EXPLORING DEEP LEARNING USING INFORMATION THEORY TOOLS AND PATCH ORDERING

Anonymous authors
Paper under double-blind review

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. An image patch is a group of pixels residing in a continuous area contained in the sample. 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. In addition, we conduct systematic experiments and provide evidence that CNN's generalization capabilities do not correlate with human recognizable features present in training samples. We utilized the framework not only to show that spatial locality of features within samples do not correlate with generalization, but also to expedite convergence while achieving similar generalization performance. 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.

1 INTRODUCTION

Advances in Deep Learning (DL) and Convolutional Neural Networks (CNN) have dramatically improved the state-of-the-art in computer vision tasks. Many of these breakthroughs are attributed to the successive feature extraction and an increasing abstract representation of the underlying training data using multi-stage simple operations such as convolution. These operations possess several model parameters such as convolution filter which are trained to amplify and refine information that are relevant to the classification, and to suppress irrelevant information (Ian Goodfellow et al., 2006). The training procedure uses backpropagation algorithm with supervision. This algorithm combined with Stochastic Gradient Descent (SGD), attempts to minimize the overall error or deviation from true label by computing the error gradient of each parameter and by performing small updates in the opposite direction. Despite their success, theoretical characterization of deep learning and CNNs is still at its infancy and valuable correlations such as number of layers needed to achieve a certain performance are not well understood. However, the success of deep learning has spawned many research avenues in order to explain deep network's exceptional generalization performance (Saxe et al., 2018) (Mehta and Schwab, 2014) (Pai, 2016; Tishby and Zaslavsky, 2015). One 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. In particular it is based on the *information bottleneck* (IB) method which is concerned with the problem of how one *extracts an efficient representation of relevant information contained in a large set of features* (Slonim, 2002). Saxe et al., (2018) proposes 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 (Saxe et al., 2018). In this context a *relevant information*, of a cat vs dog classification task for instance, is the information pattern present in all the cat samples useful for predicting any picture of a cat. With this view, the amount of information relating the training set and the labels encoded in the hidden layers can be measured over the course of training (Tishby and Zaslavsky, 2015). Inspired by this view, we use information theoretic measures of entropy extended to measure image characteristics, to develop preprocessing techniques that enable robust features extraction during training. 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:

[1] *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?*

In supervised learning, we are interested in good feature representations of the input pattern that enable good prediction of the label (Janocha and Czarnecki, 2017). As a result, 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 Zhang et al. (2016) with intriguing outcomes. The authors presented 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 highlighted in Zhang et al. (2016) and 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 contribution 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.

Inception (Szegedy et al., 2015) architecture, known for achieving exceptional results on image classification tasks, is used for evaluation. The network is first evaluated 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 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.

2 PATCH ORDERING FOR ROBUST FEATURE EXTRACTION

The success of CNNs stem from their ability to automatically learn feature extractors. During training, CNNs construct hierarchy of feature representations and use superposition of the hierarchical features

[1] This work is supported in part by NSF award # 1301885

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 used are described below. The results are summarized in section 4.

2.1 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

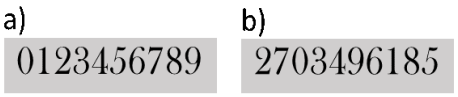

Figure 1: Toy images to illustrate the importance of ordering for classification. The two images in this context are of the same class. The label can be 'digits 0-9'

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).

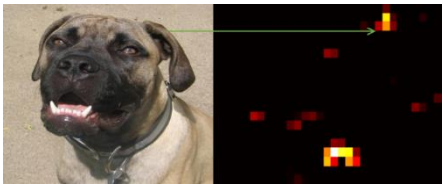

Figure 2: a) input image b) feature map after convolving input with filter.

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 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 Tishby and Zaslavsky (2015) 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. 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 measures discussed in subsections below are L1-norm, L2-norm, Structural Similarity, Joint Entropy, KL-Divergence, and Mutual Information.

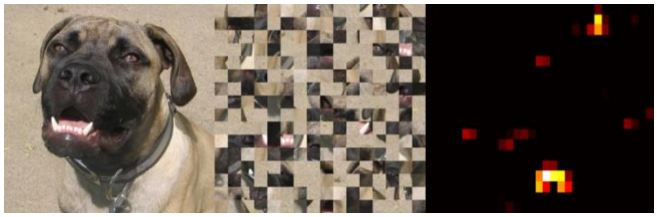

Figure 3. An illustration of patch ordering. a) Input image, b) reconstruction of the input using structural similarity of patches and c) feature map generated by convolving b). Note that the encoding of spatial location of a feature is not present in the feature map. The original image (a) is reconstructed 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. The results are summarized in Section 4.

2.1.1 ENTROPY-BASED MEASURES FOR PATCH ORDERING

2.1.1.1 ENTROPY

Information theory provides a theoretical foundation to quantify information content, or the uncertainty, of a random variable represented as a distribution (Cover and Thomas, 2006; Feixas et al., 2014). Information theoretic measures of content can be extended to image processing and computer vision (Leff and Rex, 1990). 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 (Feixas et al., 2014). 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

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

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. (Bonev, 2010). 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 (Feixas et al., 2014). A simple model would assume that each pixel is an independent and identically distributed random variable *(i.i.d)* realization (Feixas et al., 2014). 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:

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

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 pixel values. While individual entropy is the basic index used for ordering, we also consider strategies that relate two image patches. These measures include *joint entropy*(Feixas et al., 2014), *kl-divergence* (Szeliski, 2010), and *mutual information*(Russakoff et al., 2004) .

2.1.1.2 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:

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

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:

$$JE(p_1, p_2) = \sum_i b_s(i) \log b_s(i), \tag{4}$$

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

2.1.1.3 MUTUAL INFORMATION

Mutual information (MI) is the measure of the statistical dependency between two or more random variables (Feixas et al., 2014). 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 pixel values of the patches p_1, p_2 the mutual information between the two patches is

$$MI(p_1, p_2) = E(p_1) + E(p_2) - JE(p_1, p_2). \tag{5}$$

As noted in Russakoff et al. (2004), maximizing the mutual information between patches 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.

2.1.2 ADITIONAL MEASURES

2.1.2.1 KULLBACK-LEIBLER (K-L) DIVERGENCE

K-L Divergence is another measure we use to assess similarity of patches with in a sample. It's a natural distance measure from a pixel distribution p_1 to another distribution p_2 and is defined as:

$$D_{k||L}(p_1, p_2) = \sum_i p_{1\,i} \log \frac{p_{1_i}}{p_{2_i}}, \tag{6}$$

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

2.1.2.2 L1 NORM

Given two equal sized vectors a and b representing two patches of an image, the L_1 distance (Mitchell, 2010) is defined as

$$L_1(p_1, p_2) = ||p_1 - p_2|| = \sum_{i=1} |p_{1_i} - p_{2\,i}|, \tag{7}$$

This is sum of lengths between corresponding pixel value at index i over the size of the patch.

2.1.2.3 L2 NORM

L2 norm is a common measure used to assess similarity between images.

$$L_2(p_1, p_2) = ||p_1 - p_2||_2 = \sqrt{\sum_{i=1}(p_{1_i} - p_{2_i})^2}. \quad (8)$$

This can be interpreted as the Euclidean distance between the two vectors p_1 and p_2 representing the patches (Mitchell, 2010).

2.1.2.4 STRUCTURAL SIMILARITY INDEX (SSIM)

SSIM is usually used for predicting image quality using a reference image. Given two vectors $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ the SSIM index (Horé and Ziou, 2010) is given by:

$$SSIM(p_2) = \frac{(2\mu_{p_1}\mu_{p_2} + C_1)(2\sigma_{p_1p_2} + C_2)}{(\mu_{p_1}{}^2 + \mu_{p_2}{}^2 + C_1)(\sigma_{p_1}{}^2 + \sigma_{p_2}{}^2 + C_2)} \quad (9)$$

where the terms μ and σ are the mean and variances of the two vectors and $\sigma_{p_1p_2}$ is the covariance of p_1 and p_2. See (Horé and Ziou, 2010) for detail on this measure.

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.

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 $\boldsymbol{Network}$

2.1.2.5 PEAK SIGNAL TO NOISE RATIO (PSNR)

PSNR (Horé and Ziou, 2010) is another objective metric widely used in CODECs to assess picture quality. PSNR can be defined in terms of the mean squared error (MSE). The MSE of two metrices having the same size N is defined as:

$$MSE(a,b) = \frac{1}{N^2}\sum_{i}^{N}\sum_{j}^{N}(a_{ij} - b_{ij})^2. \quad (10)$$

The PSNR measure of two patches p_1 and p_2 can then be expressed as:

$$PSNR = 20\log_{10}\left(\frac{MAX}{\sqrt{MSE(p_1,p_2)}}\right), \quad (11)$$

where MAX is the maximum possible pixel value of the reference patch p_1.

3 EXPERIMENTAL SETUPS

3.1 DATASETS

For evaluation we used CATSvsDOGS (Parkhi et al., 2012) and CIFAR100 (Krizhevsky and Hinton, 2009). The techniques described above along with the several network architectures, 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. The results are summarized in section 4.

4 RESULTS AND ANALYSIS

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.

4.1 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 (Janocha and Czarnecki, 2017). 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.

4.2 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 (Garcia-Gasulla et al., 2017) and the gradient decent optimization (Bengio, 2012) approach. This algorithm employs Adam optimization technique and the SoftMax cross-entropy loss, to update network parameters.

We trained the networks using online training (Loshchilov and Hutter, 2015) mechanism, where error calculations and weight updates occur after every sample. Our hypothesis is that *samples preprocessed using the 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.

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.

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

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 x_i 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.

$$\boldsymbol{x_i} = \begin{bmatrix} 6 & 7 \\ 1 & 1 \end{bmatrix}$$

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).

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

The probabilities of each class using $\boldsymbol{f}(\boldsymbol{W}, \boldsymbol{x}) = (\boldsymbol{W}\boldsymbol{x_i} + \boldsymbol{b})$ 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.

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

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 (Eqn. 8) 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.

4.3 PATCH SIZE IMAPCT 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.

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

If the patches are reordered using MI measure against a reference patch p(0,0), and convolve the reordered sample, $I''' = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$, 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.

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.

5 SUMMARY AND DISCUSSION

We proposed several automated patch ordering techniques to assess their impact on training and assess the relationship between dataset characteristics and training and generalization performances. Our methods rank, and reorder patches of every sample based on a standalone measure and based on similarity

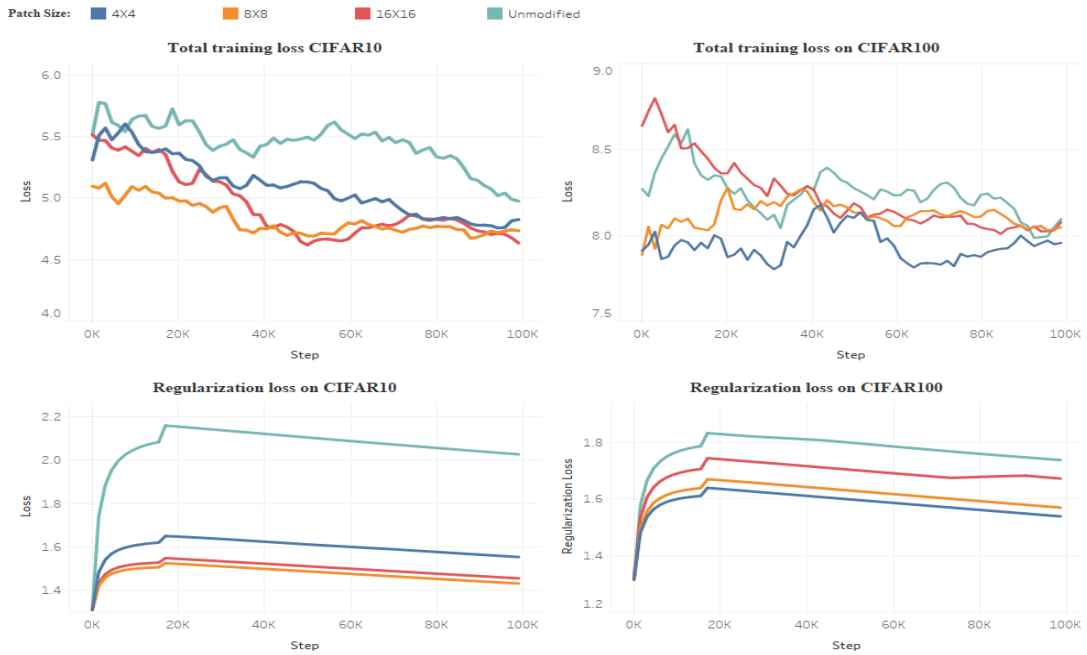

Figure 5: Comparison of training performance of Inception using different patch sizes. CIFAR10 (left) and CIFAR100 (right) datasets. Total training loss (top) and regularization loss (bottom) for Unmodified dataset, and datasets modified by applying Algorithm 1 using the MI metric and patch sizes 4x4, 8x8 an 16x16). The overall size of each sample is 32 by 32.

between patches. We used traditional image similarity measures as well as information theory-based content measures of images to reconstruct training samples. We started off with theoretical foundations for measures used and highlighted the intuition regarding ordering and classification performance. We tested the proposed methods using several architectures, each effectively designed to achieve high accuracy on image classification tasks. 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 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.

REFERENCES

Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. ArXiv12065533 Cs.

Bishop, C.M., 2006. Pattern recognition and machine learning, Information science and statistics. Springer, New York.

Bonev, B.I., 2010. Feature Selection Based on Information Theory 200.

Cover, T.M., Thomas, J.A., 2006. Elements of Information Theory 774.

Feixas, M., Bardera, A., Rigau, J., Xu, Q., Sbert, M., 2014. Information theory tools for image processing. Synth. Lect. Comput. Graph. Animat. 6, 1–164.

Garcia-Gasulla, D., Parés, F., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., Suzumura, T., 2017. On the Behavior of Convolutional Nets for Feature Extraction. ArXiv170301127 Cs Stat.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT Press.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90

Horé, A., Ziou, D., 2010. Image quality metrics: PSNR vs. SSIM. pp. 2366–2369. https://doi.org/10.1109/ICPR.2010.579

Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2006. Deep Learning.

Janocha, K., Czarnecki, W.M., 2017. On Loss Functions for Deep Neural Networks in Classification. ArXiv170205659 Cs.

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images.

Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P., n.d. Exploring Strategies for Training Deep Neural Networks 40.

Leff, H.S., Rex, A.F. (Eds.), 1990. Maxwell's demon: entropy, information, computing, Princeton series in physics. Princeton University Press, Princeton, N.J.

Loshchilov, I., Hutter, F., 2015. Online Batch Selection for Faster Training of Neural Networks. ArXiv151106343 Cs Math.

Mehta, P., Schwab, D.J., 2014. An exact mapping between the Variational Renormalization Group and Deep Learning. ArXiv14103831 Cond-Mat Stat.

Mitchell, H.B., 2010. Image Similarity Measures, in: Image Fusion. Springer, Berlin, Heidelberg, pp. 167–185. https://doi.org/10.1007/978-3-642-11216-4_14

Pai, S., 2016. Convolutional Neural Networks Arise From Ising Models and Restricted Boltzmann Machines 10.

Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V., 2012. Cats and dogs, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition. Presented at the 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3498–3505. https://doi.org/10.1109/CVPR.2012.6248092

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

Saxe, A.M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B.D., Cox, D.D., 2018. ON THE INFORMATION BOTTLENECK THEORY OF DEEP LEARNING 27.

Simonyan, K., Zisserman, A., 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv14091556 Cs.

Slonim, N., 2002. The Information Bottleneck: Theory and Applications 157. https://doi.org/2002

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2015. Rethinking the Inception Architecture for Computer Vision. ArXiv151200567 Cs.

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

Tishby, N., Zaslavsky, N., 2015. Deep Learning and the Information Bottleneck Principle. ArXiv150302406 Cs.

Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O., 2016. Understanding deep learning requires rethinking generalization. ArXiv161103530 Cs.

RELEVANT DETAIL ON TRAINING CNN

A typical CNN architecture is structured as a series of data processing and classification stages. It consists of several layers with tens of thousands of neurons in each layer, as well as millions of connections between neurons. In the data processing stages, there are two kinds of layers: *convolutional* and *pooling* layers (Ian Goodfellow et al., 2006). In a convolutional layer, each neuron representing a filter is connected to a small patch of a feature map from the previous layer through a set of weights. The result of the weighted sum is then passed through an activation function that performs non-linear transformation and prevent learning trivial linear combinations of the inputs. The pooling layers are used to reduce computation time by sub-sampling from convolution outputs and to gradually build up further spatial and configural invariance (Ian Goodfellow et al., 2006).

Discrete image convolution [10] is used to extract information from training samples. For 2D functions I and K, the convolution operation is defined as:

$$I(u,v) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u-i, v-j) K(i,j) \tag{13}$$

In CNNs, convolution at a given layer is applied to the output of the previous layer and the limits of the summation are determined by the size the input I and of the filter K. For a given layer l, the input comprises $(fm)_1^{l-1}$ feature maps from the previous layer (Goodfellow et al., 2016). When $l = 1$, the input is a single image consisting of one or more channels. The output of layer l consists of $(fm)_1^l$ feature maps.

FEATURE MAPS

Feature maps are encodings of features and their locations present in the input (Ian Goodfellow et al., 2006). They are obtained by convolving the input with a fixed sized *filter* (or kernel, $\boldsymbol{K}$) usually having dimensions significantly smaller than the input. The i^{th} feature map of layer l, represented y_i^l is computed as

$$y_i^l = B_i^l + \sum_{j=1}^{m^{(l-1)}} K_{i,j}^l * y_j^{l-1} \tag{14}$$

where $m^{(l-1)}$ is the total number of feature maps generated by the previous layer, B_i^l is a bias matrix and $K_{i,j}^l$ is a filter connecting the j^{th} feature map in layer l (Ian Goodfellow et al., 2006). The trainable weights are found in the filters $K_{i,j}^l$ and the bias matrices B_i^l.

TRAINING AND THE BACKPROPAGATION ALGORITHM

During training, CNNs attempt to determine the filter weights to approximate target mapping g (Goodfellow et al., 2016). In practice, g is a function fitted by the training data using supervised training procedures. The training set

$$T_s = \{(x_n, y_n): 1 \leq n \leq N\} \tag{15}$$

comprises both the input values x_n and corresponding desired output values $y_n \approx g(x_n)$. N is the total number of examples in the training set.

BACKPROPAGATION ALGORITHM

Supervised training is accomplished by adjusting the weights w of the network to minimize a chosen objective function that measures the deviation of the network output, y_n, from the desired target output x_n (Goodfellow et al., 2016). Some of the common measures are cross-entropy error measure (Janocha and Czarnecki, 2017) given by

$$E(w) = \sum_{n=1}^{N} E_n(w) = \sum_{n=1}^{c}\sum_{k=1}^{c} t_{n,k}\log\big(y_k(x_n,w)\big), \tag{16}$$

and the squared-error measure (Bishop, 2006) given by

$$E(w) = \sum_{n=1}^{N} E_n(w) = \sum_{n=1}^{c}\sum_{k=1}^{c} t_{n,k}\log\big(y_k(x_n,w)\big), \tag{17}$$

where $t_{n,k}$ is the k^{th} entry of the target value t_n and c is the number of distinct classes in T_s.

Deep learning with stochastic training seeks to minimize $E_n(\mathrm{w})$ with respect to the network weights w. The necessary criterion can be written as

$$\frac{\partial E_n}{\partial w} = 0, \tag{18}$$

where $\frac{\partial E_n}{\partial w}$ is the gradient of the error E_n (Goodfellow et al., 2016). Since E_n is a high dimensional function, a closed-form exact solution is too expensive. Iterative optimization approach, commonly referred to as gradient descent *(Algorithm 1)*, is used to find optimal values of the parameters that best

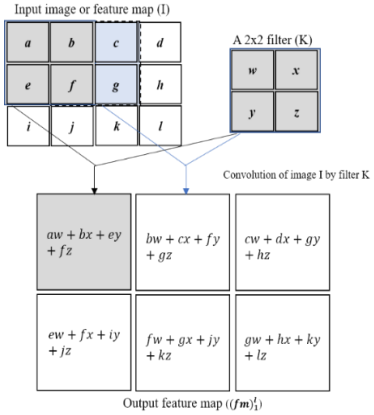

Figure 1. Illustration of convolution in a single convolution layer. If layer l is a convolutional layer, the input image (if l = 1) or a feature map of the previous layer is convolved by different filters to yield the output feature maps, $(\boldsymbol{fm})_1^l$, of layer l.

approximate a mapping between each sample in the training set to the desired output.
At each iteration, for a given weight vector $w[t]$, gradient descent takes a step in the direction of the steepest descent to reach a global minimum (Goodfellow et al., 2016) by computing weight update $\Delta w[t]$ and updating weight accordingly:

$$w[t+1] = w[t] + \Delta w[t] \tag{19}$$

where $\Delta w[t] = -\alpha\frac{\partial E_n}{\partial w[t]} = -\alpha\Delta E_n$ is the gradient of the error function with respect to w and α is the learning rate.

There are few different training protocols used for parameter optimization. These protocols are summarized in (Larochelle et al., n.d.). The most common ones are:
Stochastic training: when this protocol is employed, an input sample is chosen at random and the network weights are updated based on the error function $E_n(w)$.

Batch Training: with this protocol, all inputs of size N are processed and the weights are updated based on the overall error

$$\boldsymbol{E(w) = \sum_{n=1}^{N} E_n(w).} \tag{20}$$

Online training: every sample is processed only once, and the weights are updated using the error $E_n(w)$.

Mini-batch training: during mini-batch training a random subset (mini-batch) of samples of size M from the training set is processed and the weights are updated based on the cumulative error

$$E_M(w) = \sum_{n=1}^{M} E_n(w). \tag{21}$$

Table 2: The backpropagation algorithm using mini-batch training protocol.

Input: $\boldsymbol{Learnig\ rate\ (\alpha), max_{iteration}, T_s}$
Output: $\boldsymbol{Network}$

1. **Initialize** $\boldsymbol{weights}$ *to 0*
2. **Draw** a batch of size *BatchSize*

For *(i=1 to BatchSize):*

3. **Select** $\boldsymbol{input\ x_i}$
4. **PropagateForward** $\boldsymbol{x_i}$
5. **ComputeError** $\boldsymbol{E_i}$
6. **Compute** $\boldsymbol{gradient\ update}\ \Delta\boldsymbol{E_i}$
7. **Backpropagate** and update weights
8. **Go to** 2

End
Return $\boldsymbol{Network}$