Expediting Training Using Information Theory-Based Patch Ordering Algorithm

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 framework for automatically ordering image patches that enables in-depth analysis of dataset relationship to learnability of a classification task using convolutional neural network. Our preliminary experimental results show that an informed smart shuffling of patches at a sample level can expedite training by exposing important features at early stages of training. Using multiple network architectures and datasets, we show that ordering image regions using mutual information measure between adjacent patches, enables CNNs to converge in a third of the total steps required to train the same network without patch ordering.**

Keywords—Convolutional neural network, deep learning, entropy, information theory, patch ordering

I. Introduction

An emerging promising theoretical characterization of deep learning that supports an intuition that motivated this work is the characterization that uses an information theoretic view of feature extraction [1]. The authors propose to study deep learning through the lens of information theory using the IB principle. In this characterization, deep learning is modeled as a representation learning. Each layer of a deep neural network can be seen as a set of summary statistics which contain some of the information present in the training set, while retaining as much information about the target output as possible [2]. One relevant insight presented in these papers is that the goal of DL is to capture and efficiently represent the relevant information in the input variable that describe the output variable. This is equivalent to the IB method whose goal is to find maximally compressed mapping of the input while preserving as much relevant information of the output as possible. This characterization leads us to ask the question: *Can we utilize information theoretic techniques for images to make training efficient? Particularly, can we preprocess training set and feature maps such that the relevant information is captured in the early stages of training?*

A training set for image classification tasks that employ supervised learning is constructed with the help of human labeler. For instance, for a cat vs dog classification problem, the human labeler must categorize each sample into either one of the classes. During this process, the labeler must recognize and classify each input using their own experience and distinguishing capabilities. Considering this, a natural question we first must answer before addressing the question above is: *Does human classification performance on the training dataset affect learnability of the task?* In other words, can the networks learn from 'scrambled' samples that cannot be classified by the naked eye? This question was investigated in [3] with intriguing outcomes. The authors present results that indicate that CNNs are capable of easily fitting training set containing samples that have no correlation with labels (see Fig. 3 for illustration). These results have us reconsider the traditional view that networks build hierarchy of features in increasing abstraction, i.e., *learn combination pixels that make edges in the lower layers, learn combinations of edges that make up object parts in the middle layers, learn combinations of parts that make up an object the next layer* etc. This view is challenged by the findings in this paper (see section V for detail). We use the information theoretic characterization of deep learning to shed light on the questions by developing preprocessing and learning techniques that reduce convergence time by improving features extraction from images using multilayered CNNs. We first rule out that human recognizable features matching labels are not necessary for CNNs and that they are able to fit training set containing scrambled samples with minimal impact on generalization. Equipped with this result we then utilize similarity and information theoretic measures of image characteristics to preprocess and ease feature extraction from images during training. Our methods aim to expose important features of each training sample earlier in training by reorganizing image regions. The contributions of our approach are:

1. We provide a framework and algorithms for preprocessing dataset to reorder image patches using techniques that minimize mutual entropy of adjacent image patches of each training sample. As the results demonstrate, organizing patches, of each training sample using measures such as entropy of a patch and mutual information index between patches enable faster convergence.
2. We present several techniques for ranking samples that use information theoretic measures of the relationship between adjacent patches and present results that show faster convergence compared to standard training.

Two benchmark datasets and Inception [4], VGG [5] and ResNet [6] architectures, known for achieving exceptional results on image classification tasks, are used for evaluation. The networks are first trained on the corresponding datasets to create baseline reference performance metrics for comparison. For each network we used Adams optimization technique with cross-entropy loss to gather emperical training, validation and test data.

The remaining content is presented as follows. In section 2, we present the patch ordering approach and highlight the

978-1-7281-1360-9/18/$31.00 ©2018 IEEE
DOI 10.1109/CSCI.2018.00223

design and implementation of algorithms used to preprocess data and feature maps based on patch ordering. In Section 3, we discuss the experimental setup. Then, section 4 presents analysis of our results obtained by training Inception using multiple unmodified and patch-ordered datasets. Finally, we conclude by offering our insight as to why the outcomes are important for deep learning and future generation networks.

II. METHOD

During training, CNNs construct hierarchy of feature representations and use superposition of the hierarchical features when generalizing to unseen input (Ian Goodfellow et al. 2006). However, we believe learnability of a classification task is closely related to the *amount of information contained in the dataset that enable distinguishability of one class from the others.* To further explore this claim, we developed techniques and conducted several experiments by preprocessing training set using various techniques. The techniques and the general procedure are described below.

A. Patch Ordering

Our intuition is that some ordering at a sample level can expedite training by exposing features that are important for separating the classes in the early stages of training. For illustration, consider the toy images in Fig. 1. If a person with knowledge of the number system, was asked to classify or label the two images, they can give several answers depending on their experiences. At first glance, they can label a) as 'large number 1' and b) as 'large number 2'. If they were asked to give more details, upon elaboration of the context, the labeler can quickly scan a) and realize that it is a picture of digits 0 through 9. Similarly, b) would be classified as such, but analyzing and classifying b) can cost more time because the labeler must ensure every digit is present (we encourage the readers to do the experiment). It's the time cost that is of interest to us in the context of learning systems. The mere ordering of the numbers enables the labeler to classify a) faster than b).

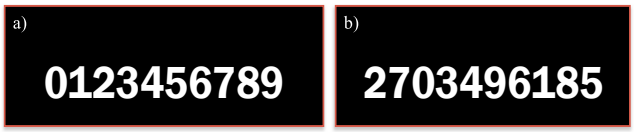

Figure 1: Toy images to illustrate the importance of ordering for classification. The two images in this context are of the same class. The label can be 'digits 0-9'

Given this intuition, we asked if ordering patches of training images such that the adjacent patches are 'closer' to each other by similarity measure, could expedite training and improve generalization. Based on the mental exercise, the procedure can intuitively be justified by the fact that toy sample a) is easier to classify because, as our eyes scan from left to right the features (0,1,2. . .) are captured in order. Whereas it might take several scans of b) to determine the same outcome. Convolution based feature extractors use a similar concept to capture features used to distinguish one class from the others. The features are extracted by scanning the input image using convolution filters. The output of convolution at each spatial location are then stacked to construct the feature map. Implementation of this operation in most deep learning frameworks maintain spatial locations of features which then can be obtained by deconvolution. In other words, there is a one-to-one mapping between the location of a feature in a feature map and its location on the original input (Fig.2.). Note that the feature map not only encodes the feature (ear or head) but it also implicitly encodes the location of the feature on the

Fig. 2: a) input image b) feature map after convolving input with filter.

input image (green arrow in Fig. 2.). *The encoding of location is required for detection and localization tasks but not for classification tasks.* Another question that arises from these observations is:

Can we control feature map construction such that the resulting feature map has characteristics that enables efficient learning while maintaining or improving generalization?

To answer this question, we searched for DL characterization that aligns with this intuition and found the work of [1] captures this intuition by relating DL training from images to the Information Bottleneck principle (discussed below). While the authors discuss IB in the context of the entire training set and end-to-end training of deep networks, our exploration is limited to individual training samples and aim to expose information that can be captured and presented to the network earlier during training. We developed techniques to reconstruct training images by breaking up the inputs into equal sized patches and reconstruct them using the concept of ordering (Fig.3). Information-theory-based and traditional measures of images were used for ranking and ordering. These measures can generally be divided into two:

1. **Standalone measures** –measure some characteristic of a patch. For example, the peak signal- to-noise ratio measure returns a ratio between maximum useful signal to the amount of noise present in a patch.
2. **Similarity measures** – these measures on the other hand, compare a pair of patches. The comparison measures can be measures of similarity or dissimilarity like L1-norm and structural similarity or information-theoretic-measures that compare distribution of pixel values such as joint entropy. The similarity measures discussed in the subsections below include Joint Entropy and Mutual Information.

Fig. 3. An illustration of patch ordering. a) Input image, b) reconstruction of the input using structural similarity of patches and c) feature map generated by convolving b). Note that the encoding of spatial location of a feature is not present in the feature map. The original image (a) is reconstructed using structural similarity measure. This reconstruction is performed prior to convolution at a preprocessing stage. Similar procedure can be applied to feature maps deep in the learning pipeline.

Below we summarize the measures and present the sorting and reconstruction algorithm

1) Entropy-Based Measures for Patch Ordering

The procedure for generating, ordering image patches and for reconstructing image from ordered patches is below.

Require: **MeasureType**, **PatchSize**

1. **Obtain** training batch $\boldsymbol{B}$ of size **BatchSize**

For $\boldsymbol{i = 1}$ **to BatchSize do:**

For each input image $\mathbf{x_i}$ $\boldsymbol{in}$ **B**, **do**:

a. $\boldsymbol{p_r = Generate - Patches(x_i)}$ *r: 0, …, number of patches in* $\boldsymbol{x_i}$

b. If MeasureType *is* standalone

i.$\boldsymbol{Compute - Individual - Index(p_r)}$

c. Otherwise

i.Select *a* reference patch $\boldsymbol{p_0}$

ii.$\boldsymbol{Compute - Individual - Index\ (p_r, p_0)}$

d. *Sort-Pachtes* in order according to **MeasureType** *and* indices

e. *Reconstruct-Sample(*$\boldsymbol{X_i'}$**)**

2. Train network on $\boldsymbol{B}$

3. Repeat

End

Return $\boldsymbol{Network}$

Table 1: Patch Ordering and Reconstruction (POR) Algorithm. The function **Generate-Patches** generates equal sized patches of the input image. **Compute-Individual-Index** calculates the index of a given patch when the **MeasureType** is of type standalone while **Compute-Mutual-Index** computes an index of similarity between two patches. **Sort-Pachtes** sorts the patches according to indices and **Reconstruct-Sample** constructs a sample using sorted patches. For computational efficiency $\boldsymbol{PatchSize}$ is taken from (4x4, 8x8, 16x16) and all samples are resized to 32x32 prior to preprocessing. Since the dataset consists of color (RGB) images the algorithm computes the index of each channels and returns the average .

a) Entropy

Information theory provides a theoretical foundation to quantify information content, or the uncertainty, of a random variable represented as a distribution [8], [9]. This can be extended to image processing and computer vision [10]. 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 [9]. Formally, let X be a discrete random variable with alphabet χ and a probability mass function $p(x), x \in \chi$. The Shannon entropy or information content of X is defined as

$$E(X) = \sum_{x \in \chi} p(x) \log \frac{1}{p(x)} \quad (1)$$

where $0 \log \infty = 0$ and the base of the logarithm determines the unit, e.g. if base 2 the measure is in *bits* etc. [11]. The term $log\ \frac{1}{p(x)}$ can be viewed as the amount of information gained by observing the outcome $p(x)$. This measure can be extended to analyze images as realizations of random variables [9]. A simple model would assume that each pixel is an independent and identically distributed random variable *(i.i.d)* realization [9]. When dealing with discrete images, we express all entropies with sums. One can obtain the probability distribution associated with each image by binning the pixel values into histograms. The normalized histogram can be used as an estimate of the underlying probability of pixel intensities, i.e., $p(i) = b_s(i)/N$, where $b_s(i)$ denotes the histogram entry of intensity value i in sample S and N is the total number of pixels of S. With this model the entropy of an image S can be computed using:

$$E(S) = \sum_{i \in \chi(s), s \in T_s} b_s(i) \log \frac{N}{b_s(i)}, \quad (2)$$

where $T_s = \{(x_n, y_n): 1 \leq n \leq N\}$ is the training set comprising both the input values x_n and corresponding desired output values y_n. N is the total number of examples in the training set. $\chi(s)$ represents the image as a vector of pixels. While individual entropy is the basic index used for ordering, we also consider strategies that relate two image patches. These measures include *joint entropy*[9], and *mutual information*[13] .

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 variables with joint distribution $p(x, y)$ as follows:

$$JE(Y, X) = -\sum_x \sum_y p(x, y) \log p(x, y). \quad (3)$$

When we model images as random variables, the joint entropy is computed by gathering joint histogram between the two images. For two patches, $p_1, p_2 \in S_i \in T_s$ the joint entropy is given by:

$$JE(p_1, p_2) = \sum_i b_s(i) \log b_s(i) \quad (4)$$

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

c) Mutual Information

Mutual information (MI) is the measure of the statistical dependency between two or more random variables [9]. The mutual information of two random variables X and Y can be defined in terms of the individual entropies of both X and Y and the joint entropy of the two variables $JE(X, Y)$. Assuming pixels of the patches p_1, p_2 the mutual information between the two patches is

$$MI(p_1, p_2) = E(p_1) + E(p_2) - JE(p_1, p_2). \quad (5)$$

As image similarity measure, MI has been found to be successful in many application domains.

In addition to the entropy-based measures, we also utilized traditional image similarity metrics including Kullback-Leibler(KL) divergence [14], L1 and L2 norms [16], Structural Similarity Index (SSIM) [15] and Peak-signal-to-noise ratio (PSNR) [15].

III. Results and Analysis

For evaluation we used CATSvsDOGS [16], and CIFAR100 [17] datasets. The techniques described above were employed to learn and classify these datasets. To gather enough data that enable characterization of each preprocessing technique, we set up a consistent training environment with fixed network architectures, training procedure, as well as hyper parameters configuration. We performed two sets of experiments to determine the impacts of algorithm POR (Table 1) on training. The first experiment was designed to determine correlation between the preprocessing techniques and network

training performance while the second was conducted to characterize the impact of granularity of patches on training. Below we present the analysis of results obtained using each approach. The results are summarized in Figs. 4 and 5.

A. Patch Ordering

Figure 4 shows results obtained when training Inception network to classify CIFAR100 (Top*)* and Cats vs Dogs (Bottom) datasets using slow learning rate and Adams optimization [18]. Plots on the right side depict test performance of the network at different iterations. In both setups, the mutual information technique speeds up learning rate more than all others while some techniques degrade the learning rate compared to regular training.

However, all techniques converge to the same performance as the regular training when trained for 10000 iterations. Given these results we answer the questions posed in the earlier sections. The question of whether ordering patches of the input based on some measure to help training can partially be answered by the empirical evidence that indicate reconstructing the input using the *MI* measure enables faster convergence. Dataset reordered using the MI measure achieves similar accuracy as the unmodified dataset in ¼ of the total iterations. In support of this we hypothesize that *informed ordering techniques enable robust feature extraction and make learning efficient.* To conclusively prove this hypothesis, one must consider variety of experimental setup. For instance, to rule out other factors for the observed results, we must perform similar experiments using different datasets, learning techniques, hyper parameter configuration and network architectures.

Given that most of these techniques remove human recognizable features by reordering (Figure 3) and the experimental results that not all ordering techniques compromise training or testing accuracy, we make the following claim: *Training and generalization performance of classification networks based on the deep convolutional neural network architecture is uncorrelated with human ability to separate the training set into the various classes.*

B. Patch ordering impact on Training

In this section we provide analysis of the impact of the patch-ordering preprocessing technique on training convolutional neural networks.

Let us consider the mutual information (MI) metric, which outperforms all other metrics. As mentioned in previous sections the MI index is used as a measure of statistical dependency between patches for patch ordering. Given two patches (also applies to images) p_1, p_2 the mutual information formula (Eqn. 5) computes an index that describes how well you can predict p_2 given the pixel values in p_1. This measures the amount of information that image p_1 contains about p_2. When this index is used to order patches of an input, the result consists of patches ordered in descending order according to their MI index. For instance, consider a 32 by 32 wide image with sixteen 8 by 8 patches (see representation, I, below). If we take patch $p(0,0)$ to be the reference patch, Algorithm 1 in the first iteration computes MI index of every other patch with the reference patch and moves the one with the highest index to position (0,1) and updates the reference patch. At the end, the algorithm generates an image such that the patch at (0,0) has more similarity to patch at (0,1) which has more similarity to patch at (0,2) etc. In other words, adjacent patches explain each other well more than patches that are further away from each other. *How does this impact training?*

To answer this question let us consider the convolution operator [19] and the gradient decent optimization [20] training technique. This algorithm employs Adam optimization and the SoftMax cross-entropy loss, to update network parameters.

We trained the networks using online training [21] mechanism, where error calculations and weight updates occur after every sample. Our hypothesis is that *samples preprocessed using the*

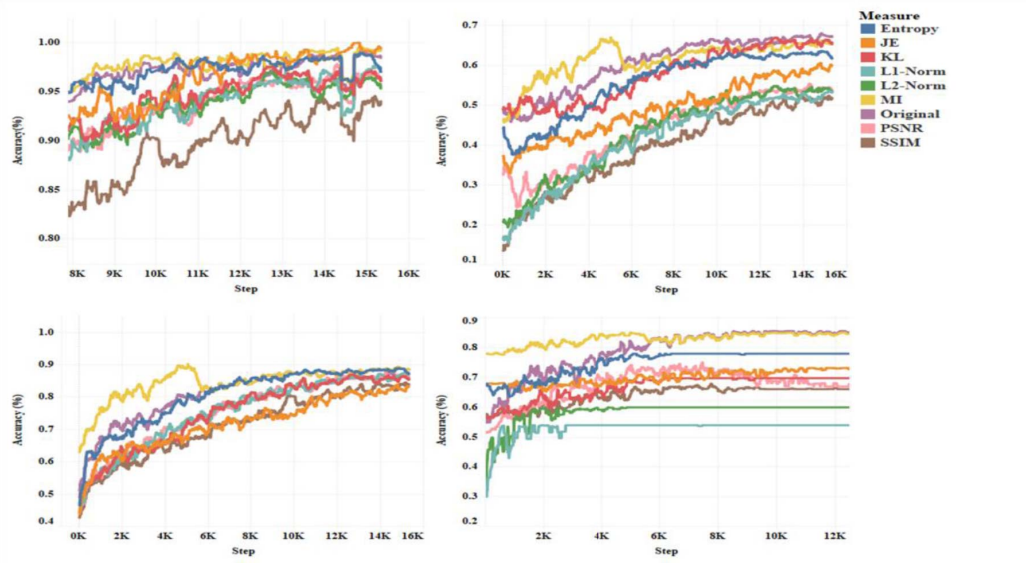

Figure 4: Accuracy in validation classification as a function of training iterations of CIFAR100 (top) and CATSvsDOGS (bottom) datasets using Inception network architecture. We show training (left and testing (right) results of all the similarity and statistical measure-based patch ordering techniques: patch ordering using mutual information (MI, yellow) between adjacent samples outperforms all other techniques. During training all parameters except for the training dataset are fixed.

MI measure enable rapid progress lowering the cost in the initial stages of the training.

In other words, when the input is rearranged such that adjacent samples have similar pixel value distribution, the convolution filters extract features that produce smaller error. To illustrate this let us assume the following values for the first few patches of an image (color coded in the matrix below). For simplicity let us assume the image is binary and all the pixel values are either 0 or 1.

$$I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Also consider the following 3x3 convolution filter whose values are initialized randomly: $K = \begin{bmatrix} 1 & 5 \\ 0 & 1 \end{bmatrix}$ .If one performs convolution of the original image with the above kernel K, the resulting feature map consists of the following values.

$$I * K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 6 & 6 & 6 \\ 1 & 6 & 6 & 7 & 7 & 7 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

To maintain resolution of the original image we use 0-padding before applying convolution. Applying a 3x3 max pooling operation with stride 3 to the convolved sample generates a down-sampled feature-map of the i^{th} training sample, $\boldsymbol{x_i} = \begin{bmatrix} 6 & 7 \\ 1 & 1 \end{bmatrix}$,which is used as an input to compute probability score of each class in a classifier. In this illustration we consider a binary classifier with two possible outcomes.

$$I' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Given the weight matrix $\boldsymbol{W} = \begin{bmatrix} 0.01 & -0.05 & 0.1 & 0.05 \\ 0.7 & 0.2 & 0.05 & 0.16 \end{bmatrix}$ and a bias vector $\boldsymbol{b} = \begin{bmatrix} 0.2 \\ -0.4 \end{bmatrix}$, the effective SoftMax cross-entropy loss for the correct class can be computed using the normalized probabilities assigned to the correct label $\boldsymbol{y_i}$ given the image $\boldsymbol{x_i}$ parameterized by $\boldsymbol{W}$ (Eqn. 12).

$$\boldsymbol{P(y_i|x_i; W)} = \frac{\boldsymbol{e^{f_{y_i}}}}{\sum \boldsymbol{e^{f_i}}} \tag{12}$$

The probabilities of each class using $\boldsymbol{f(W, x)} = \boldsymbol{(Wx_i + b)}$ objective function after normalization are $\begin{bmatrix} 0.01 \\ 0.99 \end{bmatrix}$. Assuming the probability of the correct class is **0.01** the cross-entropy loss becomes **4.60**. Note here patches are ordered left to right and adjacent patches have MI indices that are larger than those that are not adjacent. After ranking each 3x3 patch using the MI measure and preprocessing the sample using Algorithm 1, the resulting sample I' has ordering grey, green, pink and blue. In this example MI of the green with the grey patch is 0.557 while the blue has MI index equal to 0.224 against the same reference patch.

Once I' is convolved using the same kernel K, the resulting downscaled feature map, $\boldsymbol{x_i}' = \begin{bmatrix} 6 & 5 \\ 6 & 7 \end{bmatrix}$, produces $\begin{bmatrix} 0.13 \\ 0.87 \end{bmatrix}$ probabilities for each class. Taking the negative logarithm of the correct class results in a prediction loss equal to **2.01**.

This is the underlying effect we would like all measure to have when reordering the training dataset. However, it is not guaranteed. For instance, if we use l2-norm measure to sort the patches, the resulting loss becomes **4.71**, which is higher compared to the unmodified original sample. As a result, the training is slowed down since larger loss means more iterations are required for the iterative optimization to converge.

C. Patch Size Impact on Training

To characterize the effect of patch size, we performed controlled experiments where only the patch size is the varying parameter. The results and unmodified and preprocessed samples are depicted in Fig. 5.

As can clearly be seen in the plot, the network makes rapid progress lowering the cost when trained on a 4x4 patch ordered datasets. *Based on the empirical evidence and observations, we believe patch-ordering impact is more effective when mutual information index is combined with small patch size.* To clarify consider dividing the above sample into nine 2x2 patches (matrix I'').

$$I'' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

If the patches are reordered using MI measure(I''') against a reference patch p(0,0), and convolve the reordered sample, , using the same filter $K = \begin{bmatrix} 1 & 5 \\ 0 & 1 \end{bmatrix}$, the resulting normalized prediction probabilities are $\begin{bmatrix} 0.14 \\ 0.86 \end{bmatrix}$, which results in a loss of **1.96** after the first iteration.

$$I''' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

This is one explanation for the observed results, however, we cannot draw a conclusion regarding proportionality of patch size to training performance. If the pink and red patches of the above sample, which have same MI index, were to swap places, the resulting loss would have been **4.71** which is greater than the loss generated using 3x3 patch size. In this scenario training is slowed down which explains the behavior of training VGG on the different patch sizes.

IV. Summary and Discussion

We proposed several automated patch ordering techniques to assess their impact on training and characterize the relationship between dataset characteristics and training and generalization performances. We used traditional image similarity measures as well as information theory-based content measures of images to reconstruct training samples. The empirical evidence and our analysis using multiple datasets and Inception network architecture, suggest that training a convolutional neural network by supplying inputs that have some ordering, at patch level, according to some measure, are effective in allowing a gradient step to be taken in a direction that minimizes cost at every iteration. Specifically, our experiments show that supplying training sample such that the mutual information between adjacent patches is minimum, reduces the loss faster than all other techniques when optimizing a non-convex loss function. In addition, using these systematic approaches, we have shown that image

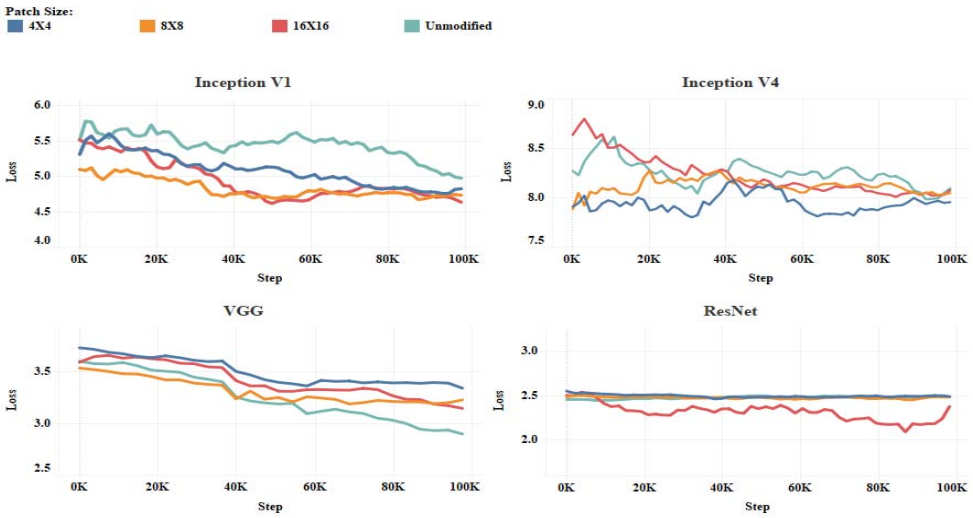

Figure 5: Patch granularity impact on training multiple networks using the CIFAR100 dataset. Total training loss for Unmodified dataset, and datasets modified by applying Algorithm 1 using the MI metric and patch sizes 4x4, 8x8 and 16x16. The overall size of each sample is 32 by 32.

characteristics and human recognizable features contained within training samples are uncorrelated with network performance. In other words, the view that CNNs learn combination of features in increasing abstraction does not explain their ability to fit images that have no recognizable features for the human eyes. Such a view also discounts the ability of the networks to fit random noise during training. Instead further investigation using theoretical characterizations such as the IB method are necessary to formally characterize learnability of a given training set using CNN.

V. RFERECNCES

[1] N. Tishby and N. Zaslavsky, "Deep Learning and the Information Bottleneck Principle," *ArXiv150302406 Cs*, Mar. 2015.

[2] A. M. Saxe *et al.*, "ON THE INFORMATION BOTTLENECK THEORY OF DEEP LEARNING," p. 27, 2018.

[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *ArXiv161103530 Cs*, Nov. 2016.

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

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. 2006.

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

[9] 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.

[10] H. S. Leff and A. F. Rex, Eds., *Maxwell's demon: entropy, information, computing*. Princeton, N.J: Princeton University Press, 1990.

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

[12] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[13] 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.

[14] H. B. Mitchell, "Image Similarity Measures," in *Image Fusion*, Springer, Berlin, Heidelberg, 2010, pp. 167–185.

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

[16] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3498–3505.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[18] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," *ArXiv170205659 Cs*, Feb. 2017.

[19] D. Garcia-Gasulla *et al.*, "On the Behavior of Convolutional Nets for Feature Extraction," *ArXiv170301127 Cs Stat*, Mar. 2017.

[20] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *ArXiv12065533 Cs*, Jun. 2012.

[21] I. Loshchilov and F. Hutter, “Online Batch Selection for Faster Training of Neural Networks,” *ArXiv151106343 Cs Math*, Nov. 2015.