Label-similarity Curriculum Learning

Ürün Dogan[1], Aniket Anand Deshmukh[1], Marcin Bronislaw Machura[1], and Christian Igel[2]

[1] Microsoft Ads, Microsoft, Sunnyvale, CA, USA
udogan@microsoft.com, andeshm@microsoft.com, mmachura@microsoft.com
[2] Department of Computer Science, University of Copenhagen, Denmark
igel@di.ku.dk

Abstract. Curriculum learning can improve neural network training by guiding the optimization to desirable optima. We propose a novel curriculum learning approach for image classification that adapts the loss function by changing the label representation.
The idea is to use a probability distribution over classes as target label, where the class probabilities reflect the similarity to the true class. Gradually, this label representation is shifted towards the standard one-hot-encoding. That is, in the beginning minor mistakes are corrected less than large mistakes, resembling a teaching process in which broad concepts are explained first before subtle differences are taught.
The class similarity can be based on prior knowledge. For the special case of the labels being natural words, we propose a generic way to automatically compute the similarities. The natural words are embedded into Euclidean space using a standard word embedding. The probability of each class is then a function of the cosine similarity between the vector representations of the class and the true label.
The proposed label-similarity curriculum learning (LCL) approach was empirically evaluated using several popular deep learning architectures for image classification tasks applied to five datasets including ImageNet, CIFAR100, and AWA2. In all scenarios, LCL was able to improve the classification accuracy on the test data compared to standard training. Code to reproduce results is available at https://github.com/speedystream/LCL.

Keywords: Curriculum Learning; Deep Learning; Multi-modal Learning, Classification

1 Introduction

When educating humans, the teaching material is typically presented with increasing difficulty. *Curriculum learning* adopts this principle for machine learning to guide an iterative optimization method to a desirable optimum. In curriculum learning for neural networks as proposed by Bengio et al. [1], the training examples are weighted. In the beginning of the training, more weight is put on "easier" examples. The weighting is gradually changed to uniform weights corresponding to the canonical objective function.

Inspired by Bengio et al. [1], we propose *label-similarity curriculum learning* (LCL) as another way to "learn easier aspects of the task or easier sub-tasks, and then gradually increase the difficulty level." If a toddler who is just learning to speak points at a car and utters "cow", a parent will typically react with some teaching signal. However, a young infant is not expected to discriminate between a cheetah and a leopard, and mixing up the two would only lead to a very mild correction signal – if at all. With increasing age, smaller errors will also be communicated.

We transfer this approach to neural network training for classification tasks. Instead of a one-hot-encoding, the target represents a probability distribution over all possible classes. The probability of each class depends on the similarity between the class and the true label. That is, instead of solely belonging to its true class, each input can also belong to similar classes to a lesser extent. Gradually, this label representation is shifted towards the standard one-hot-encoding, where targets representing different classes are orthogonal. In the beginning of training, the targets of inputs with labels `cheetah` and `leopard` should almost be the same, but always be very different from `car`. During the training process, the label representation is gradually morphed into the one-hot encoding, decreasing the entropy of the distribution encoded by the target over time. That is, in the beginning small mistakes – in the sense that similar classes are mixed up – are corrected less than big mistakes, resembling a teaching process in which broad concepts are explained first before subtle differences are taught.

The question arises how to define a proper similarity between classes. One can get a label-similarity matrix based on prior knowledge or some known structure. For the case where the similarity is not explicitly given and the labels correspond to natural language words, we propose a way to automatically infer a representation that reflects semantic similarity. We map the labels into a Euclidean space using a word embedding. Concretely, this is done by applying a generic document embedding to a document explaining the label (its Wikipedia entry). Then the cosine similarities between the vector representations of the label and all possible classes are computed. Based on these values, a distribution over the possible classes is defined which serves as the learning target.

Our way to define the target representation resembles the idea of *hierarchical loss functions* [27,32] ("We define a metric that, inter alia, can penalize failure to distinguish between a sheepdog and a skyscraper more than failure to distinguish between a sheepdog and a poodle." [32]). However, there are two decisive differences. First, we propose to gradually shift from a "hierarchical loss" to a "flat loss". Second, unlike in [32], our approach does not necessarily presume a given hierarchy. When dealing with natural language labels, we propose a way to automatically infer the similarity from a generic word embedding under the assumption that exploiting semantic similarity can be helpful in guiding the learning process.

For evaluating label-similarity curriculum learning (LCL), we need data with some structure in the label space that curriculum learning can exploit. Furthermore, there should be sufficiently many classes and the task should not

be easy to learn. To get label similarity based on word embeddings, we need a dataset with natural language labels. In this study, we focus on three popular benchmark datasets, ImageNet [4], CIFAR100 [17], and Animals with Attributes (AwA) [33]. To show the generality of our approach, we consider different deep learning architectures, and also different preprocessing and learning processes. The time schedule for increasing the "difficulty" of the learning task is an obvious hyperparameter, which we carefully study and show to have little importance.

The next section points to related literature and Section 3 introduces the new *label-similarity curriculum learning*. Section 4 describes the experiments and Section 5 the results before we conclude.

2 Related Work

Starting from the work by Bengio et al. [1], a variety of curriculum learning approaches has been studied. However, they all define a curriculum at the level of training examples. For instance, *self-paced learning* by Kumar et al. [18] introduces latent variables for modelling "easiness" of an examples. Graves et al. [10] consider example-based improvement measures as reward signals for multi-armed bandits, which then build stochastic syllabi for neural networks. Florensa et al. [5] study curriculum learning in the context of reinforcement learning in robotics. They propose to train a robot by gradually increasing the complexity of the task at hand (e.g., the robot learns to reach a goal by setting starting points increasingly far from the goal). In recent work, Weinshall et al. [31] consider learning tasks with convex linear regression loss and prove that the convergence rate of a perfect curriculum learning method increases with the difficulty of the examples. In addition, they propose a method which infers the curriculum using transfer learning from another network (e.g., ResNet-50) pretrained on a different task. They train a linear classifier using features extracted from the pretrained model and score each training example using the linear classifier's confidence (e.g., the margin of an SVM). Finally, they train a smaller deep neural network for the transfer learning task following a curriculum based on these scores.

Buciluǎ et al. [2] have proposed compressing a large model into a simple model which reduces space requirements and increases inference speed at the cost of a small performance loss. This idea has been revisited in [13] under the name *knowledge distillation* (KD) and received a significant amount of attention (e.g., [23,25,36,35,22]). KD methods typically require a pretrained model to start with or train a series of models on the same training data. Standard KD considers a teacher network and a student network. The powerful teacher network is used to support the training of the student network which may be less complex or may have access to less data for training. KD is related to curriculum learning methods because the teacher network guides the learning of student networks [13]. A variant of KD, *born again neural network*, trains a series of models, not only one [7].

Deep mutual learning (DML) is also loosely related to our proposed approach [13,38]. In DML, two models solve the same classification problem collaboratively

and are jointly optimised [38]. Each model acts as a teacher for the other model, and each network is trained with two losses. The first loss is the standard cross-entropy between the model's predictions and target labels. The second is a *mimicry loss* that aligns both model's class posteriors with the class probabilities of the respective other model.

Another related approach is *CurriculumNet* [11], a clustering based curriculum strategy for learning from noisy data. CurriculumNet consists of three steps. First, a deep neural network is trained on the noisy label data. Second, features are extracted by using the model trained in the first step. Using clustering algorithms, these features are then grouped into different sets and sorted into easy and difficult examples. Finally, a new deep neural network is trained using example-weighted curriculum learning. Sorting of examples from easy to hard and clustering algorithms add many hyper-parameters (e.g., number of clusters), and one has to train two neural network models of almost the same size.

Our algorithm can be considered as a multi-modal deep learning method, where text data is used for estimating the class similarity matrix to improve image classification. However, it is different from standard multimodal methods as it does not use text data as an input to the deep neural network. The *DeVise* algorithm is a popular multi-modal method which utilizes the text modality in order to learn a mapping from an image classifier's feature space to a semantic space. [6]. DeVise requires a pretrained deep neural network. Furthermore, as stated in [6], it does not improve the accuracy on the original task but aims at training a model for zero-shot learning.

There is an obvious relation between LCL and *label smoothing* (LS) [21], which we will discuss in Section 4.

The computational requirements of KD, DML, and CurriculumNet are significantly higher compared to our method, which is rather simple. Furthermore, our method does not require training more than one model and adds only a single hyper-parameter.

3 Method

We assume a discrete set of training examples $(\boldsymbol{x}_1, c_1), \ldots, (\boldsymbol{x}_\ell, c_\ell) \in \mathcal{X} \times \mathcal{C}$, with input space $\mathcal{X}$ and finite label space $\mathcal{C}$ with cardinality $|\mathcal{C}| = C$. Let $n : \mathcal{C} \to \{1, \ldots, C\}$ be a bijective mapping assigning each label to a unique integer. This allows a straight-forward definition of the one-hot encoding $\boldsymbol{y}_i \in \mathbb{R}^C$ for each training example $(\boldsymbol{x}_i, c_i)$. The j-th component of $\boldsymbol{y}_i$, which is denoted by $[\boldsymbol{y}_i]_j$, equals 1 if $n(c_i) = j$ and 0 otherwise.

3.1 Document embedding for defining label similarity

Our learning curriculum is based on the pairwise similarities between the C classes, which are defined based on the semantic similarity of the class labels. Now assume that the labels are natural language words, for example $\mathcal{C} = \{\ldots, \texttt{flute}, \ldots, \texttt{strawberry}, \ldots, \texttt{backpack}, \ldots\}$. To quantify semantic similarity, we

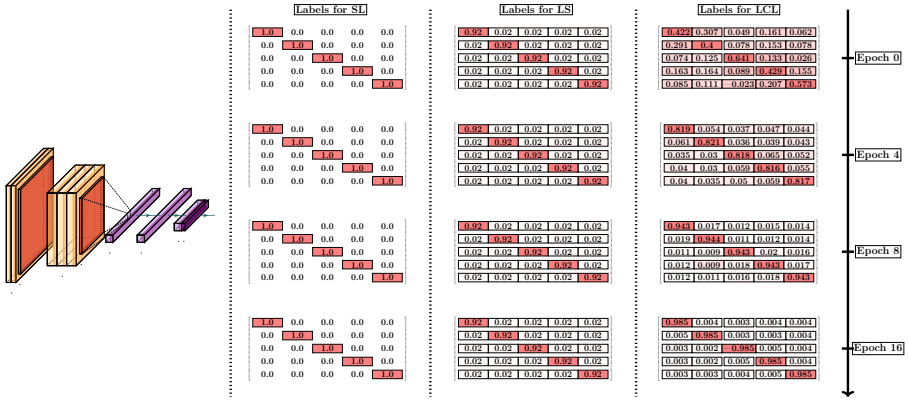

Fig. 1. A deep network (left) trained with three different encodings on a five-class dataset with labels Lion, Tiger, Aircraft Carrier, Alaskan Wolf and Mushroom. The SL (standard learning, see Section 4 for details) column shows the label matrix for one-hot-encoding. When using LS (label smoothing, see Section 4), the loss between the network output and a smoothed version of the label, which does not change over time, is minimized. We propose to use a probability distribution over classes as target label, where the class probabilities reflect the similarity to the true class. This is shown in the LCL column. Unlike LS the proposed label encoding changes during training and converges to the original optimization problem solved when using SL.

embed the natural language labels into Euclidean space using a word embedding [20] such that similar words are nearby in the new representation.

ImageNet labels are given by WordNet identifiers representing synsets, and we redefine the labels for other datasets in a similar way. First, we convert synsest to words, for example, `n02119789` to "fox". Then, we find the Wikipedia article describing each word, for instance, "Orange (fruit)" was selected for `orange`. Then we apply `doc2vec` [19] for mapping the article into Euclidean space. We used a generic `doc2vec` embedding trained on the English Wikipedia corpus. This gives us the encoding $f_{\text{enc}} : \mathcal{C} \to \mathbb{R}^d$, mapping each class label to the corresponding Wikipedia article and then computing the corresponding vector representation using `doc2vec` (with $d = 100$, see below). Now we can compute the similarity between two classes c_i and c_j by the cosine similarity

$$s(c_i, c_j) = \frac{\langle f_{\text{enc}}(c_i), f_{\text{enc}}(c_j) \rangle}{\|f_{\text{enc}}(c_i)\| \|f_{\text{enc}}(c_j)\|} \quad , \tag{1}$$

which in our setting is always non-negative. The resulting label dissimilarity matrix for the ImageNet labels is visualized in the supplementary material.

3.2 Label encoding

We adopt the formal definition of a curriculum from the seminal paper by Bengio et al. [1]. In [1], a weighting of the training data is adapted, so that in the

beginning a larger weight is put on easy examples. To distinguish this work from our approach, we refer to it as *example-weighting curriculum*.

Let $t \geq 0$ denote some notion of training time (e.g., a counter of training epochs). In [1], there is a sequence of weights associated with each example $i = 1, \ldots, \ell$, which we denote by $w_i^{(t)} \in [0, 1]$. These weights are normalized so that $\sum_{i=1}^{\ell} w_i^{(t)} = 1$ to describe a proper probability distribution over the training examples.

For the weight sequence to be a proper (example-weighting) curriculum, Bengio et al. [1] demand that the entropy of the weights

$$H(\boldsymbol{w}^{(t)}) = -\sum_{i=1}^{\ell} w_i^{(t)} \ln w_i^{(t)} \tag{2}$$

is monotonically increasing with t (the weights should converge to the uniform distribution).

We define our *label-weighting curriculum* in a similar axiomatic way. Instead of a sequence of weights for the training examples varying with t, we have a sequence of label vectors for each training example. Let $\boldsymbol{v}_i^{(t)}$ denote the C-dimensional label vector for training pattern i at time t. For the sequence to be a label-weighting curriculum, the entropy of the label vector components

$$\forall i = 1, \ldots, \ell : H(\boldsymbol{v}_i^{(t)}) = -\sum_{c=1}^{C} [\boldsymbol{v}_i]_c^{(t)} \ln [\boldsymbol{v}_i]_c^{(t)} \tag{3}$$

should be monotonically *decreasing* under the constraints that for each label vector $\boldsymbol{v}_i^{(t)}$ we have $[\boldsymbol{v}]_j \geq 0$ for all j, $\|\boldsymbol{v}_i^{(t)}\|_1 = 1$, and $\operatorname{argmax}_j [\boldsymbol{v}]_j^{(t)} = n(c_i)$ for all t. The conditions imply that $\boldsymbol{v}_i$ is always an element of the probability simplex, the class label given in the training set always gets the highest probability, and $\boldsymbol{v}_i^{(t)}$ converges to $\boldsymbol{y}_i$.

We now give an example of how to adapt the label vectors. Similar as in [1], we define for each training example i the simple update rule:

$$[\boldsymbol{v}_i]_j^{(t+1)} = \begin{cases} \frac{1}{1+\epsilon \sum_{k \neq n(c_i)} [\boldsymbol{v}_i]_k^{(t)}} & \text{if } j = n(c_i) \\ \frac{\epsilon [\boldsymbol{v}_i]_j^{(t)}}{1+\epsilon \sum_{k \neq n(c_i)} [\boldsymbol{v}_i]_k^{(t)}} & \text{otherwise} \end{cases} \tag{4}$$

The constant parameter $0 < \epsilon < 1$ controls how quickly the label vectors converge to the one-hot-encoded labels. This update rule leads to a proper label-weighting curriculum. During learning, the entries for all components except $n(c_i)$ drop with $\mathcal{O}(\epsilon^t)$. Note that $[\boldsymbol{v}_i]_{n(c_i)}^{(t+1)} \geq [\boldsymbol{v}_i]_{n(c_i)}^{(t)}$. The vectors are initialized using the label similarity defined in (1):

$$[\boldsymbol{v}_i]_j^{(0)} = \frac{s\big(c_i, n^{-1}(j)\big)}{\sum_{k=1}^{C} s\big(c_i, n^{-1}(k)\big)} \tag{5}$$

Recall that $n^{-1}(j)$ denotes the "j-th" natural language class label.

3.3 Loss function

Let $\mathcal{L}$ be a loss function between two probability distributions and $f_\theta(x)$ be the predicted distribution for example $\boldsymbol{x}$ for some model parameters θ. At time step t we optimize $J^{(t)}(\theta) = \sum_{i=1}^{n} \mathcal{L}(f_\theta(\boldsymbol{x}_i), \boldsymbol{v}_i^{(t)}) + \lambda r(\theta)$, where λ is a positive constant and $r(\theta)$ is a regularization function. In this paper, the networks are trained using the standard cross-entropy loss function with normalized targets $\boldsymbol{v}_i$ for the inputs $\boldsymbol{x}_i$, $i = 1, \ldots, \ell$. Hence, in the beginning, predicting the correct one-hot encoded label $\boldsymbol{y}_i$ causes an error signal. That is, initially it is less penalized if an object is not correctly classified with maximum confidence. Later in the training process, $\boldsymbol{v}_i$ converges to $\boldsymbol{y}_i$ and the classifier is then pushed to build up confidence.

4 Experiments

We evaluated our curriculum learning strategy by running extensive experiments on ImageNet [4], CIFAR100 [17], and AWA2 [33] plus additional experiments on CUB-200-2011 [30] and NABirds [29] (see supplementary material). On CUB-200-2011 and AwA2, we evaluated our approach using both the proposed semantic similarity of the labels as well as visual similarity. On NABirds, we evaluated our approach also using similarity based on the given (biological) hierarchy, where we used simrank [16] for calculating the similarity matrix.

For each dataset we considered at least two different models and two different baselines. Descriptive statistics of the datasets and a summary of the experimental setup are given in Table 1 and Table 2. We considered different training set sizes, where DR $\in \{5\%, 10\%, 20\%, 100\%\}$ refers to the fraction of training data used. The remaining training data was discarded (i.e., not used in the training process at all); the test data were always the same.

Table 1. ℓ_{train} denotes the number of training images, ℓ_{test} denotes the number of test images and C the number of classes in a given dataset; DR indicates the data set sizes (percentage of ℓ_{train}) and ϵ the cooling parameters. The column Sim. indicates which similarity measures were used, where l stands for the semantic similarity using the word embedding of the labels, v for a measure based on the similarity of the images, and h for similarity based on a given label hierarchy. For each experimental setup, #Rep= 4 repetitions with different initializations/seeds were conducted.

	ℓ_{train}	ℓ_{test}	C	**DR**	$\boldsymbol{\epsilon}$	**Sim.**
AWA2	29865	7457	50	5%, 10%, 20%, 100%	0.9, 0.99, 0.999	l, v
CIFAR100	50000	10000	100	5%, 10%, 20%, 100%	0.9, 0.99, 0.999	l
ImageNet	1281167	50000	1000	5%, 10%, 20%, 100%	0.9, 0.99, 0.999	l
NABirds	23912	24615	555	100%	0.9,0.99,.999	l, h
CUB-200-2011	5994	5794	201	100%	0.9,0.99,.999	l, v

We empirically compared the following algorithms:

1. Label-similarity curriculum learning (LCL): Proposed method with label update rule (4). The time step t is the epoch number.
2. Standard Learning (SL): This is a standard setup with fixed one-hot encoding.
3. Label Smoothing (LS): Label smoothing uses soft targets instead of one-hot encoding. It has been argued that LS prevents the network from becoming over-confident and improves the empirical performance of the algorithm [21]. For $0 \leq \alpha \leq 1$ label smoothing uses following label vector

$$[\boldsymbol{v}_i]_j^{(t)} = \begin{cases} (1-\alpha) + \frac{\alpha}{C} & \text{if } j = n(c_i) \\ \frac{\alpha}{C} & \text{otherwise} \end{cases} \quad \text{for all } t. \tag{6}$$

   We set$\alpha = 0.1$ for the evaluations in this study.
4. Deep Mutual Learning (DML): In DML, two models, referred to as $\text{DML}_1$and DML_2, solve the same classification problem collaboratively and are optimised jointly [38]. It uses one hot-encoding along with cross-entropy loss as in SL but adds additional terms $\text{KL}(\hat{\boldsymbol{v}}^{(t)}_{\text{DML}_1} \,\|\, \hat{\boldsymbol{v}}^{(t)}_{\text{DML}_2}) + \text{KL}(\hat{\boldsymbol{v}}^{(t)}_{\text{DML}_2} \,\|\, \hat{\boldsymbol{v}}^{(t)}_{\text{DML}_1})$, where KL denotes the Kullback–Leibler divergence and $\hat{\boldsymbol{v}}^{(t)}_{\text{DML}_1}$ and $\hat{\boldsymbol{v}}^{(t)}_{\text{DML}_2}$ are the predicted label probability vectors for both models. We report the classification performance of both DML_1 and DML_2.
5. Knowledge Distillation (KD): In KD, one model is trained first using one-hot encoded targets, and then the class probabilities produced by the first model are used as "soft targets" for training the second model [13].
6. Curriculum Net (CN): In CN, example-weighted curriculum is built by sorting examples from easy to hard [11].

Table 2. ResNeXt-101 denotes ResNeXt-101 (32×8d) , WRN denotes WRN (28-10-dropout), and DenseNet-BC denotes DenseNet-BC (k = 40, depth=190) .

Model	**Dataset**	**Baselines**
ResNet-18 [34]	CUB-200-2011	LS, DML, SL, KD, CN
ResNet-34 [34]	CUB-200-2011, NABirds	LS, DML, SL, KD, CN
ResNet-50 [12]	ImageNet, NABirds	SL, LS, KD, CN
ResNeXt-101 [34]	ImageNet	SL, LS, KD, CN
SENet-154 [14]	ImageNet	SL, LS, KD, CN
ResNet-101 [12]	AWA2	LS, DML, SL, KD, CN
InceptionResNetV2 [28]	AWA2	LS, DML, SL, KD, CN
WRN [37]	CIFAR100	LS, DML, SL, KD, CN
DenseNet-BC [15]	CIFAR100	LS, DML, SL, KD, CN

For all architectures, we have followed the experimental protocols described in the original publications [12,37,34,14,28,15]. All experiments were conducted using the PyTorch deep learning library [24].[3] For all experiments, except ImageNet DR = 100%, we used stochastic gradient descent (SGD) for optimization.

[3] The code to reproduce our results is available in the supplementary material.

For ImageNet DR = 100% we used the distributed SGD algorithm [9] with Horovod[4] [26] support because of the computational demands. The distributed SGD algorithm [9] is one of the state-of-the-art methods for large scale training. It is expected to lead to a slight loss in performance when a large batch size is used (see [9] for details).

Our approach introduces the hyperparameter ϵ, see (4). In order to assess the stability of the proposed method, we present results for $\epsilon \in \{0.9, 0.99, 0.999\}$.[5] We repeated all experiments four times. We report the top-1 and top-5 classification accuracy on the test datasets (standard deviations are reported in the supplementary material).

For estimating the label similarity matrix, we used *pretrained* `doc2vec` emebeddings with dimensions $d \in \{100, 300, 500\}$ with ResNet-50 and ResNet-101 . We did not observe any significant differences in the classification accuracies. The maximum difference between compatible settings were less than 0.06 %. Hence, we only report results for the $d = 100$ dimensional `doc2vec` embeddings.

For each experiment, we used workstations having 4 Tesla P100 GPUs (with 16GB GPU RAM) each. For network communication we used InfiniBand, which is a computer-networking communications standard designed for high throughput and low-latency scenarios.

We tuned the hyperparameters for the baseline method (SL) only. For 100% data, we took the hyperparameters from the original publications. For all other settings, we optimized learning rate, batch size and weight-decay for the standard baseline (SL). Then we use the very same parameters for our approach (we just varied the new parameter epsilon). Thus, hyperparameter tuning would rather increase the gain from using our method. Thus, one might argue that the new algorithm is using sub-optimal hyperparameters compared to the baselines. However, our goal was to show that the proposed algorithm can improve any model on different datasets without tuning hyperparameters.

5 Results and Discussion

We will focus on the results for ImageNet, CIFAR100, and AWA2 and the similarity measure introduced in Section 3, result tables for the other data sets and other similarity measures can be found in the supplementary material. Before we present the learning results, we will discuss the structure of the label similarity matrices for the data sets ImageNet, CIFAR100, and AWA2.

Label Similarities. For a better understanding of the label similarity matrices, we visualized their eigenspectra in Figure 2. Consider two extreme scenarios: If

[4] Horovod is a method which uses large batches over multiple GPU nodes and some accuracy loss is expected for the baseline method and this is well established. For more details please see Table 1 and Table 2.c in [24].

[5] We have tried $\epsilon \in \{0.8, 0.9, 0.91, \ldots, 0.98, 0.99, 0.992, \ldots, 0.998, 0.999\}$ for ResNet-50 and ResNet-101 . The results showed that the search space for ϵ can be less granular and we have limited the search space accordingly.

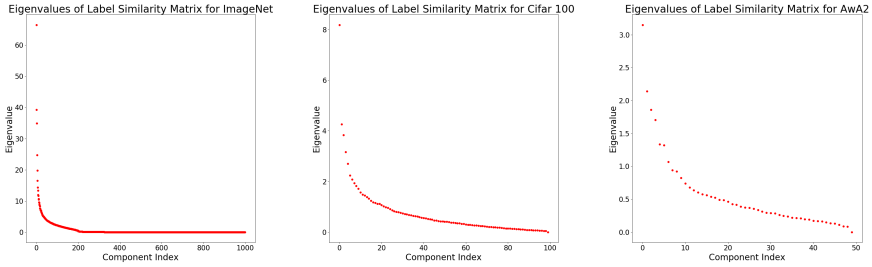

Fig. 2. Eigenvalue distributions of the class similarity matrices for ImageNet, CIFAR100, and AwA2.

a label similarity matrix has rank 1, all classes are exactly the same and there cannot be any discriminatory learning. In contrast, the full rank case with equal eigenvalues is the standard learning case where all classes are orthogonal to each other (one-hot-encoding). Figure 2 shows exponential eigenvalues decays, which means there are clusters of similar classes. Distinguishing between these clusters of classes is an easier task than distinguishing between classes within one cluster.

Table 3. ImageNet. Top-1 results, averaged over four trials.

					ϵ		
	SL	**LS**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
	DR = 5%						
ResNet-50	38.21	38.43	39.81	36.12	41.24	41.6	**42.21**
ResNeXt-101	45.46	45.71	46.21	44.14	46.1	46.92	**47.12**
SENet-154	48.29	48.57	48.44	46.21	49.8	50.04	**50.19**
	DR = 10%						
ResNet-50	51.95	52.25	53.64	52.17	55.39	55.62	**55.64**
ResNeXt-101	58.63	58.92	58.94	57.64	59.78	**60.07**	59.92
SENet-154	60.61	60.74	60.82	60.14	60.99	61.18	**62.28**
	DR = 20%						
ResNet-50	61.87	62.11	63.17	62.41	64.41	64.42	**64.44**
ResNeXt-101	67.96	68.13	68.29	68.14	68.48	68.47	**68.57**
SENet-154	67.77	67.71	67.64	67.43	68.14	**68.4**	68.33
	DR = 100%						
ResNet-50	76.25	76.4	76.38	76.1	76.71	76.75	**76.89**
ResNeXt-101	78.05	78.17	78.21	77.94	78.31	78.5	**78.64**
SENet-154	79.33	79.65	79.44	79.44	80.11	80.03	**80.21**

Table 4. ImageNet. Top-5 results, averaged over four trials.

					ϵ		
	SL	**LS**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
	DR = 5%						
ResNet-50	64.04	64.35	67.22	65.12	64.41	67.59	**67.94**
ResNeXt-101	70.52	70.76	71.84	70.92	70.67	72.05	**72.18**
SENet-154	73.35	73.52	74.54	73.92	73.56	74.65	**74.92**
	DR = 10%						
ResNet-50	76.86	77.04	79.73	78.14	77.1	**79.8**	79.69
ResNeXt-101	81.52	81.87	82.56	81.92	81.67	82.66	**82.74**
SENet-154	82.53	**83.71**	83.38	82.76	82.94	83.6	83.5
	DR = 20%						
ResNet-50	84.41	84.57	86.1	85.36	84.91	**86.15**	86.14
ResNeXt-101	87.96	88.11	88.2	88.04	87.84	**88.36**	**88.36**
SENet-154	88.16	88.23	88.17	88.17	88.11	**88.37**	88.31
	DR = 100%						
ResNet-50	92.87	92.91	92.94	92.41	92.84	**92.95**	92.93
ResNeXt-101	93.95	93.92	93.96	93.85	93.87	94.07	**96.15**
SENet-154	94.33	94.44	94.84	94.02	94.35	**94.93**	94.79

Classification Performance. We measured the top-1 and top-5 classification accuracy after the last epoch. The results are summarized in Table 3 and Table 4 for ImageNet, in Table 5 and Table 6 for CIFAR100, and in Table 7 and Table 8 for AWA2 (for standard deviations see the supplementary material). All results are averaged over four trials. It is important to note that we compare against baseline results achieved with architectures and hyperparameters tuned for excellent performance. Furthermore, we compare to baseline results from our own experiments, not to results taken from the literature. We ran each experiment 4 times with same seeds for all algorithms. This allows for a fair comparison. Our averaged results also provide a more reliable estimate of the performance of the systems compared to single trials reported in the original works.

The results show that for all datasets and in all experimental cases using LCL outperformed all baselines, with SeNet with DR = 10% and top-5 metric being the only exception. The improvement was more pronounced when DR < 100%. It is quite intuitive that a curriculum is much more important when the training data is limited (i.e., the learning problem is more difficult). Loosely speaking, the importance of a teacher decreases when a student has access to unlimited information without any computational and/or time budget. For example, for ResNet-50 on ImageNet LCL improved the top-1 accuracy on average by 4 percentage points (p.p.) over the baseline when DR = 5%, and 2 p.p. in top-5

Table 5. CIFAR100. Top-1 results, averaged over four trials.

							ϵ		
	SL	**LS**	**DML$_1$**	**DML$_2$**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
					DR = 5%				
WRN	40.2	40.31	40.16	40.47	40.94	38.14	41.36	41.86	**41.92**
DenseNet-BC	43.34	43.5	43.89	44.14	43.76	43.42	**44.66**	45.37	44.53
					DR = 10%				
WRN	60.2	60.1	60.38	60.34	60.45	60.14	60.49	60.86	**61.19**
DenseNet-BC	60.85	61.1	61.22	61.34	60.81	59.83	61.5	61.4	**61.65**
					DR = 20%				
WRN	71.05	71.25	71.61	71.65	71.53	71.37	71.64	71.67	**71.83**
DenseNet-BC	72.38	72.39	71.5	71.34	72.24	71.54	72.71	72.65	**72.87**
					DR = 100%				
WRN	79.52	79.84	80.32	80.20	80.14	78.64	81.17	81.15	**81.25**
DenseNet-BC	82.85	83.01	82.91	82.57	82.67	81.14	82.96	83.11	**83.2**

Table 6. CIFAR100. Top-5 results, averaged over four trials.

							ϵ		
	SL	**LS**	**DML$_1$**	**DML$_2$**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
					DR = 5%				
WRN	68.82	68.95	68.74	68.89	68.94	69.12	69.47	69.39	**69.63**
DenseNet-BC	70.61	70.85	70.7	70.92	71.14	71.27	72.1	71.13	**72.52**
					DR = 10%				
WRN	83.64	83.82	84.05	84.11	83.94	83.77	83.99	84.15	**84.3**
DenseNet-BC	84.07	84.21	84.27	84.45	84.07	84.31	84.34	**84.63**	84.52
					DR = 20%				
WRN	90.52	90.38	90.3	90.27	90.71	90.45	91.02	**91.19**	90.95
DenseNet-BC	91.24	91.37	91.33	91.30	91.39	91.38	91.4	91.31	**91.47**
					DR = 100%				
WRN	94.04	94.23	94.44	94.42	94.52	94.52	95.29	95.17	**95.49**
DenseNet-BC	95.22	95.28	95.34	95.27	95.37	95.63	95.72	95.74	**95.88**

accuracy were gained with DR = 100% on ImageNet for the ResNeXt architecture. The biggest improvements were achieved on the AWA2 dataset. For ResNet-101 and DR = 5%, average improvements of more than 22 p.p. and 23 p.p. could be achieved in the top-1 and top-5 accuracy, respectively. As could be expected, the

Table 7. AWA2. Top-1 results, averaged over four trials.

							ϵ		
	SL	**LS**	**DML$_1$**	**DML$_2$**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
	DR = 5%								
ResNet-101	23.09	27.82	39.22	37.19	41.58	24.1	45.51	45.55	**45.78**
InceptionResNetV2	57.42	57.95	58.69	58.14	59.3	56.9	60.85	**61.07**	60.71
	DR = 10%								
ResNet-101	41.86	44.98	48.92	50.5	44.02	43.12	47.21	51.67	**53.39**
InceptionResNetV2	71.47	71.86	71.82	72.37	71.49	72.01	72.61	72.97	**73.01**
	DR = 20%								
ResNet-101	77.11	78.23	78.34	78.32	78.28	77.64	80.03	**80.07**	79.86
InceptionResNetV2	83.64	83.92	83.87	83.76	83.83	84.12	**84.27**	84.05	**84.27**
	DR = 100%								
ResNet-101	88.73	89.25	89.01	89.11	89.17	88.92	89.44	**89.64**	89.63
InceptionResNetV2	89.69	89.94	90.05	90.22	89.94	89.29	**90.49**	90.34	90.47

Table 8. AWA2. Top-5 results, averaged over four trials.

							ϵ		
	SL	**LS**	**DML$_1$**	**DML$_2$**	**KD**	**CN**	**0.9**	**0.99**	**0.999**
	DR = 5%								
ResNet-101	53.31	54.19	65.14	63.12	56.19	54.17	76.02	76.14	**76.47**
InceptionResNetV2	83.06	83.14	84.07	84.18	84.12	83.77	84.94	**85.24**	84.84
	DR = 10%								
ResNet-101	72.59	72.43	75.07	76.14	76.61	76.34	77.04	**80.46**	80.11
InceptionResNetV2	91.37	91.42	91.35	91.43	91.48	91.35	**91.9**	91.89	91.71
	DR = 20%								
ResNet-101	94.21	94.56	94.79	95.01	94.61	94.45	**95.2**	95.07	95.12
InceptionResNetV2	96.03	96.23	96.28	96.13	96.21	95.19	96.18	96.49	**96.57**
	DR = 100%								
ResNet-101	97.85	97.92	98.1	97.95	97.43	97.32	98.11	**98.14**	98.1
InceptionResNetV2	98.01	98.07	98.25	98.17	97.67	97.56	98.25	**98.41**	98.2

performance gains in the top-5 setting were typically smaller than for top-1. Still, nothing changed with respect to the ranking of the network architectures.

Larger values of ϵ mean slower convergence to the one-hot-encoding and therefore more emphasis on the curriculum learning. In most experiments, $\epsilon =$

0.999 performed best. The observation that larger ϵ values gave better results than small ones provides additional evidence that the curriculum really supports the learning process (and that we are not discussing random artifacts).

Under the assumption that the experimental scenarios are statistically independent, we performed a statistical comparisons of the classifiers over multiple data sets for all pairwise comparisons following [3,8]. Using the Iman and Daveport test, all but one result were statistically significant. If we consider all ImageNet top-1 accuracy results, our method with $\epsilon = 0.999$ ranked best, followed by $\epsilon = 0.99$, $\epsilon = 0.9$, LS and then SL. This ranking was highly significant (Iman and Daveport test, $p < 0.001$). Similarly, our method with $\epsilon = 0.999$ was best for both CIFAR-100 and AWA2 ($p < 0.001$).

6 Conclusions

We proposed a novel curriculum learning approach referred to as label-similarity curriculum learning. In contrast to previous methods, which change the weighting of training examples, it is based on adapting the label representation during training. This adaptation considers the semantic similarity of labels. It implements the basic idea that at an early stage of learning it is less important to distinguish between similar classes compared to separating very different classes. The class similarity can be based on arbitrary *a priori* knowledge, in particular on additional information not directly encoded in the training data. For the case where the class labels are natural language words, we proposed a way to automatically define class similarity via a word embedding. We also considered other similarity measures for datasets where these similarity measures were available.

We extensively evaluated the approach on five datasets. For each dataset, two to three deep learning architectures proposed in the literature were considered. We looked at simple label smoothing and, for the two smaller datasets, also at deep mutual learning (DML) as additional baselines. In each case, we considered four different training data set sizes. Each experiment was repeated four times. The empirical results strongly support our approach. *Label-similarity curriculum learning was able to improve the average classification accuracy on the test data compared to standard training in all scenarios.* The improvements achieved by our method were more pronounced for smaller training data sets. When considering only 10% of the AWA2 training data, label-similarity curriculum learning increased the Resnet101 top-1 test accuracy by more than 22 percentage points on average compared to the standard baseline. Our curriculum learning also outperformed simple label smoothing and DML in all but a single case. Our method turned out to be robust with respect to the choice of the single hyperparameter controlling how quickly the learning process converges to minimizing the standard cross-entropy loss. In contrast to related approaches such as knowledge distillation and DML, the additional computational and memory requirements can be neglected.

The proposed label-similarity curriculum learning is a general approach, which also works for settings where the class similarity is not based on the semantic similarity of natural language words (see supplementary material).

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: International Conference on Machine Learning (ICML). pp. 41–48. ACM (2009) 1, 2, 3, 5, 6
2. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). pp. 535–541. ACM (2006) 3
3. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006) 14
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255. IEEE (2009) 3, 7
5. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., Abbeel, P.: Reverse curriculum generation for reinforcement learning. In: Conference on Robot Learning (CoRL). pp. 482–495 (2017) 3
6. Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M., Mikolov, T.: Devise: A deep visual-semantic embedding model. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2121–2129 (2013) 4
7. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born again neural networks (2018), arXiv:1805.04770 [stat.ML] 3
8. García, S., Herrera, F.: An extension on statistical "comparisons of classifiers over multiple data sets" for all pairwise comparisons. Journal of Machine Learning Research **9**, 2677–2694 (2008) 14
9. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: Training ImageNet in 1 hour (2017), arXiv:1706.02677v2 [cs.CV] 9
10. Graves, A., Bellemare, M.G., Menick, J., Munos, R., Kavukcuoglu, K.: Automated curriculum learning for neural networks. In: International Conference on Machine Learning (ICML). pp. 1311–1320. PMLR (2017) 3
11. Guo, S., Huang, W., Zhang, H., Zhuang, C., Dong, D., Scott, M.R., Huang, D.: CurriculumNet: Weakly supervised learning from large-scale web images. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 135–150 (2018) 4, 8
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. IEEE (2016) 8
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Deep Learning and Representation Learning Workshop: NIPS 2014 (2014) 3, 8
14. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7132–7141. IEEE (2018) 8
15. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708. IEEE (2017) 8
16. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 538–543 (2002) 7
17. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009) 3, 7

18. Kumar, M.P., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 1189–1197 (2010) 3
19. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning (ICML). pp. 1188–1196 (2014) 5
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013), arXiv:1301.3781v3 [cs.CL] 5
21. Müller, R., Kornblith, S., Hinton, G.: When does label smoothing help? (2019), arXiv:1906.02629 [cs.LG] 4, 8
22. Orbes-Artega, M., Cardoso, J., Sørensen, L., Igel, C., Ourselin, S., Modat, M., Nielsen, M., Pai, A.: Knowledge distillation for semi-supervised domain adaptation. In: Machine Learnig in Clinical Neuroimaging (MLCN 2019). LNCS, vol. 11796, pp. 68–76. Springer (2019) 3
23. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 582–597. IEEE (2016) 3
24. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NeurIPS 2017 Workshop Autodiff (2017) 8, 9
25. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: FitNets: Hints for thin deep nets (2014), arXiv:1412.6550 [cs.LG] 3
26. Sergeev, A., Del Balso, M.: Horovod: fast and easy distributed deep learning in TensorFlow (2018), arXiv:1802.05799v3 [cs.LG] 9
27. Silla, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1), 31–72 (2011) 2
28. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence (AAAI). pp. 4278–4284 (2017) 8
29. Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., Belongie, S.: Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In: roceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 7
30. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 7
31. Weinshall, D., Cohen, G., Amir, D.: Curriculum learning by transfer learning: Theory and experiments with deep networks. In: International Conference on Machine Learning (ICML). pp. 5235–5243. PMLR (2018) 3
32. Wu, C., Tygert, M., LeCun, Y.: Hierarchical loss for classification (2017), arXiv:1709.01062v1 [cs.LG] 2
33. Xian, Y., Lampert, C.H., Schiele, B., Akata, Z.: Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018) 3, 7
34. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1492–1500. IEEE (2017) 8
35. Yim, J., Joo, D., Bae, J., Kim, J.: A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4133–4141 (2017) 3

36. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer (2016), arXiv:1612.03928 [cs.CV] 3
37. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: British Machine Vision Conference (BMVC). BMVA Press (2016) 8
38. Zhang, Y., Xiang, T., Hospedales, T.M., Lu, H.: Deep mutual learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4320–4328 (2018) 3, 4, 8