CUNet: A Compact Unsupervised Network for Image Classification

Le Dong, *Member, IEEE,* Ling He, Mengdie Mao, Gaipeng Kong, Xi Wu, Qianni Zhang, Xiaochun Cao, and Ebroul Izquierdo

***Abstract*—In this paper, we propose a compact network called CUNet (compact unsupervised network) to address the image classification challenge. Contrasting the usual learning approach of convolutional neural networks, learning is achieved by the simple K-means on diverse image patches. This approach performs well even with scarcely labelled training images, greatly reducing the computational cost, while maintaining a high discriminative power. Furthermore, we propose a new weighted pooling method in which different weighting values of adjacent neurons are considered. This strategy leads to improved classification since the network becomes more robust against small image distortions. In the output layer, CUNet integrates feature maps obtained in the last hidden layer, and straightforwardly computes histograms in non-overlapped blocks. To reduce feature redundancy, we also implement the max-pooling operation on adjacent blocks to select the most competitive features. Comprehensive experiments on well-established databases are conducted to validate the classification performances of the introduced CUNet approach.**

***Index Terms*—Unsupervised Learning, Convolutional Network, Image Classification, K-means.**

I. Introduction

IMAGE classification is a challenging task in computer vision, especially when the image databases and intra-class variability are large and continue increasing. Numerous efforts have been made over the last decades to address this difficult task. Among others, the bag-of-features (BoF) model has shown reasonable performance. It works by extracting local features from the images, e.g. SIFT, vector quantizing them and then representing images as histograms of such visual words. Clearly and unfortunately, in a BoF representation the spatial information is neglected. An extension of BoF, Spatial Pyramid Matching (SPM), takes into account of the spatial information of images, improving the classification performance on relatively small benchmarks like Caltech101 and Caltech256. However, such model designs fail to demonstrate competitive performance on mid-scale datasets such as STL-10 or large-scale datasets such as CIFAR-10. While parallel processing based on distributed resources [1] seems to have overcome the bottleneck introduced by the increasing scales of image datasets, the fundamental solution to this problem should still be derived from the processing algorithms. The search for such solutions has attracted considerable interest. A major stream of research relies on the use of mid-level features [2, 3] and the feature learning approach in general [4, 5, 6].

L. Dong, L. He, M. Mao and G. Kong are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), 2006 Xiyuan Avenue, Gaoxin West Zone, Chengdu, Sichuan, 611731, China.
E-mail: ledong@uestc.edu.cn
X. Wu is with the School of Computer Science, Chengdu University of Information Technology.
X. Cao is with the Institute of Information Engineering, CAS.
Q. Zhang and E. Izquierdo are with the School of Electronic Engineering and Computer Science, Queen Mary, University of London.

In recent years, deep convolutional neural networks (C-NNs) have demonstrated outstanding capabilities for large-scale image classification [7]. These results have encouraged extensive studies towards better CNN architectures [8, 9, 10]. Trained with sufficient and diversified datasets, the improved CNNs successfully obtain exceptional performance on visual recognition tasks. The success of CNNs is mainly attributed to their ability in learning rich mid-level image representations instead of hand-designed low-level features. Typically, the convolutional neural networks adopt a three-stage formulation, including the filter bank convolution, neuron activation, and pooling stages. Among these stages, filter bank convolution plays a central role. To learn an effective filter bank at each convolution stage, a variety of methods have been proposed, such as the restricted Boltzmann machines (RBM) [11, 12], regularized auto-encoders and their variations [11]. In general, previous CNNs optimize the filter bank by utilizing the stochastic gradient descent (SGD) method on a large number of labelled images. Such approaches are largely dependent on the expertise of parameter initiation and fine tuning. In addition, such filter learning procedures are computationally very intensive. The emergence of GPU computing [13] and dedicated fast deep learning frameworks, like Caffe, to some extent facilitate the learning procedures in CNNs. However, the fundamental problem of extremely high computational cost in such algorithms still remains. Furthermore, traditional CNNs take a supervised approach and rely on large-scale training sets to produce good performance. Nowadays, the available image data grows exponentially, making the associated image labels more and more scarce. the lack of labelled training samples becomes another major problem that hampers the application of CNNs in the image classification domain.

To remove the uncertainty in CNNs' filter bank learning procedure, researchers proposed another mathematically justified model, namely, the wavelet scattering networks (ScatNet) [14, 15]. ScatNet is similar with CNNs except for the design of its filter bank. The filter bank in ScatNet is a set of predefined wavelet operators. In this way, the weights learning procedure in traditional CNNs is avoided. Despite its simple design of the wavelet filter bank, the outcome reported in [14] and [15]

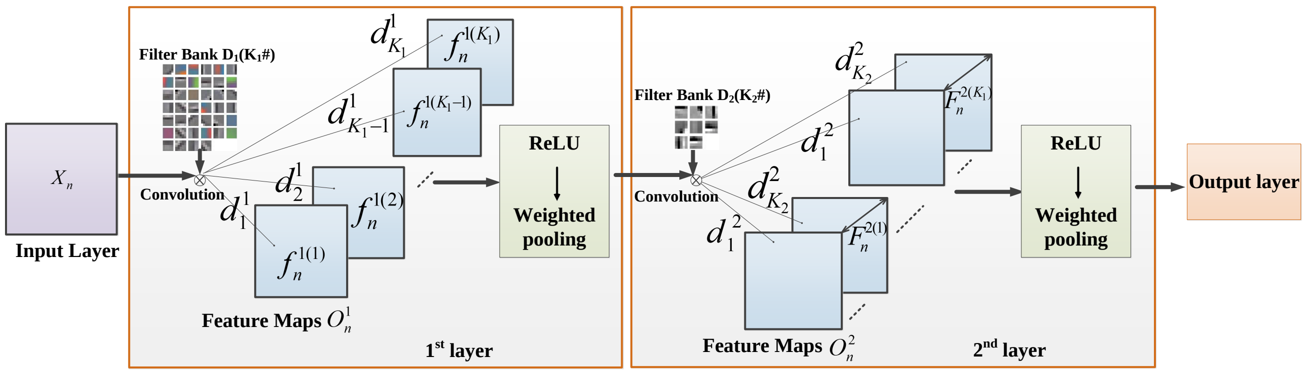

Fig. 1: The CUNet structure.

have verified that ScatNet can achieve superior performance on handwritten digit and texture recognition, based on a similar multistage architecture as in CNN. However, such predefined filter bank fails to capture the discriminative information in datasets of other image types. They are not generalisable and cannot handle different image understanding tasks in a broader domain.

Motivated to address the above problems in the literature, in this paper, we propose a compact unsupervised network (CUNet) for image classification. Inspired by ScatNet, this network aim to employ a neat design in its filter bank while keeping it generalisable for potential applications. The filter bank is constructed by applying the classical K-means on a set of randomly extracted image patches. Here, image labels are not necessary because un-supervised learning of the filters can be achieved through K-means. After the convolution, the Rectified Linear Units (ReLUs) are maintained to activate neurons, followed by a proposed weighted pooling strategy. Subsequent hidden layers are constructed in the same way, except that the filter banks are learned from previous feature map patches. In the output layer, each neuron is binary-mapped, and each group of feature maps are synthesized into a coarse representation of the input image. Then, histograms are computed in each non-overlapped block, followed by the max-pooling operation on adjacent blocks to reduce feature redundancy and select the most competitive features.

The contribution of CUNet can be summarized in three aspects:

(1) The compact and un-supervised manner of filter bank learning avoids the initialization and fine tuning of millions of parameters. This approach significantly reduces the computation load, and more importantly, overcomes the problem of lacking image labels for training. Thus, CUNet effectively avoids falling into a local optimum which traditional CNN usually suffers from;

(2) The proposed weighted pooling jointly considers the effects of all the activations in the pooling region. It helps improve the network robustness to small image distortions;

(3) The histogram formation is achieved in a straight-forward manner. We choose to compute histograms in multiple blocks in order to help obtaining the spatial information to a certain extent. The max-pooling trick further improve the feature competitiveness.

The rest of the paper is organized as follows: Section 2 analyses the related works; Section 3 gives the formulation details of CUNet; Section 4 provides comprehensive experimental results to validate the performance of CUNet; and finally, Section 5 concludes the paper with directions for future work.

II. Related Work

Convolutional networks have recently demonstrated impressive progress in a variety of image classification and recognition tasks [13, 23]. The promising perspective of CNNs encourages researchers to make further attempts for better performance. Multiple layers of unpooled convolution have been utilized lately with considerable success [7], despite that such architectures must be carefully designed and sized using good intuition along with extensive trial-and-error experiments on a validation set. The work in [19] proposes to transfer image representations learned with CNNs on large datasets to other visual recognition tasks with limited training data. Some success has been achieved when reusing the ImageNet representation to compute mid-level image representation for the PASCALVOC dataset, at the cost of intensive and challenging training on ImageNet. Besides, the representation learned from larger datasets may incur overfitting issues when the knowledge is transferred to smaller datasets. In [8], a new activation function called maxout is proposed to avoid pitfalls such as missing to use many filters of a model, so that the training of deeper networks becomes possible. Compared with conventional convolutional layers which perform linear separation, the maxout network is more potent as it can separate concepts that lie within convex sets. However, maxout network imposes the prior that instances of a latent concept lie within a convex set in the input space, which does not necessarily hold.

In [9], a NIN network is proposed, which is composed of mlpconv layers. It uses multi-layer perceptrons to convolve the input and a global average pooling layer as a replacement for the fully connected layers in conventional CNN. While mlpconv layers model the local patches better and the global average pooling prevents overfitting globally, NIN still faces the difficulty in training and fine tuning millions of parameters.

The training of recurrent neural networks usually involves the vanishing and exploding gradient problems. The work in [16] proposes a new multi-task feature selection algorithm, by utilizing the common knowledge of multiple tasks as supplementary information to facilitate decision making. The work in [17] demonstrates a new clustering algorithm which employs both manifold information and discriminant information for data clustering. In [18], a semi-supervised learning algorithm is reported for image representation inference. The works in [21, 23] show that appropriate active learning method would improve the performance of image classification. In [20], a gradient norm clipping strategy is proposed to deal with the exploding gradients problem, and used a regularization term that prevents the error signal from vanishing as it travels back in time to relieve the vanishing gradient restriction. Though some improvements on the gradient training have been achieved, the work in [20] still fails to simplify the inherent complexity of current neural networks.

Overall, these approaches all more-or-less suffer from the intensive computation load and the lack of labelled data for training. With a clear goal to address these issues, we propose the filter bank learning procedure that is designed to work in a compact and unsupervised manner. The neat design ensures its high efficiency. More importantly, by abandoning the initialization and fine tuning of millions of parameters, and the system is no longer restricted by the limitation of scarce annotation on images. The proposed CUNet does not use any image transformations or other regularization such as dropout or maxout [8], but focuses on four main steps: preprocessing image patches, learning the K-means filter bank, computing histograms and selecting the most competitive histogram bins. This concise design reduces the computation cost, and at the same time guarantees superior performance compared to existing delicate network designs.

III. Compact Unsupervised Network

In this section, we present the detailed formulation of our proposed CUNet. The two-layer CUNet structure is illustrated in Fig.1, and the output layer is illustrated in Fig.2. In the next subsections, we will elaborate each component of the block diagram.

A. The pre-processing of the input layer

Suppose we are given N input training images $\{X_n\}_{n=1}^N$ of size $W \times H \times d$, where $d = 1$ for gray images and $d = 3$ for RGB ones. CUNet begins by extracting random patches from the training images $\{X_n\}_{n=1}^N$. Each $w \times h$ patch can be denoted as a vector in $\mathscr{R}^M$ of pixel intensity values, with $M = w \times h \times d$. Then, we can construct a dataset containing T randomly extracted patches, $P = \{p_1, \cdots, p_t, \cdots, p_T\}$, where $p_t \in \mathscr{R}^M$. Given this patch dataset, we apply some necessary pre-processing operations on P to obtain better configuration.

It is a common practice for vision tasks to perform some simple normalization steps before attempting to generate features from the input data. In this work, each patch p_t is normalized by subtracting the mean and dividing by the standard deviation of its elements. After normalizing each input vector, we apply the whitening operation over the whole dataset P. In [29], the superiority of whitened images over non-whitened has been discussed. Then, we obtain the pre-processed input dataset $\bar{P} = \{\bar{p_1}, \cdots, \bar{p_2}, \cdots, \bar{p_T}\}$. Assuming that the number of filters in the first layer is K_1, we run K-means on $\bar{P}$ to acquire the filter bank, denoted as $D_1 = \{d_1, \cdots, d_{k_1}, \cdots, d_{K_1}\} \in \mathscr{R}^{M \times K_1}$ where each centroid d_{k_1} will act as a convolution filter in the subsequent convolution stage.

B. The formulation of the hidden layer

We maintain the typical processing stages of traditional CNNs, *i.e.*, filter convolution, pooling and neuron activation. In this section, we will describe in details each stage with its special design in CUNet.

Filter convolution: Given the first layer's convolution filter bank $D_1 = \{d_1, \cdots, d_{K_1}\}$, we convolve each training image X_n with the K_1 filters:

$$O_n^1 = X_n \otimes D_1, n = 1, \cdots, N, \tag{1}$$

where $O_n^1 = \left\{f_n^{1(1)}, \cdots, f_n^{1(k_1)}, \cdots, f_n^{1(K_1)}\right\}$ is the first layer's feature map set of X_n, $f_n^{1(k_1)}$ is the feature map of X_n convolved by the filter d_{k_1}, and $\otimes$ denotes the $2D$ convolution operation.

Nonlinear activation: Then, the neurons in the feature maps need to be activated through a pre-defined activation function. The Tangent function $f(x) = tanh(x)$ and Sigmoid function $f(x) = (1 + e^{-x})^{-1}$ are commonly used in previous networks and have been proved to be effective. However, considering the training time, these saturating nonlinearities are much slower than the non-saturating nonlinearity $f(x) = max(0, x)$. Following [24], we refer to the neurons activated by this nonlinearity as Rectified Linear Units (ReLUs). In [13], it has been verified that deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units. Therefore, CUNet adopts ReLUs to accomplish subsequent process. In fact, we have tried the Tangent function and Sigmoid function in CUNet and found that they are not competitive with the ReLUs.

Weighted pooling: To ensure the network robustness against small distortions, we set a pooling layer after the activation layer, as most of the traditional CNN architectures do. Conventional pooling usually uses either max pooling or average pooling. Max pooling always captures the largest response values, but may lose the useful information of the smaller ones. As for average pooling, it aggregates local statistics information by preventing large response values overwhelming and small values being ignored. However, since average pooling treats each neuron equally, the usefulness of each neuron's response is considered the same. The work in [22] proposes stochastic pooling, which replace the conventional deterministic pooling operations with a stochastic procedure. It randomly picks the activation within each pooling region according to a multinomial distribution defined by the activities within the pooling region. Obviously, the choice of the multinomial distribution has dominating effect on the pooling performance. Inspired by these previous pooling strategies, we propose a new pooling method called weighted pooling. It considers each neuron's response as well as the usefulness of its response. That is, each neuron in the pooling region owns a weighting factor representing the usefulness of its response. Suppose that the pooling window is of size $p_w \times p_h$, the response value of each neuron is $a_{i,j}$ with $i = 1, \cdots, w; j = 1, \cdots, h$. Then, the pooling results of the $p_w \times p_h$ window can be calculated according to Eq.(2):

$$P_{result} = w_{i,j} * a_{i,j} \tag{2}$$

where $w_{i,j}$ is the weight of $a_{i,j}$. In this paper, we compute each neuron's value in proportion to the pooling region as its weight, i.e., $w_{i,j} = \frac{a_{i,j}}{\sum_i \sum_j a_{i,j}}$. The proposed weighted pooling will capture different proportions of local information of each neuron in the original feature map, thus leading to a better local representation. To reveal the effect of the proposed weighted pooling method, we conducted experiments in Section 4 to compare the network performance under different pooling strategies. Conventionally, a pooling operation summarizes the non-overlapping neighbourhoods containing adjacent units. Such an approach reduces the computation complexity but leads to coarse pooling results. To be acquire a more precise pooling, CUNet applies an overlapping sliding window with a stride s for fine grained results.

The three main stages: convolution, non-linear activation and weighted pooling, form a complete layer of CUNet. Note that these three steps maintain a feature map that is of the same size with the original input image. The convolution and pooling operations both pad the images (or feature maps) with zeros. We tested the model with a fix-sized feature map and observed that it outperformed traditional models with feature maps whose sizes change. The second layer has a similar formulation with the first layer, except that the filter bank D_2 is obtained by running K-means on the patches randomly extracted from the first layer's output. Although stacking multiple layers together can lead to higher level features, as reported in some works like [9], we find that more layers than two only bring subtle performance improvement. Thus, the proposed CUNet adopts a two-layered architecture, while a deeper model can be implemented in the same approach, where necessary.

C. The design of the output layer

The detailed design of the output layer is illustrated in Fig.2. In the second layer, each of the K_1 feature maps $f_n^{1(k_1)}$ has K_2 outputs $F_n^{2(k_1)} = \left\{ f_n^{2(1)}, f_n^{2(2)}, \cdots, f_n^{2(K_2)} \right\}$. First, each set of the K_2 feature maps are binary-mapped, *i.e.* each unit value is set as 1 if it is positive and 0 if non-positive. The resulting feature maps are composed of ones and zeros, and thus are referred to as B-maps. Such crude mapping inevitably leads to loss of some useful feature information. To integrate the complementary feature information into the B-maps, we take the inspiration from [25] and transform the K_2 B-maps in $F_n^{2(k_1)}$ into an integer-valued image by multiplying each feature map with a coefficient λ_i:

$$I = \sum_{i=1}^{K_2} \lambda_i f_n^{2(k_2)}, \tag{3}$$

where $\lambda_i = 2^{i-1}$, $f_n^{2(k_2)}$ is the k_2-th B-map in $F_n^{2(k_1)}$. The ordering and weighting of the K_2 B-maps do not affect the network performance.

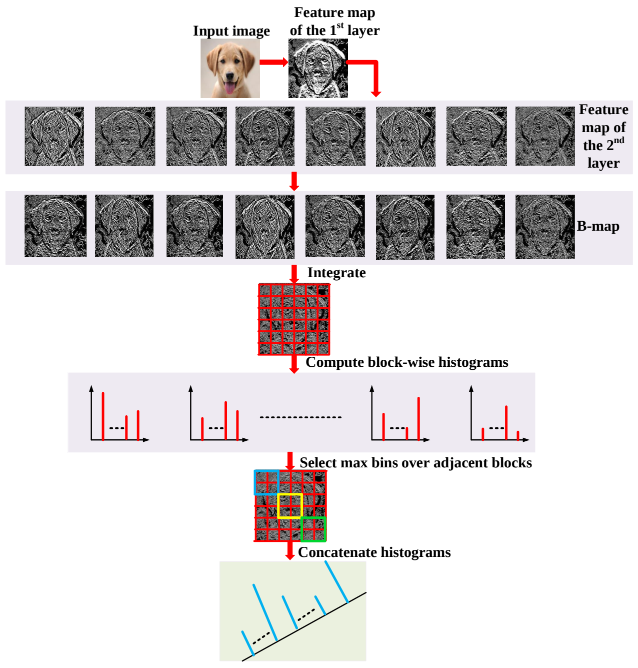
Input image
Feature map of the 1st layer
Feature map of the 2nd layer
B-map
Integrate
Compute block-wise histograms
Select max bins over adjacent blocks
Concatenate histograms

Fig. 2: Details of CUNet output layer.

For each of the K_1 feature maps in O_n^1, we can obtain its corresponding image I_{k_1} with $k_1 = 1, \cdots, K_1$. Next, the histogram of each I_{k_1} can be computed and used as the final image representation. To ensure the model's robustness against image geometric variances, the histograms are computed in a sliding-window manner. The work in [25] has demonstrated the effectiveness of such a histogram extraction approach in matching images of highly variable scenes. However, we argue that this histogram computing approach may cause feature redundancy in high dimensionality. In order to avoid this issue, we execute max-pooling operation on histogram bins in adjacent blocks. In particular, for the adjacent $w \times w$ blocks, we select the maximum bins in each block, leading to one single histogram. Such max-pooling operation helps obtain the most competitive image features, avoids feature redundancy, and keeps the feature dimension in a reasonable range. Finally, we concatenate the histograms gained from each group of $w \times w$ blocks as the image feature, followed by a classification of images based on a linear SVM.

IV. Experimental Evaluation

We evaluate the performance of CUNet on four benchmarking datasets: STL-10, Caltech101, CIFAR-10 and MNIST. The networks used for these four datasets all consist of two stacked layers, followed by a linear SVM classifier. More particular experimental settings are presented in the subsequent sections. For the comparison purpose, we directly quote results from the literature since it is often not possible to reproduce their results, largely due to subtle engineering details.

A. The Classification Performance

1) CIFAR-10: The CIFAR-10 dataset is composed of 10 classes of natural images, among which 50,000 are used for training and 10,000 are for testing. The images are of a uniform size 32×32. Images of each class vary largely in object position, size, colors and textures. Besides, the background of each image shows significant differences.

In particular, we learn $K_1 = 40$ filters of size 5×5 in the first layer and $K_2 = 8$ filters of size 5×5 in the second layer. The size of weighted pooling in both the two layers are set as 2×2, and the pooling windows overlap with one pixel stride. The blocks for histogram computing are all of size 4×4, non-overlapped. After acquiring the block-wise histograms, we select the maximum bins over the adjacent $2 \times 2 = 4$ blocks to build one single histogram.

TABLE 1 presents the classification accuracies of different methods on CIFAR-10. We observe that CUNet, with weighted pooling, achieves desirable performance among these methods. Besides, the results show that the pooling strategy influences the final classification performance when all the other settings remain the same. Among the three pooling strategies, namely, our proposed weighted pooling, the prevalent max and average pooling, weighted pooling shows the best performance - about 0.38% higher than max pooling and 0.85% higher than average pooling. Note that the same filter banks used in weighted pooling are employed in max and average pooling. This is to avoid the potential influence from filters randomly learned by K-means. This setting is similarly applied in experiments on the STL-10, Caltech101 and MNIST datasets for a fair evaluation.

TABLE I: Comparison of accuracy(%) by different methods on CIFAR-10 without data augmentation.

Methods	Accuracy(%)
CUNet + Weighted pooling	80.31
CUNet + Max pooling	79.93
CUNet + Average pooling	79.46
Tiled CNN [26]	73.10
Improved LCC [27]	74.50
KDES-A [28]	76.00
K-means (Triangle,4000features) [29]	79.60
Cuda-convnet2 [30]	82.00
CKN-CO [10]	82.18
Discriminative SPN [31]	83.96
TIOMP-1/T (combined, K= 4,000) [32]	82.20
2x PDL (1600 codes) [33]	78.71

2) STL-10: The STL-10 dataset consists of colour images of 96×96 pixel size, belonging to 10 different classes. This dataset similarly organised with CIFAR-10 while providing fewer training samples (500 per class) and test samples (800 per class). This set-up forces algorithms to rely on acquired prior knowledge of image statistics. We down-sampled the STL-10 images into 32×32 pixels for a simpler configuration.

Experimental settings on STL-10 are similar to that of CIFAR-10 experiments, except that $K_1 = 30$ filters are employed. TABLE 2 gives the comparison of results from different methods on STL-10. We observe that CUNet, with weighted pooling, provides more desirable performance compared to the previous works. With the same settings elsewhere, weighted pooling in CUNet helps increase the classification accuracy by 0.6% compared to max pooling and by 0.4% compared to average pooling.

TABLE II: Comparison of accuracy(%) by different methods on STL-10 without data augmentation.

Methods	Accuracy(%)
CUNet + Weighted pooling	63.00
CUNet + Max pooling	62.40
CUNet + Average pooling	62.60
2x PDL (1600 codes) [33]	58.28
CKN-CO [10]	62.32
EPLS [34]	61.00
Discriminative SPN [31]	62.30
sparse TIRBM (combined) [32]	58.70

To further demonstrate the performance of CUNet, some sample images are listed from each class in Fig.3, and the classification accuracy of each class is given next to the corresponding image classes. From the results, it can be observed that higher accuracy can commonly be achieved in the classification of simpler classes, such as airplane(81.38%), ship(81.00%) and car(80.13%). These classes of objects usually entails relatively distinctive and coherent visual appearances. Besides, they are rigid objects and thus rarely incur confusing variations such as activity variance. In contrary, the classes of living objects, such as monkey(53.50%), cat(43.50%) and dog(31.00%), commonly lead to lower accuracies. From the sample images, it can be observed that animal classes often includes various sub-categories with different appearances, and the animals are in different poses, with a high probability of occlusion. All these factors increase the difficulty in classifying these classes.

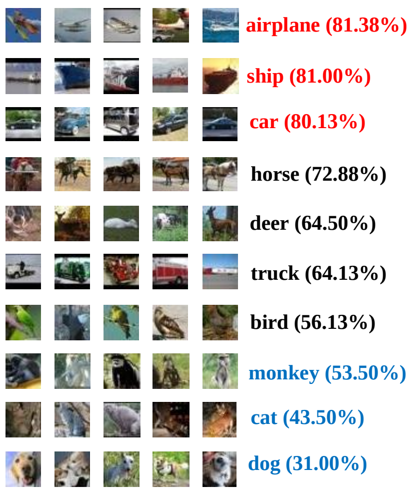

Fig. 3: Example images of each class and their classification accuracies in STL-10.

3) Caltech101: Caltech101 dataset contains 101 classes, including animals, vehicles, flowers, *etc.* with significant visual variation of the objects and the background. The number of images per category varies from 31 to 800. For experimental convenience, we convert all the images into grey scale, and resize them into 32×32 without keeping the aspect ratio.

Following the traditional settings, we randomly select 15 and 30 training images per class, including the background classes. TABLE 3 presents the classification results on Caltech101. In the cases of both 15 and 30 training images per class, we train $K_1 = 30$ filters. Other settings are the same with CIFAR-10.

From TABLE 3, we observe that the proposed CUNet with weighted pooling achieves the best performance among the state-of-the-arts methods based on raw pixels. It is worth noting that in our experiment setting, all images are resized into 32×32 pixels without keeping the aspect ratio, while in previous works image aspect ratios are usually kept. Still, the proposed CUNet shows its competitive capability of classification on Caltech101. Similar to the experiments on CIFAR-10 and STTL-10, the propose weighted pooling successfully outperforms max pooling and average pooling.

TABLE III: Comparison of accuracies(%) by different methods on Caltech101

Training size	15	30
CUNet + Weighted pooling	58.62	66.72
CUNet + Max pooling	58.00	66.34
CUNet + Average pooling	58.14	66.48
CDBN [38]	57.70	65.40
ConvNet [39]	57.60	66.30
DeconvNet [40]	58.60	66.90
Chen *et al.* [36]	58.20	65.80
Zou *et al.* [37]	-	66.50

Fig.4 shows some classification results of Caltech101. Here, some classes that are semantically similar are considered. It is interesting to observe that the classification accuracies of each pair of similar classes largely differ. For example, the classification accuracy of the class **Faces** is 76.90%, which is about 19.77% lower than the class **Faces easy**(96.67%). From the example images, it can be seen that the faces in the **Faces easy** class are placed in the center of the images and little background is included. In comparison, the faces in class **Faces** class are arbitrarily positioned in the images, and all the images present a complex background. Therefore, the **Faces** class is more difficult to classify than the **Faces easy** class. Besides, when the images are uniformly resized, the **Faces** are often get distorted, which makes the classification of this class more difficult. Similarly, the classification results of the two class **Chair** and **Windsor chair** show great difference. The accuracy of the class **Chair** is 23.40%, which is 69.28% lower than that of the **Windsor chair** class (92.68%). As shown in the example images, the objects in **Chair** demonstrate significant visual variance and the background is relatively complex. In comparison, the intra-class variance of **Windsor chair** is subtle, and the background is much simpler than that of **Chair**. Thus, it is not a surprise that **Windsor chair** is much better classified than **Chair**. Similar observations can be made for the listed classes **Cougar body** (50.82%) and **Cougar face** (47.54%), **Crocodile head** (16.67%) and **Crocodile** (2.86%). From the above analysis, CUNet has superior classification performance in the classes with simple background and small little intra-class variance. However, we also admit that CUNet shows less competitiveness for the classes that have more complex background and greater intra-class variance.

4) MNIST: The basic MNIST dataset consists of 28×28 grey-scale images of handwritten digits from 0 to 9. It has 10,000 training, 2,000 validation, and 10,000 testing samples. For the same of experimental convenience in the baseline, we resize the MNIST images into 32×32 pixels, and keep all other settings the same with the aforementioned three datasets, except that the number of filters is $K_1 = 5$.

TABLE 4 presents the classification error rates on basic MNIST from different methods. Again, the proposed weighted pooling outperforms the average and max pooling. Since MNIST is a relatively simple dataset, all methods perform well with very small differences. The subtle difference in their performance is not statistically significant and thus is not analysed here.

TABLE IV: Comparison of error rates(%) obtained by different methods on MNIST without data augmentation.

Methods	Error rate(%)
CUNet + Weighted pooling	1.80
CUNet + Max pooling	1.86
CUNet + Average pooling	1.90
CAE-2 [35]	2.48
ScatNet-2 [14]	1.27

B. Impact of the number of filters

In this section, we report further experiments to validate the impact of filter number on the performance of CUNet. We fix the experimental settings as aforementioned in Section 4.1 (*i.e.*, the settings that achieved the best performance on each dataset, while adjusting the number of filters in the first layer. In particular, since CIFAR-10 is a relatively complicated dataset, we vary K_1 from 20 to 40. For the mid-scale datasets STL-10 and Caltech101, we vary K_1 from 10 to 30. For the simpler dataset MNIST, we vary K_1 from 5 to 15.

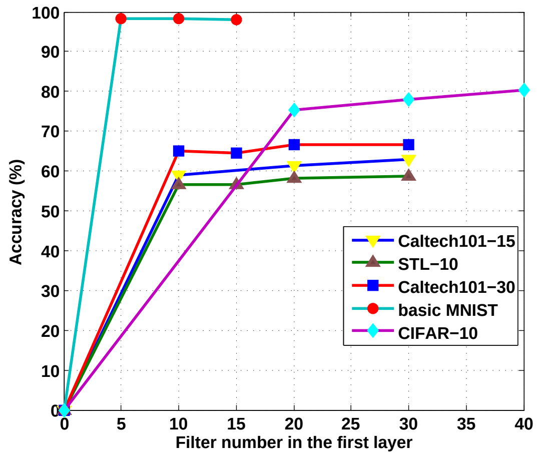

Fig. 5: Impact of the filter number on classification accuracy.

Fig.5 illustrates the impact of the number of filters on CUNet's classification performance. As shown in Fig.5, the classification accuracy generally improves when the number of filters increases, despite of different datasets. However, the improvements does not always last. When the number of filters reaches a saturated value, the improvement becomes subtle. This is because, when the number of filters is larger than a saturated value, some filters will be duplicated, introducing no additional contribution or even bringing adverse effects. Hence, the number of filters plays an important role in CUNet.

C. Impact of the block size

Fig.6 illustrates the impact of the block size on CUNet performance. Here, the block size refers to the width and height of the windows for histogram extraction. For each dataset, we set the block size as 4×4, 8×8 and 16×16, and fix the other settings as reported in Section 4.1. From Fig.6, it can be observed that the classification performance generally drops when the block size increases. Common for all the datasets, the highest classification accuracy is achieved when the block size is 4×4. When the block size increases to 8×8, the classification performance slightly declines, and the declining continues when the block size further increases to 16×16. However, it is worth mentioning that although the classification accuracy goes down along with the increase of block size, feature dimension also decreases, leading to a reduction of computation load. In particular, when the block size is 4×4, the feature dimension is $K_1 \times 2^{K_2} \times 16$. When the block size is set as 8×8, the feature dimension is $K_1 \times 2^{K_2} \times 4$, 4 times smaller than that of 4×4 block size. When the block size is 16×16, the feature dimension is $K_1 \times 2^{K_2} \times 1$, which is 1/16 of the feature dimension of block size 4×4.

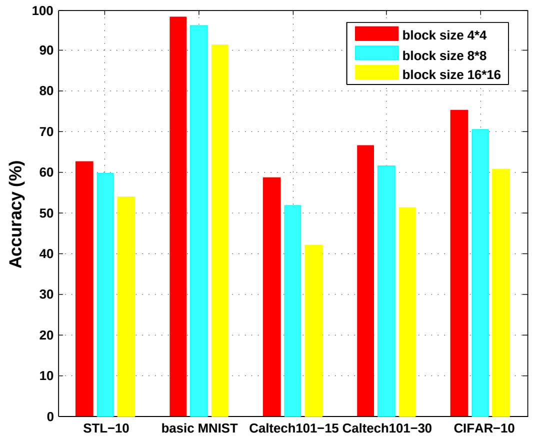

Fig. 6: Impact of the block size on classification accuracy.

Based on the above analysis, the influence of block size on CUNet is two-folded. On the one hand, the increase of block size results in a decline classification accuracy. On the other hand, feature dimension desirably decreases along with the increase of the block size. Hence, the choice of the block size is largely dependent on the specific requirements. If accuracy is dominantly pursued, then a smaller block size should be chosen. On the contrary, if the experimental devices struggles to meet the dataset scale, choosing a larger block size may help the experiments perform smoothly.

D. Discussion

The aforementioned experiments have successfully validated the effectiveness of CUNet from different aspects. Four different datasets, namely, CIFAR-10, STL-10, Caltech101, MNIST, are employed to test the performance of CUNet on different image classification tasks. The classification performance of CUNet are quantitatively compared with some the state-of-the-art methods. In particular, the sample classification output of some classes are visually presented to demonstrate the desired performance of CUNet. From the results, it has been shown that CUNet is superior in classifying static objects with little inner-class variance, *e.g.* , airplane, ship and car. In comparison, CUNet shows less advantage in classifying dynamic objects with high intra-class variance, such as dog, cat or monkey. Secondly, we test the effect of some key settings of CUNet on classification performance: 1) it has been proven that, despite of different datasets, more filters will certainly help improve the classification performance, but such increase becomes small when the number of filters reaches a saturated value; 2) an increase of the block size leads to declined classification performance, but at the same time, some computation load can be desirably released due to the reduction of feature dimensionality, and vice versa. Good block sizes bring balance between the classification accuracy and computation efficiency. A good choice of block size should be decided based on the application requirements.

V. Conclusion

We propose a compact unsupervised network, namely, the CUNet, that can handle various image classification tasks. The main objective is to simplify the complicated processes in traditional convolutional neural networks while achieving equivalent performance. The compact design of CUNet avoids the tuning of millions of parameters and does not require a numerical optimization solver. Besides, the unsupervised learning approach in the convolution filters resolves the issue of lacking labelled training data, which convolutional neural networks usually face. Experimental results verify that CUNet is highly competitive among the state-of-the-art works. In future work, a main target is to further improve CUNet, enabling it to handle more challenging, large-scale benchmarking datasets in which significant intra-class variance is present.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 61370149, in part by the Fundamental Research Funds for the Central Universities (No. ZYGX2016J077).

References

[1] R. Ji, L. Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao. Learning to distribute vocabulary indexing for scalable visual search. IEEE Transactions on Multimedia, vol. 15, no. 1, pp. 153-166, 2013.
[2] S. Liu, S. Yan, T. Zhang, C. Xu, J. Liu, H. Lu. Weakly supervised graph propagation towards collective image parsing. IEEE Transactions on Multimedia, vol. 14, no. 2, pp. 361-373, 2012.
[3] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: distinctive parts for scene classification. In CVPR, vol. 9, no. 4, pp. 923-930, 2013.

[4] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. In CVPR, vol. 9, no. 4, pp. 3246-3253, 2013.
[5] Qi, G. - J., X. S. Hua, Y. Rui, J. Tang, and H. J. Zhang. Image classification with kernelized spatial-context. IEEE Transactions on Multimedia, vol. 12, no. 4, pp. 278-287, 2010.
[6] P. Li, M. Wang, J. Cheng, C. Xu, H. Liu. Spectral hashing with semantically consistent graph for image indexing. IEEE Transactions on Multimedia, vol. 15, no. 1, pp. 141-152, 2013.
[7] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, pp. 818-833, 2013.
[8] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In ICML, pp. 1319-1327, 2013.
[9] M. Lin, Q. Chen, and S. Yan. Network in network. In ICLR, 2014.
[10] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In NIPS, pp.2627-2635, 2014.
[11] Y. Bengio, A. Courville, and P. Vincent. Representation learning: a review and new perspectives. IEEE TPAMI, vol. 35, no. 8, pp. 1798-1828, 2013.
[12] K. Sohn, G. Zhou, C. Lee, and H. Lee. Learning and selecting features jointly with point-wise gated Boltzmann machine. In ICML, 2013.
[13] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural network. In NIPS, vol. 25, no. 2, pp. 1097-1105, 2012.
[14] J. Bruna and S. Mallat. Invariant scattering convolution networks. IEEE TPAMI, vol. 35, no. 8, pp. 1872-1886, 2013.
[15] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In CVPR, vol. 9, no. 4, pp. 1233-1240, 2013.
[16] Y. Yang, Z. Ma, A. G. Hauptmann, and N. Sebe. Feature Selection for Multimedia Analysis by Sharing Information Among Multiple Tasks. IEEE Transactions on Multimedia, vol. 15, no. 3, pp. 661-669, 2013.
[17] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang. Image clustering using local discriminant models and global integration. In TIP, vol. 19, no. 10, pp. 2761-2773, 2010.
[18] Y. Zhuang, J. Luo, Y. Yang, F. Nie, D. Xu, and Y. Pan. A multimedia retrieval framework based on semi-supervised ranking and relevance feedback. IEEE TPAMI, vol. 34, no. 4, pp. 723-742, 2012.
[19] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-Level image representations using convolutional neural networks. In CVPR, pp. 1717-1724, 2014.
[20] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In Computer Science, vol. 52, no. 3, pp. III-1310, 2014.
[21] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. In IJCV, vol. 113, no. 2, pp. 113-127, 2015.
[22] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In ICLR, 2013.
[23] F. Nie, W. Li, C. Gao, Y. Yang, and D. Xu. Image Classification by Cross-Media Active Learning With Privileged Information. IEEE Transactions on Multimedia, vol. 18, no. 12, pp. 2494-2502, 2016.
[24] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In ICML, pp. 807-814, 2010.
[25] T. H. Chan, K. Jia, S. H. Gao, J. W. Lu, Z. N. Zeng, and Y. Ma. PCANet: A simple deep learning baseline for image classification?. In TIP, vol. 24, no. 12, pp. 5017, 2014.
[26] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng. Tiled convolutional neural networks. In NIPS, pp. 1279-1287, 2010.
[27] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In ICML, vol. 127, no. 14, pp. 1215-1222, 2010.
[28] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In NIPS, pp. 244-252, 2010.
[29] A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In AISTATS, 2011.
[30] A. Krizhevsky. cuda-convnet. http://code.google.com/p/cuda-convnet/, July 18, 2014.
[31] R. Gens, and P. Domingos. Discriminative learning of sum-product networks. In NIPS, vol. 4, pp. 3239-3247, 2012.
[32] K. Sohn, and H. Lee. Learning Invariant representations with local transformations. In ICML, pp. 1339-1346, 2012.
[33] Y. Q. Jia, O. Vinyals, and T. Darrell. Pooling-invariant image feature learning. In Computer Science, 2013.
[34] A. Romero, P. Radeva, and C. Gatta. No more meta-parameter tuning in unsupervised sparse feature learning. In Computer Science, 2014.
[35] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: explicit invariance during feature extraction. In ICML, 2011.
[36] B. Chen, G. Polatkan, G. Sapiro, D. B. Dunson, and L. Carin. The hierarchical beta process for convolutional factor analysis and deep learning. In ICML, 2011.
[37] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu. Deep learning of invariant features via simulated fixations in video. In NIPS, vol. 25, pp. 3212-3220, 2012.
[38] H. Lee, R. Grosse, R. Rananth, and A. Ng. Convolutional deep belief networks for scalable unsupervised learnig of hierachical representation. In ICML, 2009.
[39] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In NIPS, pp. 1090-1098, 2010.
[40] M. D. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional networks. In CVPR, vol. 238, no. 6, pp. 2528-2535, 2010.

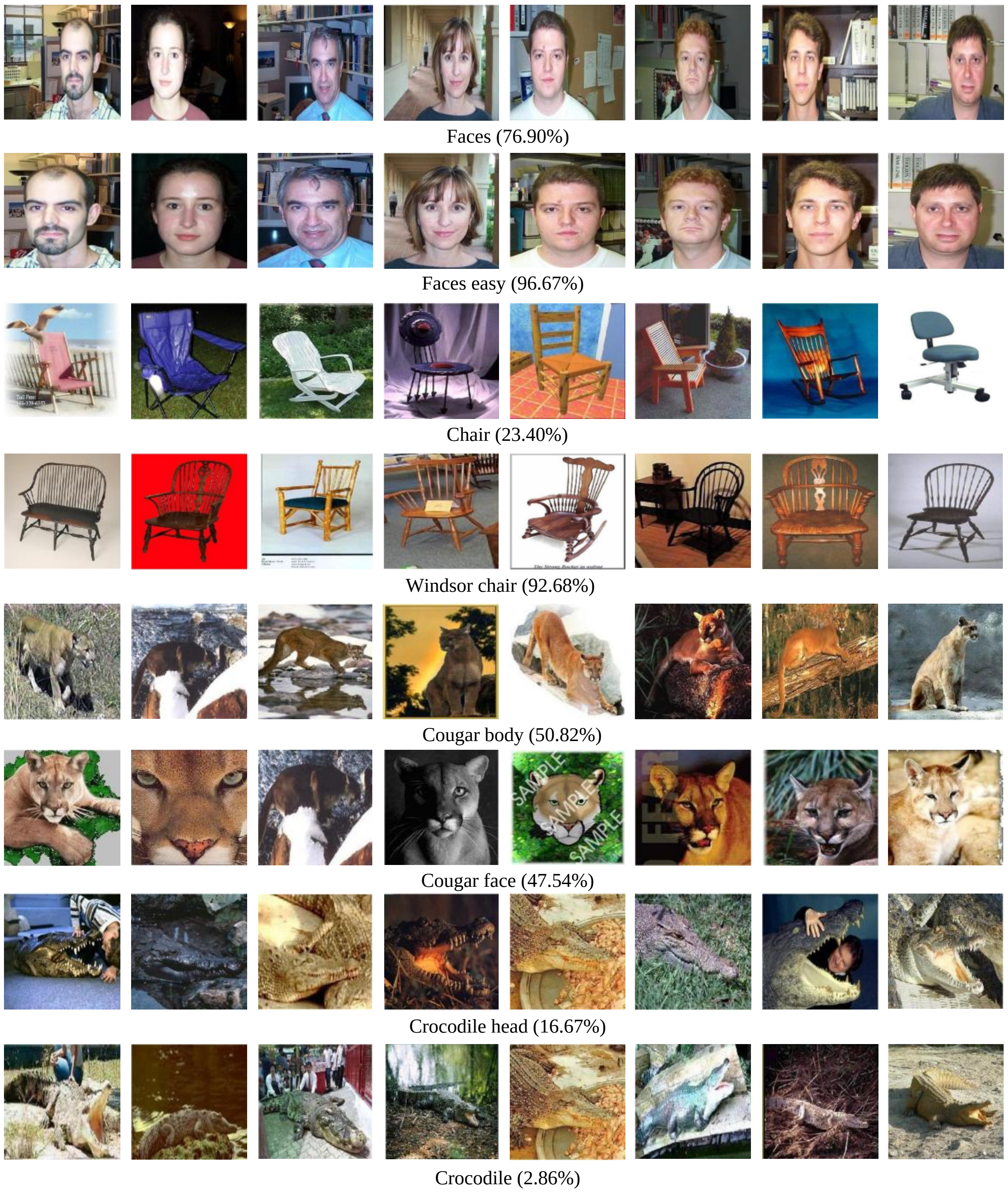

Fig. 4: Classification results on Caltech101 using 15 training images per class.