Received January 22, 2018, accepted February 25, 2018, date of publication February 28, 2018, date of current version March 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2810264

CS-CNN: Enabling Robust and Efficient Convolutional Neural Networks Inference for Internet-of-Things Applications

YIRAN SHEN[1], (Member, IEEE), TAO HAN[1], QING YANG[1], XU YANG[2], YONG WANG[1], FENG LI[3], AND HONGKAI WEN[4]

[1]College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
[2]College of Textile and Light Industry, Inner Mongolia University of Technology, Hohhot 010062, China
[3]School of Computer Science and Technology, Shandong University, Qingdao 266237, China
[4]Department of Computer Science, University of Warwick, Coventry CV4 7EZ, U.K.

Corresponding author: Yong Wang (wangyongcs@hrbeu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702133 and Grant 61702304, in part by the Natural Science Foundation of Heilongjiang Province under Grant QC2017069 and Grant QC2017071, in part by the Fundamental Research Funds for the Central Universities under Grant HEUCFJ160601, in part by the China Postdoctoral Science Foundation under Grant 166875, in part by the Heilongjiang Postdoctoral Fund under Grant LBH-Z16042, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2017QF005.

ABSTRACT Recent advances have shown that convolutional neural networks (CNNs) perform excellent in the tasks of image classification and face recognition when the size of data sets is sufficiently large, i.e., over hundreds of thousands training images. Nevertheless, when public data sets are not suitable for training the model for new application scenarios, it is painful to obtain sufficient training examples, especially when the samples have to be labeled manually. Besides, training and inference using CNNs requires significant resources of energy, computation, and memory usage. Therefore, implanting deep CNN models trained and executed on high performance GPU clusters to resource constrained devices, i.e., Internet of Things (IoT) devices, which have permeated into every aspect of modern life, is not appropriate and impractical. Compression technology is an important and popularly used tool to accelerate the training and inference of the CNN models. In this paper, we aim for a step forward in this area: we propose a new compressed CNN model termed CS-CNN for image classification by incorporating the theory of compressive sensing at the input layer of the CNN models to both reduce the resources consumption (evaluated as computation time in this paper) and a required number of training samples. According to our extensive evaluations on the multiple public data sets for deep learning tasks, e.g., MINST and CIFAR-10, using different metrics, we illustrate that the CS-CNN is able to speed up the process of training and inference by a factor of magnitude. Meanwhile, it achieves higher classification accuracy compared with the traditional large CNN models when the size of training database is small.

INDEX TERMS Convolutional neural network, compressive sensing, singular value decomposition, IoT, image classification.

I. INTRODUCTION

The fast development of computational capability of the hardware enables the portable embedded devices such as smartphones, tablet PC and smart wearables undertaking more computationally intensive tasks, such as images or videos processing. Image classification is one of the most popular adopted image processing techniques in a wide variety of vision-based applications [1], [58], [60]. Its major task is to distinguish different objectives shown in the scene in the images according to the different characteristics extracted from the images. Image classification has broad applications in different research and engineering fields, such as smart city [2], entrance or monitoring security [3], [4], [59], and information search [5]. It is one of the basic and most challenging problem in the field of pattern recognition and machine learning. Finding an intelligent, efficient and accurate classification method is an urgent expectation.

Image classification mainly consists of two major components, first is the extraction of image features, e.g., Scale-Invariant Feature Transform(SIFT) [6] and Local Binary

Pattern (LBP) [7], second is designing a robust classification algorithms, such as Decision Tree, Support Vector Machine(SVM) and Neural Networks. The recent success on deep learning [8]–[17] promotes the development of various classification tasks, including natural language processing [8]–[10], speech recognition [11], [12], action recognition [13], [14], [57] or image classification [15]–[17]. Since LeNet5 [18] was introduced in the early 1990's, Convolution Neural Networks (CNNs) based deep learning models have gradually become one of the best choices for image classification [19]–[21] and face detection [22], [23]. Especially, in recent years, CNNs have achieved state-of-art results in many challenging classification tasks. He *et al.* [24] broke the record on ImageNet 2015 classification benchmark with their CNN model. Most notably, Wang *et al.* [25] achieved state-of-the-art image classification performance by a CNN using attention mechanism.

CNN is a feed-forward neutral network which extracts features using multiple convolutional layers and makes inference using fully-connected layers with softmax [26]. CNN provides reliable classification results and is able to accommodate the image translation, scale difference, rotation and other forms of deformations. In a word, it has good generalization capability on noisy inputs. The success of the CNN models on classification tasks is due to the following factors: 1) the availability of extremely large training sets with labelled groundtruth, e.g., ImageNet [27]; 2) the high speed GPU clusters implementations for training large number of parameters; 3) carefully designed regularization strategies, such as dropout [28], improves the generation capability.

However, the encouraging performance is achieved at extremely high cost of resources on platform and requirement of large amount of labelling effort when useful public datasets are not available. As the number of parameters are huge in CNN, it takes a while to execute the whole network even for the relative simple inference (matrix multiplications and convolutions). Therefore, it is difficult to guarantee real-time response when running on embedded system. Besides, huge amount of labelled data is essential for training a reliable CNN model, otherwise, the issue of over-fitting may occur. However, collecting large dataset especially obtaining the ground-truth is labour intensive for the new application scenarios.

The state-of-the-art solution for accelerating the deep CNN models is to slim the network structure while preserving most of its accuracy by utilising some compression or matrix factorization techniques. For example, the Region-based Convolution Neural Network(R-CNN) [29] achieves excellent performance on automatically extracting and classifying on a natural image however its computation is prohibitive for most of the low-cost platforms as it runs CNN model on over thousands of image region proposals. To speed up R-CNN, the Spatial pyramid pooling(SPP) [30] - net introduces SPP layer to share and schedule the intensive computing of the network and its results to a factor of 24-102 times faster execution time on inference (classifying an image) than fast R-CNN. Fast R-CNN [31] proposes a region of interest (ROI) pooling layer and maps each feature vector into a sequence of fully connected layer to reduce the computation needed for inference. It accelerates the inference process of R-CNN and SPP-net. Ren *et al.* [32] introduces a Region Proposal Network (RPN) that enables nearly cost-free proposals by sharing full-image convolutional network. RPN and Fast R-CNN are merged into Faster R-CNN by sharing their convolutional features. It is an effective way of improving the speed of inference. Another major stream of reducing inference time is to slim the architecture of the deep networks [33]–[35]. Iandola *et al.* [33] propose a small CNN architecture called SqeezeNet to compress the model. SqeezeNet has fewer paramaters but it does not have significant impact on speeding up the process of inference. Hinton *et al.* [34] transfer the knowledge from a large models to a small one to train the small model easier. Zhang *et al.* [35] build an efficient architecture called ShuffleNet which allows more feature channels to encode more information. Besides the method of changing the architecture of the network, some papers compress the weights of the network. Han *et al.* [36] present a compression method "deep compression" to speed up the process by pruning the network, training quantization and Huffman coding. Some work applies matrix decomposition layer-wise either on fully-connected layer [38] or convolution layer [39]. These approaches work on the existing big deep networks trained on the publicly available datasets which on one hand avoids the tedious retraining, however, on the other hand, it is not applicable for new application scenario where large number of training samples are needed to collect and get labelled by the developers.

In this paper, we aim for a step forward to propose a new architecture of CNN model by modifying the input layer meanwhile slimming the width (number of nodes in each layer) of the network. The theory of compressive sensing [37] (CS) is introduced to reduce the dimensionality of the input layer meanwhile extracting most informative features (projections) using the Singular Value Decomposition (SVD) for generating the optimal projection matrix. CS is implemented and computed using the tensor convolution function embedded in tensorflow framework to reduce the system cost introduced by CS. As the number of parameters in input layer is dominant in the total parameters of the whole network, the significant compression of input layer has the potential to avoid the issue of over-fitting when training samples are insufficient.

The contributions of this paper are as follows:

- We propose a new framework. It adds a convolutional layer for compressing with fixed weights between the input layer and the first convolution layer in order to compress raw data and then the useful information are extracted. Parameters of this layer are fixed and not updated during training. It is an optimized CNN and can run on many embedded vision systems and a wide variety of IoT devices.

- We generate the compression matrices in different ways and compare which is the best way to produce it to ensure higher accuracy. Then it will be the best way to compress the original input.
- The accuracy of the proposed method is proved to be almost identical with that of the original network [18], meanwhile, the inference time of the proposed method is remarkably less than that of the original network. It is worth emphasizing that both the accuracy and the inference time of the proposed method are completely superior to that of the original network under the circumstance of insufficient data.
- Due to the exploitation of the SVD-based compression strategy, the proposed method eliminates considerable redundant information which is useless for classification of the original images. Therefore, not only the inference time is decreased significantly, but also the over-fitting phenomenon can be avoided effectively.

The rest of the paper is organized as follows. Section II introduces the background of compressive sensing. Section III describes the datails of the architecture of the original network and the proposed network. Then the experiment is performed and the proposed network is evaluated in Section IV. Finally, we conclude this research in Section V.

II. BACKGROUND OF COMPRESSIVE SENSING

As our approach incorporates compressive sensing into the architecture of convolutional neural networks, to make the paper self-contained, we provide a brief overview on the basis of compressive sensing. Compressive sensing has brought considerately successful applications in computer vision [40]–[42], [59], [60], signal processing [43]–[45] and other research areas.

The major benefit that we borrow from compressive sensing is its strength on reducing the dimensionality of the original signal while preserving most of its information. Given a vector $x \in R^n$ where its dimensionality n is huge, compressive sensing can be applied if the sparse representation condition holds. A signal is which is sparse in some transform domain $\Psi \in R^{n\times n}$ can be expressed as,

$$x = \Psi\theta \tag{1}$$

where $\theta \in R^n$ is the sparse representation of x in the transform basis Ψ. θ is sparse or compressible if it only contains few dominant non-zero coefficients and rest of the coefficients are all zeros (sparse) or close to zeros (compressible).

When the sparse condition holds, compressive sensing can be applied to reduce the dimensionality of the original vector x as with simple matrix multiplication,

$$y = \Theta x \tag{2}$$

where $\Theta \in R^{m\times n}$ is a projection matrix with $m \ll n$. Therefore the dimensionality of the projection vector $y \in R^m$ is significantly smaller than the original signal x.

In compressive sensing, random matrices are usually used as the projection matrix to reduce the dimensionality of the original signal [46]. However, as the intensive discussion, random projections are never optimal or stable due to the randomness introduced therefore different methods on optimising projection matrices have been proposed [47]–[50]. In this paper, we use Singular Value Decomposition(SVD) to produce the random projection matrix [51]–[53].

Suppose $D \in R^{n\times l}$ is a matrix consists of many signal observations as its columns, we can apply SVD to decompose the data matrix to reveal projection matrix,

$$D = U\Sigma V^T \tag{3}$$

where $U \in R^{n\times n}$ and $V \in R^{l\times l}$ are unitary matrices and $\Sigma \in R^{n\times l}$ is a diagonal matrix whose elements on diagonal are eigenvalues. Then the projection matrix Θ is formed of the m rows from the transpose of U corresponding to the first m largest eigenvalues in Σ.

III. COMPRESSIVE SENSING BASED CONVOLUTIONAL NEURAL NETWORKS

In this section, we introduce our proposed innovative deep learning framework, i.e., CS-CNN, basing on convolutional neural networks (CNN).

A typical CNN consists of a input layer, multiple convolutional layers, pooling layers, fully-connected layers and an output layer. To reduce the model complexity while preserving most of the information, our proposed CS-CNN as shown in Fig. 1 modifies the input layer of the typical CNN architecture by incorporating SVD-based compressive projections. The details of the framework are described as follows.

A. PRELIMINARY OF CONVOLUTIONAL NEURAL NETWORK

The input layer of CNN generally takes the original data without any special processing therefore the number of nodes in input layer is determined by the size of the input data, e.g., the resolution of an image. The convolutional layers are named by the convolution computation. They are also regarded as the key to extract the most discriminate features from the original data space. They generate feature maps by convolutional kernels followed by nonlinear activation functions, such as sigmoid function, tanh function and rectified linear units (ReLu) function.

Suppose the input of the convolutional layer is X, then the feature map generated by the convolutional layer can be expressed as

$$Y = f(Conv(X, W) + b) \tag{4}$$

where W is the convolutional kernel, b is the bias, $Conv(X, W)$ stands for convolution of X and W, $f(\cdot)$ is the activation function. The layer following the convolutional layer is the pooling layer. The convolved features are partitioned into some sub-regions. Then in each sub-region, maximum pooling or averaging pooling is selected. The features extracted by the alternation of several convolutional layers and pooling layers can be utilized for classification through fully connected layers. The output layer usually is a classifier,

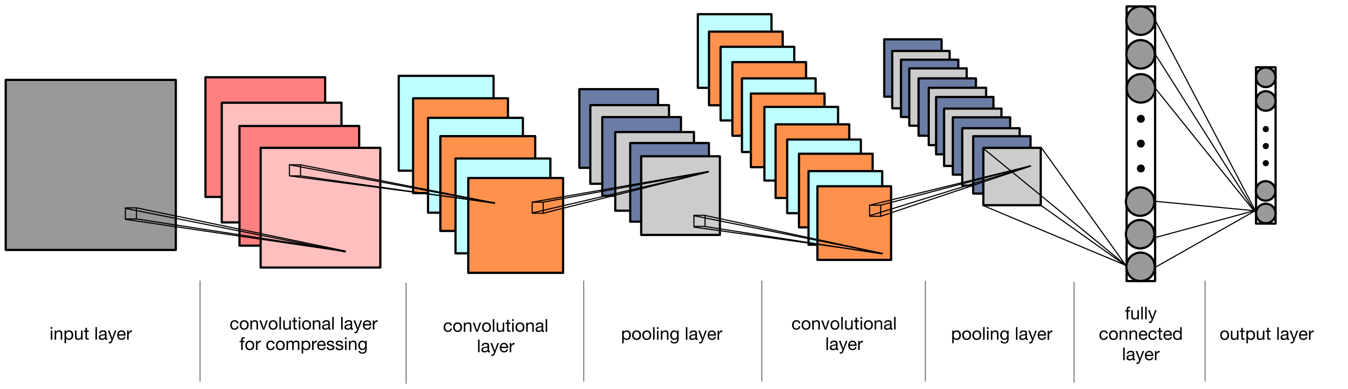


FIGURE 1. **The structure of CS-CNN.**

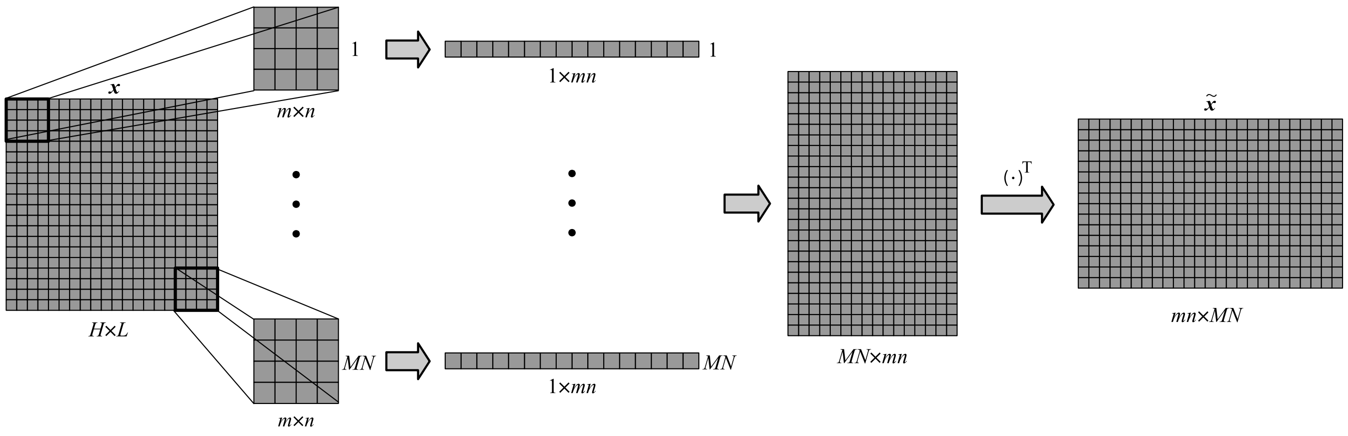


FIGURE 2. **The pretreatment process of the original image.**

such as softmax classifier and support vector machine (SVM) classifier. The classifier transforms the output of the fully connected layer into a probability distribution, each element of the distribution is a scalar ranging from 0 to 1 and it presents the probability of the input corresponding with the class in the classifier. And the class related to the maximum value of the probability is just the correct classification of the input.

As described above, the CNN network essentially is a mathematical model that transforms the input image X into a new feature expression Y in the form of probability matrix by passing through multiple layers.

$$Y = P(L_{min}|X; (W, b)) \tag{5}$$

in which L is the loss function of the network. The training objective is to minimize the L of the network and the parameters of the network are updated layer by layer.

B. PROCESSING OF IMAGE

The original images usually have a mass of redundant information which is useless for classification and may reduce the performance of the CNN networks. So it is sensible to employ the CS to extract the most useful information of the original images before they are input into the CNN. The image processing step of the CS method is introduced as following.

First, the original image needs to be preprocessed before being compressed. As shown in Fig. 2, the original image x with the size of $H \times L$ is divided into MN sub-images and the size of each sub-image is $m \times n$, where $m = H/M$, $n = L/N$. Then the elements of each sub-image is arranged in order into a long fragment with size of $1 \times mn$ and MN long fragments can be obtained totally in this way. Combining these MN long fragments together to form a $MN \times mn$ image and transposing it, we can form a new image $\tilde{x}$ with size of $mm \times MN$, which is just the image that will be applied to compress.

Secondly, the projection matrix Θ is generated by applying the SVD, which has been introduced in the Section II. For this method, matrix D is built at first by combining k (a positive integer) images $\tilde{x}$ by their columns without overlapping. Evidently, the dimensionality of D is $mn \times kMN$. Next, the SVD operation is implemented on D. For better readability, we rewrite (3) as:

$$D = U \Lambda V^{\mathrm{T}} \tag{6}$$

where $^{\mathrm{T}}$ denotes matrix transpose, $\Lambda \in R^{mn\times kMN}$ is a diagonal matrix whose diagonal elements are singular values and in decreasing order, $U \in R^{mn\times mn}$ and $V \in R^{kMN\times kMN}$ are unitary matrices. Finally, the wanted projection matrix Θ can be constructed by extracting the first p rows of the transpose of the unitary matrix U, i.e., the rows corresponds to the first p largest singular values in Λ. Obviously, the size of Θ formed in this way is $p \times mn$.

Third, we compress the images with the projection matrix Θ produced above. As shown in Fig. 3, the $mn \times MN$ image $\widetilde{x}$, transformed from the original image x, is compressed by Θ into a image noted as y with the size of $p \times MN$. Based on the assumption that $p \ll mn$, it's obvious that the dimensionality of $\widetilde{x}$ is reduced significantly.

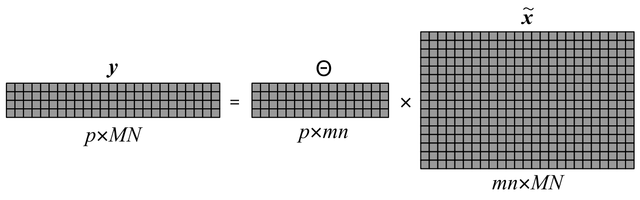


FIGURE 3. **The process of image compression.**

Finally, the compressed image y is transformed into a new image $\widetilde{y}$ with p channels, which is prove to be easier for the CNN to process. As illustrated in Fig. 4, the rows of y are used to form the channels of $\widetilde{y}$. In detail, for example, we divide the first row of y uniformly into M vectors with the size of $1 \times N$, then integrate these M vectors by row into a $M \times N$ matrix, which is regarded as a channel of $\widetilde{y}$. After several similar operations implementing on the other rows of y, the new image $\widetilde{y}$ with p channels can be obtained.

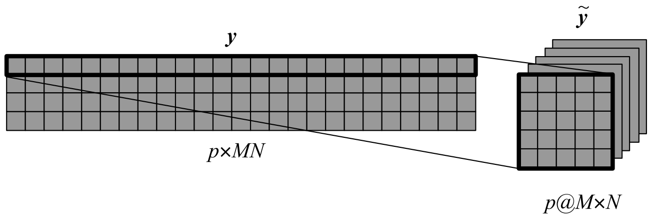


FIGURE 4. **The process of transforming the compressed image y into the image $\widetilde{y}$ with p channels.**

C. CS-CNN

To get the image $\widetilde{y}$ with p channels, as mention above, we have to transform the original image x into $\widetilde{x}$, compress $\widetilde{x}$ with Θ, and transform the compressed image y into $\widetilde{y}$. Undoubtedly, all these processes need several complicated steps to achieve and are time-consuming especially when the number of the original image is large. To reduce the complexity significantly, we replace the projection matrix Θ in CS with convolution filters. As shown in Fig. 5, p convolution filters noted as W_i, $i = 1, 2, \cdots, p$ are constructed by reshaping the $p \times mn$ projection matrix Θ. The process of this construction is identical with the process of transforming y into $\widetilde{y}$ shown in Fig. 4, so we don't interpret it again. Once the convolution

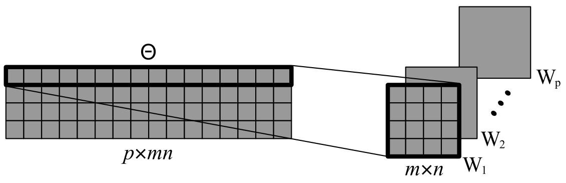


FIGURE 5. **The process of transforming the projection matrix into the convolution filters.**

filters are acquired, the image $\widetilde{y}$ can be obtained directly by executing a convolution on the original image x and the convolution filters.

Through replacing the projection matrix with the convolution filters, we can call the mature program of convolution to achieve the compression of images. What's more, the process of the images compression can be regard as a convolutional layer without overlapping, so we can put it between the input layer and the first convolutional layer of CNN to form a new framework named as CS-CNN, whose performance will be demonstrated to be superior to that of the troditional CNN in the Section IV. It should be noted that the parameters of this layer are fixed and not updated during training.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we evaluate CS-CNN and compare its performance with traditional CNN using two datasets: MNIST [54] and CIFA-10 [55]. We use multiple evaluation metrics including classification accuracy, inference speed and the performance with different sizes of training set.

A. MNIST

The MNIST dataset is composed of 60,000 training examples and 10,000 testing examples in total, spreading over hand written digits 0-9. The size of each digit image is 28×28. For comparison, we adopt LeNet-5 [18] as the benchmark CNN model and we refer to it as LeNet-5-like in this paper. We insert the SVD-based CS layer into the input layer to form the CS-CNN. The detailed setup of the LeNet-5-like network and the corresponding CS-CNN is shown in TABLE 1. The term *num* in the table stands for the number of projections after operating SVD-based compressive sensing. We use 10 images from each digit to learn the SVD projection matrices to get the SVD matrix, then we divide each image into 7×7 patches with the size of 4×4 to do SVD-based projection. To investigate its performance improvement, we also include the naive image scale technique, i.e., averaging the pixel values within a small squared area, which is termed as Mean-CNN.

To evaluate the inference speed of CS-CNN, we test it on the task of MNIST image classification and the test result is shown in Fig. 6. From the Fig. 6, we can get that the inference time of CS-CNN and Mean-CNN ($num = 1, 2, 4$) is about 0.13s while that of the LeNet-5-like is about 0.70s. In other word, our CS-CNN is similar to Mean-CNN and has almost $6\times$ inference time reduction. At the same time,

TABLE 1. The structure of LeNet-5-like and CS-CNN.

Network	Layer	Purpose	Filter	Number of filters	Stride	Weights	Bias
LeNet-5-like	1	Input layer					
	2	Convolutional layer+ReLu	5×5	32	[1,1]	$5 \times 5 \times 1 \times 32$	1×32
	3	Convolutional layer+ReLu	5×5	64	[1,1]	$5 \times 5 \times 32 \times 64$	1×64
	4	Fully connected layer				50176×1024	1×1024
	5	Fully connected layer+ReLu				1024×84	1×84
	6	Output layer+				84×10	1×10
CS-CNN	1	Input layer					
	2	Compressing layer	4×4	num	[4,4]	$4 \times 4 \times 1 \times num$	
	3	Convolutional layer+ReLu	5×5	32	[1,1]	$5 \times 5 \times num \times 32$	1×32
	4	Convolutional layer+ReLu	2×2	64	[1,1]	$2 \times 2 \times 32 \times 64$	1×64
	5	Fully connected layer				3136×1024	1×1024
	6	Fully connected layer				1024×84	1×84
	7	Output layer				84×10	1×10

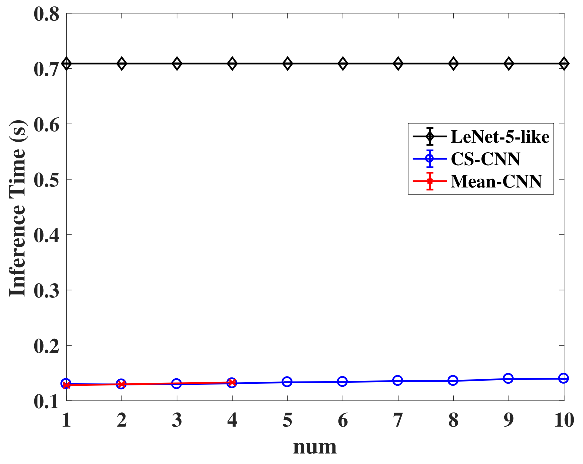


FIGURE 6. The inference time of LeNet-5-like, CS-CNN and Mean-CNN.

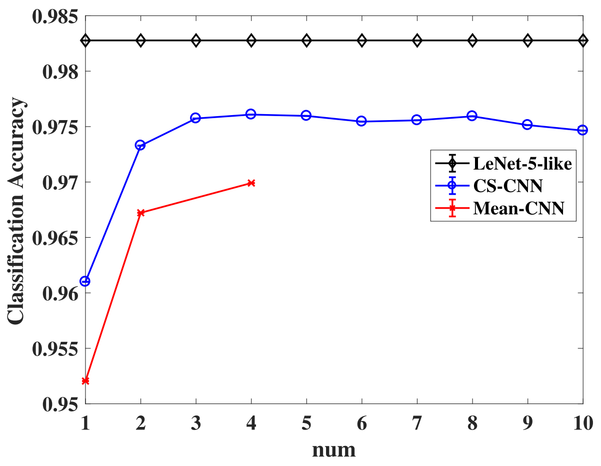


FIGURE 7. The classification accuracy of LeNet-5-like, CS-CNN and Mean-CNN.

the accuracy of these networks is simulated and the result is shown in Fig. 7. It can be seen from Fig. 7 that with the *num* increasing from 1 to 10, the accuracy of CS-CNN changes from 0.961 to 0.976 accordingly, which is quite close to the accuracy of the LeNet-5-like. What's more, the accuracy of the CS-CNN is better than that of Mean-CNN when the $num = 1, 2, 4$. By connecting the simulation results shown in Fig. 6 and Fig. 7, we can come to the conclusion that our CS-CNN achieves significant reduction of inference time with its accuracy quite close to that of the LeNet-5-like.

Furthermore, to verify the CS-CNN is superior to the LeNet-5-like under the circumstance that the training set is insufficient, we compare the performance of the LeNet-5-like, CS-CNN and Mean-CNN in some small training sets. The new small training sets for this experiment consist of 100, 200, 300, 500, 700, 1000, 1500, 2000, 3000, 5000, 7000, 10000 training examples respectively. Then we train the LeNet-5-like, CS-CNN and Mean-CNN with the $num = 4$ with these training sets. And afterwards we evaluate their performance in the classification accuracy on the testing set, the result of which is shown in Fig. 8. Fig. 8 indicates that the accuracy of the CS-CNN with $num = 4$ is higher than that of the other two networks obviously. The accuracy of the Mean-CNN with $num = 4$ is always almost the same with or a little higher than that of the LeNet-5-like. The result has demonstrated that the CS-CNN performs best among these three methods in the situation that the training set

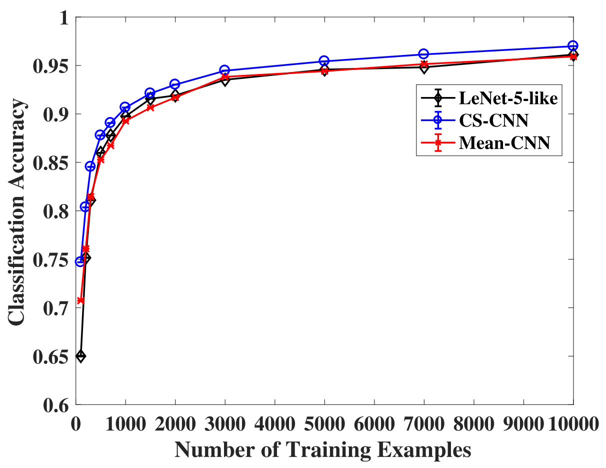


FIGURE 8. The classification accuracy in small training dataset.

TABLE 2. The structure of LeNet-5-like and CS-CNN.

Network	Layer	Purpose	Filter	Number of filters	Stride	Weights	Bias
LeNet-5-like	1	Input layer					
	2	Convolutional layer+ReLu+LRN	5×5	64	[1,1]	$5 \times 5 \times 3 \times 64$	1×64
	3	Convolutional layer+ReLu+LRN	5×5	64	[1,1]	$5 \times 5 \times 64 \times 64$	1×64
	4	Fully connected layer				65536×1024	1×1024
	5	Fully connected layer+ReLu				1024×192	1×192
	6	Output layer+				192×10	1×10
CS-CNN	1	Input layer					
	2	Compressing layer	4×4	num	[4,4]	$4 \times 4 \times 3 \times num$	
	3	Convolutional layer+ReLu+LRN	5×5	64	[1,1]	$5 \times 5 \times 3num \times 64$	1×64
	4	Convolutional layer+ReLu+LRN	5×5	64	[1,1]	$5 \times 5 \times 64 \times 64$	1×64
	5	Fully connected layer				4096×1024	1×1024
	6	Fully connected layer				1024×192	1×192
	7	Output layer				192×10	1×10

is insufficient. It can be explained with the reason that the SVD matrix contains the main information of the images so that the compressed image can preserve most useful information while the scale of the image is decreased significantly.

B. CIFAR-10

The CIFAR-10 dataset consists of 50000 training examples and 10000 testing examples in total, spread over 10 classes of natural images. All the images in this dataset are RGB images of size 32×32. For this dataset, we add the Local Response Normalization(LRN) [56] after each convolutional layer. The details of the architecture of the LeNet-5-like and the corresponding CS-CNN are shown in TABLE 2. Different from the operation in the MINIST dataset, We use 20 images from each class to get the SVD matrix, then we divide each image into 8×8 patches with the size of 4×4 to do SVD-based projection and Mean-based projection.

Similarly, we compare our method CS-CNN with the LeNet-5-like and the Mean-CNN on this dataset to evaluate their performance in terms of the inference speed and the classification accuracy, and the comparison results of performance are shown in Fig. 9 and Fig. 10. The inference time of CS-CNN shown in Fig. 9 is always about 32s within the range that the *num* changes from 1 to 10, which is less than 50% of the inference time of the LeNet-5-like 69s. In addition, the inference time of the Mean-CNN is almost equal to that of the CS-CNN when the $num = 1, 2, 4, 8$. The variation tendency of the accuracy shown in Fig. 10 is also consistent with that shown in Fig. 7. On the basis of the above analysis, we can get a same conclusion as that given in the subsection MINIST that the proposed method CS-CNN takes less inference time than LeNet-5-like and keeps its accuracy quite close to that of the LeNet-5-like at the same time.

Finally, parts of the training set are chosen to construct several small training sets to evaluate the performance of LeNet-5-like, CS-CNN, and Mean-CNN when the training set is insufficient. The new training sets consist of 100, 200, 300, 500, 700, 1000, 1500, 2000, 3000, 5000, 7000, 10000 training examples respectively. Then we train LeNet-5-like, CS-CNN and Mean-CNN with the $num = 8$ with these training sets. Fig. 11 has shown the simulation result. We can see from Fig. 11 that the classification accuracy of CS-CNN with the $num = 8$ is higher than the other two networks obviously. This result also demonstrates that our CS-CNN performs

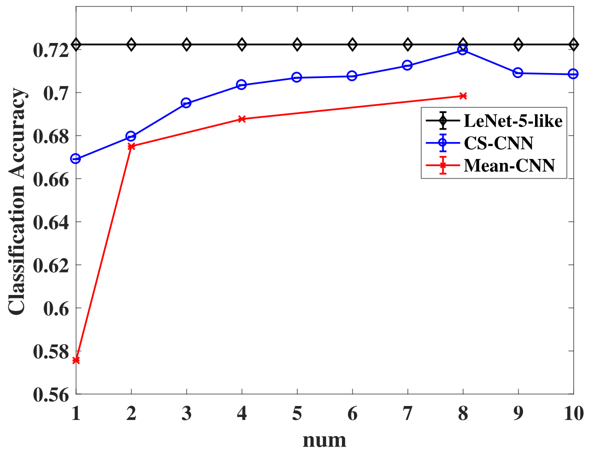


FIGURE 10. The classification accuracy of LeNet-5-like, CS-CNN and Mean-CNN.

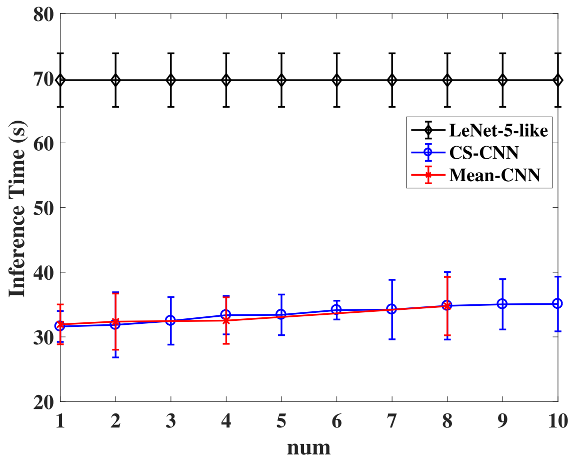


FIGURE 9. The inference time of LeNet-5-like, CS-CNN and Mean-CNN.

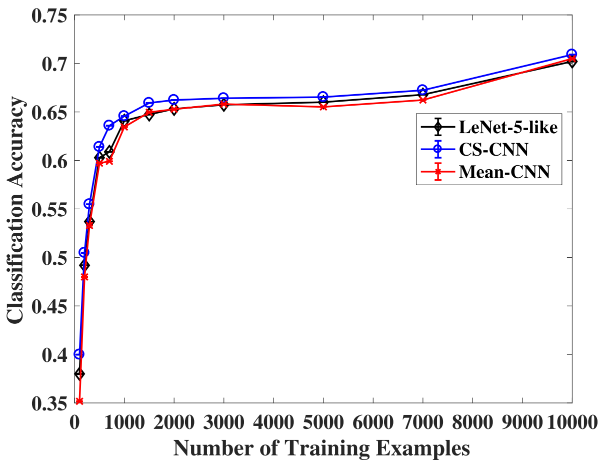


FIGURE 11. **The classification accuracy in small training dataset.**

best with respect to LeNet-5-like and Mean-CNN when the training set is insufficient.

V. CONCLUSION

In this paper, we propose a new deep learning framework, CS-CNN, for images classification task under resource-constrained conditions. We incorporate the theory of compressive sensing in the input layer of CNN models to reduce the high requirement on computation and amount of training data. According to our evaluation on the two popularly used dataset, i.e., MINST and CIFAR-10, we demonstrate that CS-CNN is able to accelerate the training and inference stages of CNN based image classification by an order of magnitude meanwhile it achieves better classification accuracy when the amount of training data is limited. Based on the advantages mentioned above, it's evidently that the proposed CS-CNN can achieve remarkable overall performance and is more appropriate and practical to be implanted to the resource constrained devices such as various IoT devices.

REFERENCES

[1] K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," *Neurocomputing*, Dec. 2017. doi: 10.1016/j.neucom.2017.04.083.

[2] M. Sajjad *et al.*, "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities," *Future Generat. Comput. Syst.*, Nov. 2017.

[3] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. H. G. Wang, and S. W. Baik, "Secure surveillance framework for IoT systems using probabilistic image encryption," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2018.2791944.

[4] K. Muhammad, M. Sajjad, and S. W. Baik, "Dual-level security based cyclic18 steganographic method and its application for secure transmission of keyframes during wireless capsule endoscopy," *J. Med. Syst.*, vol. 40, no. 5, p. 114, 2016.

[5] H. Li, P. Su, Z. Chi, and J. Wang, "Image retrieval and classification on deep convolutional SparkNet," in *Proc. IEEE Int. Conf. Signal Process., Commun. Comput.*, Hong Kong, Aug. 2016, pp. 1–6.

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[7] T. Ahonen, A. Hadid, and M. Pietikainen, "Face recognition with local binary patterns," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 469–481.

[8] N. Majumder *et al.*, "Deep learning-based document modeling for personality detection from text," *IEEE Intell. Syst.*, vol. 32, no. 2, pp. 74–79, Mar. 2017, doi: 10.1109/MIS.2017.23.

[9] P. Semberecki *et al.*, "Deep learning methods for subject text classification of articles," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, 2017, pp. 357–360.

[10] R. Sarikaya, G. E. Hinton, and A. Deoras, "Application of deep belief networks for natural language understanding," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 4, pp. 778–784, Apr. 2014, doi: 10.1109/TASLP.2014.2303296.

[11] B. Wu *et al.*, "An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 99, pp. 1289–1300, Sep. 2017, doi: 10.1109/JSTSP.2017.2756439.

[12] A. M. Badshah *et al.*, "Deep features-based speech emotion recognition for smart affective services," *Multimedia Tools Appl.*, pp. 1–19, Oct. 2017.

[13] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017, doi: 10.1109/ACCESS.2017.2778011.

[14] Y. Liu *et al.*, "Action2activity: Recognizing complex activities from sensor data," in *Proc. Int. Conf. Artif. Intell.*, 2015, pp. 1617–1623.

[15] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using ImageNet pretrained networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 1, pp. 105–109, Jan. 2016, doi: 10.1109/LGRS.2015.2499239.

[16] M. Chen, L. Zhang, and J. P. Allebach, "Learning deep features for image emotion classification," in *Proc. IEEE Int. Conf. Image Process.*, Quebec City, QC, Canada, Sep. 2015, pp. 4491–4495.

[17] A. S. Razavian *et al.*, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. Comput. Vis. Pattern Recognit. Workshops*, Columbus, OH, USA, 2014, pp. 512–519.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 1097–1105, 2012.

[20] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Dec. 2014, pp. 818–833.

[21] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. (2013). "OverFeat: Integrated recognition, localization and detection using convolutional networks." [Onlines]. Available: https://arxiv.org/abs/1312.6229

[22] H. Qin *et al.*, "Joint training of cascaded CNN for face detection," in *Proc. Comput. Vis. Pattern Recognit.*, 2016, pp. 3456–3465.

[23] D. Triantafyllidou and A. Tefas, "Face detection based on deep convolutional neural networks exploiting incremental facial part learning," in *Proc. Int. Conf. Pattern Recognit.*, 2017, pp. 3560–3565.

[24] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 1063–1119.

[25] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. Comput. Vis. Pattern Recognit.*, vol. 11. Jul. 2017, p. 1, doi: 10.1109/CVPR.2017.683.

[26] B. Chen and W. Deng, "Deep learning backend for single and multisession i-vector speaker recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 807–817, doi: 10.1109/CVPR.2017.428.

[27] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, 2014, pp. 580–587.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[31] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2016.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (Feb. 2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360

[34] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *Comput. Sci.*, vol. 14, no. 7, pp. 38–39, 2015.

[35] X. Zhang *et al.* (Jul. 2017). "ShuffleNet: An extremely efficient convolutional neural network for mobile devices." [Online]. Available: https://arxiv.org/abs/1707.01083

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *Fiber*, vol. 56, no. 4, pp. 3–7, 2016.

[37] R. Baraniuk, "Compressive sensing," in *Information Sciences and Systems*. Princeton, NJ, USA: Princeton Univ. Press, 2008.

[38] N. D. Lane *et al.*, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2016, pp. 1–12.

[39] S. Bhattacharya and L. D. Lane, "Sparsification and separation of deep learning layers for constrained resource inference on wearables," in *Proc. 14th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Nov. 2016, pp. 176–189.

[40] M. F. Duarte *et al.*, "Single-pixel imaging via compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 83–91, Mar. 2008.

[41] X. Shu and N. Ahuja, "Imaging via three-dimensional compressive sampling (3DCS)," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, vol. 23. no. 5, pp. 439–446.

[42] L. Jacques, P. Vandergheynst, A. Bibet, V. Majidzadeh, A. Schmid, and Y. Leblebici, "CMOS compressed imaging by random convolution," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 1113–1116.

[43] C. Luo *et al.*, "Compressive data gathering for large-scale wireless sensor networks," in *Proc. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 145–156.

[44] Y. Shen *et al.*, "Efficient background subtraction for real-time tracking in embedded camera networks," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2012, pp. 295–308.

[45] Y. Gu and N. A. Goodman, "Information-theoretic compressive sensing kernel optimization and Bayesian Cramér–Rao bound for time delay estimation," *IEEE Trans. Signal Process.*, vol. 65, no. 17, pp. 4525–4537, Sep. 2017.

[46] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[47] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.

[48] A. C. Gurbuz, J. H. McClellan, and W. R. Scott, "A compressive sensing data acquisition and imaging method for stepped frequency GPRs," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2640–2650, Jul. 2009.

[49] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2082–2102, Apr. 2013.

[50] R. Robucci, J. D. Gray, L. K. Chiu, J. Romberg, and P. Hasler, "Compressive sensing on a CMOS separable-transform image sensor," *Proc. IEEE*, vol. 98, no. 6, pp. 1089–1101, Jun. 2010.

[51] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[52] J. C. Wang *et al.*, "Compressive sensing-based speech enhancement," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 11, pp. 2122–2131, Nov. 2016.

[53] M. Sadeghi, M. Babaie-Zadeh, and C. Jutten, "Learning overcomplete dictionaries based on atom-by-atom updating," *IEEE Trans. Signal Process.*, vol. 62, no. 4, pp. 883–891, Feb. 2014.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[55] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., Jul. 2017.

[56] J. Feng *et al.*, "Injurious or noninjurious defect identification from MFL images in pipeline inspection using convolutional neural network," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 7, pp. 1883–1892, Mar. 2017.

[57] Y. Shen *et al.*, "GaitLock: Protect virtual and augmented reality headsets using gait," *IEEE Trans. Depend. Sec. Comput.*, to be published.

[58] Y. Shen, C. Luo, D. Yin, H. Wen, R. Daniela, and W. Hu, "Privacy-preserving sparse representation classification in cloud-enabled mobile applications," *Comput. Netw.*, vol. 133, pp. 59–72, Mar. 2018.

[59] W. Xu, Y. Shen, N. Bergmann, and W. Hu, "Sensor-assisted multi-view face recognition system on smart glass," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 197–210, Jan. 2018.

[60] Y. Shen, M. Yang, B. Wei, C. Chou, and W. Hu, "Learn to recognise: Exploring priors of sparse face recognition on smartphones," *IEEE Trans. Mobile Comput.*, vol. 16, no. 6, pp. 1705–1717, Jun. 2017.

YIRAN SHEN (M'12) received the Ph.D. degree in computer science and engineering from the University of New South Wales, Australia. He was a SMART Scholar with Singapore-MIT Alliance for Research and Technology. He is currently an Associate Professor with the College of Computer Science and Technology, Harbin Engineering University. He publishes regularly at top-tier conferences and journals. His current research interests include wearable/ mobile computing, wireless sensor networks, and the applications of compressive sensing.

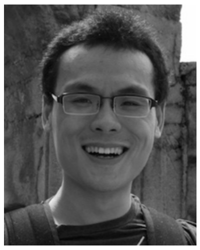
TAO HAN received the Bachelor's degree from the Harbin Institute of Technology, Weihai. She is currently a Postgraduate Research Student with the College of Computer Science and Technology, Harbin Engineering University. Her research interests include privacy protection in IoT and deep learning on embedded systems.

QING YANG received the Bachelor's degree in software engineering from Harbin Engineering University, where she is currently pursuing the Ph.D. degree with the College of Computer Science and Technology. She is also a visiting Ph.D. student with the Commonwealth Scientific and Industrial Organization, Australia. Her main research interests include wearable/mobile computing and privacy preserving in mobile sensor networks.

XU YANG received the Master's degree from the University of New South Wales, Australia. She is currently a Lecturer with the Inner Mongolia University of Technology. Her major research interests include machine learning and data analytics.

YONG WANG received the Master's degree in computer science from Harbin Engineering University (HEU). He is currently a Lecturer with the College of Computer Science and Technology, HEU. His main research interests include social computing, wireless sensor networks, and database systems.

FENG LI received the B.S. degree from Shandong Normal University, China, in 2007, the M.S. degree from Shandong University, China, in 2010, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2015, all in computer science. From 2014 to 2015, he was a Research Fellow with the National University of Singapore, Singapore. He is currently an Assistant Professor with the School of Computer Science and Technology, Shandong University. His research interests include applied optimization, distributed systems and algorithms, wireless networking, and mobile/pervasive computing.



HONGKAI WEN received the D.Phil. degree from the University of Oxford. Then, he became a Post-Doctoral Researcher in a joint project between the Oxford Computer Science and Robotics Institute. He is currently an Assistant Professor with the Department of Computer Science, University of Warwick. His research interests include cyber-physical systems, which use networked smart devices to sense and interact with the physical world.