Robust Classification with Convolutional Prototype Learning

Hong-Ming Yang[1,2], Xu-Yao Zhang[1,2], Fei Yin[1,2], Cheng-Lin Liu[1,2,3]

[1]NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China

[2]University of Chinese Academy of Sciences, Beijing, P.R. China

[3]CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing, P.R. China

{hongming.yang, xyz, fyin, liucl}@nlpr.ia.ac.cn

Abstract

Convolutional neural networks (CNNs) have been widely used for image classification. Despite its high accuracies, CNN has been shown to be easily fooled by some adversarial examples, indicating that CNN is not robust enough for pattern classification. In this paper, we argue that the lack of robustness for CNN is caused by the softmax layer, which is a totally discriminative model and based on the assumption of closed world (i.e., with a fixed number of categories). To improve the robustness, we propose a novel learning framework called convolutional prototype learning (CPL). The advantage of using prototypes is that it can well handle the open world recognition problem and therefore improve the robustness. Under the framework of C-PL, we design multiple classification criteria to train the network. Moreover, a prototype loss (PL) is proposed as a regularization to improve the intra-class compactness of the feature representation, which can be viewed as a generative model based on the Gaussian assumption of different classes. Experiments on several datasets demonstrate that CPL can achieve comparable or even better results than traditional CNN, and from the robustness perspective, C-PL shows great advantages for both the rejection and incremental category learning tasks.

1. Introduction

In recent years, convolutional neural networks [16] (C-NNs) have achieved great success for pattern recognition and computer vision, leading to important progress in a variety of tasks, like image classification [15, 8, 7, 34], object detection [1, 29, 9], instance segmentation [28, 6] and so on.

Despite the success of CNN, it still suffer from some serious problems. One example is the existence of adversarial samples [37], when we add small noises or make some small changes to the initial samples, CNN will give different predictions for these samples with high confidence, al-
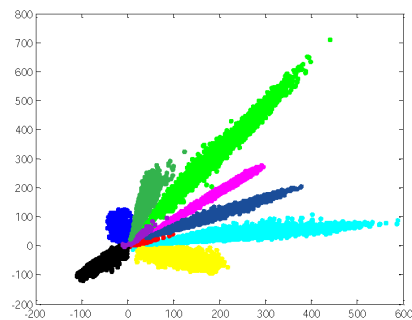

Figure 1. Feature representation learned by traditional CNN model on MNIST. Different colors represent different classes. It is shown that the inter-class variation is even smaller than the intra-class variation.

though visually we can hardly find any significant changes in the images. Another example is the rejection ability of CNN, when feed a sample from an unseen class to CN-N, it will still allocate the sample to a known class with high confidence. These two phenomena indicate that CNN is not robust, though it can achieve human-level or even better accuracy on some specific datasets, its performance will degenerate obviously in the complex scenes of the reality. This greatly limits the application of CNN in real worlds. The main reasons for these problems include two aspects: First, CNN is a purely discriminative model, it essentially learns a partition of the "whole" feature space, therefore, the samples from unseen classes are still predicted to some specific regions under the partition, and CNN still views these samples as some known classes with high confidence. This explains why the rejection ability of CNN is poor; Second, from the perspective of representation learning, the learned representation of CNN is linear separable, see Fig. 1 for an illustration, and under this kind of representation, the inter-class distance is sometimes even smaller than the intra-class distance, this significantly reduces the robustness of CNN in real and complicated environments.

Several methods have been proposed to improve the ro-

bustness of CNN and most of them concentrate on designing better loss functions. [36] and [33] proposed the contrastive loss and triplet loss to learn a more robust feature representation, in which the input pairs and triplets need to be carefully selected from the training data to ensure the convergence and stability. [38] proposed the center loss to improve the performance of softmax-based CNN, however, the centers can not be learned jointly with the CNN and are only updated according to some pre-defined rules rather than learned directly from data. Our CPL is more general than the center loss, since we totally abandon softmax layer and all the prototypes are learned automatically from data. Moreover, previous work of [26] and [25] also made improvements and extensions for the softmax based loss but they still kept the softmax layer with the traditional framework of CNN for classification.

In this paper, we propose a novel framework called convolutional prototype learning (CPL) for image classification. In the bottom of CPL, the convolutional layers are used to extract discriminative features just like traditional CNN, but in the top of CPL we assign multiple prototypes to represent different classes. The classification is simply implemented by finding the nearest prototype (using Euclidean distance) in the feature space. We design multiple loss functions for this framework, making the CNN feature extractor and the prototypes being learned jointly from the raw data. Therefore, the whole framework can be trained efficiently and effectively. Experiments on several datasets demonstrate that the CPL framework can achieve comparable or even better classification accuracies compared with traditional CNN models.

Benefited from the prototype-based decision function, a natural prototype loss (PL) can be added to our CPL framework, to pull the feature vector closer to their corresponding prototypes (genuine class representation). The PL is akin to the maximum likelihood (ML) regularization proposed in [23, 22]. On one hand, it acts like a regularizer, which can prevents the model form over-fitting and improves the performance of CPL. On the other hand, it can also improves the intra-class compactness in feature representation. Therefore, the final learned representation is intra-class compact and inter-class separable, which makes the representation more discriminative and robust. From the perspective of probability, our CPL and PL framework essentially extract, transform, and model the data of each class as a Gaussian mixture distribution and the prototypes act as the means of Gaussian components for each class, this enables integrating probabilistic methods such as Bayesian models into our framework. Compared with the traditional CNN framework, we do not make partition for the "whole" feature space, but project the samples to some specific regions of the feature space (near the prototypes), thus our model is more robust to samples from unseen classes and

more suitable for rejection. CPL can also be viewed as a hybrid discriminative and generative model (like the discriminative density model in [23]) which will lead to better generalization performance.

2. Related works

In this section, we describe related works from two aspects: robust representation learning and prototype learning. Most recent methods concentrate on modifying or proposing new loss functions to learn discriminative and robust representations. Among these methods, [36] combined the cross entropy loss and contrastive loss [5] to train the CNN, the cross entropy loss can increase the inter-personal variations while the contrastive loss can reduce the intra-personal variations, and both losses guide the CNN to learn more discriminative representations. [33] designed a triplet loss for CNN to learn representations in a compact Euclidean space where distances directly correspond to a measure of similarity, and the learned representation performs well on several tasks including recognition, verification, and clustering. [38] proposed a center loss and combine it with cross entropy loss to train the CNN for learning more discriminative features, and they also propose a mini-batch based update method for the centers, which was proved to be useful for face recognition and verification. [26] proposed a generalized large-margin softmax loss which explicitly encourages intra-class compactness and inter-class separability between learned representations, making the representation more discriminative and robust. [25] further proposed a angular soft-max loss, which can ensure the learned representations more angularly discriminative, and this method was proved to be efficient under open-set protocols.

Prototype learning is a classical and representative method in patter recognition society. The earliest prototype learning method is k-nearest-neighbor (K-NN). In order to reduce the heavy burden of storages space and computation requirement of K-NN, an improvement called learning vector quantization (LVQ) is proposed [12]. The LVQ has been studied in many works and it has a lot of variations. According to the updating methods of the prototypes, we can classify the LVQ methods into two main categories. The first category concentrates on designing suitable updating conditions and rules to learn the prototypes, and the representing works include [12, 13, 3, 18, 20]. The other category learns the prototypes in a parameter optimization way, by defining loss functions with regard to the prototypes and learning the prototypes through optimizing the loss functions. The representative methods include [31, 32, 10, 27, 2]. A detailed review and evaluation of the prototype based learning methods can be found in [21]. Previous prototype learning methods are mainly based on hand-designed features and they were widely used in different pattern recognition tasks be-
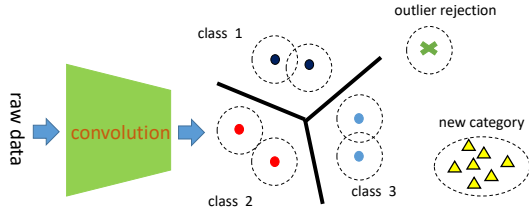
Figure 2. An illustration of convolutional prototype learning.

fore the arrival of CNN. To the best of our knowledge, this is the first work on combining the prototype based classifiers with deep convolutional neural networks for both high accuracy and robust pattern classification.

3. Convolutional prototype learning

3.1. Architecture of the framework

Compared with hand-designed features, the features automatically learned from data usually perform better for classification. Thus, we use a CNN as feature extractor in our framework, which is denoted as $f(x; \theta)$, x and θ denote the raw input and parameters of the CNN respectively. Different from the traditional CNN which use softmax layer for linear classification on the learned features, we maintain and learn several prototypes on the features for each class and use prototype matching for classification. The prototypes are denoted as m_{ij} where $i \in \{1, 2, ..., C\}$ represents the index of the classes and $j \in \{1, 2, ..., K\}$ represents the index of the prototypes in each class. Here we assume each class having equal number of K prototypes and this assumption can be easily relaxed in real application.

The CNN feature extractor $f(x; \theta)$ and the prototypes $\{m_{ij}\}$ are jointly trained from data. In the classification stage, we classify the objects by prototype matching, i.e., we find the nearest prototype according the Euclidean distance and assign the class of this prototype to the particular object. A graphic description of our framework can be seen in Fig. 2.

3.2. Feedforward for prediction

Given an input pattern x, we first get its abstract representation by the CNN feature extractor, then we compare the abstract feature with all prototypes and classify it to the category where the nearest prototype belongs to:

$$x \in class \ \ \arg\max_{i=1}^{C} g_i(x) \tag{1}$$

where $g_i(x)$ is the discriminant function for class i:

$$g_i(x) = -\min_{j=1}^{K} \|f(x; \theta) - m_{ij}\|_2^2 \tag{2}$$

3.3. Backward for training

The trainable parameters in our framework are composed by two parts. One is the parameters of the CNN extractor, which is denoted as θ; the other is the prototypes in each class, which is denoted as $M = \{m_{ij} | i = 1, ..., C; \ j = 1, ..., K\}$. The parameters of θ and M should be trained jointly in an end-to-end manner, and this can make them cooperate better with each other, which is beneficial for the performance of classification. To train the framework, we should first define the corresponding loss function. Besides, the loss function should be derivable with respect to θ and M as well. The loss function should also be closely related to the classification accuracy. In following subsections, we introduce multiple loss functions designed to train CPL.

3.3.1 Minimum classification error loss (MCE)

Minimum classification error (MCE) loss is firstly proposed by [30]. We modify this loss function and make it applicable in our framework. In prototype learning, the discriminant function is defined as Eq. 2. Then the misclassification measure of a sample from class y is given by:

$$\mu_y(x) = -g_y(x) + \left[\frac{1}{C-1} \sum_{j \neq y} g_j(x)^\eta \right]^{1/\eta} \tag{3}$$

when η approaches infinity, the misclassification measure becomes:

$$\mu_y(x) = -g_y(x) + g_r(x) \tag{4}$$

where $g_r(x)$ is the most competitive class, i.e.,

$$g_r(x) = \max_{k \neq y} g_k(x) \tag{5}$$

then we can rewritten the misclassification measure as

$$\mu_y(x) = \|f(x) - m_{yi}\|_2^2 - \|f(x) - m_{rj}\|_2^2 \tag{6}$$

where m_{yi} is the closest prototype from the genuine class while m_{rj} is the closest prototype from incorrect classes. Then, the loss function is defined as:

$$l((x, y); \theta, M) = \frac{1}{1 + e^{-\xi \mu_y}}. \tag{7}$$

From the definition of the loss function, we can see that during minimizing the loss function, the μ_y is also minimized. This means $\|f(x) - m_{yi}\|_2^2$ is decreased and $\|f(x) - m_{rj}\|_2^2$ is increased, which acts like pull the feature closer to its class but push it away from the other classes. Thus, the training can help the framework to achieve better classification performance on the training samples.

The MCE loss is derivable with regard to M and f, the derivatives can be calculated as:

$$\frac{\partial l}{\partial f} = 2\xi l(1-l)(m_{rj} - m_{yi}). \qquad (8)$$

$$\frac{\partial l}{\partial m_{yi}} = 2\xi l(1-l)(m_{yi} - f(x)) \qquad (9)$$

$$\frac{\partial l}{\partial m_{rj}} = 2\xi l(1-l)(f(x) - m_{rj}) \qquad (10)$$

and for the remainder prototypes, we have:

$$\frac{\partial l}{\partial m} = 0. \qquad (11)$$

Note that f is the output of the CNN feature extractor, according to the error back propagation algorithm, the derivatives with regard to the parameters of CNN can be calculated start from $\partial l / \partial f$. Given the gradients of the loss function, the gradient based optimization methods can be used to update the whole framework.

3.3.2   Margin based classification loss (MCL)

For a training sample (x, y), let m_{yi} and m_{rj} denote the closest prototypes from the correct class and the most competitive class respectively. If the sample is classified correctly, then $d(f(x), m_{yi}) < d(f(x), m_{rj})$ and we think the loss should be 0. If the sample is misclassified, then $d(f(x), m_{yi}) > d(f(x), m_{rj})$ and we think the loss exists in this situation. Naturally, the loss now can be defined as $d(f(x), m_{yi}) - d(f(x), m_{rj})$. Putting the two situations together, we can define the loss function as:

$$l((x, y); \theta, M) = [d(f(x), m_{yi}) - d(f(x), m_{rj})]_+ \quad (12)$$

To increase the classification ability of the framework, a margin is added to the loss function 12, leading to the new margin based classification loss (MCL) function, which is denoted as:

$$l((x, y); \theta, M) = [d(f(x), m_{yi}) - d(f(x), m_{rj}) + m]_+ \qquad (13)$$

where m is a positive number and acts as the margin. Compared with Eq. 12, MCL is stricter, it penalize the framework even though it classifies some sample correctly (within the margin). Thus, it can increase the discriminative ability of the framework.

In order to apply MCL successfully, the margin m should be carefully selected and it should have same scale with $d(f(x), m_{yi}) - d(f(x), m_{rj})$. However, the scale of $d(f(x), m_{yi}) - d(f(x), m_{rj})$ is unknown and we have to compute and estimate it from the training data. To avoid this problem, a generalized margin based classification loss (GMCL) function is proposed and defined as:

$$l((x, y); \theta, M) = \left[\frac{d(f(x), m_{yi}) - d(f(x), m_{rj})}{d(f(x), m_{yi}) + d(f(x), m_{rj})} + m\right]_+ \qquad (14)$$

In Eq. 14, $-1 < \frac{d(f(x), m_{yi}) - d(f(x), m_{rj})}{d(f(x), m_{yi}) + d(f(x), m_{rj})} < 1$, thus we can simply choose margin m from $(0, 1)$.

Both MCL and GMCL are derivable with regard to M and f, for MCL, the gradients can be computed by:

$$\frac{\partial l}{\partial f} = \begin{cases} 2(m_{rj} - m_{yi}) & l > 0 \\ 0 & l \le 0 \end{cases} \qquad (15)$$

$$\frac{\partial l}{\partial m_{yi}} = \begin{cases} 2(m_{yi} - f) & l > 0 \\ 0 & l \le 0 \end{cases} \qquad (16)$$

$$\frac{\partial l}{\partial m_{rj}} = \begin{cases} 2(f - m_{rj}) & l > 0 \\ 0 & l \le 0 \end{cases} \qquad (17)$$

for other prototypes, we have

$$\partial l / \partial m_{ij} = 0 \qquad (18)$$

Similarly, the gradients of GMCL can also be computed directly and we do not list the equations any more.

Equally, the gradients of the loss function with regard to θ can also be computed by the error back propagation algorithm beginning from $\partial l / \partial f$. In MCL and GMCL, the framework is updated only when the loss exists and during the updating, only two prototypes are trained but the other prototypes are kept unchanged.

3.3.3   Distance based cross entropy loss (DCE)

In our CPL framework, the distance can be used to measure the similarity between the samples and the prototypes. Thus, the probability of a sample (x, y) belonging to the prototype m_{ij} can be measured by the distance between them:

$$p(x \in m_{ij}|x) \propto - \|f(x) - m_{ij}\|_2^2. \qquad (19)$$

To satisfy the non-negative and sum-to-one properties of the probability, we further define the probability $p(x \in m_{ij}|x)$ as:

$$p(x \in m_{ij}|x) = \frac{e^{-\gamma d(f(x), m_{ij})}}{\sum_{k=1}^{C} \sum_{l=1}^{K} e^{-\gamma d(f(x), m_{kl})}} \qquad (20)$$

where $d(f(x), m_{ij}) = \|f(x) - m_{ij}\|_2^2$ represents the distance between $f(x)$ and m_{ij}. γ is a hyper-parameter that control the hardness of probability assignment. Given the definition of $p(x \in m_{ij}|x)$, we can further define the probability of $p(y|x)$ as:

$$p(y|x) = \sum_{j=1}^{K} p(x \in m_{yj}|x). \qquad (21)$$

Based on the probability of $p(y|x)$, we can define the cross entropy (CE) loss under our framework as:

$$l((x, y); \theta, M) = -log p(y|x). \qquad (22)$$

This loss function is defined based on the distance, to distinguish it from the traditional cross entropy loss, we call it distance based cross entropy (DCE) loss. From Eq. 20, 21 and 22, we can see that minimizing the loss function essentially means decreasing the distance between the samples with the prototypes which come from the genuine class of the samples.

Obviously, the DCE is also derivable with regard to M and f, here we don't list the equation of the corresponding gradients any more. Similarly, the derivatives with regard to the parameters of CNN can also be calculated starting from $\partial l / \partial f$ by the chain rule. Compared with MCL and GM-CL, DCE updates all the prototypes every time during the training, thus it converges faster than the MCL and GMCL.

3.4. Generalized CPL with prototype loss

The loss functions defined in section 3.3 are used as measurements of classification accuracy, and by minimizing these losses, we can train the model to classify the data correctly. However, directly minimizing the classification loss may lead to over-fitting. In light of this, we propose a new prototype loss (PL) as a regularization, which acts like a generative model to improve the generalization performance of CPL.

From the prediction function defined in Eq. 1, we can derive the decision boundary of the CPL:

$$\|f - m_{ij}\|_2^2 = \|f - m_{kl}\|_2^2 \qquad (23)$$

$$2f \cdot (m_{kl} - m_{ij}) + \|m_{ij}\|_2^2 - \|m_{kl}\|_2^2 = 0. \qquad (24)$$

We can see that the resulted decision boundary is still linear. Like traditional CNN framework for classification, the CPL still separates the whole feature space and the learned representation is still linearly separable. As stated before, this kind of framework is not robust, it can not reject the samples from unseen classes and can not be extended to new classes conveniently. To overcome this problem, a new loss function called prototype loss (PL) is added in our framework, which is defined as:

$$pl((x, y); \theta, M) = \|f(x) - m_{yj}\|_2^2 \qquad (25)$$

where m_{yj} is the closest prototype with $f(x)$ from the corresponding class y. The prototype loss can be combined with the classification loss defined in section 3.3 to train the model. Then the total loss can be defined as:

$$loss((x, y); \theta, M) = l((x, y); \theta, M) + \lambda pl((x, y); \theta, M) \qquad (26)$$

where λ is a hyper-parameter which control the weight of prototype loss. The PL can also be viewed as the maximum-likelihood (ML) regularization [23, 22] which is widely used in pattern recognition [11, 19].

PL can further boost the performance of CPL, because: (1) PL pull the features of samples close to their corresponding prototypes, making the features within the same class more compact, this can implicitly increase the distance between the classes, which is beneficial for classification; (2) the classification loss stresses the separation property of the representation and the prototype loss stresses the compactness property of the representation, by combining them together, we can learn intra-class compact and inter-class separable representations, which are more robust and more appropriate for rejection and open set problems. We denote CPL equipped with PL as generalized convolutional prototype learning (GCPL).

4. Application of GCPL

Besides classification, GCPL can also be used for rejection and class-incremental learning. In this paper, we did not invent new rejection and class-incremental learning methods, but only show that our framework is suitable for these two tasks. Most rejection strategies are based on the probabilities (confidences) produced by the softmax layer of CNN model. In our framework, we can also obtain classification probabilities by Eq. 21, so the same strategies can also be used in our framework. The distance is also a meaningful measurement for the classification confidence in our framework, thus the same strategies can also be implemented based on the distance outputted by GCPL.

For class-incremental learning, we only consider the most general case. That is, given a trained framework and some samples from a new class, we should extend the framework to recognize the new class correctly and keep the accuracy on the old classes. In GCPL framework, the learned representations have better clustering property, for samples of the new class, the resulted features are also compact. Thus we can cluster the features of new class or use the mean of these features as the prototypes for the new class. Therefore, the framework can be expanded to the new class easily. In the following experimental section, we will show that this approach is very effective for class-incremental learning.

5. Experiments

5.1. Experiments and analysis on MNIST [17]

In this experiment, the architecture of CNN feature extractor is the same as the network used in [38] with the ReLU activation function. The output of the CNN feature vector is set to be two, thus we can directly plot the features on the 2-D surface for visualization. We maintain one prototype in each class and train the framework under the DCE loss and DCE+PL loss respectively. Meanwhile, we also give the accuracy of the traditional softmax based CNN framework under the same architecture as [38]. We set

method		test accuracy (%)
soft-max		99.08
CPL (DCE)		99.28
GCPL(DCE+PL)	$\lambda = 0.001$	99.33
	$\lambda = 0.01$	99.29
	$\lambda = 0.1$	99.30

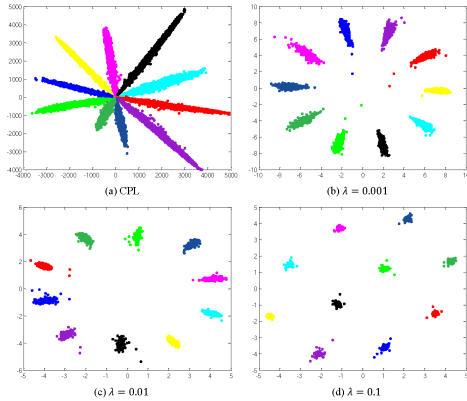Table 1. Test accuracy of different methods on MNIST

Figure 3. The learned representations of CPL and GCPL on M-NIST. Different colors represent different classes

the initial learning rate as 0.001, the batch size as 50, and the hyper-parameter γ in DCE as 1.0 during the training. The final results can be seen in table 1 and Fig. 3.

From table 1, we can see that our new proposed CPL framework can achieve comparable performance with traditional softmax based CNN under the same structure. Moreover, by cooperating with PL, our GCPL can achieve even better results. This demonstrates that the PL, which acts as an implicit regularization, is beneficial for classification as well. Meanwhile, we can also see that the test accuracy is not sensitive to the parameter λ, the change of λ did not greatly impact the results. From Fig. 3 (a), we can see that when only use the classification loss, the resulted representations are still linear separable, this demonstrates that our analysis in section 3.4 is correct. From Fig. 3 (b) to (d), we can see the learned representations under GCPL framework are really inter-class separable and intra-class compact. With the increasing of the weight λ on PL loss, the learned representations within the same class become more and more compact. This demonstrate our GCPL framework can really learn robust and discriminative representations.

5.2. Experiments and analysis on CIFAR-10

We realized several CNN structures on CIFAR-10 [14], including the model C appeared in [35], a modified version of model C that adds batch normalization layers after each convolutional and fully connected layers in model C, residual net 20 and residual net 32 in [8]. Then we compare the

CNN structure	soft-max	CPL	GCPL
model C [35]	90.26 [35]	90.70	90.80
model C with BN	91.37	91.59	91.90
ResNet 20	91.32	91.46	91.63
ResNet 32	92.50	92.60	92.63

Table 2. The accuracy of different CNN structures and different models on CIFAR-10

loss function	accuracy (%)
soft-max	97.55 [40]
MCE	97.35
MCL	97.61
GMCL	97.36
DCE	97.58

Table 3. The accuracy of GCPL on OLHWDB dataset

performance of the traditional softmax based classification, CPL, and GCPL methods under these CNN structures. In all experiments, we maintain one prototype for each class and the prototypes are initialized as zero vectors. The images data are whitened and contrast normalized following [4]. The experimental results are shown in table 2.

From table 2, we can see that the proposed CPL framework can achieve comparable or even better results than the traditional softmax based classification method under different network structures. This further demonstrates the efficiency and generality of the proposed framework. Besides, the GCPL framework, which has an additional prototype loss during training, performs best in all experiments on CIFAR-10.

5.3. Experiments and analysis on OLHWDB

Online handwriting database (OLHWDB [24]) is a large scale Chinese handwriting dataset. Following the settings in [40] and [39], the training and test datasets include 2,697,673 and 224,590 samples respectively, which are come from 3755 classes. We use the same CNN structure as [40] and make little modifications with batch normalization and ReLU to improve training process. We mainly test the GCPL framework on this dataset and adopt different classification loss functions to study their influence on the performance. We maintain only one prototype for each class and the prototypes are initialized as the mean of training features in the corresponding classes. We use the same data pre-processing method as [40]. The experimental results are listed in table 3. From table 3, we can see that our framework also performs well or even better on large scale classification problems, which again demonstrate its efficiency and generality.

Note that our purpose is not to achieve significantly better accuracy than previous softmax-based CNN model, we only want to show that, from the accuracy perspective, our framework can match or work slightly better than tradition-

softmax		GCPL Prob		GCPL Dist	
AR	RR	AR	RR	AR	RR
100.0	0.000	100.0	0.000	**94.20**	**100.0**
99.98	0.200	99.99	15.67	**97.39**	**100.0**
99.72	8.110	99.80	46.12	**98.07**	**100.0**
99.14	25.17	99.39	87.07	**98.43**	**100.0**
98.52	40.60	99.30	93.43	**98.57**	**99.99**
97.61	57.54	99.21	96.31	**98.73**	**99.99**
83.95	71.66	98.96	98.69	**98.89**	**99.99**
76.67	85.97	98.73	99.46	**99.09**	**99.99**
75.49	98.02	98.21	99.86	**99.20**	**99.99**

Table 4. The tradeoff between acceptance rate AR (%) and rejection rate RR (%) for different methods.

al CNN model. The advantage of our model in another perspective is that it can significantly improve the robustness of pattern recognition. In following subsections, we will show this from the viewpoints of rejection and class-incremental learning.

5.4. Experiments for rejection

To test the robustness of our GCPL framework, we further conduct experiments for rejection. We firstly train a network on MNIST training set. To evaluate its robustness, we use two test sets (MNIST and CIFAR-10 test sets) for this network. The CIFAR-10 test samples are not digits, and therefore, they should be viewed as outliers and then be rejected by this network. At the same time, the samples from the MNIST test set should still be accepted since they are from the same domain as the training data. Actually, the rejection and acceptance performance are closely coupled, we can only get a tradeoff between them.

To fairly evaluate the performance, we use two measurements of acceptance rate (AR) and rejection rate (RR). AR denotes the percentage of accepted samples in MNIST test set (how many MNIST samples have been accepted), while RR denotes the percentage of rejected samples in CIFAR-10 test set (how many CIFAR-10 samples have been rejected). The higher of these two measurements, the better of the model in robustness. We adopt the most frequently used threshold-based rejection strategy, i.e., if the output confidence for a sample is larger than the pre-defined threshold, then it will be accepted, otherwise it will be rejected. The confidence can be obtained by the output probability or distance in GCPL (section 4). Different from the probability, the smaller distance represents larger confidence. We use the same structure of CNN feature extractor as section 5.1. For comparison, we also test the rejection performance of traditional softmax based framework under the same CNN structure. The results are showed in table 4. Note these results are obtained by using different (smoothly changed) thresholds to give the AR-RR tradeoffs.

From table 4, we can see that the softmax-based model is confused by the MNIST and CIFAR-10 samples, the high AR and high RR is not able to coexist. This explains that the softmax-based model is not robust in outlier detection. Differently, our GCPL model can achieve better rejection performance and meantime keep satisfactory acceptance rate. For example, while over 99% CIFAR-10 samples being rejected, we can still keep over 99% MNIST samples accepted. This is a significant advantage compared with softmax based approach, which demonstrates the robustness of the GCPL framework.

5.5. Experiments for class-incremental learning

We conduct experiments on MNIST and CIFAR dataset to demonstrate the superiority of GCPL framework for class-incremental learning. We treat all MNIST samples (from class 0 to 9) as the known class data and choose one class from CIFAR-10 as the new class (which should be learned incrementally). We use the same network structure as described in section 5.1 and train the GCPL on the MNIST training set, then we feed the test data from both known and unknown classes to the trained GCPL and obtain their representations, the results are shown in Fig. 4.

(a) CIFAR-3 as the new class (b) CIFAR-6 as the new class

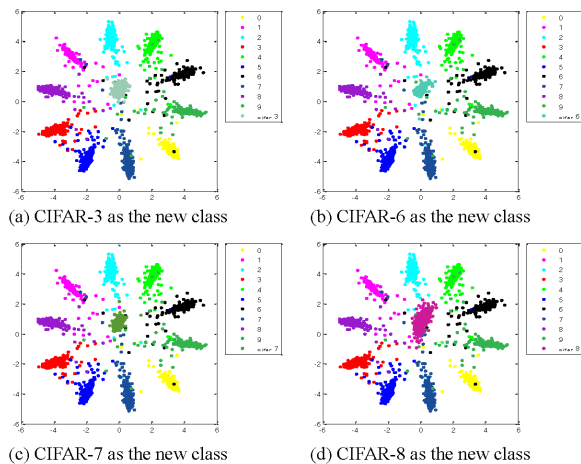(c) CIFAR-7 as the new class (d) CIFAR-8 as the new class

Figure 4. The learned representations for both known and unknown (new) classes

From Fig. 4, we can find that the learned representations for both known and unknown (new) classes are intra-class compact and inter-class separable, this again demonstrates the robustness of the GCPL framework. Based on such representations, we can simply use the mean of the training data from the new class as the prototype for the new class, then we can directly extend the GCPL to make predictions for both the new class and the previous classes. To evaluate the accuracy of GCPL for class-incremental learning, we further give the accuracy of the extended GCPL on the test

new class ID (CIFAR)	test accuracy (11-class)
0	99.23
1	99.23
3	99.24
5	99.23
6	99.24
7	99.22
8	99.20
9	99.23
without new class	99.27

Table 5. Accuracy (%) of GCPL on test samples from both known and new classes

samples from both known and new classes (total 11 classes) in table 5. From table 5, we can see the GCPL still keeps high performance when extended to the new class. Note that in this class-incremental learning process, we did not re-train any part of the model, and due to the advantage of prototype-based decision making, we can directly add a new prototype to represent the new class. This demonstrates the advantage of GCLP for class-incremental learning.

5.6. Experiments with small sample size

Another aspect of robust classification is the ability to deal with small sample size (SSS). To further demonstrate the robustness of GCPL, we conduct experiments under the condition of SSS. We use different number (percentages) of samples in MNIST training set to train the models and then observe their accuracy on the MNIST test set. For GCPL, We use the same net structure and training settings as described in section 5.1, the weight λ for prototype loss is 0.001. For comparison, we also give the test accuracy of the traditional softmax-based CNN framework under the same architecture and training data. We repeat the experiments for five times and show the statistical results in table 6.

sample size (%)	soft-max	GCPL
100	99.08 ± 0.10	99.33 ± 0.10
50	98.07 ± 0.39	99.12 ± 0.10
30	92.68 ± 4.52	98.89 ± 0.10
10	86.12 ± 6.00	97.80 ± 0.22
5	73.95 ± 6.10	96.44 ± 0.40
3	50.79 ± 17.44	94.90 ± 0.58

Table 6. Test accuracy (%) under different percentages of training samples.

From table 6, we can find that the decreasing of training samples has less impact on the performance of GCPL. Compared with the softmax-based framework, the performance of GCPL declines much slower when the sample size is reduced. In particular, when training with very small number of training samples (e.g., with only 5% or 3% training samples), the test accuracies for softmax-based framework are

very low (with large variance), which demonstrates that the softmax-based model is not robust for SSS problem. On the contrary, under the same situation, the GCPL is still very effective for different sizes of training samples. GCPL not only achieves much higher accuracy but also shows more stable results with smaller variances. This again demonstrates the robustness of GCPL in dealing with SSS problem.

5.7. Experiments with multiple prototypes

In all previous experiments, we set the number of prototypes K in each class as 1. In this section, we adopt different values for K and investigate its effect on the classification performance. We use model C (described in section 5.2) with DCE loss, and conduct the experiment on CIFAR-10 dataset. For different K, we use the same settings for the hyper-parameters during the training, the results are shown in table 7.

K	1	2	3	4	5
accuracy (%)	90.70	90.40	90.37	90.67	90.46

Table 7. Test accuracy (%) under different values of K.

From table 7, we can see that more prototypes didn't lead to better results. Actually, CNN is very powerful for feature extraction, even though the initial intra-class distribution may be very complex, after CNN transformation, it can still be well modeled with a single Gaussian distribution (a single prototype). However, in more complicated scenarios, where the data distributions are difficult to model, more prototypes may be beneficial.

6. Conclusion

This paper proposed a convolutional prototype learning (CPL) framework for pattern recognition. Different from the softmax-based models, CPL directly learn multiple prototypes (in convolutional feature space) for each class and then use prototype matching for decision making. CPL can achieve comparable or even better classification accuracy than softmax-based CNN models. To further improve the robustness, we propose a prototype loss (PL) to increase the intra-class compactness, resulting in a generalized CPL (GCPL) model. The GCPL has great advantage compared with traditional CNN models in the perspectives of outlier rejection and class-incremental learning. In future, we will try to conduct more detailed experiments to evaluate other properties of GCPL such as the adaptation ability in changing environment, dealing with weakly labeled data, and so on.

References

[1] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *NIPS*, pages 379–387, 2016.

[2] C. Decaestecker. Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2):281–288, 1997.

[3] S. Geva and J. Sitte. Adaptive nearest neighbor pattern classification. *IEEE Trans. Neural Networks*, 2(2):318–22, 1991.

[4] I. J. Goodfellow, D. Wardefarley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[5] R. Hadsell, S. Chopra, and Y. Lecun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, pages 1735–1742, 2006.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[9] W. He, X.-Y. Zhang, F. Yin, and C.-L. Liu. Deep direct regression for multi-oriented scene text detection. In *ICCV*, pages 745–753, 2017.

[10] Y. S. Huang, K. Liu, C. Suen, A. J. Shie, I. I. Shyu, M. C. Liang, R. Y. Tsay, and P. K. Huang. A simulated annealing approach to construct optimized prototypes for nearest-neighbor classification. In *ICPR*, page 483, 1996.

[11] X.-B. Jin, C.-L. Liu, and X. Hou. Regularized margin-based conditional log-likelihood loss for prototype learning. *Pattern Recognition*, 43(7):2428–2438, 2010.

[12] T. Kohonen. The self-organizing map. *Neurocomputing*, 21(1C3):1–6, 1990.

[13] T. Kohonen. Improved versions of learning vector quantization. In *IJCNN*, pages 545–550 vol.1, 2012.

[14] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[16] Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In *NIPS*, pages 396–404, 1990.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] S. W. Lee and H. H. Song. Optimal design of reference models for large-set handwritten character recognition . *Pattern Recognition*, 27(9):1267–1274, 1994.

[19] C.-L. Liu. One-vs-all training of prototype classifier for pattern classification and retrieval. In *ICPR*, pages 3328–3331. IEEE, 2010.

[20] C.-L. Liu, I. J. Kim, and H. K. Jin. High accuracy handwritten Chinese character recognition by improved feature matching method. In *ICDAR*, pages 1033–1037 vol.2, 1997.

[21] C.-L. Liu and M. Nakagawa. Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition. *Pattern Recognition*, 34(3):601–615, 2001.

[22] C.-L. Liu, H. Sako, and H. Fujisawa. Discriminative learning quadratic discriminant function for handwriting recognition. *IEEE Trans. Neural Networks*, 15(2):430–444, 2004.

[23] C.-L. Liu, H. Sako, and H. Fujisawa. Effects of classifier structures and training regimes on integrated segmentation and recognition of handwritten numeral strings. *IEEE Trans. PAMI*, 26(11):1395–1407, 2004.

[24] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. CASIA online and offline Chinese handwriting databases. In *ICDAR*, pages 37–41. IEEE, 2011.

[25] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, pages 6738–6746.

[26] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, 2016.

[27] D. Miller, A. V. Rao, K. Rose, and A. Gersho. A global optimization technique for statistical classifier design. *IEEE Trans. Signal Processing*, 44(12):3108–3122, 1996.

[28] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, pages 1990–1998, 2015.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015.

[30] B.-H. J. S. Katagiri. Discriminative learning for minimization error classification. *IEEE Trans. Signal Process*, 40(12):3043–3054, 1992.

[31] A. Sato. Generalized learning vector quantization. *NIPS*, 8227:19–26, 1996.

[32] A. Sato and K. Yamada. A formulation of learning vector quantization using a new misclassification measure. In *ICPR*, pages 322–325 vol.1, 1998.

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[35] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2014.

[36] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014.

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[38] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016.

[39] F. Yin, Q.-F. Wang, X.-Y. Zhang, and C.-L. Liu. Icdar 2013 Chinese handwriting recognition competition. In *ICDAR*, pages 1464–1470. IEEE, 2013.

[40] X.-Y. Zhang, Y. Bengio, and C.-L. Liu. Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition*, 61(61):348–360, 2016.