SN Computer Science (2020) 1:245
https://doi.org/10.1007/s42979-020-00251-7

ORIGINAL RESEARCH

Deep curriculum learning optimization

Henok Ghebrechristos[1] · Gita Alaghband[1]

Received: 31 March 2020 / Accepted: 11 July 2020

Abstract

We describe a quantitative and practical framework to integrate curriculum learning (CL) into deep learning training pipeline to improve feature learning in deep feed-forward networks. The framework has several unique characteristics: (1) *dynamicity*—it proposes a set of batch-level training strategies (syllabi or curricula) that are sensitive to data complexity (2) *adaptivity*—it dynamically estimates the effectiveness of a given strategy and performs objective comparison with alternative strategies making the method suitable both for practical and research purposes. (3) Employs *replace–retrain* mechanism when a strategy is unfit to the task at hand. In addition to these traits, the framework can combine CL with several variants of gradient descent (GD) algorithms and has been used to generate efficient *batch-specific* or *data-set* specific strategies. Comparative studies of various current state-of-the-art vision models, such as FixEfficentNet and BiT-L (ResNet), on several benchmark datasets including CIFAR10 demonstrate the effectiveness of the proposed method. We present results that show training loss reduction by as much as a factor 5. Additionally, we present a set of practical curriculum strategies to improve the generalization performance of select networks on various datasets.

Keywords Curriculum learning optimization · Convolutional neural network · Deep learning · Information theory · Syllabus · Curriculum strategy

Introduction

Curriculum learning, which initially, in the context of machine learning, was formalized by Bengio et al. has in recent years gained some traction as a potential technique to *further improve deep learning* [1–5]. The general question CL attempts to answer is the question of how to find ordering of samples in which to supply and *effectively* train a model for a given task. Most curriculum learning techniques get their inspiration from human learning *where training is highly organized, based on education system and a curriculum which usually enables learning concepts in gradually increasing levels of difficulty* while considering previously learned concepts [1]. In machine learning, CL attempts to find some *optimal sequence* of training input (or *training tasks* for transfer learning) in which to present to the learning system to optimize the learning process compared to no-curriculum training.

In this text, we extend curriculum learning based on ranking or weighing (as defined by Bengio et al.) of individual samples with *dependency ranks* that relate two or more samples. The core idea as described in Ghebrechristos et al. is to present training sample to the system with samples such that adjacent samples have higher dependency rank than those that are not adjacent. Ranking may be based on a metric that measures the information content, overlap or statistical dependency between samples.

The main challenge and the reason why CL remains in the fringes of practical machine learning research, is partly due to our inability to efficiently determine the 'presumed difficulty' of training sample, or effectiveness of a syllabus, without supervision. Moreover, even when such a sequence is provided by a human teacher, it may not reflect the true difficulty in the perspective of the learning system. For example, in visual object recognition using feed-forward

This article is part of the topical collection "Machine Learning inPattern Analysis" guest edited by Reinhard Klette, Brendan McCane,Gabriella Sanniti di Baja, Palaiahnakote Shivakumara and LiangWang.

Henok Ghebrechristos
henok.ghebrechristos@ucdenver.edu

Gita Alaghband
gita.alaghband@ucdenver.edu

[1] Department of Computer Science, University of Colorado, Denver, CO 80014, USA

Published online: 29 July 2020

networks, it has been demonstrated that what makes an image difficult to a human observer may not always match whatever makes it difficult to a neural network classifier [6, 7]. This observation has been taken advantage of in the recent work on adversarial examples [8] and improved feature extraction [9] and is potentially why handcrafting an optimal training syllabus is impractical.

In the second part of this text, we explore the practicality of curriculum learning using a framework designed to automatically rank samples, propose and evaluate the effectiveness of an input sequence (syllabus) all by dynamically analyzing training samples, without supervision. This also alleviates the need for a human teacher to provide a reliable difficulty score (i.e. handcrafted syllabus), or when obtaining such a score by human teachers or transfer learning is too costly. The framework is also adaptive, in that it dynamically detects and updates a syllabus that is unfit for the task at hand.

In contrast to previous work, our method adopts image processing and information theory techniques to discover *optimal input paths* in the optimization landscape by assessing content of individual samples and their relationship to other samples in the dataset. Why is this a promising idea? There are two reasons. First, let us again consider human learning via curriculum. Two concepts that are close to each other (i.e. one concept depends on the other) are typically adjacent to each other in the syllabus. For instance, take a syllabus for Algebra course, where it is only sensible to present *relations and functions* after presenting *multi-step equations and inequalities* which in turn depend on basic concepts, such as *arithmetic*. For the learning to be effective, a concept or related concepts are presented after their prerequisites have been presented. Hence, in human learning, a syllabus ensures presentation of concepts that minimize confusion or maximize the chances of the student grasping and applying the concepts. This idea can directly be translated to machine learning language where a training syllabus presents samples such that their ordering ensures the minimization of training loss at each iteration while improving generalization performance. Although, this is the main motivation behind the design and implementation of our framework, we also consider two unique properties of deep neural networks that informed our methodology, namely the fact that most feed-forward networks behave like universal approximators [10] and their ability to learn and make relevant predictions when trained with noise-only data [6]. Intuitively, we believe these characteristics are reminiscent of their ability to discover complex functions that map individual training samples in a dataset to a corresponding label regardless of the features present in each sample. That is to say, the networks will find the most approximate function based solely on the dataset. With this observation, we can safely hypothesize that deep neural networks, when trained using deep learning techniques, are universal pattern extractors, not only the human recognizable pattern ones, but also patterns not visible to a human observer (noise-only dataset). This in turn allows us to venture out to explore weighting or raking strategies that do not rely on neither the network itself nor human but rather the dataset and its characteristics. Section 2 describes a framework, Deep-CLO, which by design exploits these observations for non-convex optimization.

To quantify our approach, we perform empirical evaluation and compare the performance of various models when using a curriculum which is based on different ranking options, and one control condition where low-ranked examples are presented first. The main results of this empirical study can be summarized as follows: (1) Convergence is always faster with curriculum learning. That is, models trained using curriculum learning achieve similar loss as no-curriculum training in a smaller number of iterations. (2) CLO has favorable impact on generalization, especially when the conditions for learning are hard: the task is difficult, a number of classes are significant, etc. (3) Networks training using curriculum sometimes achieve better generalization results in a smaller number of iterations compared to baseline, no-curriculum, training.

The contribution of this work is twofold: (1) we present the design and implementation of Deep-CLO that enables in-depth experiments of CL strategies while simplifying the experimentation process involving curriculum learning, (2) subsequently, we use this framework to identify training strategies that improve training and generalization performance of convolutional neural network (CNN) on select classification tasks.

The following section presents the theoretical merit of our extended definition, termed *curriculum learning optimization* (*CLO*), in the context of gradient descent (GD) algorithms and non-convex objective functions. We first define the effectiveness metric of a curricula for a given task *as the aggregated loss with respect to the optimal classifier over a fixed number of training steps*. We then prove that curriculum learning combined with various optimizers under different setups can significantly alter the optimization landscape in favor of convergence speed. This is supported by empirical evidence obtained by training several past and present state-of-the-art convolutional network architectures. We also show an improved generalization performance of networks trained using CLO.

Deep-CLO: Deep Curriculum Learning Optimization

We consider supervised learning tasks, such as classification [11, 12], comprising a training set $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ consisting n pairs of feature-label vectors, where the feature vector $x_i \in \mathbb{R}^{d_{2x}}$ and the corresponding label vector $y_i \in \mathbb{R}^{d_{2y}}$ are of dimensions d_x and d_y, respectively. A classification model can then be represented as mapping $F(;\theta) : X \to Y$, where X and Y denote the feature and label spaces, respectively, and θ denotes the variable vector (candidate *network weights*) involved in the mapping [13]. Given a feature vector $x_i \in X$, we represented the output of the model as $\hat{y}_i = F(x_i;\theta)$ and the error or loss of the model relative to the true label y_i as $e(\hat{y}_i, y_i)$. *Mean Squared Error* (*MSE*), *Mean Absolute Error* (*MAE*), *Hinges Loss* and *Cross-Entropy Loss* are amongst the loss functions used to measure errors introduced bythe model [14]. In this text, we emphasize the error function commonly used in classification tasks, the *cross-entropy loss*:

$$e_{\mathbf{CE}}(\hat{y}_i, y_i) = -\sum_{j=\mathbf{1}}^{d_y} y_i(j)\mathbf{log}\hat{y}_i(j). \tag{1}$$

Depending on the number of classes involved as well as the activation functions being used for the model, the cross-entropy function can be specified as *sigmoid cross entropy* [13] which is great for binary classification tasks with sigmoid function as the activation function for the output layer or *softmax cross entropy* [14] used for multi-class classification tasks with softmax function as the activation function. Given a loss function, the error of the model on the task can be represented as the sum of the errors on individual examples:

$$E(\theta : T) = \sum_{(x_i,y_i)\in T} e(\hat{y}_i, y_i) = \sum_{(x_i,y_i)\in T} e(F(x_i;\theta), y_i). \tag{2}$$

The purpose of curriculum learning optimization (CLO) is then to minimize the total loss introduced on the training set to achieve the most optimal set of weights θ. Formally, the objective function or criterion of deep learning model training can be represented as follows:

$$E^* = \min_{\theta\in\Theta} E(\theta : T), \tag{3}$$

where $\Theta = \mathbb{R}^{d_{2\theta}}$ denotes the variable domain or weight space of the model and d_θ denotes the dimension of variable vector θ.

CLO achieves E^* by proposing training input sequence (syllabus) for training via curriculum and by employing gradient decent (GD) to iteratively move in the direction of the steepest decent using directional derivative and negative gradient [15]. In other words, since the negative gradient $-\nabla_\theta E$ points directly opposite to E, we can minimize E by moving in the direction of the negative gradient using an input sequence that favors the global minima, i.e. at every iteration, gradient decent proposes a new value $E' = E - \epsilon\nabla_\theta E(\theta)$, where ϵ is the learning rate, a positive scalar determining the size of the update step. CLO is said to have converged when every element of the gradient is close to zero.

Deep curriculum learning optimization (Deep-CLO) is an umbrella term we use to describe our framework which can be used for training by combining CL with conventional deep learning techniques. As stated above, the loss functions used in deep learning usually have highly non-convex shape with many local minima, so the order in which training samples are presented have impact on learning. In contrast to conventional training, which only shuffles batches of training samples at most to achieve decent local minima, Deep-CLO uses an informed and deterministic ordering of samples to a similar end.

As described in Ghebrechristos et al., a curriculum strategy (or a syllabus) is decoupled from the core optimization algorithm as well as the model architecture. This choice is also inspired by how curriculum is employed in human learning (Sect. 1). A curriculum in human learning is applied to several students whose learning abilities may vary. Similarly, Deep-CLO is designed to enable optimization of various models having different architectures. This alleviates the need for the cumbersome task of modifying the native objective function of every model to capture the impact of a strategy. In our framework, a curriculum corresponds to the training examples. At an abstract level, it is a sequence of training criteria. Each training criterion in the sequence is associated with a separate set of weights or ranks of the straining examples. Formally, a syllabus $S = (x_\mathbf{1}, x_\mathbf{2}, \dots, x_\mathbf{M}) \in T$ of a training set containing $M \ll n$ samples is a computationally found order set such that $\epsilon_{x_1} \le \epsilon_{x_2} \le \dots \le \epsilon_{x_M}$ where ϵ_{x_i} corresponds to the rank of the ith sample as measure by a metric $\boldsymbol{m}$ taken from Table 1. A syllabus $\boldsymbol{S}$ for a given training set and model is considered effective if it outperforms a no-curriculum training of the same model after a fixed number of training updates.

Deep-CLO is built using a three-stage processing pipeline depicted in Fig. 1. In stage I, all samples of a batch are assessed and ranked using a prespecified metric m (Table 1). In stage II, the batch is ordered according to the rank of each sample. The ordered batch, or syllabus, is then supplied to the network for training. In stage III, the effectiveness of the syllabus is determined using the network's native loss function after training for a fixed number of batches. The number of batches used to control how often the syllabus is evaluated is a configurable hyperparameter. Below we discuss each stage in detail. The full recipe in an end-to-end training pipeline is presented in Table (Algorithm) 2.

Table 1 List of measures used in this study

Metric	Category	Implementation—given samples $x, x_1, x_2 \in T$ where b_x is normalized histogram of pixel intensities and i is an index of a pixel value in a sample
Entropy	Standalone	$E(x) = \sum_{i \in \chi, x \in T} b_x(i) \mathbf{log} \frac{N}{b_x(i)}$
Joint entropy (JE)	Distance	$\mathbf{JE}(x_1, x_2) = \sum_i b_x(i) \mathbf{log} b_x(i)$
Mutual information (MI)	Distance	$\mathbf{MI}(x_1, x_2) = E(x_1) + E(x_2) - \mathbf{JE}(x_1, x_2)$
K–L divergence (K–L)	Distance	$D_{\mathbf{K}\vert\vert\mathbf{L}}(x_1, x_2) = \sum_i x_{1_i} \mathbf{log} \frac{b_{x_1}(i)}{b_{x_2}(i)}$
Information variation (IV)	Distance	$\mathbf{IV}(x_1, x_2) = E(x_1) + E(x_2) - \mathbf{MI}(x_1, x_1)$
Conditional entropy (CE)	Distance	$\mathbf{CE}(x_1 \vert x_2) = E(x_1, x_2) - E(x_1)$, where $E(x_1, x_2)$ is the sum entropies of x_1 and x_2
Structural Similarity index (SSIM)	Distance	$\mathbf{SSIM}(x_1, x_2) = \frac{(2\mu_{x_1}\mu_{x_2} + C_1)(2\sigma_{x_1 x_2} + C_2)}{(\mu_{x_1}^2 + \mu_{x_2}^2 + C_1)(\sigma_{x_1}^2 + \sigma_{x_2}^2 + C_2)}$

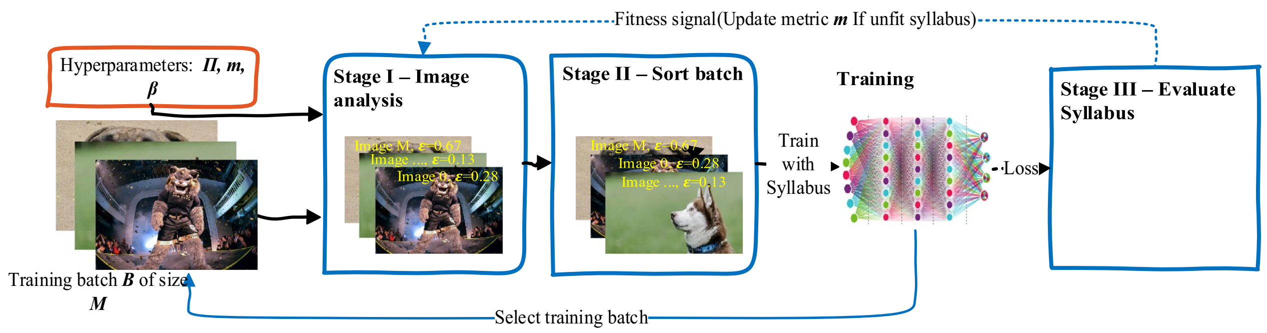

Fig. 1 (Ghebrechristos et al.). Processing stages of Deep-CLO. From left to right given batch B and hyperparameters read once at start: I. Rank each sample. II. Generate syllabus by ordering B according to rank of each sample which is then used to train a network. III. Evaluate syllabus using network loss

Assessing and Raking Training Samples

Feed-forward networks learn patterns of features from training and use layer-wise superposition of the features to generalize to unseen samples. To enable robust feature extraction and ease the pattern discovery, we are interested in generating curricula based on how samples are related to each other. This is not considered when employing conventional, batch shuffling-based training. We consider two types of metrics to measure these relationships: *statistical* and *information-theoretic measures*. These measures are further categorized into *standalone* and *distance* depending on the input(s) to the measure. If a measure takes two samples as input and returns a single value that relates the two samples, it is considered a distance measure. Otherwise, the measure is standalone and takes a single sample as input and returns a value that captures a certain characteristic of the sample.

To use information-theoretic measures, we model all samples as 2D random variables where each pixel is an independent and identically distributed random variable (*i.i.d*)

Table 2 (Algorithm 2) Curriculum training of a CNN network η

```
Input: Training Set $T$, Metric $m$, Order $o$, β, π
Outputs: Model Parameters $\theta$ (network weights)
1. Initialize iteration (training batch) counter iter and $\beta_{\eta \to S}$ to 0 and $f_S$ with
   continue
2. Draw training batch $B$ of size $M$ from $T$
      For $i = 0$ to $< M - 1$
       | 1.  Select input $s_i$
       | 2.  Compute rank $\varepsilon_{s_i}$
      End
    Syllabus = Sorted batch according to $\varepsilon$
3. Train network on $B$ using Syllabus, increment iter
4. If iter is equal to π
      Calculate syllabus-to-baseline loss ratio □ and set fitness signal $f_S$
         If $f_S$ is set to continue
            go-to step 2
         If $f_S$ is set to replace-rerun
            go-to step 1
        Otherwise go to 5
5. Yield $\theta$
```

Here, at least two *m* values, a primary measure and backup measures, from Table 2 are pre-specified. If no *m is* prespecified, *deep-clo* picks a primary and backup measures randomly from the set of measures listed in Table 1 realization. With this model, we utilize information-theoretic measures, such as entropy, to quantify information content of training samples. Below we discuss few measures. A complete list is presented in Table 1.

Information-Theoretic Measures

Information theory provides a theoretical foundation to quantify information content, or the uncertainty, of a random variable represented as a distribution [16, 17]. Information-theoretic measures of content can be extended to image processing and computer vision [18]. One such measure is *entropy*. Intuitively, entropy measures how much relevant information is contained within an image when representing an image as a discrete information source that is random [17]. Formally, let X be a discrete random variable with alphabet χ and a probability distribution function $p(x), x \in \chi$. The Shannon entropy [19] of χ is defined as

$$H(X) = \sum_{x\in\chi} p(x) \log \frac{1}{p(x)}, \tag{4}$$

where $0 \log \infty = 0$ and the base of the logarithm determines the unit, e.g. if base 2, the measure is in *bits* [20]. The term $-\log p(x)$ can be viewed as the amount of information gained by observing the outcome $p(x)$. Entropy is usually meant to measure the uncertainty of a continuous random variable. However, when applied to discrete images, this measures how much relevant information is contained within an image when representing the image as a discrete information source [17]. Here, we construct probability distribution associated with each image by binning the pixel values into histograms. The normalized histogram can then be used as an estimate of the underlying probability of pixel intensities, i.e., $p(i) = b_x(i)/N$, where $b_i(x)$ denotes the histogram entry of intensity value i in x, and N is the total number of pixels of x. With this representation, the entropy of an image x can be computed as:

$$E(x) = \sum_{i\in\chi, x\in T} b_x(i) \log \frac{N}{b_x(i)}, \tag{5}$$

where T is the training set and $\chi(s)$ represents the image as a vector of pixel values. While individual entropy is used to measure the standalone rank of a sample, we also used metrics that relate training samples. These include *joint entropy* (JE), *K–L divergence* (K–L), *mutual information* (MI), *information variation* (IV), *and conditional entropy* (CE). A complete list of the metrics used for this study is listed in Table 2. Readers are encouraged to refer to [16, 17, 21] for detailed treatment of these metrics and others.

Joint Entropy

By considering two random variables (X, Y) as a single vector-valued random variable, we can define the joint entropy $\mathrm{JE}(X, Y)$ with joint distribution $p(x, y)$ as follows:

$$\mathrm{JE}(Y, X) = -\sum_{x}\sum_{y} p(x, y) \log p(x, y). \tag{6}$$

When we model images as random variables, the joint entropy is computed by gathering joint histogram between the two images. For two samples, $x_1, x_2 \in T$, the joint entropy is given by:

$$\mathrm{JE}\left(x_1, x_2\right) = \sum_{i} b_x(i) \log b_x(i), \tag{7}$$

where $b_x(i)$ is the ith value in the joint histogram.

Kullback–Leibler (K–L) Divergence

K–L divergence [17] is another measure we use to assess similarity of adjacent training samples. It is a natural distance measure from the pixel distribution of a sample x_1 to another distribution x_2 and is defined as:

$$D_{\mathrm{K||L}}\left(x_1, x_2\right) = \sum_{i} x_{1i} \log \frac{x_{1i}}{x_{2i}}, \tag{8}$$

where i the index of a pixel value taken from the distribution.

Mutual Information

Mutual information (MI) is the measure of the statistical dependency between two or more random variables [16]. The mutual information of samples $x_1, x_2 \in T$ can be defined in terms of the individual entropies of both x_1 and x_2 and the joint entropy of the two samples $\mathrm{JE}\left(x_1, x_2\right)$:

$$\mathrm{MI}\left(x_1, x_2\right) = E\left(x_1\right) + E\left(x_2\right) - \mathrm{JE}\left(x_1, x_2\right). \tag{9}$$

As noted in [18], maximizing the mutual information between samples seems to try and find the most complex overlapping regions by maximizing the individual entropies such that they explain each other well by minimizing the joint entropy. As image similarity measure, MI has been found to be successful in many application domains [22].

Statistical Measures

Statistical metrics, on the other hand, measure the similarity (dissimilarity) of samples and typically use statistical measurements, such as mean 2μ and standard deviation 2σ. Few of the statistical measures used in this study are discussed below. Readers interested in further detail of these metrics are welcome to refer to [21].

Structural Similarity Index (SSIM)

SSIM is often used for predicting image quality using a reference image. Given two samples x_1 and x_2, the SSIM index [21] is given by:

$$\mathrm{SSIM}(x_1, x_2) = \frac{(2\mu_{x_1}\mu_{x_2} + C_1)(2\sigma_{x_1x_2} + C_2)}{\left(\mu_{x_1}^2 + \mu_{x_2}^2 + C_1\right)\left(\sigma_{x_1}^2 + \sigma_{x_2}^2 + C_2\right)}, \tag{10}$$

where the terms μ and σ are the mean and variances of the two vectors and $\sigma_{x_1x_2}$ is the covariance of x_1 and x_2. The constant terms C_1 and C_2 are used to avoid a null denominator.

Peak Signal-to-Noise Ratio (PSNR)

PSNR [21] is another objective metric widely used in CODECs to assess picture quality. PSNR can be defined in terms of the mean squared error (MSE). The MSE of two samples having the same size N is defined as:

$$\mathrm{MSE}(x_\mathbf{1}, x_\mathbf{2}) = \frac{1}{N^2}\sum_{i}^{N}\sum_{j}^{N}\left(x_{\mathbf{1}ij} - x_{\mathbf{2}ij}\right)^2. \tag{11}$$

The PSNR measure can then be expressed as:

$$\mathrm{PSNR} = 20\log_{10}\left(\frac{\mathrm{MAX}}{\sqrt{\mathrm{MSE}}}\right), \tag{12}$$

where MAX is the maximum possible pixel value of a reference image.

Sorting Batches of Samples

In contrast to traditional training approach, our proposed method does not shuffle individual batches. Instead it reorganizes then based on a concrete measure of each sample in the batch. The approach works as follows: a batch of training samples $B = \{x_1, x_2, \ldots, x_\mathrm{M}\} \subset T$ is selected from the training set. Each sample $x_k \in B$ is assigned a rank by analysing its pixel distribution using the specified metric m. We use two types of metrics: *distance* and *standalone*. If m is a distance metric, a reference sample $x_r \in B$ is used to rank a moving sample $x_m \in B$. Initially, the reference sample is chosen at random. For instance, consider the following setup: let m be the mutual information (MI) measure, the algorithm first selects an initial reference sample, $x_r = x_1$ and computes the MI-index or rank (ε) of every other sample, $x_2, \ldots, x_\mathrm{M}$, in the batch against x_r. If *asc* ordering is used, the sample with the smallest ε value is promoted to become a reference sample. This is repeated until the last sample is promoted and a syllabus is proposed. Note here, the syllabus,S_B, is an ordering of the samples according to their mutual information index. Given a proposed syllabus $S_\mathrm{B} = \{x'_1, x'_2, \ldots x'_\mathrm{M}\}$, the network first sees the initial reference sample, then the sample having the smallest dependency rank, ε value is fed to the network. The overall behaviour is that adjacent samples are closer to each other than those that are not adjacent. Closeness in this context is measured by the metric in use. The smaller the value ε, the closer the two samples are. When using a *standalone* metric, such as entropy, each sample is ranked. The entire batch is then sorted based on the specified ordering and the rank of each sample. m selected from a set of metrics is pre-specified as a learning parameter and can be updated during training if corresponding syllabus is deemed unfit. We experimented with several metrics and *asc* sorting order to observe the impact on training.

Syllabus Fitness Evaluation

We use the network's native loss function to determine the fitness of a given syllabus. The syllabus is evaluated after training for a fixed number of iterations. Fitness of a syllabus for a given network and training set T is determined using two configurable hyperparameters: number of iterations (can also be number of batches) $\boldsymbol{\pi}$ and the baseline performance $\boldsymbol{\beta}$ of the network on T averaged over $\boldsymbol{\pi}$. $\boldsymbol{\beta}$ is the threshold by which the syllabus's fitness is determined and is chosen to be the average baseline loss of the network over $\boldsymbol{\pi}$ number of iterations. Baseline performance of a network is the network's training performance without curriculum.

Syllabus fitness criteria: Once the network is trained on T for $\boldsymbol{\pi}$ number of iterations using a syllabus S, the losses are aggregated and the average loss,

$$\beta_{\eta\to S} = \frac{\sum_{i=0}^{\pi}\mathbf{loss}(i)}{\pi}, \tag{13}$$

where loss (i) is the ith iteration training loss, of the network associated with $\boldsymbol{S}$ is computed. The syllabus-to-baseline loss ratio, $\omega = {}^{\beta_{\eta\to S}}/_{\beta}$, is then used as the sole criterion to determine the fitness of the syllabus. Depending on the value of ω, a fitness signal f_S, that can take on one of three forms: *continue, stop,* or *replace–rerun,* and is propagated to the image analysis submodule. A syllabus is deemed fit if the ratio is less than or equal to **1** and f_S is set to *continue*. Otherwise f_S is set to *stop or replace* and the syllabus is considered unfit and discarded. If *replace–rerun* is propagated, then Deep-CLO adaptively proposes a new syllabus using a prespecified backup metric. Here, we make a naive assumption that the syllabus's training performance is as good as the baseline if the ratio is close to 1.

Experiments

Implementation detail and datasets—Our method is implemented with the TensorFlow library[1] [23] and training was done using a system optimized for deep learning research and developments. We present training and classification results obtained by training state-of-the-art image classification networks using different curriculum strategies described in Sect. 2 on CATSvsDOGS [24], CIFAR10, CIFAR100 [25].

Training—We trained several past and current state-of-the-art CNNs using open-source TensorFlow implementations.[2] Each network is first evaluated on the corresponding datasets to create baseline reference performance metrics for comparison. For each network, we used stochastic gradient descent (SGD) optimizer and its variant, Adaptive moment estimation (Adam) [26], a fixed momentum of 0.9, batch size of 8, and an exponentially decaying learning rate with factor 0.94 starting at 0.01. For the rest of training, we used recommended configurations by respective authors. We report empirical results gathered by training each network for at least 1 million iterations. We ensure all learning parameters and environment are identical, with varying networks, learning methods (curriculum vs no-curriculum) and optimizers, to rule out other factors of influence.

Networks—Variants of EfficientNet [27] and FixEfficientNet-L2 [28] are among the networks evaluated in this study. EfficientNet is a family of architectures generated using a novel model scaling method that uses simple yet highly effective compound coefficients to scale up CNNs in a more structured manner [27]. We experimented with EfficientNet-L2 [29] which surpassed state-of-the-art accuracy in 2019. FixEfficientNet-L2 is another iteration over EfficientNet which uses a new weight update procedure, FixRes [30] during training. This model currently holds the state-of-the-art in the ImageNet ILSVRC 2012 benchmark [31]. In addition to these models, we also experimented with various notable models including variants of the ResNet architecture [32], BiT-L [33] and ResNeXt [34] as well as one variant, Inception v2, of the GoogleNet (Inception) model family as well as MobileNet v3 [35]—a model architecture high optimized for mobile and embedded devices. To observe the training trends, we use curriculum settings or syllabi with varying measure m, $\pi = 100{,}000$, $o = asc$ and β value that is unique to each network and training set.

Testing scenarios—The standard testing scenario of a classification task is to train the models using a portion of the dataset and then test it in previously unseen, held-out test set. There is no prior exploration on the test set. This setting is preferred and able to clearly measure the generalizability of the networks with and without the proposed method, so we evaluate our Deep-CLO approach under the standard testing scenario.

Evaluation metrics—In a typical data classification problem, evaluation metrics are employed into two stages: training stage (learning) and testing stage. In training stage, the evaluation metric was used to optimize the classification algorithm. In other words, the evaluation metric was employed as the discriminator to discriminate and to select the optimal solution which can produce a more accurate prediction of future evaluation of a particular classifier. Meanwhile, in the testing stage, the evaluation metric was used as the evaluator to measure the effectiveness of produced classifier when tested with the unseen data. There are various types of evaluation metrics that can be used to evaluate the quality of classifiers with different aims. We report several metrics that show the impact of our method on training and generalization performance: *Training Trends*—the training loss over a period of training updates and trends in *Test Accuracy*—the ratio of correct predictions over the total number of test samples. In addition to these standard metrics, we also report metrics that are suited for operational models. Metrics that capture the tradeoff between correct classifications and misclassification, which are tuned to imbalance classifications and address the asymmetry of real-world costs associated with each class [36]. These metrics include:

- Precision (p) [37]—is used to measure the patterns of a given category that are correctly predicted from the total predicted patterns in that category. For instance, if we consider a cat category in a cat vs dog classification task to be a positive class (and dog a negative class), precision is used to measure the overall accuracy of the model on a cat from the total predicted patterns of a cat category, including false alarms. Formally, it is defined as the ratio of true positives and the sum of true positives and false positives. Precision value takes on values in the range 0–1 with values close to 1 indicating a classifier with high degree of precision.
- Recall (r) [37]—on the other hand captures the fraction of cat predictions that are correct. It is defined as the ration of true positive to the sum of true positive and true negative. It is used to assess model's ability to accurately predict (recall) a given category from a bucket of samples containing all categories. Similarly, recall takes value on the scale 0–1.
- F1-Score [37]—is the harmonic mean between recall and precision values. i.e. $\frac{2rp}{r+p}$.
- Receiver operator characteristic (ROC) curve [38]—another effective method commonly used for assessing

[1] https://github.com/h3nok/curriculum_learning_optimization.

[2] https://github.com/tensorflow/models/tree/master/research/slim.

the performance of a diagnostic test tools for medical applications. ROC considers the true positive, true negative, false positive and false negative predictions of the model. It is a plot that depicts the trade-off between sensitivity and 1-specificity where sensitivity is the fraction of positive patterns that are correctly classified, and specificity is the fraction of negative patterns that are correctly classified. The area under a ROC curve (AUC) is a single and most commonly used index for measuring the performance of a test. Unlike the threshold metrics above, the AUC value reflects the overall prediction performance of a classifier. The closer the AUC value is to 1, the better the performance of the models is on a real-world data.

Results and Analysis

Training

First, we evaluated training performance of all models using various curriculum strategies and compare the results to the baseline, state-of-the-art performance of each model on the corresponding dataset. We use the total loss, the sum of cross-entropy and regularization losses, as the primary evaluation criteria of the impact of each strategy on training. The results are depicted in Fig. 2.

With most curriculum strategies, we observe large improvement in training performance over the baseline. This shows that the proposed method accelerates training in agreement with prior theoretical investigation [1, 4]. For instance, we notice 15.6% absolute percentage improvement in reducing the training loss when training EfficientNet-B7 using MI syllabus on CIFAR10 dataset. The improvement is consistently observed on the other curriculum strategies. As can be seen in Fig. 3 left column plots, similar trends are observed with the ResNeXt and FixEfficientNet. Moreover, we notice that almost all curriculum strategies outperform the baseline and the improvements are statistically significant.

Whilst these results are interesting, to rule out other factors of influence and ensure repeatability across datasets, we whitelisted the best performing strategy in our CIFAR10 experiment and performed similar experiments on CIFAR100. The results are depicted in Fig. 3.

We noticed the test accuracy and training loss follow their usual course observed during CIFAR10 training. The results on both datasets show that CL has favorable impact on training performance. CL-trained models achieve the same level of loss but only require 2/3 of the steps compared to conventional training. These results also are in line with the results on ImageNet presented in Ghebrechristos et al.

Results on test set (standard scenario)—another observation from this experiment is that some curriculum strategies produce models that generalize better to unseen samples compared with the baseline (Fig. 3, right column). For instance, if we look at the test performance of Inception V2 trained using MI on CIFAR10 (Fig. 3, top right), the improvement on the test set is close to 15%. However, we also observe a degradation in generalizability with some curriculum strategies. SSIM syllabus, for instance, is a strategy that reduces generalization performance of ResNeXt-101 by 5% compared to the baseline.

This is interesting because it implies that the training improvements induced by the proposed methods directly translate to performance gains in generalization. This is beneficial because most models achieve similar accuracy as the baseline in significantly a smaller number of iterations. Based on the experiments, VGG is the only model that does not conform to these gains. The order of sample presentation in case of VGG may not be as significant compared to the other architectures. We believe this is somewhat related to how the model performs successive feature extraction in comparison to other. More experiments with more network architectures and datasets are needed to confirm this intuition.

The results suggest that the curriculum learning technique that incorporates complexity of training samples may be more effective not only in traversing the input space towards the ideal minimum as a way to expedite training and feature extraction, but also it has statistically significant impact on the classifier's generalization performance even when considering real-world scenarios (discussed below). Specifically, we see that the variance in gradient direction of points (capture by the trends in loss) decrease much faster when training using syllabus per batch in comparison to the tradition, random shuffling approach.

Further Study on Generalization

When considering applicability of these models to real-world scenarios, the models must go through validation process that considers variations and imbalance in real-world input. To this end, we further assess generalization performance of models trained with the proposed using the CATSvsDOGS dataset for binary classification task. We chose binary classification partly due to the rich and proven family of metrics for assessing applicability of classifiers to a given domain. Taking sample complexity into account, in the context of curriculum learning may, therefore, increase the likelihood to achieve a higher-quality local minimum during training which in turn produces robust models with better generalization performance but in a smaller number of training iterations. The metrics, including the number of iterations, are presented in column 3 of Table 3. Figure 4

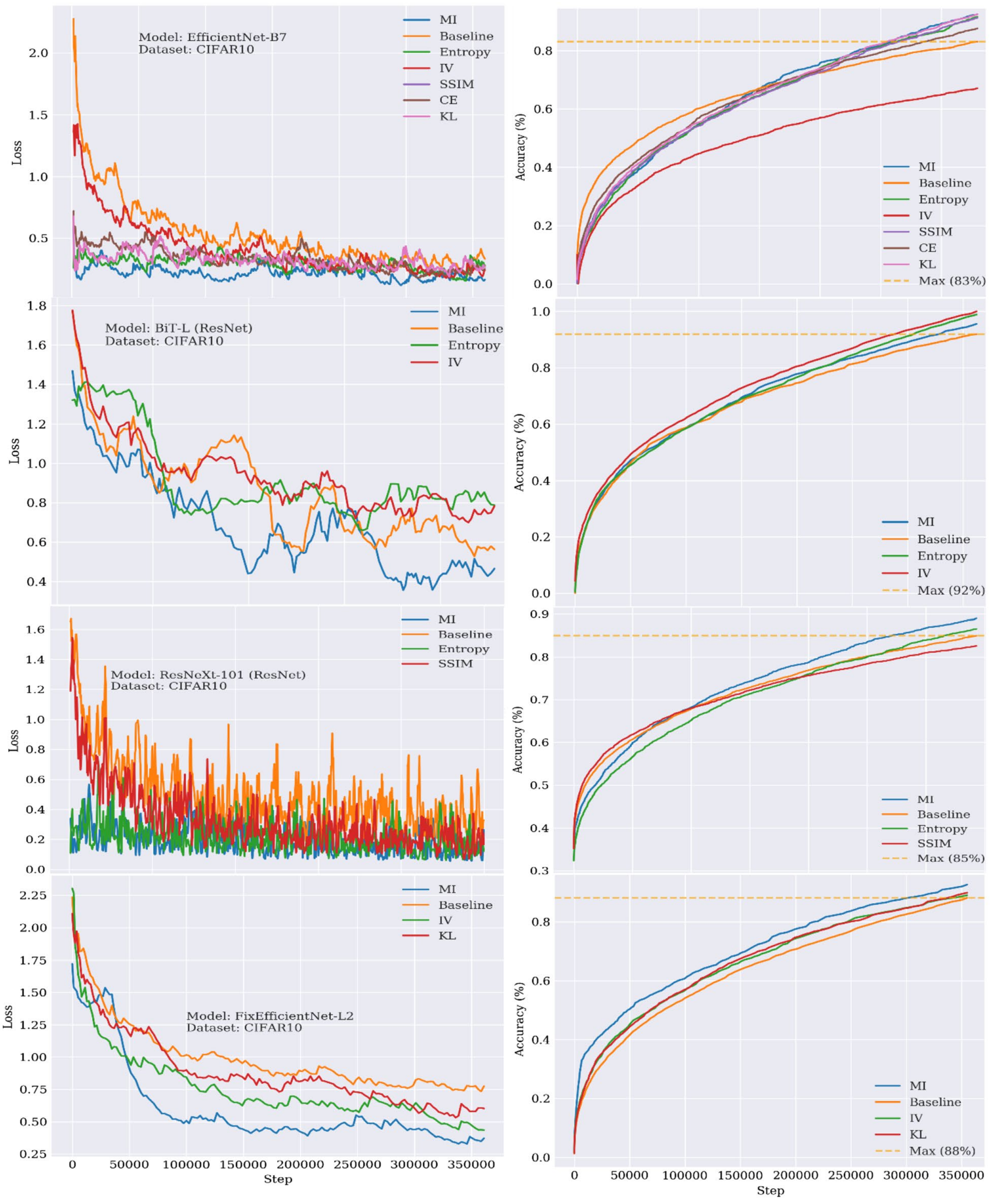

Fig. 2 Left. Comparison of CIFAR10 training performance of various models with and without CLO. Right. Comparison of test performance of the various models on CIFAR10 held-out test set. MobileNet achieve similar accuracy as the baseline in 2/3 of number of training steps. The ease the interpretation of the plots the *median* loss of the baseline during training and the *max* accuracy achieved during testing are highlighted in orange dashed lines

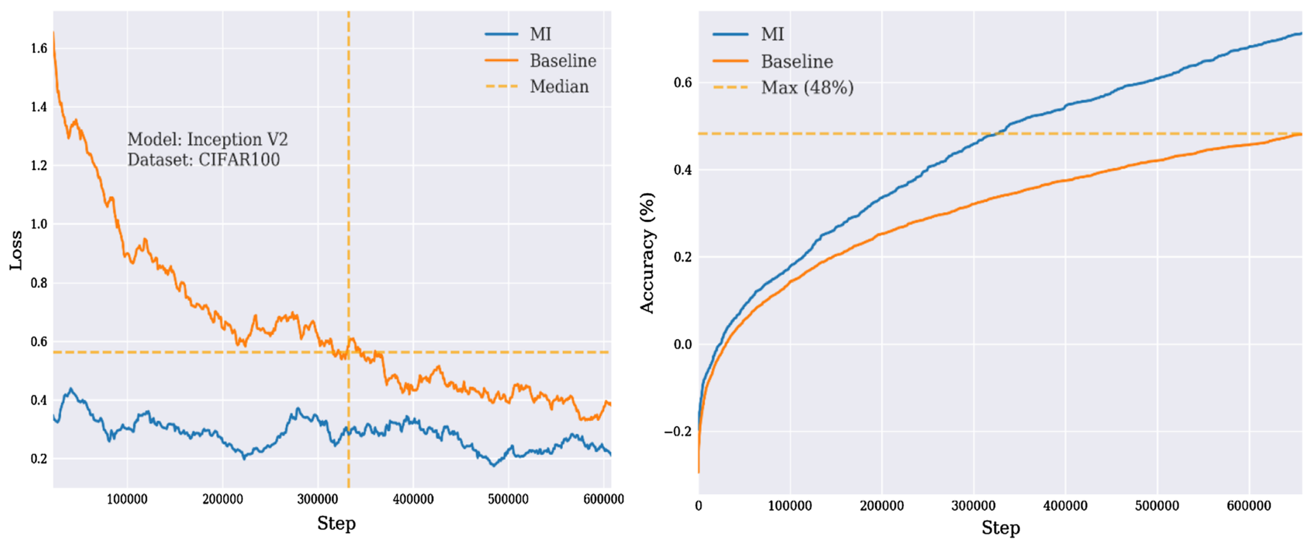

Fig. 3 Left. Comparison of inception training performance with and without MI syllabus. Right. Comparison of test performance of the same the network on test set. The model achieves similar test accuracy as the baseline in almost 1/2 of number of training steps

below depicts the ROC curve of MobileNet performance on CATSvsDOGS using various curriculum strategies. As can be seen in the plot, most CL strategies have AUC values greater than the baseline. For instance, K–L syllabus-trained MobileNet has AUC value of 0.985 compared to 0.943 of the baselines. These values indicate that CL-trained models have good measure of separability, making the proposed method suitable for both research purposes, to identify training strategies that expedite training, but also to produce more robust models for practical applications.

Optimization Methods

SGD and its variant Adam [2] work well for many optimization problems and can converge to a promising local or global optimum within a reasonable computation cost. Instead of computing loss on the whole dataset, SGD computes loss and weight updates on a batch of samples and updates the model variables by computing the loss function gradient instances by instances. SGD produces the same performance as regular gradient descent when the learning rate is low. Another variant of gradient descent widely used in practice is Adam. It is an adaptive learning rate method that combines the advantages of two SGD extensions—Root Mean Squared Propagation (RMSProp) [13] and Adaptive Gradient Algorithm (AdaGrad) [13]. It computes individual adaptive learning rates for different parameters of the model. Unlike SGD, Adam updates exponentially moving averages of the gradients and the square gradients whose decay rates are controlled by hyperparameters [26]. In this section of the experiment, we use both optimization techniques to verify that our proposed method's performance is invariant to the type of technique used to update model parameters. The results are depicted in Fig. 5.

The results show that the proposed method has comparable impact on training performance when combined with SGD or Adam optimizer. In line with previous observations, SGD progresses to find a minimum, but it takes significantly longer than Adam. We observe a training loss reduction by factor of 1.75 when Deep-CLO is employed alongside of Adam, while the loss reduction with SGD is expedited by a factor of 1.8. The gaps in loss reduction as depicted in Fig. 5

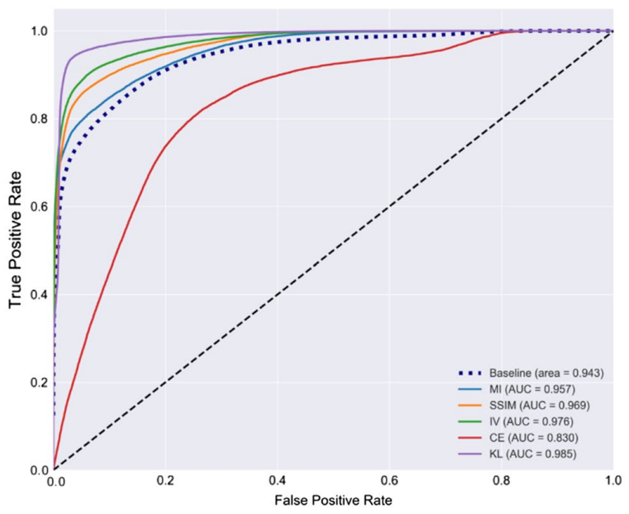

Fig. 4 ROC curve that captures generalization performance of MobileNet on Cats vs Dogs with various CL strategies. The closer the AUC value is to 1 the better the model is at generalizing to real-world data

Table 3 Comparison of generalization (real-time inference) capabilities of various models with and without curriculum learning

Model	Strategy	# Iterations (10k)	*p*	*r*	F1-Score	AUC	Acc (%)
ResNeXt-101	Baseline	19	0.54	0.55	0.54	0.74	0.63
ResNeXt-101-MI	MI	**15**	**0.67**	**0.73**	**0.70**	**0.81**	**0.78**
VGG16	Baseline	6	0.83	0.85	0.84	0.89	0.91
VGG16-MI	MI	6	**0.86**	0.75	0.80	0.82	0.84
BiT-ResNet	Baseline	15	0.67	0.74	0.70	0.73	0.77
BiT-ResNet-MI	MI	**19**	**0.75**	**0.88**	**0.81**	**0.85**	**0.88**
EfficientNet-B7	Baseline	8	0.58	0.53	0.55	0.992	0.88
EfficientNet-B7-MI	MI	**6.7**	**0.63**	**0.65**	**0.64**	**0.994**	**0.95**
MobileNet V1	Baseline	6	0.88	0.88	0.88	0.992	0.88
MobileNet V1-MI	MI	**4.8**	**0.95**	**0.95**	**0.95**	**0.994**	**0.95**
Inception v2	Baseline	6	0.94	0.95	0.94	0.94	0.93
Inception v2-MI	MI	**4.75**	**0.97**	**0.93**	**0.95**	**0.96**	**0.94**

The metrics that outperform the baseline are highlighted in bold

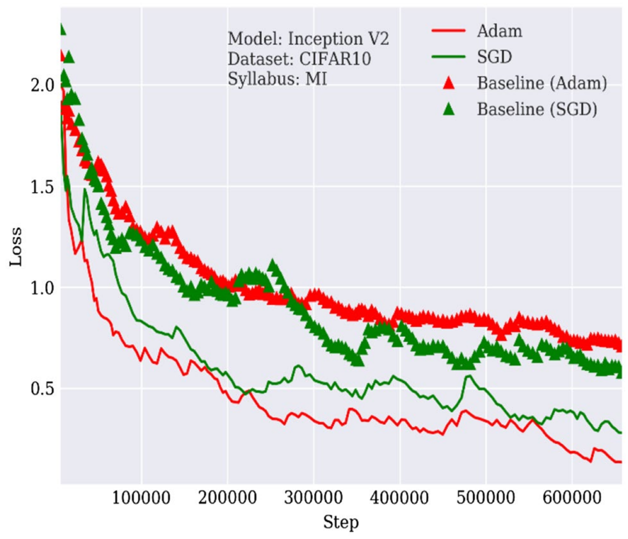

Fig. 5 Comparison of Adam and SGD optimizers on training performance of Inception model when trained using MI CL strategy

are inherent to the optimizers. This is because SGD is much more reliant on a robust initialization and annealing schedule and may get stuck in saddle points rather than local minima. So usually, SGD takes a greater number of iterations compared to its variants. Adam, on the other hand, is an adaptive momentum and adaptive learning rate algorithm which does not rely on robust initialization. Given these results, we hypothesize that our proposed method is optimizer agnostic. In other words, the various CL strategies have similar impact on training regardless of which optimizer is used.

Selecting a Strategy

Most neural network-based machine learning algorithms come with millions of tunable and user-specified parameters. Many of these parameters are encoded in the network architecture and are tuned by the training procedure. In CNNs, for instance, these types of parameters include the convolution filter banks or weights which are tuned for a given dataset by the backpropagation algorithm. While these are types of parameters that are automatically fine-tuned, there also exist parameters, such as learning rate and batch-size, commonly known as hyperparameters, which are considered as tuning knobs of the learning system that researchers and practitioners use to control the behavior of algorithm when optimizing its performance on a given dataset. Hyperparameter tuning for performance optimization is a challenging problem and is a subject of many machine learning researchers. As it currently stands, there are no established rules for predicting the right set of parameters that guarantee best performance for a given dataset [39]. Deep-CLO introduced one such hyperparameter, namely the metric (m), used to construct a curriculum strategy. In this section, we discuss our observation of one potential approach to identify a strategy that is best suited for a given dataset. A more in-depth study related to this is under way and will be considered in future publications.

Capturing Structural Information of Dataset

Figure 3 depicts the probability distribution of entropy metric of the CIFAR10 and 100 datasets. The plots capture the distribution of the entropy value of each sample as well as the histogram of these values across classes (labels). We have found a strong correlation between extracted the structural information of a dataset (such as distribution

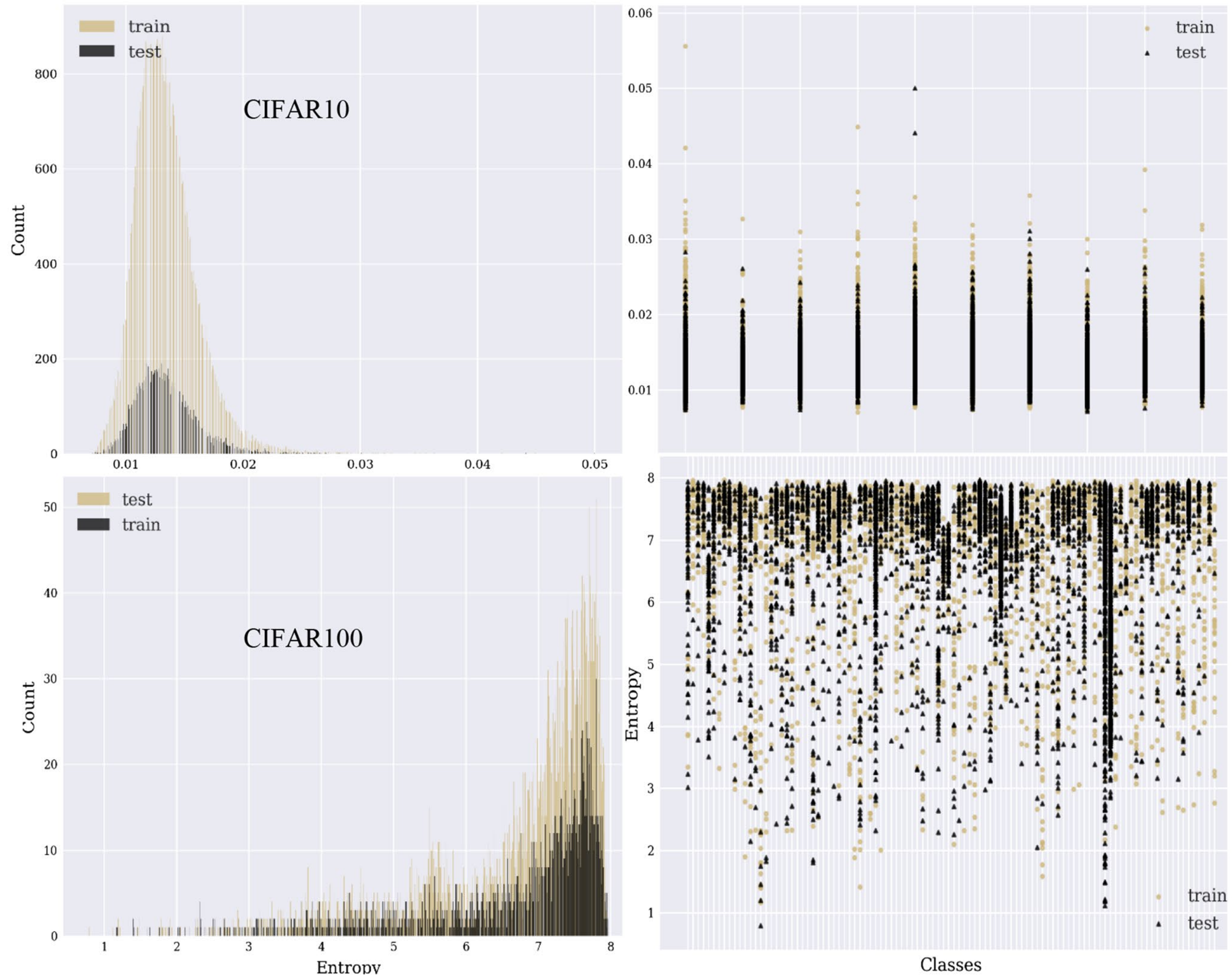

Fig. 6 Entropy value distribution (left) and distribution of Entropy value across classes for both CIFAR10 (top) and CIFAR100 (bottom) datasets

of Entropy) to the training and generalization trends of select models on that dataset. To illustrate, consider the EfficientNet-B7 model performance, (Fig. 2), first column. The model achieves training loss of 0.3 when trained using entropy strategy on CIFAR10 dataset compared to 0.9 of the baseline—a statistically significant loss reduction which is induced by the proposed learning strategy. We believe this performance optimization is somehow related to the structural information of the dataset as captured by the metric. Entropy-based strategy favors such a reduction since entropy distribution (Fig. 6 top) of the training set is a normal distribution with a mean of 0.1 and standard deviation of 0.0002. This structural information is also consistent with the test set which enables us to predict the generalization gains of the strategy. As can be seen in Fig. 2, the entropy-based model achieves similar accuracy as that of the Baseline in 3/4th of the number of training steps. At the end, it beats the Baseline model by achieving 91% accuracy compared to 82%. On the other hand, we noticed performance degradation when the same model is trained using the same strategy on the CIFAR100 dataset. This observation that we argue is predictable if one sees the structural information of CIFAR100 as captured by the strategy in use (Fig. 6 bottom). We see that the entropy of every sample in CIFAR100 dataset significantly deviates from one another by at least standard deviation of 2.6. In addition, entropy value across the labels also shares such discrepancy making the strategy less effective at optimizing the model.

Although these kinds of characteristics are well suited to describe datasets, we believe, when combined with our recommended training strategy, they also enable us to recommend appropriate classification model and strategy for a given dataset. Extracting and visualizing structural information of the various metrics used in this study are relatively cumbersome and we do not have a thought-out and efficient solution at this stage.

Conclusion

In this paper, we present a framework for assessing and ranking training samples and used this framework to investigate *curriculum learning optimization*, an extension of stochastic gradient descent in which samples of a batch are presented to the learning system based on a rank that captures inherent characteristics of each sample and its relationship to other samples. We start with an extended definition of curriculum learning in the context of *deep learning* which we termed Deep-CLO and present results that showcase both training and generalization performance improvements compared to conventional, no-curriculum training. Our proposed training framework, which is designed to make research in curriculum learning practical, provides the groundwork that enabled us to construct and investigate various training strategies for classification models using varying *convolutional architectures* and benchmark datasets. Compare to previous work, ours alleviates the need to rank difficulty of samples by hand (hand-engineering syllabus) or using multiple training passes. Our framework dynamically proposes and evaluates syllabus by integrating image analysis techniques that capture characteristics of each training example into deep learning-based training pipeline. Deep-CLO is also modular and independent of the model architecture, which allows each component to be improved separately without inducing any change to the model architecture. The results suggest that while sample ordering does affect the training process, the optimal order in which samples are presented may vary based on the dataset and algorithm used. With all strategies, we found loss reduction at the initial stages of training to be the most consistent signal that showcases the impact of our method. We believe our approach is optimizer agnostic. However, it is sensitive to the type of dataset used for training and potentially the model architecture. In addition to training performance, we also notice improvements in generalization performance both in standard testing scenarios and scenarios that consider real-world variations in input.

Our primary aim with this work is to investigate the impact and practicality of curriculum learning for off-the-shelf, computer vision models. However, it would also be nice to understand the general principles that make some curriculum strategies work better than others. This is the subject of our future work. In particular, correlating a strategy to a network and to a dataset or both will allow us to reap the advantages of CL by minimizing human involvement and introducing determinism into the training and model deployment processes. Currently, these processes are based on trial-and-error approaches that also rely on significant human expertise.

Compliance with ethical standards

Conflict of interest On behalf of all authors, I state that there is no conflict of interest.

References

1. Bengio Y, Louradour J, Collobert R, Weston J. Curriculum learning. 2009. https://doi.org/10.1145/1553374.1553380.
2. Graves A, Bellemare MG, Menick J, Munos R, Kavukcuoglu K. Automated curriculum learning for neural networks. In: Proceedings of the 34th international conference on machine learning, vol. 70. ICML'17. 2017, pp. 1311–1320.
3. Avramova V. Curriculum learning with deep convolutional neural networks. Thesis. KTH Royal Institute of Technology. 2015, p. 119.
4. Weinshall D, Cohen G, Amir D. Curriculum learning by transfer learning: theory and experiments with deep networks. ArXiv180203796 Cs, Feb. 2018, [Online]. arXiv:1802.03796. Accessed 15 Jun 2018.
5. Henok G, Gita A. Information theory-based curriculum learning factory to optimize training. In: Asian conference on pattern recognition, Auckland, Zew Zealand, 2019.
6. Zhang C, Bengio S, Hardt M, Recht B, Vinyals O. Understanding deep learning requires rethinking generalization. ArXiv161103530 Cs, Nov. 2016, [Online]. arXiv:1611.03530. Accessed 02 Apr 2018.
7. Martin CH, Mahoney MW. Rethinking generalization requires revisiting old ideas: statistical mechanics approaches and complex learning behavior. ArXiv171009553 Cs Stat, Oct. 2017. [Online]. arXiv:1710.09553. Accessed 12 Nov 2018.
8. Szegedy C et al. Intriguing properties of neural networks. ArXiv13126199 Cs, Dec. 2013. [Online]. https://arXiv.org/abs/1312.6199. Accessed 26 Oct 2018.
9. Ghebrechristos H, Alaghband G. Expediting training using information theory-based patch ordering algorithm. Las Vegas: CSCI; 2018. p. 6.
10. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Netw. 1989;2(5):359–66. https://doi.org/10.1016/0893-6080(89)90020-8.
11. Kotsiantis SB. Supervised machine learning: a review of classification techniques. Emerg Artif Intell Appl Comput Eng. 2017;160:20.
12. Deming WE, Morgan SL. The elements of statistical learning. Amsterdam: Elsevier; 1993.
13. Zhang J. Gradient descent based optimization algorithms for deep learning models training. ArXiv190303614 Cs Stat, Mar. 2019. [Online]. arXiv:1903.03614. Accessed 16 Nov 2019.
14. Janocha K, Czarnecki WM. On loss functions for deep neural networks in classification. *ArXiv170205659 Cs*, Feb. 2017, [Online]. arXiv:1702.05659. Accessed 16 Jun 2018.
15. Goodfellow I, Bengio Y, Courville A. Deep learning. New York: MIT Press; 2016.
16. Cover TM, Thomas JA. Elements of information theory. New York: Wiley; 2006. p. 774.
17. Feixas M, Bardera A, Rigau J, Xu Q, Sbert M. Information theory tools for image processing. Synth Lect Comput Graph Animat. 2014;6(1):1–164.
18. Leff HS, Rex AF, editors. Maxwell's demon: entropy, information, computing. Princeton: Princeton University Press; 1990.
19. Shannon CE. A mathematical theory of communication. Bell Syst Tech J. 1948;27:55.

20. Bonev BI. Feature selection based on information theory. New York: Springer; 2010. p. 200.
21. Horé A, Ziou D. Image quality metrics: PSNR vs. SSIM. Aug. 2010, pp. 2366–2369. https://doi.org/10.1109/ICPR.2010.579.
22. Russakoff DB, Tomasi C, Rohlfing T, Maurer Jr CR. Image similarity using mutual information of Torsten Rohlfing. In: 8th European conference on computer vision (ECCV, 2004, pp. 596–607).
23. Abadi M, et al. TensorFlow: a system for large-scale machine learning. ArXiv160508695 Cs, May 2016. [Online]. arXiv:1605.08695. Accessed 23 Jun 2018.
24. Parkhi OM, Vedaldi A, Zisserman A, Jawahar CV. Cats and dogs. In: 2012 IEEE conference on computer vision and pattern recognition, Jun. 2012, pp. 3498–3505. https://doi.org/10.1109/CVPR.2012.6248092.
25. Krizhevsky A. Learning multiple layers of features from tiny images. New York: Springer; 2009. p. 60.
26. Kingma DP, Ba J. Adam: a method for stochastic optimization. ArXiv14126980 Cs, Dec. 2014. [Online]. arXiv:1412.6980. Accessed 16 Jun 2018.
27. Tan M, Le QV. EfficientNet: rethinking model scaling for convolutional neural networks. ArXiv190511946 Cs Stat, Nov. 2019, [Online]. arXiv:1905.11946. Accessed 19 Mar 2020.
28. Touvron H, Vedaldi A, Douze M, Jégou H. Fixing the train-test resolution discrepancy: FixEfficientNet. ArXiv200308237 Cs, Apr. 2020. [Online]. arXiv:2003.08237. Accessed 20 May 2020.
29. Xie Q, Luong M-T, Hovy E, Le QV. Self-training with noisy student improves ImageNet classification. ArXiv191104252 Cs Stat, Apr. 2020. [Online]. arXiv:1911.04252. Accessed 20 May 2020.
30. Touvron H, Vedaldi A, Douze M, Jégou H. Fixing the train-test resolution discrepancy. ArXiv190606423 Cs, Mar. 2020, [Online]. arXiv:1906.06423. Accessed 22 May 2020.
31. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, 2012, pp. 1097–1105, [Online]. https://papers.nips.cc/paper/4824-imagenet-classification-w. Accessed 17 Sept 2016.
32. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
33. Kolesnikov A, et al. Big transfer (BiT): general visual representation learning. ArXiv191211370 Cs, May 2020, [Online]. arXiv:1912.11370. Accessed 20 May 2020.
34. Yalniz IZ, Jégou H, Chen K, Paluri M, Mahajan D. Billion-scale semi-supervised learning for image classification. ArXiv190500546 Cs, May 2019, [Online]. arXiv:1905.00546. Accessed 22 May 2020.
35. Howard AG, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. ArXiv170404861 Cs, Apr. 2017. [Online]. arXiv:1704.04861. Accessed 06 Apr 2019.
36. Koyejo OO, Natarajan N, Ravikumar PK, Dhillon IS. Consistent binary classification with generalized performance metrics. New York: Springer; 2014. p. 9.
37. Hossin M, Sulaiman MN. A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manag Process. 2015;5(2):1–11. https://doi.org/10.5121/ijdkp.2015.5201.
38. Kumar R, Indrayan A. Receiver operating characteristic (ROC) curve for medical researchers. Indian Pediatr. 2011;48(4):277–87. https://doi.org/10.1007/s13312-011-0055-4.
39. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res. 2012;13:25.