Neurocomputing 398 (2020) 185–196

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Sparse low rank factorization for deep neural network compression

Sridhar Swaminathan[a,*], Deepak Garg[b], Rajkumar Kannan[c], Frederic Andres[d]

[a] Bennett University, Greater Noida 201310, India
[b] Bennett University, Greater Noida 201310, India
[c] Bishop Heber College (Autonomous), Tiruchirappalli 620017, India
[d] National Institute of Informatics, Tokyo, Japan

ARTICLE INFO

Article history:
Received 30 September 2019
Revised 11 January 2020
Accepted 8 February 2020
Available online 15 February 2020

Communicated by Dr. Xin Luo

Keywords:
Low-rank approximation
Singular value decomposition
Sparse matrix
Deep neural networks
Convolutional neural networks

ABSTRACT

Storing and processing millions of parameters in deep neural networks is highly challenging during the deployment of model in real-time application on resource constrained devices. Popular low-rank approximation approach singular value decomposition (SVD) is generally applied to the weights of fully connected layers where compact storage is achieved by keeping only the most prominent components of the decomposed matrices. Years of research on pruning-based neural network model compression revealed that the relative importance or contribution of each neuron in a layer highly vary among each other. Recently, synapses pruning has also demonstrated that having sparse matrices in network architecture achieve lower space and faster computation during inference time. We extend these arguments by proposing that the low-rank decomposition of weight matrices should also consider significance of both input as well as output neurons of a layer. Combining the ideas of sparsity and existence of unequal contributions of neurons towards achieving the target, we propose sparse low rank (SLR) method which sparsifies SVD matrices to achieve better compression rate by keeping lower rank for unimportant neurons. We demonstrate the effectiveness of our method in compressing famous convolutional neural networks based image recognition frameworks which are trained on popular datasets. Experimental results show that the proposed approach SLR outperforms vanilla truncated SVD and a pruning baseline, achieving better compression rates with minimal or no loss in the accuracy. Code of the proposed approach is avaialble at https://github.com/sridarah/slr.

1. Introduction

Recently, deep neural networks achieved remarkable accuracy in many applications involving visual, audio, and textual data. In specific, convolutional neural networks and recurrent neural networks are widely applied for applications such as image recognition, object detection, style transfer, speech recognition, neural machine translation, etc. Even though the training of deep learning models is accelerated with GPU computations, the model deployment takes place on lower-end computing architectures with limited memory, limited computational power, and limited battery power [1]. Many deep learning architectures are computationally expensive and memory intensive, hindering their deployment in devices with low memory resources like mobile phones or in applications with strict latency requirements [2].

While earlier CNN models are deeper and overparameterized, many recent CNN architectures followed *compact network designing* strategies in building light-weight models. Utilization of 1 × 1 convolution, resulted in reduction of both computation as well as space complexity. Famous models such as GoogleNet [3] and ResNet [4] avoided fully connected layers rather used global average pooling for handling images of varying sizes. Google's another popular network architecture MobileNet [5] used combination of depth-wise convolutions with 1 × 1 convolution for building deeper but compact network suitable for deployment in mobile phones. Even though these design strategies help to build deeper and compact networks, still there is lot of necessity for compression of these models as they are over-parameterized most of the time.

In general, model compression and optimization approaches operate at different levels in a neural network. One straightforward way to achieve model compression is to apply *network quantization* to the layer weights which can be represented using minimum number of bits by applying methods, such as k-means scalar quantization [6] or vector quantization [7]. Approaches related to

* Corresponding author.
E-mail addresses: sridhar.swaminathan@bennett.edu.in (S. Swaminathan), deepak.garg@bennett.edu.in (D. Garg), rajkumar@bhc.edu.in (R. Kannan), andres@nii.ac.jp (F. Andres).

https://doi.org/10.1016/j.neucom.2020.02.035
0925-2312/© 2020 Elsevier B.V. All rights reserved.

bit-level representation of weights achieved highly compressed networks such as 1-bit representation in binary weight network (BWN) called XNOR-Net [8] and 2-bits representation in ternary weight networks [9] (TWN). These approaches either quantize or binarize the network weights after or even during the training where the importance or redundancy among the parameters are totally ignored. Despite achieving high compression rate, accuracy of these methods drastically drops after the compression.

Structured matrix representation and their transformation is sometimes leveraged for compressing fully-connected layer. Structured matrix representation is attained using circulant projections of weight matrices [10] for approximation using Fast Fourier Transform (FFT) or using reparameterization of fully connected layers using Adaptive Fastfood transform approach [11]. Even though structured matrices have their own advantages, layer weights representation in term of structured matrices is considered challenging and often resulted in accuracy loss. Another recent compression strategy called *teacher-student network* builds compact models by training a smaller student network mimicking the functionality of deep and larger teacher network. Famous approach Knowledge Distillation (KD) method [12] mimics deep wide networks using a shallow network. The student networks are made shallow but deeper [13], since deep networks have better representation. Knowledge Transfer transfer (KT) is sometimes used for training a compact student binary weighted network for object detection [14]. This teacher-student network strategy is seldom used in practice as it is not always guaranteed to have an accurate student network. *Neural Architecture Search* (NAS) is another recent approach for automatically searching for suitable neural network architectures based on certain constraints. Few approaches use reinforcement learning [15] and evolutionary algorithms [16] for the dynamic exploration of architectures. In practice, NAS is considered highly time-consuming due to its complex nature.

Due to the stochastic initialization of weights and dynamism of backpropagation, neural networks with random initial weights are trained in an uncontrollable way, results in trained but highly redundant neurons with varying contributions in achieving the goal [1,2]. This often happens since the deep neural network architectures are overparameterized. A successful compression strategy for this common scenario *network pruning* removes unimportant parameters of neurons to reduce the redundancy among the network weights, which eventually minimizes the complexity of the network [17]. Removal of parameters in a network happens at different levels in a network, such as synapses [18], nodes of fully-connected layers [19], convolutional layers [20], at filter-level [21,22], and rows or columns of dense layers [23]. Importance of a connection, node or filter is determined based on various factors such as weights, activations, effect in accuracy etc. Recently, successful online pruning approach used in TensorFlow model_pruning library [18] and comparative study on pruning [24] clearly indicates that synapses pruning on large sparse networks [18] have better accuracy than node pruning, which results smaller dense networks. These studies have also mentioned the advantages of having sparse weight matrices, which take fewer computations than dense network weights, as the multiplications of sparse entries are omitted in new-age sparse supportive libraries and hardware architectures. Sparse matrices also consume lesser memory space when stored in compressed sparse row (CSR) or compressed sparse column (CSC) formats.

Low rank factorization (LRF) applied to convolution filters [25] helps to speed up the time-taking convolution operation as well as to compress the parameters of convolutional kernels. Low rank decomposition method SVD is stable and highly effective in compressing weights of fully connected layer [26] by estimating the informativeness of the parameters. The factor matrices can be stored well efficiently in resource constrained devices. Unlike pruning, the model weights compressed using low rank decomposition do not require retraining. Low-rank factorization is applied for wide variety of applications starting from reducing dynamic parameters [27], compression of final layer weights in an acoustic DNN [28], speeding-up object detection in Fast-RCNN [29], iterative compression of CNN [30], to compression of fully connected layer in transfer learning scenario [31].

Inspired from the theories of pruning and sparsity, we argue that significance or contribution of neurons in layer should be taken into account in low-rank approximation where structured sparsity in decomposed matrices is a highly desired property for achieving both compact storage and faster computations. To achieve this objective, we propose a novel *sparse low rank* (SLR) method that improves compression of SVD by sparsifying the decomposed matrix, giving minimal rank for unimportant neurons while retaining the rank of important ones.

Contributions of this work are as follows.

1. We propose a novel *Sparse Low Rank* approach that improves the truncated SVD based on theories related to pruning and sparsity. In contrast to the traditional truncated SVD, which treats all the neurons in layer equally while performing the low-rank approximation, the proposed approach relatively represents the decomposition matrix entries based on widely used pruning strategies for neuron ranking. Structured sparsity introduced to the truncated SVD by our approach can achieve higher speed-up using Basic Linear Algebra Subprograms (BLAS) libraries in addition to higher compression. To the best of our knowledge, no other work has been proposed to incorporate pruning strategies to compress SVD in the context of deep neural network compression.
2. Since the proposed approach builds on popular strategies used in years of research on pruning and sparsity, it can be easily generalized and further extended to any low rank factorization-based compression method. The proposed approach allows easy way to balance the trade-off between sparsity, compression rate and accuracy using flexible hyperparameters. This work presents detailed experimental results for verities of deep learning models evaluated on popular datasets. The experimental results on influence of different parameters suggest interesting insights on how the hyper-parameters of the proposed approach can be tuned for better results.

2. Related work

Many approaches have been proposed in the recent years for deep learning model compression where comprehensive survey on this topic can be found in [1,2,32]. Deep learning model optimization methods broadly fall into any of the aforementioned classes of approaches, such as compact network designing, network quantization, structured matrix representation, teacher-student network, Neural Architecture Search, network pruning, and low-rank factorization. Specifically, influential works on pruning, sparsity and low-rank factorization closely related to ours are discussed in this section. For more broader review on other classes compression approaches, readers are recommended to refer recent surveys.

2.1. Pruning

Pruning nowadays is an extensively applied neural network optimization strategy even though the ideology was popular known even before deep learning arose to mainstream research. While earlier approaches removed the connections [18] or nodes [19] permanently in a single shot, it was later found that iterative removal and retraining results in high compression rate and lesser loss in accuracy [24]. Pruning on convolution filters has revealed many interesting factors about desired level of pruning in a network. Weights of kernels can be pruned at random locations (i.e. fine-grained pruning) or at the level of vectors (i.e. vector-level pruning) or slice in kernels (i.e. kernel-level pruning) or even entire kernel itself (i.e. filter-level pruning) [32]. The work in [33] experimented with different granularities of pruning and found that vector-level pruning is better than the fine-grained pruning for optimal storage where kernel-level and filter-level pruning are hardware friendly for easier computation.

2.2. Structured sparsity

Lebedev and Lempitsky [34] discovered that similar sparsity pattern across multiple filter weights can utilize the parallel processing libraries with BLAS for speed-up. This concludes that *structured sparsity* is considered a desirable property for weight matrices. Similar to [34], our proposed work also endorses structured sparsity in SVD matrices. Filter-level pruning applied in ThiNet [35] removes convolution filters in a greedy manner which are guided based on the feature maps of next layer. Absolute removal of filters attains better storage and speed but mostly produces a narrow inaccurate model. Sparsity in weight matrices is endorsed during the training using approaches, such as L0 regularization [35], L1 norm [36,37] and dynamic pruning of weights using binary masking [24].

2.3. Hybrid approaches

Han et al. [38] applied combination of multiple compression methods where a model is first pruned and further retrained with sparse connections. Fine-tuned sparse weights are further quantized using a weight sharing approach where Huffman coding is finally applied to the quantized weights for even further compression. In a similar fashion, our proposed approach can also be applied to any dense weight matrix before or even after pruning. One major challenge in network pruning is, the network cannot recover from the incorrectly removed connections, nodes or filters. To handle this, dynamic network surgery framework in [39] recovers mistakenly pruned synapses or connection using an operation called splicing. Famous work in [18] and comparative study on sparse pruning in [24] discovered that gradual fine-grained pruning at training results in unstructured sparsity in weight matrix in wider sparse models which are better than narrow dense models resulted from node pruning. In a similar spirit, our proposed approach also aims to compress the layer weights using SVD without applying node pruning to the layer.

2.4. Matrix factorization

Compression approaches based on low-rank approximation focused reducing computations of convolutions [25,30,40,41,42,43], parameters of dense layers [26,28–31,40] or both [30,40]. One of the earlier approaches proposed by Denton et al. [25] achieved $2\times$ speedup with 1% accuracy drop when low-rank approximation is applied to convolution kernels. A 4D convolution tensor can be factorized into decomposed matrices of two-components [41], three-components [42], or four components [43]. Jaderberg et al. [41] decomposed convolution weights into two matrices resulted decent speedup in text character recognition model with small drop in accuracy. Wang et al. [42] exploited Tucker Decomposition (TD) for decomposing VGG16 model's filters into three factors. This three-component decomposition achieved high speed up for object detection and image retrieval tasks. Xu et al. [14] used canonical polyadic decomposition (CPD) for 4-way decomposition where higher speed-up ratio was achieved for Alexnet model. It should be noted that the speedup is proportional to number of components in the decomposition with the negative effect in the accuracy as a counterpart. It should be noted that factorization of fully connected layer weights is always two components decomposition.

Other than SVD, TD, and CPD methods, another class of matrix decomposition approach called non-negative matrix factorization (NMF) is also considered useful strategy for latent factor analysis (LFA). The NMF guarantees to have decomposed matrices with positive entries in latent factors that are easily interpretable. It has many applications in speech enhancement [44], image clustering [45], deep learning model compression [46], recommender systems [47], large-scale networking [48] and QoS prediction [49–51]. The authors in [46] use semi non-negative matrix factorization where one of the decomposed matrices is kept non-negative. While NMF is better and easier for interpretation of latent factors, SVD is a commonly used approach for deep learning model compression. This is because SVD always results in decomposed matrices with orthonormal basis vectors (in U and V matrices) with their relative importance (Sigma matrix S). Orthonormal latent factors and their importance are useful properties in determining importance of entries for removing redundancy in model compression. Hence, the property of non-negativity seldom considered in the context of deep learning model compression. In our proposed approach, we use naïve two-way decomposition method SVD for the matrix factorization task.

2.5. Singular value decomposition

Singular value decomposition (SVD) is the most commonly used approach for low-rank approximation of 2D matrix weights in stable way. Other than image classification applications, SVD is applied for deep neural networks based acoustic modeling [26,28] as well as object detection [29]. Xue et al. [26] proposed used SVD for restructuring the deep neural networks of acoustic models with minimal loss in accuracy. The authors exploited the sparseness in the original weight matrix for restructuring the DNN using matrix decomposition. As acoustic and language models have large number of target outputs, authors in [28] applied low-rank matrix factorization to the weights of final layer to reduce 30–50% of layer parameters. Girshick [29] applied SVD as a post-processing method for reducing complexity of fully-connected layer in the famous object detection model Fast R-CNN. Since recent object detection approaches are end-to-end fully convolutional networks, the application of low-rank decomposition to object detection is now focused more on convolution kernels. While SVD is a widely used approach for 2D matrix decomposition, Canonical Polyadic Decomposition (CPD), Tucker Decomposition, and other tensor decomposition approaches [52] are commonly used for generalized low-rank decomposition for multidimensional matrices such as 3D or 4D tensors of convolutional kernel weights.

Few approaches [30,40] applied low-rank decomposition for both convolution layers as well as fully connected layers. Kholiavchenko et al. [30] proposed an iterative approach to low-rank decomposition using dynamic rank selection scheme applied to image classification and object detection models. However, iteratively applying low-rank decomposition consumes longer time and huge computations for rank selection in larger models. Approach proposed in [40] takes care the properties of both low-rank and sparseness of weight matrices while aiming to reconstruct the original matrix. Recent approach DALR [31] has discovered that considering activation statistics of layer improves SVD compression based on the intuition that only few of the neurons in base CNN network will be activated in a transfer learning scenario. Their decomposition loses the original characteristics of the weight matrix when adapting to the new domain. Keeping the properties of original weights is a major concern in models, which are either trained from scratch or to be used in online learning and incre-

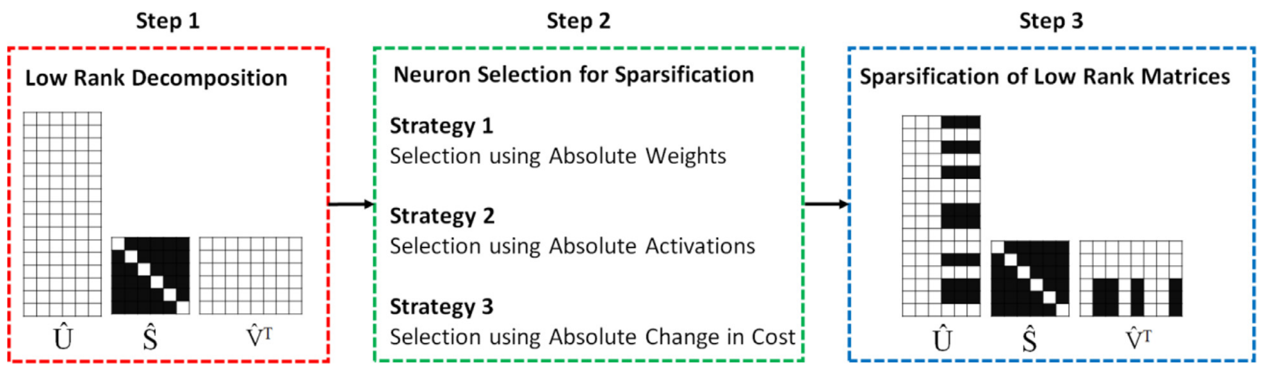
Step 1
Low Rank Decomposition
Û
Ŝ
V̂T
Step 2
Neuron Selection for Sparsification
Strategy 1
Selection using Absolute Weights
Strategy 2
Selection using Absolute Activations
Strategy 3
Selection using Absolute Change in Cost
Step 3
Sparsification of Low Rank Matrices
Û
Ŝ
V̂T

Fig. 1. Architecture of the proposed approach Sparse Low Rank Decomposition.

mental learning where the retraining or finetuning of the model is required. Objective of most of the existing low-rank approaches for compressing of fully connected layers to minimize the reconstruction error. In contrast, we propose to incorporate the theories from pruning to low-rank approximation to consider the significance of nodes in a layer.

3. Proposed approach

In this section, we present our proposed approach in three major steps as shown in the Fig. 1. Initially, traditional low-rank decomposition SVD is applied to the weight matrix of a fully connected layer. Then, input and output neurons in the layer are selected for sparsification using any one of the three neuron selection strategies. Finally, sparsification is applied to the selected input and output neuron components in the decomposed matrices. In general, low-rank decomposition is considered as a post-processing step aimed only at optimal storage of trained layer weights. In the same spirit, the proposed approach is also an improvised extension of the traditional low-rank factorization, which has no influence in the model training, hyperparameter optimization, etc.

3.1. Low rank decomposition

Low-rank factorization can be exploited for decomposition of neural network weights of any type. A convolutional neural network usually comprises lot of convolutional layers and fully connected layers. As discussed earlier, the former takes huge computation time and the later uses lot of storage space during inference. Low-rank decomposition applied to convolutional kernels weights is mainly aimed at optimizing the inference speed of the convolution operation since the time taken for multiplication with factorized matrices is lesser than that of multiplication with 3D weights of kernels. In contrast, our proposed approach works mainly on decomposition of fully connected layers aiming mainly for optimal storage where computation is also boosted as a result of factorization.

In a fully connected (FC) layer having m input and n output neurons, activation $a \in \mathbb{R}^n$ of the layer with n nodes is represented as

$$a = g\left(W^T X + b\right) \tag{1}$$

where X is the input to the layer and g() is any activation function. FC layer forms a weight matrix $W \in \mathbb{R}^{m \times n}$ and a bias vector $b \in \mathbb{R}^n$ where each parameter in the weight matrix W is $w_{ij} \in \mathbb{R}$ $(1 \le i \le m,\ 1 \le j \le n)$ and bias matrix b is $b_j \in \mathbb{R}$ $(1 \le j \le n)$. The proposed approach is applied to weight matrix W after the neural network is trained fully. SVD approach decomposes the weight matrix W as

$$W = U S V^T \tag{2}$$

where $U \in \mathbb{R}^{m \times m}$, $V^T \in \mathbb{R}^{n \times n}$ are orthogonal matrices and $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix.

3.2. Sparse low rank decomposition

The matrix sigma S is a diagonal matrix comprising n non-negative singular values in a decreasing order. Truncated SVD keeps the most significant k singular values where the decomposed matrices U, S, and V^T become $\hat{U}, \hat{S}, \hat{V} \in \mathbb{R}^{m \times k}, \mathbb{R}^{k \times k}, \mathbb{R}^{k \times n}$. This approximates the original weights W into reconstructed approximated weight $\hat{W}$ as

$$\hat{W} = \hat{U} \hat{S} \hat{V}^T. \tag{3}$$

SVD always results diagonal matrix sigma S with the most significant singular values from the upper left to lower right in a decreasing order. The truncation process always keeps the first k rows of U and columns of V^T. Hence the truncation cannot happen at random indices or from the final rows or columns of U and V^T. Many previous works [25–30,40] intended to find the low-rank subspace by focusing only on minimizing the weight matrix reconstruction error $||W - \hat{W}||_F$. The number of non-zero parameters (shortly referred as params) in the truncated SVD is $k(m + n + 1)$. Compression Ratio cr is the ratio between parameters after and before the decomposition which is calculated as $k(m + n + 1)/(m \times n)$. In addition, the weight matrix W does not need to be reconstructed as $\hat{W}$ during inference where the input X can be directly multiped with $\hat{U}$, $\hat{S}$ and $\hat{V}^T$ matrices, which takes lesser computation than direct multiplication of $\hat{W}$ and X.

The diagonal values of the decomposed matrix $\hat{S}$ i.e. non-negative singular values signify k latent concepts related to the parameters in W. Magnitude of a singular values in $\hat{S}$ signifies the importance of the kth latent factor of W. Each row of $\hat{U}$ signifies the correlation between each input neuron and the k hidden concepts where each column of $\hat{V}^T$ denotes the correlation between each output neuron and the same k latent concepts. Truncated SVD is variation of SVD which removes insignificant entries of $\hat{U}$ and $\hat{V}^T$ by truncating their columns and rows correspondingly.

Choosing rank k: In general, value of k is chosen using a measure called *explained variance ratio*. Each kth component of the SVD results is capable of explaining or reconstructing some percentage of variance in W. For instance, if any target application requires only 95% of the variance of the original W, then we choose a maximum value of k whose sum of explained variance rate is greater than or equal to 95%. However, in the case of truncated SVD for model compression, it may also be possible that a k with explained variance even lesser than 95% can still achieve decent accuracy. Here k is chosen as the smallest possible value which achieves

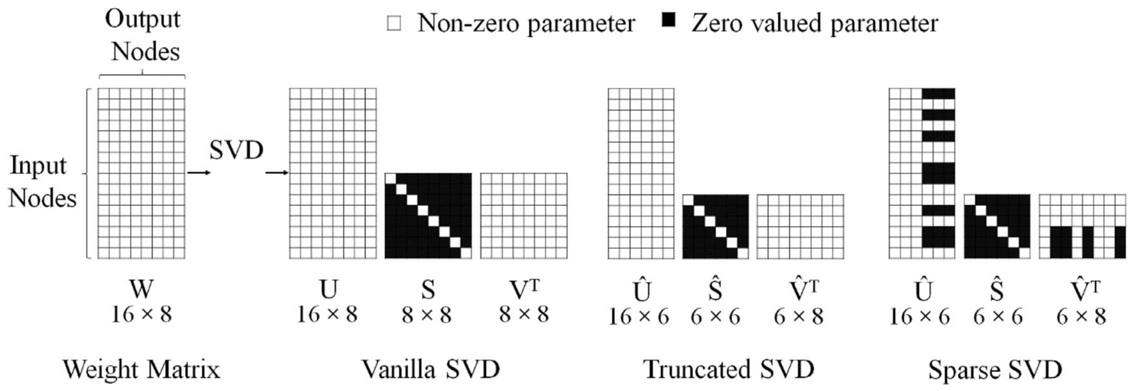

Fig. 2. Singular value decomposition (SVD), truncated SVD, and our proposed approach sparse SVD called as SLR.

lower compression ratio *cr*, minimal reconstruction error $||W-\hat{W}||_F$ and minimal loss in the testing set accuracy after the compression. Rank of the decomposition *k* can be chosen by compressing the matrix with different *k* values then analyzing the drop in testing set accuracy for each *k*. Hence the rank *k* here depends upon percentage of drop in testing accuracy rather than the explained variance ratio.

We propose that truncated matrices $\hat{U}$, $\hat{V}^T$ can be further compressed based on the importance of the *m* input and *n* output neurons. This can be achieved by representing few rows of $\hat{U}$ and columns of $\hat{V}^T$ with a rank lesser than *k*, called as *reduced rank rk* i.e. insignificant or lesser important input and output neurons can be represented with smaller rank $rk < k$. Entries of those rows and columns represented using reduced rank will have values as zero where this reduction operation is called as *sparsification*. From the perspective of pruning based optimization research, this sparsification is a form of *pruning* individual neurons' reconstruction components associated with corresponding latent concepts as described earlier.

Fig. 2 depicts the comparison between traditional truncated SVD and our proposed *sparse low rank* approach. As in the figure, the truncated SVD represents the decomposition using rank $k = 6$, where the proposed sparse SVD represents half of the rows and columns of the $\hat{U}$ and $\hat{V}^T$ using reduced rank $rk = 3$ (i.e. half of original rank $k = 6$) and rest of the rows and columns of $\hat{U}$ and $\hat{V}^T$ with full rank $k = 6$. It can be seen that the number of zero valued parameters of sparse SVD are much more than the truncated SVD.

It should be noted that the sparsification of entries in $\hat{U}$ and $\hat{V}^T$ should be done only by keeping individual neurons' ranks lesser than *k*. Sparsity should not be achieved by directly setting lower magnitude values of $\hat{U}$ and $\hat{V}^T$ as zeros using L1 norm. The importance of latent concepts contributing to reconstruction of *W* starts from left to right columns of $\hat{U}$ and top to bottom rows of $\hat{V}^T$. Since the lower magnitude values in $\hat{U}$ and $\hat{V}^T$ might be there randomly everywhere, sparsifying (i.e. unstructured sparsity) an entry related to an important latent concept in the matrix $\hat{U}$'s left most columns or $V^{\hat{}T}$'s top most rows (as in Fig. 2) would lead to huge impact in the matrix reconstruction, thus leading to drop in model's accuracy. So, only the vector level sparsification is desired rather than random unstructured pruning or sparsification of the decomposed matrices. Having same reduced rank for few rows and columns is also storage efficient where sparse matrix compression approaches require only fewer indices to indicate the sparse entries. This structured sparsity enables our proposed approach to utilize BLAS libraries for higher speed-up.

The reduced rank *rk* of the unimportant rows and columns is decide by a parameter *reduction rate* $rr \in \mathbb{R}$, $0 \leq rr \leq 1$ where *rk* is computed as $\lfloor k \times rr \rceil$. The number of sparse rows (rm) and sparse columns (rn) to be represented using reduced rank *rk* are determined by the *sparsity rate* $sr \in \mathbb{R}$, $0 \leq sr \leq 1$ where rm and *rn* are computed as $\lfloor m \times sr \rceil$ and $\lfloor n \times sr \rceil$. Our proposed approach SLR assumes that the percentage of unimportant neurons of both input and output are same. Hence same reduction rate is applied to both rows and columns of $\hat{U}$ and $\hat{V}^T$. When the matrices $\hat{U}$, $\hat{S}$, $\hat{V}^T$ are sparsified with *sr* and *rr*, the total number of non-zero parameters of the $\hat{U}$, $\hat{S}$, $\hat{V}^T$ become $k(m\text{-}rm\text{+}n\text{-}rn\text{+}1) + rk(rm\text{+}rn)$, which is lesser than the number of non-zero parameters of truncated SVD $k(m + n + 1)$. Parameters, such as reduction rate *rr* and sparsity rate *sr* can be considered as hyperparameters that should be tuned for better compression where their influence is investigated in the experiments section.

Example: If a fully connected layer with 40 input nodes and 20 output nodes ($m = 20$, $n = 10$) forms a weight matrix *W* with $40 \times 20=$**800** params is decomposed using SVD and truncated with rank $k = 10$, then the total number of non-zero parameters in truncated SVD is 10(40+20+1)=**610** params (i.e. $k(m + n + 1)$). If we further apply the proposed approach sparse SVD with reduction rate *rr*=0.5 and sparsity rate *sr*=0.6, then rm=24 input nodes ($\lfloor m \times sr \rceil$ i.e. $\lfloor 40\times0.6 \rceil$) and *rn*=12 output nodes ($\lfloor n \times sr \rceil$ i.e. $\lfloor 20\times0.6 \rceil$) will be represented with the reduced rank $rk = 5$ ($\lfloor k \times rr \rceil$ i.e. $\lfloor 10\times0.5 \rceil$) where rest of the 16 input nodes and eight output nodes will be represented using full rank $k = 10$. Now the total number of non-zero parameters in sparse SVD is 10(16+8 + 1)+5(24+12)= **430** params (i.e. $k(m\text{-}rm\text{+}n\text{-}rn\text{+}1)\text{+}rk(rm\text{+}rn)$). Hence the compression ratio *cr* of truncated SVD is 610/800=**0.76** and of sparse SVD is 430/800=**0.53**. In this parameter setup, reduction rate *rr* and sparsity rate *sr* should be estimated based on empirical study, aimed to give minimal compression ratio without much loss in the accuracy. In this example, the parameter setup conveys that there are 60% unimportant input and output neurons in the layer, which can be represented using just half of the original rank. However, the number of significant and lesser significant neurons depend upon multiple influential factors, such as size of the network and layer, dynamics of the training etc.

Theoretically, it is evident that the expected accuracy of sparse SVD for a specific rank *k* will be equal to or lower than the accuracy of regular truncated SVD with same rank *k*. This is since the process of sparsification removes the lesser important features from the decomposed matrices without compensating the improvement in feature representation. However, we practically prove through experiments that sparse SVD achieves better compression than truncated SVD with no or marginal loss in accuracy.

3.3. Neurons selection for sparsification

Selection of the rows and columns for sparsification is determined based on significance or importance of input and output neurons. The importance or significance of a specific ith input neuron and jth output neuron $is_i \in \mathbb{R}^m$, $os_j \in \mathbb{R}^n$ $(1 \le i \le m, 1 \le j \le n)$ can be computed using one of the three ways 1. absolute weight, 2. absolute activation, and 3. absolute change in cost. These criteria can be compared to the strategies used in the traditional neural network node pruning approaches.

3.4. Selection based on absolute weights

Since neural network are trained from small random weights, larger magnitude of weights between neurons emphasize importance of connections. One way to estimate the significance of input and output neurons to analyze the absolute weights of connections associated with them. Input and output neurons significance *is* and os is calculated as the sum of absolute weights of connections that are coming from the input neurons and are going to the output neurons correspondingly. They are calculated as absolute sum based on rows or columns of W as

$$is_i = \sum_{j=1}^{n} \left|W_{ij}\right| \tag{4}$$

$$os_j = \sum_{i=1}^{m} \left|W_{ij}\right|. \tag{5}$$

Here, it should be noted that only absolute values of the weights are considered so that the magnitudes of the weights will be influential not the direction.

3.5. Selection based on absolute activations

Activation of a neuron in a network signifies whether a specific feature is detected from an input or not. In other words, it is the degree of similarity between input and the neuron weights. Also, a neuron is considered important if it is activated often during training phase. Let $A \in \mathbb{R}^{m \times n \times p}$ be 3D matrix comprising the activations of m input neurons and n output neurons for p samples in the training set. Here, this activation matrix A is constructed for fully trained model. Significances *is* and os can be estimated as the sum of absolute activations for the samples in the training set as

$$is_i = \sum_{l=1}^{p} \sum_{i=1}^{m} |A_{il}| \tag{6}$$

$$os_j = \sum_{l=1}^{p} \sum_{j=1}^{n} \left|A_{jl}\right|. \tag{7}$$

The choice of activation for a network among the functions such as sigmoid, tanh, relu, leaky relu, linear, etc., is determined totally based on training and testing accuracies where each of these activation functions work specifically well on certain domain of applications. As few activation functions (tanh, linear. leaky relu) are capable of producing both positive and negative values, only the absolute values of activations are considered here to analyze how often they are activated in a given training set. Irrespective of the activation function, the importance of a neuron is solely based on its rate of activation that is computed here using sum of absolute activations. It should be noted that our proposed approach is independent to any activation function with any range where the choice of activation is left to the model developer.

3.6. Selection based on absolute change in cost

A neuron is considered less significant if there is no change in the network performance after its removal. Let c is the default cost of the neural network with original trained weight W estimated for the p training samples, computed using any loss function. Let $\hat{c}$ is the cost value of the network with reconstructed weights $\hat{W}$ using sparse SVD where a single ith row of $\hat{U}$ or jth column of $\hat{V}^T$ is represented with reduced with rank rk. Now, is and os can be estimated as the absolute change in the cost value when the specific row of $\hat{U}$ or column of $\hat{V}^T$ is truncated with reduced rank rr. Cost based is and os are calculated as

$$is_i = \left|c - \hat{c}_i\right| \tag{8}$$

$$os_j = \left|c - \hat{c}_j\right|. \tag{9}$$

Only the absolute change in the cost value is considered since the sparsification process aims not to reduce the overall network cost or improve accuracy but to ensure the functionality of the network does not change after compression. Estimation of absolute change in cost value is similar to the strategy used in [53] for network pruning. In practice, cost based significance estimation is computationally very expensive for training set with large number of samples. However, this can be sometimes reduced by measuring the cost with subset of training samples which is not much recommended.

3.7. Sparsification

After the input, output significances are computed using any one of the three aforementioned strategies, the matrices $\hat{U}$ and $\hat{V}^T$ are sparsified. Selection of rm rows and rn columns of $\hat{U}$ and $\hat{V}^T$ for sparsification is formulated as a minimization problem where set of rows or columns are chosen as

$$\min \sum_{i=1}^{m} is_i \; x_i \, subject\, to \sum_{i=1}^{m} x_i = rm, \; x_i \in \{0, 1\} \tag{10}$$

$$\min \sum_{j=1}^{n} os_j \; y_j \, subject\, to \sum_{j=1}^{n} y_j = rn, \; y_j \in \{0, 1\} \tag{11}$$

where x_i and y_j are binary values, i and j are indices of rows and columns of $\hat{U}$ and $\hat{V}^T$ to be represented with rank rk. After the sparsification process, matrices $\hat{U}$ and $\hat{V}^T$ can be compressed and stored in a resource constrained device where the inference input X can be directly multiplied with decomposed sparse matrices $\hat{U}$ and $\hat{V}^T$ in a faster manner.

4. Experimental results

In this section, we present the datasets and models used for experimentation, parameter setup and evaluation measure and the comparative results.

4.1. Dataset and models

Following many previous work, architectures and datasets were chosen comprising good mix of layer depths: shallow vs. deep, number of classes: small vs. large, and training modes: fresh training vs transfer learning. Our proposed approach is highly suitable for compressing dense layers of any network architecture. However, experiments were conducted in specific on famous image recognition CNN models, such as VGG16 [54], VGG19 [54], Lenet5 [55] where VGG architecture is much known for its memory intensive fully connected layers. It should be noted that VGG is the commonly used architecture with fully connected layers where other popular image recognition models, such as ResNet, Inception, MobileNet, ResNet, DenseNet, and object detection models, do not have fully connected layers except the final softmax layer.

Table 1 depicts the datasets used for experiments CIFAR-10 [56], CIFAR-100 [56], MNIST [57], Dogs vs. Cats [58], Oxford 102

Table 1
Datasets used for experiments.

Dataset	Image type	Max/min image dim.	No. of classes	Train images	Test/Val-images
CIFAR-10	Color	32 × 32	10	50,000	10,000
CIFAR-100	Color	32 × 32	100	50,000	10,000
MNIST	Gray	28 × 28	10	60,000	10,000
Dogs vs. Cats	Color	500 × 500	2	23,000	2000
Oxford 102	Color	500 × 500	101	6149	1020

Table 2
VGG16 and VGG19 models trained for 224 × 224 and 32 × 32 images.

Layer name	Layer type		Feature map	Output size for 224 × 224	Output size for 32 × 32	Kernel size	Stride	Activation
	VGG16	VGG19						
Input	Image	Image	1	224 × 224 × 3	32 × 32 × 3	–	–	–
Conv1	2 × Conv	2 × Conv	64	224 × 224 × 64	32 × 32 × 64	3 × 3	1	relu
Pool1	Max pool	Max pool	64	112 × 112 × 64	16 × 16 × 64	3 × 3	2	relu
Conv2	2 × Conv	2 × Conv	128	112 × 112 × 128	16 × 16 × 128	3 × 3	1	relu
Pool2	Max pool	Max pool	128	56 × 56 × 128	8 × 8 × 128	3 × 3	2	relu
Conv3	2 × Conv	3 × Conv	256	56 × 56 × 256	8 × 8 × 256	3 × 3	1	relu
Pool3	Max pool	Max pool	256	28 × 28 × 256	4 × 4 × 256	3 × 3	2	relu
Conv4	2 × Conv	3 × Conv	512	28 × 28 × 512	4 × 4 × 512	3 × 3	1	relu
Pool4	Max pool	Max pool	512	14 × 14 × 512	2 × 2 × 512	3 × 3	2	relu
Conv5	2 × Conv	3 × Conv	512	14 × 14 × 512	2 × 2 × 512	3 × 3	1	relu
Pool5	Max pool	Max pool	512	7 × 7 × 512	1 × 1 × 512	3 × 3	2	relu
Flatten	Flatten	Flatten	–	25,088	512	–	–	relu
FC6	Dense	Dense	–	4096	4096	–	–	relu
FC7	Dense	Dense	–	4096	4096	–	–	relu
FC8	Dense	Dense	–	# classes	# classes	–	–	softmax

Table 3
Lenet5 model trained for 32 × 32 images.

Layer name	Layer type	Feature map	Output size	Kernel size	Stride	Activation
Input	Image	1	32 × 32 × 1	–	–	–
Conv1	1 × Conv	6	28 × 28 × 6	5 × 5	1	tanh
Pool1	Avg pool	6	14 × 14 × 6	2 × 2	2	tanh
Conv2	1 × Conv	16	10 × 10 × 16	5 × 5	1	tanh
Pool2	Avg pool	16	5 × 5 × 16	2 × 2	2	tanh
Flatten	Flatten	–	400	–	–	tanh
FC3	Dense	–	120	–	–	relu
FC4	Dense	–	84	–	–	relu
FC5	Dense	–	# classes	–	–	softmax

Table 4
Models trained for compression.

Model	Dataset	Model input dim.	Transfer learning	Batch size/epochs	Learning rate	Optimizer	Testing acc
VGG16	CIFAR-10	32 × 32	×	1 28/500	0.1	Momentum	92.63
VGG16	CIFAR-100	32 × 32	×	1 28/500	0.1	Momentum	67.59
Lenet5	MNIST	32 × 32	×	1 28/50	0.001	Adam	98.49
VGG16	Dogs vs. Cats	224 × 224	√	10/50	0.01	Adam	96.54
VGG19	Oxford 102	224 × 224	√	128/50	0.001	Adam	83.57

[59]. Dataset used for experiments comprises good mix of different image types, sizes, number of classes. CIFAR-10 and CIFAR-100 has general purpose image classes where MNIST dataset contains handwritten digit images. Dogs vs. Cats dataset has images of only two classes and Oxford 102 dataset contains 102 different classes of flower images.

Tables 2 and 3 show the architecture of VGG16, VGG19 and Lenet5 models with layer names, output shapes, kernel, and stride sizes, etc. It can be seen that the VGG19 model has three extra convolutional layers comparing to VGG16. The VGG models trained on CIFAR-10, CIFAR-100 have fully-connected layers FC6 with 512 × 4096 params, where for other datasets FC6 has 25,088 × 4096 params. All the trained VGG models have FC7 with 4096 × 4096 params. Lenet5 model has FC3 with 400 × 120 params. These aforementioned fully-connected dense layers are compressed using the proposed approach in the further experiments.

Table 4 shows the experimental setups and results of corresponding CNN models trained for compression. Accuracy of VGG16 model in CIFAR-10 is better than in CIFAR-200, because the former dataset has lesser number of target classes than the later. Performance of Lenet5 in MNIST is comparatively better than others due to model's easier visual discrimination of classes in the gray scale images. The VGG16 and VGG19 models used in transfer learning were publicly available pretrained models from Keras [60], which were originally trained on ImageNet dataset. In the transfer learning scenario for Dogs vs. Cats and Oxford102 datasets, the base network's convolution and fully connected layer weights are fine-tuned with data augmentation such as rotation, shift, shear, zoom, and horizontal flip. VGG16 model with Dogs vs. Cats dataset reaches decent accuracy by discriminating the "dogs" and "cats" classes. Since the 102 flower classes in Oxford 102 dataset has more between-class similarity VGG19 performs averagely in the dataset despite the deeper nature of VGG19 model.

4.2. Evaluation criteria and comparative methods

Performance of compression is accessed using metrics: accuracy (acc) computed on the testing set of the corresponding datasets, number of non-zero parameters (params) in the decomposed matrices and compression ratio *cr*, which is a ratio between number of non-zero parameters after and before the matrix decomposition. Compression ratio directly indicates the scale of non-zero parameters in decomposed matrices comparing to the original weight matrix. In order to assess the efficiency of the proposed approach, *sparsified compression ratio C* is computed as ratio between compression ratios of truncated SVD and proposed SLR. Model training, compression and testing experiments were conducted on NVIDIA DGX-V100 server comprising 8 Tesla V100 GPUs with total of 128 GB of GPU memory.

Proposed approach SLR is compared against truncated SVD and a pruning strategy PRU [23]. In PRU, rows or columns of weight matrix are removed based on the importance of neurons. Accuracy reported for PRU is the best result among either row wise or column wise pruning. Number of rows or columns to be removed in PRU is determined based on compression ratio *cr* of SVD. Comparison with pruning approach PRU is intended to exhibit performance of straightforward removal or sparsification of weight matrix. Due to proposed and baseline approaches' *unsupervised, lossy,* and *post-processing* compressive nature, comparison against other classes of supervised, lossless, and online pruning and low-rank approaches are not considered here. Hence the comparison is performed only against truncated SVD and direct weight matrix sparsification using pruning base line PRU for fair comparative evaluation.

Three variants of the proposed approach SLR-w, SLR-a, and SLR-c are experimented based on neuron significance computed using absolute weights, activations and change in cost correspondingly where same criterion and notation are followed for variants of pruning method PRU. Cost-based models SLR-c and PRU-c are experimented only for small datasets with 32 $\times$ 32 images to avoid exponential processing time in larger datasets. The proposed approach and comparative methods were implemented using deep learning library Keras [60].

4.3. Comparative evaluation

Tables 5–9 depict the comparison of our proposed approach SLR with SVD and PRU in compressing FC layers of different models. Table 5 compares all the variants of comparative methods where other tables display only the best among each category. For simplicity, sparsity and reduction rates of the proposed approach are kept constant as *sr*=0.5, *rr*=0.5. This parameter setup generally assumes that half of the input and output neurons in a layer are insignificant, which can be represented with half of the original rank *k*. However, these hyperparameters have to be tuned for each dataset separately where best results of our approach with different *sr* and *rr* are presented later.

Overall, the accuracies of SVD and SLR drastically dropped for very smaller *k* values due to loss of important features in the decomposed matrices. Since FC7 has larger weight matrix decomposition, it requires larger rank *k* for maintaining good accuracy. The proposed approach with reduced rank representation consistently performs well in many cases even sometimes under lower ranks. While aiming to achieve SVD's *cr*, pruning approach PRU performs poorly in the most scenarios. This happens because the pruning is applied directly to the original weight matrix through removal of either input or output neuron weights. It can be seen that the variants of the proposed approach SLR with default *sr*=0.5, *rr*=0.5 achieve smaller compression ratio and sometimes better accuracy.

Cost-based criterion of SLR-c works well for neuron selection for CIFAR and MNIST datasets. But it is considered as computationally infeasible for the larger datasets. In contrast, pruning with the cost based criterion PRU-c does a poor job in neuron selection. This is might be due to the fact that the cost based selection would have considered few neurons as insignificant since they contributed less in training set, but in reality, those neurons are helpful in model generalization with more contribution to the performance in test set. Among the PRU's variants, PRU-a works better in most cases except the compression of FC7 of VGG19 model trained in Oxford 102 dataset. Surprisingly PRU-a outperformed SVD and SLR under few ranks in compressing VGG16 in

Table 5
Compression of FC6 and FC7 layers of VGG16 on CIFAR-10. *k*—rank value for SVD and SLR. Value *cr* is the compression ratio (in%) of SVD and PRU, SLR-cr is compression ratio (in%) of SLR (*sr*=0.5, *rr*=0.5).

Layer	Approach	Rank *k*						
		2	4	8	16	32	64	128
FC6	Compression ratio							
	SVD-cr	0.43	0.87	1.75	3.51	7.03	14.06	28.13
	SLR-cr	0.32	0.65	1.31	2.63	5.27	10.54	21.09
	Accuracy							
	SVD	44.13	73.53	92.61	92.62	92.62	92.62	92.62
	PRU-w	11.44	15.79	24.64	28.51	36.24	79.56	91.95
	PRU-a	13.19	24.52	27.03	28.14	37.59	85.23	92.47
	PRU-c	15.00	15.5	18.67	28.29	29.93	68.51	91.45
	SLR-w	39.31	76.31	92.01	92.59	92.63	92.63	92.63
	SLR-a	38.13	74.3	92.05	92.62	92.64	92.63	92.63
	SLR-c	40.59	72.98	92.16	92.62	92.63	92.63	92.63
FC7	Compression ratio							
	SVD-cr	0.09	0.19	0.39	0.78	1.56	3.12	6.25
	SLR-cr	0.07	0.14	0.29	0.58	1.17	2.34	4.68
	Accuracy							
	SVD	28.00	63.31	92.59	92.65	92.62	92.62	92.62
	PRU-w	10.00	10.00	10.00	10.00	10.15	23.32	68.02
	PRU-a	10.00	10.00	10.00	10.00	23.07	72.50	91.11
	PRU-c	10.00	10.00	10.00	10.00	10.00	10.00	10.00
	SLR-w	28.03	63.91	92.23	92.65	92.63	92.63	92.63
	SLR-a	28.04	63.33	92.30	92.66	92.62	92.63	92.63
	SLR-c	28.04	63.76	92.36	92.60	92.62	92.63	92.63

Table 6
Compression of FC6 and FC7 layer of VGG16 for CIFAR-100.

Layer	Approach	Rank *k*						
		2	4	8	16	32	64	128
FC6	Compression ratio							
	SVD-cr	0.43	0.87	1.75	3.51	7.03	14.06	28.13
	SLR-cr	0.32	0.65	1.31	2.63	5.27	10.54	21.09
	Accuracy							
	SVD	5.49	21.03	62.29	67.50	67.55	67.58	67.57
	PRU-a	1.00	1.00	1.22	2.24	7.96	25.94	58.60
	SLR-c	4.92	14.53	55.77	67.04	67.57	67.61	67.57
FC7	Compression ratio							
	SVD-cr	0.09	0.19	0.39	0.78	1.56	3.12	6.25
	SLR-cr	0.07	0.14	0.29	0.58	1.17	2.34	4.68
	Accuracy							
	SVD	6.73	19.03	61.53	67.43	67.48	67.55	67.54
	PRU-a	1.00	1.00	1.00	1.00	1.73	2.41	10.90
	SLR-c	6.65	18.61	62.36	67.21	67.45	67.58	67.57

Table 7
Compression of FC3 layer of Lenet5 for MNIST.

Approach	Rank *k*					
	2	4	8	16	32	64
	Compression ratio					
SVD-cr	2.17	4.34	8.68	17.36	34.73	69.46
SLR-cr	1.62	3.25	6.51	13.03	26.06	52.13
	Accuracy					
SVD	37.8	61.71	86.5	97.15	98.26	98.56
PRU-w	31.44	45.08	63.06	79.02	93.29	98.03
SLR-c	38.84	56.02	69.87	95.45	98.16	98.50

Table 8
Compression of FC6 and FC7 layer of VGG16 for dogs vs. cats.

Layer	Approach	Rank k						
		2	4	8	16	32	64	128
FC6	Compression ratio							
	SVD-cr	0.05	0.11	0.22	0.45	0.90	1.81	3.63
	SLR-cr	0.01	0.01	0.02	0.05	0.10	0.21	0.43
	Accuracy							
	SVD	50.00	53.25	89.34	88.34	90.19	96.49	96.99
	PRU-a	62.60	72.25	82.40	91.69	93.24	94.14	95.39
	SLR-a	50.00	53.25	87.94	90.99	89.94	96.09	96.84
FC7	Compression ratio							
	SVD-cr	0.09	0.19	0.39	0.78	1.56	3.12	6.25
	SLR-cr	0.07	0.14	0.29	0.58	1.17	2.34	4.68
	Accuracy							
	SVD	50.85	54.10	79.95	89.89	93.79	95.64	96.34
	PRU-a	55.25	77.65	89.94	92.14	94.89	93.49	92.44
	SLR-a	50.30	55.35	83.00	90.69	93.54	95.39	96.14

Table 9
Compression of FC6 and FC7 layer of VGG19 for Oxford 102.

Layer	Approach	Rank k						
		4	8	16	32	64	128	256
FC6	Compression ratio							
	SVD-cr	0.11	0.22	0.45	0.90	1.81	3.63	7.27
	SLR-cr	0.01	0.02	0.05	0.10	0.21	0.43	5.45
	Accuracy							
	SVD	8.05	30.05	65.50	79.80	83.05	84.10	84.80
	PRU-a	8.10	12.90	23.90	40.15	58.65	70.60	79.25
	SLR-a	7.55	27.85	63.35	79.05	82.75	83.90	84.00
FC7	Compression ratio							
	SVD-cr	0.19	0.39	0.78	1.56	3.12	6.25	12.50
	SLR-cr	0.14	0.29	0.58	1.17	2.34	4.68	9.37
	Accuracy							
	SVD	15.15	44.40	68.45	77.25	80.75	83.00	84.15
	PRU-w	7.45	7.60	14.25	23.80	43.10	71.30	79.15
	SLR-a	13.50	42.60	67.25	76.20	79.55	82.10	83.75

Table 10
Best compression results of SVD and SLR. *C* is the ratio between compression ratios of SVD and SLR.

S. no	Dataset	SVD			SLR				
		k	cr	acc	k	sr, rr	cr	acc	C
1	CIFAR-10	6	0.29	91.57	6	0.7,0.0	0.08	**91.84**	**3.6**
2	CIFAR-10	1	0.04	18.79	6	0.8,0.0	0.04	**86.82**	1.0
3	CIFAR-100	9	0.44	**66.60**	9	0.4,0.0	0.26	66.50	**1.6**
4	CIFAR-100	5	0.24	31.74	9	0.5,0.0	0.21	**65.89**	1.1
5	MNIST	16	17.36	**97.15**	16	0.3,0.5	14.76	96.40	**1.1**
6	MNIST	10	10.85	90.85	12	0.5,0.7	10.85	**92.56**	1.0
7	Dogs vs. Cats	46	2.24	**95.69**	46	0.5,0.0	1.12	95.55	**2.0**
8	Dogs vs. Cats	9	0.44	85.25	46	0.8,0.0	0.44	**93.85**	1.0
9	Oxford 102	112	5.46	82.60	112	0.3,0.5	4.64	**82.60**	**1.1**
10	Oxford 102	33	1.61	77.15	56	0.4,0.0	1.64	**79.35**	1.0

Cats vs. Dogs dataset. In this specific case of transfer learning model, performance of PRU indicates that the total removal of few input or output neuron weights was better than keeping all of the neuron represented in low-rank subspace.

In most cases, SLR and PRU approaches exploiting activation-based significance estimation outperform weight-based significance. This is due to the fact that activations of the neurons are much important rather than their weight magnitudes. This fact is further proven under the transfer learning scenario where SLR-a performs better than the SLR-w since the activation statistics for the target domain is more important than the magnitudes of the fine-tuned weights of the base network. These performance differences among the neuron selection strategies clearly indicate that the choice of these strategies is totally different to models and dataset where it solely depends upon the experimental results.

Table 10 shows the best compression results achieved by SVD and proposed SLR. Here compression is performed on FC7 layer of all the models except for Lenet5. Also, SLR-a is experimented for all the datasets except Lenet5 which uses SLR-c. The parameters k, sr, and rr were tuned to achieve best possible compression with minimal loss in accuracy. Rank k in SVD is chosen as the possible minimum value that has minimal drop (max. of 1% drop) in original test accuracy.

Our approach SLR-a exhibits minimal or no loss in accuracy with comparatively $1.1\times$ to $3.6\times$ smaller compression ratio for the same k of SVD. Even number of index entries (i.e. S. No 2, 4, 6, 8, and 10) in the table show that the SVD has to reduce its rank k into a very small number to achieve compression ratio of SLR that aims for very low compression ratio (accuracy drop from 2% to max. of 6%) where SVD's very low rank k results in huge loss in accuracy. It was found that SLR achieves minimal cr by representing most of rows and columns in U and V^T matrices with reduced rank rk=0. This conveys that most of the input and output neurons are not contributing to the weight matrix reconstruction for model performance. However, those neuron weights cannot be permanently removed using pruning PRU which may result in drastic drop in accuracy. Hence, SLR with rr=0 can be considered as applying pruning to the low-rank factorization. In few cases, reduction rate $0.5 \le rr \le 0.7$ works better for achieving better compression.

Table 11 shows the influence of hyperparameters sparsity rate sr and reduction rate rr in SLR's performance on all testing models. These experiments were conducted on all the testing models with optimal k value obtained using even number of index entries in Table 10. It can be noted that all different models behave in the same manner while adjusting the sparsity rate sr and reduction rate rr. However, the impact is witnessed more in the models that are trained from scratch than in the transfer learning models. This happens since the transfer learning models are bigger models with larger optimal k value (max of 1% drop in accuracy), which is already sufficient enough for the model to achieve good accuracy after weight matrix reconstruction. The impact of sr and rr will be more on the transfer learning models if the base rank k is a smaller value (as in Table 10).

4.3.1. Influence of reduction rate *rr*

The results show that smaller reduced ranks rk achieved by smaller reduction rate rr have huge effect on both compression ratio and accuracy as the number of neurons to be sparsified using sparsity rate sr increases. This happens since large number of neurons represented with smaller reduced ranks will represent highly contributing neurons also with lower ranks, which leads to good compression but poor accuracy. Larger reduction rate rr and sparsity rate sr do not have comparatively much impact on compression as lot of neurons will represented using reduced rank rk, which is close to original rank k.

4.3.2. Influence of sparsity rate *sr*

Smaller sparsity rate sr with smaller reduction rate rr do not impact the SLR's compression ratio and accuracy as the small number of non-contributing neurons are represented with very small reduced rank rk. As in Table 11, the reduction rate rr does not impact much on the accuracy of CIFAR-10 model with sr=0.4, CIFAR-100 model with sr=0.2, MNIST model with sr=0.2, Dogs vs. Cats and Oxford 102 models with sr=0.6. This behavior shows the overall percentage insignificant neurons in a layer. Value sr=0.4 shows that 40% input and output neurons in CIFAR-10 are not contributing much to the model performance. This percentage is different for individual models. In general, the percentage of non-contributing or insignificant neurons increases as the layer size increases. This can be validated from Table 10 where the best compression for small layer FC3 for the MNIST is achieved only by

Table 11
Influence of reduction rate and sparsity rate. Performance of SLR-a with optimal rank k values compressing FC6 layer of VGG16, VGG19 and FC3 of Lenet tested on corresponding datasets under different reduction and sparsity rates.

Model setup	Sparsity rate sr	Reduction rate rr							
		0.2		0.4		0.6		0.8	
		cr	acc	cr	acc	cr	acc	cr	acc
SLR-a with $k = 6$ applied to FC6 layer of VGG16 tested on CIFAR-10	0.2	1.09	92.59	1.14	92.53	1.18	92.55	1.23	92.55
	0.4	0.87	92.40	0.96	92.30	1.05	92.18	1.14	92.25
	0.6	0.65	88.42	0.79	90.77	0.92	90.55	1.05	91.66
	0.8	0.44	49.05	0.61	57.92	0.79	68.49	0.96	86.56
SLR-a with $k = 9$ applied to FC6 layer of VGG16 tested on CIFAR-100	0.2	1.62	66.48	1.71	66.80	1.80	66.82	1.89	66.70
	0.4	1.27	63.00	1.45	63.86	1.62	65.48	1.80	66.16
	0.6	0.92	27.77	1.18	43.09	1.45	58.59	1.71	63.78
	0.8	0.57	04.44	0.92	17.16	1.27	38.82	1.62	58.73
SLR-a with $k = 16$ applied to FC3 layer of Lenet tested on MNIST	0.2	14.55	94.56	15.2	95.77	15.85	95.91	16.5	96.73
	0.4	11.73	84.64	13.03	92.48	14.33	94.49	15.63	95.70
	0.6	8.91	74.02	10.86	89.68	12.81	93.00	14.76	94.62
	0.8	6.10	58.97	8.70	85.43	11.30	91.41	13.90	94.05
SLR-a with $k = 46$ applied to FC6 layer of VGG16 tested on Dogs vs. Cats	0.2	1.09	95.1	1.14	95.00	1.19	95.10	1.24	95.00
	0.4	0.88	95.45	0.98	94.65	1.09	95.05	1.19	94.80
	0.6	0.67	95.00	0.82	93.50	0.98	94.45	1.13	94.40
	0.8	0.46	93.55	0.67	92.55	0.87	93.80	1.07	93.30
SLR-a with $k = 112$ applied to FC6 layer of VGG19 tested on Oxford 102	0.2	2.66	84.05	2.79	83.95	2.92	84.05	3.05	84.05
	0.4	2.15	83.50	2.40	83.65	2.66	83.45	2.91	83.65
	0.6	1.64	82.50	2.02	83.05	2.41	83.45	2.78	83.65
	0.8	1.13	80.55	1.63	82.65	2.15	83.25	2.65	83.60

keeping small number of lesser significant neurons with reduction rate rr between 0.5 and 0.7 where for other larger layers of VGG16 and VGG19, the large number of non-contributing neurons can be better represented with zero sub-ranks rk using reduction rate rr=0 for better compression.

Finally, reduction rate rr has huge effect on accuracy when sr is higher. In this case, only the rr can determine how much rank to reduce for all neurons. For sr=1.0, the SLR model behaves like truncated SVD where rr can solely make the rank rk equal to original rank k. In general, both sr and rr have equal importance in attaining best the compression ratio where both of these parameters have to be tuned for the specific model and dataset.

4.3.3. *Choosing ranks* k *and* rk

As discussed earlier, selection of rank k in SVD for deep learning model compression solely depends on preferred or tolerable drop in testing accuracy after the model is compressed using low-rank decomposition. In our proposed SLR, the model is initially compressed with different k values starting from larger where reduction rate rr and sparsity rate sr are kept as 0. At this stage the proposed approach behaves as naïve truncated SVD without any sparsity. Once the suitable rank k is obtained, then different sparsity rate sr can be tested for a fixed reduction rate rr (rr=0.5 for example). Then, the sparsity rate sr can be analyzed to find the percentage of insignificant or non-contributing neurons in a layer. For the chosen sr, the model can be further experimented with different reduction rate rr, which eventually gives the preferred reduced rank rk. Both rank k and reduced rank rk of the proposed approach SLR can be tuned in this way.

5. Conclusion and future work

We proposed a novel sparse low-rank decomposition approach SLR for deep neural network compression. Our approach aims to compress the results of vanilla Singular Value Decomposition by sparsifying insignificant rows and columns of its factor matrices. Selection of neurons for sparsification was determined based on factors, such as absolute weight, activations, or change in cost. At the core, the proposed approach contributes to the research in neural network model compression by incorporating the sparsity into low-rank decomposition approach. Experiments were conducted on popular CNN architectures trained on variety of datasets under different training setups. The comparative results have shown that our proposed approach attains even up to 3.6× smaller compression ratio in compressing SVD, with marginal or no loss in accuracy. This indicates that we can save 3.6× storage space of SVD without much effect on the model performance. The structured sparsity achieved by the proposed approach also has advantage of speedup in the computation. This provides our approach a better storage and speed optimization for dense layer weights comparing to the traditional truncated SVD and pruning baseline methods. Our approach has constant sparsity rate for both input and output neurons, which are chosen for sparsification. However, efficient way to determine different sparsity rate for input and output neurons, and combination of multiple neuron selection criterion will be explored in the future.

Declaration of Competing Interest

None.

Acknowledgment

We acknowledge the project Leadingindia.ai (IAPP1R2\100197) −−Nationwide Initiative on “AI and Deep Learning Skilling and Research.”

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2020.02.035.

References

[1] V. Lebedev, V. Lempitsky, Speeding-up convolutional neural networks: a survey. Bulletin of the Polish academy of sciences, Tech. Sci. 66 (2018) 799–810.
[2] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, IEEE Signal Process. Mag. (2018).
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–773.

[5] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: efficient convolutional neural networks for mobile vision applications, (2017) arXiv:1704.04861.
[6] Y.W.Q.H. Jiaxiang Wu, Cong Leng J. Cheng, Quantized convolutional neural networks for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4820–4828.
[7] N. Floropoulos, A. Tefas, Complete vector quantization of feedforward neural networks, Neurocomputing 367 (2019) 55–63.
[8] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-Net: imagenet classification using binary convolutional neural networks, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 525–542.
[9] F. Li, B. Zhang, B. Liu, Ternary weight networks, Proceedings of the International Conference on Learning Representations, 2017, 1–10.
[10] Y. Cheng, F.X. Yu, R.S. Feris, S. Kumar, A.N. Choudhary, S. Chang, Fast neural networks with circulant projections, CoRR (2015).
[11] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, Z. Wang, Deep fried convnets, in: Proceedings of the International Conference on Computer Vision, 2015, pp. 1476–1483.
[12] J. Ba, R. Caruana, Do deep nets really need to be deep? in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 2654–2662.
[13] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: hints for thin deep nets, Proceedings of the International Conference on Learning Representations (2015) 1–13.
[14] J. Xu, Y. Nie, P. Wang, A.M. López, Training a binary weight object detector by knowledge transfer for autonomous driving, in: Proceedings of the International Conference on Robotics and Automation, 2019, pp. 2379–2384.
[15] Y. He, J. Lin, Z. Liu, H. Wang, L.J. Li, S. Han, Amc: automl for model compression and acceleration on mobile devices, in: Proceedings of the Proceedings of the European Conference on Computer Vision, 2018, pp. 784–800.
[16] Y. Chen, T. Yang, X. Zhang, G. Meng, C. Pan, J. Sun, DetNAS: neural architecture search on object detection, Proceedings of the Conference on Neural Information Processing Systems, 2019.
[17] A.H. Ashouri, T.S. Abdelrahman, A. Dos Remedios, Retraining-Free methods for fast on-the-fly pruning of convolutional neural networks, Neurocomputing 370 (2019) 56–69.
[18] M. Zhu, S. Gupta, To prune, or not to prune: exploring the efficacy of pruning for model compression, Proceedings of the NIPS Workshop on Machine Learning of Phones and other Consumer Devices, 2017.
[19] J.M. Alvarez, M. Salzmann, Learning the number of neurons in deep networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 2262–2270.
[20] H. Zhou, J.M. Alvarez, F. Porikli, Less is more: towards compact CNNs, in: Proceedings of the European Conference on Computer Vision, 2016, p. 9908.
[21] D. Mehta, K.I. Kim, C. Theobalt, On implicit filter level sparsity in convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 520–528.
[22] X. Ding, G. Ding, Y. Guo, J. Han, Centripetal SGD for pruning very deep convolutional networks with complicated structure, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4943–4953.
[23] M. Masana, J. van de Weijer, A.D. Bagdanov, On-the-fly network pruning for object detection, Proceedings of the International Conference on Learning Representations - workshop track, 2016.
[24] T. Gale, E. Elsen, S. Hooker, The state of sparsity in deep neural networks, (2019) arXiv:1902.09574.
[25] E.L. Denton, W. Zaremba, J. Bruna, Y. LeCun, R. Fergus, Exploiting linear structure within convolutional networks for efficient evaluation, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 1269–1277.
[26] J. Xue, J. Li, Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, in: Proceedings of the Interspeech, 2013, pp. 2365–2369.
[27] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, N.D. Freitas, Predicting parameters in deep learning, in: Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 2148–2156.
[28] T.N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, B. Ramabhadran, Low-rank matrix factorization for deep neural network training with high-dimensional output targets, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6655–6659.
[29] R. Girshick, Fast r-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[30] M. Kholiavchenko, Iterative low-rank approximation for CNN compression, (2018) arXiv:1803.08995.
[31] M. Masana, J. van de Weijer, L. Herranz, A.D. Bagdanov, J.M. Alvarez, Domain-adaptive deep network compression, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4289–4297.
[32] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, B. Yu, Recent advances in convolutional neural network acceleration, Neurocomputing 323 (2019) 37–51.
[33] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, W.J. Dally, Exploring the regularity of sparse structure in convolutional neural networks, Proceedings of the Conference on Neural Information Processing Systems, 2017.
[34] V. Lebedev, V. Lempitsky, Fast convnets using groupwise brain damage, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2554–2564.
[35] C. Louizos, M. Welling, D.P. Kingma, Learning sparse neural networks through L0 regularization, Proceedings of the International Conference on Learning Representations (2018) 1–13.
[36] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the International Conference on Computer Vision, 2017, pp. 2755–2763.
[37] C. Yang, Z. Yang, A.M. Khattak, L. Yang, W. Zhang, W. Gao, M. Wang, Structured pruning of convolutional neural networks via L1 regularization, IEEE Access 7 (2019) 106385–106394.
[38] S. Han, H. Mao, W.J. Dally, Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, in: Proceedings of the International Conference on Learning Representations, 2016, pp. 1–14.
[39] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient DNNs, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 1379–1387.
[40] X. Yu, T. Liu, X. Wang, D. Tao, On compressing deep models by low rank and sparse decomposition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7370–7379.
[41] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, Proceedings of the British Machine Vision Conference, 2014.
[42] P. Wang, Q. Hu, Z. Fang, C. Zhao, J. Cheng, Deepsearch: a fast image search framework for mobile devices, ACM Trans. Multimedia Comput. Commun. Appl. 14 (2018) 1–11.
[43] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, V. Lempitsky, Speeding-up convolutional neural networks using fine-tuned cp-decomposition, Proceedings of the International Conference on Learning Representations, 2015.
[44] H.T. Fan, J.W. Hung, X. Lu, S.S. Wang, Y. Tsao, Speech enhancement using segmental nonnegative matrix factorization, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2014, pp. 4483–4487.
[45] Xuelong Li, Guosheng Cui, Yongsheng Dong, Graph regularized non-negative low-rank matrix factorization for image clustering, IEEE Trans. Cybern. 47 (11) (2017) 3840–3853.
[46] Winata, G.I., Madotto, A., Shin, J., Barezi, E.J., Fung, P., On the effectiveness of low-rank matrix factorization for LSTM model compression, Proceedings of the 33rd Pacific Asia Conference on Language, Information and Computation, 253–262, 2019.
[47] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2015) 579–592.
[48] Y. Song, M. Li, X. Luo, G. Yang, C. Wang, Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: a triple factorization-based approach, IEEE Trans. Ind. Informat. 16 (5) (2019) 3006–3017.
[49] X. Luo, H. Wu, H. Yuan, M. Zhou, Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors, IEEE Trans. Cybern. (2019) 1–12. https://ieeexplore.ieee.org/abstract/document/8681714.
[50] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A.C. Ammari, A. Alabdulwahab, Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models, IEEE Trans. Neural Netw. Learn Syst. 27 (3) (2015) 524–537.
[51] X. Luo, M. Zhou, S. Li, Y. Xia, Z.H. You, Q. Zhu, H. Leung, Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data, IEEE Trans. Cybern. 48 (4) (2017) 1216–1228.
[52] S. Rabanser, O. Shchur, S. Günnemann, Introduction to tensor decompositions and their applications in machine learning, (2017) arXiv:1711.10781.
[53] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, Proceedings of the International Conference on Learning Representations, 2017.
[54] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, Proceeding of the International Conference on Learning Representations, 2015, pp. 1–14.
[55] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[56] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. Rep., University of Toronto, 2009 1, 4.
[57] Y. LeCun, C. Cortes, C.J.C. Burges, The MNIST database of handwritten digits, 1998, Available at: http://yann.lecun.com/exdb/mnist/.
[58] J. Elson, J.J. Douceur, J. Howell, J. Saul, Asirra: a captcha that exploits interest-aligned manual image categorization, in: Proceedings of the ACM Conference on Computer and Communications Security, 2007, pp. 366–374.
[59] M.E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: Proceedings of the Indian conference on computer vision, Graphics and Image Processing, 2008, pp. 722–729.
[60] F. Chollet, Keras, 2015, Available at: https://github.com/fchollet/keras.

Sridhar Swaminathan received his B.Sc., M.Sc., and Ph.D. degrees in Computer Science from Bishop Heber College (Autonomous), Tiruchirappalli, India, in 2010, 2012, and 2015, respectively. He is currently an Assistant Professor in the Bennett University, Greater Noida, India. His main interests include deep learning, computer vision, and information retrieval.

Deepak Garg is a Professor, and the Head of the Computer Science Engineering Department, Bennett University, India, and the Head of the NVIDIA-Bennett Center of Research on Artificial Intelligence. His active research interests are designing efficient deep learning algorithms and quality in higher education. He served as the Chair of the IEEE Computer Society, India Council, from 2012 to 2016, and on the Board of Governors of the IEEE Education Society, USA, from 2013 to 2015. He has managed research funding of INR 300 million.

Rajkumar Kannan received the B.Sc. and M.Sc. degrees in computer science from Bharathidasan University, Tiruchirappalli, India, in 1991 and 1993, respectively, and the Ph.D. degree in multimedia datamining from National Institute of Technology, Tiruchirappalli, India, in 2007. Rajkumar works for Bishop Heber College (Autonomous), India in the Department of Computer Science. Previously, he worked for King Faisal University, Saudi Arabia in the Faculty of Computer Sciences and Information Technology. His research activities primarily include the confluence of multimedia, information retrieval, semantic web, social informatics, and collective intelligence. Rajkumar is a member of CSI-India, ISTE-India and senior member of ACM.

Frederic Andres is an Associate Professor in Digital Content and Media Sciences Research Division at National Institute of Informatics, Tokyo, Japan. His current research focuses on distributed semantic services for collective intelligence oriented applications (Cooking Recipe without Border, MindFlow, Agriculture Mass Warning Service, Skill2share, Learning by caption, healing service) and social project management platform. It is a vertical research on advanced model-based architecture platform including specific researches on collective intelligence and semantic management, advanced collaborative portals related to digital humanities and semantic digital library, ontological topic maps-based metadata services, and multi-lingual multi-cultural cross-disciplinary ontology services. Part of this research, he has also created Image-learning Ontology and Human Stress Ontology services.