Deep Embedded Complementary and Interactive Information for Multi-view Classification

Jinglin Xu[1 †], **Wenbin Li**[2 †], **Xinwang Liu**[3], **Dingwen Zhang**[1,4], **Ji Liu**[5], **Junwei Han**[1*]
[1]Northwestern Polytechnical University, China [2]Nanjing University, China
[3]National University of Defense Technology, China [4]Xidian University, China [5]Kwai Inc.

Abstract

Multi-view classification optimally integrates various features from different views to improve classification tasks. Though most of the existing works demonstrate promising performance in various computer vision applications, we observe that they can be further improved by sufficiently utilizing complementary view-specific information, deep interactive information between different views, and the strategy of fusing various views. In this work, we propose a novel multi-view learning framework that seamlessly embeds various view-specific information and deep interactive information and introduces a novel multi-view fusion strategy to make a joint decision during the optimization for classification. Specifically, we utilize different deep neural networks to learn multiple view-specific representations, and model deep interactive information through a shared interactive network using the cross-correlations between attributes of these representations. After that, we adaptively integrate multiple neural networks by flexibly tuning the power exponent of weight, which not only avoids the trivial solution of weight but also provides a new approach to fuse outputs from different deterministic neural networks. Extensive experiments on several public datasets demonstrate the rationality and effectiveness of our method.

Introduction

In recent years, multi-view classification has been a fundamental topic of computer vision community (Su et al. 2009; Sun et al. 2009; Ozay, Walas, and Leonardis 2014; Farfade, Saberian, and Li 2015; Qi et al. 2016; Kanezaki, Matsushita, and Nishida 2018). The ability of multi-view classification can be used for object recognition, scene interpretation, and visual search, which is crucial for an intelligent visual recognition system (Savarese and Li 2010).

It is worth mentioning that 'multi-view' in this paper means multiple distinct representations of the object and comprehensively describes all the information of the object. In practical applications, many objects have a set of diverse and complementary representations in the form of multiple views. The most common example is that images can be described by heterogeneous visual descriptors. These extracted visual feature sets may show tremendous diversity and complementarity in heterogeneous feature spaces. Therefore, many efforts have been devoted to combining various features from different views to help the classification tasks.

A straightforward way (Dong et al. 2013; Simonyan et al. 2013) is to concatenate all the views into a single view and then treat the task as a single view task. This seems unreasonable in some applications since the concatenation of different views either causes very high dimensional feature vectors or neglects the inter-view discriminant information (Kan et al. 2016). Moreover, it might be suboptimal to treat the important view and the less important view equally (Cai, Nie, and Huang 2013).

Several prior algorithms (Hotelling 1936; Akaho 2006; Hardoon et al. 2007; Blaschko and Lampert 2008; Rupnik and Shawe-Taylor 2010) focus on constructing the linear or non-linear transformations projected into a new space and encourage different projected feature sets to be as correlative as possible. Typically, Canonical Correlation Analysis (CCA) (Hotelling 1936), attempts to learn two transformations which project two views into a common space by maximizing their cross-correlation. Kernel Canonical Correlation Analysis (KCCA) (Akaho 2006) finds maximally correlated nonlinear projections limited to the Reproducing Kernel Hilbert Spaces. With the increased number of views, Multi-view CCA (MCCA) (Rupnik and Shawe-Taylor 2010) is proposed to obtain multiple transformations by maximizing the total correlations of all the pairwise views. However, the above works are unsupervised, which may lead to the obtained transformations adverse for classification.

To further take label information into account, as a supervised extension of CCA, Generalized Multi-view Analysis (GMA) (Sharma et al. 2012) learns a common discriminative subspace to solve the problem of cross-view classification. As a combination of CCA and Uncorrelated Linear Discriminant Analysis (ULDA), Multi-view Uncorrelated Linear Discriminant Analysis (MULDA) (Yang and Sun 2014) extracts mutual uncorrelated features for different views and computes their projected transformations in a common subspace. Besides, Multi-view Discriminant Anal-

*Corresponding author, † Equal contribution

ysis (MvDA) (Kan et al. 2016) tries to find a discriminant common space by learning different view-specific linear transformations jointly. DeepLDA (Dorfer, Kelz, and Widmer 2016) utilizes an end-to-end manner to learn linearly separable latent representations by maximizing the eigenvalues of the general LDA eigenvalue problem. However, these LDA-based methods cannot capture some subtle but important structures in some challenging scenarios.

Recently, most of the researches (Ngiam et al. 2011; Srivastava and Salakhutdinov 2012; Andrew et al. 2013; Wang et al. 2015; Kan, Shan, and Chen 2016) employ more flexible deep neural networks to learn nonlinear representations and achieve improved performance. Because deep learning methods can help multi-view methods to extract multi-view representations, relationships, and complementarity information. Specifically, Multimodal Deep Autoencoder (Ngiam et al. 2011) and Multimodal Deep Boltzmann Machine (Srivastava and Salakhutdinov 2012) jointly learn a shared representation/distribution of two views. Deep CCA (Andrew et al. 2013) learns the complex nonlinear transformations to make the resulting representations highly linearly correlated. Deep CCA Autoencoder (DCCAE) (Wang et al. 2015) adds two autoencoders based on DCCA, where the canonical correlation of their reconstruction errors and learned representations are combined to be optimized. However, both DCCA and DCCAE are limited to the two-view inputs with the same dimensionality. Multi-view Deep Network (Kan, Shan, and Chen 2016) can be seen as a nonlinear version of MvDA, which uses neural networks as the mapping functions instead of linear transformations. Both of these two methods require the within-scatter matrix to be nonsingular during optimizing their objective functions. Probabilistic Multi-view Graph Embedding (Okuno, Hada, and Shimodaira 2018) is a probabilistic model for predicting new associations between two data vectors, which is modeled by using the inner product of feature vectors. This is quite different from the cross-correlations of attributes between feature vectors in our method. Besides, (Lin, RoyChowdhury, and Maji ; Yang et al. 2016) based on two convolutional neural networks utilize the inner product (or matrix multiplication) as a pooling operation for a single image, whereas our method is based on multiple neural networks with inputting various features extracted from different views.

Although many existing works demonstrate promising performance in various computer vision tasks, we observe that there is not a unified framework can simultaneously consider multiple view-specific information, deep interactive information between views, and a reliable fusion strategy of various views. To deal with these issues, we design a suitable model termed as MvNNcor. In our proposed MvNNcor, (1) the multiple view-specific information, coming from different faceted representations of the data instance, ensures the *diversity* and *complementarity* among different views to enhance multi-view learning. (2) The deep interactive information is defined as the interactive information passed by a deep and shared interactive subnetwork, where the interactive information is generated by the cross-correlations of attributes between different view-specific representations. It explicitly models the relationship between views, which is an important aspect of multi-view learning and a key to improving performance. (3) The multi-view loss fusion strategy is to calculate multiple losses for multiple views and to fuse them in an adaptive weighted way, where the weight is learned and adjusted flexibly during training. This not only avoids the trivial solution of weight but also provides a new way of fusing outputs of different deterministic neural networks. Note that our multi-view fusion strategy is a loss fusion, not image or feature fusion in existing methods.

The main contributions of this paper are as follows.

- We propose a unified framework, which seamlessly embeds various view-specific information, deep interactive information, and a novel multi-view loss fusion strategy to make a joint decision during the optimization to improve the classification performance.
- We model the cross-correlations between attributes of different view-specific information and learn deep interactive information from all cross-correlations of each view through an interactive network. We further make deep interactive information incorporated with the view-specific information in a proper proportion and calculate the corresponding loss of each view, and then fuse them in an adaptive weighted way.
- We perform extensive experiments on several public datasets to prove the rationality and effectiveness of our model. Furthermore, we demonstrate the power of multi-view learning with the CNN feature representations, which provides a novel idea of fusing outputs of any deterministic neural networks in further work.

Related Work

In this section, we briefly review several works close to our proposed method. Given a data matrix $\boldsymbol{X}^v$ for the v-th view $(v = 1, \cdots, M)$, $f_v(\boldsymbol{X}^v)$ denotes linear transformation or feature extraction network performed on $\boldsymbol{X}^v$.

MvDA (Kan et al. 2016) seeks for a discriminant common space by M linear transformations $f_v|_{v=1}^M$ which are optimized using a generalized Rayleigh quotient. DeepLDA (Dorfer, Kelz, and Widmer 2016) proposes an objective function that pushes the network to produce feature distributions having low intraclass variance and high interclass variance. DeepLDA is derived from a general LDA eigenvalue problem, which needs to be trained by Stochastic Gradient Descent and back-propagation with a large batchsize to get stable covariance estimates.

DCCA (Andrew et al. 2013) aims to jointly learn both f_1 and f_2 networks such that the canonical correlation of $f_1(\boldsymbol{X}^1)$ and $f_2(\boldsymbol{X}^2)$ is as high as possible. DCCAE (Wang et al. 2015) consists of two autoencoders $g_1(f_1(\boldsymbol{X}^1))$ and $g_2(f_2(\boldsymbol{X}^2))$ (where g_1 and g_2 are the reconstruction networks for each view), and optimizes the combination of the canonical correlation of $f_1(\boldsymbol{X}^1)$ and $f_2(\boldsymbol{X}^2)$ and the reconstruction errors of autoencoders. GradKCCA is a kernel-based non-linear CCA that exploits the gradients of the preimages of the projection directions to maximize the canonical correlation in the kernel-induced feature spaces.

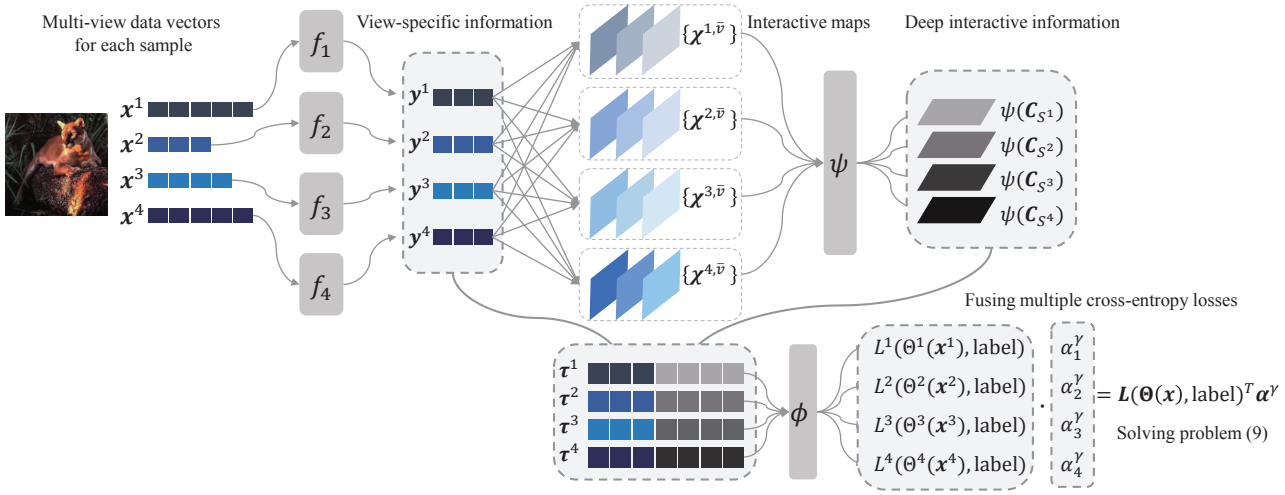

Figure 1: The architecture of our proposed MvNNcor. It is a unified and embedded framework where various view-specific information ensures the diversity and complementarity, deep interactive information explores the relationships between views, and a novel multi-view loss fusion strategy realizes a joint decision of multiple views in an adaptive weighted way.

The Proposed Method

In this section, we propose a novel multi-view neural networks framework, seamlessly embedding various view-specific information and deep interactive information, and introducing a novel multi-view fusion strategy, to make a joint decision during the optimization for classification. Intuitively, the architecture is shown in Figure 1.

Various View-specific Information

For each instance, we collect various visual feature vectors $\{\boldsymbol{x}^v\}_{v=1}^M$ from M views to ensure the diversity and complementarity of multi-view information. Defining a set of neural networks $\{f_v\}_{v=1}^M$ where f_v captures the high-level view-specific information for the v-th view and transforms $\boldsymbol{x}^v$ from $\mathbb{R}^{d_v}$ into $\mathbb{R}^d$, that is,

$$\boldsymbol{y}^v = f_v(\boldsymbol{x}^v), \tag{1}$$

where $\boldsymbol{y}^v \in \mathbb{R}^d$ and f_v is a neural network with L layers,

$$\boldsymbol{h}_{f_v}^l = \boldsymbol{\sigma}(\boldsymbol{W}_{f_v}^l \boldsymbol{h}_{f_v}^{l-1} + \boldsymbol{b}_{f_v}^l). \tag{2}$$

For the l-th layer ($l = 1, \cdots, L$), $\boldsymbol{W}_{f_v}^l \in \mathbb{R}^{m_l \times m_{l-1}}$ is the weight matrix ($m_0 = d_v, m_L = d$) and $\boldsymbol{b}_{f_v}^l \in \mathbb{R}^{m_l}$ denotes the bias vector, and $\boldsymbol{h}_{f_v}^l \in \mathbb{R}^{m_l}$ is the output ($\boldsymbol{h}_{f_v}^0 = \boldsymbol{x}^v$ and $\boldsymbol{h}_{f_v}^L = \boldsymbol{y}^v$) and $\boldsymbol{\sigma}$ is the activation function applied component-wise. $\{f_v\}_{v=1}^M$ are trained simultaneously by minimizing the final loss, where parameters of each f_v are learned in parallel and independently.

Multiple Pairwise Deep Interactive Information

We propose deep interactive information which is defined as the interactive information passed by a deep and shared interactive sub-network. The interactive information is generated by the cross-correlations of attributes between different view-specific representations, which explicitly models the relationship between views and explores the potential consistent information.

For the v-th view, we define a set S^v which contains different view pairs with respect to the v-th view, that is,

$$S^v = \{(v, \bar{v})\}_{\bar{v}=\{1,\cdots,M\}\setminus v}, \tag{3}$$

where $v \in \{1, \cdots, M\}$ and $(v, \bar{v}) = (\bar{v}, v)$ means undirected. We calculate the cross-correlation matrix between $\boldsymbol{y}^v$ and $\boldsymbol{y}^{\bar{v}}$, which results in a two-dimensional interactive map $\boldsymbol{\chi}^{v,\bar{v}} \in \mathbb{R}^{d \times d}$,

$$\boldsymbol{\chi}^{v,\bar{v}} = \mathbb{E}[\boldsymbol{y}^v \boldsymbol{y}^{\bar{v}\top}], \tag{4}$$

whose (i, j)-th entry is $\mathbb{E}[y_i^v y_j^{\bar{v}}]$, the expectation of $y_i^v y_j^{\bar{v}}$. Extending to all the view pairs, the interactive maps of the v-th view with respect to other $M-1$ views can be collected into a set $\boldsymbol{C}_{S^v} = \{\boldsymbol{\chi}^{v,\bar{v}}\}_{(v,\bar{v})=S^v}$.

Based on the above interactive map of each pairwise views, we further introduce an interactive network ψ to project each $\boldsymbol{\chi}^{v,\bar{v}}$ of $\boldsymbol{C}_{S^v}$ into an embedded space $\mathbb{R}^d$, which learns the deep interactive information and makes it incorporate with $\boldsymbol{y}^v$ in a proper proportion. That is,

$$\psi(\boldsymbol{\chi}^{v,\bar{v}}) = \boldsymbol{\sigma}(\boldsymbol{W}_\psi \mathrm{vec}(\boldsymbol{\chi}^{v,\bar{v}}) + \boldsymbol{b}_\psi), \tag{5}$$

where $\boldsymbol{W}_\psi \in \mathbb{R}^{d \times d^2}$, $\boldsymbol{b}_\psi \in \mathbb{R}^d$, and vec() denotes the vectorization of a matrix. For all the $M-1$ view pairs, the deep interactive information of the v-th view can be described as,

$$\boldsymbol{\psi}(\boldsymbol{C}_{S^v}) = \{\psi(\boldsymbol{\chi}^{v,\bar{v}})\}_{(v,\bar{v})=S^v}, \tag{6}$$

where $\boldsymbol{\psi}$ denotes applying ψ on each entry of $\boldsymbol{C}_{S^v}$, $\boldsymbol{\psi}(\boldsymbol{C}_{S^v}) \in \mathbb{R}^{d \times (M-1)}$, and parameters of ψ are learned and shared across M sets $\{\boldsymbol{C}_{S^v}\}|_{v=1}^M$.

Integrating Different Kinds of Information

Based on the above two subsections, we combine the view-specific information $\boldsymbol{y}^v$ and the corresponding deep interactive information $\boldsymbol{\psi}(\boldsymbol{C}_{S^v})$ as follows,

$$\boldsymbol{\tau}^v = \begin{bmatrix} \boldsymbol{y}^v \\ \mathrm{vec}(\boldsymbol{\psi}(\boldsymbol{C}_{S^v})) \end{bmatrix}, \tag{7}$$

where $\boldsymbol{\tau}^v \in \mathbb{R}^{dM}$ is fed into a neural network with two layers, i.e., $\phi = \{\boldsymbol{W}_\phi^l, \boldsymbol{b}_\phi^l\}_{l=1,2}$, that is,

$$\boldsymbol{z}^v = \phi(\boldsymbol{\tau}^v) = \boldsymbol{W}_\phi^2 \boldsymbol{\sigma}(\boldsymbol{W}_\phi^1 \boldsymbol{\tau}^v + \boldsymbol{b}_\phi^1) + \boldsymbol{b}_\phi^2, \tag{8}$$

where $\boldsymbol{W}_\phi^2 \in \mathbb{R}^{C \times d_h}$ and $\boldsymbol{W}_\phi^1 \in \mathbb{R}^{d_h \times dM}$ are the weight matrices, $\boldsymbol{b}_\phi^2 \in \mathbb{R}^C$ and $\boldsymbol{b}_\phi^1 \in \mathbb{R}^{d_h}$ are the bias vectors, and $\boldsymbol{z}^v \in \mathbb{R}^C$ produces a distribution over the possible classes. d_h denotes the number of hidden units in ϕ. C is the number of categories. It is obvious that $\boldsymbol{\tau}^v$ is passed through ϕ to obtain the predictions $\boldsymbol{z}^v$.

Multi-view Loss Fusion Strategy

In this subsection, we provide an adaptive-weighting loss fusion strategy for multiple neural networks to make a joint decision and implement multi-view classification, which can be described as,

$$\min_{\boldsymbol{\alpha}} \sum_{v=1}^{M} \alpha_v^\gamma L^v(\Theta^v(\boldsymbol{x}^v), \text{label}) \quad s.t.\ \boldsymbol{\alpha}^\top \mathbf{1} = \mathbf{1}, \boldsymbol{\alpha} \geq \mathbf{0}, \tag{9}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^M$, $\{f_v, \psi, \phi\}$ is simply denoted as Θ^v and $\Theta^v(\boldsymbol{x}^v) = \boldsymbol{z}^v$. $\gamma > 1$ is the power exponent parameter of the weight α_v of the v-th view, which adjusts the weight distribution of different views flexibly and avoids the trivial solution of $\boldsymbol{\alpha}$ during the classification. The cross-entropy loss of the v-th view $L^v(\boldsymbol{z}^v, \text{label})$ is defined as,

$$L^v(\boldsymbol{z}^v, \text{label}) = -\log\left(\frac{\exp(z^v_{\text{label}})}{\sum_{o=1}^{C} \exp(z_o^v)}\right), \tag{10}$$

where label means the ground truth. It can be seen that the prediction of multi-view classification is obtained through updating and optimizing the problem (9).

Optimization

We alternately optimize network parameters $\{f_v, \psi, \phi\}$ and weight α_v of each view, respectively.

Update neural networks $\{f_v, \psi, \phi\}$

We update $\{f_v, \psi, \phi\}$ by fixing the weight vector $\boldsymbol{\alpha}$ and utilizing the autograd package in PyTorch and suppose that f_v and ϕ are two-layer neural networks and ψ is a one-layer neural network.

Update weights $\boldsymbol{\alpha}$ of multiple views

Through learning α_v for each view, our method assigns a higher weight to a more discriminative view. We fix the parameters $\{f_v, \psi, \phi\}$ and update α_v by solving the following constrained optimization problem, that is,

$$\min_{\boldsymbol{\alpha}} \sum_{v=1}^{M} \alpha_v^\gamma L_v(\boldsymbol{z}^v, \text{label}) \quad s.t.\ \boldsymbol{\alpha}^\top \mathbf{1} = \mathbf{1}, \boldsymbol{\alpha} \geq \mathbf{0}. \tag{11}$$

The corresponding Lagrangian function is,

$$\mathfrak{L}(\boldsymbol{\alpha}, \xi) = \sum_{v=1}^{M} \alpha_v^\gamma L_v(\boldsymbol{z}^v, \text{label}) - \xi\left(\sum_{v=1}^{M} \alpha_v - 1\right), \tag{12}$$

where ξ is the Lagrange multiplier. Getting the derivatives of Eq. (12) with respect to α_v and ξ, and then setting them to zero, we can obtain the updating equation of α_v,

$$\alpha_v = \frac{L_v^{\frac{1}{1-\gamma}}}{\sum_{m=1}^{M} L_m^{\frac{1}{1-\gamma}}}, \tag{13}$$

where $\gamma > 1$ denotes the power exponent parameter.

Experiments

In this section, to make the experiments closer to a real-life setting, we evaluate the performance of our MvNNcor on several public datasets.

Datasets

Since many state-of-the-art multi-view methods in the literature use multiple pre-extracted features to make evaluations, we have to follow their settings to make the experimental comparison fair. We utilize the pre-extracted feature vectors for all the images of the dataset. Different kinds of pre-extracted feature vectors show different aspects of the images (e.g., color, shape). They are not directly visible but related to the visual information, and are listed as follows.

Caltech101/20. The dataset (Fei-Fei, Fergus, and Perona 2007) consists of 101 categories of images. Following the work (Li et al. 2015), we select the widely used 2386 images of 20 classes and 9144 images of 102 classes (101 object categories and an additional background class), respectively, denoted as Caltech20 and Caltech101. There are 6 kinds of features to be extracted from all the images, i.e., 48-D (D: dimensions) Gabor, 40-D Wavelet moments, 254-D CENTRIST, 1984-D HOG, 512-D GIST, and 928-D LBP.

AWA. This dataset (Lampert, Nickisch, and Harmeling 2009) contains 30475 images of 50 animals classes with 6 kinds of pre-extracted features for each image. They include 2688-D Color Histogram, 2000-D Local Self-Similarity, 252-D Pyramid HOG, 2000-D SIFT, 2000-D color SIFT, and 2000-D SURF.

NUSOBJ. This is a subset of NUS-WIDE (Chua et al. 2009) and contains 31 object categories and 30,000 images in total. It has 5 types of low-level features extracted from all the images, including 64-D color histogram, 225-D block-wise color moments, 144-D color correlogram, 73-D edge direction histogram, and 128-D wavelet texture.

Reuters. It (Amini, Usunier, and Goutte 2009) is a document dataset and contains 18758 documents that are written in 5 different languages. All the documents are categorized into 6 classes. Different languages can be seen as different views, that is, English (21531-D), French (24892-D), German (34251-D), Italian (15506-D) and Spanish (11547-D).

Hand. This dataset (Dheeru and Karra Taniskidou 2017) consists of features of handwritten numerals ('0'$\sim$'9') extracted from a collection of Dutch utility maps, 200 patterns per class (a total of 2000 patterns). These digits are represented in terms of 6 feature sets, containing 76-D Fou, 216-D Fac, 64-D Kar, 240 Pix, 47-D Zer, and 6-D Mor.

Table 1: The performance of MvNNcor with respect to different values of batch size on all the datasets.

batch	Caltech101	Caltech20	AWA	NUSOBJ	Reuters	Hand
32	74.25	94.64	46.84	51.36	88.85	99.48
64	76.00	97.92	47.69	52.05	89.28	99.48
128	75.11	96.88	46.35	50.95	89.06	99.22
256	75.13	/	46.81	50.76	88.95	/

Referring to (Andrew et al. 2013; Wang et al. 2015), we split each dataset into three parts: 70% samples for training, two-thirds of the rest samples for validation, and one-third of that for testing. We utilize the classification accuracy to evaluate the performance of all the methods and report the final results in Tables 1∼5.

Experimental Settings

Comparison Methods. We compare our MvNNCorr method with several state-of-the-art methods for multi-view classification, including DCCA (Andrew et al. 2013), DCCAE (Wang et al. 2015), DeepLDA (Dorfer, Kelz, and Widmer 2016), MvDA (Kan et al. 2016), GradKCCA (Uurtio, Bhadra, and Rousu 2019) and SVMcon reported in Table 3. Specifically, DCCA, DCCAE, and GradKCCA are limited to two views with the same dimensionality. MvDA requires that the dimensionalities of different views keep consistent with each other. DeepLDA puts LDA on top of a deep neural network and concatenates different views as input. As a baseline, SVMcon is to concatenate all the views as input and directly feeds it into an SVM classifier.

Furthermore, in Table 4, we make comparisons between MvNNcorr and its variations, i.e., mvNN, MvNN, and MvNNw. Concretely, mvNN concatenates different views as input and feeds it into a multi-layer perceptron. MvNN learns M view-specific networks for M views and integrates them in a concatenated way, which is the feature level fusion. MvNNw captures M view-specific networks from M views and combines them in an adaptive weighted way, which is implemented by minimizing the sum of M losses and belongs to the loss fusion. Our MvNNcor method shows the advantages of integrating the view-specific information and deep interactive information between views, along with a loss fusion strategy.

Parameter Setup. For Deep LDA, the architecture is a regular fully-connected neural network with three hidden layers, including 400, 200, and 300 units, equipped with ReLU activation function. For DCCAE, each feature extraction network is a fully connected network with three hidden layers, including 400, 200, and 300 units, with ReLU activation function, followed by a linear output of C units; the reconstruction network has the same settings of hidden layers to the feature extraction network and a linear output of d_v units. The capacities of these networks are the same as those of their counterparts in DCCA.

For our MvNNcor, there are three kinds of networks $\{\{f_v\}_{v=1}^M, \psi, \phi\}$ needed to learn. Each of f_v is a fully-connected network which consists of d_v input units and two hidden layers with 400 and 200 units equipped with ReLU activation function. ψ consists of 200^2 input units and 200 hidden units with ReLU activation function. ϕ consists of $200\times M$ input units and 300 hidden units with ReLU activation function, followed by a linear output layer with C units. That is, the input of ψ is the outer product of the output of f_v with vectorization and the input of ϕ is concatenated by the outputs of f_v and ψ. The capacities of the above networks are the same as those of their counterparts in mvNN, MvNN, and MvNNw.

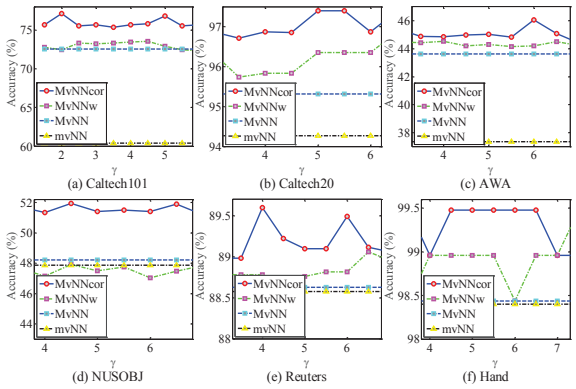

Figure 2: The performance of MvNNcor and its variants with respect to different values of γ on all the datasets.

Table 2: The performance of MvNNcor with respect to different numbers of ψ hidden units on all the datasets.

ψ	Caltech101	Caltech20	AWA	NUSOBJ	Reuters	Hand
50	73.55	96.35	45.87	51.46	88.63	98.96
100	74.55	96.83	46.81	51.49	88.42	99.48
200	76.00	97.92	47.69	52.05	89.28	99.48
400	75.22	95.31	47.56	51.73	88.69	99.48
800	75.45	96.35	47.69	51.83	88.69	98.96

All the networks in this paper are trained by Adam with batch normalization, where the learning rate is 10^{-3}, $\beta_1 = 0.5$, $\beta_2 = 0.9$. In addition, we study the impact of batch size on the classification performance of our MvNNcor by setting batch size like 32, 64, 128, and 256 respectively. According to Table 1, it is obvious that batch size being 64 can achieve the best performance.

Furthermore, we vary γ from 1.5 to 10 with a step of 0.5 to explore the influence of different values of γ on classification accuracy. Based on the optimal γ, we can train the optimal model to achieve the best performance, and then we evaluate the optimal model on testing set. The results are shown in Figure 2 where Caltech101/20, AWA, NUSOBJ, Reuters, and Hand datasets can achieve the best performance when γ is set as 2.0, 5.0, 6.0, 4.5, 4.0, and 6.5, respectively.

Besides, to explore the combination proportion of view-specific information and deep interactive information, we change the number of ψ hidden units, i.e., 50, 100, 200, 400, and 800, and show the results in Table 2. It can be seen that the classification accuracy is the highest when the number of

Table 3: Comparison results of MvNNcor and several state-of-the-art methods on all the datasets.

Method	Caltech101	Caltech20	AWA	NUSOBJ	Reuters	Hand
SVMcon	47.90±0.78	83.83±0.73	31.04±0.56	42.72±0.53	88.18±0.64	97.67±0.67
DeepLDA	45.65±0.50	76.51±0.45	25.60±0.49	20.32±0.97	84.91±0.49	97.67±0.50
MvDA	45.20±0.00	76.28±0.00	9.79±0.00	11.46±0.00	78.83±0.00	21.33±0.00
DCCA	66.18±0.92	86.50±0.89	20.68±0.91	28.75±0.55	64.92±2.33	91.60±0.90
DCCAE	26.89±0.00	50.27±0.00	13.48±0.00	27.48±0.00	56.53±0.00	80.00±0.00
GradKCCA	50.53±0.53	92.92±0.83	33.33±0.00	48.15±1.85	43.39±0.53	95.74±1.07
MvNNcor	76.00±0.34	97.92±0.52	47.69±0.03	52.05±0.32	89.28±0.10	99.48±0.00

Table 4: Comparison results of MvNNcor and its variations on all the datasets.

Method	Caltech101	Caltech20	AWA	NUSOBJ	Reuters	Hand
mvNN (ϕ)	60.04±0.45	94.79±0.52	37.34±0.00	48.85±0.06	88.58±0.05	98.70±0.26
MvNN ($\{f_v\}_{v=1}^M + \phi$)	73.24±0.31	96.01±0.78	43.62±0.20	50.37±0.03	88.77±0.03	98.96±0.52
MvNNw ($\{f_v\}_{v=1}^M$+ϕ+$\boldsymbol{\alpha}$)	73.75±0.36	96.62±0.78	45.44±0.05	48.06±0.05	89.17±0.12	99.22±0.26
MvNNcor ($\{f_v\}_{v=1}^M$+ψ+ϕ+$\boldsymbol{\alpha}$)	76.00±0.34	97.92±0.52	47.69±0.03	52.05±0.32	89.28±0.10	99.48±0.00

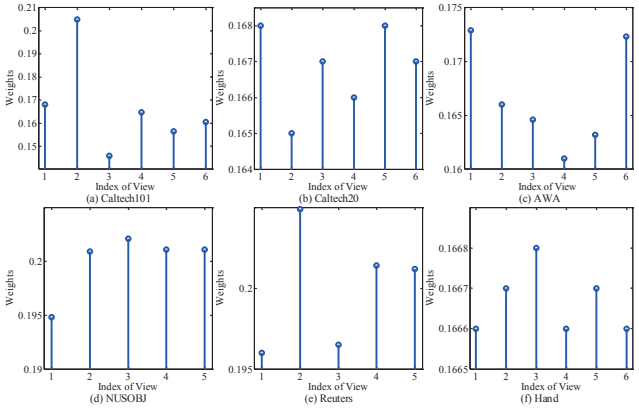

Figure 3: Learned weights of different views by MvNNcor on all the datasets.

ψ hidden units is 200 and the combination proportion also has an impact on the classification performance.

Experimental Results

Tables 3 and 4 show the performance of all compared methods. Firstly, compared with SVMcon and DeepLDA, our MvNNcor consistently outperforms them on all the datasets. For example, in Caltech101 dataset, MvNNcor achieves 28.10% and 30.35% improvements, respectively. Because the concatenation of all the views may confuse the view-specific information and miss the interactive information during the multi-view classification. And we make a comparison between MvNNcor and MvDA, where the performance of MvNNcor is better than that of MvDA on all the datasets. For instance, MvNNcor obtains 30.80% improvements on the Caltech101 dataset compared with MvDA. Because the linear transformations of MvDA cannot deal well with some subtle but important structures in some challenging scenarios and limit the performance of classification.

Secondly, we compare our MvNNcor with three CCA-based methods: DCCA, DCCAE, and GradKCCA. Table 3 shows that MvNNcor performs better than the above three methods. For instance, on AWA dataset, compared with DCCA, DCCAE, and GradKCCA, MvNNcor achieves 27.01%, 34.21%, and 14.36% improvements, respectively. Although DCCA and DCCAE are based on the deep neural networks, their CCA-based frameworks are limited to two views with the same dimensionality, which cannot capture more diverse and complementary information from more views.

Finally, compared with mvNN, MvNN, and MvNNw, our MvNNcorr achieves 3.20%, 1.68%, and 3.99% improvements, respectively, on NUSOBJ dataset. These results successively demonstrate the effectiveness of integrating the view-specific information and deep interactive information between views as well as the superiority of the adaptive weighted loss strategy.

Moreover, Figure 3 provides the learned weights of different views on each dataset, where the x-axis denotes the index of different views and the y-axis corresponds to the weight of each view. The higher weight indicates that the view provides more valuable information and makes more contributions. Figure 4 visualizes the embedding feature spaces learned by our MvNNcor and other state-of-the-art methods using t-SNE, which intuitively demonstrates that the classes in terms of the ground-truth labels with MvNNcor are more compact and separable. Because the number of categories on Caltech101, AWA, and NUSOBJ datasets are too large to be clearly plotted within a small space, we skip them here.

Discussion

It is worth mentioning that our MvNNcor is a general approach which can improve not only handcrafted features (such as HOG, LBP, or SURF) but also deep-learned features to perform the image classification.

We apply several popular deep networks, including AlexNet (Krizhevsky, Sutskever, and Hinton 2012),

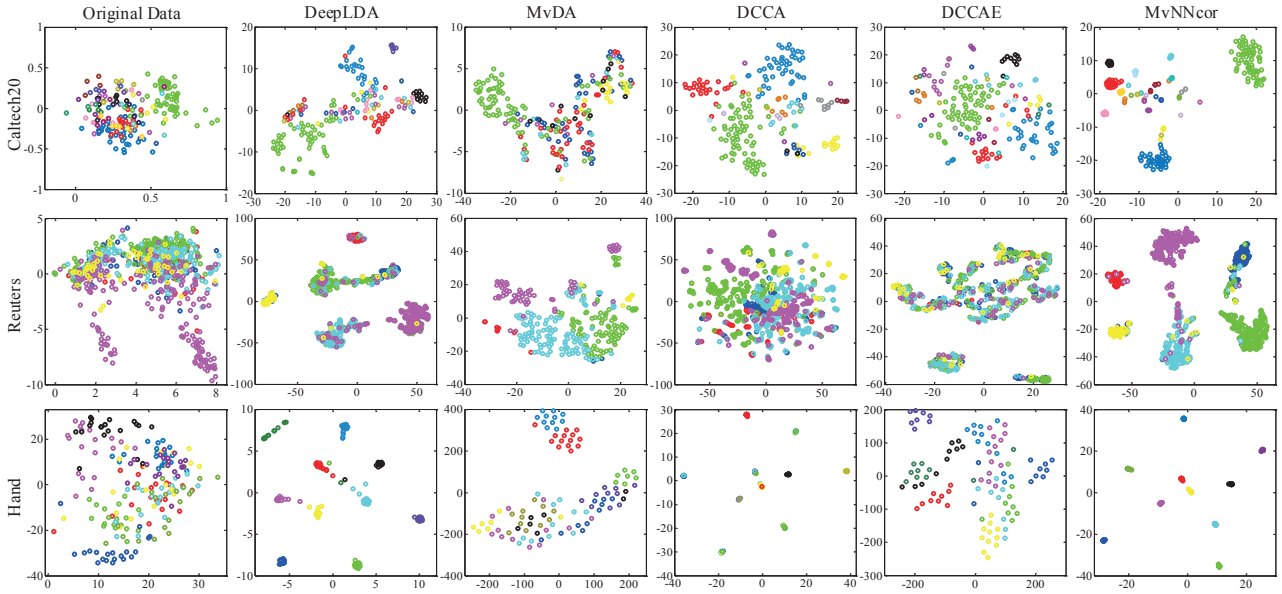

Figure 4: Visualizations of MvNNcor and other state-of-the-art methods using t-SNE on Caltech20, Reuters, and Hand datasets, respectively. The first column is the original data and other columns are the outputs of several compared methods.

Table 5: Comparison results of our MvNNcor and several deep convolutional neural network architectures on image datasets Caltech101 and NUSOBJ.

Methods	Caltech101		NUSOBJ	
	transferred	fine-tuned	transferred	fine-tuned
AlexNet	85.86	87.83	59.24	58.98
GoogLeNet	88.05	89.69	63.62	64.99
ResNet-101	90.24	92.43	69.69	70.26
VGGNet-16	86.18	90.24	64.72	67.57
MvNNcor	93.53	98.10	70.49	71.28

GoogLeNet (Szegedy et al. 2015), VGGNet-16 (Simonyan and Zisserman 2015), and ResNet-101 (He et al. 2016), on Caltech101 and NUSOBJ datasets, respectively. Split all the images of each dataset into training, validation, and testing sets referring to the experimental setting of MvNNcor.

To demonstrate the power of the multi-view learning on the CNN feature representations, we compare our MvNNcor method with the above CNN feature-based methods which contain four transferred and four fine-tuned CNN features. For the transferred CNN feature-based methods, we treat VGGNet-16, ResNet-101, AlexNet, and GoogLeNet as the general feature extractors to obtain the CNN features and then use linear SVMs (C=0.001) for classification. For the fine-tuned CNN feature-based methods, the aforementioned four CNN models are fine-tuned on the training datasets to learn better CNN features and then utilize linear SVMs (C=0.001) for classification. It is worth mentioning that we utilize SVM rather than Softmax for all the CNN features since SVM can obtain better results (Nogueira, Penatti, and dos Santos 2017). For our MvNNcor method, based on four transferred CNN features, we form four views for each image and apply them into MvNNcor to implement multi-view classification; based on four fine-tuned CNN features, we do the same thing. The experimental results are shown in Table 5.

Taking Caltech101 dataset as an example, it can be seen that our MvNNcor method outperforms all the single-view methods (i.e., AlexNet, GoogLeNet, ResNet-101, and VGGNet-16) on both transferred and fine-tuned CNN features, and averagely achieves 8.97%, 6.95%, 4.48%, and 7.61% improvements, respectively. This verifies the superiority of our multi-view learning framework.

Conclusion

In this work, we propose a novel multi-view learning framework named MvNNcor, which seamlessly embeds various view-specific information and deep interaction information, and introduces a new multi-view loss fusion strategy to jointly make decisions and infer categories. Extensive experiments on several public datasets demonstrate the rationality and effectiveness of the proposed MvNNcor method. Furthermore, we demonstrate the power of multi-view learning on the CNN feature representations, which provides a novel idea of fusing outputs of any deterministic neural networks in further work.

Acknowledgments

This work is supported by the Innovation Foundation for Doctor Dissertation of Northwestern Polytechnical University (No. CX201814), State Key Laboratory of Geo-Information Engineering (No. SKLGIE2017-Z-3-2), National NSF of China (Nos. 61432008, 61806092), and Jiangsu Natural Science Foundation (No. BK20180326).

References

Akaho, S. 2006. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*.

Amini, M.; Usunier, N.; and Goutte, C. 2009. Learning from multiple partially observed views-an application to multilingual text categorization. In *NIPS*.

Andrew, G.; Arora, R.; Bilmes, J.; and Livescu, K. 2013. Deep canonical correlation analysis. In *ICML*.

Blaschko, M. B., and Lampert, C. H. 2008. Correlational spectral clustering. In *CVPR*.

Cai, X.; Nie, F.; and Huang, H. 2013. Multi-view k-means clustering on big data. In *IJCAI*.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*.

Dheeru, D., and Karra Taniskidou, E. 2017. UCI machine learning repository.

Dong, C.; Cao, X.; Fang, W.; and Jian, S. 2013. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *CVPR*.

Dorfer, M.; Kelz, R.; and Widmer, G. 2016. Deep linear discriminant analysis. In *ICLR*.

Farfade, S. S.; Saberian, M.; and Li, L. J. 2015. Multi-view face detection using deep convolutional neural networks. In *ICMR*.

Fei-Fei, L.; Fergus, R.; and Perona, P. 2007. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding* 106(1):59–70.

Hardoon, D. R.; Mourao-Miranda, J.; Brammer, M.; and Shawe-Taylor, J. 2007. Unsupervised analysis of fmri data using kernel canonical correlation. *NeuroImage* 37(4):1250–1259.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

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

Kan, M.; Shan, S.; Zhang, H.; Lao, S.; and Chen, X. 2016. Multi-view discriminant analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(1):188–194.

Kan, M.; Shan, S.; and Chen, X. 2016. Multi-view deep network for cross-view classification. In *CVPR*.

Kanezaki, A.; Matsushita, Y.; and Nishida, Y. 2018. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Lampert, C. H.; Nickisch, H.; and Harmeling, S. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *AAAI*.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. Bilinear cnn models for fine-grained visual recognition. In *ICCV*.

Ngiam, J.; Khosla, A.; Kim, M.; Nam, J.; Lee, H.; and Ng, A. Y. 2011. Multimodal deep learning. In *ICML*.

Nogueira, K.; Penatti, O. A.; and dos Santos, J. A. 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition* 61:539–556.

Okuno, A.; Hada, T.; and Shimodaira, H. 2018. A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks. In *ICML*.

Ozay, M.; Walas, K.; and Leonardis, A. 2014. A hierarchical approach for joint multi-view object pose estimation and categorization. In *ICRA*.

Qi, C. R.; Su, H.; Niessner, M.; Dai, A.; Yan, M.; and Guibas, L. J. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*.

Rupnik, J., and Shawe-Taylor, J. 2010. Multi-view canonical correlation analysis. In *SiKDD*.

Savarese, S., and Li, F. F. 2010. *Multi-view Object Categorization and Pose Estimation*.

Sharma, A.; Kumar, A.; Daume, H.; and Jacobs, D. W. 2012. Generalized multiview analysis: A discriminative latent space. In *CVPR*.

Simonyan, K., and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Simonyan, K.; Parkhi, O.; Vedaldi, A.; and Zisserman, A. 2013. Fisher vector faces in the wild.

Srivastava, N., and Salakhutdinov, R. R. 2012. Multimodal learning with deep boltzmann machines. In *NIPS*.

Su, H.; Sun, M.; Li, F. F.; and Savarese, S. 2009. Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories. In *ICCV*.

Sun, M.; Su, H.; Savarese, S.; and Li, F. F. 2009. A multi-view probabilistic model for 3d object classes. In *CVPR*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Uurtio, V.; Bhadra, S.; and Rousu, J. 2019. Large-scale sparse kernel canonical correlation analysis. In *ICML*.

Wang, W.; Arora, R.; Livescu, K.; and Bilmes, J. 2015. On deep multi-view representation learning. In *ICML*.

Yang, M., and Sun, S. 2014. Multi-view uncorrelated linear discriminant analysis with applications to handwritten digit recognition. In *IJCNN*.

Yang, G.; Beijbom, O.; Ning, Z.; and Darrell, T. 2016. Compact bilinear pooling. In *CVPR*.