Self-paced Convolutional Neural Networks

Hao Li, Maoguo Gong*
Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education
Xidian University, Xi'an, China
omegalihao@gmail.com, gong@ieee.org

Abstract

Convolutional neural networks (CNNs) have achieved breakthrough performance in many pattern recognition tasks. In order to distinguish the reliable data from the noisy and confusing data, we improve CNNs with self-paced learning (SPL) for enhancing the learning robustness of CNNs. In the proposed self-paced convolutional network (SPCN), each sample is assigned to a weight to reflect the easiness of the sample. Then a dynamic self-paced function is incorporated into the leaning objective of CNN to jointly learn the parameters of CNN and the latent weight variable. SPCN learns the samples from easy to complex and the sample weights can dynamically control the learning rates for converging to better values. To gain more insights of SPCN, theoretical studies are conducted to show that SPCN converges to a stationary solution and is robust to the noisy and confusing data. Experimental results on MNIST and *rectangles* datasets demonstrate that the proposed method outperforms baseline methods.

1 Introduction

In recent years, convolutional neural networks (CNNs) [LeCun *et al.*, 1998] have attracted widespread interests in the field of machine learning and computer vision. CNN is a supervised learning algorithm that aims to learn the mapping between the input data and its corresponding label. In general, CNN consists of three main components: convolution kernels, active function and pooling. Convolution kernels work on small local receptive fields of input data in a sliding-window fashion and then the active function is imposed on the convolution kernel output. Each kernel can be considered as a specific feature detector. The pooling can reduce the data size to make the final prediction be robust to the translation of input data. The parameters of CNN, i.e., the coefficients of convolution kernels and the final fully connected layer, are learnt by using stochastic gradient descent with the standard back-propagation algorithm. The success of CNNs brought about a revolution through the efficient use of graphics processing units (GPUs), rectified linear units, new dropout regularization, and effective data augmentation [LeCun *et al.*, 2015]. Convolutional networks have been successfully applied to various applications, such as image and video classification [Krizhevsky *et al.*, 2012; Karpathy *et al.*, 2014], face recognition [Taigman *et al.*, 2014] and objective detection [Girshick *et al.*, 2014].

The goal of robust learning is to reduce the influence of the noisy and confusing data. The learning robustness relies on a sample selection to distinguish the reliable samples from the confusing ones [Pi *et al.*, 2016]. The recently proposed *self-paced learning* (SPL) is such a representative method for robust learning. SPL can trace back to *curriculum learning* (CL) [Bengio *et al.*, 2009] proposed by Bengio *et al.*, in which a curriculum defines a set of training samples organized in ascending order of learning difficulty. However, in CL, the curriculum remains unchanged during the iterations and is determined independently of the subsequent learning [Bengio *et al.*, 2009; Jiang *et al.*, 2015]. Then self-paced learning [Kumar *et al.*, 2010] have been proposed to dynamically generate the curriculum according to what the learner has already learned. SPL is inspired by the learning process of humans/animals, which learns with easier concepts at first and then gradually involves more complex ones into training. Both CL and SPL have been proved to be beneficial in avoiding bad local minima and in achieving a better generalization result [Khan *et al.*, 2011; Basu and Christensen, 2013; Tang *et al.*, 2012b].

Based on the above analysis, we notice that convolutional neural networks and self-paced learning are consistent and complementary. For consistency, both convolutional networks and self-paced learning are biologically-inspired. In fact, CNN is nicely related to the self-paced learning idea that it may be much easier to learn simpler concepts first and then build higher level ones on top of simpler ones [Bengio *et al.*, 2013]. Furthermore, convolutional neural networks and self-paced learning are complementary because CNNs aim to extract features and self-paced learning tends to explore the data smoothly with more robustness. Therefore the two schemes can benefit from each other.

In this paper, a *self-paced convolutional network* (SPCN) is proposed to enhance the learning robustness of CNNs. In the proposed SPCN model, each sample is assigned to a weight to reflect the easiness of the sample. Then a convolution-

*Corresponding author

al network is established to learn the weighted samples. In SPCN, the weighting and minimization are integrated in a single function by incorporating a novel dynamic self-paced function into the learning objective of CNN. SPCN iteratively updates the model parameters and the weight variable. The learning rates of SPCN are dynamically changed based on the current sample weights. Theoretical studies show that SPCN converges to a stationary solution and is robust to the noisy and confusing data. Experimental results on MNIST and *rectangles* datasets demonstrate the effectiveness and robustness of the proposed method.

The contribution of this paper are summarized as follows: (1) We improve CNNs with self-paced learning for enhancing the learning robustness of CNNs. (2) A dynamic self-paced function is proposed for the proposed SPCN model. (3) We conduct theoretical studies to show that SPCN converges to a stationary solution and is robust to the noisy and confusing data.

2 Self-paced Learning

Compared to the other machine learning methods, SPL incorporates a self-paced function and a pace parameter into the learning objectives to jointly learn the curriculum and model parameters. Formally, we denote the training dataset as $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ denotes the ith observed sample, and y_i represents its label. Let $L(y_i, g(\mathbf{x}_i, \mathbf{w}))$ denote the loss function which calculates the cost between the ground truth label y_i and the estimated label $g(\mathbf{x}_i, \mathbf{w})$. Here $\mathbf{w}$ represents the model parameter inside the decision function g. In SPL, variable $\mathbf{v}$ is introduced into the learning objective to indicate whether the ith sample is easy or not. The target of SPL is to jointly learn the model parameter $\mathbf{w}$ and the latent weight variable $\mathbf{v} = [v_1, \cdots, v_n]$ by minimizing:

$$\min_{\mathbf{w},\mathbf{v}} \mathbb{E}(\mathbf{w}, \mathbf{v}) = \sum_{i=1}^{n} v_i L(y_i, g(\mathbf{x}_i, \mathbf{w})) + \lambda^t f(\mathbf{v}), \qquad (1)$$

where $\mathbf{v} \in [0, 1]^n$ and $f(\mathbf{v})$ is the self-paced function. The parameter λ is the "age" of the SPL model to control the learning pace. In the process of the SPL calculation, we gradually increase λ to learn new samples. When λ is small, only "easy" samples with small losses will be considered into training. As λ grows, more samples with larger losses will be gradually appended to training a more "mature" model.

The algorithm of SPL is shown in **Algorithm 1**. In the inner loop, the model parameter $\mathbf{w}$ and the weight variable $\mathbf{v}$ are iteratively updated with a fixed pace parameter λ. Then the pace parameter λ increases in the outer loop to progressively approach the rational solution. Kumar *et al.* proposed a hard weighting scheme [Kumar *et al.*, 2010] to indicate whether the sample is easy or not. Then Jiang *et al.* proposed three soft weighting methods [Jiang *et al.*, 2014a] to assign real-valued weights to reflect the importance of samples, such as the linear soft weighting, logarithmic soft weighting and mixture weighting schemes. Then several useful sample importance priors have been incorporated into the SPL model, such as spatial smoothness [Zhang *et al.*, 2015], partial order [Jiang *et al.*, 2015] and diversity [Jiang *et al.*, 2014b]. In [Meng and Zhao, 2015], Meng *et al.* have proved that the solving strategy on SPL exactly accords with a majorization minimization algorithm implemented on a latent objective and the loss function contained in the latent objective has a similar configuration with non-convex regularized penalty. SPL has been successfully applied to various applications, such as segmentation [Kumar *et al.*, 2011], domain adaption [Tang *et al.*, 2012a], dictionary leaning [Tang *et al.*, 2012b], long-term tracking [Supančič and Ramanan, 2013], reranking [Jiang *et al.*, 2014a], multi-view learning [Xu *et al.*, 2015], action and event detection [Jiang *et al.*, 2014b; 2015; Li *et al.*, 2016], matrix factorization [Zhao *et al.*, 2015; Li *et al.*, 2016], and co-saliency detection [Zhang *et al.*, 2015].

Algorithm 1 Algorithm of Self-paced Learning.

Input: The training dataset $\mathcal{D}$, the initial step λ, the step size μ.
Output: Model parameter $\mathbf{w}$.

```
Initialize w*.
while not converged do //Outer Loop
    while not converged do //Inner Loop
        Update v* = arg min_v E(w*, v).
        Update w* = arg min_w E(w, v*).
    end while
    λ ← λ + μ.
end while
return w = w*.
```

3 Self-paced Convolutional Networks

3.1 SPCN Model

The cost function of CNNs can be expressed as

$$E_{CN}(\mathbf{w}) = \sum_{i=1}^{n} L(y_i, g(\mathbf{x}_i, \mathbf{w})). \qquad (2)$$

When the softmax layer is the last layer of a CNN, the cross entropy between the softmax activations $\alpha_1, \cdots, \alpha_K$ of the output neurons and the binary target vector $\mathbf{y}$ is $-\sum_{k=1}^{K} y_k \ln(\alpha_k)$. In this paper, we propose a *self-paced convolutional network* (SPCN) to enhance the learning robustness. Figure 1 presents the schematic diagram of the proposed SPCN model. Each sample is assigned to a weight to reflect the easiness of the sample. Then we incorporate a self-paced function into the learning objective of CNNs to jointly learn the model parameters and the latent weight variable. The proposed SPCN model is shown as follows:

$$\min_{\mathbf{w},\mathbf{v}} \mathbb{E}(\mathbf{w}, \mathbf{v}) = \sum_{i=1}^{n} v_i L_i^t + \lambda^t f(\mathbf{v}; t), \qquad (3)$$

where $f(\mathbf{v}; t)$ is a dynamic self-paced function with respect to $\mathbf{v}$ and t.

In this paper, we use a *majorization minimization* algorithm to solve the proposed SPCN model. MM algorithms have been widely used in machine learning and aim to convert a

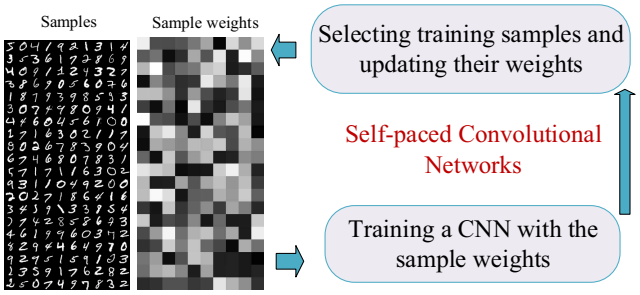


Figure 1: The framework of the proposed self-paced convolutional networks.

complicated optimization problem into an easy one by alternatively iterating the majorization step and the minimization step. We can get the integrative function of $v^*(\lambda, L)$ as

$$F_\lambda(L) = \int_0^L \mathbf{v}^*(\lambda; L)dL. \tag{4}$$

We use $Q_\lambda(\mathbf{w}|\mathbf{w}^*)$ to represent a tractable surrogate for $F_\lambda(L(\mathbf{w}))$. In [Meng and Zhao, 2015], Meng *et al.* have proved that

$$\begin{aligned} Q_\lambda^{(i)}(\mathbf{w}|\mathbf{w}^*) &= F_\lambda(L_i(\mathbf{w}^*)) \\ &+v_i^*(\lambda; L_i(\mathbf{w}^*))(L_i(\mathbf{w} - L_i(\mathbf{w}^*), \end{aligned} \tag{5}$$

and

$$\sum_{i=1}^n F_\lambda(L_i(\mathbf{w})) \leq \sum_{i=1}^n Q_\lambda^{(i)}(\mathbf{w}|\mathbf{w}^*). \tag{6}$$

In this paper, we denote $\mathbf{w}^t$ as the model parameters in the tth iteration of the MM algorithm.

Majorization Step

We can calculate $v_i^*(\lambda; L_n(\mathbf{w}^t))$ by solving the following problem to obtain each $Q_\lambda^{(i)}(\mathbf{w}|\mathbf{w}^t)$

$$v_i^*(\lambda; L_i(\mathbf{w}^t)) = \arg\min_{v_i \in [0,1]} v_i L_i(\mathbf{w}^t) + \lambda^t f(v_i; t). \tag{7}$$

The self-paced function needs to be specified for solving $v_i^*(\lambda; L_i(\mathbf{w}^t))$ in Eq. (7). In recent years, several efficient S-PL functions have been proposed for various data with different characteristics, such as linear soft weighting, logarithmic soft weighting and mixture weighting [Jiang *et al.*, 2014a]. However, all these methods assign low weights to the easy samples in the early stage of self-paced learning. To address this issue, we propose a dynamic self-paced function for reasonably assigning weights. The proposed dynamic self-paced function $f(\mathbf{v}; t)$ is described as follows:

$$f(\mathbf{v}; t) = \frac{1}{q(t)}\|\mathbf{v}\|_2^{q(t)} - \sum_{i=1}^n v_i, \tag{8}$$

where $q(t) > 1$, which is a monotonic decreasing function with respect to the time t.

In order to analysis the proposed dynamic self-paced function, we deduce the closed-form solutions of the SPCN model under the proposed dynamic self-paced function. Eq. (8) is a convex function of $\mathbf{v}$ in [0,1] and thus the global minimum can be obtained at $\nabla_\mathbf{v}\mathbb{E}(\mathbf{v}) = \mathbf{0}$. Therefore we have

$$\frac{\partial \mathbb{E}}{\partial v_i} = L_i + \lambda^t(v_i^{q(t)-1} - 1) = 0. \tag{9}$$

The closed-form optimal solution for $v_i (i = 1, 2, \cdots, n)$ can be written as:

$$v_i^* = \begin{cases} (1 - \frac{L_i}{\lambda^t})^{(1/(q(t)-1))} & L_i < \lambda^t \\ 0 & L_i \geq \lambda^t. \end{cases} \tag{10}$$

We can also deduce the closed-form solutions of the SPCN model under the hard, linear soft, logarithmic soft and mixture weighting functions. Then we can graph the curves of the sample weight with respect to the sample loss based on the obtained closed-form solutions. Figure 2 shows the comparison of the hard, linear soft, logarithmic soft, mixture weighting and dynamic self-paced functions. Obviously, when $q(t)$ is smaller than 2, the proposed self-paced function is similar to the logarithmic soft weighting method. When $q(t)$ is equal to 2, the proposed self-paced function is to linearly discriminate samples with respect to their losses, which is equivalent to the linear soft weighting method. When $q(t)$ is larger than 2, the proposed self-paced function obtains higher weight values when the loss is low. Therefore both the proposed method and mixture weighting scheme can tolerate small errors.

The proposed SPCN model has some important properties. First, at the tth iteration (or the time t), $q(t)$ can be considered as a constant. The second derivative of Eq. (8) is

$$\frac{\partial^2 f}{\partial^2 v_i} = \lambda(q(t) - 1)v_i^{q(t)-2} \geq 0. \tag{11}$$

Therefore $f(\mathbf{v}; t)$ is convex with respect to $\mathbf{v} \in [0, 1]^n$ because the above second derivatives are non-negative and the sum of convex functions is convex. Second, the optimal solution $\mathbf{v}^*$ is shown in Eq. (10). Obviously, v_i is decreasing with respect to L_i and we have that $\lim_{L_i \to 0} v_i^* = 1$, $\lim_{L_i \to \infty} v_i^* = 0$. It indicates that the SPCN model favors easy samples because the easy samples have lower loss values and larger weights. Finally, each individual v_i^* increases with respect to λ^t in the closed-form solution in Eq. (10). In an extreme case, when λ^t approaches positive infinity, we have $\lim_{\lambda^t \to \infty} v_i^* = 1$. Similarly, when λ^t approaches 0, we have $\lim_{\lambda^t \to 0} v_i^* = 0$. When the model "age" gets larger, it trends to incorporate more samples into training.

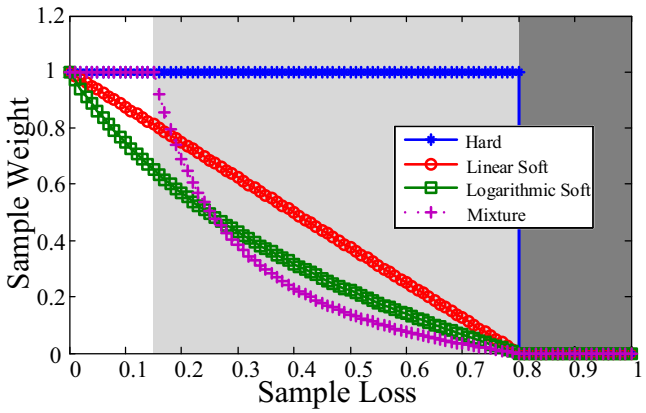
Minimization Step

In this step, we need to calculate

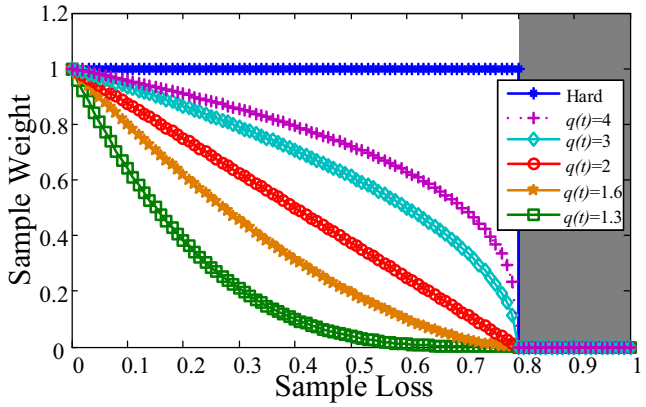
$$\mathbf{w}^{t+1} = \arg\min_\mathbf{w} \sum_{i=1}^n v_i L_i(\mathbf{w}^t). \tag{12}$$

Then the cost function of SPCN is

$$E_{SPCN}(\mathbf{w}) = \sum_{i=1}^n v_i L(y_i, g(\mathbf{x}_i, \mathbf{w})). \tag{13}$$



(a)



(b)

Figure 2: The comparison of different self-paced functions. (a) is the comparison of hard, linear soft, logarithmic soft and mixture weighting schemes. (b) represents the comparison of the proposed dynamic self-paced function with $q(t)$ = 1.3, 1.6, 2, 3 and 4.

E_{CN} and E_{SPCN} are the error functions of the standard convolutional networks and the proposed SPCN model, respectively. For a single pattern, the two error functions have the following relationship

$$\begin{aligned} \Delta \mathbf{w} &= -\eta \frac{\partial E^i_{SPCN}}{\partial \mathbf{w}} \\ &= -v_i \cdot \eta \cdot \frac{\partial E^i_{CN}}{\partial \mathbf{w}} \end{aligned} \tag{14}$$

where $\mathbf{w}$ is the weights in a certain layer of the network, and η is the learning rate. It is obvious that the sample weight v_i can control the learning rate and the sample weight v_i are constrained in [0,1]. When the samples are easy, the learning rates are high values and the network can update the parameters in a large step. The samples have small learning rates when the samples are difficult. With the small learning rates, the network is able to update the parameters slowly for converging better values.

3.2 SPCN Implementation

It is difficult to decide the increasing pace λ^t [Jiang *et al.*, 2014a]. In general, we need the range of loss values in advance and give the initial value and the step size. A robust way is to search the solution path with respect to the sample number involved into training instead of extracting the solution path with respect to the pace parameter. Therefore we can predefine a sample number sequence $\mathbf{N} = \{N_1, N_2, \cdots, N_{maxgen}\}$ ($N_i < N_j$ for $i < j$) representing the number of selected samples in self-paced learning process. Each N_t denotes how many samples will be selected in the tth SPL stage, and $N_{maxgen} = n$ means finally all samples are chosen into training. For tth iteration, we can get $Lsort$ by sorting the loss L in ascending order and select the N_tth loss value of $Lsort$ as the values of λ^t, i.e., λ^t can be represented as

$$\lambda^t = Lsort_{N_t} \tag{15}$$

where $Lsort$ is obtained by sorting the loss L in ascending order.

Algorithm 2 Algorithm of Self-paced Convolutional Networks.

Input: The training dataset $\mathcal{D}$.
Output: Model parameter $\mathbf{w}$.

```
Initialize w*, and predefine the pace sequence N = {N_1, N_2, ..., N_maxgen}.
for t=1 to maxgen do //Outer Loop
    Calculate λ^t by Eq. (15) and q(t) by Eq. (16).
    while not converged do //Inner Loop
        Update v* = arg min_v E(w*, v).
        Update w* = arg min_w E(w, v*).
    end while
end for
return w = w*.
```

The function $q(t)$ controls the selection of the learning schemes. In the early stage of SPCN, only simple samples are involved into training and their weights should be large. As shown in Figure 2 (b), the self-paced function with large $q(t)$ works better in this stage. During iterations, more complex samples are taken into consideration. Therefore $q(t)$ should be set to a small value. To implement the SPCN model, the solution path is searched with respect to the sample number. Then $q(t)$ can be also defined with respect to the sample number, which is shown as follows

$$q(t) = 2\tan((1 - \frac{N_t}{N_{maxgen}+1}) \cdot \frac{\pi}{2}). \tag{16}$$

The algorithm of SPCN is shown in **Algorithm 2**. In the initialization, a sample number sequence $\mathbf{N} = \{N_1, N_2, \cdots, N_{maxgen}\}$ is predefined. In the sequence, we make $N_i < N_j$ for $i < j$ to simulate the increasing process of the pace parameter. Then we obtain λ^t and $q(t)$ by Eq. (15) and Eq. (16) in the outer loop, respectively. The model parameter $\mathbf{w}$ and the weight variable $\mathbf{v}$ are iteratively updated with the fixed λ^t and $q(t)$ in the inner loop.

4 Theoretical Analysis

In this section, we analyze the convergence and properties of the proposed method. Let $\mathbf{w}^t$ and $\mathbf{v}^t$ indicate the

values of $\mathbf{w}$ and $\mathbf{v}$ in the tth iteration. As $\min_{\mathbf{w}} \mathbb{E}(\mathbf{w}, \mathbf{v})$ is a quadratic programming problem [Jiang *et al.*, 2014a; 2014b], the solution $\mathbf{w}^t$ is the global optimum, i.e.

$$\mathbb{E}(\mathbf{w}^t, \mathbf{v}^{t-1}) \leq \mathbb{E}(\mathbf{w}^{t-1}, \mathbf{v}^{t-1}). \tag{17}$$

$f(\mathbf{v}, t)$ is a convex function of $\mathbf{v}$ at the time t. Therefore Eq. (3) is a convex funtion of $\mathbf{v}$. Suppose that $\mathbf{v}^t$ is a solution found by gradient descent, due to the convexity, $\mathbf{v}^t$ is the global optimum for Eq. (3), i.e.

$$\mathbb{E}(\mathbf{w}^t, \mathbf{v}^t) \leq \mathbb{E}(\mathbf{w}^t, \mathbf{v}^{t-1}). \tag{18}$$

Substitute Eq. (18) back into Eq. (17), we have

$$\mathbb{E}(\mathbf{w}^t, \mathbf{v}^t) \leq \mathbb{E}(\mathbf{w}^{t-1}, \mathbf{v}^{t-1}). \tag{19}$$

Obviously, the objective values decrease in every iteration in SPCN algorithm. The algorithm is bounded from below because $\mathbb{E}$ is the sum of finite elements. Therefore **Algorithm 2** can converge to a stationary solution.

The solving strategy on SPL exactly accords with a majorization minimization algorithm implemented on a latent objective and the loss function contained in this latent objective has a similar configuration with non-convex regularized penalty [Meng and Zhao, 2015]. In SPCN, we can obtain the solution $\mathbf{v}^*$ as follows:

$$\mathbf{v}^*(\lambda^t; L) = \arg\min_{\mathbf{v}} \mathbf{v}L + \lambda^t f(\mathbf{v}, t). \tag{20}$$

We can get the integrative function of $\mathbf{v}^*(\lambda; L)$ calculated by Eq. (20) as:

$$F_{\lambda^t}(L) = \int_0^L \mathbf{v}^*(\lambda^t; L)dL + c, \tag{21}$$

where c is a constant.

Now we try to discover more interesting insight under the proposed SPCN method from this latent objective. To this aim, we first calculate the latent losses under the proposed dynamic self-paced function (8) by Eq. (21) as follows:

$$F_{\lambda^t}(L) = \begin{cases} -\frac{(q(t)-1)\lambda^t}{q(t)}(1 - \frac{L}{\lambda^t})^{\frac{q(t)}{q(t)-1}} + c & L < \lambda^t \\ c & L \geq \lambda^t. \end{cases} \tag{22}$$

Note that when $\lambda^t = \infty$, the latent loss $F_{\lambda^t}(L)$ will degenerate to the original loss L. There is an evident suppressing effect of $F_{\lambda^t}(L)$ on large losses as compared with the original loss function L. When L is larger than a certain threshold, $F_{\lambda^t}(L)$ will become a constant thereafter. This provides a rational explanation on why SPCN can perform robust in the presence of noisy and confusing data: The samples with loss values larger than the age threshold will have no influence to the model training due to their 0 gradients. Corresponding to the original SPL model, these large loss samples will be with 0 importance weights v_i, and thus have no effect on the optimization of model parameters.

5 Experiments

In order to evaluate the performance of the proposed SPCN, we studied variants of MNIST digits [Larochelle *et al.*, 2007] and the *rectangle* datasets [Larochelle *et al.*, 2007] in the

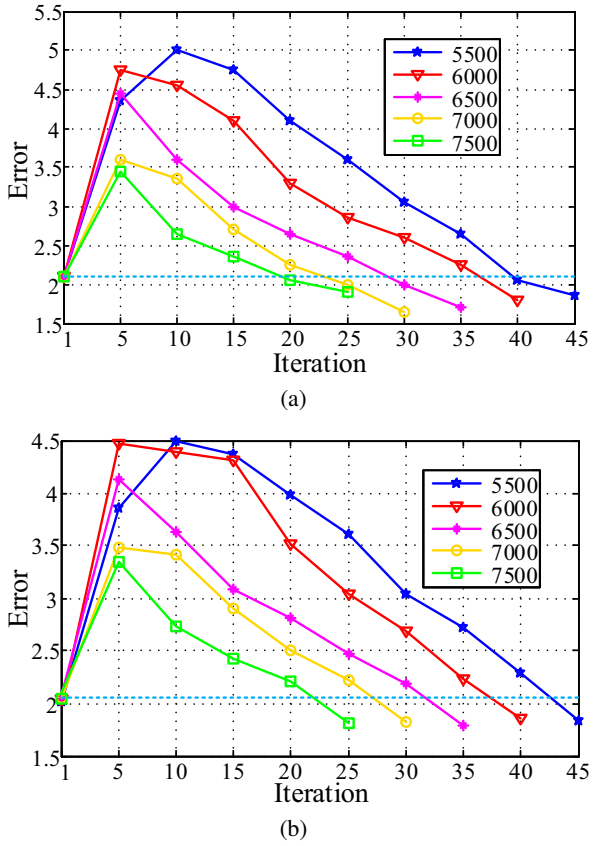


Figure 3: The results of SPCN with different pace sequence on the *mnist-basic* dataset. The initial value of the pace sequence N_1 is set to 5500, 6000, 6500, 7000 and 7500. (a) shows the results of the *validation-set*. (b) presents the results of the *test-set*.

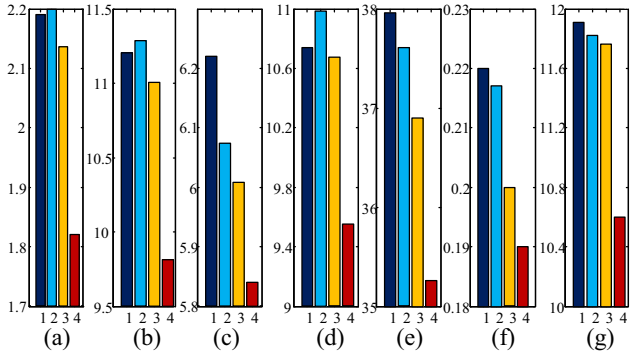


Figure 4: The results of SPCN under 1: linear soft weighting, 2: logarithmic soft weighting, 3: mixture weighting and 4: dynamic self-paced function.

experiments. The first database consists of five variants of MNIST digits, i.e. {*mnist-basic, mnist-rot, mnist-back-rand, mnist-back-image, mnist-rot-back-image*}. Each variant includes 10,000 labeled training, 2,000 labeled validation, and 50,000 labeled test images. The second database includes two subsets, i.e. {*rectangle, rectangle-image*}. The dataset *rect-*

Table 1: Comparison of self-paced convolutional network (SPCN) with other models. The best performer is marked in bold.

Datasets	mnist-basic	mnist-rot	mnist-back-rand	mnist-back-image	mnist-rot-back-rand	rectangles	rectangles-image
SVMrbf	3.03±0.15	11.11±0.28	14.58±0.31	22.61±0.37	55.18±0.44	2.15±0.13	24.04±0.37
SVMpoly	3.69±0.17	15.42±0.32	16.62±0.33	24.01±0.37	56.41±0.43	2.15±0.13	24.05±0.37
NNet	4.69±0.19	18.11±0.34	20.04±0.35	27.41±0.39	62.16±0.43	7.16±0.23	33.20±0.41
DBN-1	3.94±0.17	14.69±0.31	9.80±0.26	16.15±0.32	52.21±0.44	4.71±0.19	23.69±0.37
DBN-3	3.11±0.15	10.30±0.27	6.73±0.22	16.31±0.32	47.39±0.44	2.60±0.14	22.50±0.37
SAA-3	3.46±0.16	10.30±0.27	11.28±0.28	23.00±0.37	51.93±0.44	2.41±0.13	24.05±0.37
SdA	2.80±0.14	10.29±0.27	10.38±0.27	16.68±0.33	44.49±0.44	1.99±0.12	21.59±0.36
CAE-1	2.83±0.15	11.59±0.28	13.57±0.30	16.70±0.33	48.10±0.44	1.48±0.10	21.86±0.36
CAE-2	2.48±0.14	9.66±0.26	10.90±0.27	15.50±0.32	45.23±0.44	1.21±0.10	21.54±0.36
GSN	2.40±0.04	**8.66±0.08**	9.38±0.03	16.04±0.07	43.86±0.05	2.04±0.04	22.10±0.03
SPCN	**1.82±0.04**	9.81±0.07	**5.84±0.03**	**9.55±0.06**	**35.26±0.05**	**0.19±0.03**	**10.60±0.03**

angle consists of 1000 labeled training, 200 labeled validation, and 50,000 labeled test images. The dataset *rectangle-image* includes 10,000 labeled train, 2000 labeled validation and 50,000 labeled test images.

5.1 Analysis of the Pace Sequence

Concerning the influence of the pace sequence, we tested the initial value of the pace sequence in the range $N_1 \in \{5500, 6000, 6500, 7000, 7500\}$. The step size of the pace sequence is same to the batch size. In this experiment, all training samples are exploited to train a CNN and then this trained CNN is used to initialize the proposed SPCN. Figure 3 shows the results of SPCN with different pace sequence on the *mnist-basic* dataset. As shown in Figure 3, the errors of SPCN with different pace sequences are much lower than these of the initial CNN on the *validation* and *test* sets. When the initial value of the pace sequence is small, SPCN needs many iterations to involve all samples into training. We can always obtain satisfactory results compared to the original CNN.

5.2 Analysis of the Dynamic Self-paced Function

In this experiment, we compared the proposed dynamic self-paced function with the linear soft weighting, logarithmic soft weighting and mixture weighting schemes [Jiang *et al.*, 2014a]. Figure 4 shows the results of SPCN under the four self-paced functions. It can be observed that the errors obtained by SPCN under the proposed dynamic self-paced function are much lower than those of competing methods on seven variants of MNIST and *rectangles* datasets. In the early stage of self-paced learning, the involved samples tend to be easy and should be given large weights. However, the previous self-paced functions assigned low weights to the easy samples. In the proposed dynamic self-paced function, we dynamically select suitable learning curve shown in Figure 2(b). The proposed dynamic self-paced function gives large weights to the easy samples in the early stage and can reduce the influence of the complex samples during iterations.

5.3 Experimental Results

In this experiment, we compare the results provided by the proposed SPCN model with those obtained by SVM-rbf, SVMpoly, NNet, DBN-1, DBN-3, SAA-3 [Larochelle *et al.*, 2007], SDA [Vincent *et al.*, 2008], CAE-1, CAE-2 [Rifai *et al.*, 2011] and GSN [Zöhrer and Pernkopf, 2014]. Table 1 shows that the proposed method outperforms baseline methods on six out of seven variants on MNIST and *rectangles* datasets. The proposed SPCN model ranks first except for the *mnist-rot* datasets. These statistics indicate that our approach can achieve the higher accuracy and better stability in the statistical sense when compared with the other competing methods.

6 Conclusion

In order to distinguish the reliable data from the noisy and confusing data, we have proposed a self-paced convolutional neural network model. In the proposed SPCN model, the loss of a sample is discounted by a weight. Then a dynamic self-paced function is incorporated the learning objective of CNNs. SPCN uses a *majorization minimization* algorithm to jointly learn the model parameters of CNN and the laten variable. In the majorization step, SPCN selects the reliable samples based on the acquired model parameters. In the minimization step, SPCN updates the model parameters with the sample weights. The leaning rates are dynamically changed based on the current sample weights during iterations. The complex samples always have small learning rates to get better results.

Our theoretical analysis shows that SPCN can converge to a stationary solution and SPCN can perform robust in the presence of noisy and confusing data. Furthermore, we also achieved state-of-the-art performance on variants of MNIST digits and the *rectangle* datasets. The proposed method can enhance the learning robustness of convolutional neural networks. In future research, we will explore other deep architectures with self-paced learning.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61422209, 61672409), and the National Program for Support of Top-notch Young Professionals of China.

References

[Basu and Christensen, 2013] Sumit Basu and Janara Christensen. Teaching classification boundaries to humans. In *AAAI*, pages 109–115, 2013.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.

[Bengio *et al.*, 2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.

[Jiang *et al.*, 2014a] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, pages 547–556, 2014.

[Jiang *et al.*, 2014b] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *NIPS*, pages 2078–2086, 2014.

[Jiang *et al.*, 2015] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, pages 2694–2700, 2015.

[Karpathy *et al.*, 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.

[Khan *et al.*, 2011] Faisal Khan, Bilge Mutlu, and Xiaojin Zhu. How do humans teach: On curriculum learning and teaching dimension. In *NIPS*, pages 1449–1457, 2011.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Kumar *et al.*, 2010] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *NIPS*, pages 1189–1197, 2010.

[Kumar *et al.*, 2011] M Pawan Kumar, Haithem Turki, Dan Preston, and Daphne Koller. Learning specific-class segmentation from diverse data. In *ICCV*, pages 1800–1807, 2011.

[Larochelle *et al.*, 2007] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, pages 473–480, 2007.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[Li *et al.*, 2016] Hao Li, Maoguo Gong, Deyu Meng, and Qiguang Miao. Multi-objective self-paced learning. In *AAAI*, pages 1802–1808, 2016.

[Meng and Zhao, 2015] Deyu Meng and Qian Zhao. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015.

[Pi *et al.*, 2016] Te Pi, Xi Li, Zhongfei Zhang, Deyu Meng, Fei Wu, Jun Xiao, and Yueting Zhuang. Self-paced boost learning for classification. In *IJCAI*, pages 1932–1938, 2016.

[Rifai *et al.*, 2011] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, pages 833–840, 2011.

[Supančič and Ramanan, 2013] James Steven Supančič and Deva Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013.

[Taigman *et al.*, 2014] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[Tang *et al.*, 2012a] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In *NIPS*, pages 638–646, 2012.

[Tang *et al.*, 2012b] Ye Tang, Yu-Bin Yang, and Yang Gao. Self-paced dictionary learning for image classification. In *ACM MM*, pages 833–836, 2012.

[Vincent *et al.*, 2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[Xu *et al.*, 2015] Chang Xu, Dacheng Tao, and Chao Xu. Multi-view self-paced learning for clustering. In *IJCAI*, pages 3974–3980, 2015.

[Zhang *et al.*, 2015] Dingwen Zhang, Deyu Meng, Chao Li, Lu Jiang, Qian Zhao, and Junwei Han. A self-paced multiple-instance learning framework for co-saliency detection. In *ICCV*, pages 594–602, 2015.

[Zhao *et al.*, 2015] Qian Zhao, Deyu Meng, Lu Jiang, Qi Xie, Zongben Xu, and Alexander G Hauptmann. Self-paced learning for matrix factorization. In *AAAI*, pages 3196–3202, 2015.

[Zöhrer and Pernkopf, 2014] Matthias Zöhrer and Franz Pernkopf. General stochastic networks for classification. In *NIPS*, pages 2015–2023, 2014.