Aggregated Learning: A Vector-Quantization Approach to Learning Neural Network Classifiers

Masoumeh Soflaei, Hongyu Guo, Ali Al-Bashabsheh, Yongyi Mao, Richong Zhang

1University of Ottawa, Ottawa, Canada, 2National Research Council Canada, 3Beijing Advanced Institution on Big Data and Brain Computing, Beihang University, Beijing, China

Abstract

We consider the problem of learning a neural network classifier. Under the information bottleneck (IB) principle, we associate with this classification problem a representation learning problem, which we call “IB learning”. We show that IB learning is, in fact, equivalent to a special class of the quantization problem. The classical results in rate-distortion theory then suggest that IB learning can benefit from a “vector quantization” approach, namely, simultaneously learning the representations of multiple input objects. Such an approach assisted with some variational techniques, result in a novel learning framework, “Aggregated Learning”, for classification with neural network models. In this framework, several objects are jointly classified by a single neural network. The effectiveness of this framework is verified through extensive experiments on standard image recognition and text classification tasks.

Introduction


Under the IB principle, the core of learning a neural network classifier is to find a representation \( T \) of the input example \( X \), that contains as much as possible the information about \( X \) and as little as possible the information about the label \( Y \). The conflict between these two requirements can be formulated as a constrained optimization problem in which one requirement is implemented as the objective function and another requirement as the constraint (Navot and Tishby 2003; Tishby, Pereira, and Bialek 1999; Shamir, Sabato, and Tishby 2010). In this paper, we call this problem IB learning.

A key observation that has inspired this work is that the optimization formulation of IB learning resembles greatly the rate-distortion function in rate-distortion theory, i.e., the theory for quantizing signals (Shannon 1959). A careful investigation along this direction indeed reveals that, conceptually, there is an unconventional quantization problem that is closely related to IB learning. To that end, we formulate this problem, which we refer to as IB quantization. We prove that the objective of IB quantization, namely, designing quantizers that achieve the rate-distortion limit, is equivalent to the objective of IB learning. This result establishes an equivalence between the two problems.

In rate-distortion theory, it is well known that scalar quantizers, which quantize signals one at a time, are in general inferior to vector quantizers, which quantize multiple signals at once. The discovered equivalence between IB learning and IB quantization then suggests that IB learning may benefit from a “vector quantization” approach, in which the representations of multiple inputs are learned jointly. Exploiting variational techniques and the recently proposed mutual information neural estimation (MINE) method (Belghazi et al. 2018), we show that such a vector quantization approach to IB learning naturally results in a novel framework for learning neural network classifiers. We call this framework Aggregated Learning (AgrLearn).

Briefly, in AgrLearn, \( n \) random training objects are aggregated into a single amalgamated object and passed to the model; the model predicts the soft labels for all \( n \) examples jointly. The training of an AgrLearn model is carried out by solving a min-max optimization problem, derived a variational relaxation of the IB learning problem and a MINE approximation of mutual information.

We conducted extensive experiments, applying AgrLearn to the current art of deep learning architectures for image and text classification. Our experimental results suggest that AgrLearn brings significant gain in classification accuracy. In practice, AgrLearn can be easily integrated into existing neural network architectures. The proofs of some theoretical results are omitted due to length constraints. They will be included in an extended version of this paper.

1Our implementation of AgrLearn is available at https://github.com/SITE5039/AgrLearn
**Information Bottleneck Learning**

The overall context of this work is a classification setting, where we let \( X \) denote the space of objects to be classified and \( Y \) denote the space of class labels. Assume that the objects and labels are distributed according to an unknown distribution \( p_{XY} \) on \( X \times Y \), where instead we are given a set \( D := \{(X_1, Y_1), \ldots, (X_N, Y_N)\} \) of i.i.d samples from \( p_{XY} \). The objective of learning here is to find a classifier from \( D \) that classifies \( X \) into its label \( Y \).

Central to this classification problem is arguably the following representation learning problem: Find a representation of \( X \) that only contains the information about \( X \) relevant to its class label \( Y \). Such a problem can be naturally formulated using the information bottleneck principle (Tishby, Pereira, and Bialek 1999) and will be referred to as the Information Bottleneck (IB) learning problem.

In IB learning, one is interested in learning a representation \( T \) of \( X \) in some space \( T \) such that the mutual information \( I(X; T) \) between \( X \) and \( T \) is as small as possible whereas the mutual information \( I(Y; T) \) between \( T \) and the class label \( Y \) is as large as possible. Such a representation is sensible since it aims at squeezing away all information in \( X \) that is irrelevant to the classification task while keeping the relevant information intact. Intuitively, minimizing \( I(X; T) \) forces the model not to over-fit to the irrelevant features of \( X \), whereas maximizing \( I(Y; T) \) extracts all features useful for the classification task. The two optimization objectives are in conflict with each other. A natural formulation to the IB learning problem is to consider one objective as the optimization objective and the other as a constraint. This gives rise to the following constrained optimization problem, subject to the Markov chain \( Y \rightarrow X \rightarrow T \), find

\[
\hat{p}_{T|X} = \arg\min_{p_{T|X} : I(X; T) \leq A} -I(Y; T),
\]

for a nonnegative value \( A \), or equivalently,

\[
\hat{p}_{T|X} = \arg\min_{p_{T|X} : I(Y; T) \geq A'} I(X; T),
\]

for a nonnegative value \( A' \). The Markov chain assumption ensures that any information in feature \( T \) about label \( Y \) is obtained from \( X \) only. For later use, we denote the minimum mutual information in (2) as \( R_{IBL}(A') \), i.e.,

\[
R_{IBL}(A') = \min_{p_{T|X} : I(Y; T) \geq A'} I(X; T).
\]

We note that solving this IB learning problem, i.e., obtaining the optimal \( \hat{p}_{T|X} \) and its corresponding bottleneck representation \( T \) does not automatically solve the classification problem. It is still required to build a classifier that predicts the class label \( Y \) based on the representation \( T \) of \( X \). Nonetheless later in this paper, we will show that solving a variational approximation of the IB learning problem may, in fact, provide a direct solution to the classification problem of interest.

**Information Bottleneck Quantization**

We now formulate the Information Bottleneck (IB) quantization problem. Our objective in this section is to show that the IB quantization and IB learning problems are equivalent.

Let \((X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\) be drawn i.i.d from \( p_{XY} \). The sequences \((X_1, X_2, \ldots, X_n)\) and \((Y_1, Y_2, \ldots, Y_n)\) are denoted by \( X^n \) and \( Y^n \), respectively.

An \((n, 2^nR)\) IB-quantization code is a pair \((f_n, g_n)\) in which \( f_n \) maps each sequence \( X^n \) to an integer in \( \{1, 2, \ldots, 2^nR\} \) and \( g_n \) maps an integer in \( \{1, 2, \ldots, 2^nR\} \) to a sequence \( T^n := (T_1, T_2, \ldots, T_n) \in T^n \). Using the standard nomenclature in quantization, the quantity \( R \) is referred to as the rate of the code and \( n \) as the length of the code. Using this code, \( f_n \) encodes the sequence \( X^n \) as the integer \( f_n(X^n) \) and \( g_n \) reconstructs \( X^n \) as a representation \( T^n := g_n(f_n(X^n)) \).

Unlike standard quantization problems, the IB quantization problem uses a distortion measure that may depend on the code. To that end, for any \( x \in X \), \( t \in T \) and any two conditional distributions \( q_{Y|X} \) and \( q_{Y|T} \), define

\[
d_{IB}(x, t; q_{Y|X}, q_{Y|T}) := \text{KL}(q_{Y|X}(\cdot|x) \| q_{Y|T}(\cdot|t)),
\]

where \( \text{KL}(\cdot) \) is the Kullback–Leibler (KL) divergence.

Note that the code \((f_n, g_n)\), together with \( p_{XY} \), induce a joint distribution over the Markov chain \( Y^n \rightarrow X^n \rightarrow T^n \). Under this joint distribution the conditional distributions \( p_{Y|X} \) and \( p_{Y|T} \) are well defined for each \( i = 1, 2, \ldots, n \). Hence, given the code \((f_n, g_n)\) and for any two sequences \( x^n \in X^n \) and \( t^n \in T^n \), their IB distortion is defined as:

\[
\overline{d}_{IB}(x^n, t^n) := \frac{1}{n} \sum_{i=1}^{n} d_{IB}(x_i, t_i; p_{Y|X}, p_{Y|T}).
\]

We note that the quantity \( \overline{d}_{IB}(x^n, t^n) \) measures a “loss of information about \( Y^n \)” when the code \((f_n, g_n)\) is used to represent \( x^n \) as \( t^n \). Specifically, consider the source coding problem of compressing \( Y^n \) based on observing \( X^n = x^n \). If the conditional distribution \( p_{Y|X}(\cdot|x) \) for each \( x \) is misspecified as \( p_{Y|T}(\cdot|t_i) \) in the design of the source code, the average additional coding overhead per \( Y \)-symbol is precisely \( \overline{d}_{IB}(x^n, t^n) \).

Using this distortion measure, the IB quantization problem is to find a code \((f_n, g_n)\) having the smallest rate \( R \) subject to the constraint \( \mathbb{E}\overline{d}_{IB}(X^n, T^n) \leq D \), where \( \mathbb{E} \) denotes expectation. For given \( p_{XY} \) and \( T \), a rate distortion pair \((R, D)\) is called achievable if \( \mathbb{E}\overline{d}_{IB}(X^n, T^n) \leq D \) for some sequence of \((f_n, g_n)\) codes. As usual, the rate-distortion function for the IB quantization problem, which we denote by \( R_{IBQ}(D) \), is defined as the smallest rate \( R \) such that \((R, D)\) is achievable.

**Theorem 1** Given \( p_{XY} \) and \( T \), the rate-distortion function for the IB quantization problem can be written as

\[
R_{IBQ}(D) = \min_{p_{T|X} : \mathbb{E}d_{IB}(X,T) \leq D} I(X; T)
\]

where the expectation is defined as

\[
\mathbb{E}d_{IB}(X,T) := \sum_{x,t} d_{IB}(x,t; p_{Y|X}, p_{Y|T}) p_{XT}(x,t).
\]

This theorem provides a limit on the achievable rates of the IB quantization problem. We note that this result was
first shown in (Navot and Tishby 2003). However in (Navot and Tishby 2003), the result relies on the assumption that \(|T| \geq |X| + 2\), whereas in this theorem the condition is removed.

The form of the rate-distortion function \(R_{IBQ}\) for the IB quantization problem given in Theorem 1 resembles greatly the optimal objective of IB learning \(R_{IBL}\) in (3). More precisely, we have

**Theorem 2** \(R_{IBL}(A') = R_{IBQ}(I(X; Y) - A')\)

**Proof:** We have

\[
\begin{align*}
\mathbb{E}d_{IB}(X, T) & := \sum_{x, t} d_{IB}(x, t; p_{Y|X}, p_{Y|T})p_{XT}(x, t) \\
& = I(X; Y) - I(Y; T)
\end{align*}
\]

where the second equality is by the definition of \(d_{IB}\) and the Markov chain \(Y \rightarrow X \rightarrow T\) assumption. Hence, we may rewrite (6) in Theorem 1 as

\[
R_{IBQ}(D) = \min_{p_{T|X}: I(Y; T) \leq D} I(X; T) = \min_{p_{T|X}: \alpha I(Y; T)} I(X; T) = R_{IBL}(I(X; Y) - D)
\]

The theorem follows by substituting \(A' := I(X; Y) - D. \Box\)

This theorem relates the IB learning and IB quantization problems, where we note that \(I(X; Y)\) is a constant that only depends on \(p_{XY}\). By this theorem, solving the IB learning problem where the information about \(Y\) contained in \(T\) needs to be no less than \(A'\) is equivalent to solving the IB quantization problem so that the distortion is no more than \(I(X; Y) - A'\).

**Variational Approach to IB Learning**

Having established the equivalence between IB learning and IB quantization, we now turn to solve the IB learning problem. The objective of this section is to develop a variational approach to this problem which not only provides a bottleneck representation \(T\) for \(X\) but also leads to a classifier for the classification problem at hand. We note that the results presented in this section also underlies the “variational information bottleneck” approach of (Alemi et al. 2016).

We first establish the following result.

**Theorem 3** Under any distribution \(p_{YXT}\) that satisfies the Markov chain \(Y \rightarrow X \rightarrow T\), we have

\[
I(Y; T) \geq \mathbb{E}_{(x, y) \sim p_{XY}, t \sim p_{T|X}(|x|)} \log q_{Y|T}(y|t) + H(Y)
\]

for any conditional distribution \(q_{Y|T}\) of a random variable \(Y\) conditioned on \(T\). In addition, the above inequality holds with equality if and only if \(q_{Y|T}\) is equal to \(p_{Y|T}\).

As a consequence of this theorem, the mutual information \(I(Y; T)\) can be written as

\[
I(Y; T) = \max_{q_{Y|T}} \mathbb{E}_{(x, y) \sim p_{XY}, t \sim p_{T|X}(|x|)} \log q_{Y|T}(y|t) + H(Y).
\]

Substituting this in the IB learning problem as formulated in (1), we have

\[
\hat{p}_{T|X} = \arg \min_{p_{T|X}: I(Y; T) \leq A} -I(Y; T)
\]

\[
= \arg \min_{p_{T|X}: I(Y; T) \leq A} \left\{ \max_{q_{Y|T}} -E_{(x, y) \sim p_{XY}, t \sim p_{T|X}(|x|)} \log q_{Y|T}(y|t) \right\}
\]

\[
= \arg \min_{p_{T|X}: I(Y; T) \leq A} \left\{ -E_{(x, y) \sim p_{XY}, t \sim p_{T|X}(|x|)} \log q_{Y|T}(y|t) \right\}
\]

Now suppose we have a neural network representing the mapping \(p_{T|X}\) and that we represent \(q_{Y|T}\) using another network. Then we may construct an overall network by concatenating the two networks. Specifically, each object \(x\) will be first passed to the network \(p_{T|X}\), and the output \(T\) of the network is passed to the network \(q_{Y|T}\). If the true class label \(y\) is modeled as being generated from this concatenated network, it is easy to see that the cross-entropy loss \(\ell_{CE}\) of the network is the expectation above, i.e.,

\[
\ell_{CE} = -E_{(x, y) \sim p_{XY}, t \sim p_{T|X}(|x|)} \log q_{Y|T}(y|t).
\]

In other words, the IB learning problem can be formulated as solving the following optimization problem:

\[
\min_{p_{T|X}, q_{Y|T}} \ell_{CE}(\hat{p}_{T|X}, q_{Y|T}) \text{ subject to } I(Y; T) \leq A
\]

(9)

Hence, introducing a Lagrange multiplier, subsequently we will focus on the following unconstrained problem

\[
\min_{p_{T|X}, q_{Y|T}} \ell_{CE}(\hat{p}_{T|X}, q_{Y|T}) + \alpha I(X; T)
\]

for nonnegative \(\alpha\).

An apparent advantage of this approach to IB learning is that when the optimization problem (10) is solved, not only is the bottleneck representation \(T\) found, but also the entire classification network is obtained.

It is worth noting that the variational formulation (10) of IB learning can be viewed as a generalization of learning with standard neural networks under the cross-entropy loss. Specifically, learning with standard neural networks is a reduction of (10) in which the standard neural network contains no term \(\alpha I(X; T)\), or equivalently has \(\alpha = 0\).

The generalization of learning with standard neural networks to the formulation of IB learning in (10) is arguably beneficial in two respects:

1. The \(\alpha I(X; T)\) regularization term in (10) serves to control the model complexity so as to reduce the generalization gap.

2. Generalizing the deterministic map from \(X\) to \(T\) in standard neural networks to a stochastic one in (10) minimizes the cross-entropy loss \(\ell_{CE}\) over a larger space; this potentially allows further decrease of \(\ell_{CE}\), thereby achieving better classification accuracy. We note that the “Deep Variational Information Bottleneck” (DVIB) approach of (Alemi et al. 2016), not necessarily motivated by the same reason, uses the same variational bound of \(I(Y; T)\) and arrives at the same formulation as (10).

In the remainder of this paper, we present a new strategy, termed “Aggregated Learning”, to implement the IB learning formulation (10).
**Aggregated Learning (AgrLearn)**

We now introduce the Aggregated Learning (AgrLearn) framework for learning with neural networks. We will stay with the IB learning formulation of (10) while keeping in mind that it results from a variational approximation of the formulation in (1).

Recall from Theorem 1 that the IB learning problem is equivalent to the IB quantization problem. In the classical rate-distortion theory (Shannon 1959), it is well known that in order to achieve the rate-distortion limit of quantization, in general, one must consider the use of vector quantizers.

In the context of IB quantization, a vector quantizer is an IB-quantization code \((f_n, g_n)\) with \(n > 1\) whereas a scalar quantizer is an IB-quantization code \((f_n, g_n)\) with \(n = 1\). From rate-distortion theory, better quantizers result from using quantization codes with larger length \(n\). In particular, in order to achieve the rate-distortion function, it is in general required that the length \(n\) of the rate-distortion code be made asymptotically large.

Note that a scalar IB-quantization code \((f_1, g_1)\) maps \(x\) to \(T\) by

\[
T = g_1(f_1(x)) := (g_1 \circ f_1)(x).
\]

Under the equivalence between IB quantization and IB learning, the mapping \(g_1 \circ f_1\) induced by the scalar quantizer \((f_1, g_1)\) essentially defines a conditional distribution \(p_{T|X}\) in IB learning, which simply reduces to the deterministic function \(g_1 \circ f_1\). On the other hand, in learning with a standard neural network, the deterministic mapping, say \(h\), from the input space \(X\) to the bottleneck space \(T\) (which could refer to the space of feature representation at any intermediate layer of the network), can be regarded as implementing a scalar IB-quantization code \((f_1, g_1)\) with

\[
g_1 \circ f_1 = h.
\]

The superiority of vector quantizers to scalar quantizers then motivates us to develop a vector-quantization approach to IB learning, which we call Aggregated Learning or AgrLearn in short. – Like a vector quantizer, which quantizes \(n\) signals simultaneously, AgrLearn classifies \(n\) input objects jointly at the same time, the details of which are given below.

The framework of AgrLearn consists of two networks, which we refer to as the “main network” and the “regularizing network” respectively.

**The Main Network**

The main network takes as its input the concatenation of \(n\) objects \((X_1, X_2, \ldots, X_n) := X^n\). Such a concatenated input will be referred to as an “\(n\)-fold aggregated input”.

The main network consists of two parts, as seen in Figure 1. The first part, or the “pre-bottleneck” part, implements a deterministic mapping \(h : X^n \rightarrow T^n\) that maps an aggregated input \(X^n\) to an “aggregated bottleneck” \(T^n\) via

\[
T^n := (T_1, T_2, \ldots, T_n) := h(X^n).
\]

The second part, or the “post-bottleneck” part, implements a stochastic mapping \(q_{Y^n|T^n}\) from \(T^n\) to \(Y^n\) that factorizes according to

\[
q_{Y^n|T^n}(y^n|t^n) := \prod_{i=1}^{n} q_{Y_i|T^n}(y_i|t^n)
\]

Overall the main network expresses a stochastic mapping from \(X^n\) to \(Y^n\), which can be expressed as

\[
q_{Y^n|X^n}(y^n|x^n) := \prod_{i=1}^{n} q_{Y_i|T^n}(y_i|h(x^n))
\]

On the main network as specified by (13), define

\[
\ell_{CE}^{(n)} := -\mathbb{E}_{x^n,y^n \sim p_{X^n Y^n}} \log q_{Y^n|X^n}(y^n|x^n)
\]

where \(p_{X^n Y^n}\) is the distribution on \((X \times Y)^n\) induced by drawing \(n\) samples i.i.d. from \(p_{XY}\). Clearly \(\ell_{CE}^{(n)}\) is nothing more than the cross-entropy loss of the network’s predictive distribution \(q_{Y^n|X^n}\) for the aggregated input \(X^n\) with respect to their labels \(Y^n\). As we will be minimizing this cross-entropy loss function, we next discuss its properties.

Following Theorem 3,

\[
\min_{q_{Y^n|T^n}} \ell_{CE}^{(n)} = nH(Y) - I(Y^n; T^n).
\]

That is if the post-bottleneck component has sufficient capacity, then minimizing \(\ell_{CE}^{(n)}\) over the entire main network also maximizes \(I(Y^n; T^n)\).

**The Regularizing Network**

The regularizing network is essentially a mutual information neural estimator (MINE) network (Belghazi et al. 2018), which serves to estimate \(I(X; T)\) and penalizes it during the training of the main network. For a careful development of MINE, the reader is referred to (Belghazi et al. 2018). Here we only give a brief description.

**MINE in a Nutshell** Suppose that \(U\) and \(V\) are two spaces and that there is a joint distribution \(p_{UV}\) on \(U \times V\) defining a pair \((U, V)\) of random variables. Suppose that we can perform i.i.d. sampling of \(p_{UV}\) and we wish to estimate the mutual information \(I(U; V)\) from the samples. In the framework of MINE, a family \(\Gamma\) of functions is constructed as a neural network, where each \(\gamma \in \Gamma\) is a function mapping \(U \times V\) to the set \(\mathbb{R}\) of real numbers. Then due to dual representation of KL divergence (Donsker and Varadhan 1983), the mutual information \(I(U; V)\) can be estimated as

\[
\hat{I}(U; V) := \max_{\gamma \in \Gamma} \left\{ \mathbb{E}_{(u,v) \sim p_{U,V}} \gamma(u, v) \right\} - \log \mathbb{E}_{(u,v) \sim p_{U,V}} \exp (\gamma(u, v))
\]

We will denote the term that gets maximized in (16) by \(J(U, V; \gamma)\), namely,

\[
J(U, V; \gamma) := \mathbb{E}_{(u,v) \sim p_{U,V}} \gamma(u, v) - \log \mathbb{E}_{(u,v) \sim p_{U,V}} \exp (\gamma(u, v))
\]

and re-express \(\hat{I}(U; V)\) as

\[
\hat{I}(U; V) = \max_{\gamma \in \Gamma} J(U, V; \gamma)
\]
Figure 1: The main network in AgrLearn. The small circle denotes concatenation.

As usual, practical computation of $J(U, V; \gamma)$ exploits Monte-Carlo approximation based on samples drawn from $p_U p_V$. A natural way to apply MINE to the estimation of $I(X; T)$ in AgrLearn is taking $U := X^n$, $V := T^n$, $U = X^n$, $V = T^n$.

This allows us to estimate $I(X^n; T^n)$ by

$$\hat{I}(X^n; T^n) = \max_{\gamma \in \Gamma} J(X^n, T^n; \gamma)$$

where $T^n$ is computed by the pre-bottleneck component of the main network with $X^n$ as its input. We may then take $\hat{I}(X^n; T^n)$ as an approximation of $n I(X; T)$. The network implementing the computation of $J(X^n, T^n; \gamma)$ is referred to as the regularizing network.

**Training and Prediction**

With this development, we may define an overall objective function $\Omega(h, q_{Y^n} | T^n, \gamma)$ as

$$\Omega(h, q_{Y^n} | T^n, \gamma) := \ell(n) + \alpha J(X^n, T^n; \gamma)$$

where we note that the term $\alpha J(X^n, T^n; \gamma)$ also depends on $h$ implicitly. The above development then suggests that solving the IB learning problem in the form of (10) can be approximated by solving the following min-max problem:

$$\min_{h, q_{Y^n} | T^n} \max_{\gamma} \Omega(h, q_{Y^n} | T^n, \gamma)$$

In the training of AgrLearn, mini-batched SGD can be used to solve the above min-max problem. The training algorithm is given in Algorithm 1.

**Experimental Studies**

We evaluate AgrLearn with deep network architectures such as ResNet for classification tasks in both image and natural language domains. Standard benchmarking datasets are used.

We use mini-batched backprop for 400 epochs\(^3\) with exactly the same hyper-parameter settings without dropout. Specifically, weight decay is $10^{-4}$, and each mini-batch contains 64 aggregated training examples. The learning rate for the main network is set to 0.1 initially and decays by a factor of 10 after 100, 150, and 250 epochs. Each reported performance value (error rate or accuracy) is the median of the performance values obtained in the final 10 epochs by averaging that value over running the same setting 7 times.

**Image Recognition**

Experiments are conducted on the CIFAR-10, CIFAR-100 datasets with two widely used deep network architectures, namely ResNet (He et al. 2016) and WideResNet (Zagoruyko and Komodakis 2016b). The CIFAR-10\(^2\) Two additional protocols were investigated. Contextual Classification: For each object $X$, $n - 1$ random examples are drawn from the training set $D_X$ and concatenated with $X$ to form the input; the predictive distribution for $X$ generated by the model is then retrieved. This process is repeated $k$ times, and the average of the $k$ predictive distributions is taken as the label predictive distribution for $X$. Batched Classification: Let $D_X^n$ denote the set of all objects to be classified. In Batched Classification, $D_X^n$ are classified jointly through drawing $k$ random batches of $n$ objects from $D_X^n$. The objects in the $i^{th}$ batch $B_i$ are concatenated to form the input and passed to the model. The final label predictive distribution for each object $X$ in $D_X^n$ is taken as the average of the predictive distributions of $X$ output by the model for all batches $B_i$ containing $X$. Since we observe that all three protocols result in comparable performances, all results reported in the paper are obtained using the Replicated Classification protocol.

\(^3\) Here an epoch refers to going over $N$ aggregated training examples, where $N = |D_X|$. 

5814
dataset has 50,000 training images, 10,000 test images, and 10 image classes, and the CIFAR-100 dataset is similar to CIFAR-10 but with 100 classes.

We apply AgrLearn to the 18-layer and 34-layer Pre-activation ResNet (ResNet-18 and ResNet-34) (He et al. 2016) as implemented in (Liu 2017), and the 22-layer WideResNet (WideResNet-22-10) (Zagoruyko and Komodakis 2016b) as implemented in (Zagoruyko and Komodakis 2016a). The resulting AgrLearn model differs from original ResNet and WideResNet in its $n$ parallel soft-max layers in post-bottleneck part (as opposed to the single soft-max layer in ResNet and WideResNet) and the number of filters in the last layer of pre-bottleneck part, which is expanded by factor $n$. This expanding by factor $n$ is required because the input dimension in AgrLearn increases significantly, and the model is required to extract joint features across individual objects in the amalgamated example.

Note that fold number 1 (fold-1) denotes the standard neural network in which just one object passes to the network and fold number greater than 1 denotes an AgrLearn framework wherein multiple objects are aggregated and passed to the network. The quantity $\alpha$ is the coefficient of the second term in (19), in which $\alpha = 0$ corresponds to that only the cross-entropy loss is considered, and $\alpha > 0$ corresponds to that the regularization network is added to the main network.

### Predictive Performance

The prediction error rates of AgrLearn for different number of folds are shown in Tables 1, 2, and 3.

It can be seen that AgrLearn significantly boosts the performance of ResNet-18, ResNet-34 and WideResNet-22-10. For example, with respect to ResNet-18, the relative error reductions achieved by fold-2, where $\alpha = 0$ are 3.74%, and 2.83% on CIFAR-10, and CIFAR-100, and where $\alpha > 0$ the reductions are 3.86%, and 3.21% on CIFAR-10, and CIFAR-100 respectively.

Similarly significant improvement upon ResNet-34 and WideResNet is also observed. For example, with respect to WideResNet-22-10, the relative error reductions achieved by fold-2, where $\alpha = 0$, are 2.56%, and 3.93% on CIFAR-10, and CIFAR-100, and where $\alpha > 0$, the reductions are 1.18%, and 3.89% on CIFAR-10, and CIFAR-100 respectively. The relative error reductions with respect to ResNet-34, achieved by fold-2, where $\alpha = 0$ are 5.26%, and 5.16% on CIFAR-10, and CIFAR-100, and where $\alpha > 0$, the reductions are 5.3%, and 6.59% on CIFAR-10, and CIFAR-100 respectively.

### Model Behavior During Training

The typical behavior of ResNet-18 for fold-1 and fold-4 (in terms of test error rate) across training epochs is shown in Figure 2. It is seen that in the “stable phase” of training, the test error of fold-4 (black curve) continues to decrease whereas the test performance of fold-1 (red curve) fails to further improve. This can be explained by the training loss curve of fold-1 (blue curve), which drops to zero quickly in this phase and provides no training signal for further tuning the network parameters. In contrast, the training curve of fold-4 (purple curve) maintains a relatively high level, allowing the model to keep tuning itself. The relatively higher training loss of fold-4 is due to the much larger space of the amalgamated examples. Even in the stable phase, one expects that the model is still seeing new combinations of images. In other words, we argue that aggregating several examples into a single input can be seen as an implicit form of regularization, preventing the model from over-fitting by limited the number of individual examples.

### Sensitivity to Model Complexity

With fold-$n$ AgrLearn, the output label space becomes $Y^n$. This significantly larger label space seems to suggest that AgrLearn favors a more complex model. In this study, we start with ResNet-18 for fold-2 and investigate the behavior of the model when it becomes more complex. The options we investigate include increasing the model width (by doubling the number of filters per layer) and increasing the model depth (from 18 layers to

---

**Table 1**: Test error rates (%) of ResNet-18 and its AgrLearn counterparts on CIFAR-10, and CIFAR-100

<table>
<thead>
<tr>
<th>Dataset</th>
<th>fold-1 $\alpha = 0$</th>
<th>fold-1 $\alpha = 0.7$</th>
<th>fold-2 $\alpha = 0$</th>
<th>fold-2 $\alpha = 0.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>5.08</td>
<td>4.92</td>
<td>4.89</td>
<td>4.73</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>23.7</td>
<td>23.7</td>
<td>23.03</td>
<td>22.94</td>
</tr>
</tbody>
</table>

**Table 2**: Test error rates (%) of WideResNet-22-10 and its AgrLearn counterparts on CIFAR-10, and CIFAR-100

<table>
<thead>
<tr>
<th>Dataset</th>
<th>fold-1 $\alpha = 0$</th>
<th>fold-1 $\alpha = 0.7$</th>
<th>fold-2 $\alpha = 0$</th>
<th>fold-2 $\alpha = 0.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>4.3</td>
<td>4.23</td>
<td>4.19</td>
<td>4.18</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>21.13</td>
<td>21.1</td>
<td>20.3</td>
<td>20.28</td>
</tr>
</tbody>
</table>

**Table 3**: Test error rates (%) of ResNet-34 and its AgrLearn counterparts on CIFAR-10, and CIFAR-100

<table>
<thead>
<tr>
<th>Dataset</th>
<th>fold-1 $\alpha = 0$</th>
<th>fold-1 $\alpha = 0.7$</th>
<th>fold-2 $\alpha = 0$</th>
<th>fold-2 $\alpha = 0.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>4.94</td>
<td>4.91</td>
<td>4.68</td>
<td>4.65</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>23.86</td>
<td>23.82</td>
<td>22.63</td>
<td>22.25</td>
</tr>
</tbody>
</table>

**Figure 2**: Training loss and test error on CIFAR-10.
34 layers). The performances of these models are given in Table 4.

Table 4 shows that increasing the model width with respect to ResNet-18, and ResNet-34, improves the performance of AgrLearn on both CIFAR-10 and CIFAR-100. For example, doubling the number of filters in ResNet-18 reduces the error rate for fold-2 where $\alpha$ is equal to 0.3 from 4.73% to 4.3% on CIFAR-10, and from 22.94% to 21.78% on CIFAR-100, respectively. It also shows that increasing the model width with respect to ResNet-34 by factor 2, reduces the error rate from 4.65% to 4.45% on CIFAR-10, and from 22.25% to 21.68% on CIFAR-100.

We hypothesize that with AgrLearn, the width of a model plays a critical role. This is because the input dimension in AgrLearn increases significantly and the model is required to extract joint features across individual objects in the amalgamated example.

Moreover, increasing the model depth improves performance. For example, the relative error reductions from ResNet-18 to ResNet-34, where $\alpha$ is equal to 0.3 are 1.7%, and 3% on CIFAR-10, and CIFAR-100 respectively.

Behavior with Respect to Fold Number We also conduct experiments investigating the performance of ResNet-18 with varying fold number $n$. Table 5 suggests that the performance of ResNet-18 is significantly boosted by increasing the number of folds $n$. For example, the relative error reductions achieved by fold-4, where $\alpha$ is equal to 0 are 4.72%, and 5.11% on CIFAR-10, and CIFAR-100, while the relative error reductions achieved by fold-2, are 3.74%, and 2.83% on CIFAR-10, and CIFAR-100. This shows that increasing the number of folds improves the performance of AgrLearn on both CIFAR-10 and CIFAR-100. Moreover, the relative error reductions achieved by fold-4, where $\alpha > 0$ are 4.7%, and 5.8% on CIFAR-10, and CIFAR-100 respectively.

Table 5: Test error rates (%) of ResNet-18 for varying fold numbers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>fold-1</th>
<th>fold-2</th>
<th>fold-3</th>
<th>fold-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0$</td>
<td>$\alpha = 0.3$</td>
<td>$\alpha = 0.7$</td>
<td>$\alpha = 0.7$</td>
<td></td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>5.08</td>
<td>4.73</td>
<td>4.89</td>
<td>4.84</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>23.7</td>
<td>23.03</td>
<td>22.94</td>
<td>22.49</td>
</tr>
</tbody>
</table>

Text Classification

We test AgrLearn with two widely adopted NLP deep-learning architectures, CNN and LSTM (Hochreiter and Schmidhuber 1997), using two benchmark sentence-classification datasets, Movie Review (Pang and Lee 2005) and Subjectivity (Pang and Lee 2004). Movie Review and Subjectivity contain respectively 10,662 and 10,000 sentences, with binary labels. We use 10% of random examples in each dataset for testing and the rest for training, as explained in (Kim 2014a).

For CNN, we adopt CNN-sentence (Kim 2014a) and implement it exactly as (Kim 2014b). For LSTM, we just simply replace the convolution and pooling components in CNN-sentence with standard LSTM units as implemented in (Abadi et al. 2016). The final feature map of CNN and the final state of LSTM are passed to a logistic regression classifier for label prediction. Each sentence enters the models via a learnable, randomly initialized word-embedding dictionary. For CNN, all sentences are zero-padded to the same length.

Table 6: Test error rates (%) of ResNet-18 for varying fold numbers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\alpha = 0$</th>
<th>$\alpha = 0.7$</th>
<th>$\alpha = 0.7$</th>
<th>$\alpha = 0.3$</th>
<th>$\alpha = 0.3$</th>
<th>$\alpha = 0.7$</th>
<th>$\alpha = 0.7$</th>
<th>$\alpha = 0.3$</th>
<th>$\alpha = 0.3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>4.73</td>
<td>22.94</td>
<td>4.89</td>
<td>4.84</td>
<td>4.84</td>
<td>4.73</td>
<td>4.89</td>
<td>4.84</td>
<td>4.73</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>21.78</td>
<td>90.2</td>
<td>92.1</td>
<td>22.94</td>
<td>22.94</td>
<td>22.94</td>
<td>22.94</td>
<td>22.94</td>
<td>22.94</td>
</tr>
</tbody>
</table>

Conclusion

Aggregated Learning, or AgrLearn, is a simple and effective neural network modeling framework, justified information theoretically. It builds on an equivalence between IB learning and IB quantization and exploits the power of vector quantization, which is well known in information theory. We have demonstrated its effectiveness through the significant performance gain it brings to the current art of deep network models.

We believe that the proposal and successful application of AgrLearn in this paper signals the beginning of a promising and rich theme of research. Many interesting questions deserve further investigation. For example, how can we characterize the interaction between model complexity, fold number and sample size in AgrLearn? Additionally, the aggregation of inputs provides additional freedom in the architectural design of the network; how can such freedom be better exploited?
Acknowledgments

This work is supported partly by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC).

References


