Multi-loss Regularized Deep Neural Network

Chunyan Xu, Canyi Lu, Xiaodan Liang, Junbin Gao, Wei Zheng, Tianjiang Wang, and Shuicheng Yan

***Abstract*— A proper strategy to alleviate overfitting is critical to a deep neural network (DNN). In this paper, we introduce the cross-loss-function regularization for boosting the generalization capability of the DNN, which results in the multi-loss regularized DNN (ML-DNN) framework. For a particular learning task, e.g., image classification, only a single-loss function is used for all previous DNNs, and the intuition behind the multi-loss framework is that the extra loss functions with different theoretical motivations (e.g., pairwise loss and LambdaRank loss) may drag the algorithm away from overfitting to one particular single-loss function (e.g., softmax loss). In the training stage, we pretrain the model with the single-core-loss function and then warm start the whole ML-DNN with the convolutional parameters transferred from the pretrained model. In the testing stage, the outputs by the ML-DNN from different loss functions are fused with average pooling to produce the ultimate prediction. The experiments conducted on several benchmark datasets (CIFAR-10, CIFAR-100, MNIST, and SVHN) demonstrate that the proposed ML-DNN framework, instantiated by the recently proposed network in network, considerably outperforms all other state-of-the-art methods.**

***Index Terms*— Deep neural network (DNN), multi-loss, overfitting, visual classification.**

I. Introduction

DUE to the increasing computing power and availability of large training data, there has been a resurgence of interest in neural networks. Especially, the deep neural network (DNN) learning framework [1]–[4] has drawn much attention recently, which has achieved very promising performance on the kind of vision tasks, e.g., image classification [5], [6], pedestrian detection [7], and scene labeling [8]. The successes of the DNN frameworks have largely been attributed to a considerable number of model parameters in many convolution layers. For example, the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 classification challenge, i.e., GoogLeNet [9], employed a 22-layer-deep network. Although the DNN with a large number of parameters has a powerful machine learning capability, it often suffers from overfitting.

In recent literature, some regularization techniques have been proposed to prevent the heavy overfitting during training DNNs, such as data augmentation [10], [11], weight decay [12], dropout [13], [14], dropconnect [15], and stochastic pooling [16]. Specifically, the most simple method to alleviate overfitting [10], [11] is to manually enlarge the training images with label-preserving transformations, such as horizontal/vertical reflection and image translation. The weight decay [12] is also a technique to help prevent overfitting by adding a penalty to the maximum likelihood. In [13] and [14], the dropout technique stochastically sets half of the activations within a layer to zero for each training sample in the training stage, while the dropconnect method [15] is proposed to regularize large fully connected (fc) layers in neural networks. The stochastic pooling method [16], which replaces the conventional deterministic pooling operation with a stochastic procedure, randomly picks up the activations within each pooling region according to a multinomial distribution of the activations within the pooling region. However, these approaches may suffer from a certain limitation (e.g., overfitting) for a particular loss function. The optimization target (e.g., loss function) of visual recognition has not been well explored to prevent overfitting, which is truly critical to the training of the network.

In this paper, we propose the multi-loss regularized DNN (ML-DNN) framework to harness the regularizations among a set of different loss functions. The intuition behind this multi-loss framework is that the loss functions (e.g., pairwise ranking loss and LambdaRank loss) with different theoretical motivations may prevent the algorithmic overfitting to one single-loss function (e.g., softmax loss). Specifically, the softmax loss function [1] can be used for minimizing the cross-entropy loss over all training samples. The pairwise ranking loss [18] takes the preference order of label pairs into count in classification problems, while the LambdaRank loss function [19], [20] is introduced for optimizing the top-k classification accuracy. Our ML-DNN learning framework can effectively combine all the valuable properties of these loss functions by the cross-loss-function regularization. Each kind of loss function in our ML-DNN is selected based on different theoretical motivations. Multiple-loss functions of our ML-DNN can constrain the parameters of a neural network from different

Manuscript received January 25, 2015; revised May 4, 2015 and July 12, 2015; accepted September 1, 2015. Date of publication September 11, 2015; date of current version December 2, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61572214 and Grant U1233119 and in part by the Australian Research Council Discovery Projects Funding Scheme under Project DP140102270. This paper was recommended by Associate Editor W. Zeng.

C. Xu was with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583. She is now with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: xuchunyan01@gmail.com).

C. Lu, X. Liang, and S. Yan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583 (e-mail: canyilu@gmail.com; xdliang328@gmail.com; eleyans@nus.edu.sg).

J. Gao is with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia (e-mail: jbgao@csu.edu.au).

W. Zheng is with the Beijing Samsung Telecom Research and Development Center, Beijing, China (e-mail: w0209.zheng@samsung.com).

T. Wang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: tjwang@hust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2015.2477937

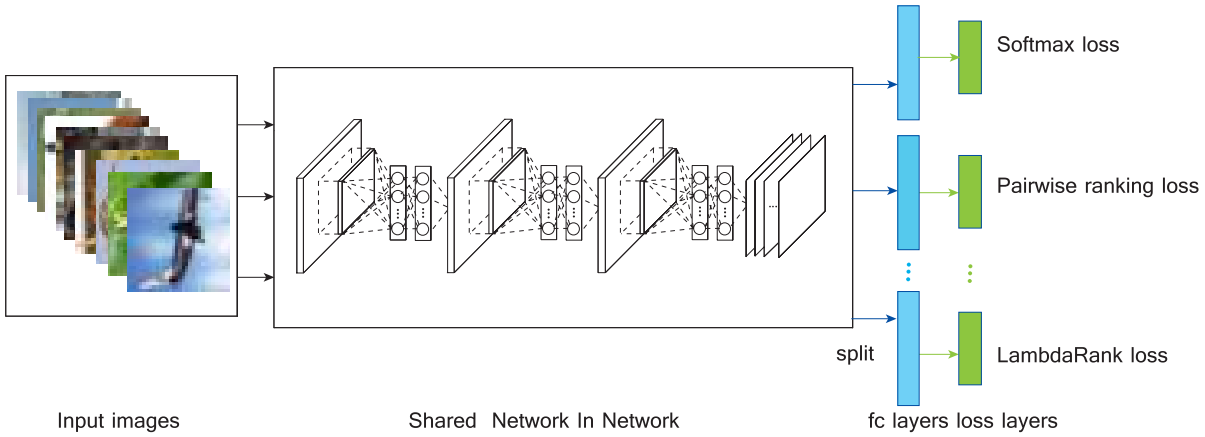


Fig. 1. Illustration of the structure of the proposed ML-DNN. The shared NIN has a network structure similar to that in [4], except for the last loss layer. We feed some given images into the shared NIN and pass the output feature maps from the shared NIN into multiple-loss branches. Each loss branch is composed of an fc layer and a loss layer. Example images are from the CIFAR-10 dataset [17].

aspects and help to regularly learn a DNN for boosting its discriminative capability.

As shown in Fig. 1, our proposed framework proposes to use the cross-loss-function regularization in the output layers, which combines the softmax, pairwise ranking LambdaRank top-1, and LambdaRank top-2 loss functions. In this paper, we instantiate our ML-DNN framework under the architecture of network in network (NIN) [4], which is one of the most popular architectures used in computer vision problems due to its model discriminability for local patches within the receptive field and its reduced number of parameters. During the ML-DNN training, the training images are first fed into the shared NIN and several parallel fc layers, each of which corresponds to some different loss layers. We pretrain the model with the single-loss function, and then warm start the whole ML-DNN with the convolutional parameters transferred from the pretrained model. We feed the convolutional feature maps of the shared NIN into multiple branches of the loss functions. Based on these multiple-loss functions, the network is trained by simultaneously optimizing all loss functions with backpropagation. In the testing stage, the outputs by the ML-DNN from different loss functions are fused with average pooling to produce the ultimate prediction.

The major contributions of this paper can be summarized as follows.

1) We propose a novel ML-DNN framework, which gracefully optimizes the architecture of DNN based on the cross-loss-function regularization.
2) For the image classification task, we present some loss functions (e.g., pairwise loss and LambdaRank top-k loss) for learning a DNN. Multiple-loss functions are simultaneously optimized with the stochastic gradient descent (SGD) learning method.
3) Our ML-DNN is a very general framework for alleviating the overfitting during learning a DNN. Any CNN architectures and any loss functions for different vision tasks can be conveniently incorporated into our framework.
4) The classification results on several standard datasets well verify the effectiveness of our proposed ML-DNN framework.

II. Related Work

DNN learning has long been studied and applied in the field of computer vision [1], [21]–[24]. More than a decade ago, LeCun *et al.* [2] trained multilayer neural networks with the back-propagation algorithm and the gradient learning technique, and then demonstrated its effectiveness on the handwritten digit recognition task. Recently, there has been a resurgence of research interest in neural networks.

A. Neural Network Structure of Deep Learning

A classic convolutional network is composed of alternatively stacked convolutional layers and spatial pooling layers. The convolutional layer is to extract feature maps by linear convolutional filters followed by nonlinear activation functions (e.g., rectifier, sigmoid, and tanh) Spatial pooling is to group the local features together from spatially adjacent pixels, which is typically done to improve the robustness to slight deformations of objects. The convolutional neural network (CNN) [1] is a special type of neural network that consists of five convolution layers, some of which are followed by max-pooling layers, and three fc layers with a final 1000-way softmax. The deep CNN has exhibited good generalization power in image-related applications. Recently, Krizhevsky *et al.* [1] has achieved a breakthrough, outperforming the existing handcrafted features on ILSVRC 2012, which contains 1000 object classes. Another deep network structure, namely, NIN [4], is proposed to build a micronetwork with more complex structures to abstract the data within the receptive field. It enhances model discriminability for local patches within the receptive field.

B. Regularization Techniques of Deep Learning

Learning neural network models is prone to overfitting because of the large number of parameters of the models. Some regularization techniques [11]–[13] are necessary for learning DNNs. Specifically, some data augmentation techniques [10], [11] are usually used to enlarge the training data, such as image translations, horizontal reflections, and image rotation and scaling. Dropout, recently proposed in [13], is another regularization approach that stochastically sets half the activations within a layer to zero for each training sample

during training. Wan *et al.* [15] proposed the dropconnect regularization, which sets a randomly selected subset of weights within the network to zero. Weight decay [12] was also proposed to prevent overfitting by adding a penalty to the maximum likelihood. The stochastic pooling method [16] was proposed to randomly select the pooled map responses by sampling from a multinomial distribution formed from the activations of each pooling region.

C. Loss Functions of Deep Learning

Some loss functions have also been introduced for learning a single DNN, e.g., softmax loss function for the classification task [4], [5], ℓ_2-norm loss function for the detection task, and sigmoid cross-entropy loss function for the multilabel classification task. Recently, Gkioxari *et al.* [25] have trained a single CNN [1] jointly for solving person detection, pose estimation, and action classification tasks, where each task is associated with a loss function. Li *et al.* [26] simultaneously learned a pose-joint regressor and a sliding-window body-part detector in a DNN, which can be seen as a heterogeneous multitask learning method. The above works can be classified into two categories: 1) single-task learning with a single-loss function and 2) multitask learning with multiple-loss functions, both in a single DNN. Recently, Szegedy *et al.* [9] also proposed a deep CNN architecture, in which three softmax loss functions are adopted, one is a total loss of the network and the other two can be regarded as the auxiliary classifiers in the training process. At the testing state, the two auxiliary loss functions and their corresponding subnetworks are discarded. This work mainly improves the discriminative capability of a neural network by appending multiple same loss functions in different stages of neural network, which alleviates the problem of vanishing gradient in a deeper CNN architecture to some extent. However, different from their usage of same loss functions, we use multiple different loss functions with different theoretical motivations at the end of a neural network, which allow each training sample to impose much more constraint on the parameters of a neural network and prevent the neural network from being too sure. From the aspect of regularization, it also should be noted that our ML-DNN can be posed as a good regularizer for preventing overfitting using multiple-loss functions in one network.

III. Multi-Loss Regularized DNN

In this section, we first introduce the network structure of the proposed ML-DNN. Second, several loss functions are investigated for cross-loss-function regularization. Third, we show how to learn an ML-DNN by simultaneously optimizing multiple-loss functions. Finally, testing with the ML-DNN is conducted with the average pooling algorithm.

A. Network Structure

The structure of the overall ML-DNN framework is mainly composed of two parts, namely, the shared NIN and multiple-loss branches, as shown in Fig. 1.

1) Shared NIN: The shared NIN has a similar network architecture as NIN [4]. By building microneural networks with more complex structures to abstract the data within the receptive field, Lin *et al.* [4] proposed a novel deep network structure called NIN to enhance model discriminability for local patches within the receptive field. As shown in Fig. 1, the microneural network is instantiated with a multilayer preceptron (MLP), which is a potent function approximator. The feature maps are obtained by sliding the micronetworks over the input in a similar way as CNN [1], [2]. The multilayer perceptron convolution (MLP-conv) maps the input local patch to the output feature vector with an MLP, which consists of multiple fc layers with nonlinear activation functions. The shared NIN is implemented by stacking the above three MLP-conv layers. For different loss functions, we use one shared NIN in the structure of the ML-DNN. The reason is that the MLP-conv layers form the general hierarchical feature representation for image data, which should be shared by all loss functions.

2) Multi-Loss Regularization: The multiple-loss functions correspond to multiple-loss branches of our proposed ML-DNN framework. Multiple-loss functions may be able to prevent the algorithm away from overfitting to one single-loss function, which may occur with only one loss function used. As shown in Fig. 1, multiple-loss branches are embedded into the shared NIN. For each loss branch, one fc layer and one output layer are included. The fc layer is followed by the Rectified Linear Units (ReLu) nonlinearity and 0.5 dropout. We optimize each output layer with different loss functions. During the network training, we feed the convolutional feature maps of the shared NIN into multiple branches of the loss functions. Based on these multiple-loss functions, the network is trained by simultaneously optimizing all loss functions with backpropagation. The last convolution layer in the shared NIN is connected to multiple fc layers. The parameter gradients from different loss functions can thus be conveniently used to optimize the parameters of the shared NIN. Different kinds of loss functions are based on different theoretical motivations. For example, the softmax loss, pairwise ranking loss, and LambdaRank top-1 loss belong to the top-1 loss, while the LambdaRank top-2 loss is a kind of top-2 loss. Thus, different loss functions have certain complementariness and the gradients brought by them help iteratively learn parameters from different aspects. In this way, the whole ML-DNN is able to consider all these multiple-loss functions simultaneously and avoid the overfitting problem during training. It is worth noting that our framework can be instantiated by any CNN architectures and any loss functions that may be designed for different application scenarios in computer vision. In this paper, we used several most widely used loss functions for visual recognition.

B. Loss Functions

In this section, we introduce several widely used loss functions in our proposed multi-loss regularization with more details. We use the softmax loss, pairwise ranking loss, and LambdaRank loss, which enjoy quite different regularization properties to regularize the training of DNNs. It is expected that using these loss functions will effectively alleviate the overfitting issue in the previous DNNs.

For clarity, some notations used in the following are introduced here. We use c_i^+ and c_i^- to denote the positive and

negative labels of the image $X_i, i = 1, \ldots, N$, where N is the total number of training images and $|c_i^+|$ is the number of positive labels. We use $q_j(X_i)$ to specify the discrete probability distribution of the image X_i in the jth class, $j = 1, \ldots, C$, where C is the number of possible classes.

1) Softmax Loss: Most of previous DNNs [1], [4], [5] employ the softmax loss function for optimizing DNN parameters. With the softmax function, the normalized probability of the image X_i in the jth class can be computed by

$$p_{i,j} = \frac{\exp(q_j(X_i))}{\sum_{j=1}^{C} \exp(q_j(X_i))}. \tag{1}$$

To minimize the Kullback Leibler divergence between the predictions and the ground-truth probabilities, the softmax cost function is defined as

$$J_{\text{softmax}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} \bar{p}_{i,j} \log(p_{i,j}) \tag{2}$$

where $\bar{p}_{i,j}$ denotes the ground-truth probability between the image X_i in class j. Each image X_i corresponds to a label vector $y_{i,j} \in \mathbb{R}^C$, with $y_{i,j} = 1$ and $\bar{p}_{i,j} = 1$ indicating the presence of a label for the image X_i, and $y_{i,j} = 0$ and $\bar{p}_{i,j} = 0$ indicating the absence of a label for the image X_i.

2) Pairwise Ranking Loss: The second loss function we consider is the pairwise ranking loss [18], [27]. The pairwise ranking method can transform the classification problem into a task of classifying the preference order of label pairs: given any two labels, it will decide which label should be ranked first. The pairwise ranking loss method minimizes the classification error on the pairs of labels. Specifically, the goal of the pairwise ranking method is to rank all the labels so that positive labels always have higher prediction scores than negative labels. The pairwise ranking loss J_{pairwise} is defined as

$$J_{\text{pairwise}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{m=1}^{c_i^+} \sum_{n=1}^{c_i^-} \max(0, 1 - q_m(X_i) + q_n(X_i)) \tag{3}$$

where $m = 1, \ldots, c_i^+$ and $n = 1, \ldots, c_i^-$ represent the indexes of the positive labels and negative labels, respectively. By minimizing J_{pairwise}, we compute the subgradient of this loss function during the training of the ML-DNN.

3) LambdaRank Loss: In order to directly optimize the top-k classification accuracy, the third loss function, LambdaRank loss, is introduced to learn an ML-DNN. The LambdaRank method [28] obtains the desired gradients directly, rather than deriving them from a cost loss function. In this way, it can bypass the difficulties brought by the ranking loss function. The key observation of LambdaRank is that in order to train a DNN model, we do not need the losses themselves; we need only the gradients of the losses with respect to the DNN model outputs.

Before discussing the LambdaRank loss function, we first introduce some information about RankNet [19]. For a given image X_i, the set of the label pairs is denoted by S, $(m, n) \in S$ if and only if $m \in c_i^+$ and $n \in c_i^-$. The two outputs $q_m(X_i)$ and $q_n(X_i)$ are mapped to a learned probability that the mth label is ranked higher than the nth label by a sigmoid function, and thus

$$p_{m,n} = \frac{1}{1 + \exp(-\gamma\,(q_m(X_i) - q_n(X_i)))} \tag{4}$$

where the parameter γ is directly related to scaling. We adopt the cross-entropy loss function, which penalizes the deviation of the model output probabilities from the desired probabilities. Then the cost is

$$J_{mn} = \log(1 + \exp(-\gamma\,(q_m(X_i) - q_n(X_i)))). \tag{5}$$

The gradient of the above loss function is

$$\begin{aligned} \frac{\partial J_{mn}}{\partial q_m(X_i)} &= \frac{-\gamma}{1 + \exp(-\gamma\,(q_m(X_i) - q_n(X_i))} \\ \frac{\partial J_{mn}}{\partial q_n(X_i)} &= -\frac{\partial J_{mn}}{\partial q_m(X_i)}. \end{aligned} \tag{6}$$

From the RankNet to LambdaRank loss, the gradient is multiplied by η_{mn}, which calculates the changes in evaluation measures by swapping the rank position of the mth label and the nth label.

The model can directly optimize the evaluation measures [20], and the gradient of $q_m(X_i)$ is

$$\frac{\partial J_{mn}}{\partial q_m(X_i)} = \sum_{(m,n) \in \mathcal{S}} \eta_{mn} \frac{-\gamma}{1 + \exp(\gamma\,(q_m(X_i) - q_n(X_i))}. \tag{7}$$

Similarly

$$\frac{\partial J_{mn}}{\partial q_n(X_i)} = \sum_{(m,n) \in \mathcal{S}} \eta_{mn} \frac{\gamma}{1 + \exp(\gamma\,(q_m(X_i) - q_n(X_i))}. \tag{8}$$

In order to optimize the loss function with top-k evaluation measures, $\eta_{mn}(k)$ is defined as the order change in top-k. It is easily observed that $\eta_{mn}(k) = (1/|c_i^+|)$, if m is in the top-k prediction ($m \leq k$) and n is not in the top-k prediction ($n > k$). Otherwise, $\eta(m, n)$ is zero. In this paper, we consider only the top-1 and top-2 evaluation measures, and then name them as LambdaRank top-1 loss function and LambdaRank top-2 loss function, respectively.

C. Parameter Learning of an ML-DNN

The DNN with a large number of parameters has a powerful machine learning capability, but it is difficult to prevent it from overfitting. To address this problem, some regularization techniques are proposed for training DNNs, such as data augmentation [10], [11], weight decay [12], dropout [13], [14], dropconnected [15], and stochastic pooling [16]. These approaches may suffer from a certain limitation for a particular loss function. The optimization of loss functions has not been well explored to prevent the overfitting problem, which is truly critical to learning parameters.

In order to alleviate overfitting, we propose to train an ML-DNN model with the multiple-loss functions in this paper. For learning an ML-DNN, we adopt four kinds of loss functions, including softmax loss, pairwise loss, LambdaRank top-1 loss, and LambdaRank top-2 loss. The motivation of this ML-DNN framework is that other loss functions, e.g., pairwise

ranking loss and LambdaRank loss, may have the potential to prevent the algorithm overfitting to one softmax loss function. Therefore, the key idea of our ML-DNN learning framework is to use the cross-loss-function regularization for boosting the generalization capability of the ML-DNN model. Based on the above loss functions, we propose the multi-loss regularization method for learning an effective ML-DNN model.

First, we warm start the shared NIN by initializing it with the pretrained parameters in [4]. In order to improve the generalization capability of DNNs, fine tuning is then performed to adjust the parameters of the ML-DNN with the cross-loss-function regularization method. We learn an ML-DNN model by simultaneously optimizing multiple-loss functions in the training stage. The underlying reason for warm start is that our offline experiments show that when the step size is large in the early stages, the gradients from different objects are quite diverse, which makes the optimization quite slow in convergence. Warm start can well avoid this issue. We then use the back-propagation technique [2] to update the parameters of the ML-DNN model. Given some training images, the predicted probabilities from the multiple-loss functions are calculated and the corresponding gradients are obtained by backpropagation through the network. For the shared NIN, the gradients from the fc layers are summed together for parameter updating. In each iteration, the neurons in dropout layers will be stochastically selected with a probability of 0.5 to forward their activation to the output units and only the selected neurons will participate in the backpropagation during this iteration. Similarly, we adopt all the neurons for prediction with their activation value multiplied by 0.5 for normalization.

D. Testing With Average Pooling

For each test image, we feed it into our ML-DNN framework and output several probability outputs corresponding to the different loss functions in the training. We then fuse the probabilities from the different loss functions. We adopt the average pooling technique to produce the ultimate prediction of the testing image. The probability of the image X_i with the jth label is given by

$$P_j(X_i) = \frac{\sum_{l=1}^{L} P_{l,j}(X_i)}{L} \tag{9}$$

where L is the total number of loss functions and $P_{l,j}(X_i)$ denotes the probability of the image X_i from the lth loss function.

Our testing process with the ML-DNN model is different from simply combining the predictions of several DNNs. We can analyze their differences from three characteristics.

1) Previous combination methods often separately train different DNNs with same/different loss functions, and their predictions are then combined with the pooling technique in the testing. It is difficult to alleviate the overfitting problem for learning DNNs.
2) The training time of training one multi-loss convolutional neural network is only slightly more than training a DNN with a single-loss function, that is, only the additional fc layer and output layers are trained instead of training all convolution layers many times for all lost functions.
3) The ML-DNN uses the cross-loss-function regularization, which can boost the generalization capability of the DNN. The diverse properties of different loss functions can be combined in our unified framework. Therefore, learning the ML-DNN model is not only faster than training multiple DNNs but also can improve the performance of DNN by alleviating the overfitting problem.

IV. Experiments

We evaluate the effectiveness of our proposed ML-DNN method on four standard benchmark datasets: Canadian Institute for Advanced Research-10 (CIFAR-10) [17], Canadian Institute for Advanced Research-100 (CIFAR-100) [17], Mixed National Institute of Standards and Technology (MNIST) [2], and Street View House Numbers (SVHN) [31].

A. Experimental Settings

To ensure the fairness of comparisons with previous baselines, all experiments are conducted based on the following experimental setups. Test error is used as the evaluation metric. We use four different loss functions for learning an ML-DNN model: softmax loss, pairwise ranking loss, LambdaRank top-1, and LambdaRank top-2 loss functions. For the shared NIN, we follow the same network definitions used in [4], which are publicly available.[1] For each loss branch of our ML-DNN, the size of the fc layer is 256 for the CIFAR-10, MNIST, and SVHN datasets, and 2560 for the CIFAR-100 dataset, followed by the ReLU nonlinearity and dropout (50%) layer. For updating the ML-DNN parameters, we use the SGD learning method with the minibatch size of 128 at a fixed constant momentum value of 0.9. The weight decay is set to 0.001. The global learning rate of learning the ML-DNN model is set to 0.01. We decrease the learning rate every 100k iterations by 10. For the parameter learning of ML-DNN, the local learning rate of the shared NIN is set to 0.01. For each loss branch, the learning rate for all fc layers and all loss layers is the same and set to 1.0. The maximum of iterations is set to 120 000.

B. CIFAR-10

The CIFAR-10 dataset [17] consists of 60 000 32×32 color images of ten classes, with 6000 images per class. There are 50 000 training images and 10 000 test images. The dataset is divided into five training batches and one test batch, each with 10 000 images. The test batch contains 1000 randomly selected images from each class. Following the same experiment settings used in [5], we process the data with the same global contrast normalization and zero component analysis whitening. To fairly compare with the other previous algorithms [4], [5], we have conducted extensive experiments under two settings (i.e., without data augmentation and with data augmentation). In terms of data augmentation, we augment the images by zero padding 4 pixels on each direction and then perform corner cropping and random flipping during training.

[1] https://github.com/mavenlin/cuda-convnet/tree/master/NIN

TABLE I
CIFAR-10 CLASSIFICATION ERRORS OF VARIOUS METHODS

Method	Test Error (%)
Without data augmentation	
Stochastic Pooling [16]	15.13
Maxout Networks [5]	11.68
Maxout Units [29]	11.35
NIN [4]	10.41
ML-DNN	**9.55**
With data augmentation	
Maxout Networks [5]	9.38
Drop-Connect [15]	9.32
NIN [4]	8.81
ML-DNN	**8.12**

Under above two settings, we compare our proposed ML-DNN method with previous DNN regularization techniques, including stochastic pooling [16], maxout networks [5], NIN [4], and probabilistic maxout units [29]. The detailed classification error results are shown in Table I. The comparison results on CIFAR-10 indicate that the proposed ML-DNN method achieves 9.55% test errors without data augmentation and 8.12% test error with data augmentation, both of which perform better than all the baselines. The ML-DNN improves the performance of the NIN [4] of 10.41% by more than 0.8% without data augmentation and that of 8.81% by higher than 0.69% with data augmentation. The baseline methods achieve the classification error of 15.13% for the stochastic pooling method [16], 11.68% for maxout networks [5], and 11.35% for maxout units [29], all of which are much higher than the test error of our ML-DNN without data augmentation. Similarly, we also improve a test error by 0.69%, compared with the baseline NIN [4]. This well verifies the superiority of our ML-DNN method, which is very effective for achieving better generalization capability during the ML-DNN training.

C. CIFAR-100

This CIFAR-100 dataset [17] has the same number of images as the CIFAR-10 database, but contains 100 classes, with only one tenth as labeled examples per class. There are 500 training images and 100 testing images per class. For the CIFAR-100 dataset, we adopt the same network settings and data preprocessing (including the data augmentation technique) as the CIFAR-10, described in [4].

Table II shows the performances of our proposed ML-DNN and other state-of-the-art methods. It can be seen that the ML-DNN method gets a misclassification test error of 34.18%, which significantly improves the performance over the NIN [4] by less 1.5% errors. In particular, our method also beats all other regularization methods with a large margin, e.g., 42.51%, 38.57%, and 38.14% for stochastic pooling [16], maxout networks [5], and probabilistic maxout units [29], respectively. Moreover, we also have conducted an experiment with data augmentation and achieved a test error of 31.47%, which is lower than NIN [4] by 2.06%. Our proposed ML-DNN method can obtain better performances over other state-of-the-art methods on classifying 100 classes.

TABLE II
CIFAR-100 CLASSIFICATION ERRORS OF VARIOUS METHODS

Method	Test Error (%)
Without data augmentation	
Stochastic Pooling [16]	42.51
Maxout Networks [5]	38.57
Maxout Units [29]	38.14
NIN [4]	35.68
ML-DNN	**34.18**
With data augmentation	
NIN [4]	33.53
ML-DNN	**31.47**

TABLE III
MNIST CLASSIFICATION ERRORS OF VARIOUS METHODS

Method	Test Error (%)
2layer CNN+2layer NN [16]	0.53
Stochastic Pooling [16]	0.47
NIN [4]	0.47
Maxout Networks [5]	0.45
ML-DNN	**0.42**

TABLE IV
SVHN CLASSIFICATION ERRORS OF VARIOUS METHODS

Method	Test Error (%)
Stochastic Pooling [16]	2.80
Maxout Networks [5]	2.47
NIN [4]	2.35
Multi-digit num. recog. [30]	2.16
ML-DNN	**1.92**

D. MNIST

The MNIST handwritten digit classification dataset [2] consists of 28×28 pixel grayscale images of handwritten digits (from 0 to 9). There are 60 000 training images and 10 000 testing images in total.

The comparison results are presented in Table III. It can be observed that our proposed method outperforms the competing methods on the MNIST dataset. It is noted that the test error of our ML-DNN is lower than all the competing methods (such as NIN [4], maxout networks [5], and 2 layer CNN + 2 layer NN and stochastic pooling [16]). Our ML-DNN method gets 0.42% classification error and gets a lower classification error than the baseline method NIN by 0.05%. The performance on MNIST also demonstrates the advantage of the ML-DNN method.

E. SVHN

The SVHN dataset [31] is composed of 73 257 images for training, 26 032 images for testing, and 531 131 extra training color images of 32×32. The task of the dataset is to classify the digit located at the center of each image. Following [5], we select out 400 samples per class from the training set and 200 samples per class from the extra set. The rest of the training set and the extra set are used for training. We preprocess the dataset by local contrast normalization.

As reported in Table IV, the classification error of the ML-DNN significantly outperforms the other state-of-the-art methods including the stochastic pooling [16], maxout

networks [5], multidigit number recognition method [30], and NIN [4]. The proposed ML-DNN method improves the performance by 0.24% and 0.43% over the state-of-the-art methods [30] and [4], respectively. Compared with the existing stochastic pooling and maxout networks methods, the ML-DNN significantly improves the performance by 0.88% and 0.55%, respectively. It demonstrates that the method effectively works for boosting the generalization capability of the ML-DNN model.

F. Algorithm Analysis

As observed from Tables I–IV, the classification errors achieved by our ML-DNN method are reported on the above four datasets, 9.55% for CIFAR-10, 34.18% for CIFAR-100, 0.42% for MNIST, and 1.92% for SVHN. The following can be concluded from Tables I–IV.

1) The ML-DNN method does improve the performance of the current DNN classification framework and even on the difficult CIFAR-100 dataset.
2) Our regularized technique with the cross-loss function plays an important role in alleviating overfitting problem for learning a capable DNN model and the resulting ML-DNN can obviously improve the classification performance.
3) Compared with some existing regularization techniques (e.g., dropout, stochastic pooling, and maxout network), the ML-DNN does improve the capability of the DNN model by multi-loss regularization.

To validate the regularization capability of the ML-DNN, we show the comparison results of different DNNs using different single-loss functions, their combined average results (namely as Average results from four model combination), and our ML-DNN results, which are presented in Table V. The architecture of single-loss DNNs also utilizes the NIN architecture, followed by a single-loss function. The difference between the single-loss DNNs (e.g., NIN + softmax loss, NIN + pairwise ranking loss, NIN + LambdaRank top-1 loss, and NIN + LambdaRank top-2 loss) and the ML-DNN model is the strategy of using loss function (e.g., single loss or multiple different loss functions). The ML-DNN method outperforms the four kinds of the single-loss DNNs and even surpasses the most popular softmax loss for learning a DNN model. Moreover, our ML-DNN outperforms the average results by combining the four DNNs with single loss by 0.03% and 0.6% over CIFAR10 and CIFAR-100 without data augmentation, and 0.22% and 0.98% over CIFAR10 and CIFAR-100 with data augmentation. This superiority can be observed in different datasets, which can further validate the effectiveness of the ML-DNN.

We furthermore analyze the results of each loss branch in the framework of our ML-DNN. Each loss branch of the ML-DNN corresponds to a fc layer and a loss layer. The detailed classification test errors are shown in Table VI. The result of our ML-DNN, which combines the classification results of four branches, is better than the results from each branch of the ML-DNN. Compared the results of the single-loss DNN in Table V, each branch of the ML-DNN can get a higher classification result. For example, the softmax loss branch of our ML-DNN outperforms the single-loss DNN with softmax loss, e.g., 10.41% versus 9.66% on CIFAR-10 without data augmentation and 33.53% versus 32.34% on CIFAR-100 with data augmentation. It demonstrates that compared with the single-loss DNN, the ML-DNN method can boost the performance for the image classification task on four benchmark datasets.

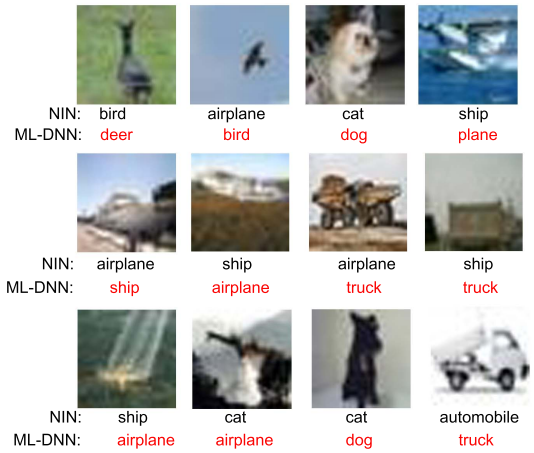


Fig. 2. Exemplar test images from CIFAR-10 [17]. These images are wrongly classified by the NIN, but correctly classified by the ML-DNN. There are two label names marked below each images: the wrong classification by the NIN, in black, and the right classification by the ML-DNN, in red.

Some test images from the CIFAR-10 dataset [17] are shown in Fig. 2. These images are wrongly classified by the NIN, but correctly classified by the ML-DNN. There are two label names marked below each image, one of which is its wrong class with black font by the NIN and the other is its right label (i.e., the ground-truth label) by the ML-DNN. For example, the bird and airplane have the blue sky, two wings, and a similar color. The automobile and truck are also similar in appearance (such as wheels, seat, and color). Our proposed ML-DNN learning employs cross-loss-function regularization, while the NIN learns the parameters with only a softmax loss function. The effectiveness of our proposed ML-DNN again speaks well that our method can successfully recognize the confusing and difficult objects.

As shown in Fig. 3, we further report some predicted probabilities from the four different loss functions and the fused results using the ML-DNN model. The output layers predict different label probabilities, due to the intrinsic properties of different loss functions. For example, for the image of the first column, the predicted label by the softmax loss and pairwise ranking loss are different from that of the LambdaRank top-1 loss and the LambdaRank top-1 loss functions, while the ultimate predicted label is the same with that of the LambdaRank top-1 loss and the LambdaRank top-1 loss functions. By employing the average pooling technique to fuse all the probabilities from all loss layers, we can achieve the best results by balancing their probabilities. The average pooling results can overall be considered the outputs of the ML-DNN from four different loss functions and improve the generalization capability of the ML-DNN.

TABLE V

CLASSIFICATION ERROR COMPARISONS BETWEEN THE SINGLE-LOSS DNNS AND THE ML-DNN

Method	Test Error (%)			
	Without data augmentation		With data augmentation	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
NIN + Softmax loss	10.41	35.68	8.81	33.53
NIN + Pairwise ranking loss	10.57	36.76	9.27	34.47
NIN + LambdaRank top-1 loss	10.37	36.83	9.34	34.62
NIN + LambdaRank top-2 loss	9.94	36.24	9.37	33.87
Average results (four model combination)	9.58	34.78	8.34	32.35
ML-DNN	**9.55**	**34.18**	**8.12**	**31.47**

TABLE VI

CLASSIFICATION ERRORS OF EACH BRANCH OF THE ML-DNN AND THE ML-DNN

Method	Test Error (%)			
	Without data augmentation		With data augmentation	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
ML-DNN's Softmax loss branch	9.66	35.54	8.35	32.34
ML-DNN's Pairwise ranking loss branch	9.75	36.66	8.82	33.27
ML-DNN's LambdaRank top-1 loss branch	9.84	35.62	8.37	31.93
ML-DNN's LambdaRank top-2 loss branch	9.93	35.68	8.64	31.82
ML-DNN	**9.55**	**34.18**	**8.12**	**31.47**

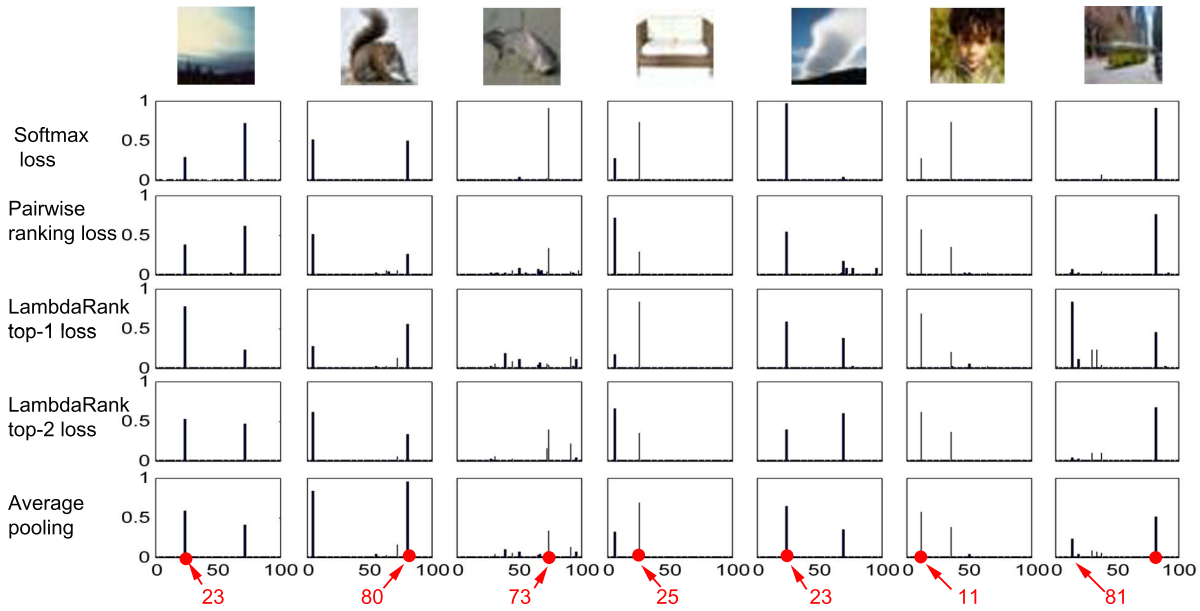


Fig. 3. Exemplar images with the predicted probabilities on the CIFAR-100 test dataset [17] using the ML-DNN. Some testing images and 100 predicted probabilities for all labels are shown. The predictions from the second to fifth rows are from four different loss functions. The ultimate results in the last row are produced by fusing the predictions from four different loss functions with the average pooling technique. In addition, the red points in the last row are marked for the ground-truth label of the corresponding test images.

To further analyze the complementary among multiple-loss functions, we show the gradient values brought by the four different loss functions in our ML-DNN framework. As can be observed in Fig. 4, the four different curves represent the gradient values brought by the four different loss branches, such as softmax loss (blue curve), pairwise ranking loss (green curve), LambdaRank top-1 loss (magenta curve), and LambdaRank top-2 loss (light blue curve). The light blue curve, which presents the gradient values brought from the LambdaRank top-2 loss branch, is different from other three curves of top-1 loss branches (e.g., softmax loss, pairwise ranking loss and LambdaRank top-1 loss). We can thus say that the top-2 loss function is complementary to these top-1 loss functions to some extent. Moreover, it can be observed that the gradient values (blue, green, and magenta curves) brought by the three top-1 loss functions also have a certain degree of difference. Therefore, multiple-loss functions of our ML-DNN can help constrain the parameters of a neural network from different aspects. From the above analyses and the practical numerical experiments conducted in this paper, we can conclude that the multi-loss can actually guide learning a better network and improve the classification performance of it.

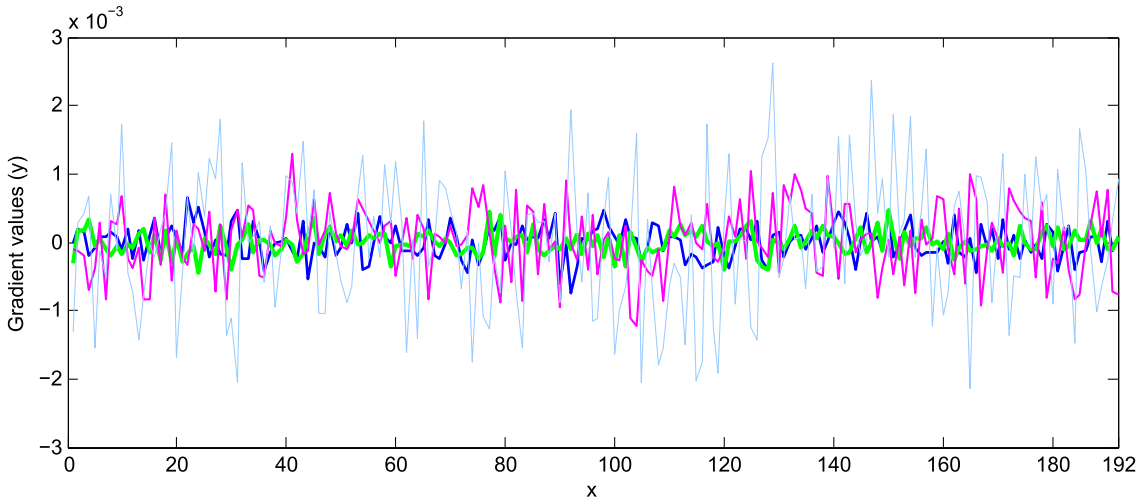


Fig. 4. Gradient values brought by four loss branches on the CIFAR-100 dataset. In order to conveniently present the gradient values (y) brought by the four loss branches, we just show some gradient values (i.e., the first gradient values of the upper left corner of each feature maps over all channels) in the 100th iteration of the optimization process. The horizontal axis ($x = 192$) is the number of channels. The four different curves (blue, green, magenta, and light blue curves) denote the gradient values brought by four different loss branches, e.g., softmax loss, pairwise ranking loss, LambdaRank top-1 loss, and LambdaRank top-2 loss, respectively.

TABLE VII
CLASSIFICATION ERRORS OF VARIOUS METHODS ON THE ImageNet DATASET

Method	Test Error (%)	
	Top-1	Top-5
Sparse coding [34]	47.1	28.2
SIFT+FVs [35]	45.7	25.7
Alex-net [1]	37.5	17.0
VGG(deep16) [33]	28.97	10.15
VGG(deep16)+ML-DNN	**28.46**	**9.8**

To evaluate the performance of our ML-DNN on the compelling and significantly harder vision task, we perform an experiment on the ImageNet dataset [32]. For the ImageNet dataset, we adopt the same setting with [33], which extracted random 224×224 patches from 256×256 training images and then trained a DNN on these patches. In the training process, we also crop 224×224 patches from the middle of test images. As described in Section III, the structure of the ML-DNN framework is composed of the shared NIN and multiple-loss branches, while our ML-DNN on this compelling problem also employs the same multiple-loss branches, but adopts a deeper DNN structure, i.e., Visual Geometry Group (VGG) (deep16) network [33]. Following [33], we also evaluate the performance from two aspects, the top-1 test error and the top-5 test error. The detailed test error of various methods on the ImageNet dataset can be observed in Table VII. Our ML-DNN slightly ourperforms the VGG (deep16) network with single loss: 28.46% versus 28.97% for the top-1 test error and 9.8% versus 10.15% for the top-5 test error. The main reason for these improvements on this challenging task may be that the ML-DNN can well boost its discriminative capability by considering cross-loss-function regularization, even with the deeper network architecture.

V. CONCLUSION

In this paper, we proposed a general multi-loss regularized DNN framework for alleviating the overfitting issue of DNN. To boost the generalization capability of DNN, this general scheme allows us to learn an ML-DNN model by simultaneously optimizing multiple-loss functions. For the image classification task, we studied the loss functions, pairwise loss, and LambdaRank top-k loss, for learning the ML-DNN model. With the average pooling technique, the final prediction can be simply computed from the outputs of the ML-DNN model from different loss functions. Extensive experimental results on the CIFAR-10, CIFAR-100, MNIST, and SVHN datasets clearly demonstrate that the proposed ML-DNN framework achieved the state-of-the-art performances. In the future, we plan to further explore the performance of the ML-DNN with other vision tasks, e.g., object detection, image retrieval, and image annotation.

REFERENCES

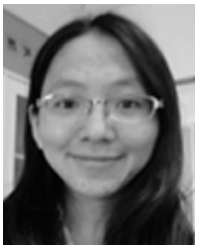
[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[3] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 609–616.
[4] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent.*, 2014.
[5] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. (2013). "Maxout networks." [Online]. Available: http://arxiv.org/abs/1302.4389
[6] Y. Wei *et al.* (2014). "CNN: Single-label to multi-label." [Online]. Available: http://arxiv.org/abs/1406.5726
[7] X. Zeng, W. Ouyang, M. Wang, and X. Wang, "Deep learning of scene-specific classifier for pedestrian detection," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 472–487.
[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.
[9] C. Szegedy *et al.* (2014). "Going deeper with convolutions." [Online]. Available: http://arxiv.org/abs/1409.4842
[10] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3642–3649.

[11] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, vol. 2. 2003, pp. 958–963.
[12] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.
[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. (2012). "Improving neural networks by preventing co-adaptation of feature detectors." [Online]. Available: http://arxiv.org/abs/1207.0580
[14] 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.
[15] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.
[16] M. D. Zeiler and R. Fergus. (2013). "Stochastic pooling for regularization of deep convolutional neural networks." [Online]. Available: http://arxiv.org/abs/1301.3557
[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[18] T. Joachims, "Optimizing search engines using clickthrough data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 133–142.
[19] C. Burges *et al.*, "Learning to rank using gradient descent," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 89–96.
[20] P. Donmez, K. M. Svore, and C. J. C. Burges, "On the local optimality of LambdaRank," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2009, pp. 460–467.
[21] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1071–1083, Dec. 2002.
[22] H. Choi and C. Lee, "Motion adaptive deinterlacing with modular neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 6, pp. 844–849, Jun. 2011.
[23] K. Kim, S. Lee, J.-Y. Kim, M. Kim, and H.-J. Yoo, "A configurable heterogeneous multicore architecture with cellular neural network for real-time object recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 11, pp. 1612–1622, Nov. 2009.
[24] N. Sudha, A. R. Mohan, and P. K. Meher, "A self-configurable systolic architecture for face recognition system based on principal component neural network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 8, pp. 1071–1084, Aug. 2011.
[25] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. (2014). "R-CNNs for pose estimation and action detection." [Online]. Available: http://arxiv.org/abs/1406.5212
[26] S. Li, Z.-Q. Liu, and A. B. Chan, "Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 488–495.
[27] J. Li, Y. Tian, T. Huang, and W. Gao, "Multi-task rank learning for visual saliency estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 623–636, May 2011.
[28] C. J. C. Burges, "From RankNet to LambdaRank to LambdaMart: An overview," *Learning*, vol. 11, pp. 523–581, 2010.
[29] J. T. Springenberg and M. Riedmiller. (2013). "Improving deep neural networks with probabilistic maxout units." [Online]. Available: http://arxiv.org/abs/1312.6116
[30] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. (2013). "Multi-digit number recognition from street view imagery using deep convolutional neural networks." [Online]. Available: http://arxiv.org/abs/1312.6082
[31] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 4–12.
[32] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. CVPR*, Jun. 2009, pp. 248–255.
[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2014.
[34] A. Berg, J. Deng, and F. F. Li. (2010). *ImagNet Large-Scale Visual Recognition Challenge*. [Online]. Available: http://image-net.org/challenges/LSVRC/2010/
[35] J. Sanchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 1665–1672.

Chunyan Xu received the B.Sc. degree from Shandong Normal University, Jinan, China, in 2007, the M.Sc. degree from Huazhong Normal University, Wuhan, China, in 2010, and the Ph.D. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, in 2015.

She was a Visiting Scholar with the National University of Singapore, Singapore, from 2013 to 2015. She is currently a Lecturer with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her current research interests include deep neural network, computer vision, manifold learning, and kernel methods.

Canyi Lu received the bachelor of mathematics from Fuzhou University, in 2009, and the master's degree from the University of Science and Technology of China in the pattern recognition and intelligent system in 2012. From August 2013, he was a Ph.D. Student with the Department of Electrical and Computer Engineering, National University of Singapore. His research interests include computer vision and machine learning. His homepage is https://sites.google.com/site/canyilu.



Xiaodan Liang is currently pursuing the Ph.D. degree with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China.

She is with the National University of Singapore, Singapore, as a Research Intern. Her current research interests include semantic segmentation, object/action recognition, and medical image analysis.

Junbin Gao received the B.Sc. degree in computational mathematics from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1991.

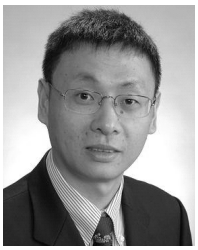
He was an Associate Lecturer, a Lecturer, an Associate Professor, and a Professor with the Department of Mathematics, HUST, from 1982 to 2001. He was a Senior Lecturer and Lecturer in Computer Science with the University of New England, Armidale, NSW, Australia, from 2001 to 2005. He is currently a Professor of Computing Science with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW, Australia. His current research interests include machine learning, data mining, Bayesian learning and inference, and image analysis.

Wei Zheng received the bachelor's degree from Tsinghua University, Beijing, China, in 2006, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2013.

He is currently a Researcher with the Beijing Samsung Telecom Research and Development Center, Beijing. His current research interests include image categorization, object detection, and scene analysis.

Tianjiang Wang received the B.Sc. degree in computational mathematics and the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982 and 1999, respectively.

He is currently a Professor with the School of Computer Science, HUST. He has finished some related projects and authored over 20 related papers. His current research interests include machine learning, computer vision, and data mining.



Shuicheng Yan was an ISI Highly Cited Researcher in 2014. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, and the Founding Leader of the Learning and Vision Research Group. He has authored or co-authored hundreds of technical papers over a wide range of research topics, with Google Scholar citation $>15\,000$ times and an H-index of 51. His current research interests include machine learning, computer vision, and multimedia.

Dr. Yan was a fellow of the International Association for Pattern Recognition in 2014. He received best paper awards from ACM Multimedia (ACM MM) (Best Paper and Best Student Paper) in 2013, ACM MM (Best Demo) in 2012, the Pacific-Rim Conference on Multimedia in 2011, ACM MM in 2010, the International Conference on Multimedia and Expo in 2010, and the International Conference on Internet Multimedia Computing and Service in 2009, the Runner-Up Prize in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2013, the winner prize of the detection task in ILSVRC in 2014, the winner prizes of the classification task in PASCAL VOC from 2010 to 2012, the winner prize of the segmentation task in PASCAL VOC in 2012, the honorable mention prize of the detection task in PASCAL VOC in 2010, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY Best Associate Editor Award in 2010, the Young Faculty Research Award in 2010, the Singapore Young Scientist Award in 2011, and the NUS Young Researcher Award in 2012.