Information Theory-Based Curriculum Learning Factory to Optimize Training

Henok Ghebrechristos(✉) and Gita Alaghband

University of Colorado, Denver, CO 80014, USA
{henok.ghebrechristos,gita.alaghband}@ucdenver.edu

Abstract. We present a new system to optimize feature extraction from 2D-topological data like images in the context of deep learning using correlation among training samples and curriculum learning optimization (CLO). The system treats every sample as 2D random variable, where a pixel contained in the sample is modelled as an independent and identically distributed random variable (i.i.d) realization. With this modelling we utilize information-theoretic and statistical measures of random variables to rank individual training samples and relationship between samples to construct syllabus. The rank of each sample is then used when the sample is fed to the network during training. Comparative evaluation of multiple state-of-the-art networks, including, ResNet, GoogleNet, and VGG, on benchmark datasets demonstrate a syllabus that ranks samples using measures such as Joint Entropy between adjacent samples, can improve learning and significantly reduce the amount of training steps required to achieve desirable training accuracy. We present results that indicate our approach can produce robust feature maps that in turn contribute to reduction of loss by as much as factors of 9 compared to conventional, no-curriculum, training.

Keywords: Deep learning · Curriculum learning · Convolutional neural network

1 Introduction

Humans and other organisms, as in supervised learning, can learn to acquire knowledge and perform tasks by observing a sequence of labelled concepts. Supervision is often accompanied with a curriculum in human teaching. Hence the order in which topics are presented is not random when a teacher uses curriculum to teach. The ordering is such that simple concepts are presented early, progressing through concepts with increasing difficulty. When used in supervised machine learning, curriculum-based training (Fig. 1) exposes samples to the learning system in a predetermined order. The basic idea is to present samples that have low complexity at the start of training and gradually increase complexity of samples fed to the network over the course of training. Hence, at core of this approach lies ranking (weighting) training samples based on their level of presumed difficulty.

Many techniques described in the literature consider difficulty level of a sample to be proportional to a chosen distance metric between the output label and the actual truth label of the sample [3, 5]. The distance metric is often provided by the *loss function* used

S. Palaiahnakote et al. (Eds.): ACPR 2019, LNCS 12046, pp. 409–423, 2020.
https://doi.org/10.1007/978-3-030-41404-7_29

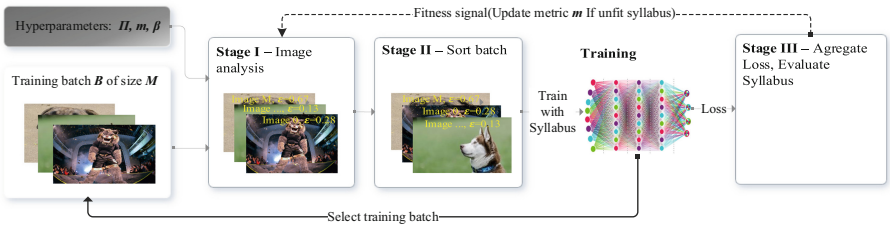

Fig. 1. Curriculum learning

by the learning algorithm, since it already attempts to minimize the distance between the expected and truth sample labels. In order to use this approach, one must employ two training passes: the first to measure the difference between the true label and predicted label value of each sample to get the presumed difficulty (or rank) of the sample. The training set is then ordered according to rank of each sample to form a curriculum used to train the network in the second training pass. Although one could potentially determine a good curriculum.

In contrast to the existing class of curriculum algorithms, our proposed method combines information theory tools (ITT) with curriculum learning to assess and adaptively order training samples to form an input path for training a network (Fig. 1). We utilize ITT-based image analysis techniques to assess each sample and its relationship with other samples to determine the time the sample is fed to the network.

A unique feature our approach is that the syllabus is generated, enforced and evaluated at training time using a node, curriculum factory, integrated into training pipeline. The node is designed to handle both online and batch training modes. When training using batches, a random subset (batch) of samples of size $\boldsymbol{M}$ from the training set is processed and the weights are updated based on the cumulative error. With online training, one sample is fed to the network at every iteration and weights are updated based on error corresponding to that sample. During batch training, the samples in a batch are ordered to form a syllabus corresponding to that batch. The primary means of ordering is a metric $\boldsymbol{m}$ that ranks a sample by measuring its content. The syllabus and batch are then supplied to the network to train via curriculum learning.

Supervised training seeks to minimize a global loss function that includes distance component as well as a regularization term. The distance component, that measures learning progress overtime, is used to evaluate the fitness of a syllabus. A syllabus deemed unfit is discarded and\or replaced in the early stages of training. A syllabus that incurs significant overhead to training time is also considered unfit and blacklisted. We report experimental results conducted using batch sizes of 8 and 16.

To summarize, this paper makes two main contributions. Currently there exists no methods in the literature that take characteristics of training data into account to expedite non-convex optimization via curriculum learning or other means. As the first contribution, we present a curriculum learning algorithm that reduces training loss at each iteration by ordering batches to form a syllabus. When used in stochastic gradient descent (SGD)-based training, our algorithm expedites training and reduces the overall loss by as much as a factor of 9 without compromising generalization performance. Second,

we present results that showcase improved generalization performance of popular CNN models on benchmark datasets in comparison to baseline, state-of-the-art performance.

2 Related Work

Most work on curriculum learning utilize error-based correction [3] to uniformly sample from training set as a means to speed up training. Other methods have been proposed [4, 5] to sample, weigh and sort training samples to improve accuracy and expedite training. Most of these methods either require modification of the objective function with a term for sample valuation [6] or run training procedure twice to obtain accurate representation of sample effectiveness [7]. Use optimization loss to valuate and increase the replay probability of samples that have high expected-learning-progress. The authors use a weighted importance sampling method to counter bias of sampling. Similarly, [8] proposed to sample batches non-uniformly [5]. Proposed an automatic curriculum learning method for LSTM for the NLP application. They use a non-stationary multi-armed bandit algorithm of getting a reward signal from each training sample to define a stochastic syllabus [9]. Use influence functions from robust statistics to measure the effect of parameter changes or perturbations of the training data such as pixel values which was applied to debugging models, and training-set attack. In addition, [10] describes a pre-processing algorithm that divides each input image into patches (image regions) and constructs a new sample by reorganizing the patches according their statistical or content summary. The authors present results that support training a CNN network in this manner can aid feature extraction and produce networks that tolerate adversarial attacks.

Unlike these approaches, our curriculum was formed by presenting the training samples to the network in order of increasing similarity or dissimilarity, measured by the sample's statistical pixel (content) distribution. Our method takes advantage of similarity and image content measures to define and propose a stochastic syllabus. *During batch training, all samples within a batch are sorted in ascending order based on a standalone or mutual measure.* When a standalone measure is used, every sample in the batch is ranked according to an index that measures some characteristics of that sample. For instance, when entropy is used as a ranking metric, the batch is sorted based on the entropy value of each sample. If mutual or distance measure is employed, the entire batch is sorted based on similarity or dissimilarity to a reference sample. *Our results indicate that this approach* expedites a search for local minima which helps expedite the overall training.

3 Proposed Method

The goal of training CNNs is to try and determine optimal network weights to approximate target mapping $\boldsymbol{g} : \boldsymbol{x_i} \rightarrow \boldsymbol{y_i}$ [11] where $\boldsymbol{x_i}$ is the input vector and $\boldsymbol{y_i}$ is the corresponding desired output taken from a training set,

$$\boldsymbol{T_s} = \left\{\left(\boldsymbol{x_n}, \boldsymbol{y_n}\right) : 1 \leq \boldsymbol{n} \leq \boldsymbol{N}\right\}, \tag{1}$$

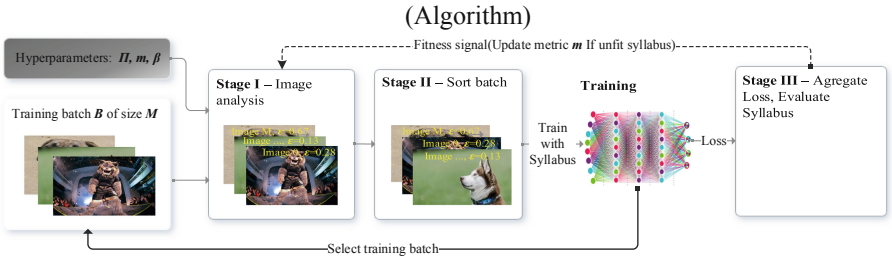

Fig. 2. Processing stages within *sp-module*. From left to right given batch B and hyperparameters read once at start: I. Rank each sample. II. Generate syllabus by ordering B according to rank of each sample which is then used to train a network. III. Evaluate syllabus using network loss.

consisting of $\boldsymbol{N}$ samples. If we consider a $\boldsymbol{j^{th}}$ training batch $\boldsymbol{B_j} \subset \boldsymbol{T_s}$ containing $\boldsymbol{M}$ samples, a curriculum training syllabus, $\boldsymbol{Syllabus_{B_j}}$ of $\boldsymbol{B_j}$ is generated by ranking and ordering every sample in the batch based on some *measure* $\boldsymbol{m}$. The syllabus is then applied to train the network. Note here, $\boldsymbol{M} = 1$ corresponds to online training where samples (not batches) are fed one by one. In our system, $M = 1$ is a special case where the syllabus corresponds to ordering of the entire training set.

Curriculum factory, an original feature of our proposal, is a system that generates ordering of samples of a selected batch during training. It comprises a syllabus proposal submodule (*sp-module*) that takes a batch of randomly ordered samples as input and returns an ordering of the samples that serves as a syllabus. Given $\boldsymbol{T_s}$ we denote the j^{th} mini-batch containing $\boldsymbol{M} \ll \boldsymbol{N}$ samples by $\boldsymbol{B_j} = \{s_1, s_2, \ldots, s_M\}$. A training syllabus, $\boldsymbol{Syllabus_{B_j}}$ is defined as an ordering of every sample, $\boldsymbol{s_k} = (\boldsymbol{x_i}, \boldsymbol{y_i}) \in \boldsymbol{B_j}$ for$1 \leq \boldsymbol{i} \leq \boldsymbol{M}$, in ascending (*asc*) or descending (*dec*) order. Ordering is determined by the rank, $\boldsymbol{\varepsilon}$, of each sample as measured by a metric m taken from Table 1. Formally, $\boldsymbol{Syllabus_{B_j}} = (\boldsymbol{s'}_1, \boldsymbol{s'}_2, \ldots, \boldsymbol{s'_M})$ is a computationally found ordered set such that $\boldsymbol{\varepsilon_{s'_1}} \leq \boldsymbol{\varepsilon_{s'_1}} \leq \ldots \leq \boldsymbol{\varepsilon_{s'_M}}$ if ordering is *asc* and $\boldsymbol{\varepsilon_{s'_1}} \geq \boldsymbol{\varepsilon_{s'_1}} \geq \ldots \geq \boldsymbol{\varepsilon_{s'_M}}$ if *dec*.

The *sp-module* is built using a three-step processing depicted in Fig. 2. In stage I, all samples of a batch are assessed and ranked using a prespecified metric $\boldsymbol{m}$ (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 with 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 1 (Algorithm).

3.1 Stage I: Assessing Content of Training Samples

CNNs learn patterns of features from training and use layer-wise superposition of these 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. 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, then it is considered a distance measure. Otherwise, the measure is standalone and takes a single sample as input and returns a value that describes certain characteristics of the sample.

In order 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)* 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.

Entropy. Let $\boldsymbol{X}$ be a discrete random variable with alphabet $\boldsymbol{\chi}$ and a probability distribution function $\boldsymbol{p}(\boldsymbol{x}), \boldsymbol{x} \in \boldsymbol{\chi}$. The Shannon entropy [12] of $\boldsymbol{\chi}$ is defined as

$$\boldsymbol{H}(\boldsymbol{X}) = \sum_{\boldsymbol{x} \in \boldsymbol{\chi}} \boldsymbol{p}(\boldsymbol{x}) \mathbf{log} \frac{1}{\boldsymbol{p}(\boldsymbol{x})} \tag{2}$$

where $0\mathbf{log}\infty = 0$ and the base of the logarithm determines the unit, e.g. if base 2 the measure is in *bits* [13]. The term $-\boldsymbol{log}\ \boldsymbol{p}(\boldsymbol{x})$ can be viewed as the amount of information gained by observing the outcome $\boldsymbol{p}(\boldsymbol{x})$. 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 that is random [14]. 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., $\boldsymbol{p}(\boldsymbol{i}) = \boldsymbol{b_s}(\boldsymbol{i})/\boldsymbol{N}$, where $\boldsymbol{b_s}(\boldsymbol{i})$ denotes the histogram entry of intensity value $\boldsymbol{i}$ in $\boldsymbol{s}$, and $\boldsymbol{N}$ is the total number of pixels of $\boldsymbol{s}$. With this representation the entropy of an image $\boldsymbol{s}$ can be computed as:

$$\boldsymbol{E}(\boldsymbol{s}) = \sum_{\boldsymbol{i} \in \boldsymbol{\chi}, \boldsymbol{s} \in \boldsymbol{T_s}} \boldsymbol{b_s}(\boldsymbol{i}) \mathbf{log} \frac{\boldsymbol{N}}{\boldsymbol{b_s}(\boldsymbol{i})}, \tag{3}$$

where $\boldsymbol{T_s}$ (Eq. 1) is the training set and $\boldsymbol{\chi}(\boldsymbol{s})$ 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), kl-divergence (KL), mutual information (MI), information variation (IV), conditional entropy (CE) and their variants such as normalized mutual information (MIN).* A complete list of the metrics used for this study are listed in Table 2. Readers are encouraged to refer to [14–16] for detailed treatment of these metrics and others.

Statistical metrics on the other had measure the similarity (dissimilarity) of samples and typically use statistical measurements such as mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$. One such measure is the Structural Similarity Index (SSIM). SSIM is often used for predicting image quality using a reference image. Given two samples $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ the SSIM index [16] is given by:

$$\boldsymbol{SSIM}(\boldsymbol{s}_1, \boldsymbol{s}_2) = \frac{(2\mu_{\boldsymbol{s}_1}\boldsymbol{\mu}_{\boldsymbol{s}_2} + \boldsymbol{C}_1)(2\boldsymbol{\sigma}_{\boldsymbol{s}_1\boldsymbol{s}_2} + \boldsymbol{C}_2)}{(\mu_{\boldsymbol{s}_1}^2 + \mu_{\boldsymbol{s}_2}^2 + \boldsymbol{C}_1)(\boldsymbol{\sigma}_{\boldsymbol{s}_1}^2 + \boldsymbol{\sigma}_{\boldsymbol{s}_2}^2 + \boldsymbol{C}_2)} \tag{4}$$

Table 1. (Algorithm 1) Curriculum training of a CNN network η. Here, at least two m values, a primary measure and backup measures, from Table 2 are pre-specified. If no $\boldsymbol{m}$ *is* prespecified, *sp-module* picks a primary and backup measures randomly from the set of measures listed in Table 2.

Input: $\boldsymbol{Training\ set\ T_s, Metric\ m, Order\ o}$, β, π
Outputs: ***Trained model*** η (network weights)

1. *Initialize* iteration (training batch) counter *iter and* $\boldsymbol{\beta_{\eta \to S}}$ *to* $\boldsymbol{0}$ *and* $\boldsymbol{f_S}$ with *continue*
2. *Draw* training batch $\boldsymbol{B}$ of size $\boldsymbol{M}$ from $\boldsymbol{T_s}$
   For $\boldsymbol{i = 0\ to < M - 1}$
   1. ***Select input*** $\boldsymbol{s_i}$
   2. ***Compute rank*** $\boldsymbol{\varepsilon_{s_i}}$

   End
   Syllabus = *Sorted batch according to* $\boldsymbol{\varepsilon}$
3. ***Train network on B using Syllabus and increment*** *iter*
4. ***If*** *iter* **is equal to** π
   Calculate syllabus-to-baseline loss ratio ? and set fitness signal $\boldsymbol{f_S}$
   If $\boldsymbol{f_S}$ *is* set to *continue*
   go-to step *2*
   If $\boldsymbol{f_S}$ *is* set to *replace*
   go-to step 1
   Otherwise Halt training

where the terms μ and $\boldsymbol{\sigma}$ are the mean and variances of the two vectors and $\boldsymbol{\sigma_{s_1 s_2}}$ is the covariance of $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$. The constant terms $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ are used to avoid a null denominator. We also use simple measures such as $\boldsymbol{L_1}$ norm to compare the pixel histograms of two samples.

3.2 Stage II: Sorting Batches of Samples

A batch of training samples $\boldsymbol{B} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_\mathbf{M}\} \subset \boldsymbol{T_s}$ is selected from the training set. Each sample $\boldsymbol{s_k} \in \boldsymbol{B}$ is assigned a rank by analysing its pixel distribution using the specified metric $\boldsymbol{m}$. We use two types of metrics; *distance* and *standalone.* If $\boldsymbol{m}$ is a distance metric, a reference sample $\mathbf{s}_r \in \boldsymbol{B}$ is used to rank a moving sample $\mathbf{s}_m \in \boldsymbol{B}$. Initially, the reference sample is chosen at random. For instance, consider the following setup: let $\boldsymbol{m}$ be the mutual information (***MI***) measure, the algorithm first selects an initial reference sample, $\mathbf{s}_r = \mathbf{s}_1$ and computes the MI-index or rank ($\boldsymbol{\varepsilon}$) of every other sample, $\mathbf{s}_2, \ldots, \mathbf{s}_\mathbf{M}$, in the batch against $\mathbf{s}_r$. If *asc* ordering is used, the sample with the smallest $\boldsymbol{\varepsilon}$ 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, $\boldsymbol{S_B}$, is an ordering of the samples according to their mutual information index. Given a proposed syllabus $\boldsymbol{S_B} = \left\{s'_1, s'_5, s'_{2,}, s'_8 \ldots, s'_M\right\}$, the network first sees the initial reference sample, then

the sample having the smallest $\boldsymbol{\varepsilon}$ 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 $\boldsymbol{\varepsilon}$ the closer the two samples are. When using a *standalone* metric, each sample is ranked. The entire batch is then sorted based on the specified ordering and the rank of each sample. $\boldsymbol{m}$ is pre-specified as a learning parameter and can be updated during training if corresponding syllabus is deemed unfit. We experimented with several metrics to observe their impact on training.

Table 2. List of measures used in this study

Metric	Category	Implementation - given samples $s_1, s_2 \in \boldsymbol{T_s}$ where $\boldsymbol{b_s}$ is normalized histogram of pixel intensities and i is an index of a pixel value in a sample
Entropy	Standalone	Section 3.1
Joint Entropy (JE)	Distance	$\boldsymbol{JE}(s_1, s_2) = \sum_i \boldsymbol{b_s}(i)\log \boldsymbol{b_s}(i)$,
Mutual Information (MI)	Distance	$\boldsymbol{MI}(s_1, s_1) = \boldsymbol{E}(s_1) + \boldsymbol{E}(s_1) - \boldsymbol{JE}(s_1, s_2)$
KL-Divergence	Distance	$\boldsymbol{D_{k\|L}}(s_1, s_2) = \sum_i s_{1i} \log \frac{\boldsymbol{b}_{s1}(i)}{\boldsymbol{b}_{s2}(i)}$
Information Variation (IV)	Distance	$\boldsymbol{IV}(s_1, s_2) = \boldsymbol{E}(s_1) + \boldsymbol{E}(s_2) - 2\boldsymbol{MI}(s_1, s_1)$
Conditional Entropy (CE)	Distance	$\boldsymbol{CE}(s_1\|s_2) = \boldsymbol{E}(s_1, s_2) - \boldsymbol{E}(s_1)$, where $\boldsymbol{E}(s_1, s_2)$ is the sum entropies of s_1 and s_2
L1 Norm (L1)	Distance	$\boldsymbol{L}_1(s_1, s_2) = \|\|s_1 - s_2\|\| = \sum_{i=1} \|s_{1i} - s_{2i}\|$
L2 Norm (L2)	Distance	$\boldsymbol{L}_2(s_1, s_2) = \|\|s_1 - s_2\|\|_2 = \sqrt{\sum_{i=1}(s_{1i} - s_{2i})^2}$
Max Norm (MN)	Distance	This is like L1norm where, instead of every entry, the maximum entries' magnitude is used to calculate the norm [17]
Peak-signal-to-noise ratio (PSNR)	Standalone	$\boldsymbol{PSNR} = 20\log_{10}(\frac{\boldsymbol{MAX}}{\sqrt{\boldsymbol{MSE}(S_1, S_2)}}$ where, $\boldsymbol{MSE}(a, b) = \frac{1}{N^2}\sum_i^N \sum_j^N (a_{ij} - b_{ij})^2$
Structural Similarity index (SSIM)	Distance	Section 3.1

3.3 Stage III: Syllabus 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 η and training set T_s is determined using two configurable hyperparameters; number of iterations (can also be number of batches) π and the baseline performance β of the network on T_s averaged over π. β is the threshold by which the syllabuses' s fitness is determined and is chosen to be the average baseline loss of the network over π number of iterations. Baseline performance of a network is the network's training performance without curriculum.

Syllabus Fitness Criteria. Once η is trained on T_s for π number of iterations using a syllabus $Syllabus$, the losses are aggregated and the average loss,

$$\beta_{\eta \rightarrow Syllabus} = \frac{\sum_{i=0}^{\pi} loss(i)}{\pi}, \tag{5}$$

where $loss(i)$ is the i^{th} iteration training loss, of the network associated with S is computed. The syllabus-to-baseline loss ratio, $\omega = \beta_{\eta \rightarrow Syllabus}/\beta$, is then used as the sole criteria 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,* 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* is propagated, the curriculum factory 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.

4 Experiments

4.1 Datasets

Our method is implemented with the TensorFlow library [18] and training was done using a system equipped with four NVIDIA Tesla P100 GPUs. We present training and classification results obtained by training state-of-the-art image classification networks using different curriculum strategies described in Sect. 3 on CIFAR10, CIFAR100 [19] and ILSVRC-2015 ImageNet [20] datasets. Specifically, given the moderate cost of time associated with training a network using CIFAR10, CIFAR10 was used to perform an in-depth study of the proposed method using several network architectures. Based on the training trends on CIFAR10, we then perform repeatability study using CIFAR100. Finally, a syllabus that exhibits exceptional performance on those datasets is selected and compared with baseline performance on ImageNet.

4.2 Training

We trained several past and current state-of-the-art CNNs using open-source TensorFlow implementations. 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 optimizer with cross-entropy loss, 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 100 thousand iterations. We ensure all learning parameters and environment are identical, with varying networks and learning methods, to rule out other factors of influence.

4.3 Networks

GoogleNet (Inception) versions 1 [21] and 4 [22] were two of the networks evaluated in this study. Inception V1 was placed number 1 in the 2014 ImageNet [23] competition for classification and detection challenges. It's a 22 layered network that comprises of basic units referred to as *inception cells*. For each cell, a set of 1×1, 3×3 and 5×5 filters are learned to extract features at different scales from the input. The architecture also consists of two auxiliary classifiers that prevent the middle part of the network from dying out. The auxiliary losses are weighted and added to the Softmax loss as a way to combat vanishing gradient problem and provide regularization [21]. Inception V4 [22] evolved from Inception V1 and has a more uniformly simplified architecture and more inception cells. Additional network architectures include ResNet [24], VGG [25] which placed top in ILSVRC 2015 and ILSVRC 2014 respectively as well as several version of the MobileNet [26] architecture which were designed for computational efficiency.

To observe the training trends, we use curriculum settings with varying measure $\boldsymbol{m}$, $\boldsymbol{\pi} = 10000$, $\boldsymbol{o} = \boldsymbol{asc}$ and $\boldsymbol{\beta}$ value that is unique to each network and training set. To capture the classification results each network is trained for 500,000 iterations or until the learning curve is stable and the loss converges. Batch sizes of 8 and 16 were used for these experiments.

5 Results and Analysis

5.1 Training Trends

To capture the impact of the proposed approach on training, we trained all networks on CIFAR10 dataset. We use the total loss, which is the sum of cross-entropy and regularization losses as implemented in TensorFlow [27] framework as the primary evaluation criteria of the impact of the proposed method.

The training loss of select metrics and that of the baseline (blue) are depicted in Fig. 3. Clear trends can be observed from the plots. First, in all cases, curriculum-based training performs remarkably well in reducing training loss. For instance, when training GoogleNet using JE-based curriculum, it achieves a loss of 0.163 compared to 1.494 after training for only 100 K iterations. This is a loss reduction by a factor of 9. Similarly, MobileNet's loss is reduced by factor of 4. The second and most impressive

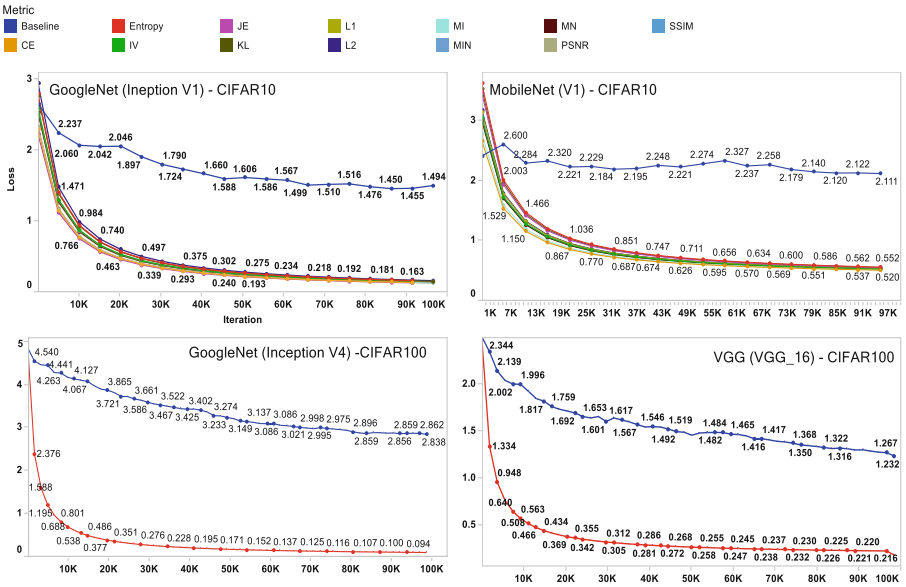

Fig. 3. Training loss of training MobileNet (top-right), GoogleNet V1 (top-left) on CIFAR10 using all metrics listed in Table 1, and GoogleNet V4 (bottom-left) and VGG (bottom-right) on CIFAR100 using the entropy measure. (Color figure online)

observation that highlights the effectiveness of the proposed technique is that the baseline performance is almost always inferior to any curriculum strategy throughout training. Only in few cases do we see a strategy produce higher loss. This is partly because our method removes variability due random shuffling using consistent input path with each batch as measured by the metric in use. Whilst these results are interesting, to ensure repeatability and confirm the efficacy of the best models, we performed additional experiments on CIFAR100. The bottom two plots of Fig. 3 show the training trends of GoogleNet and VGG on CIFAR100 using Entropy-based curriculum syllabus.

5.2 Regularization Loss

Regularization methods are supplementary techniques that are used to reduce test error and prevent overfitting. In neural networks and deep learning, the term is reserved solely for a penalty term in the loss function [28]. In all the networks used for this study, regularization is achieved by adding a regularize term R to the native loss function. Unlike the loss function, which expresses the deviation of the network output with labels, R is independent of the labels. Instead, it's used to encode *other properties* of the desired model to induce bias. These properties are rooted in assumptions other than consistency of network output with labels. The most common approaches are *weight decay* [29] and *smoothness* of the learned mapping. The regularization loss trends of training various networks on CIFAR10 and CIFAR100 using JE and IV curriculum strategies are depicted in Fig. 4.

5.3 Sparsity

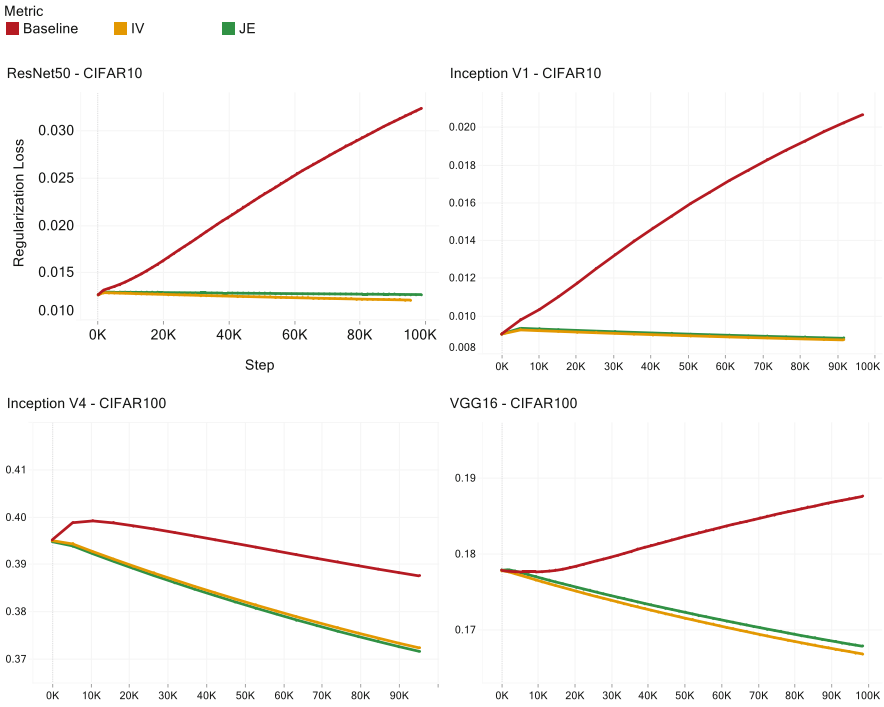

Fig. 4. Comparison of regularization loss of several curriculum strategies with the baseline on CIFAR10 (top) and CIFAR100 (bottom)

Output sparsity is in deep neural networks is highly desirable to reduce model training and inference costs. CNNs are highly sparse with many values involved in the calculations being zero or close to zero. Features map sparsity can be exploited to expedite training and reduce inference cost by skipping the values that are known to be zeros. Feature maps in CNN models usually have high sparsity. This is because convolution operation is immediately followed by an activation layer that turns all negative inputs into zeros. In addition, max-pooling layers only select a max value in a sub-region of the input and drop other values in the region. As shown in Fig. 5, we observe that the average feature-map and fully connected layer output sparsity scores are reduced much faster when training a network with the proposed method.

Fig. 5. Sparsity score of select layers of VGG16 (top) and Inception V1 (bottom)

5.4 Classification Results

Due to the lack of directly comparable techniques for curriculum-based training, we contrasted our techniques against the baseline generalization capabilities of a network on a given dataset. More specifically, we first trained the network to establish its baseline performance on the dataset. The same network is then trained using the proposed technique. The best performing weights are then chosen for performance comparison. The results on CIFAR10 and CIFAR100 datasets are presented below.

With most curriculum setups, the network's generalization capability is uncompromised and is within range of the baseline performance. However, depending on the dataset, several curriculum metrics produce networks that generalize better than the baseline. For instance, training GoogleNet (Table 3) using MI and IV based curriculum strategies perform better on CIFAR10, while IV and JE enabled syllabus performs better on CIFAR100 (Tables 4 and 5).

Table 3. Comparison of GoogleNet (Inception V1) network on CIFAR10 and CIFAR100. Reported is the test accuracy in percent of classification for network with and without the proposed training approach. Best results are highlighted in bold. The baseline performance is the first entry in the table.

Network	CIFAR10 (test acc. %)	CIFAR100 (test acc. %)	Curriculum
GoogleNet	0.528	0.433	
GoogleNet-MI	**0.615**	0.358	MI
GoogleNet-MIN	0.456	0.146	MIN
GoogleNet-IV	**0.671**	**0.456**	IV
GoogleNet-JE	0.586	**0.489**	JE

Table 4. Comparison of ResNet_V1_50 network on CIFAR10 and CIFAR100.

Network	CIFAR10 (test acc. %)	CIFAR100 (test acc. %)	Curriculum
VGG	0.922	0.645	
VGG-MI	**0.945**	0.512	MI
VGG-MIN	0.904	0.602	MIN
VGG-IV	0.897	**0.698**	IV
VGG-JE	**0.943**	0.631	JE

Table 5. Comparison of ResNet_16 network on CIFAR10 and CIFAR100.

Network	CIFAR10 (test acc. %)	CIFAR100 (test acc. %)	Curriculum
ResNet	0.954	0.791	
ResNet-MI	0.945	**0.842**	MI
ResNet-MIN	**0.973**	0.789	MIN
ResNet-IV	0.943	**0.849**	IV
ResNet-JE	0.924	**0.851**	JE

5.5 Conclusion

We have introduced a system for training CNNs using curriculum strategies. Our approach combines content measures taken from information theory with curriculum learning and alleviates the need to determine the presumed difficulty of training samples. Unlike previous works, we exploit information-theoretic and statistical relationship between training samples to propose a syllabus to guide training; we have shown that this improves training performance of CNNs. The results indicate that curriculum strategies reduce training loss faster without necessarily increasing the generalize performance compared to conventional training. Our intuition is that the proposed technique enables

faster convergence by discovering optimal path that take to local minima. However, further analysis is required to fully test and prove our *hypothesis that the proposed method combined with SGD optimization expedites a search for local minima by creating an optimal path in input space.*

References

1. Graves, A., Bellemare, M.G., Menick, J., Munos, R., Kavukcuoglu, K.: Automated curriculum learning for neural networks (2017). 10 p.
2. Kim, T.-H., Choi, J.: ScreenerNet: learning self-paced curriculum for deep neural networks. arXiv:1801.00904 Cs, January 2018
3. Hinton, G.E.: To recognize shapes, first learn to generate images. Prog. Brain Res. **165**, 535–547 (2007)
4. Zhou, H.-Y., Gao, B.-B., Wu, J.: Adaptive feeding: achieving fast and accurate detections by adaptively combining object detectors. arXiv:1707.06399 Cs, July 2017
5. Graves, A., Bellemare, M.G., Menick, J., Munos, R., Kavukcuoglu, K.: Automated curriculum learning for neural networks. arXiv:1704.03003 Cs, April 2017
6. Kumar, M.P., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, pp. 1189–1197. Curran Associates, Inc. (2010)
7. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv:1511.05952 Cs, November 2015
8. Loshchilov, I., Hutter, F.: Online batch selection for faster training of neural networks. arXiv:1511.06343 Cs Math, November 2015
9. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. arXiv:1703.04730 Cs Stat, March 2017
10. Ghebrechristos, H., Alaghband, G.: Expediting training using information theory-based patch ordering algorithm (2018). 6 p.
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
12. Shannon, C.E.: A mathematical theory of communication. Bell Syst. Tech. J. **27** (1948). 55 p.
13. Bonev, B.I.: Feature selection based on information theory (2010). 200 p.
14. Feixas, M., Bardera, A., Rigau, J., Xu, Q., Sbert, M.: Information theory tools for image processing. Synthesis Lectures on Computer Graphics and Animation, vol. 6, no. 1, pp. 1–164 (2014)
15. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley, Hoboken (2006). 774 p.
16. Horé, A., Ziou, D.: Image quality metrics: PSNR vs. SSIM, pp. 2366–2369 (2010)
17. Deming, W.E., Morgan, S.L.: The Elements of Statistical Learning. Elsevier, Amsterdam (1993)
18. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. arXiv:1605.08695 Cs, May 2016
19. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009). 60 p.
20. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. arXiv:1409.0575 Cs, September 2014
21. Szegedy, C., et al.: Going deeper with convolutions. arXiv:1409.4842 Cs, September 2014
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. arXiv:1512.00567 Cs, December 2015
23. ImageNet Large Scale Visual Recognition Competition (ILSVRC). http://image-net.org/challenges/LSVRC/. Accessed 29 Apr 2017

24. 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, pp. 770–778 (2016)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:14091556 Cs, September 2014
26. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:170404861 Cs, April 2017
27. TensorFlow: TensorFlow. https://www.tensorflow.org/. Accessed 14 Mar 2019
28. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995). 498 p.
29. Lang, K.J., Hinton, G.E.: Dimensionality reduction and prior knowledge in E-set recognition (1990). 8 p.