# *Optimizing Training using Information Theory-Based Curriculum Learning Factory*

Henok Ghebrechristos
*Department of Engineering and Computer Science*
*University of Colorado-Denver*
Denver, USA
henok.ghebrechristos@ucdenver.edu

Gita Alaghband,PhD
*Department of Engineering and Computer Science*
*University of Colorado-Denver*
Denver, USA
gita.alaghband@ucdenver.edu

***Abstract*— We present a new system that can automatically generate input paths (syllabus) for a convolutional neural network to follow through a curriculum learning to improve training performance. Our system utilizes information-theoretic content measures of training samples to form syllabus at training time. We treat every sample as 2D random variable where a data point contained in the sample (such as a pixel) is modelled as an independent and identically distributed random variable (i.i.d) realization. We use several information theory methods to rank and determine when a sample is fed to a network by measuring its pixel composition and its relationship to other samples in the training set. Comparative evaluation of multiple state-of-the-art networks, including, 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 reduce training loss by as much as a factor of 9 compared to conventional training.**

***Keywords—Deep Learning, Curriculum Learning, Convolutional Neural Network. Information Theory, Curriculum Factory***

## I. Introduction

When used in supervised machine learning, curriculum-based training (Figure 1) exposes samples to the learning system in 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 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.

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 (Figure 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 of 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.

To summarize, this paper makes the following 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

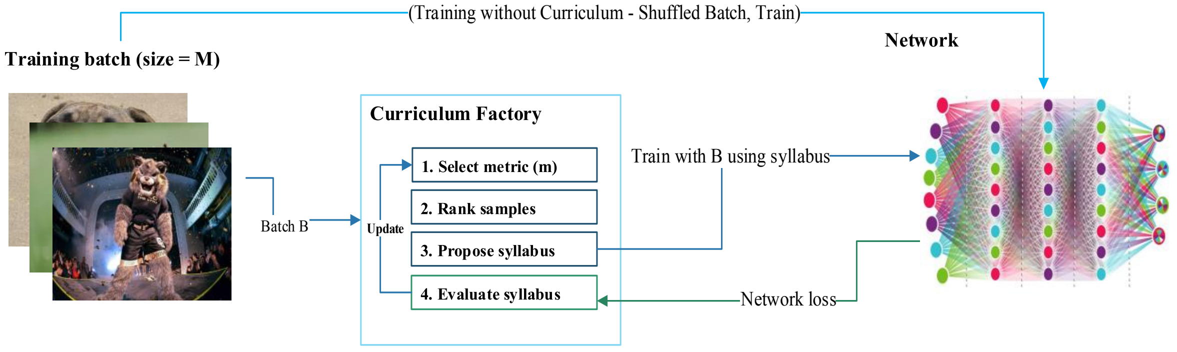

**Fig.** 1. Curriculum Learning with and without the proposed system means. As the first contribution, we present a curriculum learning algorithm that reduces training loss at each iteration by ordering batches to form a syllabus. When used in stochastic gradient descent (SGD)-based training, our algorithm 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.

## II. Proposed Method

The goal of training CNNs is to try and determine optimal network weights to approximate target mapping $\boldsymbol{g}: \boldsymbol{x_i} \rightarrow \boldsymbol{y_i}$ [3] where $\boldsymbol{x_i}$ is the input vector and $\boldsymbol{y_i}$ is the corresponding desired output taken from a training set,

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

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.

TABLE I. Curriculum Training of a CNN network. Here, at least two *M* values, a primary measure and backup measures, from Table 2 are pre-specified. If no *M is* prespecified, *SP-module* picks a primary and backup measures randomly from the set of measures listed in Table 2.

```
Input: Training set $T_s$, Metric $m$, Order $o$, $\beta$, $\pi$
Outputs: Trained model $\eta$ (network weights)
1. Initialize iteration (training batch) counter iter and $\beta_{\eta\rightarrow s}$ to 0 and $f_S$
   with continue
2. Draw training batch $B$ of size $M$ from $T_s$
      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 and increment iter
4. If iter is equal to $\pi$
      Calculate syllabus-to-baseline loss ratio $\boldsymbol{\omega}$ and set fitness signal $f_S$
         If $f_S$ is set to continue, go-to step 2
         If $f_S$ is set to replace, go-to step 1
         Otherwise Halt training
```

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* (*curriculum factory*) 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 $\boldsymbol{B_j} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_M}\}$. 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 $\boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{M}$, 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, $\boldsymbol{Syllabus_{B_j}} = (\boldsymbol{s'_1}, \boldsymbol{s'_2}, \dots, \boldsymbol{s'_M})$ is a computationally found ordered set such that $\boldsymbol{\varepsilon_{s'_1}} \leq \boldsymbol{\varepsilon_{s'_1}} \leq \dots \leq \boldsymbol{\varepsilon_{s'_M}}$ if ordering is *asc* and $\boldsymbol{\varepsilon_{s'_1}} \geq \boldsymbol{\varepsilon_{s'_1}} \geq \dots \geq \boldsymbol{\varepsilon_{s'_M}}$ if *dec*.

*Curriculum factory* 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 within an end-to-end training pipeline is presented in Table (Algorithm) 1.

### A. 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.

#### 1) Information Theoretic Measures

In order to use information-theoretic measures, we model all images 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

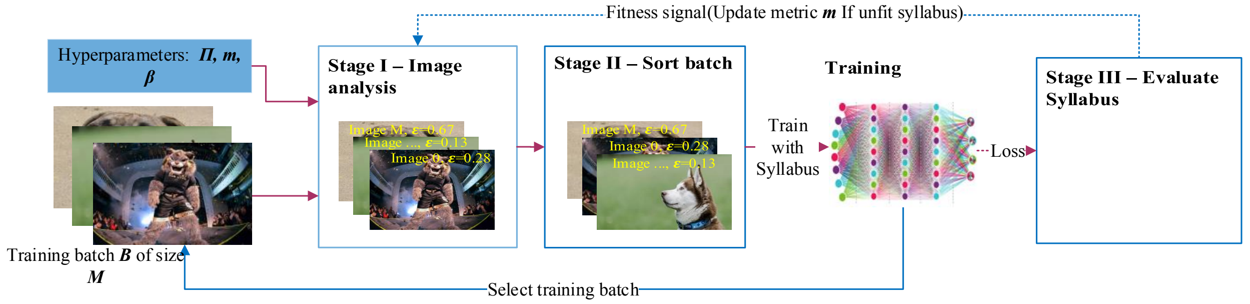

**Fig. 2.** Processing stages within *sp-module*. From left to right given batch B and hyperparameters read once at start: I. Rank each sample. II. Generate syllabus by ordering B according to rank of each sample which is then used to train a network. III. Evaluate syllabus using network loss.

training samples. Below we discuss few information theoretic measures as well as statistical measures such as SSIM. A complete list is presented in Table IIs.

TABLE II. LIST OF MEASURES USED IN THIS STUDY

| **Metric** | **Category** | **Implementation - given samples $s_1, s_2 \in T_s$ where $b_s$ is normalized histogram of pixel intensities and *i* is an index of a pixel value in a sample.** |
|---|---|---|
| Entropy | standalone | Eqns. 2 and 3 |
| Joint Entropy (JE) | distance | $JE(s_1, s_2) = \sum_i b_s(i) \log b_s(i),$ |
| Mutual Information (MI) | distance | $MI(s_1, s_1) = E(s_1) + E(s_1) - JE(s_1, s_2)$ |
| KL-Divergence | distance | $D_{k\|\|L}(s_1, s_2) = \sum_i s_{1_i} \log \frac{b_{s_1}(i)}{b_{s_2}(i)}$ |
| Information Variation (IV) | distance | $IV(s_1, s_2) = E(s_1) + E(s_2) - 2MI(s_1, s_1)$ |
| Conditional Entropy (CE) | distance | $CE(s_1 \| s_2) = E(s_1, s_2) - E(s_1)$, where $E(s_1, s_2)$ is the sum entropies of $s_1$and $s_2$ |
| L1 Norm (L1) | distance | $L_1(s_1, s_2) = \|\|s_1 - s_2\|\| = \sum_{i=1} \|s_{1_i} - s_{2_i}\|$ |
| L2 Norm (L2) | distance | $L_2(s_1, s_2) = \|\|s_1 - s_2\|\|_2 = \sqrt{\sum_{i=1} (s_{1_i} - s_{2_i})^2}$ |
| Max Norm (MN) | distance | This is like L1norm where, instead of every entry, the maximum entries' magnitude is used to calculate the norm [10] |
| Peak-signal-to-noise ratio (PSNR) | standalone | $PSNR = 20 \log_{10}(\frac{MAX}{\sqrt{MSE(S_1, S_2)}}$ where, $MSE(a, b) = \frac{1}{N^2} \sum_i^N \sum_j^N (a_{ij} - b_{ij})^2$ |
| Structural Similarity index (SSIM) | distance | Eqn. 7 |

*a) Entropy*

Let $\boldsymbol{X}$ be a discrete random variable with alphabet $\boldsymbol{\chi}$ and a probability distribution function $\boldsymbol{p(x), x \in \chi}$. The Shannon entropy [4] of $\chi$ is defined as

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

where $\mathbf{0 \log \infty = 0}$ and the base of the logarithm determines the unit, e.g. if base 2 the measure is in *bits* [5]. The term $-\boldsymbol{\log p(x)}$ can be viewed as the amount of information gained by observing the outcome $\boldsymbol{p(x)}$. 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 that is random [6]. 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., $\boldsymbol{p(i) = b_s(i)/N}$, where $\boldsymbol{b_s(i)}$ 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:

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

where $\boldsymbol{T_s}$ (eqn. 1) is the training set and $\boldsymbol{\chi(s)}$ 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)*, *kl-divergence (KL)*, *mutual information (MI), information variation (IV), conditional entropy (CE) and their variants such as normalized mutual information (MIN)*. A complete list and implementation of the metrics are listed in Table 2. Readers are encouraged to refer to [6]–[8] for detailed treatment of these metrics and others.

*b) Joint Entropy*

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

$$JE(Y, X) = -\sum_x \sum_y p(x, y) \log p(x, y). \qquad (4)$$

When we model images as random variables, the joint entropy is computed by gathering joint histogram between the two images. For two training samples, $S_1, S_2 \in T_s$ the joint entropy is given by:

$$JE(S_1, S_2) = \sum_i b_s(i) \log b_s(i), \qquad (5)$$

where $b_s(i)$ is the $i^{th}$ value of joint histogram of the two samples.

*c) Mutual Information*

Mutual information (MI) is the measure of the statistical dependency between two or more random variables [16]. The mutual information of two samples $S_1$ and $S_2$ can be defined in terms of the individual entropies of both $S_1$ and $S_2$ and the joint entropy of the two variables $JE(S_1, S_2)$.

$$MI(S_1, S_2) = E(S_1) + E(S_2) - JE(S_1, S_2) \qquad (6)$$

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 [9].

*2) Statistical Measures*

Statistical metrics on the other had measure the similarity (dissimilarity) of samples and typically use statistical measurements such as mean μ and standard deviation $\sigma$.

*a) 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 [8] is given by:

$$SSIM(s_1, s_2) = \frac{(2\mu_{s_1}\mu_{s_2} + C_1)(2\sigma_{s_1 s_2} + C_2)}{(\mu_{s_1}{}^2 + \mu_{s_2}{}^2 + C_1)(\sigma_{s_1}{}^2 + \sigma_{s_2}{}^2 + C_2)} \qquad (7)$$

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.

### B. Stage II: Sorting Batches

A batch of training samples $\boldsymbol{B} = \{\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_M}\} \subset \boldsymbol{T_s}$ 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, $\mathbf{s_2}, \ldots, \mathbf{s_M}$, 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 analysed 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 $\boldsymbol{S_B} = \{\boldsymbol{s'_1}, \boldsymbol{s'_5}, \boldsymbol{s'_2}, \boldsymbol{s'_8} \ldots, \boldsymbol{s'_M}\}$, 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 independently. 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.

### C. 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 $\boldsymbol{\eta}$ and training set $\boldsymbol{T_s}$ 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 $\boldsymbol{T_s}$ averaged over $\boldsymbol{\pi}$. $\boldsymbol{\beta}$ is the threshold by which the syllabuses' 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 $\boldsymbol{\eta}$ is trained on $\boldsymbol{T_s}$ for $\boldsymbol{\pi}$ number of iterations using a syllabus $\boldsymbol{Syllabus}$, the losses are aggregated and the average loss,

$$\boldsymbol{\beta}_{\boldsymbol{\eta} \to \boldsymbol{Syllabus}} = \frac{\sum_{i=0}^{\pi} \boldsymbol{loss(i)}}{\boldsymbol{\pi}}, \tag{10}$$

where ***loss(i)*** is the $\boldsymbol{i^{th}}$ iteration training loss, of the network associated with $\boldsymbol{S}$ is computed. The syllabus-to-baseline loss ratio, $\boldsymbol{\omega} = \boldsymbol{\beta}_{\boldsymbol{\eta} \to \boldsymbol{Syllabus}} / \boldsymbol{\beta}$, is then used as the sole criteria to determine the fitness of the syllabus. Depending on the value of $\boldsymbol{\omega}$, a fitness signal $\boldsymbol{f_S}$, 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 $\boldsymbol{f_S}$ is set to *continue.* Otherwise $\boldsymbol{f_S}$ 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.

## III. Experiments s

To observe the training trends, we use curriculum settings with varying measure $\boldsymbol{m}$, $\boldsymbol{\pi}$ = 10000, $\boldsymbol{o}$ = *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.

### A. Training

We present training and classification results obtained by training state-of-the-art image classification networks using different curriculum strategies on CIFAR10 [11], CIFAR100 [11] and ILSVRC-2015 ImageNet [12] 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. The networks include GoogleNet (Inception) versions 1 [13] and 4 [14], and VGG [15].

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.

## IV. Results and Analysis

### A. Training Trends

We use the total loss, which is the sum of *cross-entropy* and *regularization* losses as implemented in TensorFlow [16] framework as the primary evaluation criteria of the impact of the proposed method. The training loss of a 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 100K iterations. This is a loss reduction by a factor of 9 compared to baselines. Similarly, MobileNet's loss is reduced by factor of 4. The second and most impressive observation that highlights the effectiveness of the proposed

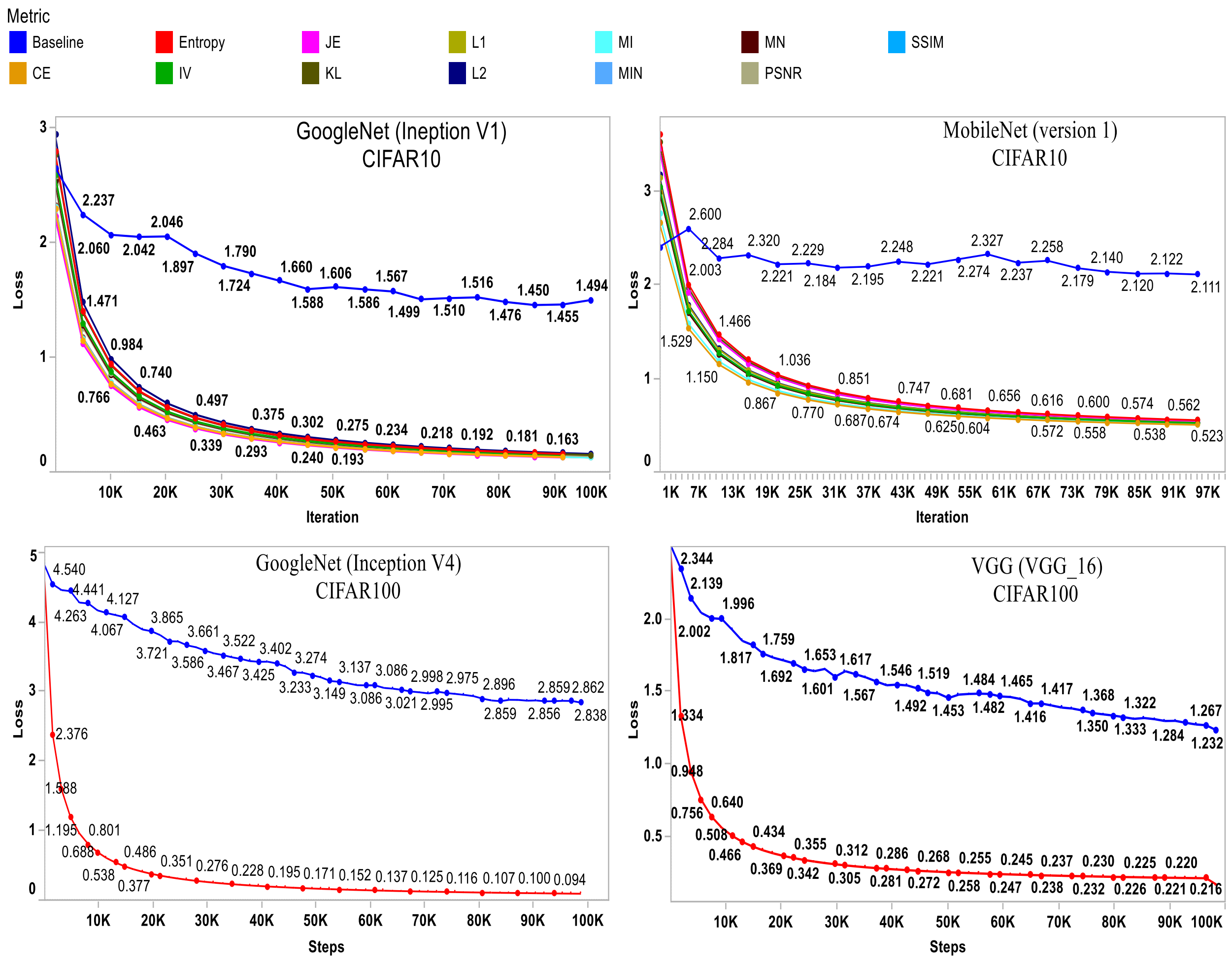

**Fig. 3.** Training loss of training MobileNet (top-right), GoogleNet V1 (top-left) on CIFAR10 using all metrics listed in Table 1, and GoogleNet V4 (bottom-left) and VGG (bottom-right) on CIFAR100 using the Entropy measure – which comparatively had better results on CIFAR10.

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. We argue this is partly because our method removes variability of sample characteristics due to random shuffling by enforcing 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 Figure 3 show the training trends of GoogleNet and VGG on CIFAR100 using Entropy-based curriculum syllabus.

## B. *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.

TABLE III. 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 IV. COMPARISON OF RESNET_V1_50 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 |

TABLE V. COMPARISON OF RESNET_16 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 |
| ssVGG-MIN | 0.904 | 0.602 | MIN |
| VGG-IV | 0.897 | **0.698** | IV |
| VGG-JE | **0.943** | 0.631 | JE |

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-based syllabus perform better on CIFAR100.

### C. *Experiments on ImageNet-1K*

Here, we conduct training experiments on the 1000-class ImageNet Classification task [17] using JE based curriculum training and the VGG architecture. Since training the networks on this dataset can take several weeks of computation on our system, we did not aim for best performance, but rather performed a proof of concept experiment to see if our techniques exhibit similar behaviour on larger datasets. We trained VGG_A with and without JE-based curriculum for 100,000 iterations with a batch size of 16 samples. The training trends are depicted in Figure 4. Although the drops in loss are not as dramatic as those in CIFAR10 or CIFAR100, one can clearly observe JE-based curriculum training has favourable trend compare to baseline. At this stage, Top-1 validation accuracy of the curriculum trained model, when evaluating on the centre 224x224 patch of the validation samples, is at 26.8% which is in par with 32.3% TOP-1 accuracy of the baseline model. The training trends are clearly reflective of the impact of proposed technique, which corroborate the trends observed with the smaller datasets.

## V. 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.*

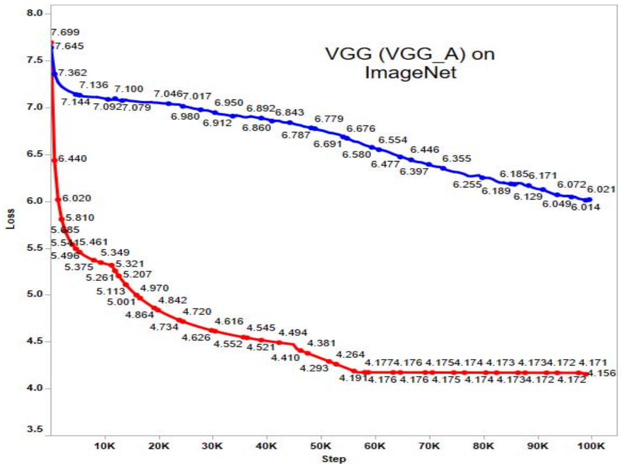

**Fig. 4**. Training loss of training VGG (VGG_A) on ILSVRC-2015 ImageNet dataset, with (red) and without (blue) JE-based curriculum syllabus.

## VI. REFERENCES

[1] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated Curriculum Learning for Neural Networks," p. 10.

[2] T.-H. Kim and J. Choi, "ScreenerNet: Learning Self-Paced Curriculum for Deep Neural Networks," *ArXiv180100904 Cs*, Jan. 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[4] C. E. Shannon, "A Mathematical Theory of Communication," p. 55, 1948.

[5] B. I. Bonev, "Feature Selection Based on Information Theory," p. 200, 2010.

[6] M. Feixas, A. Bardera, J. Rigau, Q. Xu, and M. Sbert, "Information theory tools for image processing," *Synth. Lect. Comput. Graph. Animat.*, vol. 6, no. 1, pp. 1–164, 2014.

[7] T. M. Cover and J. A. Thomas, "Elements of Information Theory," p. 774, 2006.

[8] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," 2010, pp. 2366–2369.

[9] D. B. Russakoff, C. Tomasi, T. Rohlfing, C. R. Maurer, and Jr., "Image Similarity Using Mutual Information of Torsten Rohlfing," in *8th European Conference on Computer Vision (ECCV*, 2004, pp. 596–607.

[10] W. E. Deming and S. L. Morgan, *The Elements of Statistical Learning*. Elsevier, 1993.

[11] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," p. 60.

[12] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *ArXiv14090575 Cs*, Sep. 2014.

[13] C. Szegedy *et al.*, "Going Deeper with Convolutions," *ArXiv14094842 Cs*, Sep. 2014.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *ArXiv151200567 Cs*, Dec. 2015.

[15] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ArXiv14091556 Cs*, Sep. 2014.

[16] "TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/. [Accessed: 14-Mar-2019].

[17] "ImageNet Large Scale Visual Recognition Competition (ILSVRC)." [Online]. Available: http://image-net.org/challenges/LSVRC/. [Accessed: 29-Apr-2017].