Deep Multi-view Information Bottleneck

Qi Wang* Claire Boudreau* Qixing Luo* Pang-Ning Tan* Jiayu Zhou*

Abstract

In many classification problems, the predictions can be enhanced by fusing information from different data views. In particular, when the information from different views complement each other, it is expected that multi-view learning will lead to improved predictive performance. In this paper, we proposed a supervised multi-view learning framework based on the information bottleneck principle to filter out irrelevant and noisy information from multiple views and learn an accurate joint representation. Specifically, our proposed method maximizes the mutual information between the labels and the learned joint representation while minimizing the mutual information between the learned latent representation of each view and the original data representation. As the relationships between different views are often complicated and nonlinear, we employed deep neural networks to learn the latent representation and to disentangle their complex dependencies. However, since the computation of mutual information can be intractable, we employed the variational inference method to efficiently solve the optimization problem. We performed extensive experiments on various synthetic and real-world datasets to demonstrate the effectiveness of the framework.

1 Introduction

The wide availability of heterogeneous data has brought increasing attention to multi-view learning [22, 14, 21, 25], which is a learning paradigm that integrates information from multiple views of the same data objects to improve predictive performance. For example, huge volumes of social media data are generated in various formats every day. On average, there are 510,000 new comments, 293,000 status updates, and 136,000 photos uploaded to Facebook every minute[1]. The images on social media are typically accompanied by text descriptions or categorical tags. For image classification problems, leveraging information from the tags and text descriptions can help boost classification performance as they provide complementary evidence about the true class of the image. As information from each view are potentially noisy, multi-view learning can help improve classification performance by learning the common structure of the different views and reduce the effect of noise.

In addition, each data view may have very distinct properties. For example, text data are usually encoded using a discrete bag-of-words representation, while phonetic data are encoded using a continuous representation. Due to their potential difference in scale, variance, and range of values, a simple concatenation of the features from all views may not necessarily produce the desired performance. To effectively utilize knowledge from multiple views, a variety of multi-view learning methods have been proposed in recent years. Canonical correlation analysis (CCA) [5, 11, 16] and its variations such as kernel canonical correlation analysis (KCCA) [4], deep canonical correlation analysis (DCCA) [3] and deep canonically correlated auto-encoders (DCCAE) [26] have been developed for two-view learning problems. These methods would project the original features from the two views into a shared feature space such that the canonical correlation between the views in the new space is maximized. The projected features from each view can then be used in predictive modeling tasks. Nevertheless, a major drawback of CCA-based methods is that it decouples feature learning from the predictive modeling step, and thus, unable to utilize the label information when learning the feature representation.

More recently, a new supervised learning method [29] based on the information bottleneck principle [23] has gained increasing attention due to its ability to find a concise representation of the features, taking into account the trade-off between performance and complexity from an information theory perspective. However, the main drawback of this method is that it employs a linear projection to bridge the representation of each view. As the relationship between different views are often complicated, a simple linear projection would constrain the type of information that can be fused from the different views.

Deep learning has been successfully used to learn abstract representation from the raw input data [13]. DCCA and DCCAE are two examples of successful methods using deep neural networks to extract features from each view and learn their joint representation.

*Michigan State University

[1] https://zephoria.com/top-15 -valuable-facebook-statistics/

These methods have demonstrated better performance compared to traditional linear CCA. However, adopting deep neural networks to information bottleneck based multi-view learning formulation remains a challenging problem. For the information bottleneck approach, the shared information between different representations are measured in terms of their mutual information. Computing mutual information requires estimation of the posterior distribution, which is computationally intractable when the model is complicated.

In this paper, we proposed a deep multi-view information bottleneck method to fuse knowledge from multiple views to improve predictive performance. The proposed framework consists of two parts. The first part is to extract concise latent representation from each view while the second part fuses the latent representations to learn the joint representation of all views. The proposed deep multi-view learning framework adopts the information bottleneck principle to supervise the learning by finding the best representation that balances model complexity and accuracy. The framework also employs a variational inference approach [12, 2] to overcome the challenge of computing mutual information efficiently. The variational inference approach provides an approximate solution to the original optimization problem by maximizing the variational lower bound of the target objective function. Since the variational bound can be easily optimized by standard gradient descent methods, the problem becomes computationally tractable.

2 Related work

2.1 Multi-view learning CCA[9] and its variations are widely-used multi-view learning methods. CCA identifies the relationship between two sets of variables by maximizing the correlation between the weighted linear combination of one set of variables and that of the other set of variables. It can be viewed as projecting the original two sets of variables to a low-dimensional subspace, such that the correlation between the two set of variables is maximized in the new subspace. Therefore, it is much easier to analyze variable dependence in the learned subspace than in the original spaces. To deal with the nonlinear and complex relationship between the two views, KCCA [1] is proposed. KCCA first projects data from the two sources to a Hilbert space, and then maximizes the correlation between the projected data points. This method is not easy to scale when the size of the training data is large. Moreover, the representations learned by kernel CCA is dependent on the kernels used. To solve those issues, DCCA [3, 27] uses deep neural networks to learn two deep nonlinear mappings which map the two sources to new representations such that the canonical correlation of the new representations is maximized. DCCAE [26] is an extension of DCCA. Besides having a similar structure as DCCA, this model also contains two autoencoders to reconstruct the learned views. It optimizes an objective that maximizes the canonical correlation between the projected representations and minimizes the reconstruction error of the autoencoders simultaneously. This method offers a trade-off between the information captured in the input-feature mapping within each source on one hand, and the information in the feature-feature relationship across sources on the other hand [26].

2.2 Information bottleneck Information bottleneck [23] is an approach based on information theory. It formalizes the intuitive ideas about information to provide a quantitative measure of "meaningful" and "relevant" [23]. It provides a tradeoff between accuracy and complexity. This method has been widely used in clustering [19, 24, 8], ranking [10] and classification [18]. Exact solution does not exist if the latent representation is learned by deep neural networks. In [2], the authors applied information bottleneck to single-view learning and proposed to use the variational method to optimize it. Instead of directly solving the optimization problem of information bottleneck, the authors first calculated a lower bound of the original target. Then the lower bound was maximized to push the results closer to the optimal solution to the original optimization problem. The distributions of the posteriors were learned by the neural networks. The method also utilized the reparameterization trick for efficient training. Information bottleneck is also used in multi-view learning. In [29], the authors proposed to use information bottleneck to learn a joint latent representation. The joint latent representation was a combination of the linear projection of all the views. The projection matrices were learned by the information bottleneck approach. Although the approach achieves decent results, it is limited to linear projection. Therefore, we propose a nonlinear deep version of multi-view information bottleneck to overcome this limitation.

3 Methodology

3.1 Information Bottleneck Method Information bottleneck is an information-based approach to find the best tradeoff between the accuracy and complexity. Given data X with labels Y, information bottleneck aims to find a concise and accurate latent representation of X. Denote the latent representation as Z. Information bottleneck solves the following optimization

problem:

$$\max_{Z} I(Y,Z) \quad \text{s.t.} I(X,Z) \leq \gamma, \tag{3.1}$$

where $I(Y,Z)$ is the mutual information between Y and Z whereas $I(X,Z)$ is the mutual information between X and Z. The mutual information between any two random variables X and Y is defined as:

$$I(X,Y) = \int_Y \int_X p(x,y) \log(\frac{p(x,y)}{p(x)p(y)}) dxdy,$$

where $p(x,y)$ is the joint probability density function of X and Y while $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y.

Eq. (3.1) maximizes the mutual information between Y and Z to make sure the learned Z contains information about Y as much as possible. If there is no constraint on Z, the solution would be $Z = X$. But in most cases, X contains noise or other irrelevant information to Y. Therefore, a constraint must be applied to Z to ensure that the learned Z provides a concise representation that contains less noise and irrelevant information compared with X. This constraint reduces the model complexity and improves the model's generalization ability. Eq. (3.1) can also be relaxed to the following formulation:

$$\max_{Z} I(Y,Z) - \alpha I(X,Z),$$

where α is a regularization parameter to control the tradeoff between $I(Y,Z)$ and $I(X,Z)$.

3.2 Deep multi-view information bottleneck. In multi-view learning, information bottleneck can be used to learn the joint discriminative representation as it can remove the irrelevant information and noise of each view. Since for real-world data, the relation between multiple views are likely to be nonlinear and complex, in this paper, we propose a deep multi-view information bottleneck method to map the original representation to a nonlinear representation that can make subjects easier to be separated.

Given two views X_1, X_2 and the class labels Y, the proposed method aims to learn a joint representation Z to fuse the information from all views. The model contains two parts. The first part is to learn the hidden representations from all the views. Each view has one hidden representation. This part is to remove the noise and irrelevant information from X_1 and X_2 as much as possible to make sure the learned representations are very concise. We use Z_1 and Z_2 to denote the hidden representations for X_1 and X_2, respectively. The second part is to fuse the hidden representations using a neural network as

$$Z = f_\theta(Z_1, Z_2), \tag{3.2}$$

where f denote the network and θ the network parameter. This part is to transfer knowledge from all views and learn a joint representation Z. These two parts are learned jointly by the information bottleneck as

$$\begin{aligned} \max_{Z,Z_1,Z_2} & I(Y,Z) - \alpha I(X_1,Z_1) - \beta I(X_2,Z_2), \\ \text{s.t.} \quad & Z = f_\theta(Z_1,Z_2), \end{aligned} \tag{3.3}$$

where α and β are regularization parameters. The first term is to maximize the mutual information between the joint representation and the label Y to make sure the learned joint representation are discriminative according to the class labels. The last two terms are to minimize the mutual information between the latent representation of each view and its original data representation. These two terms reduce the model complexity to make the model more generalizable, since they can filter out the irrelevant and noisy information.

3.3 Optimization The major challenge of solving Eq. (3.3) is that the mutual information terms are computationally intractable. Recently, variational methods [12, 2, 7] are widely used to deal with such problems. Variation methods maximize the variational lower bounds of the objective functions instead of directly maximizing them. These methods use some known distributions to approximate the intractable distributions, and provide lower bounds of the original objective functions. By increasing the lower bounds, we can obtain approximate solutions to the original objective functions. To obtain the variational lower bound of Eq. (3.3), we first need to find the joint probability density function of all the variables including the latent variables. Using Bayes' rule, the joint probability density function of X_1, X_2, Z_1, Z_2, Y, Z can be expressed as

$$\begin{aligned} p(x_1,x_2,z_1,z_2,y,z) = & p(z|z_1,z_2,x_1,x_2,y) \\ & p(z_1|z_2,x_1,x_2,y) \\ & p(z_2|x_1,x_2,y)p(x_1,x_2,y). \end{aligned} \tag{3.4}$$

Since Z_1 is learnt from X_1, we thus assume given X_1, Z_1 is independent of Z_2, X_2, Y. Similarly, we assume given X_2, Z_2 is independent of X_1, Y, and given Z_1, Z_2, Z is independent of X_1, X_2, Y. Therefore, we have the following equalities:

$$\begin{aligned} & p(z_1|z_2,x_1,x_2,y) = p(z_1|x_1), \\ & p(z_2|x_1,x_2,y) = p(z_2|x_2), \\ & p(z|z_1,z_2,x_1,x_2,y) = p(z|z_1,z_2). \end{aligned}$$

Using these assumptions, the joint probability density function can be simplified as

$$\begin{aligned} p(x_1, x_2, z_1, z_2, y, z) =& p(z|z_1, z_2)p(z_1|x_1) \\ & p(z_2|x_2)p(x_1, x_2, y). \end{aligned} \tag{3.5}$$

First, let us start with $I(Y, Z)$. Since $p(y|z)$ is intractable, we use a distribution $q(y|z)$, which will be learned from the network, to approximate $p(y|z)$. The KL-divergence between $p(y|z)$ and $q(y|z)$ is always non-negative. Therefore, we have

$$\begin{aligned} & KL[p(y|z), q(y|z)] \geq 0 \\ \Rightarrow & \int dydz\ p(y, z) \log(p(y|z)) \geq \int dydz\ p(y, z) \log(q(y|z)). \end{aligned} \tag{3.6}$$

The mutual information between Y and Z is

$$\begin{aligned} I(Z, Y) &= \int dydz p(y, z) \log \frac{p(y, z)}{p(y)p(z)} \\ &= \int dydz p(y, z) \log \frac{p(y|z)}{p(y)}. \end{aligned}$$

Using Eq. (3.6), we have

$$\begin{aligned} I(Y, Z) &\geq \int dydz\ p(y, z) \log \frac{q(y|z)}{p(y)} \\ &= \int dydz\ p(y, z) \log q(y|z) - \int dy\ p(y) \log p(y). \end{aligned}$$

Since $-\int dy\ p(y) \log p(y)$ is the entropy of the labels, and this term have no effect on the optimization, we can directly drop it. Therefore, the variation lower bound of $I(Y, Z)$ is

$$\begin{aligned} & I(Y, Z) \geq \int dydz\ p(y, z) \log q(y|z) \\ & = \int dydzdx_1dx_2dz_1dz_2\ p(x_1, x_2, z_1, z_2, y, z) \log q(y|z). \end{aligned}$$

By using the joint probability density function in Eq. (3.5), the variational lower bound of the mutual information between Z and Y can be written as

$$\begin{aligned} & I(Y, Z) \geq \int dx_1dx_2dy\ p(x_1, x_2, y) \\ & \int dzdz_1dz_2p(z|z_1, z_2)p(z_1|x_1)p(z_2|x_2) \log q(y|z). \end{aligned} \tag{3.7}$$

Next, we need to find the upper bound of $I(X_1, Z_1)$. Since $p(z_1)$ is intractable, we use $r_1(z_1)$ to approximate $p(z_1)$. Similarly, we use the property of the KL-divergence between $p(z_1)$ and $r_1(z_1)$.

$$\begin{aligned} & KL[p(z_1), r_1(z_1)] \geq 0 \\ \Rightarrow & \int dzp(z_1) \log p(z_1) \geq \int dzp(z_1) \log r(z_1). \end{aligned}$$

Therefore, the mutual information between Z_1 and X_1 is

$$\begin{aligned} I(Z_1, X_1) &= \int dz_1dx_1p(x_1, z_1) \log \frac{p(z_1|x_1)}{p(z_1)} \\ &\leq \int dz_1dx_1\ p(x_1, z_1) \log \frac{p(z_1|x_1)}{r_1(z_1)} \\ &= \int dx_1dx_2dydz_1\ p(x_1, x_2, z_1, y) \log \frac{p(z_1|x_1)}{r_1(z_1)}. \end{aligned}$$

Using the assumption that given x_1, z_1 is independent of all other variables, we have

$$\begin{aligned} & I(Z_1, X_1) \leq \int dx_1dx_2dyp(x_1, x_2, y) \\ & \int dz_1\ p(z_1|x_1) \log \frac{p(z_1|x_1)}{r_1(z_1)}. \end{aligned} \tag{3.8}$$

Similarly, for $I(Z_2, X_2)$, we have

$$\begin{aligned} & I(Z_2, X_2) \leq \int dx_1dx_2dyp(x_1, x_2, y) \\ & \int dz_2\ p(z_2|x_2) \log \frac{p(z_2|x_2)}{r_2(z_2)}. \end{aligned} \tag{3.9}$$

With Eq. (3.7), Eq. (3.8) and Eq. (3.9), the final variational lower bound is:

$$\begin{aligned} & I(Y, Z) - \alpha I(X_1, Z_1) - \beta I(X_2, Z_2) \\ & \geq \int dx_1dx_2dy\ p(x_1, x_2, y) \\ & (\int dzdz_1dz_2p(z|z_1, z_2)p(z_1|x_1)p(z_2|x_2) \log q(y|z) \\ & -\alpha \int dz_1\ p(z_1|x_1) \log \frac{p(z_1|x_1)}{r_1(z_1)} \\ & - \beta \int dz_2\ p(z_2|x_2) \log \frac{p(z_2|x_2)}{r_2(z_2)}). \end{aligned}$$

The integral over x_1, x_2 and y can be approximated by Monte Carlo sampling [17]. Therefore,

$$\begin{aligned} & I(Y, Z) - \alpha I(X_1, Z_1) - \beta I(X_2, Z_2) \\ & \geq \frac{1}{N} \sum_i^N \{ \int dzdz_1dz_2p(z|z_1, z_2)p(z_1|x_1)p(z_2|x_2) \\ & \log q(y|z) - \alpha \int dz_1p(z_1|x_1) \log \frac{p(z_1|x_1)}{r_1(z_1)} \\ & - \beta \int dz_2p(z_2|x_2) \log \frac{p(z_2|x_2)}{r_2(z_2)} \}, \end{aligned}$$

where N is the sample size of the total sampled data. Next, we assume $p(z_1|x_1), p(z_2|x_2)$ and $p(z|z_1, z_2)$ are Gaussian. The means and variances of the Gaussian distributions are all learned from deep neural networks, i.e.,

$$\begin{aligned} p(z_1|x_1) &= \mathcal{N}(\mu_1(x_1;\phi_1), \Sigma_1(x_1;\phi_1)), \\ p(z_2|x_2) &= \mathcal{N}(\mu_2(x_2;\phi_2), \Sigma_2(x_2;\phi_2)), \\ p(z|z_1, z_2) &= \mathcal{N}(\mu(z_1, z_2;\theta), \Sigma(z_1, z_2, ;\theta)), \end{aligned}$$

where μ_1, μ_2, μ and $\Sigma_1, \Sigma_2, \Sigma$ are the networks to learn the means and variances for $p(z_1|x_1), p(z_2|x_2)$ and $p(z|z_1, z_2)$. ϕ_1, ϕ_2 and θ are network parameters for the networks to learn $p(z_1|x_1), p(z_2|x_2)$ and $p(z|z_1, z_2)$, respectively. Since z_1, z_2 and z are all random variables, backpropagation through those random variables may cause problems. Therefore, we use the reparameterization trick here, i.e.,

$$\begin{aligned} z_1 &= \mu(x_1;\phi_1) + \Sigma(x_1;\phi_1)\epsilon_1, \\ z_2 &= \mu(x_1;\phi_1) + \Sigma(x_1;\phi_1)\epsilon_2, \\ z &= \mu(z_1, z_2;\theta) + \Sigma(z_1, z_2, ;\theta)\epsilon, \end{aligned}$$

where $\epsilon, \epsilon_1, \epsilon_2 \sim \mathcal{N}(0, I)$. By using this reparameterization trick, randomness is transfered to $\epsilon, \epsilon_1, \epsilon_2$, which do not affect the backpropagation. Therefore, the final loss is

$$\begin{aligned} \max \frac{1}{N}\sum^{N}\{&\mathbb{E}_\epsilon\mathbb{E}_{\epsilon_1}\mathbb{E}_{\epsilon_2}\log q(y|z) - \alpha\mathbb{E}_{\epsilon_1}\log\frac{p(z_1|x_1)}{r_1(z_1)} \\ &-\beta\mathbb{E}_{\epsilon_2}\log\frac{p(z_2|x_2)}{r_2(z_2)}\}. \end{aligned} \tag{3.10}$$

Three Monte Carlo sampling procedures are used are used here to approximate the integrals. $p(z_1|x_1), p(z_2|x_2)$ are all learned from neural networks. Note that the first term in Eq. (3.10) is the cross-entropy between y and z. Thus, we can use a deep neural network with a softmax layer as output to calculate the class probabilities and the cross-entropy loss.

3.4 Generalize to multiple views The proposed deep multi-view information bottleneck framework can be easily generalized to settings with more than 2 views by adding corresponding information constraint terms. Given v views $\{X_1, X_2, ..., X_v\}$, the formulation of the proposed method is

$$\max_{Z, Z_1, Z_2, ... Z_v} I(Y, Z) - \sum_{i}^{v} \alpha_i I(X_i, Z_i), \tag{3.11}$$

where Z_i is the latent representation of X_1. α_i is the regularization parameter to regularize the mutual information between X_i and Z_i. Following the procedures in Section 3.3, the final loss for Eq. (3.11) is

$$\max \frac{1}{N}\sum^{N}\{\mathbb{E}_\epsilon\mathbb{E}_{\epsilon_1}\mathbb{E}_{\epsilon_2}...\mathbb{E}_{\epsilon_v}\log q(y|z) - \alpha_i\mathbb{E}_{\epsilon_i}\log\frac{p(z_i|x_i)}{r_i(z_i)}\}, \tag{3.12}$$

where $\epsilon, \epsilon_1, \epsilon_2, ...\epsilon_v \sim \mathcal{N}(0, I)$. $r_i(z_i)$ are assumed as $r_i(z_i) \sim \mathcal{N}(0, I)$. Each $p(z_i|x_i)$ are Gaussian with μ and Σ leaned from a deep neural network.

4 Experiment

In this section, we present the experimental results on synthetic and real-world datasets. The baseline algorithms used for comparison include linear CCA [5], DCCA [3], DCCAE [26], and the fully-connected deep neural network (DNN), which uses two fully-connected neural networks to directly extract latent representations Z_1, Z_2 and then uses a deep neural network to fuse Z_1 and Z_2 to make prediction. One intuitive baseline for multi-view learning is to concatenate the features from all the views and treat the concatenated features as one view. In our experiments, we use single-view information bottleneck [2] as the model for this baseline and denote this baseline as *singleview12*. We also provide the results of single-view learning using information bottleneck [2] for each view, and use *singleview1, singleview2* to denote the baselines using the first view and the second view. We denote the proposed method as *deep IB*.

4.1 Synthetic datasets . The data are synthesized in the following way. First, we sample $2n$ points from two Gaussian distributions, i.e., $\mathcal{N}(0.5\mathbf{e}, \mathbf{I})$ and $\mathcal{N}(-0.5\mathbf{e}, \mathbf{I})$ to form Z. Samples from each distribution form one class. Each class has n data points. Then, we directly use Z to generate $X1$ and $X2$ by setting $X_i = f(D) + \text{noise}$, where $D = [Z, \text{extra-features}]$ with $i \in \{1, 2\}$ and f is a nonlinear function. Extra-features here are used to distort the classification and are sampled from another two Gaussian distributions, i.e., $\mathcal{N}(\mathbf{e}, \mathbf{I})$ and $\mathcal{N}(-\mathbf{e}, \mathbf{I})$. We sample m data from the first Gaussian distribution and $2n - m$ samples from the second Gaussian distribution. We concatenate the extra-features to the useful features to distort the classification. Extra-features are illustrated in Figure 1. In Figure 1, the row represents the samples and the column represents the features. Extra-features have different class property compared with the useful features. In all the synthetic data experiments, we set $m = 2n/3$ for the first view and $m = 2n/6$ for the second view. Extra-features widely exist in multi-view learning scenario. For example, when we collect all the genetic data from people with a gene-related disease and healthy people,

the genetic data contain not only information to classify the disease, but also gender information. The features that describe the gender information are extra-features. The effect of those features needs to be eliminated in the classification process. The noise is sampled from $\mathcal{N}(0, t*I)$, where t denotes the noise level and is changed in the experiments to test the algorithms' ability to eliminate the effect from noise. f is $tanh(tanh(D))+0.1$ for the first view and $sigmoid(D) - 0.5$ for the second view.

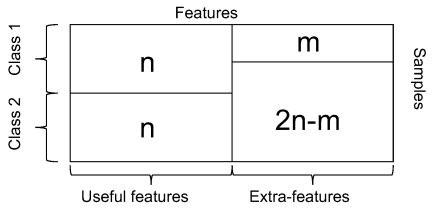

Figure 1: Illustration of extra-features in the synthetic data experiments. Extra-features have different class property with the useful features.

4.1.1 Setting 1 . In the first setting, we change the noise level t and compare our model with other baselines. t is the relative noise level which is calculated as $t = a \times \max(abs(X_i))$ for each view, where abs means the absolute value. We set a to be $\{0.2 : 0.2 : 1.2\}$. The sample size per class is set to be 500. The useful feature dimension is 20, and the extra-feature dimension is 5. α and β are tuned in $[1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2]$. For the subnetworks that extract features from X_1 and X_2 for all the deep models including DCCA, DCCAE, we tune the number of layers in $[3, 4, 5]$ and the node number for each layer is tuned in $[256, 512, 1024]$. The activation function is ReLU. For the subnetworks that fuse the extracted features from all views, we tune the number of layers in $[1, 2, 3]$ and the node number is tuned in $[128, 256, 512]$. The activation function is ReLU. For all the experiments, we use 80% data as training and the rest as testing and repeat the experiments for 5 times. We report the average errors for all methods in Table 1. From Table 1, we see when noise increases, the performance becomes worse for all the methods. Single-view methods are all worse than supervised multi-view methods. Simple concatenation of two views is not as good as deep IB. Compared with CCA-based method, we see supervision information improves the performance a lot. DNN is a challenging baseline as shown in the results. DNN has a similar network structure with deep IB. The difference between DNN and deep IB is that DNN tries to extract latent features by directly maximizing the cross-entropy between the outputs of the network and labels, while deep IB not only maximizes the cross-entropy between the outputs of the network and labels, but also constrains the model complexity by reducing the information between Z_1 and X_1, and between Z_2 and X_2. Therefore, the generalization performance of deep IB is better than DNN.

4.1.2 Setting 2 . In the second setting, we vary the sample size per class to see how the performance changes. The noise level a is set to be 1.0. The useful features dimension is 20, and the extra-feature dimension is set to be 5. The models are tuned in the same way as the first setting. We report the errors for all methods in Table 2. From the table, we see increasing the sample size improves the performance for all methods. We observe some similar patterns with that of Setting 1. For example, deep IB results are better than all other single-view methods results. CCA-based methods are not as good as supervised methods. One specific observation is that when the sample size is large enough, i.e., greater than 1100, DNN's performance is better than deep IB. That is because deep IB has the assumption to reduce the model complexity. When the sample size is large enough, deep IB underfits the data, while DNN has no assumption. Therefore, when the sample size is large enough, direct using DNN delivers the highest accuracy.

4.1.3 Setting 3 . In the third setting, we change the extra-feature dimension to see how the extra-feature dimension affects the results. In this setting, the sample size per class is set to be 500. The noise level a is 1.0. The useful feature dimension is fixed as 20. The errors are shown in Table 3. From Table 3, we see deep IB outperforms all the other methods with any extra-feature dimension. When the extra-feature dimension increases, the data contain more irrelevant information, which makes the classification to be distorted. We see when the extra-feature dimension increases, the error of DNN, CCA-based methods increase a lot. However, for IB based methods including the single-view baselines, the errors are stable when the extra-feature dimension is larger or equal to 25. From the results, we conclude that IB-based methods are more robust to extra-features.

4.2 A case study: reservoir detection . In this case study, we compare all the models on a reservoir detection dataset. Reservoirs for this dataset are sampled with ArcMap 10.3.1 by joining dam features from the US Army Corps National Inventory of Dams with lake polygons over 4 hectares from the LAGOS database [20]. For comparison, we also select a proportional number of natural lakes from the major river watershed that each reservoir is located in. The sample size for this dataset

Noise level (a)	0.2	0.4	0.6	0.8	1.0	1.2
singleview1	0.070 ± 0.016	0.111 ± 0.020	0.135 ± 0.025	0.166 ± 0.025	0.192 ± 0.030	0.206 ± 0.040
singleview2	0.132 ± 0.025	0.205 ± 0.040	0.253 ± 0.025	0.283 ± 0.031	0.313 ± 0.032	0.338 ± 0.035
singleview12	0.064 ± 0.012	0.083 ± 0.012	0.125 ± 0.011	0.154 ± 0.031	0.164 ± 0.023	0.181 ± 0.037
linear CCA	0.064 ± 0.027	0.109 ± 0.023	0.143 ± 0.035	0.165 ± 0.038	0.194 ± 0.041	0.209 ± 0.043
DCCA	0.065 ± 0.024	0.096 ± 0.023	0.132 ± 0.029	0.154 ± 0.033	0.173 ± 0.034	0.198 ± 0.043
DCCAE	0.075 ± 0.017	0.098 ± 0.008	0.139 ± 0.044	0.165 ± 0.026	0.182 ± 0.028	0.203 ± 0.041
DNN	0.061 ± 0.004	0.094 ± 0.012	0.128 ± 0.025	0.156 ± 0.024	0.164 ± 0.037	0.191 ± 0.031
deep IB	0.059 ± 0.016	0.073 ± 0.011	0.122 ± 0.019	0.139 ± 0.017	0.158 ± 0.017	0.171 ± 0.026

Table 1: Average errors of all methods under different noise levels.

Sample per class	300	500	700	900	1100	1300
singleview1	0.178 ± 0.035	0.192 ± 0.030	0.175 ± 0.022	0.179 ± 0.015	0.192 ± 0.014	0.183 ± 0.005
singleview2	0.333 ± 0.043	0.313 ± 0.032	0.319 ± 0.032	0.326 ± 0.021	0.317 ± 0.010	0.316 ± 0.024
singleview12	0.230 ± 0.080	0.173 ± 0.023	0.164 ± 0.023	0.174 ± 0.021	0.180 ± 0.008	0.185 ± 0.011
linear CCA	0.163 ± 0.030	0.194 ± 0.041	0.166 ± 0.038	0.159 ± 0.011	0.173 ± 0.011	0.170 ± 0.023
DCCA	0.165 ± 0.013	0.173 ± 0.033	0.158 ± 0.026	0.151 ± 0.018	0.165 ± 0.013	0.155 ± 0.013
DCCAE	0.169 ± 0.008	0.182 ± 0.028	0.154 ± 0.033	0.154 ± 0.003	0.178 ± 0.017	0.160 ± 0.008
DNN	0.173 ± 0.024	0.164 ± 0.032	0.161 ± 0.032	0.146 ± 0.016	0.143 ± 0.004	0.139 ± 0.008
deep IB	0.162 ± 0.015	0.158 ± 0.017	0.143 ± 0.021	0.139 ± 0.009	0.143 ± 0.007	0.143 ± 0.006

Table 2: Average errors of all methods under different sample sizes.

Extra-feature dim	5	15	25	35	45	55
singleview1	0.192 ± 0.030	0.194 ± 0.036	0.199 ± 0.020	0.198 ± 0.042	0.194 ± 0.019	0.193 ± 0.016
singleview2	0.313 ± 0.032	0.333 ± 0.024	0.342 ± 0.019	0.327 ± 0.020	0.334 ± 0.026	0.332 ± 0.021
singleview12	0.164 ± 0.023	0.181 ± 0.027	0.194 ± 0.024	0.189 ± 0.041	0.191 ± 0.026	0.189 ± 0.017
linear CCA	0.194 ± 0.041	0.192 ± 0.038	0.225 ± 0.030	0.205 ± 0.047	0.255 ± 0.027	0.286 ± 0.012
DCCA	0.173 ± 0.033	0.179 ± 0.040	0.187 ± 0.016	0.181 ± 0.018	0.183 ± 0.026	0.201 ± 0.040
DCCAE	0.182 ± 0.028	0.185 ± 0.037	0.195 ± 0.020	0.193 ± 0.023	0.215 ± 0.026	0.221 ± 0.032
DNN	0.164 ± 0.037	0.197 ± 0.020	0.201 ± 0.022	0.216 ± 0.025	0.219 ± 0.015	0.225 ± 0.032
deep IB	0.158 ± 0.017	0.174 ± 0.053	0.175 ± 0.013	0.174 ± 0.035	0.173 ± 0.019	0.176 ± 0.023

Table 3: Average errors of all methods under different extra-feature dimensions.

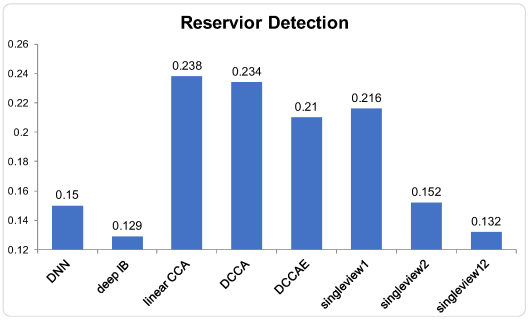

Figure 2: Average error for reservoir detection task.

is 1327 with 660 natural lakes and 667 reservoirs. There are two views available in this dataset. The first one is the boundary of the lakes. Boundary features of each lake and reservoir are exported using ArcMap. Each boundary file is a 224×224 image. To deal with the boundary data, we first use VGG16 to extract features. We use the last fully-connected layer's output as the features. The dimension is 4096. Since the sample size is not large, we use PCA to reduce the feature dimension by keeping the top 1% singular values. The reduced feature dimension is 75. The second view is the features extracted from Google Earth. The features include the area of the lakes, shape length, classes of the general types of parent material of soil on the surface, classes of landforms, NED-derived mTPI ranging from negative (valleys) values to positive (ridges) values, NED-derived CHILI index ranging from 0 (very cool) to 225 (very warm). In total, there are 21 features. We split the data into training and testing as the synthetic data experiments and report the average error in Figure 2. From the figure, we see deep IB outperforms all other methods. In Figure 3, we also qualitatively show the final joint representations learned by all methods with t-Distributed Stochastic Neighbor Embedding [15]. The

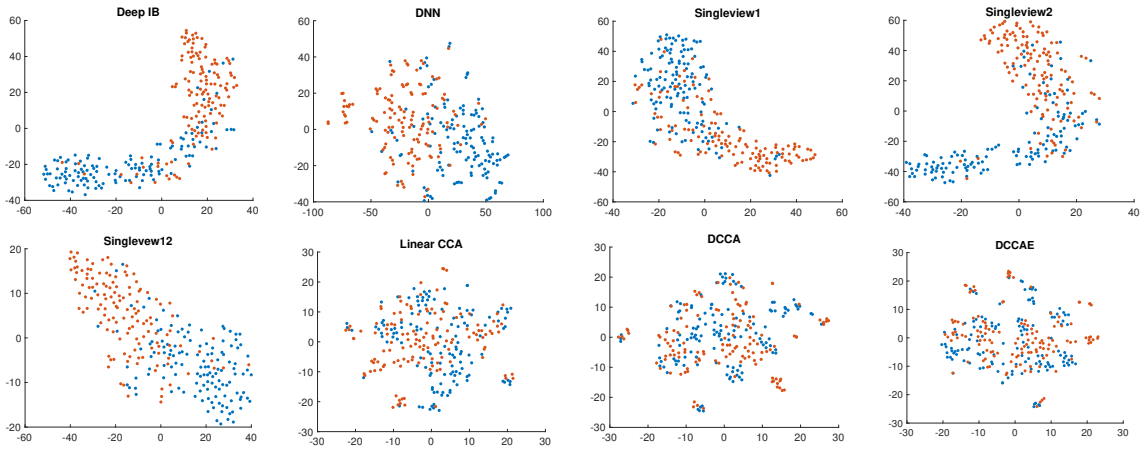

Figure 3: t-Distributed Stochastic Neighbor Embedding for the final joint representations. Blue dots are natural lakes and red dots are reservoirs.

final joint representation is the output of the layer that is connected with the final linear classifier. For example, for deep IB, DNN and the single-view methods, the final joint representations are the outputs of the layer before the last layer. For the CCA-based methods, the final learned representations are the projected representations from the first view. In Figure 3, blue dots are natural lakes and red dots are reservoirs. We see that the separation qualities are consistent with the performance in Figure 4.2.

4.2.1 Other benchmark datasets In this section, we report the performance on three benchmark datasets. The datasets we used are

- Wisconsin X-Ray Mircro-Beam (XRMB) [26, 28]: the first view is 273D acoustic inputs, the second view is 112D articulatory inputs [2].
- MNIST [26]: two views are generated from MNIST datasets. The first view is a random rotation of the original images. The second view is generated by adding noise to the original images. Both views have 784 features [3].
- Wiki [6]: the dataset contains 2866 images-text pairs. Each image is represented by 128D inputs and text is represented by 10D inputs. There are 10 classes in total.

The average errors are shown in Table 4. From the table, we see for all the benchmark datasets, the proposed method performs the best among all the methods, which verifies the effectiveness of the proposed method.

Dataset	XRMB	MNIST	Wiki
singleview1	0.185 ± 0.003	0.075 ± 0.006	0.449 ± 0.024
singleview2	0.271 ± 0.003	0.160 ± 0.012	0.337 ± 0.018
singleview12	0.179 ± 0.006	0.057 ± 0.009	0.336 ± 0.006
linear CCA	0.358 ± 0.004	0.235 ± 0.006	0.741 ± 0.017
DCCA	0.231 ± 0.006	0.187 ± 0.012	0.478 ± 0.049
DCCAE	0.226 ± 0.005	0.170 ± 0.020	0.499 ± 0.037
DNN	0.168 ± 0.006	0.060 ± 0.056	0.311 ± 0.017
deep IB	0.161 ± 0.005	0.056 ± 0.002	0.298 ± 0.005

Table 4: Average errors for three benchmark datasets.

5 Conclusion

In this paper, we presented a novel multi-view learning model based on information bottleneck. The model encouraged the latent representation keeping target information as much as possible while containing the information of original features as little as possible to reduce the model complexity. To learn the complicated relationship between views and within views, we used a deep neural network to learn the latent representation. Since the mutual information terms were intractable, we maximized the lower bound of the formulation instead of directly maximizing it. We demonstrated experiments on various synthetic and real-world datasets to show the effectiveness of the proposed method.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under Grant IIS-1749940 (JZ), IIS-1615597 (JZ), III-1615612 (PT), EF-1638679 (PT), and Office of Naval Research N00014-17-1-2265.

[2] We did not use the whole dataset since some baselines are quite slow. We randomly sampled 50000 data points for training and sampled 6000 points for testing from the first 10 classes.

[3] We did not use the whole dataset since some baselines are very time-consuming. We sampled 5000 data for training and 1000 for testing.

References

[1] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[3] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning*, pages 1247–1255, 2013.

[4] Raman Arora and Karen Livescu. Kernel cca for multi-view learning of acoustic features using articulatory measurements. In *Symposium on Machine Learning in Speech and Language Processing*, 2012.

[5] Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. Multi-view clustering via canonical correlation analysis. In *ICML*, pages 129–136. ACM, 2009.

[6] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2075–2082, 2014.

[7] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[8] Shiri Gordon, Hayit Greenspan, and Jacob Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *null*, page 370. IEEE, 2003.

[9] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[10] Winston H Hsu, Lyndon S Kennedy, and Shih-Fu Chang. Video search reranking via information bottleneck principle. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 35–44. ACM, 2006.

[11] Sham M Kakade and Dean P Foster. Multi-view regression via canonical correlation analysis. In *International Conference on Computational Learning Theory*, pages 82–96. Springer, 2007.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[14] Weifeng Liu and Dacheng Tao. Multiview hessian regularization for image annotation. *IEEE Transactions on Image Processing*, 22(7):2676–2687, 2013.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[16] Jan Rupnik and John Shawe-Taylor. Multi-view canonical correlation analysis. In *Conference on Data Mining and Data Warehouses (SiKDD 2010)*.

[17] Alexander Shapiro. Monte carlo sampling methods. *Handbooks in operations research and management science*, 10:353–425, 2003.

[18] Noam Slonim, Rachel Somerville, Naftali Tishby, and Ofer Lahav. Objective classification of galaxy spectra using the information bottleneck method. *Monthly Notices of the Royal Astronomical Society*, 323(2):270–284, 2001.

[19] Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215. ACM, 2000.

[20] Patricia A Soranno, Linda C Bacon, Michael Beauchene, Karen E Bednar, Edward G Bissell, Claire K Boudreau, Marvin G Boyer, Mary T Bremigan, Stephen R Carpenter, Jamie W Carr, et al. Lagosne: a multi-scaled geospatial and temporal database of lake ecological context and water quality for thousands of us lakes. *GigaScience*, 6(12):1–22, 2017.

[21] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[22] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[23] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[24] Naftali Tishby and Noam Slonim. Data clustering by markovian relaxation and the information bottleneck method. In *Advances in neural information processing systems*, pages 640–646, 2001.

[25] Qi Wang, Mengying Sun, Liang Zhan, Paul Thompson, Shuiwang Ji, and Jiayu Zhou. Multi-modality disease modeling via collective deep matrix factorization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1155–1164. ACM, 2017.

[26] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092, 2015.

[27] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A Bilmes. Unsupervised learning of acoustic features via deep canonical correlation analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4590–4594. IEEE, 2015.

[28] John Westbury, Paul Milenkovic, Gary Weismer, and Raymond Kent. X-ray microbeam speech production database. *The Journal of the Acoustical Society of America*, 88(S1):S56–S56, 1990.

[29] Chang Xu, Dacheng Tao, and Chao Xu. Large-margin multi-viewinformation bottleneck. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1559–1572, 2014.