Label-similarity Curriculum Learning

Urun Dogan
Bing Ads
Microsoft AI & Research
Sunnyvale, CA, USA

Aniket Anand Deshmukh
Bing Ads
Microsoft AI & Research
Sunnyvale, CA, USA

Marcin Machura
Bing Ads
Microsoft AI & Research
Sunnyvale, CA, USA

Christian Igel
Department of Computer Science
University of Copenhagen
Copenhagen, Denmark

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 on several popular deep learning architectures for image classification task applied to three datasets, ImageNet, CIFAR100, and AWA2. In all scenarios, LCL was able to improve the classification accuracy on the test data compared to standard training.

Keywords: Curriculum Learning, Image 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 [2, 3] (“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.” [3]). However, there is are two decisive differences. First, we propose to gradually shift from a “hierarchical loss” to a “flat loss”. Second, unlike in [3], our approach does not 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 a dataset 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 [5], and Animals with Attributes (AwA) [6]. 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 gives the related literature and the 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. [7] introduces latent variables for modelling “easiness” of an examples. Graves et al. [8] consider example-based improvement measures as reward signals for multi-armed bandits, which then build stochastic syllabi for neural networks. Florensa et al. [9] 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. [10] 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.

Buciluă et al. [11] 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 [12] under the name knowledge distillation (KD) and gathered significant amount of attention [13, 14, 15, 16, 17]. 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 [12]. A variant of KD, *born again neural network*, trains a series of models, not only one [18]. KD methods typically require a pretrained model to start with or train a series of models on the same training data.

*Deep mutual learning* (DML) is also loosely related to our proposed approach [12, 19]. In DML, two models solve the same classification problem collaboratively and are optimised jointly [19]. Each model acts as a teacher for the other model. Each network is trained with two losses. The first loss is a 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* [20], 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

---

1. A complete list of all references is beyond scope of this paper. Hence, we have only mentioned a set of representative papers.
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.
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. [21]. DeVise requires a
pretrained deep neural network. Furthermore, as stated in [21], 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 to label smoothing (LS) [22], which we will discuss in Section 4.

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

3. Method

We assume a discrete set of training examples \((x_1, c_1), \ldots, (x_\ell, c_\ell) \in \mathcal{X} \times \mathcal{C}\), with input space \(\mathcal{X}\) and
finite label space \(\mathcal{C}\) with cardinality \(|\mathcal{C}| = C\). The labels are assumed to be natural language words, for
example \(\mathcal{C} = \{\ldots, \text{flute}, \ldots, \text{strawberry}, \ldots, \text{backpack}, \ldots\}\). Let \(n : \mathcal{C} \rightarrow \{1, \ldots, C\}\) be
a bijective mapping assigning each label to a unique integer. This allows a straightforward definition
of the one-hot encoding \(y_i \in \mathbb{R}^C\) for each training example \((x_i, c_i)\). The \(j\)-th component of \(y_i\),
which is denoted by \([y_i]_j\), 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 \(C\) different classes, which
are defined based on the semantic similarity of the class labels. To this end, we embed the natural
language labels into Euclidean space using a word embedding [23], 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 synset 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 [24] for mapping the article into Euclidean space.
We used a generic doc2vec embedding trained on the English Wikipedia corpus. This gives us
the encoding \(f_{\text{enc}} : \mathcal{C} \rightarrow \mathbb{R}^d\), mapping each class label to the corresponding Wikipedia article and
then computing the corresponding vector representation using \(\text{doc2vec}\) (with \(d = 100\), see below). Now we can compute the similarity between two classes \(c_i\) and \(c_j\) by the cosine similarity

\[
s(c_i, c_j) = \frac{\langle f_{\text{enc}}(c_i), f_{\text{enc}}(c_j) \rangle}{\|f_{\text{enc}}(c_i)\| \|f_{\text{enc}}(c_j)\|},
\]

which in our setting is always non-negative. The resulting label dissimilarity matrix for the ImageNet
labels is visualized in Figure 2.
Figure 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.

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

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

(2)

is monotonically increasing with $t$ (the weights should converge to the uniform distribution).
Figure 2: Label dissimilarity matrix visualizing $1 - s(c_i, c_j)$, $1 \leq i, j \leq 1000$, for the ImageNet labels.

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 $v_i^{(t)}$ 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

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

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

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:

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

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 $O(\epsilon^t)$. Note that $[v_i]_{n(c_i)}^{(t+1)} \geq [v_i]_{n(c_i)}^{(t)}$. 


The vectors are initialized using the label similarity defined in (1):

\[
[v_i]_j^{(0)} = \frac{s(c_i, n^{-1}(j))}{\sum_{k=1}^{C} s(c_i, n^{-1}(k))}
\]  

Recall that \(n^{-1}(j)\) denotes the “j-th” natural language class label.

3.3 Loss function

Let \(L\) be a loss function between two probability distributions and \(f_\theta(x)\) be the predicted distribution for example \(x\) for some model parameters \(\theta\). At time step \(t\) we optimize

\[
J^t(\theta) = \sum_{i=1}^{n} L(f_\theta(x_i), v_i^{(t)}) + \lambda r(\theta),
\]

where \(\lambda\) 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 \(v_i\) for the inputs \(x_i\), \(i = 1, \ldots, \ell\). Hence, in the beginning, predicting the correct one-hot encoded label \(y_i\) causes an error signal. That is, in the beginning it is less penalized if an object is not correctly classified with maximum confidence. Later in the training process, \(v_i\) converges to \(y_i\) 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 [5], and AWA2 [6]. Descriptive statistics of these three datasets are given in Table 4. For each dataset we considered at least two different models and two different baselines. A summary of the experimental setup is given Table 2. We considered different training set sizes, where \(DR \in \{10\%, 20\%, 40\%, 100\%\}\) refers to the fraction of training data used (the test data is always the same).

| Dataset       | \(\ell_{\text{train}}\) | \(\ell_{\text{test}}\) | \(C\) |
|---------------|-------------------------|-------------------------|-------|
| AWA2          | 29865                   | 7457                    | 50    |
| CIFAR100      | 50000                   | 10000                   | 100   |
| ImageNet      | 1281167                 | 50000                   | 1000  |

Table 1: \(\ell_{\text{train}}\) denotes the number of training images, \(\ell_{\text{test}}\) denotes the number of test images and \(C\) the number of classes in a given dataset.

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 one hot-encoding,

\[
[v_i]_j^{(t)} = \begin{cases} 
1 & \text{if } j = n(c_i) \\
0 & \text{otherwise}
\end{cases}
\text{ for all } t. 
\]  

(7)
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 [22]. For $0 \leq \alpha \leq 1$ Label smoothing uses following label vector

$$[v_i]_j^{(t)} = \begin{cases} (1 - \alpha) + \frac{\alpha}{C} & \text{if } j = n(c_i) \\ \frac{\alpha}{C} & \text{otherwise} \end{cases} \text{ for all } t. \quad (8)$$

We use $\alpha = 0.1$ for the evaluations in this study.

4. Deep Mutual Learning (DML): In DML, two models, referred to as DML$_1$ and DML$_2$, solve the same classification problem collaboratively and are optimised jointly [19]. It uses one-hot-encoding along with cross-entropy loss as in SL but adds additional terms

$$\text{KL}(\hat{v}_\text{DML}_1^{(t)} \parallel \hat{v}_\text{DML}_2^{(t)}) + \text{KL}(\hat{v}_\text{DML}_2^{(t)} \parallel \hat{v}_\text{DML}_1^{(t)}), \quad (9)$$

where KL denotes the Kullback–Leibler divergence and $\hat{v}_\text{DML}_1^{(t)}$ and $\hat{v}_\text{DML}_2^{(t)}$ are the predicted label probability vectors for both models. We report the classification performance of both DML$_1$ and DML$_2$.

| Model                | Dataset       | Baselines | DR    | #Rep | $\epsilon$ |
|----------------------|---------------|-----------|-------|------|-------------|
| ResNet-50 [25]       | ImageNet      | SL, LS    | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| ResNet-101-32×8d [26] | ImageNet    | SL, LS    | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| SENet-154 [27]       | ImageNet      | SL, LS    | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| ResNet-101 [25]      | AWA2          | LS, DML, SL | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| InceptionResNetV2 [28] | AWA2       | LS, DML, SL | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| WRN-28-10-dropout [29] | CIFAR100     | LS, DML, SL | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |
| DenseNet-BC ($k = 40$, depth=190) [30] | CIFAR100 | LS, DML, SL | 10%, 20%, 40%, 100% | 4    | 0.9, 0.99, 0.999 |

Table 2: LS denotes label smoothing, DML denotes mutual learning, SL denotes training without curriculum, DR refers to the fraction of training data used, $\epsilon$ is the cooling parameters, and #Rep denotes the number of repetitions with different initializations/seeds.

For all architectures, we have followed the experimental protocols described in the original publications [25, 29, 26, 27, 28, 30]. All experiments were conducted using the PyTorch deep learning library [31]. For all experiments, except ImageNet DR = 100%, we have used stochastic gradient descent (SGD) for optimization. However, for ImageNet DR = 100% because of the computational demands, we used the distributed SGD algorithm [32] with Horovod[3] support in order to scale the experimental design. The distributed SGD algorithm [32] is one of the state-of-the-art methods for large scale training. It is expected to lead to a slight loss in performance whenever a large batch size is used (see [32] for details).

Our approach introduces the hyperparameter $\epsilon$, see (4). In order to assess the stability of the proposed method, we present results for $\epsilon \in \{0.9, 0.99, 0.999\}$[4]. We repeated all experiments four

2. The code to reproduce our results is available in the supplementary material.
3. 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 of [31].
4. 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 $\epsilon$ can be less granular and we have limited the search space accordingly.
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 have used pretrained doc2vec embeddings with dimensions $d \in \{100, 300, 500\}$ with ResNet-50 and ResNet-101. We did not observe any significant differences in the classification accuracies. 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 did not tune any of the original hyperparameters (e.g., learning rates) of the models for our method. But note that these hyperparameters were tuned for the baseline. 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

Before we present the learning results, we will discuss the structure of the label similarity matrices for the three different data sets.

**Label Similarities** For a better understanding of the label similarity matrices, we visualized their eigenspectra in Figure 3. Consider two extreme scenarios: If 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 3 shows an 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.

**Classification Performance** We measured the top-1 classification accuracy and top-5 classification accuracy after the last epoch. The results are summarized in Table 3 and Table 4 for ImageNet, Table 5 and Table 6 for CIFAR100, 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 keep in mind 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 improved over all baselines, with SeNet with DR = 20% and top-5 metric being the only exception. The improvement was more significant 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 ResNet50 on ImageNet LCL improved the top-1 accuracy on average by 4 percentage points (p.p.) over the baseline when DR = 10%, and 2 p.p. in top-5 accuracy were gained with DR = 100% on ImageNet for the ResNeXt architecture. The biggest improvements were achieved on the AWA2 dataset. For ResNet101 and DR = 10%, 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 performance gains in the top-5 setting were typically smaller than for top-1. That is, nothing changed with respect to the ranking of the network architectures.

| DR   | SL  | LS  | 0.9 | 0.99 | 0.999 |
|------|-----|-----|-----|------|-------|
|      | ResNet50 | 38.21 | 38.43 | 41.24 | 41.6 | **42.21** |
| 10%  | ResNet50 | 38.21 | 38.43 | 41.24 | 41.6 | **42.21** |
|      | ResNeXt | 45.46 | 45.71 | 46.1 | 46.92 | **47.12** |
|      | SeNet   | 48.29 | 48.57 | 49.8 | 50.04 | **50.19** |
| 20%  | ResNet50 | 51.95 | 52.25 | 55.39 | 55.62 | **55.64** |
|      | ResNeXt | 58.63 | 58.92 | 59.78 | **60.07** | 59.92 |
|      | SeNet   | 60.61 | 60.74 | 60.99 | 61.18 | **62.28** |
| 40%  | ResNet50 | 61.87 | 62.11 | 64.41 | 64.42 | **64.44** |
|      | ResNeXt | 67.96 | 68.13 | 68.48 | 68.47 | **68.57** |
|      | SeNet   | 67.77 | 67.71 | 68.14 | **68.4** | 68.33 |
| 100% | ResNet50 | 76.25 | 76.4 | 76.71 | 76.75 | **76.89** |
|      | ResNeXt | 78.05 | 78.17 | 78.31 | 78.5 | **78.64** |
|      | SeNet   | 79.33 | 79.65 | 80.11 | 80.03 | **80.21** |

Table 3: ImageNet. Top-1 results, averaged over four trials.
Larger values of $\epsilon$ 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 $\epsilon$ values perform better than small ones provides additional evidence that the curriculum really supports the learning process (and that we are not discussing random artifacts).

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. We considered the case where the class labels are natural language words and proposed a way to automatically define class similarity by a metric induced by a word embedding.

We extensively evaluated the approach on three 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
Table 5: CIFAR100. Top-1 results, averaged over four trials.

|         | SL | LS | DML₁ | DML₂ | 0.9  | 0.99 | 0.999 |
|---------|----|----|------|------|------|------|-------|
| **DR = 10%** |    |    |      |      |      |      |       |
| WRN     | 40.2 | 40.31 | 40.16 | 40.47 | 41.36 | 41.86 | 41.92 |
| DenseNet | 43.34 | 43.5 | 43.89 | 44.14 | **44.66** | 45.37 | 44.53 |
| **DR = 20%** |    |    |      |      |      |      |       |
| WRN     | 60.2 | 60.1 | 60.38 | 60.34 | 60.49 | 60.86 | **61.19** |
| DenseNet | 60.85 | 61.1 | 61.22 | 61.34 | 61.5 | 61.4 | **61.65** |
| **DR = 40%** |    |    |      |      |      |      |       |
| WRN     | 71.05 | 71.25 | 71.61 | 71.65 | 71.64 | 71.67 | **71.83** |
| DenseNet | 72.38 | 72.39 | 71.5 | 71.34 | 72.71 | 72.65 | **72.87** |
| **DR = 100%** |    |    |      |      |      |      |       |
| WRN     | 79.52 | 79.84 | 80.32 | 80.20 | 81.17 | 81.15 | **81.25** |
| DenseNet | 82.85 | 83.01 | 82.91 | 82.57 | 82.96 | 83.11 | **83.2** |

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

|         | SL | LS | DML₁ | DML₂ | 0.9  | 0.99 | 0.999 |
|---------|----|----|------|------|------|------|-------|
| **DR = 10%** |    |    |      |      |      |      |       |
| WRN     | 68.82 | 68.95 | 68.74 | 68.89 | 69.47 | 69.39 | **69.63** |
| DenseNet | 70.61 | 70.85 | 70.7 | 70.92 | 72.1 | 71.13 | **72.52** |
| **DR = 20%** |    |    |      |      |      |      |       |
| WRN     | 83.64 | 83.82 | 84.05 | 84.11 | 83.99 | 84.15 | **84.3** |
| DenseNet | 84.07 | 84.21 | 84.27 | 84.45 | 84.34 | **84.63** | 84.52 |
| **DR = 40%** |    |    |      |      |      |      |       |
| WRN     | 90.52 | 90.38 | 90.3 | 90.27 | 91.02 | **91.19** | 90.95 |
| DenseNet | 91.24 | 91.37 | 91.33 | 91.30 | 91.4 | 91.31 | **91.47** |
| **DR = 100%** |    |    |      |      |      |      |       |
| WRN     | 94.04 | 94.23 | 94.44 | 94.42 | 95.29 | 95.17 | **95.49** |
| DenseNet | 95.22 | 95.28 | 95.34 | 95.27 | 95.72 | 95.74 | **95.88** |

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

|          | SL   | LS   | DML\(_1\) | DML\(_2\) | \(\epsilon\) 0.9 | \(\epsilon\) 0.99 | \(\epsilon\) 0.999 |
|----------|------|------|-----------|-----------|------------------|------------------|-----------------|
| **DR = 10\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 23.09| 27.82| 39.22     | 37.19     | 45.51            | 45.55            | **45.78**       |
| InceptionResNetv2 | 57.42| 57.95| 58.69     | 58.14     | 60.85            | **61.07**        | 60.71           |
| **DR = 20\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 41.86| 44.98| 48.92     | 50.5      | 47.21            | 51.67            | **53.39**       |
| InceptionResNetv2 | 71.47| 71.86| 71.82     | 72.37     | 72.61            | 72.97            | **73.01**       |
| **DR = 40\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 77.11| 78.23| 78.34     | 78.32     | 80.03            | **80.07**        | 79.86           |
| InceptionResNetv2 | 83.64| 83.92| 83.87     | 83.76     | **84.27**        | 84.05            | **84.27**       |
| **DR = 100\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 88.73| 89.25| 89.01     | 89.11     | 89.44            | **89.64**        | 89.63           |
| InceptionResNetv2 | 89.69| 89.94| 90.05     | 90.22     | **90.49**        | 90.34            | 90.47           |

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

|          | SL   | LS   | DML\(_1\) | DML\(_2\) | \(\epsilon\) 0.9 | \(\epsilon\) 0.99 | \(\epsilon\) 0.999 |
|----------|------|------|-----------|-----------|------------------|------------------|-----------------|
| **DR = 10\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 53.31| 54.19| 65.14     | 63.12     | 76.02            | 76.14            | **76.47**       |
| InceptionResNetv2 | 83.06| 83.14| 84.07     | 84.18     | 84.94            | **85.24**        | 84.84           |
| **DR = 20\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 72.59| 72.43| 75.07     | 76.14     | 77.04            | **80.46**        | 80.11           |
| InceptionResNetv2 | 91.37| 91.42| 91.35     | 91.43     | **91.9**         | 91.89            | 91.71           |
| **DR = 40\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 94.21| 94.56| 94.79     | 95.01     | **95.2**         | 95.07            | 95.12           |
| InceptionResNetv2 | 96.03| 96.23| 96.28     | 96.13     | 96.18            | 96.49            | **96.57**       |
| **DR = 100\%** |      |      |           |           |                  |                  |                 |
| Resnet101 | 97.85| 97.92| 98.1      | 97.95     | 98.11            | **98.14**        | 98.1            |
| InceptionResNetv2 | 98.01| 98.07| 98.25     | 98.17     | **98.41**        | 98.41            | 98.2            |

To improve the average classification accuracy on the test data compared to standard training in all scenarios. It 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, and in future work it
will be interesting to see whether our results carry over to other machine learning tasks where the
class similarity is not based on the semantic similarity of natural language words.

References

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in International
Conference on Machine Learning (ICML), pp. 41–48, ACM, 2009.

[2] C. N. Silla and A. A. Freitas, “A survey of hierarchical classification across different application
domains,” Data Mining and Knowledge Discovery, vol. 22, no. 1, pp. 31–72, 2011.

[3] C. Wu, M. Tygert, and Y. LeCun, “Hierarchical loss for classification,” 2017.
arXiv:1709.01062v1 [cs.LG].

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “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.

[5] A. Krizhevsky, “Learning multiple layers of features from tiny images,” tech. rep., University
of Toronto, 2009.

[6] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, “Zero-shot learning—a comprehensive
evaluation of the good, the bad and the ugly,” IEEE Transactions on Pattern Analysis and
Machine Intelligence, 2018.

[7] M. P. Kumar, B. Packer, and D. Koller, “Self-paced learning for latent variable models,” in
Advances in Neural Information Processing Systems (NeurIPS), pp. 1189–1197, 2010.

[8] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, “Automated curriculum
learning for neural networks,” in International Conference on Machine Learning (ICML),
pp. 1311–1320, PMLR, 2017.

[9] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, “Reverse curriculum generation
for reinforcement learning,” in Conference on Robot Learning (CoRL), pp. 482–495, 2017.

[10] D. Weinshall, G. Cohen, and D. Amir, “Curriculum learning by transfer learning: Theory and
experiments with deep networks,” in International Conference on Machine Learning (ICML),
pp. 5235–5243, PMLR, 2018.

[11] C. Bucilă, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in Proceedings of
the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining
(KDD), pp. 535–541, ACM, 2006.

[12] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” in Deep
Learning and Representation Learning Workshop: NIPS 2014, 2014.
[13] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, “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.

[14] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “FitNets: Hints for thin deep nets,” 2014. arXiv:1412.6550 [cs.LG].

[15] S. Zagoruyko and N. Komodakis, “Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer,” 2016. arXiv:1612.03928 [cs.CV].

[16] J. Yim, D. Joo, J. Bae, and J. Kim, “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.

[17] M. Orbes-Artega, J. Cardoso, L. Sørensen, C. Igel, S. Ourselin, M. Modat, M. Nielsen, and A. Pai, “Knowledge distillation for semi-supervised domain adaptation,” in Machine Learning in Clinical Neuroimaging (MLCN 2019), vol. 11796 of LNCS, pp. 68–76, Springer, 2019.

[18] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, “Born again neural networks,” 2018. arXiv:1805.04770 [stat.ML].

[19] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4320–4328, 2018.

[20] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, “Curriculumnet: Weakly supervised learning from large-scale web images,” in Proceedings of the European Conference on Computer Vision (ECCV), pp. 135–150, 2018.

[21] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, “Devise: A deep visual-semantic embedding model,” in Advances in Neural Information Processing Systems (NeurIPS), pp. 2121–2129, 2013.

[22] R. Müller, S. Kornblith, and G. Hinton, “When does label smoothing help?,” 2019. arXiv:1906.02629 [cs.LG].

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 2013. arXiv:1301.3781v3 [cs.CL].

[24] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in International Conference on Machine Learning (ICML), pp. 1188–1196, 2014.

[25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, IEEE, 2016.

[26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “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.
[27] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, IEEE, 2018.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “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.

[29] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in *British Machine Vision Conference (BMVC)*, BMVA Press, 2016.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, IEEE, 2017.

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in *NeurIPS 2017 Workshop Autodiff*, 2017.

[32] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, “Accurate, large minibatch SGD: Training ImageNet in 1 hour,” 2017. arXiv:1706.02677v2 [cs.CV].

[33] A. Sergeev and M. Del Balso, “Horovod: fast and easy distributed deep learning in TensorFlow,” 2018. arXiv:1802.05799v3 [cs.LG].
7. Appendix

7.1 Standard Deviations for Results

|     | SL | LS | 0.9 | 0.99 | 0.999 |
|-----|----|----|-----|------|-------|
| DR = 10% |    |    |     |      |       |
| Resnet50 | 0.6 | 0.47 | 0.89 | 0.62 | **0.78** |
| ResNeXt  | 0.46 | 0.53 | 0.15 | 0.07 | **0.11** |
| SeNet    | 0.34 | 0.41 | 0.15 | 0.09 | **0.25** |
| DR = 20% |    |    |     |      |       |
| Resnet50 | 1.31 | 0.64 | 0.21 | 0.27 | **0.14** |
| ResNeXt  | 0.39 | 0.27 | 0.13 | **0.08** | 0.16 |
| SeNet    | 0.02 | 0.3  | 0.15 | 0.21 | **0.23** |
| DR = 40% |    |    |     |      |       |
| Resnet50 | 0.1  | 0.34 | 0.08 | 0.18 | **0.1** |
| ResNeXt  | 0.23 | 0.37 | 0.12 | 0.07 | **0.14** |
| SeNet    | 0.15 | 0.23 | 0.13 | **0.14** | 0.02 |
| DR = 100%|    |    |     |      |       |
| Resnet50 | 0.14 | 0.17 | 0.12 | 0.14 | **0.11** |
| ResNeXt  | 0.05 | 0.23 | 0.04 | 0.06 | **0.08** |
| SeNet    | 0.12 | 0.24 | 0.14 | 0.11 | **0.14** |

Table 9: ImageNet. Top-1 results, standard deviations over four trials.
| DR = 10% | ResNet50 | 0.81 | 0.92 | 1.06 | 0.66 | 0.73 |
|---------|----------|------|------|------|------|------|
|         | ResNeXt  | 0.4  | 0.72 | 0.18 | 0.1  | 0.18 |
|         | SeNet    | 0.56 | 0.41 | 0.13 | 0.22 | 0.26 |
| DR = 20%| ResNet50 | 1.02 | 0.86 | 0.16 | 0.07 |
|         | ResNeXt  | 0.2  | 0.21 | 0.05 | 0.14 | 0.11 |
|         | SeNet    | 0.1  | 0.37 | 0.39 | 0.2  | 0.32 |
| DR = 40%| ResNet50 | 0.04 | 0.42 | 0.08 | 0.03 | 0.04 |
|         | ResNeXt  | 0.16 | 0.22 | 0.13 | 0.1  | 0.08 |
|         | SeNet    | 0.05 | 0.17 | 0.15 | 0.1  | 0.11 |
| DR = 100%| ResNet50 | 0.09 | 0.1  | 0.07 | 0.09 | 0.06 |
|         | ResNeXt  | 0.04 | 0.1  | 0.05 | 0.03 | 0.04 |
|         | SeNet    | 0.27 | 0.28 | 0.15 | 0.18 | 0.22 |

Table 10: ImageNet. Top-5 results, averaged over four trials.

| DR = 10% | WRN     | 0.67 | 0.78 | 0.9  | 0.63 | 0.41 | 0.81 | 0.6 |
|----------|---------|------|------|------|------|------|------|-----|
|          | DenseNet| 0.42 | 0.61 | 0.45 | 0.35 | 0.57 | 0.51 | 0.59|
| DR = 20%| WRN     | 0.1  | 0.6  | 0.29 | 0.36 | 0.17 | 0.13 | 0.31|
|          | DenseNet| 0.44 | 0.53 | 0.18 | 0.21 | 0.28 | 0.29 | 0.17|
| DR = 40%| WRN     | 0.67 | 0.21 | 0.4  | 0.26 | 0.3  | 0.2  | 0.23|
|          | DenseNet| 0.23 | 0.41 | 0.15 | 0.21 | 0.18 | 0.18 | 0.17|
| DR = 100%| WRN     | 0.21 | 0.23 | 0.14 | 0.28 | 0.12 | 0.08 | 0.11|
|          | DenseNet| 0.27 | 0.65 | 0.21 | 0.32 | 0.17 | 0.11 | 0.07|

Table 11: CIFAR100. Top-1 results, standard deviations over four trials.
|         | $\epsilon$ | SL | LS | DML$_1$ | DML$_2$ | 0.9 | 0.99 | 0.999 |
|---------|-------------|----|----|---------|---------|-----|------|-------|
| **DR = 10%** |             |    |    |         |         |     |      |       |
| WRN     | 0.88        | 0.57 | 0.63 | 0.41   | 0.3    | 0.29 | **0.32** |       |
| DenseNet| 0.64        | 0.54 | 0.27 | 0.33   | 1.02   | 0.57 | **0.44** |       |
| **DR = 20%** |             |    |    |         |         |     |      |       |
| WRN     | 0.32        | 0.45 | 0.23 | 0.24   | 0.18   | 0.15 | **0.11** |       |
| DenseNet| 0.29        | 0.32 | 0.25 | 0.37   | 0.18   | 0.11 | **0.17** |       |
| **DR = 40%** |             |    |    |         |         |     |      |       |
| WRN     | 0.31        | 0.71 | 0.21 | 0.27   | 0.25   | **0.25** | 0.2  |       |
| DenseNet| 0.12        | 0.54 | 0.11 | 0.17   | 0.19   | 0.36 | **0.14** |       |
| **DR = 100%** |           |    |    |         |         |     |      |       |
| WRN     | 0.18        | 0.55 | 0.24 | 0.24   | 0.1    | 0.14 | **0.06** |       |
| DenseNet| 0.22        | 0.38 | 0.14 | 0.24   | 0.08   | 0.1  | **0.04** |       |

Table 12: CIFAR100. Top-5 results, standard deviations over four trials.

|         | $\epsilon$ | SL | LS | DML$_1$ | DML$_2$ | 0.9 | 0.99 | 0.999 |
|---------|-------------|----|----|---------|---------|-----|------|-------|
| **DR = 10%** |             |    |    |         |         |     |      |       |
| Resnet101| 9.15        | 6.72 | 4.94 | 5.42   | 1.74   | 1.99 | **1.14** |       |
| InceptionResNetv2 | 1.17 | 1.43 | 1.21 | 0.94   | 0.71   | **0.93** | 0.6  |       |
| **DR = 20%** |             |    |    |         |         |     |      |       |
| Resnet101| 8.21        | 5.27 | 3.5  | 2.85   | 4.78   | 2.85 | **2.91** |       |
| InceptionResNetv2 | 0.8  | 0.95 | 0.45 | 0.63   | 0.34   | 0.58 | **0.25** |       |
| **DR = 40%** |             |    |    |         |         |     |      |       |
| Resnet101| 1.44        | 1.45 | 0.4  | 0.83   | 0.88   | **1.02** | 0.73 |       |
| InceptionResNetv2 | 0.19 | 0.34 | 0.21 | 0.24   | 0.18   | 0.16 | **0.22** |       |
| **DR = 100%** |           |    |    |         |         |     |      |       |
| Resnet101| 0.68        | 0.54 | 0.42 | 0.45   | 0.37   | 0.38 | **0.24** |       |
| InceptionResNetv2 | 0.32 | 0.21 | 0.16 | 0.09   | **0.11** | 0.15 | 0.11  |       |

Table 13: AWA2. Top-1 results, standard deviations over four trials.
| DR         | Model           | SL | LS  | DML₁ | DML₂ | 0.9 | 0.99 | 0.999 |
|------------|-----------------|----|-----|------|------|-----|------|-------|
| 10%        | Resnet101       | 11.7 | 9.82 | 4.79 | 4.12 | 1.77 | 1.97 | **0.79** |
|            | InceptionResNetv2 | 0.64 | 0.65 | 0.54 | 0.13 | 0.61 | **0.81** | 0.21 |
| 20%        | Resnet101       | 7.84 | 0.45 | 3.02 | 2.19 | 4.72 | **2.01** | 2.94 |
|            | InceptionResNetv2 | 0.22 | 0.23 | 0.38 | 0.44 | **0.23** | 0.27 | 0.3 |
| 40%        | Resnet101       | 0.37 | 0.32 | 0.51 | 0.32 | **0.28** | 0.74 | 0.23 |
|            | InceptionResNetv2 | 0.07 | 0.27 | 0.07 | 0.23 | 0.14 | 0.22 | **0.16** |
| 100%       | Resnet101       | 0.08 | 0.23 | 0.23 | 0.34 | 0.02 | **0.18** | 0.05 |
|            | InceptionResNetv2 | 0.12 | 0.29 | 0.11 | 0.2 | 0.06 | **0.37** | 0.17 |

Table 14: AWA2. Top-5 results, standard deviations over four trials.