (41), for *DPCA*(3).

Example 6 (Multi-Channel 1D CNN: Derivation of Principal Eigen-Kernels and Eigen-Channels). This example explains how to compute a multi-channel 1D CNN. Assume that there are two input channels with the respective 1D input waveforms being as follows:

- *Input to Channel 1*:

(1) *For t = 1*: $\mathbf{x}_1^{(1)} = \begin{bmatrix} -8 & -5 & -2 & 2 & 5 & 8 \end{bmatrix}^T$.

(2) *For t = 2*: $\mathbf{x}_1^{(2)} = \begin{bmatrix} 4 & -7 & 3 & 1 & -6 & 5 \end{bmatrix}^T$.

(3) *For t = 3*: $\mathbf{x}_1^{(3)} = \begin{bmatrix} -4 & 5 & -9 & 2 & 8 & -1 \end{bmatrix}^T$.

Via Equation (33), they respectively lead to $\mathcal{X}_1^{(1)}$, $\mathcal{X}_1^{(2)}$, and $\mathcal{X}_1^{(3)}$.

- *Input to Channel 2*:

(1) *For t = 1*: $\mathbf{x}_2^{(1)} = \begin{bmatrix} 1 & 2 & 3 & 3 & 2 & 1 \end{bmatrix}^T$.

(2) *For t = 2*: $\mathbf{x}_2^{(2)} = \begin{bmatrix} 2 & -3 & 5 & -4 & 3 & 1 \end{bmatrix}^T$.

(3) *For t = 3*: $\mathbf{x}_2^{(3)} = \begin{bmatrix} 2 & 0 & 3 & 0 & 0 & -4 \end{bmatrix}^T$.

Via Equation (33), they respectively lead to $\mathcal{X}_2^{(1)}$, $\mathcal{X}_2^{(2)}$, and $\mathcal{X}_2^{(3)}$.

Two steps to derive DPCA are as follows:

***Step 1*:** We construct the cascaded STEM matrix $\mathbb{X} \in \Re^{6\times 12}$:

$$\begin{aligned} \mathbb{X} &= \begin{bmatrix} \mathcal{X}_1^{(1)} & \mathcal{X}_1^{(2)} & \mathcal{X}_1^{(3)} \\ \mathcal{X}_2^{(1)} & \mathcal{X}_2^{(2)} & \mathcal{X}_3^{(3)} \end{bmatrix} \\ &= \begin{bmatrix} -8 & -5 & -2 & 2 & 4 & -7 & 3 & 1 & -4 & 5 & -9 & 2 \\ -5 & -2 & 2 & 5 & -7 & 3 & 1 & -6 & 5 & -9 & 2 & 8 \\ -2 & 2 & 5 & 8 & 3 & 1 & -6 & 5 & -9 & 2 & 8 & -1 \\ 1 & 2 & 3 & 3 & 2 & -3 & 5 & -4 & 2 & 0 & 3 & 0 \\ 2 & 3 & 3 & 2 & -3 & 5 & -4 & 3 & 0 & 3 & 0 & 0 \\ 3 & 3 & 2 & 1 & 5 & -4 & 3 & 1 & 3 & 0 & 0 & -4 \end{bmatrix} \end{aligned} \tag{41}$$

The targeted matrix $\mathbb{Y}$ is equal to the top two rows of $\mathbb{X}$.

***Step 2*:** The principal eigenvector of $RCA(\mathbb{X}, \mathbb{Y})$ is

$$\mathbf{u}_1 = \begin{bmatrix} -0.998 & -0.062 & 0.261 & 0.069 & 0.205 & -0.118 \end{bmatrix}^T.$$

It follows that the eigen-kernels can be obtained from the first and second halves of $\mathbf{u}_1$:

(1) *Channel 1*: Eigen-kernel$_1$ $= \begin{bmatrix} -0.9981 & -0.0621 & 0.2609 \end{bmatrix}^T$.
(2) *Channel 2*: Eigen-kernel$_2$ $= \begin{bmatrix} 0.0685 & 0.2049 & -0.1182 \end{bmatrix}^T$.

Thereafter, the three eigen-channels may be derived as follows:

$$\mathbf{y}^{(t)} = \sum_i \text{eigen-kernel}_i \ * \ \mathbf{x}_i^{(t)}, \text{ for } t = 1, 2, \text{ and } 3.$$

Thus, we obtain three output sequences with length $4 = d - \omega + 1$ (one for each time t):

(1) *For t = 1*: $\mathbf{y}^{(1)} = \begin{bmatrix} 7.8972 & 6.0336 & 3.7603 & 0.2776 \end{bmatrix}^T$.
(2) *For t = 2*: $\mathbf{y}^{(2)} = \begin{bmatrix} -3.8437 & 8.3531 & -5.4535 & 0.9015 \end{bmatrix}^T$.
(3) *For t = 3*: $\mathbf{y}^{(3)} = \begin{bmatrix} 1.1162 & -3.2951 & 11.1514 & -2.2811 \end{bmatrix}^T$.

DPCA may be extended to EDPCA and EDRCA to further facilitate PNAS design for CNNs. This is illustrated by the following example.

Example 7 (Extension to EDPCA and EDRCA). This example explains how to compute a multi-channel 1D CNN, assuming that $\omega = 3$, $d = 6$, with the batch size $B = 3$.

- *Compute eigen-kernels and eigen-channels for* $I_{CA}(\mathbb{X}, \mathbb{A}^\perp)$. Assume that there are two channels in the input layer (i.e., $C_{in} = 2$), as well as in the hidden layer (i.e., $C_{hidden} = 2$). First, via Equation (35), we can obtain the STEM array for $\mathbb{X} \in \mathfrak{R}^{6\times 12}$, where $6 = \omega \times C_{in} = 3 \times 2$ and $12 = (d - \omega + 1) \times B = 4 \times 3$. By the same token, we can construct the STEM array for $\mathbb{A} \in \mathfrak{R}^{6\times 12}$. First, we compute $\tilde{\mathbb{X}} = \text{Res}(\mathbb{X}|\mathbb{A})$, then, via the RCA-DPCA algorithm, we can derive $I_{CA}(\tilde{\mathbb{X}})$ as the optimal eigen-channels to supplement the hidden layer $\mathbb{A}$.
- *Compute eigen-kernels and eigen-channels for* $I_{CA}(\mathbb{X}, \mathbb{Y}, \mathbb{A}^\perp)$. By now, we have already computed $\mathbb{X}$ and $\mathbb{A}$. In addition, assuming that there is only one output channel, we can also derive $\mathbb{Y} \in \mathfrak{R}^{3\times 12}$, where $3 = \omega \times C_{out} = 3 \times 1 = 3$. First, we compute $\tilde{\mathbb{Y}} = \text{Res}(\mathbb{Y}|\mathbb{A})$, then, via the DRCA algorithm, we can derive $I_{CA}(\mathbb{X}, \tilde{\mathbb{Y}})$ as the optimal eigen-channels to best augment $\mathbb{A}$.

4 X-LEARNING LEARNING PARADIGM

Based on the subspace analysis, X-learning is proposed as a joint parameter and structural learning paradigm, and it can accommodate both RNAS and PNAS in an interleaved manner. Generally speaking, X-learning can be viewed as a sort of hybrid of RNAS and PNAS. Nevertheless, in the RNAS mode alone, X-learning can also be successfully applied to certain datasets and baselines for some applications [24].

4.1 X-Learning: Augment Deep BP-Parameter Learning with Structural Learning

An evolutionary iterative learning paradigm. Now we are ready to propose an (EM-style) Evolutionary and Iterative X-learning, aiming at jointly optimizing EOM (in the parameter space) and

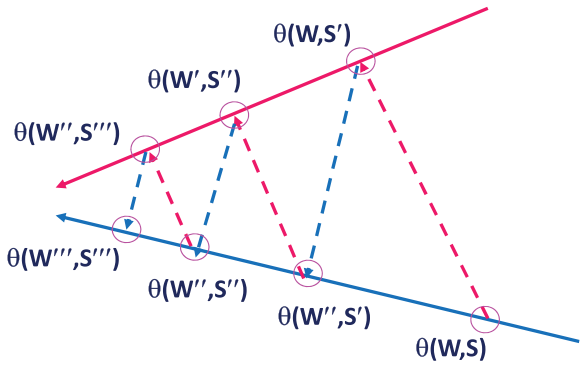


Fig. 13. Illustration of iterations in X-learning.

LOM (in the structural space). The iterations between parameter and structural spaces are pictorially illustrated by Figure 13:

- *Parameter updating phase*: For parameter learning, we shall stay with the BP learning based on the BP of the error gradients. More specifically, the BP parameter updating rule is as follows:

$$\Delta w_{ji}^{(n+1)}(l) = w_{ji}^{(n)}(l+)\Delta w_{ji}^{(n)}(l), \forall \text{ layers}, l = 1, \ldots, L,$$

  where, via the BP chain rule, we have

$$\Delta w_{ji}^{(n)}(l) \quad = \quad -\eta \frac{\partial E}{\partial w_{ji}^{(n)}(l)} = \eta\, \delta_j^{(n)}(l) f'(u_j^{(n)}(l)) a_i^{(n)}(l-1),$$

  where the *error signal* $\delta_j^{(n)}(l) \equiv -\frac{\partial E}{\partial a_j^{(n)}(l)}$.
- *Structural pruning phase*: The local teacher permits the computation of LOM (local optimization metrics), such as DI or DiLOSS, to facilitate the ranking of hidden neurons. Such a ranking provides a theoretical footing of the structural learning stage in the X-learning paradigm.

4.2 Backward Broadcast and Output Residual Learning

X-learning makes the dual use of input **forward skips (FS)** and output **Back-Broadcasting (BB)** (Figure 14). It is well known that learning of deep networks is often severely hampered by the curse of depth. To rectify this problem, the FS have been proposed to allow the input of the previous layer to become accessible as a reference for the next layer [13]. It leads to the *input residual learning* paradigm, which has been very effectively applied to allay the curse of depth.

In addition, we may also incorporate output BB so that the teacher values become directly accessible to all hidden layers. This is conceptually dual to the input residual learning. This is illustrated by Figure 14. As such, the teacher can be viewed as a *local teacher* to facilitate the use of *local SOM* to facilitate *output residual learning*, previously proposed in Section 2.3.3.

Remark. Note that the residue Res($\mathbb{Y}|\mathbb{A}$) associate with a hidden layer reflects the layer-output correlation. The higher the residue, the lower the correlation. Such a residue will have to pass along to the next layers for further learning. The future learning task will only have to focus on reducing the residues associated with the subsequent layers. In this sense, it serves the same purpose of applying input skips, and thus, hopefully, the curse of depth may be somewhat mitigated.

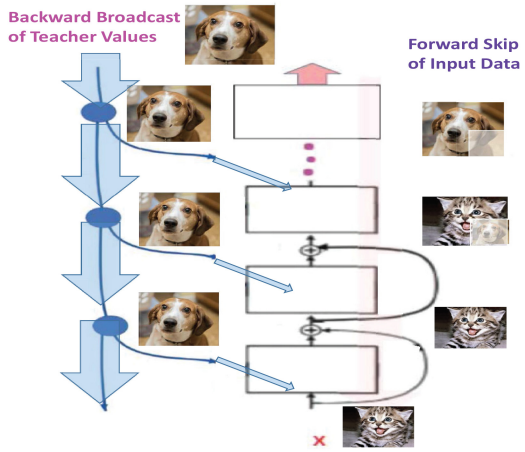


Fig. 14. X-learning employs both input and output residues, via the incorporation of input FS as well as output BB.

4.3 LASSO-Type Optimization and Deleterious Neurons

To ensure convergence of any iterative learning algorithm, it is critical to stick to a vital *minimal updating principle*. More specifically, it allows us to seamlessly inherit most of the existing weight parameters and avoid having to train the entire network from scratch. The DiLOSS pruning scheme is meant for such a strategy, as it can inherit most of the previously trained parameters. As such, it can serve as an effective substitute of RCA for RNAS design. (Recall that RCA requires retraining of all parameters in the layer.) Mathematically, the adoption of a SOM based on LASSO-regularized DI will naturally reveal the critical role of DiLOSS. More importantly, such type of SOM can do well in coercing the network to converge toward an optimal down-sized model. This point is further elaborated in the following.

Node trimming based on LASSO-type LOM. It is known that regularization (e.g., LASSO types) leads to better generalization, for both MLPs and CNNs. For the LASSO type, we propose the following *structural energy function* for both classification and regression.

$$E = DI_{local} - \lambda|\mathbf{a}|_0 \tag{42}$$

For simplicity, let us momentarily assume that we remove one neuron at a time, resulting in the following energy function ($E \equiv$ DI) before/after contrast:

$$E = DI - \lambda\|\hat{\mathbf{a}}\|_0 \text{ and } E' = DI' - \lambda\|\hat{\mathbf{a}}'\|_0 \tag{43}$$

with $\|\hat{\mathbf{a}}\|_0 = N$ and $\|\hat{\mathbf{a}}'\|_0 = N - 1$, where N denotes the number of (non-zero) neurons in the layer.

Let us further assume that what differentiates $\hat{\mathbf{a}}'$ from $\hat{\mathbf{a}}$ is the removal of its i-th neuron. Because DI monotonically decreases with removal of neuron(s), then $\Delta DI_i = DI' - DI \leq 0$. In other words,

$$\Delta\text{DI} = \Delta E = \lambda - |\Delta DI|_i. \tag{44}$$

Note that the LASSO-based metric E in Equation (43) will make a net gain by removing the i-th neuron if and only if

$$|\Delta DI|_i = \text{DiLOSS}_i < \lambda. \tag{45}$$

Therefore, Equation (45) is formally the necessary and sufficient condition that the i-th neuron be labeled as a deleterious neuron.

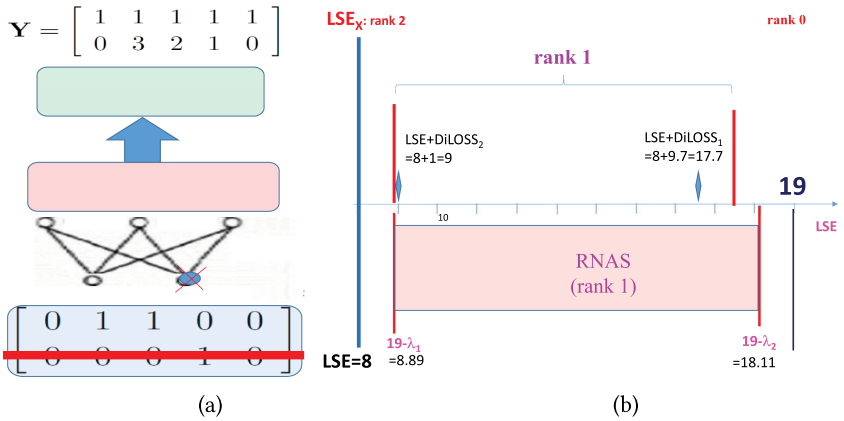


Fig. 15. Illustrations related to Example 8 on applying DiLOSS to RNAS. (a) Because $\text{DiLOSS}_2 < \text{DiLOSS}_1$, we should drop the second input, resulting in a rank-1 input space. (b) The two $\diamond$'s, in the rank-1 range, vindicate the choice of dropping the second input instead of the first input.

In each iteration, upon the completion of its parameter updating phase, we shall commence right away its structural updating phase: those neurons designated as deleterious neurons will now be dropped/pruned. The pruning process will be performed on the carefully selected hidden layers in the network, each layer with its own empirically determined threshold λ.

Example 8 (RNAS via DiLOSS-Based Pruning). Continuing on with Example 1, with the present attention focused on how to apply DiLOSS to drop one of two inputs in the X-layer.

Prune Second Neuron: Since $DiLOSS_2 = DiLOSS_{trim the second neuron} = 1$ is smaller than $DiLOSS_{trim the first neuron} = DiLOSS_1 = 9.7$, the best strategy is to drop the second neuron, leading to

$$\mathbf{X}' = \begin{bmatrix} 0 & 1.1 & 0 & 0 & 0 \end{bmatrix}$$

and the mapping matrix is $\mathbf{H}^T = \begin{bmatrix} 0.909 \\ 2.727 \end{bmatrix}$. with the best estimate obtained as $\hat{\mathbf{Y}} = \mathbf{H}^T\mathbf{X}'$ which in turn results in the following:

$$\text{LSE}_{trim\ 2nd\ node} = \text{LSE}_{LSE} + DiLOSS_2 = 8 + 1 = 9.$$

However, for comparison, by dropping the first neuron, we have

$$\text{LSE}_{trim\ 1st\ node} = \text{LSE}_{LSE} + DiLOSS_1 = 8 + 9.7 = 17.7.$$

Therefore, compared with Figure 15, we have

$$\begin{aligned} \mathcal{E}_{ELSE}\ (m' = 2) \le \mathcal{E}_{ERCA}\ (= 1.146)\ &\le \mathcal{E}_{trim\ 2nd\ input\ of\ A}\ (= 9) \\ &\le \mathcal{E}_{trim\ 1st\ input\ of\ A}\ (= 17.7 \le 18) \le 18.11. \end{aligned}$$

Thus, this example once again vindicates the range covered by rank-1 estimates, as prescribed by Lemma 1.

4.4 Component Analysis via Data-Independent Selection Criterion

Let us now study a case of locally supervised structural learning for RNAS. Now the (local) teacher's role will be taken over by $\mathbf{A} = \mathbf{WX}$, replacing the traditional teacher: $\mathbf{Y}$. For

computational robustness, it is a common practice to incorporate another hyperparameter ρ' into an information-theoretical DI [21] (i.e., output normalized), resulting in

$$\text{DI} = \texttt{tr}\left[\left(\mathbf{U}[\mathbf{X}\mathbf{X}^T + \rho\mathbf{I}]\mathbf{U}^T\right)^{-1}\left(\mathbf{U}\mathbf{X}\mathbf{A}^T\,[\mathbf{A}\mathbf{A}^T + \rho'\mathbf{I}]^{-1}\mathbf{A}\mathbf{X}^T\mathbf{U}^T\right)\right], \tag{46}$$

where $\mathbf{U}$ is a *selection matrix*—that is, each column of $\mathbf{U}$ will be all zeros except one single "1" entry, serving as the selection indicator. In this case, $\mathbf{U}\mathbf{U}^T = \mathbf{I}_m$, assuming no repetitive selection.

Data-Independent Selection Criterion DI_{di}. To become data independent, we want to obviate DI's dependency on the actual input data matrix $\mathbf{X}$. A typical way out is by simply assuming that $\mathbf{X}$ is white (i.e., $\mathbf{X}\,\mathbf{X}^T = \mathbf{I}$). As shown in the following, although the derivation of DI initially engages $\mathbf{X}$ and $\mathbf{A}$, it will ultimately become dependent only on $\mathbf{W}$:

- Note first that $\mathbf{U}[\mathbf{X}\mathbf{X}^T + \rho\mathbf{I}]\mathbf{U}^T = (1 + \rho)\mathbf{I}$, and thus this term can be inconsequentially dropped.
- Note further that $\mathbf{X}\mathbf{A}^T = \mathbf{X}\mathbf{X}^T\mathbf{W}^T = \mathbf{W}^T$ and $\mathbf{A}\mathbf{A}^T = \mathbf{W}\mathbf{X}\,\mathbf{X}^T\mathbf{W}^T = \mathbf{W}\mathbf{W}^T$.

By making these two critical observations, we can arrive at a simplified metric for channel selection:

$$\text{DI}_{di} \equiv \texttt{tr}\left(\mathbf{U}\mathbf{W}^T[\mathbf{W}\mathbf{W}^T + \rho'\mathbf{I}]^{-1}\mathbf{W}\mathbf{U}^T\right). \tag{47}$$

To highlight its data independency, a subscript *di* is purposefully inserted in the denotation: DI_{di}. This data-independent metric leads to an expedient XNAS-type structural learning strategy known as CHEX [17].

4.5 XNAS: Neural Architecture Search

Based on the popular reinforcement learning [32], many NAS designs have been proposed [18, 34, 47]. A prominent example is Google's NASNet [47], which tests and evaluates neural architectures across a search space with a prescribed search strategy and performance estimate. It adopts a policy gradient to optimally update the controller (RNN), which in turn generates the key network hyperparameters. NASNet has more comprehensive architecture space, thus the training cost is often highly prohibitive. First, the policy gradient involves a highly time-consuming search of all plausible architecture. Moreover, NASNet requires training from scratch for every architecture specification.

To tackle this concern, there are numerous computationally economical approaches being proposed. One is via a "differentiable" NAS, exemplified by DARTs [28]. It was designed to alleviate the complexity caused by the huge search space. DARTs adopts a differentiable softmax to provide a probability-like assessment the relative importance among competing substructures. This in turn facilitates an efficient gradient-descent search of suitable architectures.

Although the softmax adopted by DARTs is deemed efficient for local ranking/selection, among various substructures, it is not computationally amenable or easily expandable to facilitate highly global search/selection. An alternative approach that is highly amenable to such global search is exemplified by the so-called **Once-For-All (OFA)** net [4, 12]. By OFA, the network can be gradually trimmed pursuant to multiple hyperparameters (e.g., width, depth, kernel, size, and resolution). To assure an adequate search space, OFA starts initially with a "magnified" backbone, known as a supernet. In OFA, not only does it decouple the training and search phases but also its selection rule depends on the network weights instead of the training dataset. Such computational efficiency allows OFA to cope well with large supernets. Consequently, OFA enjoys an improvement of more than two orders of magnitude speedup over NASNet [4].

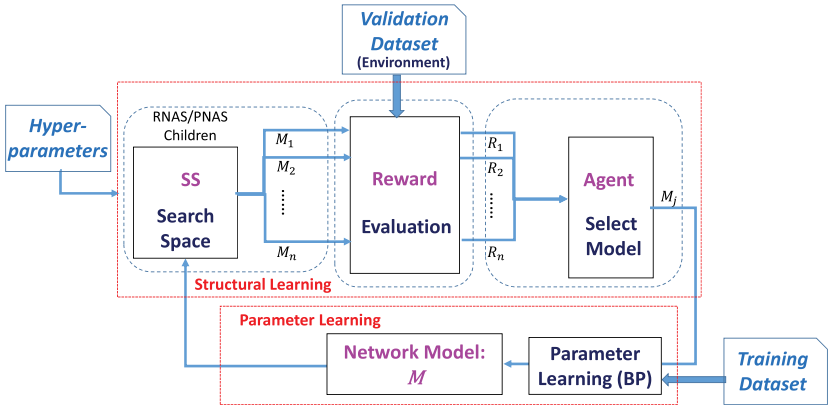


Fig. 16. A high-level XNAS flowchart.

Characterization of XNAS. To further expedite the training process, XNAS starts with a more restrictive architecture space. Indeed, XNAS is (much) faster than NASNet by a factor of two or more orders of magnitude. It is by and large competitive with OFA—each has its own merits. In an ImageNet-based NAS design example, both OFA and a data-independent XNAS (i.e., CHEX [17]) deliver the same FLOPS (=0.9G) and compatible top-1 accuracy: 76.3% vs. 76.8%. However, they differ significantly in both size and training time. For the former, OFA and XNAS require 14.5M and 7.2M parameters, respectively. For the latter, running on Nvidia V100, OFA and XNAS consume 1,200 vs. 260 GPU hours, respectively [17]. However, it is important to note that the preceding comparison bears no statistical significance, as any comparison results are highly dependent on the specific dataset, backbone, and targeted benchmarks. As a partial remedy, let us briefly highlight their respective merits in search efficiencies as follows. On one hand, OFA's search strategy is input-data independent, a vital advantage in terms of training costs. On the other hand, the subspace analysis in XNAS leads to DI-based SOM, making feasible closed-form formulation for expedient structural learning. There are several additional points on XNAS worth remarking:

- To generate the search space, an agent is adopted to select the best model of the candidates, selected according to their DI or DiLOSS scores. Moreover, our subspace analyses enable closed-form structural search strategy. Each candidate is evaluated by the validation dataset, a held-out dataset that is being used as a "fair environment" for model assessment.
- In XNAS, we shall start with a certain preferred baseline model and then gradually adjust it into a new network. For example, we would start with slimmer architectures (e.g., MobileNet for edge devices) so as to meet the stringent power/latency requirements.
- For some applications, we may resort to some kinds of hybrid of RNAS and PNAS, in an interleaved manner, so as to harness the best of the two worlds [17, 24].

As depicted in Figure 16, XNAS comprises two training subsystems that are applied in an interleaved fashion:

(1) *Parameter learning subsystem*: For parameter learning, we shall just rely on the traditional (externally supervised) BP learning [33, 37, 42].
(2) *Structural learning subsystem*: It contains three phases:
   – *Search space controller*: Unlike reinforcement learning, in XNAS, we first generate a pool of candidates (i.e., children) for RNAS/PNAS. To cope with the multiplicity of children, XNAS adopts a search criteria based on a designated SOM controlled by hyperparameters on FLOPS, storage, power, latency, and so forth.

– *Evaluation phase (reward assessment)*: Evaluation or reward assessment of the candidates is based on the generalization performances (e.g., accuracy or PSNR/SSIM) of the candidates on the validation data. From reinforced learning's perspective, the validation dataset, being disjoint with the training dataset engaged in BP learning, is poised to furnish a fair environment for evaluation.
– *Agent and action*: The major action (performed by the agent) is to select the winner(s) according to the reward assessment.

5 APPLICATIONS OF X-LEARNING

X-learning can be applied to numerous applications in both the classification and regression scenarios. X-learning starts with a baseline net and end up with a structurally reduced net, which will be termed *X-net* (with a prefix "X-") so as to differentiate it from its original baseline model.

The success of X-learning critically depends on the how do we choose the baseline models. Here, we aim at two major categories of applications: one high-performance for and the other for low-power. The former has for long time been the predominant focus of research and development. The latter has recently received a great deal of attention, and it prompts design of hardware-efficient CNNs under the power and latency constraints critical for edge/mobile devices [40, 43–45]. This is summarized by the following table:

X-Learning	Low-Power	High-Performance
Classification	XMobileNetV2 XMNasNet	XResNet50 XResNet164
Regression	XSR-ResNet	LapSRN XSR-ResNet

5.1 Experimental Results of X-Learning for Classification

For a discussion on applications of X-learning to a broad spectrum of datasets with different baselines/backbones, see the work of Kung and Hou [23] and Kung et al. [24]. Here we shall focus on reporting the experimental results on applying X-learning to the ImageNet datasets, based on two types of baseline models: ResNet and MobileNet [13, 38]. Let us now briefly highlight the experimental setup of such experiments using ResNet as the backbone:

- *Dataset characterization*:
  – *ImageNet dataset*: 1.28M images each with 224×224 image resolution
  – *Supervising teacher*: 1,000 class labels
  – *Validation dataset*: 5,000 images
  – *Preprocessing*: Normalization
  – *Data augmentation*: Translation, random resized crop, random horizontal flip
- *Backbone architecture*:
  – *Number of layers*: 50
  – *Kernel Size*: 7×7, 3×3, 1×1
- *Learning characterization*:
  – Structural fine-tuning performed once per epoch
  – Each epoch comprises 120 batches each with (batch) size 256
- *Performance metrics*: Top-1 classification accuracy
- *Design software*: PyTorch-based software system

As shown in (the last row of) Table 1, X-ResNet50 can achieve 75.60% top-1 accuracy, a 0.5% gain over the baseline, while accelerating the inference speed by 2x, which outperforms the current state-of-the-art method [31] by around 1% in accuracy with similar speeds.

Table 1. Comparison: Three X-ResNet50 Models Using Different FLOPS

Model	Top-1 Acc.	FLOPS	Param.
Baseline	75.15%	4.09B	25.60M
X-ResNet50	76.34%	2.63B	17.00M
SFP [51]	74.61%	2.42B	—
X-ResNet50	76.10%	2.31B	16.72M
Taylor [31]	74.60%	2.00B	—
X-ResNet50	75.60%	2.00B	15.60M

Table 2. Comparison with Winner of 2018 LPRIC on ImageNet

Model	Top-1 Acc.	FLOPS	Param.
MobileNetV2	71.80%	300M	3.47M
WM [38]	69.80%	210M	2.61M
X-MobileNetV2	70.80%	210M	2.33M
LPIRC 2018	65.20%	186M	-
ThiNet [50]	65.44%	170M	2.57M
DCP [46]	65.91&	170M	2.57M
X-MobileNet	68.20%	170M	1.90M

Low-power models: X-learning on MobileNet. As shown in Table 2, X-learning reduces the baseline (MobileNetV1&V2) to yield faster latency, and smaller hardware, making it amenable to edge devices. More specifically, at 30-ms/image real-time speed (required by LPIRC 2019), X-MobileNetV1 delivers 68.2% top-1 accuracy on ImageNet, a 3% improvement compared with 65.2% reported by the winner of 2018 LPIRC (MobileNet).

5.2 Regression Scenarios: X-Learning for Super-Resolution Imaging Systems

X-learning can be effectively applied to most regression-type applications, embracing, for example, restoration and enhancement problems. A typical imaging problem is as follows: given a (low-quality) input image (poor quality in resolution and/or color rendition, etc.), find a nonlinear mapping to best transform the input image into a desired output image. We shall highlight some successful applications of X-learning to enhance low-resolution images and transform them into **Super-Resolution (SR)** images. Our main focus will be placed on the application of fidelity-based SR imaging systems, which tackles the problem of recovering a high-resolution image from a single low-resolution image [1–3, 8].

Most state-of-the-art methods learn the optimal nonlinear mapping functions from multiple low- and high-resolution exemplar pairs. This approach is closely related to recent example-based methods based on deep CNNs. In fact, the sparse coding based SR methods can be viewed as a deep CNN [2, 8]. The deep learning method can be formulated for generic image SR, where the teacher will be set as the desired high-resolution images. The same techniques can be naturally carried over to other types of applications with their own training samples, such as enhancing dim-light images to a brighter image, where the teacher will be set as the desired brighter images.

X-learning approach to low-power SR imaging systems. Image SR and image enhancement have numerous practical applications [2, 8, 26]. In this problem, the original images are low-quality images, and the objective is to restore the low-quality images to a high-fidelity ones. The experimental setup for applying X-learning to the DIV2K dataset using SRGAN as the backbone is as follows:

- *Dataset characterization*:
  - *DIV2K dataset*: 800 2Kx2K images
  - *Input*: 500 × 500 low-resolution images

Table 3. Comparison with Winner of PIRM Challenge (FEQEnet)

Dataset	SRCNN	VDSR	FEQE-P	Ours
Set5	30.47/0.8610	31.53/0.8840	31.53/0.8824	31.84/0.889
Set14	27.57/0.7528	28.42/0.7830	28.21/0.7714	28.38/0.775
BSD100	26.89/0.7108	27.29/0.7262	27.32/0.7273	27.40/0.730
Urban100	24.51/0.7232	25.18/0.7534	25.32/0.7583	25.51/0.765

Table 4. Comparison with Winner of PIRM Challenge (FEQEnet)

Model	Param.	FLOPS	GPU Latency (seconds)
SRCNN	69K	128B	0.04
VDSR	668K	1231B	0.16
FEQE-P	96K	11B	0.01
Ours	92.6K	9.6B	0.004

  - *Supervising teacher (ground truth)*: 2Kx2K high-resolution images
  - *Validation dataset*: 100 2Kx2K images
- *Backbone architecture*: SRGAN-37 comprises 34 ResNet layers and three up-scaling layers
  - *Kernel size*: 7×7, 3×3, 1×1
- *Learning characterization*:
  - Structural fine-tuning performed once per epoch
  - Each epoch comprises 50 batches each with (batch) size 16
- *Performance metrics*: PSNR/SSIM
- *Design software*: PyTorch-based software system

Our baseline model is SRGAN, a predominant CNN-based model for SR imaging systems based on ResNet. Note that SRGAN's backbone model is ResNet, which is largely X-learning friendly, with the exception that SRGAN employs the the so-called subpixel layers to up-scale the LR images to the final HR output. This scheme is by itself unsuitable for X-learning. As a remedy, we replace the subpixel layers by the nearest-neighbor interpolation with learnable convolution filters for up-scaling. Thereafter, these upsampling layers become amenable to X-learning. X-SRGAN was applied to SR image enhancement with DIV2K as the training dataset. The learned SR imaging models are then applied to several test datasets, including Set5, Set14, BSD100, and Urban100.

Tables 3 and 4 summarize the performance and complexity comparison between SRCNN, VSDR, FEQE-P, and X-SRGAN. As compared with SRGAN, X-SRGAN delivers a 16× saving in model size and 14x in FLOPS while impairing a negligible loss of −0.2 dB in PSNR. As compared with SRCNN, X-SRGAN trails SRCNN in storage or number of parameters (92.6K vs. 69K). However, it outperforms SRCNN in FLOPS (9.6B vs. 128B) and GPU-latency FLOPS (0.4 vs. 40 ms). More importantly, it far outperforms SRCNN in PSNR (31.84 vs. 30.47 ms) and SSIM (0.889 vs. 0.861). Finally, X-SRGAN outperforms FEQEnet, the winner of the 2018 PIRM Challenge. More precisely, X-SRGAN delivers an advantage of shorter latency by more than two folds while holding a 0.3 dB net gain over FEQEnet in PSNR.

X-learning for high-performance SR imaging systems. We also make a comparison between the X-pruning based network compression and a network binarization method for the SR system [30], where the weights of the original CNN are converted to $[-1, +1]$ to save model size and computation complexity. For baseline CNN, we use LapSRN and SRResNet. In our experiments, we observed that X-SRGAN outperformed SR binarization by a large margin in terms of PSNR/SSIM performance metrics. Moreover, X-SRGAN achieved a higher reduction factor of model size, FLOPS, and latency. (For more details, see the work of Kung and Hou [23].)

(a) (b)

Fig. 17. The reconstructed "PPT2004" images via SRGAN (a) and JEX-SRGAN (b).

5.3 Other Application Examples: Denoising, Light Enhancing, and Image Compression

X-learning has also demonstrated great success in some other applications, including a denoising tool for noisy images) and light enhancement for night shots, among others. (For more discussion, see the work of Kung and Hou [23].) In addition, by incorporating the conventional JPEG technique into the overall CNN network, X-learning can be readily applied to further reduce the model size and yet enhance its performance. This leads to the development of a **JPEG-Embedded X-SRGAN (JEX-SRGAN)** for image compression [16]. JEX-SRGAN comprises two subnetworks: (1) a preprocessing network (JCNN) to apply JPEG to compress the input images so as to harness the forte of traditional compression tools and then (2) a reconstructing SRGAN or another SR-imaging CNN. Thereafter, X-learning is applied to prune the CNN model to attain an optimal trade-off between the compression ratio and PSNR of reconstructed images. The preliminary results of JEX-SRGAN appears to be rather impressive. For the Set14 dataset, as an example, the 16x-compressed SRGAN yields PSNR/SSIM of 28.53/0.78 while needing 1.5M network parameters. In comparison, the 20x-compressed X-JPEGAN delivers a much higher PSNR/SSIM (34.68/0.92) while only using 0.11M parameters. Figure 17 further places the reconstructed images by the SRGAN and JEX-SRGAN, side by side, with the characters "McGraw-Hill" being highlighted. It is evident that JEX-SRGAN (albeit being compressed more) reconstructs a sharper image (Figure 17) [16].

6 CONCLUSION AND FUTURE DIRECTION

This article proposes an XNAS design strategy to automatically train the network's parameters and structure. It advocates the use of subspace analysis as the theoretical footing for both PNAS and RNAS strategies. We have further extended the subspace analysis from MLPs to CNNs. Moreover, an *X-learning* paradigm is developed, which has demonstrated a broad spectrum of classification-type and regression-type applications. Moreover, it is conceivable that a proper hybrid of RNAS and PNAS may bring about further performance improvements, as exemplified by Hou et al. [17]. In addition to the reported success by applying XNAS to network compression, image enhancement, and base on the XNAS design strategy, we plan to further venture into some challenging research fronts: including 3D reconstruction, meta-learning (with few shots), bio-authentication, and many domain-driven applications. As well, the theoretical footing based on our subspace analysis distinguishes XNAS apart from most of the other NAS methods. Therefore, a proper merger of XNAS with other compatible platforms stands a good chance to jointly crack the current performance ceilings.

ACKNOWLEDGMENTS

The author wishes to thank Zejiang Hou and Yuchen Liu of Princeton University for their invaluable contribution to the research on X-learning and development of the XNAS design software system.

REFERENCES

[1] Eirikur Agustsson and Radu Timofte. 2017. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 126–135.

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. 2006. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing* 54, 11 (2006), 4311–4322.

[3] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the 2012 British Machine Vision Conference*.

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791* (2019).

[5] Han Cai, Ligeng Zhu, and Song Han. 2018. ProxylessNAS: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[6] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. 2015. PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing* 24, 12 (2015), 5017–5032.

[7] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, et al. 2019. ChamNet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2015), 295–307.

[9] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. 2018. DPP-Net: Device-aware progressive search for Pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*. 517–531.

[10] Ronald A. Fisher. 1938. The statistical utilization of multiple measurements. *Annals of Eugenics* 8 (1938), 376–386.

[11] G. H. Golub and C. F. Van Loan. 1996. *Matrix Computations* (3rd ed.). Johns Hopkins University Press, Baltimore, MD.

[12] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*. 1135–1143.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[14] Zejiang Hou and Sun-Yuan Kung. 2021. A discriminant information approach to deep neural network pruning. In *Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR'21)*. IEEE, Los Alamitos, CA, 9553–9560.

[15] Zejiang Hou and Sun-Yuan Kung. 2022. Multi-dimensional dynamic model compression for efficient image super-resolution. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

[16] Zejiang Hou and S. Y. Kung. 2019. Internal report. Princeton University (Sept. 2019).

[17] Zejiang Hou, Minghai Qin, Fei Sun, Xiaolong Ma, Kun Yuan, Yi Xu, Yen-Kuang Chen, Rong Jin, Yuan Xie, and Sun-Yuan Kung. 2022. CHEX: CHannel exploration for CNN model compression. *arXiv preprint arXiv:2203.15794* (2022).

[18] Chi-Hung Hsu, Shu-Huan Chang, Jhao-Hong Liang, Hsin-Ping Chou, Chun-Hao Liu, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. 2018. MONAS: Multi-objective neural architecture search using reinforcement learning. *arXiv preprint arXiv:1806.10332* (2018).

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[20] Sun Yuan Kung. 2014. *Kernel Methods and Machine Learning*. Cambridge University Press.

[21] Sun-Yuan Kung. 2017. Compressive privacy: From information/estimation theory to machine learning [lecture notes]. *IEEE Signal Processing Magazine* 34, 1 (2017), 94–103, 112.

[22] Sun-Yuan Kung. 2017. Discriminant component analysis for privacy protection and visualization of big data. *Multimedia Tools and Applications* 76, 3 (2017), 3999–4034.

[23] Sun-Yuan Kung and Zejiang Hou. 2020. Augment deep BP-parameter learning with local XAI-structural learning. *Communications in Information and Systems* 20, 3 (2020), 319–352.

[24] Sun-Yuan Kung, Zejiang Hou, and Yuchen Liu. 2019. Methodical design and trimming of deep learning networks: Enhancing external BP learning with internal omnipresent-supervision training paradigm. In *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'19)*. IEEE, Los Alamitos, CA.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.

[26] Juncheng Li, Faming Fang, Kangfu Mei, and Guixu Zhang. 2018. Multi-scale residual network for image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*.

[27] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV'18)*.

[28] H. Liu, K. Simonyan, and Y. Yang. 2018. DARTS: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[29] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*. 2736–2744.

[30] Yinglan Ma, Hongyu Xiong, Zhe Hu, and Lizhuang Ma. 2019. Efficient super resolution using binarized neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 694–703.

[31] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[32] P. Read Montague. 1999. Reinforcement learning: An introduction, by Sutton, R. S. and Barto, A. G. *Trends in Cognitive Sciences* 3, 9 (1999), 360.

[33] David B. Parker. 1985. *Learning Logic*. Technical Report TR-47. Center of Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA.

[34] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *Proceedings of the International Conference on Machine Learning*.

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*.

[36] Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65, 6 (1958), 386–408.

[37] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1985. *Learning Internal Representations by Error Propagation*. Technical Report. La Jolla Institute for Cognitive Science, University of California, San Diego.

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[39] Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell System Tchnical Journal* 27 (1948), 379–423, 623–656.

[40] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. 2019. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[41] Andrew R. Webb and David Lowe. 1990. The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks* 3, 4 (1990), 367–375.

[42] Paul Werbos. 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph. D. Dissertation, Harvard University.

[43] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[44] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

[45] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[46] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. 2018. Discrimination-aware channel pruning for deep neural networks. In *Proceedings of the 32nd International Conference on Advances in Neural Information Processing Systems*. 883–894.

[47] Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*.

[48] K. Pearson. 1901. On lines and planes of closest fit to systems of points is space. *Philosophical Magazine Series* 6, 2 (1901), 2:559–572.
[49] H. Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* 24 (1933), 498–520.
[50] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*. 5058–5066.
[51] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866.

Received 16 November 2021; revised 8 April 2022; accepted 24 May 2022