Hindawi
Complexity
Volume 2021, Article ID 5534385, 11 pages
https://doi.org/10.1155/2021/5534385

Research Article

Improving the Performance of Deep Learning Model-Based Classification by the Analysis of Local Probability

Guanghao Jin,[1] **Yixin Hu**,[2] **Yuming Jiao**,[2] **Junfang Wen**,[2] **and Qingzeng Song**[2]

[1]*Beijing Polytechnic, Beijing 100176, China*
[2]*School of Computer Science and Technology, Tiangong University, Tianjin 300387, China*

Correspondence should be addressed to Qingzeng Song; qingzengsong@tiangong.edu.cn

Received 13 January 2021; Revised 7 August 2021; Accepted 25 August 2021; Published 6 September 2021

Academic Editor: Atila Bueno

Generally, the performance of deep learning-based classification models is highly related to the captured features of training samples. When a sample is not clear or contains a similar number of features of many objects, we cannot easily classify what it is. Actually, human beings classify objects by not only the features but also some information such as the probability of these objects in an environment. For example, when we know further information such as one object has a higher probability in the environment than the others, we can easily give the answer about what is in the sample. We call this kind of probability as local probability as this is related to the local environment. In this paper, we carried out a new framework that is named L-PDL to improve the performance of deep learning based on the analysis of this kind of local probability. Firstly, our method trains the deep learning model on the training set. Then, we can get the probability of objects on each sample by this trained model. Secondly, we get the posterior local probability of objects on the validation set. Finally, this probability conditionally cooperates with the probability of objects on testing samples. We select three popular deep learning models on three real datasets for the evaluation. The experimental results show that our method can obviously improve the performance on the real datasets, which is better than the state-of-the-art methods.

1. Introduction

In these days, deep learning models have been proved efficient in many applications [1–8]. Generally, the performance of a deep learning-based classification model depends on the captured features [9–11]. When using a deep learning mode for the classification, the probability of each object is outputted. Then the object that has max value is selected as the final result.

In some cases, the probability of wrong object may be higher than that of the correct one. This is caused by similar features among these or the low efficiency of training models. To capture more features for higher accuracy, the structure of models becomes bigger while this is limited by many factors like the computational resource or the vanishing gradient problem [12–14]. Thus, there should be another way to improve the performance of deep learning model in real applications.

Different from deep learning models, human beings classify an object based on not only the features but also other factors. Figure 1 illustrates this kind of examples. The probabilities of person and animal may be both high in these samples, which may easily cause wrong classification results. In Figure 1(a), if we know there are no big animals in this area, the object is more likely to be a person. In Figure 1(b), if we know there is no human activity in this area, the object is more likely to be an animal. We call this as local probability, which presents the probability of objects in an environment. We believe that this is the reason why human beings can classify an object although they have not clearly seen it.

In this paper, we built a novel framework (L-PDL, Local Probability-based Deep Learning) to improve the performance of classification on the samples based on the analysis of local probability. Firstly, our method trains the deep learning model on the training set. Then, we can get the probability of objects on each sample by this trained model.

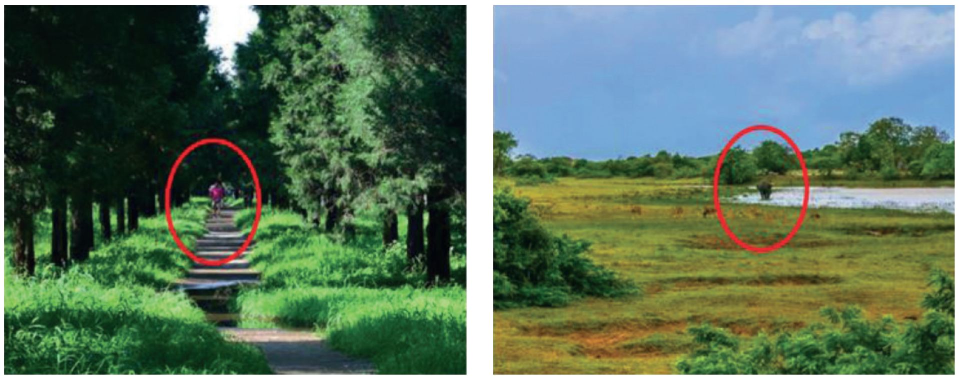

(a) (b)

Figure 1: Two examples. (a) If there are no big animals in this area, this object is more likely to be a person. (b) If there is no human activity in this area, this object is more likely to be an animal.

Secondly, we get the local probability of objects on the validation set. Finally, this probability conditionally cooperates with the probability of objects on testing samples.

Our contribution can be summarized as follows. (1) We built a novel framework that uses the local probability to increase the classification accuracy. Our framework does not need bigger models or more training samples while it can achieve higher accuracy than the existing methods. (2) Our framework increases the robustness of deep learning models for classification task. The local probability may be various in different environments. In this kind case, our framework only needs to update the local probability whose cost is lower than the retraining or transferring of models.

We performed our framework and the existing methods on the samples of CIFAR-10 [15–17], CIFAR-100 [18–20], and Mini-ImageNet [21–23]. All of these evaluations proved the effectiveness of our framework. We organize the paper as follows. Section 1 introduces the background and our contributions. Section 2 introduces the existing methods and their problems. In Section 3, we present our framework and related analyses. The experiment is organized in Section 4. Section 5 gives the conclusion and future work.

2. Related Work

(i) VoVNet-57 is designed for object detection task, which consists of a block including 3 convolution layers and 4 stages of OSA modules that output stride 32 [24]. An OSA module is comprised of 5 convolution layers with the same input/output channel for minimizing MAC. Whenever the stage goes up, the feature map is downsampled by 3×3 max pooling with stride 2. VoVNet-57 has more OSA modules at the 4th and 5th stage where downsampling is done in the last module.

(ii) VGG16 is a variant of VGG models for image recognition [25]. Figure 2 shows the structure of this model. The image is passed through a stack of convolutional layers, where the filters were used with a very small receptive field: 3×3. The convolution stride is fixed to 1 pixel. The padding is 1 pixel for 3×3 convolutional layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the convolutional layers. Max pooling is performed over a 2×2-pixel window, with stride 2. Three fully connected layers follow a stack of convolutional layers: the first two have 4096 channels each, and the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. All hidden layers are equipped with the rectification (ReLU) nonlinearity.

(iii) ResNeSt50 is a state-of-the-art deep learning framework for image classification that uses a modular Split-Attention block and enables attention across feature-map groups [26]. By stacking these Split-Attention blocks ResNet-style, it obtains a new ResNet variant which is called ResNeSt. There are four versions of ResNeSt. From ResNeSt50 to ResNeSt269, the structure becomes bigger and more complicated and can get higher accuracy when there are more and bigger size training samples. Based on the size of testing samples and computational resource, we use ResNeSt50 in this paper.

These models have been widely used, which are useful in many applications. To increase the accuracy of these models, we have to increase the number of training samples, which is hard work in many applications. Furthermore, the structure of these models has to be deeper and the training process needs some special techniques. Actually, there are many things that can be used to improve the accuracy in the real applications. The local probability is one of this kind of things, which will be introduced in the next section.

Some fusion operators have been proposed to improve the performance of classification by using multiple models [27]. In that paper, the authors solved mobile apps traffic by proposing a multiclassification approach, intelligently combining outputs from state-of-the-art classifiers proposed for mobile and encrypted traffic classification. In this paper, we also try to apply our framework on one of these fusion operators for higher accuracy.

Convolution Net Configuration					
Input sample					
Conv3-64	Conv3-64	Conv3-64	Conv3-64	Conv3-64	Conv3-64
	LRN	Conv3-64	Conv3-64	Conv3-64	Conv3-64
Maxpooling layers					
Conv3-128	Conv3-128	Conv3-128	Conv3-128	Conv3-128	Conv3-128
		Conv3-128	Conv3-128	Conv3-128	Conv3-128
Maxpooling layers					
Conv3-256	Conv3-256	Conv3-256	Conv3-256	Conv3-256	Conv3-256
Conv3-256	Conv3-256	Conv3-256	Conv3-256	Conv3-256	Conv3-256
			Conv1-256	Conv3-256	Conv3-256
					Conv3-256
Maxpooling layers					
Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512
Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512
			Conv1-512	Conv3-512	Conv3-512
					Conv3-512
Maxpooling layers					
Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512
Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512	Conv3-512
			Conv1-512	Conv3-512	Conv3-512
					Conv3-512
Maxpooling layers					
Full Connection 4096					
Full Connection 4096					
Full Connection 1000					
soft-max					

Figure 2: The structure of VGG16.

3. Our Framework

Before giving the details of our framework, we give the following definitions. These definitions are to explain the implementation of the methods.

3.1. Preliminaries. We set S_n as a sample and L_k as the label of an object. We set G_n as the ground truth on S_n where $G_n \in \{L_k\}$ [28, 29]. The label is to benefit the computation, which is generally a number [30, 31]. For example, when there are 10 objects to be classified, the label is from 0 to 9.

3.2. Our Framework. Figure 3 introduces our framework, which is named L-PDL (Local Probability-based Deep Learning). Firstly, our framework trains a deep learning model on the training set. Then, we can get the probability of labels (each label presents a kind of objects) on each sample of validation set by this trained model. Secondly, we get the posterior local probability of objects on the validation set. Thirdly, we confirm the parameters of conditional cooperation between this probability and the probability of labels. Finally, we use this conditional cooperation between posterior local probability and the output probability of models on testing samples to get final results.

3.3. The Probability by the Trained Model. We define $\mathbf{P}(\mathbf{M}(S_n) = L_k)$ as the probability of label L_k on the sample S_n by the trained model $\mathbf{M}$. Then the most possible result is selected by the following equation:

$$L_x = \text{argmax}_{L_k} \mathbf{P}(\mathbf{M}(S_n) = L_k), \quad (1)$$

which is used by deep learning models to predict the final result as Figure 4 illustrates.

Then, we define

$$\{L_k \vee \mathbf{P}(\mathbf{M}(S_n) = L_k) > \varepsilon\}, \quad (2)$$

as the set that includes the label L_k satisfying $\mathbf{P}(\mathbf{M}(S_n) = L_k) > \varepsilon$. This means that we only consider some

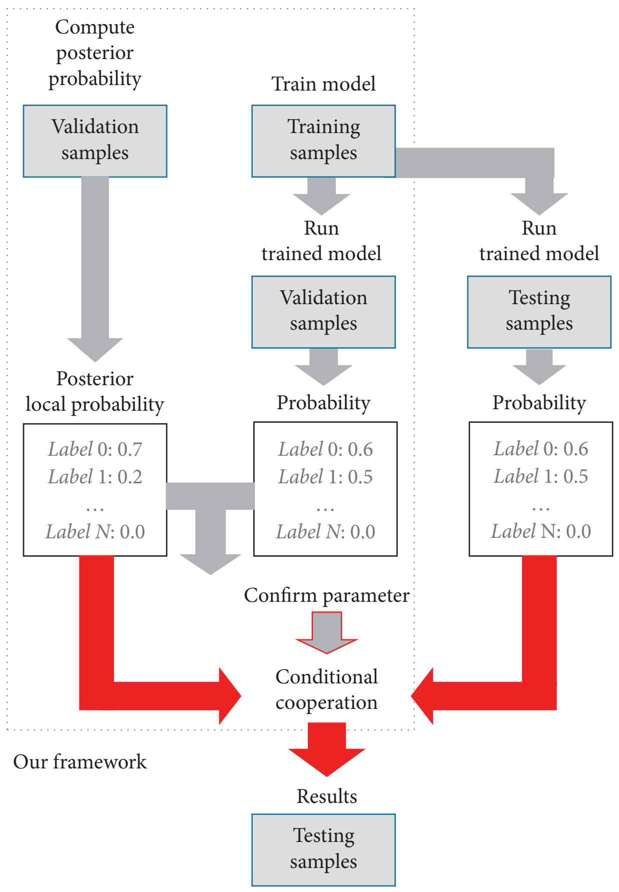


Figure 3: L-PDL, the framework of our methods.

of the L_k that have high probability to cooperate with the local probability.

3.4. The Local Probability. Figure 5(a) shows the probability of the labels that belong to the samples in training set. Figure 5(b) shows the local probability of the labels that belong to the samples in the validation and testing sets. As we can see in this figure, some labels may have less samples than the other ones.

We define $\mathbf{P}(L_k)$ as the local probability and $\widehat{P}(L_k)$ as the posterior local probability of label L_k on the validation set. Then, we make $\widehat{P}(L_k)$ assist the model probability to get correct results on the testing set.

3.5. Conditional Cooperation. In this subsection, we carried out two conditions that should be followed for the cooperation between local probability and model probability as follows:

$$\begin{aligned}\mathrm{Con}_1&: \{L_x \vee \mathbf{P}(\mathbf{M}(S_n) = L_x) < \delta\},\\ \mathrm{Con}_2&: \{L_x \vee \mathbf{P}(\mathbf{M}(S_n) = L_k) > \varepsilon\}.\end{aligned} \tag{3}$$

Con_1 means that we only reconsider the result L_x (having the max probability among all of the labels $\{L_k\}$), whose probability is smaller than δ. Then, we consider the labels, whose probabilities are bigger than ε as the potential set of the final result. Then, we can carry out two methods based on our framework.

3.5.1. Joint Cooperation (L-PDL-Joint). For $\mathrm{argmax}_{L_k}\mathbf{P}(\mathbf{M}(S_n) = L_k) \in \mathrm{Con}_1$,

$$\begin{aligned}\mathbf{F}(S_n, L_k) &= \mathbf{P}(\mathbf{M}(S_n) = L_k) \times \widehat{P}(L_k),\\ L_x &= \mathrm{argmax}_{L_k \in \mathrm{Con}_2}\mathbf{F}(S_n, L_k),\end{aligned} \tag{4}$$

and we call this L-PDL-joint from now on.

3.5.2. Weighted Cooperation (L-PDL-Weight). For $\mathrm{argmax}_{L_k}\mathbf{P}(\mathbf{M}(S_n) = L_k) \in \mathrm{Con}_1$,

$$\begin{aligned}\mathbf{F}(S_n, L_k) &= \mathbf{P}(\mathbf{M}(S_n) = L_k) + \omega \times \widehat{P}(L_k),\\ L_x &= \mathrm{argmax}_{L_k \in \mathrm{Con}_2}\mathbf{F}(S_n, L_k),\end{aligned} \tag{5}$$

and we call this L-PDL-weight from now on.

When using these methods, we should compute $\widehat{P}(L_k)$, δ, ε, and ω (only for L-PDL-weight) on the validation set. We can get the posterior local probability $\widehat{P}(L_k)$ on the validation samples. δ is the threshold that decides whether we reconsider a result or not. For example, if $\max \mathbf{P}(\mathbf{M}(S_n) = L_k) < 0.6$, we think the trained model is not highly sure about the correctness of the result. The parameter ε means that we only select some of the labels as the potential set of final result. This is to avoid the labels that have $\mathbf{P}(\mathbf{M}(S_n) = L_k) \approx 0$ being selected to be the final result because of the local probability. In other words, local probability should not be the only reason to select the final result.

3.6. Why Are Our Methods Better? In this section, we try to explain why our methods can perform better than the existing methods.

First reason: in the deep learning model case, the captured features play an important role in the classification. The number of captured features depends on the structure of layers [32, 33]. The training process of deep learning is to select the features that can present the samples. Then, the probability is used to present the distribution on these features. Thus, the object is more likely to be the label L_k than L_q when there is the following relation:

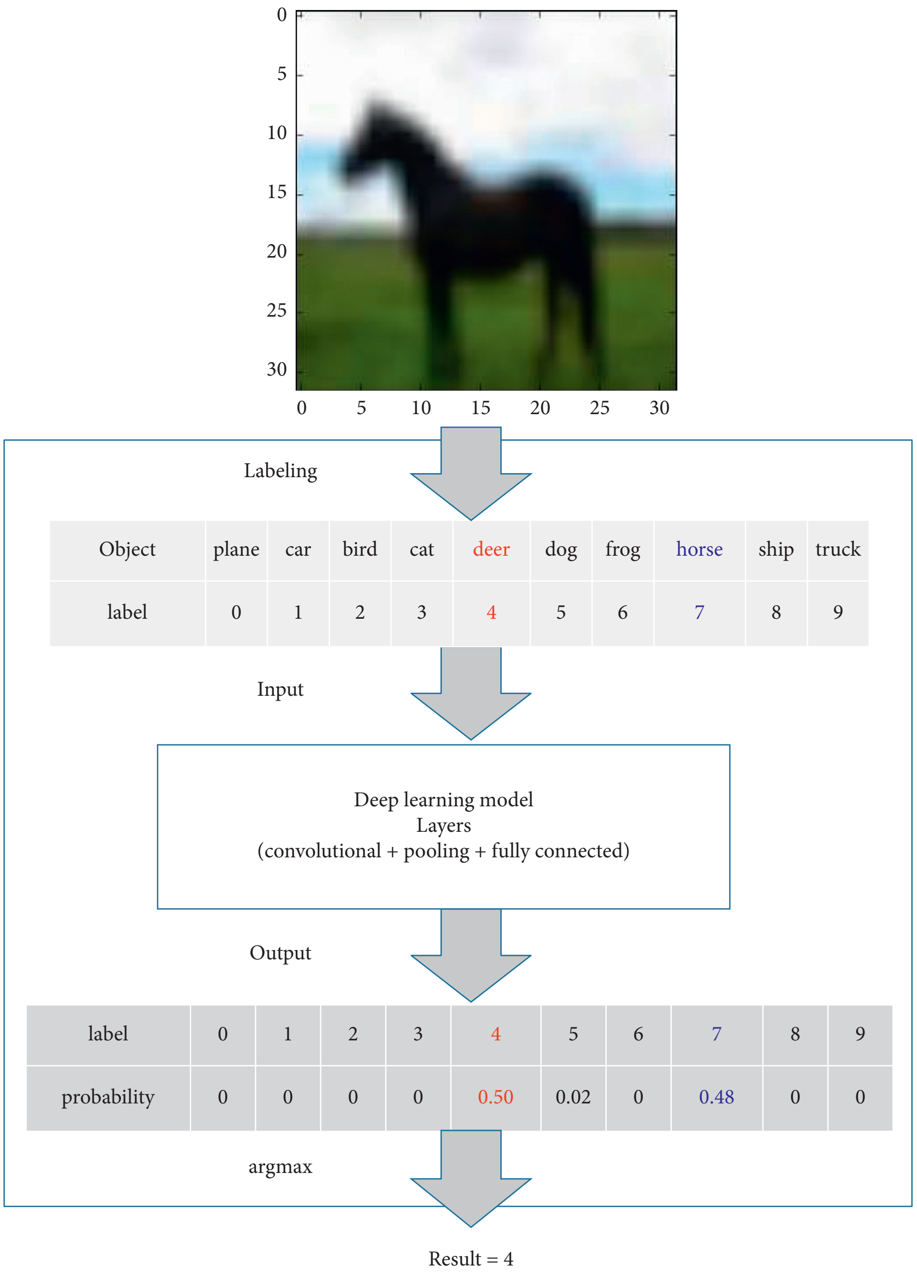
$$\mathbb{E}(\mathbf{P}(\mathbf{M}(S_n) = L_k)) > \mathbb{E}\left(\mathbf{P}\left(\mathbf{M}(S_n) = L_q\right)\right), \tag{6}$$

where here $\mathbb{E}(.)$ is the expected value and $L_k \neq L_q$. Thus, the selection of labels that have high probability is reasonable when reconsidering the result.

Second reason: there may be the following relation:

$$\mathbf{P}\left(\mathbf{M}(S_n) = L_q \neq G_n\right) > \mathbf{P}(\mathbf{M}(S_n) = L_k = G_n), \tag{7}$$

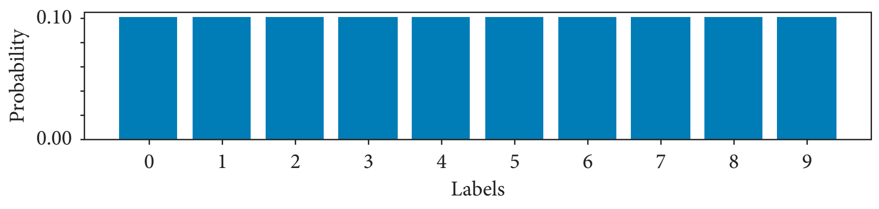
which means the trained model predicted a wrong result. In this kind of case, we believe that



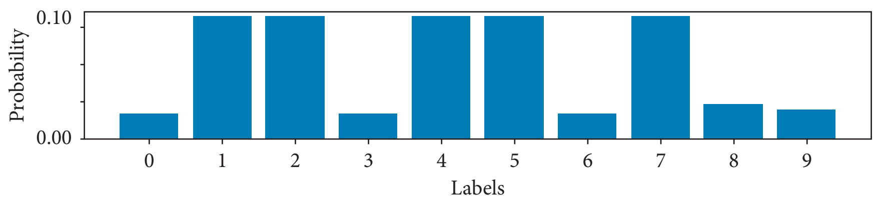
Object	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
label	0	1	2	3	4	5	6	7	8	9

label	0	1	2	3	4	5	6	7	8	9
probability	0	0	0	0	0.50	0.02	0	0.48	0	0

Figure 4: The process of the predicting result based on the probability of the labels.



(a)



(b)

Figure 5: The difference between two sets. (a) The probability of the training set. (b) A case of local probability.

$\mathbb{E}(\mathbf{P}(\mathbf{M}(S_n) = L_k = G_n)) \neq 0$. Especially when $\mathbf{P}(\mathbf{M}(S_n) = L_q) < \delta$, there may be $\mathbf{P}(\mathbf{M}(S_n) = L_k = G_n) \neq 0$, which shows the correct result may be the other. For example, $\mathbf{P}(\mathbf{M}(S_n) = L_4) = 0.50$ and $\mathbf{P}(\mathbf{M}(S_n) = L_7) = 0.48$ in Figure 4. In this kind of case, if we have the local probabilities

$$\widehat{P}(L_7) = 0.60 \gg \widehat{P}(L_4) = 0.01, \tag{8}$$

we can easily select the correct result L_7 that is "horse" in Figure 4.

Table 1 categorizes the reviewed works and our framework along with their main distinctive characteristics. ResNeSt50, VGG16, and VoVNet-57 are the deep learning models. Fusion operators [27] and our framework are fusion methods, which are based on these deep learning models. These models are needed to be trained on the training set. Our framework is needed to be trained on the validation set. Some of the fusion operators need to be trained on the validation set while the other ones do not need to be [27]. Our framework can be applied to a single model or multiple ones.

4. Experiment

We evaluate our methods with the existing ones on some real datasets in different local probability cases. When we randomized the parameters, we evaluate 1000 times. We trained the deep learning models on some real datasets by the reported default settings. We set the number of epochs [34, 35] as 10 for all these models on any training set. We do not focus on the designing of structure or tuning the hyperparameters. Instead, we focus on how to use the local probability to increase the accuracy.

4.1. The Evaluation on CIFAR-10. The evaluation on CIFAR-10 [15–17] has 50000 training samples and 10000 testing samples that belong to 10 labels. Each sample is an RGB image that has three channels: red, green, and blue. We use 50000 training samples to train the models. Then, we have 10000 samples left. We assign different local probabilities to these samples as Table 2 shows.

We use three kinds of local probability to evaluate the methods. In this table, Zero20 means 20% of the labels have zero samples. We define Zero40 (40% of the labels have zero samples) and Zero80 (80% of the labels have zero samples) by the same way. The labels to be zero samples are randomly selected. Figure 6 shows examples of these local probabilities. Then, the number of samples for the validation and testing sets is less than 10000 in these local probability cases. For example, there about 8000 samples left for these sets in the Zero20 case.

Our framework trained VoVNet-57 [24], VGG16 [25], and ResNeSt50 [26] on the training samples to generate trained models. Then, we use 1000 samples as the validation set and the remaining as the testing set. As we can see in Table 2, our methods can increase the accuracy by about 2.56% (in the Zero20 case), 5.83% (in the Zero40 case), and 13.06% (in the Zero80 case) compared to the best of the existing methods.

4.2. The Evaluation on CIFAR-100. This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each [18–20]. There are 500 training images and 100 testing images per label. We use 50000 training samples to train the models. Then, we have 10000 samples left. We assign different local probabilities to these samples.

We define Zero20, Zero40, and Zero80 by the same way as Section 4.1 has introduced. Then, we use 1000 samples as the validation set. As we can see in Table 3, our methods can increase the accuracy by about 2.57% (in the Zero20 case), 6.36% (in the Zero40 case), and 18.51% (in the Zero80 case) compared to the best of the existing methods.

4.3. The Evaluation on Mini-ImageNet. The Mini-ImageNet [21–23] dataset is for few-shot learning evaluation. Its complexity is high due to the use of ImageNet images but requires fewer resources and infrastructure than running on the full ImageNet dataset. In total, there are 100 labels with 600 samples of 84×84 colour images per label. We use 48000 training samples to train the models. Then, we have 12000 samples left. We assign different local probabilities to these samples.

We define Zero20, Zero40, and Zero80 by the same way as Section 4.1 has introduced. Then, we use 1000 samples as the validation set. As we can see in Table 4, our methods can increase the accuracy by about 2.26% (in the Zero20 case), 4.83% (in the Zero40 case), and 13.94% (in the Zero80 case) compared to the best of the existing methods.

4.4. Random Case on Two Datasets. In this subsection, we randomly assign the local probability to the CIFAR-100 and Mini-ImageNet. In more details, we randomly select the labels and assign random local probability to evaluate the methods.

Rand (.) is the function that outputs random value of probability. If the randomized value is smaller than 0, we use 0 instead of this value. Then, we can generate local probability by this function. For example, if the number of original samples for an object label is 1000 and Rand $(0,1) = 0.9$, we have 900 samples for this label in the local probability case. Figure 7 shows the examples of Rand (0, 1), Rand (−1, 1), and Rand (−2, 1).

As we can see in Table 5, our methods can increase the average accuracy by about 1.13% (in the Rand (0, 1) case), 8.76% (in the Rand (−1, 1) case), and 12.20% (in the Rand (−2, 1) case) compared to the best of the existing methods.

4.5. Multiple Models on Two Datasets. In this subsection, we apply our framework to the fusion operators, which uses the probabilities of multiple models [27]. We select the soft combiners, which require some parameters to be estimated, usually by means of a validation set. We selected the method class-conscious trainable combiner-based KL weights (named *CC-KL trainable* in Table 6) as a representative, which achieved better performance than the other methods in that paper. Then, we applied our framework to the result

Table 1: Categorizing the reviewed works and ours along with their main distinctive characteristics.

Characteristics	Works				
	VoVNet-57 [24]	VGG16 [25]	ResNeSt50 [26]	Fusion operators [27]	Our framework
Deep learning model	Yes	Yes	Yes	No	No
Fusion method	No	No	No	Yes	Yes
The number of models that are needed	Single	Single	Single	Multiple	Single/multiple
Training	Needed	Needed	Needed	Not needed/needed	Needed

Table 2: Result on CIFAR-10.

Methods	Zero20 (%)	Zero40 (%)	Zero80 (%)
VoVNet-57 [24]	82.13	81.41	82.185
VGG16 [25]	72.95	70.39	71.39
ResNeSt50 [26]	69.17	69.94	68.84
L-PDL-joint (our)	84.69	87.24	95.24
L-PDL-weight (our)	84.68	87.21	95.17

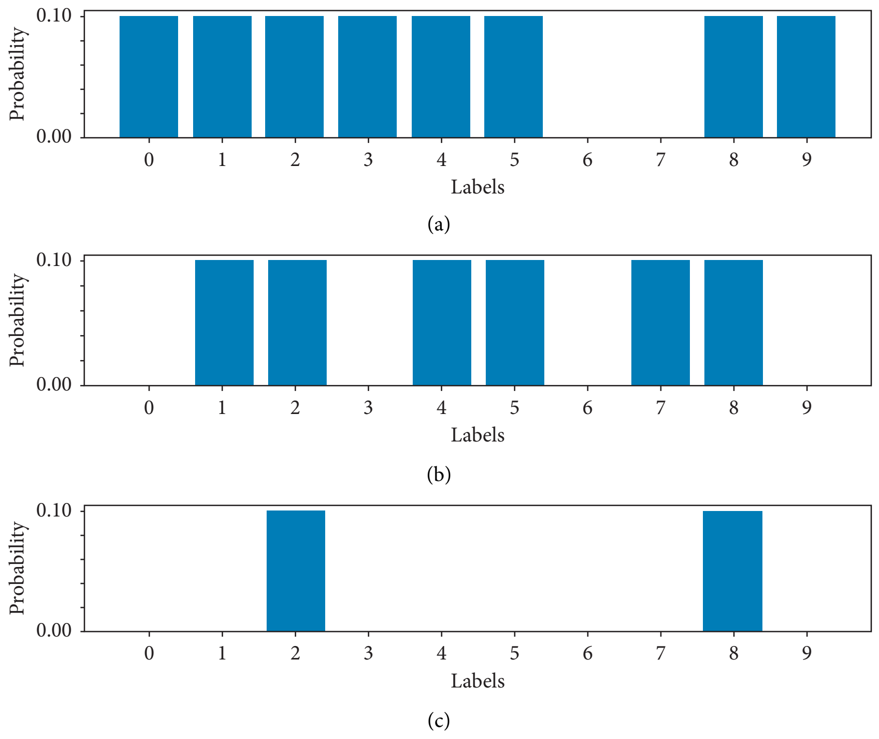


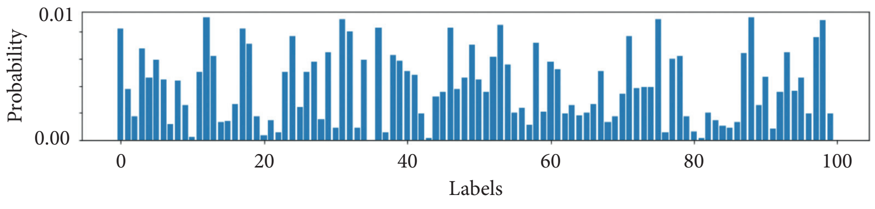
Figure 6: The examples of (a) Zero20, (b) Zero40, and (c) Zero80 on CIFAR-10.

Table 3: Result on CIFAR-100.

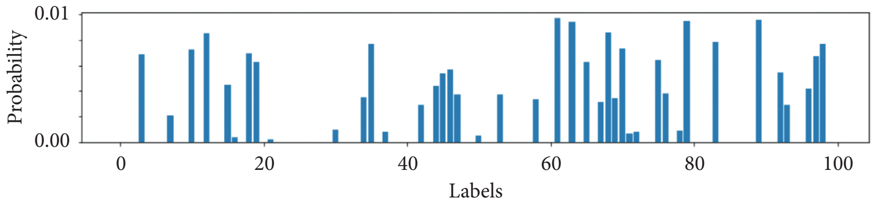
Methods	Zero20 (%)	Zero40 (%)	Zero80 (%)
VoVNet-57 [24]	64.23	63.85	63.6
VGG16 [25]	46.10	45.97	44.23
ResNeSt50 [26]	41.93	41.71	40.83
L-PDL-joint (our)	66.66	70.20	82.11
L-PDL-weight (our)	66.80	70.21	81.61

Table 4: Result on Mini-ImageNet.

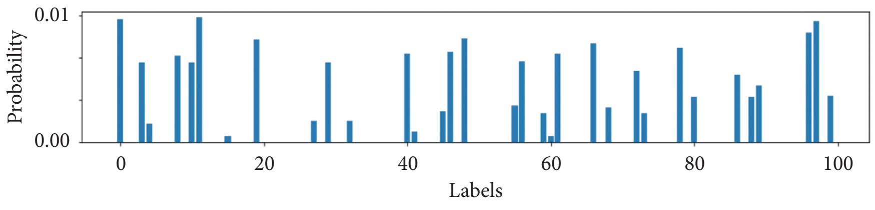
Methods	Zero20 (%)	Zero40 (%)	Zero80 (%)
VoVNet-57 [24]	72.94	72.53	71.53
VGG16 [25]	44.34	44.97	44.43
ResNeSt50 [26]	40.93	40.72	40.93
L-PDL-joint (our)	75.14	77.27	85.47
L-PDL-weight (our)	75.20	77.36	85.27



(a)



(b)



(c)

Figure 7: The examples. (a) Rand (0, 1). (b) Rand (−1, 1). (c) Rand (−2, 1).

Table 5: The result of random probability on CIFAR-100 and Mini-ImageNet.

Methods on local probability	VoVNet-57 [24]	Ours: L-PDL-joint/L-PDL-weight (%)
CIFAR-100, Rand (0, 1)	63.98	65.37
CIFAR-100, Rand (−1, 1)	63.78	73.73
CIFAR-100, Rand (−2, 1)	63.42	76.01
Mini-ImageNet, Rand (0, 1)	72.65	73.52
Mini-ImageNet, Rand (−1, 1)	72.58	80.15
Mini-ImageNet, Rand (−2, 1)	72.47	84.28

Table 6: The result of random probability on CIFAR-100 and Mini-ImageNet with fusion operators.

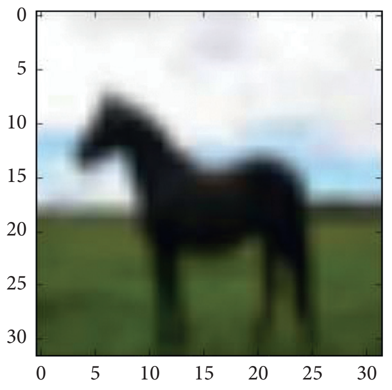
Local probability	Methods				
	ResNeSt50 [26]	VGG16 (%) [25]	VoVNet-57 (%) [24]	CC-KL trainable (%) [27]	CC-KL trainable with our framework (%)
CIFAR-100, Rand (0, 1)	41.93	46.19	63.98	65.52	66.27
CIFAR-100, Rand (−1, 1)	41.83	46.21	63.78	65.57	74.93
CIFAR-100, Rand (−2, 1)	42.03	46.35	63.42	66.16	77.02
Mini-ImageNet, Rand (0, 1)	40.34	44.54	72.65	73.39	74.55
Mini-ImageNet, Rand (−1, 1)	40.64	44.84	72.58	73.47	81.23
Mini-ImageNet, Rand (−2, 1)	41.04	45.01	72.47	74.45	85.48

of this method, which is named *CC-KL trainable with our framework* in Table 6.

As we can see in Table 6, CC-KL trainable can increase the accuracy by using the probability of models. On the other hand, the performance is limited by the accuracy of these models. As this table shows, our framework can further increase the accuracies with the cooperation of CC-KL trainable (introduced in [27]), which are about 0.96% (in the Rand (0, 1) case), 8.56% (in the Rand (−1, 1) case), and 10.95% (in the Rand (−2, 1) case) higher than the existing methods on average.

5. Analysis

We have evaluated our methods with the existing ones on real datasets with different local probabilities. The results show the effectiveness of our framework in these cases. When using deep learning models in real applications, the



Object	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
label	0	1	2	3	4	5	6	7	8	9
probability	0	0	0	0	0.50	0.02	0	0.48	0	0
Local probability	0.02	0.02	0.3	0	0.01	0.05	0	0.60	0	0

Result = 7

Figure 8: The result by the model is 4 (deer, wrong), but our framework can get the one that is 7 (horse, correct).

Table 7: The introduction of the employed acronyms.

Num	Acronyms	Introduction
1	L-PDL	Our framework, local probability-based deep learning
2	CIFAR-10	Dataset that is introduced in [15–17]
3	Mini-ImageNet	Dataset that is introduced in [21–23]
4	VoVNet-57	A deep learning model that is introduced in [24]
5	VGG16	A deep learning model that is introduced in [25]
6	ResNeSt50	A deep learning model that is introduced in [26]
7	S_n	Presents a sample
8	L_k	Presents a label
9	G_n	Presents the ground truth on S_n
10	Zero20	Means 20% of the labels have zero samples
11	Zero40	Means 40% of the labels have zero samples
12	Zero80	Means 80% of the labels have zero samples
13	L-PDL-joint	Joint cooperation based on our framework, introduced in equation (4)
14	L-PDL-weight	Weighted cooperation based on our framework, introduced in equation (5)
15	Rand (.)	Is the function that outputs random value of probability
16	CC-KL trainable	The existing class-conscious trainable combiner-based KL weights method that is introduced in the work [27]
17	CC-KL trainable with our framework	Our framework on class-conscious trainable combiner-based KL weights method that is introduced in the work [27]

accuracy can be improved by analysing local probability. The local probability can be obtained by the computation on the validation set. In some cases, the local probability can be obtained by the other way, for example, the experience of other users about the probability of the objects in an environment. Based on this kind information, we can draw a conclusion that the object in Figure 4 may not be a "deer" but a "horse" as Figure 8 shows.

Another advantage of using L-PDL is that we do not need to retrain or transfer the models to each local environment for the robustness. This is like a person that follows "when you are in Rome, do as the Romans do." This kind of ability can make a person well live anywhere as soon as possible, which can be called the robustness. In this paper, we also implemented this kind of robustness by using our framework.

As our framework is based on the existing deep learning models, the computational complexity is increased compared with these models. The models should output the probability and there should be a validation set with the ground truth for tuning the parameters, which increases the complexity of managing samples. Furthermore, our framework causes additional cost that is caused by the computation of the cooperation between the local probability and the output of models.

5.1. The Introduction of the Employed Acronyms. We use Table 7 to give the introduction of the employed acronyms in this paper for the reader's convenience.

6. Conclusions

In this paper, we have introduced a novel framework that combines the local probability with the probability of objects. Our framework uses the output of the model to present the probability of objects. Then, this probability conditionally cooperates with the local probability to achieve higher accuracy. Our framework can improve the robustness of the deep learning classification models in an environment. Furthermore, we also applied our framework to the existing fusion operators, which can further increase the accuracy. The evaluation results proved the effectiveness of our framework to the deep learning models and fusion operators on these models. Thus, our framework can be a choice to increase the accuracy in the real applications.

In the future work, we will do research about the deep cooperation between the model probability and the local probability, for example, how we can use the output before the probability of labels. This may include more information about the features of objects, which can correctly present "what the model has seen in the samples." Furthermore, the deep cooperation between our framework and the fusion operators may further increase the accuracy, which is another direction of our future work.

Data Availability

CIFAR-10 and CIFAR-100 are available at http://www.cs.toronto.edu/~kriz/cifar.html. Mini-ImageNet is available at https://www.kaggle.com/whitemoon/miniimagenet.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61802279, 6180021345, 61702281, and 61702366), Natural Science Foundation of Tianjin (Grant nos. 18JCQNJC70300, 19JCTPJC49200, 19PTZWHZ00020, and 19JCYBJC15800), Fundamental Research Funds for the Tianjin Universities (Grant no. 2019KJ019), and the Tianjin Science and Technology Program (Grant no. 19PTZWHZ00020) and in part by the State Key Laboratory of ASIC and System (Grant no. 2021KF014) and Tianjin Educational Commission Scientific Research Program Project (Grant nos. 2020KJ112 and 2018KJ215).

References

[1] J. J. Wu, Y. N. Yu, C. Huang, and K. Yu, "Deep multiple instance learning for image classification and auto-annotation," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3460–3469, Boston, MA, USA, June 2015.

[2] Z. Gao, T. T. Han, L. Zhu, H. Zhang, and Y. Wang, "Exploring the cross-domain action recognition problem by deep feature learning and cross-domain learning," *IEEE Access*, vol. 6, pp. 68989–69008, 2018.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on ImageNet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pp. 1026–1034, Santiago, Chile, December 2015.

[4] X. Li, Y. Guo, and D. Schuurmans, "Semi-supervised zero-shot classification with label representation learning," in *Proceedings of the 2016 IEEE International Conference on Computer Vision*, pp. 4211–4219, Las Vegas, NV, USA, June 2016.

[5] H.-M. Yang, X.-Y. Zhang, F. Yin, and C. L. Liu, "Robust classification with convolutional prototype learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8769–8778, Salt Lake City, UT, USA, June 2018.

[6] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 3, pp. 563–575, 2017.

[7] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé., "Toward effective mobile encrypted traffic classification through deep learning," *Neurocomputing*, vol. 409, pp. 306–315, 2020.

[8] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[9] A. Schumann, S. Gong, and T. Schuchert, "Deep learning prototype domains for person re-identification," in *Proceedings of the 2017 IEEE International Conference on Image Processing*, pp. 1767–1771, Beijing, China, September 2017.

[10] S. W. Park, J. Park, K. Bong et al., "An energy-efficient and scalable deep learning/inference processor with tetra-parallel mimd architecture for big data applications," *IEEE Transactions on Biomedical Circuits & Systems*, vol. 14, pp. 838–848, 2016.

[11] S. Huang, J. Wang, D. Wang, D. M. Zhang, and H. You, "A thread-saving schedule with graph analysis for parallel deep learning applications on embedded systems," *IEEE International Conference on Smart Cloud*, vol. 93, pp. 111–115, 2018.

[12] C. Chen, P. Zhang, H. Zhang et al., "Deep learning on computational-resource-limited platforms: a survey," *Mobile Information Systems*, vol. 2020, no. 4, pp. 1–19, 2020.

[13] I. Karabayir, O. Akbilgic, and N. Tas, "A novel learning algorithm to optimize deep neural networks: evolved gradient direction optimizer (EVGO)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 99, pp. 1–10, 2020.

[14] R. K. Agrawal and A. Juneja, "Deep learning models for medical image analysis: challenges and future directions," in

Big Data Analytics, 7th International Conference, Ahmedabad, India, December 2019.

[15] W. Woods and C. Teuscher, "Fast and accurate sparse coding of visual stimuli with a simple, ultra-low-energy spiking architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2173–2187, 2017.

[16] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 828–841, 2019.

[17] P. C. Lin, M. K. Sun, C. K. Kung, and T. D. Chiueh, "Floatsd: a new weight representation and associated update method for efficient convolutional neural network training," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 99, pp. 267–279, 2019.

[18] J. Chen and Z. Liu, "Mask dynamic routing to combined model of deep capsule network and U-net," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 99, pp. 1–12, 2020.

[19] H. Zhang and M. Xu, "Improving the generalization performance of deep networks by dual pattern learning with adversarial adaptation," *Knowledge-Based Systems*, vol. 200, Article ID 106016, 2020.

[20] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, p. 242, 2017.

[21] Y. Zhang, M. Fang, and N. Wang, "Channel-spatial attention network for fewshot classification," *PLoS One*, vol. 14, no. 12, Article ID e0225426, 2019.

[22] Y. Xie, H. Wang, B. Yu, and C. Zhang, "Secure collaborative few-shot learning," *Knowledge-Based Systems*, vol. 203, Article ID 106157, 2020.

[23] P. Wang, J. Cheng, F. Hao, L. Wang, and W. Feng, "Embedded adaptive cross-modulation neural network for few-shot learning," *Neural Computing & Applications*, vol. 32, no. 2, 2020.

[24] L. Youngwan, H. Joong-won, L. Sangrok, B. Yuseok, and P. Jongyoul, "An energy and GPU- computation efficient backbone network for real-time object detection," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Long Beach, CA, USA, June 2019.

[25] Z. Liu, J. Wu, L. Fu et al., "Improved kiwifruit detection using pre-trained VGG16 with RGB and NIR information fusion," *IEEE Access*, vol. 8, no. 1, pp. 2327–2336, 2020.

[26] H. Zhang, C. Wu, Z. Zhang et al., "ResNeSt: split-attention networks," in *Proceedings of the Computer Vision and Pattern Recognition*, Seattle, WA, USA, June 2020.

[27] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, A. Montieri, and A. Antonio, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[28] C. Gsaxner, B. Pfarrkirchner, L. Lindner, N. Jakse, and J. Egger, "Exploit 18f-FDG enhanced urinary bladder in pet data for deep learning ground truth generation in CT scans," *Spie Medical Imaging*, vol. 37, 2018.

[29] Q. Liu and Z. Y. Wang, "Automatic "ground truth" annotation and industrial workpiece dataset generation for deep learning," *International Journal of Automation and Computing*, vol. 3, pp. 1–12, 2020.

[30] S. Zhang and C. Guan, "Emotion recognition with refined labels for deep learning," in *Proceedings of the 43rd Annual Conference of the Canadian Medical and Biological Engineering*, IEEE, Montreal, Canada, July 2020.

[31] Y. Pu, G. Zhe, R. Henao, Y. Xin, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in Neural Information Processing Systems*, Barcelona, Spain, December 2016.

[32] Y. Z. Lin, Z. H. Nie, and H. W. Ma, "Structural damage detection with automatic feature-extraction through deep learning," *Computer-aided Civil & Infrastructure Engineering*, vol. 232, 2017.

[33] J. Liu, C. Fang, and C. Wu, "A fusion face recognition approach based on 7-layer deep learning neural network," *Journal of Electrical and Computer Engineering*, vol. 2016, pp. 1–7, 2016.

[34] W. Hastomo, A. Karno, N. Kalbuana, A. Meiriki, and A. Sutarno, "Characteristic parameters of epoch deep learning to predict Covid-19 data in Indonesia," *Journal of Physics: Conference Series*, vol. 1, no. 2021, Article ID 12050, 1933.

[35] G. Lim, Y. Cheng, W. Hsu, and M. L. Lee, "Integrated optic disc and cup segmentation with deep learning," in *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, IEEE, San Jose, CA, USA, November 2016.