Local to Global Learning: Gradually Adding Classes for Training Deep Neural Networks

Hao Cheng[1*], Dongze Lian[1*], Bowen Deng[1], Shenghua Gao[1], Tao Tan[2], Yanlin Geng[3†]
[1] School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
[2] Dept of Mathematics and Computer Science, Centre of Analysis, Eindhoven University of Technology
[3] State Key Lab. of ISN, Xidian University, Xi'an 710071, China
{chenghao, liandz, dengbw, gaoshh}@shanghaitech.edu.cn, t.tan1@tue.nl, ylgeng@xidian.edu.cn

Abstract

We propose a new learning paradigm, Local to Global Learning (LGL), for Deep Neural Networks (DNNs) to improve the performance of classification problems. The core of LGL is to learn a DNN model from fewer categories (local) to more categories (global) gradually within the entire training set. LGL is most related to the Self-Paced Learning (SPL) algorithm but its formulation is different from SPL. SPL trains its data from simple to complex, while LGL from local to global. In this paper, we incorporate the idea of LGL into the learning objective of DNNs and explain why LGL works better from an information-theoretic perspective. Experiments on the toy data, CIFAR-10, CIFAR-100, and ImageNet dataset show that LGL outperforms the baseline and SPL-based algorithms.

1. Introduction

Researchers have spent decades to develop the theory and techniques of Deep Neural Networks (DNNs). Now DNNs are very popular in many areas including speech recognition [9], computer vision [16, 20], natural language processing [30] *etc*. Some techniques have been proved to be effective, such as data augmentation [32, 29] and identity mapping between layers [10, 11]. Recently, some researchers have focused on how to improve the performance of DNNs by selecting training data in a certain order, such as *curriculum learning* [3] and *self-paced learning* [17].

Curriculum learning (CL) was first introduced in 2009 by Bengio *et al* [3]. CL is inspired by human and animal learning which suggests that a model should learn samples gradually from a simple level to a complex level. However, the curriculum often involves prior man-made knowledge that is independent of the subsequent learning process. To alleviate the issues of CL, Self-Paced Learning (SPL) [17] was proposed to automatically generate the curriculum during the training process. SPL assigns a binary weight to each training sample. Whether or not to choose a sample is decided based on the sample's loss at each iteration of training. Since [17], many modifications of the basic SPL algorithm have emerged. Moreover, [13] introduces a new regularization term incorporating both easiness and diversity in learning. [12] designs soft weighting (instead of binary weight) methods such as linear soft weighting and logarithmic soft weighting. [14] proposes a framework called self-paced curriculum learning (SPCL) which can exploit both prior knowledge before the training and information extracted dynamically during the training.

However, some SPL-based challenges still remain: 1) It is hard to define simple and complex levels. CL defines these levels according to prior knowledge, which needs to be annotated by human. This process is extremely complicated and time-consuming, especially when the number of categories is large. Another solution is to choose simple samples according to the loss like SPL. However, the samples' losses are related to the choice of different models and hyper-parameters, since it is likely that the loss of a sample is large for one model but small for another; 2) SPL-

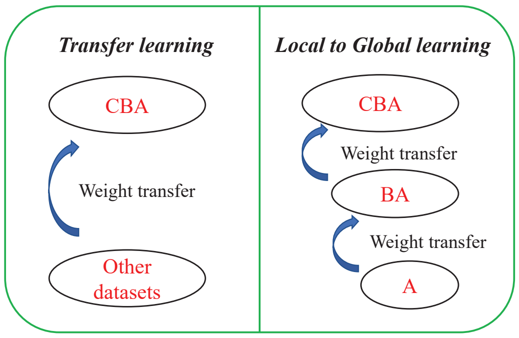

Figure 1. An illustration of the difference between transfer learning and LGL. A, B and C denote three classes in the training set.

[1*] Equal contribution; [3†] Corresponding author

based algorithms always bring additional hyper-parameters. One must tune hyper-parameters very carefully to generate a good curriculum, which increases the difficulty of training the model.

To address the above two problems, we propose a new learning paradigm called Local to Global Learning (LGL). LGL learns the neural network model from fewer categories (local) to more categories (global) gradually within the entire training set, which brings only one hyper-parameter (inverse proportional to how many classes to add at each time) to DNN. This new hyper-parameter is also easy to be tuned. Generally, we can improve the performance of DNN by increasing the value of the new hyper-parameter.

The intuition behind LGL is that the network is usually better to memorize fewer categories[1] and then gradually learns from more categories, which is consistent with the way people learn. The formulation of LGL can be better understood by comparing it with *transfer learning* shown in Figure 1. In transfer learning, the initial weights of DNNs are transferred from another dataset. But in LGL, the initial weights of DNNs are transferred from the self-domain without knowledge of other datasets. The traditional methods randomly initialize the weights, which do not consider the distributions of the training data and may end up with a bad local minimum; whereas LGL initializes the weights which capture the distributions of the trained data. So LGL can be also seen as an *initialization strategy of DNNs*. In this paper, we explain the methodology of LGL from the mathematical formulation in detail. Instead of concentrating on sample loss (as in SPL), we pay attention to training DNN effectively by continually adding a new class to DNN.

There are three main contributions from this paper:

- We propose a new learning paradigm called Local to Global Learning (LGL) and incorporate the idea of LGL into the learning objective of DNN. Unlike SPL, LGL guides DNN to learn from fewer categories (local) to more categories (global) gradually within the entire training set.
- From an information-theoretic perspective (conditional entropy), we confirm that LGL can make DNN more stable to train from the beginning.
- We perform the LGL algorithm on the toy data, CIFAR-10, CIFAR-100, and ImageNet dataset. The experiments on toy data show that the loss curve of LGL is more stable and the algorithm converges faster than the SPL algorithm when the model or data distributions vary. The experiments on CIFAR-10, CIFAR-100 and ImageNet show that the classification accuracy of LGL outperforms the baseline and SPL-based algorithms.

[1]This means that a higher classification performance is achieved compared to classifying more categories.

2. Related Work

SPL has been applied to many research fields. [24] uses SPL for long-term tracking problems to automatically select right frames for the model to learn. [28] integrates the SPL method into multiple instances learning framework for selecting efficient training samples. [27] proposes multi-view SPL for clustering which overcomes the drawback of stuck in bad local minima during the optimization. [31] introduces a new matrix factorization framework by incorporating SPL methodology with traditional factorization methods. [8] proposes a framework named self-paced sparse coding by incorporating self-paced learning methodology with sparse coding as well as manifold regularization. The proposed method can effectively relieve the effect of nonconvexity. [21] designs a new co-training algorithm called self-paced co-training. The proposed algorithm differs from the standard co-training algorithm that does not remove false labelled instances from training. [18] brings the idea of SPL into multi-task learning and proposes a framework that learns the tasks by simultaneously taking into consideration the complexity of both tasks and instances per task.

Recently, some researchers have combined SPL with modern DNNs. [19] proposes self-paced convolutional network (SPCN) which improves CNNs with SPL for enhancing the learning robustness. In SPCN, each sample is assigned a weight to reflect the easiness of the sample. A dynamic self-paced function is incorporated into the learning objective of CNNs to jointly learn the parameters of CNNs and latent weight variable. However, SPCN seems to only work well on simple dataset like MNIST. [2] shows that CNNs with the SPL strategy do not show actual improvement on the CIFAR dataset. [15] shows that when there are fewer layers in the CNN, an SPL-based algorithm may work better on CIFAR. But when the number of layers increases, like for VGG [23], the SPL algorithm performs almost equal to that of traditional CNN training. [25] proposes a variant form of self-paced learning to improve the performance of neural networks. However, the method is complicated and can not be applied to large dataset like ImageNet. Based on the above analysis of SPL's limitations, we develop a new data selection method for CNNs called Local to Global Learning (LGL). LGL brings only one hyper-parameter (easy to be tuned) to the CNN and performs better than the SPL-based algorithms.

There are still two learning regimes similar to our work called Active Learning [6] and Co-training [4] which also select the data according to some strategies. But in active learning, the labels of all the samples are not known when the samples are chosen. Co-training deals with semi-supervised learning in which some labels are missing. Thus, these two learning regimes differ in our setting where the labels of all the training data are known.

3. Self-Paced Learning

Let us first briefly review SPL before introducing LGL. Let $L(y_i, g(\mathbf{x}_i, \mathbf{w}))$ denote the loss of the ground truth label y_i and estimated label $g(\mathbf{x}_i, \mathbf{w})$, where $\mathbf{w}$ represents the parameters of the model. The goal of SPL is to jointly learn the model parameters $\mathbf{w}$ and latent variable $\mathbf{v} = [v_i, \dots, v_n]^T$ by minimizing:

$$\min_{\mathbf{w},\mathbf{v}\in[0,1]^n} \sum_{i=1}^{n} v_i L(y_i, g(\mathbf{x}_i, \mathbf{w})) + f(\mathbf{v}; \lambda) \ . \tag{1}$$

In the above, $\mathbf{v}$ denotes the weight variables reflecting the samples' importance; λ is a parameter for controlling the learning pace; f is called the self-paced function which controls the learning scheme. SPL-based algorithms are about to modify f to automatically generate a good curriculum during the learning process.

In the original SPL algorithm [17], $\mathbf{v} \in \{0, 1\}^n$, and f is chosen as:

$$f(\mathbf{v}; \lambda) = -\lambda ||\mathbf{v}||_1 = -\lambda \sum_{i=1}^{n} v_i \ . \tag{2}$$

Another popular algorithm is called SPLD (self-paced learning with diversity) [13] which considers both $||\mathbf{v}||_1$ and the sum of group-wise $||\mathbf{v}||_2$. In SPLD, f is chosen as:

$$f(\mathbf{v}; \lambda, \gamma) = -\lambda ||\mathbf{v}||_1 - \gamma ||\mathbf{v}||_{2,1} \ . \tag{3}$$

In general, iterative methods like Alternate Convex Search (ACS) are used to solve (1), where $\mathbf{w}$ and $\mathbf{v}$ are optimized alternately. When $\mathbf{v}$ is fixed, we can use existing supervised learning methods to minimize the first term in (1) to obtain the optimal $\mathbf{w}^*$. Then when $\mathbf{w}$ is fixed, and suppose f is adopted from (2), the global optimum $\mathbf{v}^* = [v_i^*, \dots, v_n^*]^T$ can be explicitly calculated as:

$$\mathbf{v}_i^* = \begin{cases} 1, & L(y_i, g(\mathbf{x}_i, \mathbf{w})) < \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

From (4), λ is a parameter that determines the difficulty of sampling the training data: When λ is small, 'easy' samples with small losses are sent into the model to train; When we gradually increase λ, the 'complex' samples will be provided to the model until the entire training set is processed.

From the above analysis, the key step in an SPL algorithm is to adjust the hyper-parameter λ at each iteration of training. In reality, however, we do not know the loss of each sample before training. Therefore sometimes one needs to run a baseline (a training algorithm without SPL) first to observe the average loss at each iteration and then set an empirical value for λ to increase. For more complex algorithms like SPLD from (3), researchers must control two parameters λ and γ, which makes the training difficult. To avoid the difficulty of tuning parameters in the SPL-based algorithms, we introduce our easy-to-train LGL algorithm.

4. Local to Global Learning

This section goes as follows: In Section 4.1, LGL is defined; In Section 4.2, we incorporate the idea of LGL into the learning objective of DNN and propose the training algorithm of LGL; In Section 4.3, we analyze the complexity of training time in LGL; In Section 4.4, we introduce selection strategies of LGL; In Section 4.5, we explain why LGL works better from an information-theoretic perspective.

4.1. Definition of LGL

Consider a K-label classification problem, where the set of labels is $\mathcal{K} = \{1, 2, \dots, K\}$. Let the training dataset be $\mathcal{D} = \{(X_j, Y_j) : j = 1, \dots, N\}$, where N is the number of training samples, X_j is the j-th data point and Y_j is its label. The i-th cluster, denoted $X_{\{i\}}$, is the set of X_j whose label is i, $i \in \mathcal{K}$. In the remaining, we will always use the word 'cluster' to represent 'class' in the training set.

Definition 1 *For a K-cluster learning problem, the local to global learning methodology is to iteratively train DNN by adding a new cluster to the training set at each time. This process can be described by a learning sequence $s = [s_1, s_2, \dots, s_K]$, which is a permutation of the labels of the clusters, to represent the learning order.*

Each cluster $X_{\{i\}}$ 'means' a *local* area in LGL. From the definition, LGL is a *cluster-based* data selection method. Unlike traditional DNN training of SPL, LGL learns a DNN model gradually from fewer clusters to more clusters within the entire training set. Also, the learning order of clusters may affect the performance of DNN. To find the order of a learning sequence s, the procedure can be split into two stages:

1. Select a cluster to start with;
2. Select the next cluster gradually based on the trained clusters.

In this paper, the initial cluster is selected randomly since we do not have prior information about all the clusters in the beginning. It is worth noting that LGL is introduced for the classification problems, possible future work can be done to expand LGL to other types of learning regimes such as regression or unsupervised learning.

4.2. General LGL Algorithm in DNN

Consider a K-label classification problem, let $\mathcal{S}$ be a subset of $\mathcal{K} = \{1, 2, \dots, K\}$, and $\{(X_\mathcal{S}, Y_\mathcal{S})\}$ be the data whose labels are in $\mathcal{S}$: $\{(X_\mathcal{S}, Y_\mathcal{S})\} = \{(X_j, Y_j) : Y_j \in \mathcal{S}\}$.

The learning objective of a traditional DNN can be written as:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w}, X_\mathcal{K}, Y_\mathcal{K}; \mathbf{w}_{initial}), \tag{5}$$

Algorithm 1 A general training algorithm of LGL

```
Input:
    Dataset $\mathcal{D} = \{(X_n, Y_n) : n = 1, \dots, N\}$ with $K$ labels, $\mathcal{K} = \{1, 2, \dots, K\}$; Initial learning rate $lr_{initial}$; Initial weight
    $\mathbf{w}_0^* = \mathbf{w}_{initial}$; The function of seletion strategy $f$; The initial data $\mathcal{G} = \{(X_{\{i\}}, Y_{\{i\}})\}$, where $i \in \mathcal{K}$.
Iteration:
    for $k = 1$ to $K$ do
        $\mathbf{w}_k^* = \arg\min_{\mathbf{w}} L(\mathcal{G}, \mathbf{w}_{k-1}^*)$    // Train DNN on $\mathcal{G}$ until convergence
        $j^* = f(\mathcal{D} \setminus \mathcal{G}, \mathbf{w}_k^*)$    // Select a new cluster from the untrained clusters
        $\mathcal{G} = \mathcal{G} \cup \{(X_{\{j^*\}}, Y_{\{j^*\}})\}$    // Add selected cluster to $\mathcal{G}$
        $lr = lr_{initial}$    // Adjust learning rate to initial learning rate since new cluster included
    end for
Output:
    $\mathbf{w}_K^*$
```

where L denotes the loss function, $\mathbf{w}_{initial}$ specifies the initial weight that the minimization process starts with. For LGL, at the k-th step, starting with a subset $\mathcal{S}_{k-1}$ of $\mathcal{K}$, the learning objective is:

$$\begin{aligned} \mathbf{w}_k^* = \arg\min_{\mathbf{w}} \quad & L(\mathbf{w}, X_{\mathcal{S}_k}, Y_{\mathcal{S}_k}; \mathbf{w}_{k-1}^*) \\ \text{s.t.} \quad & i^* = f(X_{\mathcal{S}_{k-1}^C}, Y_{\mathcal{S}_{k-1}^C}, \mathbf{w}_{k-1}^*), \\ & \mathcal{S}_k = \mathcal{S}_{k-1} \cup \{i^*\}. \end{aligned} \tag{6}$$

In the above, $\mathbf{w}_{k-1}^*$ denotes the weight or knowledge on the trained clusters at the beginning of the k-th step; $\mathcal{S}_{k-1}^C$ denotes the set of classes not in S_{k-1}; the function f denotes the selection strategy that selects the label of a cluster from the remaining untrained clusters utilizing existing knowledge .

So instead of minimizing the loss function across all the clusters like (5), we iteratively minimize the loss function by adding a new cluster to DNN. A general training algorithm of LGL for DNNs in classification problems is shown in Algorithm 1.

A key step in LGL algorithm is that before we add new clusters to DNN, we must train DNN until convergence which means the loss of DNN will not increase. This separates LGL from the other types of SPL-based algorithms which only train DNN several epochs before adding new training data. Also, suppose there are K classes in the training set, we can fix the number of nodes of the softmax layer to be K and unchanged when performing LGL. But when DNN has many layers, the softmax layer needs to be reinitialized when we add new clusters to DNN. The reason is that since the output of the softmax layer is a probability vector which adds up to 1, many weights (the weights connected to the nodes which belong to the untrained classes) of the softmax layers are approximately zero when we train DNN to convergence. Reinitializing the softmax layer can make sure that the gradients will not saturate. So in practice, when DNN has many layers, we reinitialized the weights (connected to the nodes which belong to the untrained classes) of the softmax layer before we add new clusters to DNN.

4.3. Time Complexity Analysis

From Algorithm 1, if there are K labels in a dataset, we need to minimize the loss function for K times. For a large K, the training time of LGL is very long. To alleviate this issue, instead of adding one untrained cluster at each step, we add more clusters to DNN. Suppose each cluster has an equal number of training samples. We continually add $\frac{K}{m}$ clusters to DNN and train DNN for an equal number of epochs at each step when performing LGL, where K is the number of clusters in the training set (rounding up $\frac{K}{m}$ when it is a non-integer value). Then the training time of LGL is $\frac{1}{2}t(m+1)$, where t is the training time of the baseline for DNN (the derivation is left to the supplementary material).

The training time of LGL is linear in t or m. For a comparison, SPL also increases the training time, but in a different way. In SPL, we need to run DNN on all the training samples to select the ones with lower losses at each iteration. If the training set has a large number of samples, such as ImageNet, the training time of SPL is intolerable huge. But for LGL, we can choose a proper m to control the training time. Thus LGL is more applicable than SPL for classification problems with a large training set.

4.4. Selection Strategy in LGL

In this paper, we propose three selection strategies based on the existing trained clusters:

1. Randomly select a cluster from remaining clusters;
2. Select the most dissimilar cluster to the trained clusters;
3. Select the most similar cluster to the trained clusters.

To quantify the ‘dissimilarity’ between each remaining cluster and the trained clusters, we use the conditional entropy $H_{\mathbf{w}}(T|X_{\{i\}})$, where $\mathbf{w}$ is the current weight of DNN,

$X_{\{i\}}$ is the untrained cluster with label i, T is the softmax output of DNNs. $H_{\mathbf{w}}(T|X_{\{i\}})$ can be written as:

$$H_{\mathbf{w}}(T|X_{\{i\}}) = \sum_{x \in X_{\{i\}}} p(x) H_{\mathbf{w}}(T|X_{\{i\}} = x) \tag{7}$$

$$= - \sum_{x \in X_{\{i\}}} p(x) \sum_{t \in T} p(t|x) \log p(t|x). \tag{8}$$

Suppose $X_{\{i\}}$ has M samples and the model has been trained on L clusters, then for each $x \in X_{\{i\}}$, $p(x) = \frac{1}{M}$. Since T denotes the output of the softmax layer, it has L possible values to take. For each $t \in \mathcal{T}$, $p(t|x)$ denotes the probability of x belonging to the class with respect to t. From the property of entropy, $H_{\mathbf{w}}(T|X_{\{i\}})$ achieves the maximum if for each $x \in X_{\{i\}}$, $p(t|x)$ follows the uniform distribution on T; $H_{\mathbf{w}}(T|X_{\{i\}})$ achieves the minimum if for each $x \in X_{\{i\}}$, there exists a t such that $p(t|x) = 1$. Suppose the model has been trained on L clusters successfully, then for each x in the trained clusters, $p(t|x) \approx 1$ for t being the label of x, which leads to a small $H_{\mathbf{w}}(T|x)$. Now consider an untrained cluster $X_{\{i\}}$:

- If $X_{\{i\}}$ is similar to a trained cluster, then $H_{\mathbf{w}}(T|X_{\{i\}})$ will be small from the above analysis.

- If $X_{\{i\}}$ is dissimilar to any trained cluster, then $p(t|x)$ tends to follow a uniform distribution for $x \in X_{\{i\}}$, hence $H_{\mathbf{w}}(T|X_{\{i\}})$ will be large.

So we use $H_{\mathbf{w}}(T|X_{\{i\}})$ to represent the dissimilarity between the untrained cluster $X_{\{i\}}$ and the trained clusters. A large $H_{\mathbf{w}}(T|X_{\{i\}})$ means the untrained cluster $X_{\{i\}}$ has less similarity to the trained clusters. Suppose we select the most dissimilar cluster to the trained clusters, the function of selection strategy f in (8) of the paper is:

$$i^* = \arg\max_{i \in \mathcal{S}^C_{k-1}} H_{\mathbf{w}^*_{k-1}}(T|X_{\{i\}}). \tag{9}$$

There are two reasons why we choose $H_{\mathbf{w}}(T|X_{\{i\}})$ as the dissimilarity measure. 1) Computational complexity: The other similarity measures (such as calculating mutual information or Euclidean distance between two clusters) require huge computations, especially when the data dimension is high like images, whereas calculating $H_{\mathbf{w}}(T|X_{\{i\}})$ using (8) is very fast. 2) Existing knowledge: The other similarity measures do not consider the existing knowledge (DNN's weight $\mathbf{w}$) of the trained clusters, whereas $H_{\mathbf{w}}(T|X_{\{i\}})$ is calculated by running DNN with its current weight. The experiments on different selection strategies are shown in Section 5.3.

4.5. Information-Theoretic Perspective of LGL for Deep Neural Networks

In recent years, there are some works that utilize information theory to explain DNNs. Schwartz-Ziv and Tishby [26, 22, 5] calculate the mutual information $I(X;T)$, $I(T;Y)$, where X is the input data, Y is the label and T is the layer output. Then they demonstrate the effectiveness of visualization of neural networks: the information plane reveals that there are two learning stages in network learning process (the first fitting phase and the second compression phase). But mutual information is hard to estimate accurately in the network especially when the layer of network has high dimensions. Thus in the paper, instead of using mutual information, we use the conditional entropy $H(T|X)$ to represent DNN's stability and explain why LGL algorithm works well, where T in $H(T|X)$ is the softmax output of DNN. We state that from an information-theoretic perspective, the benefit of LGL algorithm is that *LGL can lower the initial $H(T|X)$ of DNN to make the training of DNN starts at a more stable state*. Section 5.1 shows the stability of LGL algorithm.

The more detailed explanations are shown in the supplementary material.

5. Experiments

This section goes as follows: In Section 5.1, we show the loss curve of LGL is more stable than the SPL algorithm when the model or data distributions vary; In Section 5.2, we show the validation accuracy of LGL outperforms the baseline and SPL-based algorithms on the CIFAR-10 and CIFAR-100 dataset; In Section 5.3, we compare different selection strategies of LGL; In Section 5.4, we validate LGL on the ImageNet dataset. The code is available at:

`https://github.com/piratehao/Local-to-Global-Learning-for-DNNs`

5.1. Toy Data

In this experiment, we not only implement SPL but also SPL_INV (the inversion of SPL algorithm, learning samples from complex to easy) for comparison. The baseline is the traditional DNN without the use of the SPL or LGL strategy. The learning rate is 0.01 and remains fixed. The optimizer is the basic SGD. The datasets are generated from two-dimension Gaussian distributions. There are 3 labels in the toy dataset corresponding to different means of Gaussian distributions. The covariance matrix of each Gaussian distribution is identity. The results of this experiment are shown in Figure 2. Since clusters in the dataset are mutually symmetric, we randomly add clusters to DNN when performing LGL (first train 2 clusters, then add another cluster to the model). Figure 2 exhibits some interesting phenomena:

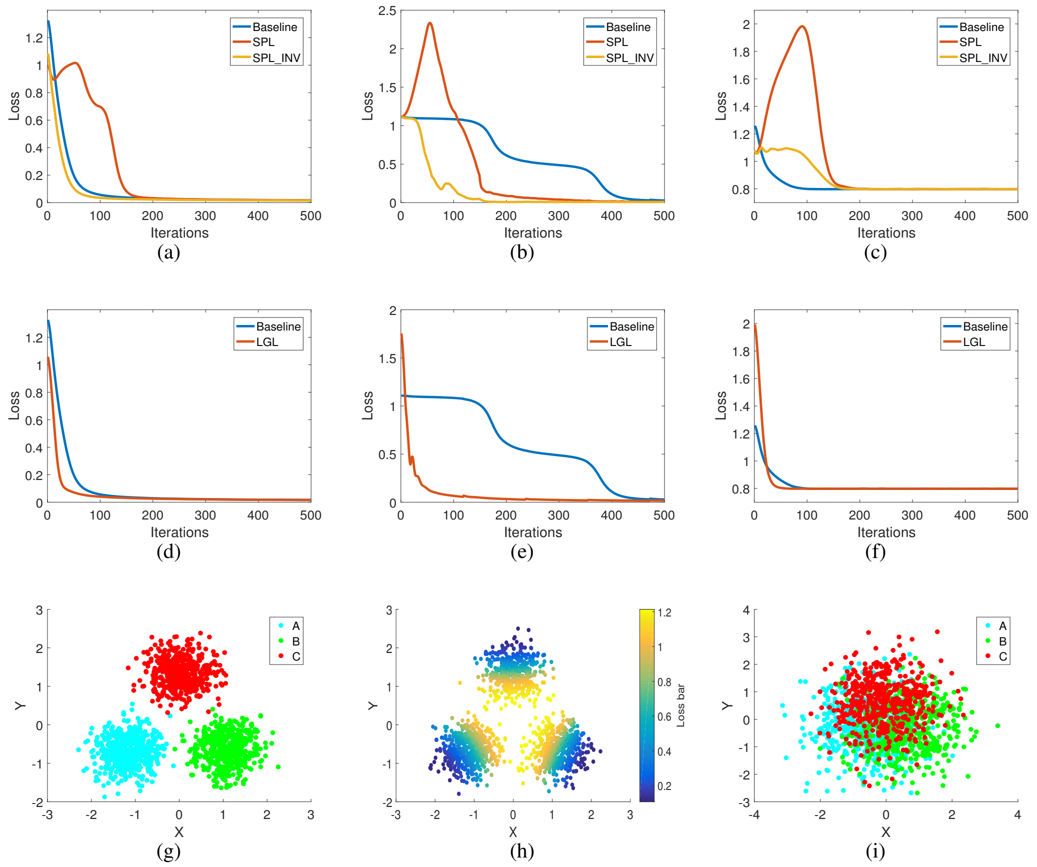

Figure 2. This figure has 9 sub-figures. For (a, d) and (b, e), the dataset is the same and chosen from (g); the model of (a, d) is a fully connected network (FC) with one hidden layer; while the model of (b, e) is a FC with three hidden layers. For (a, d) and (c, f), the model is the same; the dataset of (a, d) is chosen from (g), while the dataset of (c, f) is chosen from (i). (Best viewed in color)

- Sub-figure (a) shows that SPL_INV actually converges faster than the baseline. To see the reason for this phenomenon, we visualize the data according to their losses during the training process. The result can be seen in sub-figure (h): the samples with large training losses are very close to the other clusters. These samples are more 'informative' which means that they are sufficient for the model to classify only these 'complex' samples to get a wonderful classifier. Since SPL select the most 'uninformative' samples in the beginning, these samples actually mislead DNN to learn. From sub-figure (a), the loss of SPL increases at first iterations. Thus the SPL strategy converges slower than the baseline.

- Compared to (a), sub-figure (b) shows that the SPL strategy converges faster than the baseline when the number of hidden layers of DNN increases. The reason for this is that the SPL strategy can be seen as a continuation method for dealing with minimizing non-convex functions [3]. So when the number of hidden layers increases, SPL can prevent the model from arriving at a local minimum. SPL_INV still performs better, however, than SPL.

- Compared to (a), sub-figure (c) shows that with varied data distributions (clusters' intersection creates a confusion area in two-dimension space shown in (i)), even SPL_INV performs worse than baseline. The reason is that, for SPL_INV, the 'informative' samples now lie in the area of confusion, which makes it hard for the DNN to recognize the true labels.

- From the experiments, SPL and SPL_INV may not be the best general strategies for DNNs, whereas for LGL, (d), (e) and (f) show that when trained on all the training data, the loss curve of LGL is more stable and converges faster when the structure of the learning model or data distributions vary.

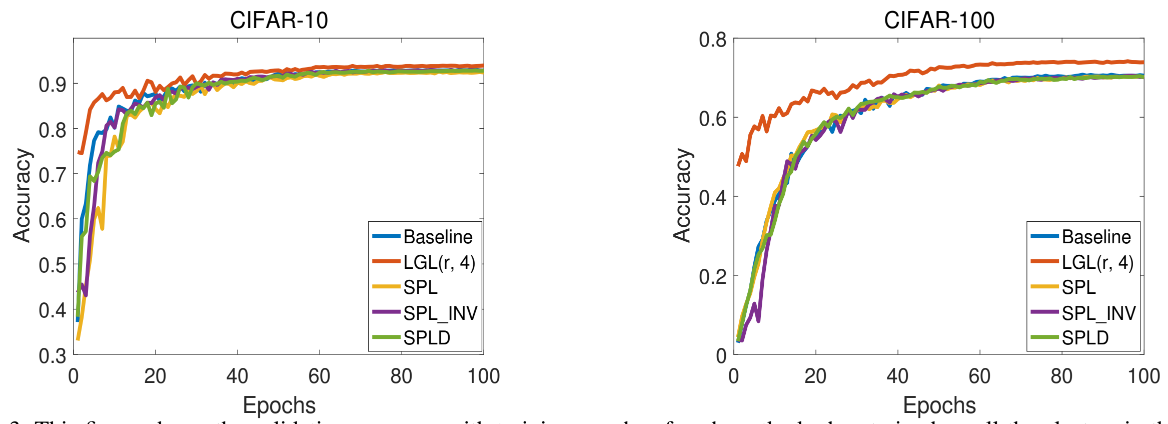

Figure 3. This figure shows the validation accuracy with training epochs of each method when trained on all the clusters in the dataset. (Best viewed in color)

CIFAR-10	Methods	Baseline	SPL	SPL_INV	SPLD	LGL$(r,2)$	LGL$(r,3)$	LGL$(r,4)$
	Accuracy	0.9303	0.9260	0.9296	0.9285	0.9311	0.9359	**0.9397**
CIFAR-100	Methods	Baseline	SPL	SPL_INV	SPLD	LGL$(r,2)$	LGL$(r,3)$	LGL$(r,4)$
	Accuracy	0.7111	0.703	0.7050	0.7025	0.7267	0.7342	**0.7417**

Table 1. The table records the validation accuracy of each method on CIFAR-10 and CIFAR-100. LGL(r,m) means the selection strategy is random and we add $\frac{K}{m}$ clusters to DNN at each step (which means we minimize the loss function for m times). SPLD is an SPL-based algorithm from [13].

5.2. CIFAR-10 and CIFAR-100

In this experiment, we show the validation accuracies of the LGL and SPL algorithms on the CIFAR-10 and CIFAR-100 dataset. CIFAR-10 has 10 classes and the classes are mutually exclusive. CIFAR-100 has 100 classes which can be grouped into 20 superclasses. Each superclass has 5 similar classes. Traditional DNN training without SPL and LGL strategy is taken as the baseline. The hyper-parameters' setting is as follows: mini-batch size (128), initial learning rate (0.05), learning rate decay (0.95), momentum (0.9), weight decay (0.0005), number of epochs (100), optimizer (SGD). The hyper-parameters' setting of the LGL algorithm is the same as that used for the baseline for equal comparison. VGG-16 is employed as the model for all the methods in each dataset. The results are shown in Table 1 and Figure 3.

From Table 1 and Figure 3, the SPL-based algorithms do not perform better than the baseline; indeed they perform worse. This observation is consistent with [2]. Since CIFAR-10 only has 10 classes, the LGL algorithm does not show major improvement. For CIFAR-100, the LGL algorithm significantly outperforms the baseline and SPL-based algorithms. We also find that if we increase the number of times to minimize the loss function of DNN, the validation accuracy increases which can be seen in Table 1. LGL$(r,4)$ performs better than LGL$(r,3)$ and LGL$(r,2)$. Furthermore, we get 0.7491 when performing LGL$(r,20)$ on CIFAR-100 dataset. Thus there exists a trade-off between the training time and validation accuracy. The validation curve from Figure 3 shows that LGL$(r,4)$ almost always achieves a better accuracy than the baseline and SPL-based algorithms at each epoch.

5.3. Comparison of Selection Strategies in LGL

In this experiment, we compare three selection strategies (from Section 4.4) on CIFAR-10 and CIFAR-100: 1) Randomly select clusters from remaining clusters; 2) Select the most dissimilar clusters to the trained clusters; 3) Select the most similar clusters to the trained clusters.

We first perform an experiment on CIFAR-100 to show that $H_{\mathbf{w}}(T|X_{\{i\}})$ can represent the dissimilarity between the trained clusters and the untrained cluster. The result is shown in Table 2. We can see that in Table 2, for example, Oak (with lower $H_{\mathbf{w}}(T|X_{\{i\}})$, second column) is very similar to Forest, while Spider (with higher $H_{\mathbf{w}}(T|X_{\{i\}})$, right column) is very dissimilar to any one in the left column.

With this similarity measure, we perform experiments on CIFAR-10 and CIFAR-100 to test different selection strategies of LGL. The initial clusters for three methods are the same to exclude the influences of initialization. The result is shown in Table 3. Surprisingly, the performance of each selection strategy in LGL does not vary too much. One hypothesis is that this phenomenon is related to the inherent generalization properties of DNN. The main advantage of LGL is training DNN to learn from fewer clusters to more clusters gradually to iteratively build good initial weights,

Trained Clusters by DNN	Clusters with lowest $H_{\mathbf{w}}(T\|X_{\{i\}})$	Clusters with highest $H_{\mathbf{w}}(T\|X_{\{i\}})$
Forest, Cloud, Bottles, Bowls, Cans	Oak, Willow, Plain, Maple, Sea	Spider, Bridge, Lizard, Beetle, Wolf
Bicycle, Bus, Bear, Lion, Leopard	Pickup Trank, Street Car, Whale, Cloud, Skyscraper	Road, Trout, Table, Roses, Baby

Table 2. DNN is trained on the clusters from the left column. The middile and right columns record the clusters we select from the remaining untrained clusters by using $H_{\mathbf{w}}(T|X_{\{i\}})$.

CIFAR-10	Methods	LGL$(r,3)$	LGL$(d,3)$	LGL$(s,3)$
	Accuracy	0.9359	0.9373	0.9363
CIFAR-100	Methods	LGL$(r,3)$	LGL$(d,3)$	LGL$(s,3)$
	Accuracy	0.7342	0.7354	0.7323

Table 3. The table records the validation accuracy of different LGL algorithms on the CIFAR-10 and CIFAR-100 dataset. LGL(r) means the selection strategy is random; LGL(d) means dissimilar clusters to trained ones are selected; LGL(s) is the opposite of LGL(d) which selects similar clusters.

but the order of clusters seems to affect DNN very little. DNN may have great capability to memorize the features of the training data regardless of the order (this may also be the reason why some SPL-based algorithms perform almost equal to the baseline). A full exploration of this phenomenon is an interesting future topic.

5.4. ImageNet

To test the performance of LGL on larger datasets, we also validate LGL on the ImageNet [7], an image dataset with more than 14 million images. Since SPL sorts all the training samples at each iteration which is training time intolerable for ImageNet, we compare LGL with the baseline (traditional DNN training without LGL) for two models (VGG-16 and ResNet-50). The hyper-parameters' setting is consistent with official PyTorch setting for ImageNet [1]: mini-batch size (256), initial learning rate (0.1), learning rate decay (0.1 at every 30 epochs), momentum (0.9), weight decay (0.0001), number of epochs (90), optimizer (SGD). The result is shown in Table 4.

In Table 4, for each model, LGL performs better than the baseline. Thus compared to the SPL algorithm, LGL can also handle a large dataset.

Models	Baseline	LGL$(r,2)$	LGL$(r,3)$
VGG-16	71.846	72.534	**72.912**
ResNet-50	75.290	75.858	**76.166**

Table 4. The table records Top-1 Accuracy of LGL and the baseline on the ImageNet dataset.

6. Conclusion and Discussion

We propose a new learning paradigm called Local to Global Learning (LGL). The core of LGL is to iteratively minimize the learning objective of DNN by adding new clusters. LGL can be seen as an initialization strategy. Unlike transfer learning, the initial weights of DNN are transferred from own dataset. The new paradigm was shown to be superior to the traditional Self-Paced algorithm in terms of stability and final classification accuracy. We also explain LGL from an information-theoretic perspective: LGL can lower the initial entropy of the network and make the training of DNN starts at a more stable state. From our work, we believe that LGL has big potentials in the field of deep learning.

There are some future investigations to be performed:

- To further explore why the three selection strategies in the paper affect the performance of DNN very little, or is there any other selection strategy that could actually improve the performance of DNN than random selection (for example, a selection strategy that put all the classes in a 'hierarchy' form)?
- We apply LGL for classification problems in this paper. Since LGL can be seen as an initialization strategy, it is a valuable direction to apply LGL to other types of learning problems like regression and unsupervised learning in DNN. The key is to pre-define the local areas of the training set.

Acknowledgements

This project is supported by NSFC (No. 61601288).

References

[1] https://github.com/pytorch/examples/blob/master/imagenet/main.py.

[2] Vanya Avramova. Curriculum learning with deep convolutional neural networks. *Master's thesis*, 2015.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

[4] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh Annual Conference on Computational Learning Theory*, pages 92–100. ACM, 1998.

[5] Hao Cheng, Dongze Lian, Shenghua Gao, and Yanlin Geng. Evaluating capability of deep neural networks for image classification via information plane. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 168–182, 2018.

[6] David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, 2009.

[8] Xiaodong Feng, Zhiwei Tang, and Sen Wu. Robust sparse coding via self-paced learning. *arXiv preprint arXiv:1709.03030*, 2017.

[9] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 3, 2017.

[12] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 547–556. ACM, 2014.

[13] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

[14] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, volume 2, page 6, 2015.

[15] Tae-Hoon Kim and Jonghyun Choi. Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904v3*, 2018.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances In Neural Information Processing Systems*, pages 1097–1105, 2012.

[17] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[18] Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, pages 2175–2181, 2017.

[19] Hao Li and Maoguo Gong. Self-paced convolutional neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2110–2116. AAAI Press, 2017.

[20] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016.

[21] Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. Self-paced co-training. In *International Conference on Machine Learning*, pages 2275–2284, 2017.

[22] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] James Steven Supancic III and Deva Ramanan. Self-paced learning for long-term tracking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2379–2386. IEEE, 2013.

[25] Vithursan Thangarasa and Graham W Taylor. Self-paced learning with adaptive deep visual embeddings. *arXiv preprint arXiv:1807.09200*, 2018.

[26] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

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

[28] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 594–602, 2015.

[29] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[30] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[31] 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.

[32] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.